SYSTEM MCQs COLLECTION

NSCT Prep

Free MCQ Practice for NSCT Test Preparation

Data Structures & Algorithms

1080 Multiple Choice Questions

nsctprep.dev

This dataset is created and compiled by Muhammad Abdullah Awais

Easy Questions

360 questions

Q1. What is a data structure?

A. A specific programming language type
B. A step-by-step problem procedure
C. A physical hardware component unit
D. A way to organize and store data

Answer: D

Q2. What does 'algorithm' mean?

A. A compiled programming language
B. A primitive data type definition
C. A processing hardware component
D. A step-by-step problem procedure

Answer: D

Q3. Which notation is used to describe the upper bound of an algorithm's time complexity?

A. Small-o notation
B. Theta notation
C. Omega notation
D. Big-O notation

Answer: D

Q4. What is the time complexity of accessing an element in an array by index?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q5. Which of the following is a linear data structure?

A. Graph
B. Array
C. Heap
D. Tree

Answer: B

Q6. What is the purpose of pseudocode?

A. To execute low-level machine instructions
B. To debug and test running software
C. To compile a program into bytecode
D. To describe an algorithm in readable form

Answer: D

Q7. Which term refers to the amount of memory an algorithm uses?

A. Time complexity
B. Code complexity
C. Space complexity
D. Cyclomatic complexity

Answer: C

Q8. What is O(1) complexity called?

A. Constant
B. Quadratic
C. Logarithmic
D. Linear

Answer: A

Q9. Which of the following is a non-linear data structure?

A. Linked List
B. Stack
C. Array
D. Tree

Answer: D

Q10. What is an Abstract Data Type (ADT)?

A. A concrete implementation of a data structure
B. A reserved programming language keyword term
C. A specific type of comparison sorting algorithm
D. A mathematical model defined by its behavior

Answer: D

Q11. What is a stack?

A. A LIFO data structure
B. A FIFO data structure
C. A random access data structure
D. A hierarchical data structure

Answer: A

Q12. What is a queue?

A. A sorted data structure
B. A FIFO data structure
C. A LIFO data structure
D. A tree-based data structure

Answer: B

Q13. Which operation adds an element to the top of a stack?

A. Push
B. Dequeue
C. Enqueue
D. Pop

Answer: A

Q14. What is a linked list?

A. Nodes where each node points to the next node
B. A hierarchical tree-based data structure
C. Elements stored in contiguous memory locations
D. A hash-based key-value pair structure

Answer: A

Q15. What is the time complexity of inserting an element at the beginning of a singly linked list?

A. O(n^2)
B. O(log n)
C. O(1)
D. O(n)

Answer: C

Q16. Which data structure is used to evaluate postfix expressions?

A. Array
B. Stack
C. Queue
D. Linked List

Answer: B

Q17. What is the main disadvantage of an array?

A. Random access is not possible
B. Fixed size in static arrays
C. It cannot store integers
D. Elements cannot be sorted

Answer: B

Q18. What does the 'dequeue' operation do?

A. Adds an element to the front of a queue
B. Removes an element from the front of a queue
C. Adds an element to the rear of a queue
D. Removes an element from the rear of a queue

Answer: B

Q19. In a singly linked list, the last node points to:

A. The first node
B. NULL (nothing)
C. The middle node
D. Itself (self)

Answer: B

Q20. Which operation removes an element from the top of a stack?

A. Enqueue
B. Push
C. Peek
D. Pop

Answer: D

Q21. What is a binary tree?

A. A directed graph with no cycles at all
B. A tree with exactly 2 total internal nodes
C. A tree where each node has at most 2 children
D. A tree where each node has at most 3 children

Answer: C

Q22. What is the root of a tree?

A. The node with the most children
B. The topmost node with no parent
C. The node with no children at all
D. The leftmost node in the tree

Answer: B

Q23. What is a leaf node?

A. A node with two children
B. A node with no children
C. The root node itself
D. A node with one child

Answer: B

Q24. What is a Binary Search Tree (BST)?

A. A binary tree where left child > parent > right child
B. A tree data structure with exactly two total nodes
C. A binary tree where left child < parent < right child
D. A balanced tree structure that is always complete

Answer: C

Q25. What is a graph in data structures?

A. A chart used for data visualization
B. A type of contiguous memory array
C. A sequential linear data structure
D. A collection of vertices and edges

Answer: D

Q26. What is the maximum number of nodes in a binary tree of height h?

A. 2^h - 1
B. h^2
C. 2h
D. 2^(h+1) - 1

Answer: D

Q27. What is an undirected graph?

A. A graph where edges have no direction
B. A graph with no edges at all
C. A hierarchical tree structure
D. A graph where edges have direction

Answer: A

Q28. What is the degree of a vertex in an undirected graph?

A. The assigned weight of the given vertex
B. The number of edges incident to the vertex
C. The total number of vertices in the graph
D. The shortest path length from the vertex

Answer: B

Q29. What is a heap?

A. A type of directed graph with cycles
B. A complete binary tree with heap property
C. A sorted array-based storage structure
D. A variant of a hash table with chains

Answer: B

Q30. How is a graph typically represented?

A. Using adjacency matrix or list
B. Using singly linked lists only
C. Using stack data structures
D. Using simple arrays only

Answer: A

Q31. What is linear search?

A. Searching using a hash table for lookups
B. Searching only in a sorted array format
C. Searching each element sequentially from start
D. Searching by dividing the array in half

Answer: C

Q32. What is the time complexity of linear search in the worst case?

A. O(n^2)
B. O(log n)
C. O(n)
D. O(1)

Answer: C

Q33. What is the prerequisite for binary search?

A. The array must be sorted
B. The array must be of fixed size
C. The array must contain unique elements
D. The array must be unsorted

Answer: A

Q34. What is the time complexity of binary search?

A. O(log n)
B. O(n^2)
C. O(n)
D. O(1)

Answer: A

Q35. In binary search, if the target is greater than the middle element, where do you search next?

A. Start over again
B. Middle element only
C. Left half only
D. Right half only

Answer: D

Q36. What is the best-case time complexity of linear search?

A. O(1)
B. O(n)
C. O(log n)
D. O(n^2)

Answer: A

Q37. Which search algorithm does not require the data to be sorted?

A. Interpolation search
B. Linear search
C. Fibonacci search
D. Binary search

Answer: B

Q38. What does binary search return if the element is not found?

A. The middle index of the array
B. The value zero always
C. The last index of the array
D. An indicator such as -1 for not found

Answer: D

Q39. How many comparisons does binary search make for an array of 1024 elements in the worst case?

A. 100
B. 10
C. 1024
D. 512

Answer: B

Q40. What is the space complexity of iterative binary search?

A. O(log n)
B. O(n)
C. O(n^2)
D. O(1)

Answer: D

Q41. What is bubble sort?

A. A sort that divides the array in half each time
B. A sort that selects the minimum element each pass
C. A sort that uses a heap for element ordering
D. A sort that swaps adjacent out-of-order elements

Answer: D

Q42. What is the worst-case time complexity of bubble sort?

A. O(log n)
B. O(n log n)
C. O(n^2)
D. O(n)

Answer: C

Q43. What is selection sort?

A. A sort that works only on linked list nodes
B. A sort that uses binary search for placement
C. A sort that finds minimum and places it correctly
D. A sort that selects a random pivot element

Answer: C

Q44. What is insertion sort?

A. A sort that uses divide and conquer recursion
B. A sort that inserts each element into correct position
C. A sort that only works on integer data types
D. A sort that inserts elements into a heap structure

Answer: B

Q45. Which sorting algorithm is best for nearly sorted data?

A. Merge sort
B. Quick sort
C. Selection sort
D. Insertion sort

Answer: D

Q46. Is bubble sort a stable sorting algorithm?

A. Yes, it is stable
B. Only for integer data
C. Only for string data
D. No, it is unstable

Answer: A

Q47. What is the best-case time complexity of insertion sort?

A. O(n log n)
B. O(n^2)
C. O(n)
D. O(1)

Answer: C

Q48. What is the space complexity of selection sort?

A. O(n^2)
B. O(1)
C. O(n)
D. O(log n)

Answer: B

Q49. How many passes does bubble sort need in the worst case for n elements?

A. n - 1
B. log n
C. n
D. n/2

Answer: A

Q50. Is selection sort a stable sorting algorithm?

A. Only for numeric data
B. No, it is unstable
C. Yes, it is stable
D. Depends on implementation

Answer: B

Q51. What is a hash function?

A. A function that encrypts all data
B. A function that compresses file data
C. A function that sorts data elements
D. A function that maps data to fixed-size value

Answer: D

Q52. What is a hash table?

A. A binary tree data structure
B. A sorted array structure
C. A type of linked list structure
D. A key-value store using hash function

Answer: D

Q53. What is a collision in hashing?

A. When a key is not found in the table
B. When two keys hash to the same index
C. When the hash table is completely full
D. When the hash function fails to compute

Answer: B

Q54. What is the average time complexity of lookup in a hash table?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(1)

Answer: D

Q55. What is chaining in hash tables?

A. Linking multiple hash tables together
B. Connecting hash functions in sequence
C. Storing colliding elements in a linked list
D. A type of hash function computation

Answer: C

Q56. What is the load factor of a hash table?

A. The ratio of entries to number of slots
B. The number of hash functions being used
C. The total number of collisions occurred
D. The total size of the hash table

Answer: A

Q57. Which of the following is a simple hash function for integers?

A. h(k) = k + table_size
B. h(k) = k × table_size
C. h(k) = k / table_size
D. h(k) = k mod table_size

Answer: D

Q58. What is the worst-case time complexity of hash table lookup?

A. O(1)
B. O(n^2)
C. O(log n)
D. O(n)

Answer: D

Q59. What is the purpose of a hash table?

A. To store data in sorted ascending order
B. To compress data for storage saving
C. To sort data elements efficiently
D. To provide fast insert, delete, and lookup

Answer: D

Q60. What happens when you insert a key that already exists in a hash table?

A. The insertion is silently ignored completely
B. Both old and new values are stored
C. The old value is typically updated with new
D. It causes a runtime error exception

Answer: C

Q61. What is in-order traversal of a binary tree?

A. Right, Root, Left
B. Left, Right, Root
C. Left, Root, Right
D. Root, Left, Right

Answer: C

Q62. What is pre-order traversal of a binary tree?

A. Right, Left, Root
B. Root, Left, Right
C. Left, Right, Root
D. Left, Root, Right

Answer: B

Q63. What is post-order traversal of a binary tree?

A. Right, Root, Left
B. Root, Left, Right
C. Left, Root, Right
D. Left, Right, Root

Answer: D

Q64. What data structure is used for level-order (BFS) traversal of a tree?

A. Array
B. Queue
C. Hash table
D. Stack

Answer: B

Q65. What is the height of a single-node tree?

A. 1
B. 0
C. -1
D. 2

Answer: B

Q66. How do you find the minimum element in a BST?

A. Check the root node
B. Traverse to the rightmost node
C. Perform in-order traversal
D. Traverse to the leftmost node

Answer: D

Q67. How do you find the maximum element in a BST?

A. Check the root node
B. Traverse to the leftmost node
C. Traverse to the rightmost node
D. Use binary search method

Answer: C

Q68. What is the time complexity of searching for an element in a BST of height h?

A. O(1)
B. O(n^2)
C. O(h)
D. O(n)

Answer: C

Q69. What is a binary tree's depth?

A. The edge count from root to deepest leaf
B. The total number of nodes in tree
C. The number of internal non-leaf nodes
D. The total number of leaf nodes only

Answer: A

Q70. Which traversal of a BST gives elements in sorted order?

A. Level-order
B. In-order
C. Post-order
D. Pre-order

Answer: B

Q71. What is Breadth-First Search (BFS)?

A. A search that goes as deep as possible first
B. A search only applicable to tree structures
C. A search that uses a stack data structure
D. A search exploring all neighbors at current depth

Answer: D

Q72. What is Depth-First Search (DFS)?

A. A search going deep along each branch first
B. A search only applicable to array structures
C. A search that uses a queue data structure
D. A search that explores all neighbors first

Answer: A

Q73. What data structure does BFS use?

A. Hash table
B. Heap
C. Stack
D. Queue

Answer: D

Q74. What data structure does DFS use?

A. Array structure
B. Stack or recursion
C. Queue structure
D. Heap structure

Answer: B

Q75. What is the time complexity of BFS on a graph with V vertices and E edges?

A. O(V + E)
B. O(E)
C. O(V × E)
D. O(V)

Answer: A

Q76. What is the time complexity of DFS on a graph with V vertices and E edges?

A. O(V + E)
B. O(V × E)
C. O(E^2)
D. O(V^2)

Answer: A

Q77. Can BFS find the shortest path in an unweighted graph?

A. Yes, BFS can do this
B. No, BFS cannot do this
C. Only in tree structures
D. Only in directed graphs

Answer: A

Q78. What is a connected component in an undirected graph?

A. A cycle within the undirected graph
B. A maximal set of vertices all connected by paths
C. An edge connecting two vertices together
D. A single isolated vertex in the graph

Answer: B

Q79. What is a cycle in a graph?

A. An isolated vertex with no edges at all
B. A path starting and ending at the same vertex
C. A straight path between two vertices
D. A tree edge connecting parent to child

Answer: B

Q80. What is a spanning tree of a connected graph?

A. A subgraph with all vertices and all edges
B. A complete subgraph with maximum edges
C. A tree subgraph containing all graph vertices
D. The shortest path tree from one source

Answer: C

Q81. What is the divide and conquer strategy?

A. Breaking into subproblems, solving, and combining
B. Solving all problems using bottom-up iteration
C. Using a greedy approach at each step
D. Solving problems by trying all possibilities

Answer: A

Q82. What is a greedy algorithm?

A. An algorithm making locally optimal choices each step
B. An algorithm that solves all subproblems first
C. An algorithm that uses backtracking for search
D. An algorithm that tries all possible combinations

Answer: A

Q83. Which of the following is an example of divide and conquer?

A. Merge sort
B. Linear search
C. Insertion sort
D. Bubble sort

Answer: A

Q84. What is dynamic programming?

A. A type of greedy algorithm with memoization
B. Solving problems by storing overlapping subproblem results
C. Programming that changes dynamically at runtime
D. An algorithm that uses only recursion for solving

Answer: B

Q85. What is backtracking?

A. Sorting data elements in reverse descending order
B. Trying solutions and abandoning invalid paths early
C. Going back to the start of an algorithm run
D. Reversing the input data before processing

Answer: B

Q86. What is memoization?

A. Writing memos about code documentation
B. A type of sorting with memory optimization
C. Memorizing the algorithm steps by heart
D. Caching results of function calls for reuse

Answer: D

Q87. Which approach builds solutions from smallest subproblems to larger ones?

A. Greedy approach
B. Top-down approach
C. Bottom-up (tabulation)
D. Brute force approach

Answer: C

Q88. What is brute force in algorithm design?

A. An optimized search using heuristics
B. Trying all possible solutions exhaustively
C. The most efficient approach available
D. A divide and conquer optimization

Answer: B

Q89. What two properties must a problem have to be solvable by dynamic programming?

A. Optimal substructure and overlapping subproblems
B. Linear structure and constant time operations
C. Greedy choice property and unique solution
D. Sorting and searching capabilities

Answer: A

Q90. Which algorithm design paradigm does the activity selection problem use?

A. Backtracking search approach
B. Dynamic programming approach
C. Divide and conquer approach
D. Greedy algorithm approach

Answer: D

Q91. What is a priority queue?

A. A type of stack that also supports priority ordering
B. A queue that processes all elements in strict FIFO order
C. A sorted array of elements ordered by their value
D. A structure where higher priority elements are served first

Answer: D

Q92. What data structure typically implements a priority queue?

A. Stack structure
B. Heap structure
C. Array structure
D. Linked list structure

Answer: B

Q93. What is a trie (prefix tree)?

A. A tree for storing and retrieving strings by prefix
B. A graph data structure for string matching
C. A binary search tree for numeric keys
D. A type of hash table for string storage

Answer: A

Q94. What is the time complexity of searching for a word of length L in a trie?

A. O(log n)
B. O(L)
C. O(n)
D. O(n × L)

Answer: B

Q95. What is a disjoint set (Union-Find)?

A. A hash set with unique elements
B. A structure tracking non-overlapping subsets
C. A set with no elements at all
D. A sorted set with ordered elements

Answer: B

Q96. What operation does 'find' perform in Union-Find?

A. Determines which set an element belongs to
B. Searches for an element in the set
C. Finds the minimum element in all sets
D. Finds all elements in a particular set

Answer: A

Q97. What is a min-heap?

A. A heap where minimum element is at root
B. A heap where maximum element is at root
C. A sorted array of elements by value
D. A balanced BST with min at leftmost

Answer: A

Q98. What is the main advantage of a trie over a hash table for string operations?

A. Has a simpler implementation overall
B. Uses less memory usage in practice
C. Supports prefix-based queries efficiently
D. Provides faster exact string lookups

Answer: C

Q99. What does 'union' operation do in Union-Find?

A. Splits a set into two parts
B. Finds common elements between sets
C. Merges two disjoint sets into one
D. Sorts the sets in ascending order

Answer: C

Q100. What is a multiset?

A. A sorted set with unique elements
B. A collection allowing duplicate elements
C. A set with multiple data types
D. A set of sets nested together

Answer: B

Q101. What is string matching?

A. Comparing two strings for equality only
B. Finding occurrences of a pattern in text
C. Sorting strings in alphabetical order
D. Concatenating two strings end to end

Answer: B

Q102. What is the brute force string matching time complexity for text of length n and pattern of length m?

A. O(n)
B. O(n + m)
C. O(m)
D. O(n × m)

Answer: D

Q103. What is a substring?

A. A rearrangement of string characters
B. A contiguous sequence of characters within a string
C. A string of exactly length 1 character
D. Any characters from a string in any order

Answer: B

Q104. What is a subsequence of a string?

A. A sequence derived by deleting characters keeping order
B. A contiguous part of the original string
C. A sorted version of the original string
D. A reversed version of the original string

Answer: A

Q105. What is a palindrome?

A. A string with only repeated characters throughout
B. A string with all unique characters
C. A string with an even number of characters
D. A string that reads the same forward and backward

Answer: D

Q106. What is an anagram?

A. A palindrome of another word
B. A prefix of another given word
C. A substring of another word
D. A rearrangement forming another word

Answer: D

Q107. How can you check if two strings are anagrams?

A. Compare their lengths only for equality
B. Check if they have the same first character
C. Reverse one string and compare with the other
D. Sort both strings and compare, or count frequencies

Answer: D

Q108. What is a prefix of a string?

A. The middle portion of the string
B. A reversed version of the string
C. The last character of the string
D. A substring starting from the beginning

Answer: D

Q109. What is a suffix of a string?

A. A substring starting from the beginning
B. A rotated version of the string
C. The first character of the string
D. A substring ending at the last character

Answer: D

Q110. What is string concatenation?

A. Splitting a string into parts
B. Reversing a string entirely
C. Joining two strings end to end
D. Sorting characters in a string

Answer: C

Q111. What does P stand for in computational complexity?

A. Problems solvable in polynomial time
B. Prime number problems class
C. Probability problems class
D. Parallel problems class

Answer: A

Q112. What does NP stand for in computational complexity?

A. New computational Problems class
B. Non-deterministic Polynomial time
C. Not Polynomial time class
D. No Proof available for it

Answer: B

Q113. What is the best time complexity achievable for comparison-based sorting?

A. O(log n)
B. O(n log n)
C. O(n)
D. O(n^2)

Answer: B

Q114. Which is more efficient: O(n) or O(n log n)?

A. Depends on constants
B. O(n) is more efficient
C. O(n log n) is better
D. They are equivalent

Answer: B

Q115. What is the time complexity of accessing an element in a hash table on average?

A. O(n^2)
B. O(log n)
C. O(1)
D. O(n)

Answer: C

Q116. What is time-space tradeoff?

A. Time complexity cannot affect space complexity
B. Time and space are always equal in measure
C. Using more time to save space or vice versa
D. A method to reduce both time and space simultaneously

Answer: C

Q117. Which is faster: O(log n) or O(" n)?

A. They are equivalent
B. O(log n) is faster
C. O(" n) is faster
D. Cannot compare them

Answer: B

Q118. What is the significance of O(n!) complexity?

A. It is essentially the same as O(n^2) complexity class
B. It is classified as a polynomial time complexity class
C. It is very efficient for all possible input sizes given
D. It is extremely inefficient, growing super-exponentially

Answer: D

Q119. What does 'in-place algorithm' mean?

A. An algorithm that runs at a specific location
B. An algorithm using only constant extra space
C. An algorithm that cannot be relocated in memory
D. An algorithm that only sorts data in place

Answer: B

Q120. What is the time complexity of finding the maximum element in an unsorted array?

A. O(n)
B. O(n^2)
C. O(1)
D. O(log n)

Answer: A

Q121. What does FIFO stand for in data structures?

A. Fast Input Fast Output
B. First Index First Order
C. First In First Out
D. Fixed In Fixed Out

Answer: C

Q122. Which term describes a function calling itself?

A. Abstraction
B. Iteration
C. Recursion
D. Selection

Answer: C

Q123. What is the base case in a recursive function?

A. The largest subproblem solved by recursion
B. The first recursive call made by function
C. The condition that stops the recursion calls
D. The condition that starts the recursion loop

Answer: C

Q124. What does LIFO stand for in data structures?

A. Linear Input Fixed Output
B. Last Index First Output
C. Least Index First Order
D. Last In First Out

Answer: D

Q125. Which complexity class grows the fastest?

A. Linear
B. Logarithmic
C. Exponential
D. Polynomial

Answer: C

Q126. What is the time complexity of a simple for loop iterating n times?

A. O(n)
B. O(1)
C. O(n^2)
D. O(log n)

Answer: A

Q127. What does the term 'traversal' mean in data structures?

A. Copying a structure into another one
B. Sorting elements in ascending order only
C. Deleting all elements from a structure
D. Visiting each element exactly one time

Answer: D

Q128. Which data structure uses the LIFO principle?

A. Stack
B. Queue
C. Graph
D. Array

Answer: A

Q129. What is a primitive data type?

A. A complex object-oriented class type
B. A user-defined composite structure type
C. A dynamically allocated reference type
D. A basic built-in type provided by language

Answer: D

Q130. What is the purpose of a flowchart?

A. To encrypt data for secure transmission
B. To visually represent an algorithm's steps
C. To compile source code into binary files
D. To allocate memory for data structures

Answer: B

Q131. What is the maximum number of elements a stack can pop at once?

A. Half of them
B. All elements
C. Two elements
D. One element

Answer: D

Q132. Which operation adds an element to the rear of a queue?

A. Enqueue
B. Pop
C. Push
D. Dequeue

Answer: A

Q133. What is a singly linked list?

A. Each node contains two separate data field values
B. Each node points to both next and previous nodes
C. Each node points only to the next node in list
D. Each node has a direct link to every other node

Answer: C

Q134. What is stored in each node of a linked list?

A. Data and a pointer to the next node
B. Only the memory address of next node
C. A pointer to both previous and next nodes
D. Only the data value without any pointer

Answer: A

Q135. Which data structure is used for undo operations in editors?

A. Queue
B. Graph
C. Stack
D. Array

Answer: C

Q136. What happens when you dequeue from an empty queue?

A. It inserts a default placeholder element
B. It returns a null or zero value silently
C. It causes an underflow error condition
D. It resets the queue to initial capacity

Answer: C

Q137. What is the index of the first element in a zero-based array?

A. n
B. -1
C. 1
D. 0

Answer: D

Q138. Which linear structure allows insertion at both ends?

A. Deque
B. Array
C. Queue
D. Stack

Answer: A

Q139. What is the time complexity of inserting at the head of a linked list?

A. O(1)
B. O(n^2)
C. O(n)
D. O(log n)

Answer: A

Q140. What is a static array?

A. An array that automatically sorts its elements
B. An array with a fixed size set at creation
C. An array whose size can grow dynamically
D. An array that stores only string data types

Answer: B

Q141. What is the root node of a tree?

A. Any leaf node at the bottom of the tree
B. The node with the most children in tree
C. The node at the deepest level of tree
D. The topmost node with no parent at all

Answer: D

Q142. What is a leaf node in a tree?

A. A node that connects to the root directly
B. A node that has no children at all
C. A node that has exactly two child nodes
D. The first node added to the tree structure

Answer: B

Q143. How many children can a binary tree node have at most?

A. Four
B. Three
C. Two
D. One

Answer: C

Q144. What is an edge in a graph?

A. A connection between two vertices
B. A path visiting all graph vertices
C. An isolated node with no neighbors
D. The weight assigned to each vertex

Answer: A

Q145. What is an undirected graph?

A. A graph where edges have specific direction only
B. A graph where edges have no specific direction
C. A graph where all vertices are fully connected
D. A graph with no edges between any vertices

Answer: B

Q146. What is the height of a single-node tree?

A. One
B. Two
C. Undefined
D. Zero

Answer: D

Q147. What is a weighted graph?

A. A graph where all vertices have the same degree
B. A graph with no cycles between any of its nodes
C. A graph where every node connects to all others
D. A graph where edges have assigned numerical values

Answer: D

Q148. What is the degree of a vertex in an undirected graph?

A. The total weight of all edges from that vertex
B. The number of edges connected to the vertex
C. The shortest path from that vertex to root
D. The number of vertices in the entire graph

Answer: B

Q149. What is a complete binary tree?

A. A tree where every internal node has no children
B. A tree with only leaf nodes and no internal nodes
C. A tree where every node has exactly one child node
D. A tree where all levels are fully filled except last

Answer: D

Q150. What is an adjacency list used for?

A. Finding shortest path in weighted graph directly
B. Storing tree nodes in level-order arrangement
C. Storing graph structure as lists of neighbors
D. Sorting vertices in topological ordering only

Answer: C

Q151. What does linear search do?

A. Uses hashing to find the target element value
B. Sorts elements first then searches for target
C. Checks elements sequentially from start to end
D. Divides the array in half at each comparison

Answer: C

Q152. What is the best-case time complexity of linear search?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(1)

Answer: D

Q153. What is required for binary search to work correctly?

A. The array must be sorted beforehand
B. The array must have an even number of items
C. The array must contain unique elements only
D. The array must be stored in a linked list

Answer: A

Q154. What is the worst-case time complexity of binary search?

A. O(log n)
B. O(n)
C. O(n^2)
D. O(1)

Answer: A

Q155. In binary search, what happens if the target is less than the middle element?

A. Search continues in the right half only
B. The search terminates with element not found
C. Search continues in the left half only
D. Search restarts from the beginning again

Answer: C

Q156. Which search algorithm does not require sorted data?

A. Binary search
B. Fibonacci search
C. Interpolation search
D. Linear search

Answer: D

Q157. What is the space complexity of iterative binary search?

A. O(n^2)
B. O(n)
C. O(log n)
D. O(1)

Answer: D

Q158. What is the worst-case time complexity of linear search?

A. O(n^2)
B. O(n)
C. O(log n)
D. O(1)

Answer: B

Q159. What value does a search algorithm return if the element is not found?

A. Typically negative one or a sentinel value
B. The last index of the array searched
C. The middle index of the array searched
D. The first index of the array searched

Answer: A

Q160. How many comparisons does binary search make on 1024 sorted elements at most?

A. 1024
B. 10
C. 100
D. 512

Answer: B

Q161. What is the basic idea behind bubble sort?

A. Insert each element into its correct sorted position
B. Select the minimum element and place it at start
C. Repeatedly swap adjacent elements if they are wrong
D. Divide array into halves and merge them sorted

Answer: C

Q162. What is the time complexity of bubble sort in the worst case?

A. O(n^2)
B. O(n)
C. O(n log n)
D. O(log n)

Answer: A

Q163. Which sorting algorithm picks the minimum and places it first?

A. Insertion sort
B. Merge sort
C. Selection sort
D. Quick sort

Answer: C

Q164. Is insertion sort a stable sorting algorithm?

A. Yes, it is stable
B. It depends on input
C. Only for integers
D. No, it is never stable

Answer: A

Q165. What is the best-case time complexity of insertion sort?

A. O(log n)
B. O(n^2)
C. O(n log n)
D. O(n)

Answer: D

Q166. What does the term 'in-place sorting' mean?

A. Sorting that requires O(n) extra memory space
B. Sorting performed on an external disk storage device
C. Sorting that creates a new copy of the input array
D. Sorting done without significant extra memory use

Answer: D

Q167. Which sort is most efficient for nearly sorted data?

A. Insertion sort
B. Heap sort
C. Merge sort
D. Selection sort

Answer: A

Q168. What is the worst-case time complexity of selection sort?

A. O(n)
B. O(log n)
C. O(n log n)
D. O(n^2)

Answer: D

Q169. What does 'stable sort' mean?

A. A sort preserving relative order of equal elements
B. A sort with guaranteed O(n log n) time complexity
C. A sort that always completes without crashing at all
D. A sort that works on all data types without error

Answer: A

Q170. How many passes does bubble sort need for n elements?

A. Exactly n passes
B. Exactly one pass
C. At most n-1 passes
D. At most log n passes

Answer: C

Q171. What is a hash function?

A. A function that sorts data into ordered buckets
B. A function that compresses data to save disk space
C. A function that encrypts data for secure storage
D. A function that maps data to a fixed-size value

Answer: D

Q172. What is a hash table?

A. A tree structure organized by hash value ordering
B. A data structure using keys mapped to array indices
C. A linked list with hash-based pointer connections
D. A table that stores elements in sorted order only

Answer: B

Q173. What is a collision in hashing?

A. When a hash table runs out of available memory
B. When two keys map to the same hash table index
C. When the hash function produces a negative value
D. When an element is deleted from the hash table

Answer: B

Q174. What is the average time complexity of hash table lookup?

A. O(log n)
B. O(n^2)
C. O(1)
D. O(n)

Answer: C

Q175. What is chaining in hash tables?

A. Linking hash tables together in a sequence chain
B. Connecting array slots with pointers to next slot
C. Chaining multiple hash functions for better results
D. Storing colliding elements in a linked list at each slot

Answer: D

Q176. What is the load factor of a hash table?

A. The ratio of stored elements to table size
B. The maximum key value stored in the table
C. The number of collisions per hash operation
D. The total memory allocated for the hash table

Answer: A

Q177. What is the division method of hashing?

A. Dividing the key by two repeatedly until zero
B. Splitting the key into digits and summing them up
C. Dividing the table into equal-sized partitions only
D. Computing hash as key modulo the table size value

Answer: D

Q178. What is an ideal property of a good hash function?

A. It should distribute keys uniformly across the table
B. It should always produce the same output value
C. It should map all keys to a single table slot only
D. It should produce values in strictly ascending order

Answer: A

Q179. What is direct addressing in hashing?

A. Using random numbers to determine storage locations
B. Using a linked list to store all elements together
C. Using a tree structure to organize hash table entries
D. Using the key itself as the index into the array

Answer: D

Q180. What data structure does a hash set implement?

A. A collection of unique elements with fast lookup
B. A hierarchical tree of categorized elements
C. An ordered sequence of duplicate elements
D. A sorted list of key-value paired entries

Answer: A

Q181. What is an in-order traversal of a binary tree?

A. Visit root, then left subtree, then right subtree
B. Visit left subtree, then root, then right subtree
C. Visit root, then right subtree, then left subtree
D. Visit left subtree, then right subtree, then root

Answer: B

Q182. What does pre-order traversal visit first?

A. Right child
B. Left child
C. Leaf node
D. Root node

Answer: D

Q183. What does post-order traversal visit last?

A. Right child
B. Leaf node
C. Left child
D. Root node

Answer: D

Q184. What is level-order traversal?

A. Visiting nodes level by level from top to bottom
B. Visiting only nodes at even numbered levels only
C. Visiting nodes from deepest level to root level
D. Visiting leaf nodes first then internal nodes last

Answer: A

Q185. Which data structure is used for level-order traversal?

A. Array
B. Queue
C. Heap
D. Stack

Answer: B

Q186. What is the time complexity of searching in a BST with n nodes?

A. O(n^2) always
B. O(n) in worst case
C. O(log n) always
D. O(1) in worst case

Answer: B

Q187. What is the in-order successor of a node in a BST?

A. The leftmost child of the given node always
B. The parent node of the given node always
C. The rightmost node in the entire BST tree
D. The node with the next larger value in tree

Answer: D

Q188. What does in-order traversal of a BST produce?

A. Elements in random unsorted order
B. Elements in level-wise grouped order
C. Elements in descending sorted order
D. Elements in ascending sorted order

Answer: D

Q189. What is the minimum value node in a BST?

A. The leftmost node in the whole tree
B. The root node of the entire tree
C. The rightmost node in the whole tree
D. Any leaf node in the entire tree

Answer: A

Q190. How many traversal orders are standard for binary trees?

A. Four
B. Three
C. Two
D. Six

Answer: A

Q191. What is Breadth-First Search (BFS)?

A. Randomly visiting nodes until target is found finally
B. Searching only the leftmost path in the graph first
C. Exploring as deep as possible before backtracking
D. Exploring all neighbors at current depth before deeper

Answer: D

Q192. What is Depth-First Search (DFS)?

A. Exploring as deep as possible then backtracking up
B. Visiting nodes in alphabetical order of their labels
C. Exploring all neighbors before going to next level
D. Only visiting nodes that are directly connected to root

Answer: A

Q193. Which data structure does BFS use?

A. Heap
B. Queue
C. Array
D. Stack

Answer: B

Q194. Which data structure does DFS typically use?

A. Queue
B. Stack
C. Heap
D. Deque

Answer: B

Q195. What is the time complexity of BFS on a graph with V vertices and E edges?

A. O(E^2)
B. O(V^2)
C. O(V * E)
D. O(V + E)

Answer: D

Q196. What is a spanning tree of a graph?

A. A complete copy of the original graph with all edges
B. A tree containing only the leaf vertices of the graph
C. A subgraph containing all vertices with minimum edges
D. A subgraph that connects only half of the graph vertices

Answer: C

Q197. What is a connected graph?

A. A graph with no isolated vertices but possible missing edges
B. A graph where there is a path between every pair of nodes
C. A graph where all vertices have the same degree exactly
D. A graph where every pair of vertices has a direct edge

Answer: B

Q198. What is a path in a graph?

A. A sequence of vertices connected by edges in graph
B. A cycle that visits every vertex exactly once only
C. The shortest distance between two vertices in graph
D. A single edge connecting exactly two graph vertices

Answer: A

Q199. What is a cycle in a graph?

A. A subgraph with all vertices having degree exactly two
B. A path that starts and ends at the same vertex
C. An edge that connects a vertex to itself directly
D. A path that visits every edge in the graph once

Answer: B

Q200. What is the time complexity of DFS?

A. O(V^2)
B. O(E^2)
C. O(V + E)
D. O(V * E)

Answer: C

Q201. What is the greedy algorithm approach?

A. Making the locally optimal choice at each step taken
B. Breaking problems into subproblems and combining them
C. Trying all possible solutions and picking the best one
D. Using random choices to approximate the best solution

Answer: A

Q202. What is divide and conquer?

A. Dividing resources equally among parallel processors
B. Breaking a problem into subproblems, solving and combining
C. Conquering the problem by brute force enumeration method
D. Splitting input into two and sorting each half only

Answer: B

Q203. What is dynamic programming?

A. Solving problems by storing solutions to subproblems
B. Programming in a dynamic language like Python code
C. Creating programs that dynamically allocate all memory
D. A programming style that changes code at runtime

Answer: A

Q204. What is a brute force algorithm?

A. An algorithm that uses force-directed graph layouts
B. An algorithm that tries all possible candidate solutions
C. An algorithm optimized for maximum efficiency speed
D. An algorithm that aggressively prunes search space

Answer: B

Q205. What is the main advantage of dynamic programming over recursion?

A. It does not use any memory for storing computations
B. It uses less code and is simpler to implement
C. It always runs in constant time for all input types
D. It avoids recomputing overlapping subproblems using cache

Answer: D

Q206. Which algorithm technique does binary search use?

A. Dynamic programming
B. Greedy approach
C. Divide and conquer
D. Backtracking

Answer: C

Q207. What is backtracking?

A. Going back to the previous version of code for fixes
B. Backing up data structures before modifying them now
C. Running the algorithm backwards from output to input
D. Systematically trying options and undoing bad choices

Answer: D

Q208. What type of problems does the greedy approach work best for?

A. Problems requiring all possible solutions enumerated
B. Problems with optimal substructure and greedy property
C. Problems that require exhaustive brute force searching
D. Problems with overlapping subproblems needing caching

Answer: B

Q209. What is memoization?

A. Memorizing the steps of an algorithm for quick recall
B. Writing memos about algorithm design decisions made
C. Creating documentation for each function in the code
D. Caching results of function calls to avoid recomputation

Answer: D

Q210. What is an example of a greedy algorithm?

A. Making change with fewest coins using largest first
B. N-Queens problem solved using backtracking exploration
C. Fibonacci sequence calculation using dynamic programming
D. Matrix chain multiplication using optimal parenthesization

Answer: A

Q211. What is a priority queue?

A. A stack that prioritizes the most recently added element
B. A linked list sorted by element insertion time stamps
C. A queue that processes elements in random arrival order
D. A queue where elements are processed by priority order

Answer: D

Q212. What is a heap data structure?

A. A hash table with elements ordered by insertion time
B. A sorted array stored in dynamic heap memory allocation
C. A complete binary tree satisfying the heap ordering rule
D. A linked list where elements are organized by priority

Answer: C

Q213. What is the difference between a max-heap and a min-heap?

A. Max-heap stores more elements; min-heap stores fewer items
B. Max-heap uses arrays; min-heap uses linked list structure
C. Max-heap has root as maximum; min-heap has root as min
D. Max-heap has root as minimum; min-heap has root as max

Answer: C

Q214. What is a trie data structure?

A. A tree for storing strings with shared prefix paths
B. A generic tree with exactly three children per node
C. A hash table variant that tries multiple hash functions
D. A balanced binary tree for efficient numeric searching

Answer: A

Q215. What is the time complexity of inserting into a binary heap?

A. O(1)
B. O(n^2)
C. O(n)
D. O(log n)

Answer: D

Q216. What is a graph adjacency matrix?

A. A tree representation of the graph traversal order
B. A 2D array indicating edges between pairs of vertices
C. A linked list of all vertices in the graph structure
D. A hash map from vertex names to edge weight values

Answer: B

Q217. What is a disjoint set?

A. A set that has been divided into two equal-sized halves
B. A set that contains overlapping elements with other sets
C. A collection of non-overlapping sets with no shared items
D. A sorted set used for binary search on set elements

Answer: C

Q218. What is the extract-max operation in a max-heap?

A. Adding a new maximum element to the top of the heap
B. Removing and returning the maximum element from heap
C. Finding the maximum element without removing it at all
D. Swapping the maximum and minimum elements in the heap

Answer: B

Q219. How is a heap typically stored in memory?

A. As a contiguous array with index-based navigation
B. As a linked list with pointers between nodes
C. As a hash table with heap position as the key
D. As a 2D matrix with rows and columns for levels

Answer: A

Q220. What is the purpose of the heapify operation?

A. Converting an arbitrary array into a valid heap structure
B. Doubling the size of the heap for more element storage
C. Removing all elements from the heap data structure now
D. Sorting all elements of the heap in ascending order output

Answer: A

Q221. What is string matching?

A. Sorting strings in lexicographic order for indexing
B. Concatenating multiple strings into one single string
C. Finding occurrences of a pattern within a longer text
D. Comparing two strings for exact equality testing

Answer: C

Q222. What is the brute force string matching time complexity?

A. O(m)
B. O(n * m)
C. O(n)
D. O(n + m)

Answer: B

Q223. What is a substring?

A. A sequence of non-adjacent characters from a string
B. A reversed copy of the original complete string
C. A rearrangement of characters in a given string
D. A contiguous sequence of characters within a string

Answer: D

Q224. What is a palindrome?

A. A string that contains only unique characters in it
B. A string that reads the same forwards and backwards
C. A string where all characters are in alphabetical order
D. A string with an even number of total characters only

Answer: B

Q225. What is the length of the string 'hello'?

A. Six
B. Five
C. Four
D. Three

Answer: B

Q226. What is string concatenation?

A. Splitting a string into individual characters apart
B. Removing duplicate characters from a given string
C. Reversing the order of characters within a string
D. Joining two or more strings end to end together

Answer: D

Q227. What is a prefix of a string?

A. The last character of the string only always
B. Any substring starting from the first character
C. The middle character of the string only always
D. Any character appearing in the string at all

Answer: B

Q228. What is a suffix of a string?

A. A reversed copy of the string prefix portion
B. The string with all vowels removed from it
C. Any substring ending at the last character point
D. The first character of the string only always

Answer: C

Q229. What is lexicographic order?

A. Ordering strings by their total length in characters
B. Ordering strings by the frequency of their characters
C. Ordering strings alphabetically like in a dictionary
D. Ordering strings by their last character value only

Answer: C

Q230. How many substrings does a string of length n have?

A. n(n+1)/2
B. n^2
C. 2n
D. n

Answer: A

Q231. What is time complexity?

A. A measure of how runtime grows with input size
B. The clock speed required to run the algorithm
C. The actual runtime of a program in seconds only
D. The time taken to write the algorithm pseudocode

Answer: A

Q232. What is space complexity?

A. The disk space needed to store source code files
B. The number of lines of code in the program total
C. The physical space the computer occupies on a desk
D. A measure of how memory usage grows with input size

Answer: D

Q233. Which is faster: O(n) or O(n^2)?

A. O(n) is faster for sufficiently large input sizes
B. O(n^2) is faster for all possible input sizes
C. Both have identical performance for any input size
D. It depends entirely on the programming language used

Answer: A

Q234. What does O(1) mean?

A. The algorithm processes only one element at any time
B. The algorithm takes exactly one single operation only
C. The algorithm runs in constant time regardless of input
D. The algorithm has exactly one line of code to execute

Answer: C

Q235. What is the Big-O of two nested loops each running n times?

A. O(n)
B. O(2n)
C. O(n^2)
D. O(log n)

Answer: C

Q236. Which complexity class represents logarithmic growth?

A. O(log n)
B. O(n^2)
C. O(n)
D. O(1)

Answer: A

Q237. What is optimization in algorithm design?

A. Converting the algorithm to a different language only
B. Adding more features to the algorithm's functionality
C. Improving algorithm efficiency in time or space usage
D. Making the code look cleaner and more readable

Answer: C

Q238. What is the time complexity of a loop that halves n each iteration?

A. O(n)
B. O(1)
C. O(n^2)
D. O(log n)

Answer: D

Q239. Why do we ignore constants in Big-O notation?

A. Constants are only relevant in space complexity analysis
B. Constants don't affect growth rate for large input sizes
C. Constants are always equal to one in all algorithms
D. Constants make the notation too hard to write out

Answer: B

Q240. What is the purpose of algorithm analysis?

A. To create documentation for the algorithm implementation
B. To predict resource usage and compare algorithm efficiency
C. To determine the programming language to use for coding
D. To find bugs and errors in the source code of program

Answer: B

Q241. What does asymptotic analysis focus on?

A. Algorithm behavior for small inputs only
B. Algorithm behavior as input size grows very large
C. Exact running time in milliseconds
D. The programming language used

Answer: B

Q242. Which of the following describes O(log n) complexity?

A. The algorithm runs in constant time
B. The running time doubles when the input size doubles
C. The running time increases by a constant when the input size doubles
D. The running time grows proportionally to the square of the input

Answer: C

Q243. What is the primary difference between a data structure and an algorithm?

A. Data structures are faster than algorithms
B. A data structure organizes data while an algorithm defines steps to solve a problem
C. Algorithms store data while data structures process data
D. There is no difference between them

Answer: B

Q244. What is the time complexity of performing two consecutive operations, one O(n) and one O(n^2)?

A. O(n)
B. O(n^3)
C. O(n^2)
D. O(2n^2)

Answer: C

Q245. Which data structure follows the First-In-First-Out principle?

A. Stack
B. Queue
C. Binary tree
D. Hash table

Answer: B

Q246. What happens when a recursive function has no base case?

A. It runs faster
B. It returns zero
C. It results in infinite recursion or a stack overflow
D. It automatically stops after 100 calls

Answer: C

Q247. What is the Big-O complexity of an algorithm that does nothing regardless of input?

A. O(n)
B. O(0)
C. O(1)
D. O(log n)

Answer: C

Q248. Which notation describes the best-case performance of an algorithm?

A. Big-O (O)
B. Big-Omega (:'•
C. Big-Theta (9,•
D. Little-o (o)

Answer: B

Q249. What is an iterative algorithm?

A. An algorithm that calls itself
B. An algorithm that uses loops to repeat steps
C. An algorithm that solves problems using random choices
D. An algorithm that cannot be analyzed

Answer: B

Q250. In Big-O notation, what is O(n) commonly called?

A. Constant time
B. Logarithmic time
C. Linear time
D. Quadratic time

Answer: C

Q251. What is the time complexity of appending an element at the end of a dynamic array (without resizing)?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q252. What is the difference between a stack and a queue?

A. A stack uses FIFO and a queue uses LIFO
B. A stack uses LIFO and a queue uses FIFO
C. Both use FIFO
D. Both use LIFO

Answer: B

Q253. Why can a doubly linked list be traversed in both directions?

A. Because it has two data fields per node
B. Because each node stores pointers to both the next and previous nodes
C. Because it stores data in pairs
D. Because it uses two separate arrays

Answer: B

Q254. What is the time complexity of deleting the first element from a singly linked list?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q255. Which operation checks the top element of a stack without removing it?

A. Pop
B. Push
C. Peek
D. Enqueue

Answer: C

Q256. What is the main advantage of an array over a linked list?

A. Dynamic size
B. O(1) random access to any element by index
C. Easier insertion in the middle
D. Less memory usage per element

Answer: B

Q257. What is a circular linked list?

A. A linked list where the last node points back to the first node
B. A linked list arranged in a circle shape on screen
C. A linked list with circular data values
D. A linked list that can only be traversed backwards

Answer: A

Q258. What is the time complexity of accessing the last element in a singly linked list?

A. O(1)
B. O(log n)
C. O(n)
D. O(n^2)

Answer: C

Q259. Which data structure is commonly used for function call management in programming languages?

A. Queue
B. Stack
C. Linked list
D. Hash table

Answer: B

Q260. What is a dynamic array?

A. An array that stores only dynamic data types
B. An array that automatically resizes when it runs out of capacity
C. An array allocated at compile time
D. An array that cannot change its values

Answer: B

Q261. How many edges does a tree with n nodes have?

A. n
B. n + 1
C. n - 1
D. 2n

Answer: C

Q262. What is the difference between a directed and an undirected graph?

A. Directed graphs have weighted edges
B. In a directed graph, edges have a direction from one vertex to another
C. Undirected graphs cannot have cycles
D. Directed graphs always have fewer edges

Answer: B

Q263. What property does a min-heap maintain between parent and child nodes?

A. Each parent is greater than its children
B. Each parent is less than or equal to its children
C. Each parent equals its children
D. Children are sorted left to right

Answer: B

Q264. What defines a simple path in a graph?

A. A path that visits every vertex
B. A path where no vertex is repeated
C. A path with exactly one edge
D. A path that forms a cycle

Answer: B

Q265. What is the in-order traversal order for a binary tree?

A. Root, Left, Right
B. Left, Root, Right
C. Left, Right, Root
D. Right, Root, Left

Answer: B

Q266. What is an adjacency matrix?

A. A list of all vertices in a graph
B. A 2D matrix where entry (i,j) indicates if there is an edge between vertex i and vertex j
C. A matrix storing the shortest paths
D. A matrix of node values in a tree

Answer: B

Q267. What is the maximum number of children a node in a binary tree can have?

A. 1
B. 2
C. 3
D. Unlimited

Answer: B

Q268. What condition must an undirected graph satisfy to be considered connected?

A. Every vertex must have the same degree
B. There must be a path between every pair of vertices
C. It must have no edges
D. Every vertex must connect to every other vertex directly

Answer: B

Q269. Where is the maximum element located in a max-heap?

A. At any leaf node
B. At the last position in the array
C. At the root
D. At the middle of the array

Answer: C

Q270. Can a simple graph contain self-loops?

A. Yes, simple graphs allow self-loops
B. No, simple graphs do not allow self-loops or multiple edges between the same pair of vertices
C. Only directed simple graphs allow self-loops
D. Only weighted simple graphs allow self-loops

Answer: B

Q271. In linear search, where does the search begin?

A. At the middle of the array
B. At the end of the array
C. At the first element of the array
D. At a random position

Answer: C

Q272. What is the best-case time complexity of binary search?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q273. Can binary search be applied to a linked list?

A. Yes, and it runs in O(log n)
B. No, because linked lists do not support efficient random access
C. Yes, but only on doubly linked lists
D. Yes, if the list is circular

Answer: B

Q274. What does binary search do at each step?

A. Checks every element one by one
B. Divides the search space in half by comparing with the middle element
C. Swaps elements to sort the array
D. Hashes the target value

Answer: B

Q275. What is the worst-case number of comparisons for linear search on an array of 100 elements?

A. 50
B. 10
C. 100
D. 7

Answer: C

Q276. Does linear search require the array to be sorted?

A. Yes, it always requires sorting
B. No, linear search works on both sorted and unsorted arrays
C. Only for integer arrays
D. Only when searching for the maximum element

Answer: B

Q277. In binary search on a sorted array, if the target equals the middle element, what happens?

A. The search continues in the left half
B. The search continues in the right half
C. The search returns the index of the middle element
D. The search restarts from the beginning

Answer: C

Q278. How many comparisons does binary search need at most for an array of 16 elements?

A. 16
B. 8
C. 4
D. 5

Answer: D

Q279. What is the average-case time complexity of linear search?

A. O(1)
B. O(n/2), which simplifies to O(n)
C. O(log n)
D. O(n^2)

Answer: B

Q280. Which searching algorithm is most suitable when the dataset is very small?

A. Binary search
B. Interpolation search
C. Linear search
D. Exponential search

Answer: C

Q281. What is the best-case time complexity of bubble sort?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(1)

Answer: C

Q282. How does selection sort work?

A. It inserts each element into its correct position
B. It repeatedly finds the minimum element and places it at the beginning of the unsorted portion
C. It divides the array and merges sorted halves
D. It swaps adjacent elements

Answer: B

Q283. What is the space complexity of bubble sort?

A. O(n)
B. O(n^2)
C. O(log n)
D. O(1)

Answer: D

Q284. How does insertion sort process elements?

A. It picks elements randomly and places them in sorted order
B. It takes each element and inserts it into its correct position in the already sorted portion
C. It divides the array into halves recursively
D. It uses a hash function to determine positions

Answer: B

Q285. Which sorting algorithm is similar to how people typically sort playing cards in their hand?

A. Bubble sort
B. Selection sort
C. Insertion sort
D. Merge sort

Answer: C

Q286. What is the worst-case time complexity of insertion sort?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(log n)

Answer: C

Q287. Is bubble sort an in-place sorting algorithm?

A. No, it requires O(n) extra space
B. Yes, it sorts the elements within the original array
C. No, it creates a copy of the array
D. It depends on the implementation

Answer: B

Q288. What does the term 'comparison-based sorting' mean?

A. Sorting that only works on numbers
B. Sorting that determines order only by comparing pairs of elements
C. Sorting that compares the array with a sorted version
D. Sorting that uses bit manipulation

Answer: B

Q289. How many swaps does selection sort perform in the worst case for n elements?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(1)

Answer: C

Q290. After the first complete pass of bubble sort on [5, 3, 8, 1, 2], which element is guaranteed to be in its final position?

A. 5
B. 1
C. 8
D. 2

Answer: C

Q291. What is the primary purpose of a hash function?

A. To sort data
B. To map data of arbitrary size to fixed-size values
C. To encrypt data
D. To compress data

Answer: B

Q292. What is chaining in the context of hash table collision resolution?

A. Linking hash tables together
B. Storing all elements that hash to the same index in a linked list at that index
C. Creating a chain of hash functions
D. Connecting the first and last elements of the hash table

Answer: B

Q293. If a hash table has 10 slots and the hash function is h(k) = k mod 10, where does the key 27 go?

A. Slot 2
B. Slot 7
C. Slot 3
D. Slot 27

Answer: B

Q294. What is the time complexity of inserting an element into a hash table in the average case?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q295. What is a hash collision?

A. When the hash table runs out of memory
B. When two different keys produce the same hash value
C. When a hash function produces a negative value
D. When the hash table is empty

Answer: B

Q296. What is the load factor of a hash table with 15 elements and 20 slots?

A. 0.25
B. 0.75
C. 1.33
D. 15

Answer: B

Q297. What property should a good hash function have?

A. It should always return the same value
B. It should distribute keys uniformly across the hash table
C. It should sort the keys
D. It should only work with string keys

Answer: B

Q298. Can two different keys have the same hash value?

A. No, hash functions always produce unique values
B. Yes, this is called a collision
C. Only if the keys are equal
D. Only in broken hash functions

Answer: B

Q299. What is the worst-case time complexity of searching in a hash table?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: C

Q300. What is the difference between a hash map and a hash table?

A. Hash maps are always faster
B. A hash map stores key-value pairs while a hash set stores only keys; both use hashing internally
C. Hash tables cannot handle collisions
D. There is no difference at all

Answer: B

Q301. What is the time complexity of finding the height of a binary tree?

A. O(1)
B. O(log n)
C. O(n)
D. O(n^2)

Answer: C

Q302. Where is the in-order successor located if a BST node has a right subtree?

A. It is the parent of the node
B. It is the leftmost node in the right subtree
C. It is the root of the tree
D. It is the rightmost node in the left subtree

Answer: B

Q303. What is the result of an in-order traversal of a BST?

A. Elements in reverse order
B. Elements in random order
C. Elements in sorted (ascending) order
D. Elements level by level

Answer: C

Q304. What data structure is used to implement level-order traversal of a tree?

A. Stack
B. Queue
C. Hash table
D. Priority queue

Answer: B

Q305. How do you count the total number of nodes in a binary tree?

A. Count only the leaf nodes
B. Recursively count nodes in left subtree, right subtree, and add 1 for the root
C. Count only the internal nodes
D. Use binary search

Answer: B

Q306. What is the time complexity of inserting a new element into a BST?

A. O(1)
B. O(n) in the worst case
C. O(n^2)
D. O(log n) always

Answer: B

Q307. In which traversal order is the root node visited before its subtrees?

A. Post-order
B. In-order
C. Pre-order
D. Level-order only

Answer: C

Q308. Which direction do you traverse in a BST to find the maximum element?

A. Always go left until a leaf is reached
B. Alternate between left and right
C. Always go right until a node with no right child is reached
D. Go to the root and stop

Answer: C

Q309. Which tree traversal is commonly used to delete a binary tree safely (deleting children before the parent)?

A. Pre-order traversal
B. In-order traversal
C. Post-order traversal
D. Level-order traversal

Answer: C

Q310. How many leaf nodes does a full binary tree with n internal nodes have?

A. n
B. n + 1
C. n - 1
D. 2n

Answer: B

Q311. What is the difference between BFS and DFS?

A. BFS uses a stack and DFS uses a queue
B. BFS explores neighbors level by level using a queue, while DFS goes deep using a stack
C. BFS is faster than DFS
D. DFS finds shortest paths but BFS does not

Answer: B

Q312. What does DFS stand for?

A. Direct First Search
B. Depth First Search
C. Double Find Search
D. Data Flow System

Answer: B

Q313. How many edges does a spanning tree of a connected graph with V vertices contain?

A. V
B. V - 1
C. V + 1
D. 2V

Answer: B

Q314. Can BFS be used to find the shortest path in an unweighted graph?

A. No, only Dijkstra's algorithm can find shortest paths
B. Yes, BFS finds the shortest path in terms of number of edges in an unweighted graph
C. Only if the graph is a tree
D. Only if the graph has no cycles

Answer: B

Q315. What is the space complexity of BFS on a graph with V vertices?

A. O(1)
B. O(V)
C. O(V^2)
D. O(E)

Answer: B

Q316. What is an acyclic graph?

A. A graph with exactly one cycle
B. A graph that contains no cycles
C. A graph with all vertices connected
D. A graph with no edges

Answer: B

Q317. In a weighted graph, what do the edge weights typically represent?

A. The number of vertices
B. Costs, distances, or capacities associated with traveling along that edge
C. The color of the edge
D. The degree of adjacent vertices

Answer: B

Q318. How is a graph typically represented in code using an adjacency list?

A. As a 2D array of size V×V
B. As an array of linked lists (or arrays), where each vertex has a list of its neighbors
C. As a single sorted array
D. As a hash table mapping edges to vertices

Answer: B

Q319. What happens if DFS visits a vertex that has already been visited in a directed graph?

A. It always means a cycle exists
B. It depends on whether the previously visited vertex is still on the current DFS path
C. DFS stops immediately
D. The graph is invalid

Answer: B

Q320. What is the minimum number of edges in a connected undirected graph with V vertices?

A. V
B. V - 1
C. V + 1
D. V / 2

Answer: B

Q321. What is the divide and conquer approach?

A. Solving a problem by trying all possible solutions
B. Breaking a problem into smaller subproblems, solving them, and combining the solutions
C. Using a greedy choice at each step
D. Storing solutions to avoid recomputation

Answer: B

Q322. What are the two key properties required for a dynamic programming solution?

A. Greedy choice and local optima
B. Optimal substructure and overlapping subproblems
C. Random choices and large inputs
D. Divide and conquer and backtracking

Answer: B

Q323. What is an example of a problem solved by the greedy approach?

A. 0/1 Knapsack problem
B. Making change with the fewest coins (when denominations are canonical)
C. Traveling Salesman Problem
D. Matrix chain multiplication

Answer: B

Q324. What is the key difference between greedy algorithms and dynamic programming?

A. Greedy algorithms are always slower
B. Greedy algorithms make locally optimal choices without reconsidering, while DP considers all subproblem solutions
C. Dynamic programming never gives optimal solutions
D. They are the same approach

Answer: B

Q325. What is the bottom-up approach in dynamic programming?

A. Solving the problem using recursion from the top
B. Solving smaller subproblems first and building up to the original problem using a table
C. Randomly solving subproblems
D. Starting from the solution and working backwards

Answer: B

Q326. How does backtracking reduce the search space compared to brute force?

A. It sorts the candidates first
B. It abandons partial solutions as soon as they are determined to be invalid, avoiding exploration of dead-end branches
C. It uses hashing to skip solutions
D. It randomly selects solutions to test

Answer: B

Q327. What is the time complexity of computing Fibonacci numbers using memoization?

A. O(2^n)
B. O(n^2)
C. O(n)
D. O(log n)

Answer: C

Q328. Which algorithm design paradigm does merge sort belong to?

A. Greedy
B. Dynamic programming
C. Divide and conquer
D. Backtracking

Answer: C

Q329. What is the purpose of the top-down approach in dynamic programming?

A. To sort elements from top to bottom
B. To solve the original problem recursively and cache results to avoid redundant computation
C. To iterate through the array from beginning to end
D. To find the maximum element first

Answer: B

Q330. What is a brute force approach to problem solving?

A. An optimized algorithm
B. An approach that tries every possible solution and checks which one is correct
C. An approach that uses heuristics
D. An approach based on random sampling

Answer: B

Q331. What is the time complexity of inserting a word of length L into a trie?

A. O(1)
B. O(L)
C. O(n)
D. O(L * n)

Answer: B

Q332. What is the advantage of a priority queue over a sorted array for maintaining a dynamic set?

A. Priority queues use less memory
B. Priority queues support O(log n) insertion while sorted arrays require O(n) for insertion
C. Priority queues allow O(1) random access
D. Sorted arrays cannot store integers

Answer: B

Q333. What does the find operation do in a Union-Find data structure?

A. Finds the minimum element in a set
B. Returns the representative (root) of the set containing a given element
C. Finds the union of two sets
D. Searches for an element in a sorted array

Answer: B

Q334. What is a min-heap's property regarding the root element?

A. The root is the largest element
B. The root is the median element
C. The root is the smallest element
D. The root has no special property

Answer: C

Q335. What is the time complexity of the extract-min operation in a binary min-heap?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: B

Q336. What is a trie primarily used for?

A. Sorting numbers
B. Efficient storage and retrieval of strings, supporting prefix-based operations
C. Graph traversal
D. Matrix operations

Answer: B

Q337. What is the union operation in Union-Find?

A. Finding all elements in a set
B. Merging two disjoint sets into one
C. Splitting a set into two parts
D. Sorting elements within a set

Answer: B

Q338. For a node at index i in a binary heap stored in an array (0-indexed), what is the index of its left child?

A. i + 1
B. 2i
C. 2i + 1
D. i / 2

Answer: C

Q339. What is a multimap data structure?

A. A map that can store multiple values for the same key
B. A map that uses multiple hash functions
C. A map with multiple levels
D. A map that can only store maps

Answer: A

Q340. What is the delete operation in a binary heap called?

A. Remove
B. Extract
C. Pop
D. Decrease-key followed by extract-min

Answer: D

Q341. What is the time complexity of comparing two strings of length n for equality?

A. O(1)
B. O(n)
C. O(n^2)
D. O(log n)

Answer: B

Q342. Is 'ace' a valid subsequence of 'abcde'?

A. No, because the characters are not contiguous
B. Yes, because the characters appear in the same relative order in the original string
C. Only if we rearrange the original string
D. No, because 'ace' is not a word

Answer: B

Q343. How many total substrings does a string of length n have (including the empty string)?

A. n
B. n^2
C. n(n+1)/2 + 1
D. 2^n

Answer: C

Q344. In lexicographic order, which string comes first: 'apple' or 'application'?

A. 'application' because it is longer
B. 'apple' because at the first differing position, 'e' < 'i'
C. They are considered equal
D. 'apple' because shorter strings always come first

Answer: B

Q345. What is the time complexity of reversing a string of length n?

A. O(1)
B. O(log n)
C. O(n)
D. O(n^2)

Answer: C

Q346. What is the time complexity of the naive (brute force) pattern matching algorithm?

A. O(n + m)
B. O(n * m) in the worst case
C. O(n)
D. O(m)

Answer: B

Q347. How can you check if a string is a palindrome?

A. Sort the string and check if it's the same
B. Compare the string with its reverse
C. Check if all characters are the same
D. Count the number of vowels

Answer: B

Q348. What is the difference between a substring and a subsequence?

A. They are the same thing
B. A substring is contiguous characters; a subsequence can skip characters but maintains order
C. A subsequence is always longer
D. A substring can skip characters

Answer: B

Q349. What does the term 'pattern matching' mean in the context of strings?

A. Creating patterns from strings
B. Finding occurrences of a pattern string within a text string
C. Matching two strings for equality
D. Sorting strings by pattern

Answer: B

Q350. What is string immutability?

A. Strings that can be modified in place
B. Strings that cannot be changed after creation; any modification creates a new string
C. Strings stored in read-only memory
D. Strings with fixed length

Answer: B

Q351. What is the Big-O complexity of an algorithm with running time 3n + 5?

A. O(5)
B. O(3n)
C. O(n)
D. O(n + 5)

Answer: C

Q352. Which grows faster: O(n^2) or O(2^n)?

A. O(n^2) grows faster
B. They grow at the same rate
C. O(2^n) grows faster
D. It depends on n

Answer: C

Q353. What is the time complexity of a loop that runs from 1 to n and increments by 2 each time?

A. O(n^2)
B. O(n)
C. O(n/2), which is O(n)
D. O(log n)

Answer: C

Q354. What does it mean to optimize an algorithm?

A. To make the code longer
B. To improve the algorithm's time or space efficiency
C. To add more features
D. To translate it to a faster programming language

Answer: B

Q355. What is the time complexity of checking if a number n is even or odd?

A. O(n)
B. O(log n)
C. O(1)
D. O(n^2)

Answer: C

Q356. Why is O(1) the most efficient time complexity?

A. Because it means the algorithm does nothing
B. Because the execution time does not increase with input size
C. Because it uses no memory
D. Because it always runs in exactly 1 millisecond

Answer: B

Q357. What is the space complexity of an algorithm that uses only a fixed number of variables?

A. O(n)
B. O(1)
C. O(log n)
D. O(n^2)

Answer: B

Q358. What does it mean when we say an algorithm runs in 'polynomial time'?

A. It always finishes in under a second
B. Its running time is bounded by n^k for some constant k
C. It uses polynomial equations
D. It can only process polynomials

Answer: B

Q359. If an algorithm has two independent steps with complexities O(n) and O(m), what is the overall complexity?

A. O(n * m)
B. O(n + m)
C. O(max(n, m))
D. O(n^2)

Answer: B

Q360. Why do we use Big-O notation instead of measuring exact execution time?

A. Because exact time is always the same
B. Because Big-O describes how the algorithm scales with input size, independent of hardware and implementation
C. Because Big-O is more precise
D. Because exact time cannot be measured

Answer: B

Medium Questions

360 questions

Q361. What is the time complexity of the recurrence T(n) = 2T(n/2) + n?

A. O(n^2)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: D

Q362. Which case of the Master Theorem applies when f(n) = 9,†åâ†Æöuö"† ’’“ð

A. Case 3 applies
B. No case applies
C. Case 2 applies
D. Case 1 applies

Answer: C

Q363. What does Omega (:’’ æ÷F F–öâ &W &W6VçCð

A. Exact tight bound complexity
B. Worst-case upper bound complexity
C. Best-case or lower bound complexity
D. Average-case expected complexity

Answer: C

Q364. If an algorithm has complexities O(n^2) in the worst case and :’†â’ –â F†R &W7B 6 6P, which is true?

A. The algorithm always runs in O(n) time
B. The algorithm always runs in O(n^2) time
C. The algorithm runs in O(n log n) always
D. Performance varies between :’†â’ æB ò†åã"•

Answer: D

Q365. What is amortized analysis?

A. Analyzing the worst case of each operation individually
B. Averaging the cost of operations over a sequence
C. Analyzing only the best case of each operation
D. Computing the cost using a sorting-based analysis

Answer: B

Q366. Which growth rate is faster: O(2^n) or O(n^3)?

A. They are the same
B. O(n^3) is faster
C. O(2^n) is faster
D. Depends on input size

Answer: C

Q367. What is the time complexity of a nested loop where both loops run n times?

A. O(n log n)
B. O(n^2)
C. O(2n)
D. O(n)

Answer: B

Q368. In asymptotic analysis, which of these is the correct ordering from slowest to fastest growth?

A. O(1) < O(log n) < O(n) < O(n^2)
B. O(1) < O(n) < O(log n) < O(n^2)
C. O(n) < O(log n) < O(1) < O(n^2)
D. O(log n) < O(1) < O(n) < O(n^2)

Answer: A

Q369. What is the recurrence relation for binary search?

A. T(n) = 2T(n/2) + 1
B. T(n) = T(n-1) + 1
C. T(n) = T(n/2) + n
D. T(n) = T(n/2) + 1

Answer: D

Q370. What does Theta (9,’ æ÷F F–öâ &W &W6VçCð

A. Only upper bound estimate
B. Tight bound (both upper and lower)
C. Only lower bound estimate
D. Average case analysis only

Answer: B

Q371. What is a circular queue?

A. A queue implemented using a linked list
B. A queue where the rear connects back to front
C. A double-ended queue with two pointers
D. A queue that automatically sorts elements

Answer: B

Q372. What is a doubly linked list?

A. A list with pointers to both next and previous
B. A list that stores two data values per node
C. A list structure with two separate heads
D. A circular list with bidirectional traversal

Answer: A

Q373. What is a deque (double-ended queue)?

A. A stack implemented as a queue structure
B. A priority queue with weighted elements
C. A queue allowing deletion from both ends only
D. A structure allowing insert and delete at both ends

Answer: D

Q374. What is the time complexity of deleting a node from the middle of a singly linked list (given a pointer to the previous node)?

A. O(1)
B. O(log n)
C. O(n)
D. O(n^2)

Answer: A

Q375. How can you implement a queue using two stacks?

A. It is not possible to implement using stacks
B. Use both stacks for enqueue operations only
C. Use one stack for enqueue and another for dequeue
D. Merge both stacks into a single combined structure

Answer: C

Q376. What is the advantage of a linked list over an array?

A. Better CPU cache performance overall
B. Faster random access by index
C. Dynamic size and efficient insertion
D. Less memory usage per element

Answer: C

Q377. What is a sentinel node in a linked list?

A. A dummy node to simplify boundary conditions
B. The node containing the maximum data value
C. A node that stores the total list size
D. The last node in the list structure

Answer: A

Q378. What is the condition for a circular queue being full (array implementation with size N)?

A. front == 0 && rear == N - 1
B. (rear + 1) % N == front
C. rear == N - 1
D. front == rear

Answer: B

Q379. What is the time complexity of searching for an element in an unsorted linked list?

A. O(n^2)
B. O(log n)
C. O(1)
D. O(n)

Answer: D

Q380. Which application commonly uses a stack?

A. Breadth-first graph search traversal
B. Level-order tree traversal processing
C. Function call management (call stack)
D. Round-robin job scheduling system

Answer: C

Q381. What is the time complexity of searching in a balanced BST?

A. O(log n)
B. O(n log n)
C. O(1)
D. O(n)

Answer: A

Q382. What is the worst-case time complexity of search in an unbalanced BST?

A. O(log n)
B. O(n^2)
C. O(1)
D. O(n)

Answer: D

Q383. What is the space complexity of an adjacency matrix for a graph with V vertices?

A. O(V)
B. O(V + E)
C. O(E)
D. O(V^2)

Answer: D

Q384. What is a complete binary tree?

A. A tree where every node has exactly 2 children always
B. A tree where all leaf nodes are at the same level
C. A tree with the maximum possible depth for its nodes
D. A tree with all levels full except possibly the last level

Answer: D

Q385. In a max-heap, where is the maximum element?

A. At the root
B. At the last level
C. At a leaf node
D. It could be anywhere

Answer: A

Q386. What is the space complexity of an adjacency list for a graph with V vertices and E edges?

A. O(E)
B. O(V^2)
C. O(V + E)
D. O(V)

Answer: C

Q387. What is the in-order traversal of a BST with values 4, 2, 6, 1, 3, 5, 7 (root=4)?

A. 1, 2, 3, 4, 5, 6, 7
B. 4, 2, 1, 3, 6, 5, 7
C. 4, 2, 6, 1, 3, 5, 7
D. 1, 3, 2, 5, 7, 6, 4

Answer: A

Q388. What type of graph has edges with associated weights?

A. Weighted graph
B. Directed graph
C. Bipartite graph
D. Complete graph

Answer: A

Q389. What is a full binary tree?

A. A tree with the maximum number of nodes
B. A tree where every level is completely filled
C. A tree where every node has 0 or 2 children
D. A tree where all leaves are at same depth

Answer: C

Q390. What is the time complexity of inserting an element into a max-heap?

A. O(n)
B. O(1)
C. O(log n)
D. O(n log n)

Answer: C

Q391. What is interpolation search?

A. A recursive linear search with memoization cache
B. A search that always checks the middle element
C. A search that uses a hash table for fast lookups
D. A search that estimates position based on value distribution

Answer: D

Q392. What is the average time complexity of interpolation search on uniformly distributed data?

A. O(log n)
B. O(1)
C. O(n)
D. O(log log n)

Answer: D

Q393. What is jump search?

A. A search that uses recursion to jump elements
B. A search that jumps by blocks then does linear scan
C. A search that jumps to random positions in array
D. A search that skips every other element each pass

Answer: B

Q394. What is the optimal block size for jump search on an array of n elements?

A. log n
B. " n
C. n
D. n/2

Answer: B

Q395. What is exponential search?

A. A search with exponential time complexity overall
B. A search designed for exponentially growing datasets
C. A search that doubles the index then uses binary search
D. A search that uses exponential mathematical functions

Answer: C

Q396. What is the time complexity of exponential search?

A. O(n)
B. O(" n)
C. O(n log n)
D. O(log n)

Answer: D

Q397. What is ternary search?

A. A three-pass sequential linear search method
B. A search specifically for the third element only
C. A search dividing array into three parts each step
D. A search that checks three elements at a time

Answer: C

Q398. Why is binary search preferred over ternary search in practice?

A. Binary search uses less memory per recursive call
B. Binary search is always faster in every case
C. Ternary search does not work on sorted arrays at all
D. Ternary search makes more comparisons per step overall

Answer: D

Q399. What is the space complexity of recursive binary search?

A. O(n)
B. O(n^2)
C. O(log n)
D. O(1)

Answer: C

Q400. In which scenario is linear search more efficient than binary search?

A. When the array has millions of elements
B. Linear search is never more efficient
C. Large sorted arrays with many elements
D. When the array is unsorted and small

Answer: D

Q401. What is the time complexity of merge sort?

A. O(n)
B. O(log n)
C. O(n log n)
D. O(n^2)

Answer: C

Q402. What is the space complexity of merge sort?

A. O(n)
B. O(n^2)
C. O(1)
D. O(log n)

Answer: A

Q403. What is the average-case time complexity of quick sort?

A. O(log n)
B. O(n log n)
C. O(n^2)
D. O(n)

Answer: B

Q404. What is the worst-case time complexity of quick sort?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(n log n)

Answer: C

Q405. What is heap sort's time complexity?

A. O(log n)
B. O(n)
C. O(n log n)
D. O(n^2)

Answer: C

Q406. Is merge sort a stable sorting algorithm?

A. Only for linked lists
B. Yes, it is stable
C. Only for array inputs
D. No, it is unstable

Answer: B

Q407. What is the divide-and-conquer strategy in merge sort?

A. Divide the array into three equal parts
B. Divide in half, sort each half, then merge
C. Divide by selecting a random pivot element
D. Divide into sorted and unsorted portions

Answer: B

Q408. What is the role of the pivot in quick sort?

A. It is always the first element chosen
B. It is always the exact median value
C. It partitions into smaller and larger groups
D. It merges two sorted subarrays together

Answer: C

Q409. What is counting sort?

A. A comparison-based sorting method
B. A non-comparison sort counting occurrences
C. A sort that uses a heap structure
D. A divide-and-conquer sorting approach

Answer: B

Q410. What is the space complexity of heap sort?

A. O(n log n)
B. O(n)
C. O(log n)
D. O(1)

Answer: D

Q411. What is open addressing in hash tables?

A. Using multiple hash tables for distribution
B. Finding another open slot when collision occurs
C. Doubling the table size on every collision
D. Using linked lists for collision resolution

Answer: B

Q412. What is linear probing?

A. Using a linear hash function for keys
B. Checking next consecutive slots on collision
C. Using a linked list for collision handling
D. Probing with a linear mathematical equation

Answer: B

Q413. What is the primary clustering problem in linear probing?

A. Keys are stored in sorted order clusters
B. Consecutive occupied slots form long clusters
C. Hash values clustering near zero index
D. Multiple hash functions produce same values

Answer: B

Q414. What is quadratic probing?

A. Using four different hash functions total
B. Squaring the hash value before modding
C. Using a quadratic hash function for keys
D. Probing at positions $h+1^2$, $h+2^2$, $h+3^2$, etc.

Answer: D

Q415. What is double hashing?

A. Using two hash functions for probe step size
B. Hashing the hash value itself recursively
C. Hashing a value twice in succession
D. Using two separate hash tables for storage

Answer: A

Q416. Why should the hash table size be a prime number?

A. It is required by all hash functions
B. It distributes hash values more uniformly
C. It speeds up the hash value computation
D. It makes the table smaller in memory

Answer: B

Q417. What is rehashing?

A. Hashing the same key multiple times in a row
B. Removing all elements and starting fresh over
C. Changing the hash function to a new one
D. Creating a larger table and reinserting all elements

Answer: D

Q418. What is the multiplication method for hashing?

A. h(k) = k mod (m × 2)
B. h(k) = k^2 mod m
C. h(k) = floor(m × (k × A mod 1)) where 0 < A < 1
D. h(k) = k × m

Answer: C

Q419. What is the advantage of chaining over open addressing?

A. Chaining is always faster for all operations
B. Chaining uses less memory per element overall
C. Chaining handles high load factors and deletion better
D. Chaining has better CPU cache performance overall

Answer: C

Q420. What is a hash set?

A. A multiset allowing duplicate elements
B. A set using hashing for fast unique lookups
C. A sorted set with ordered elements
D. A set implemented using sorted arrays

Answer: B

Q421. What is an AVL tree?

A. A tree with at most 3 children per node always
B. A self-balancing BST with height difference at most 1
C. A tree data structure used only for searching
D. A binary tree with no balancing guarantees

Answer: B

Q422. What are the rotation operations in an AVL tree?

A. Merge and split operations on subtrees
B. Insert and delete operations on tree nodes
C. Push and pop operations on internal nodes
D. Left, right, left-right, and right-left rotations

Answer: D

Q423. When is a left rotation performed in an AVL tree?

A. When the tree is already perfectly balanced
B. When the right subtree is too heavy (RR case)
C. When the left subtree is too heavy overall
D. After every single insertion operation occurs

Answer: B

Q424. How do you delete a node with two children in a BST?

A. Simply remove the node directly from tree
B. Swap with the root node then remove it
C. Replace with in-order successor then delete it
D. Set the node value to null and leave it

Answer: C

Q425. What is the time complexity of insertion in a Red-Black tree?

A. O(1)
B. O(n)
C. O(n log n)
D. O(log n)

Answer: D

Q426. What properties must a Red-Black tree satisfy?

A. All internal and external nodes must always be colored red only
B. Root is black, red nodes have black children, equal black-height on all paths
C. Height is always kept at exactly log n for any n total nodes
D. All leaf nodes are always colored red throughout the entire tree

Answer: B

Q427. What is the Lowest Common Ancestor (LCA) of two nodes in a tree?

A. The deepest node that is ancestor of both
B. The node with the smallest key value
C. The root node of the entire tree
D. The immediate parent of both given nodes

Answer: A

Q428. What is Morris traversal?

A. A traversal technique for graphs only
B. An in-order traversal using O(1) extra space
C. A traversal using an explicit stack structure
D. A breadth-first traversal using a queue

Answer: B

Q429. How can you check if a binary tree is a valid BST?

A. Check if each node has at most 2 children
B. Count the number of nodes in the tree
C. Verify in-order traversal is strictly increasing
D. Check if the root is the minimum value node

Answer: C

Q430. What is a segment tree used for?

A. Storing sorted segments of linked lists
B. Efficient range queries and point updates on arrays
C. Segmenting a network into subnetworks
D. Storing geometric line segments only

Answer: B

Q431. What is Dijkstra's algorithm used for?

A. Finding all cycles in a given graph
B. Finding shortest paths with non-negative weights
C. Sorting all vertices by their degree
D. Finding connected components in a graph

Answer: B

Q432. What is the time complexity of Dijkstra's algorithm using a binary heap?

A. O(V × E)
B. O(V + E)
C. O((V + E) log V)
D. O(V^2)

Answer: C

Q433. What is Kruskal's algorithm used for?

A. Shortest path computation
B. Topological sorting of vertices
C. Cycle detection in graphs
D. Finding Minimum Spanning Tree

Answer: D

Q434. What is Prim's algorithm used for?

A. Shortest path computation
B. Finding Minimum Spanning Tree
C. Graph coloring assignment
D. Topological sort ordering

Answer: B

Q435. What is topological sorting?

A. Linear ordering of DAG vertices respecting edge directions
B. Sorting edges by their assigned weight values
C. Sorting all vertices by their numeric label values
D. Sorting all vertices by their in-degree values

Answer: A

Q436. What type of graph is required for topological sorting?

A. An undirected graph type
B. A complete graph type
C. A weighted graph type
D. Directed Acyclic Graph (DAG)

Answer: D

Q437. What is the Bellman-Ford algorithm used for?

A. Finding connected components in undirected graphs
B. Finding the Minimum Spanning Tree of a graph
C. Finding shortest paths even with negative weights
D. Topological sorting of directed acyclic graphs

Answer: C

Q438. What is the time complexity of the Bellman-Ford algorithm?

A. O(V × E)
B. O(V + E)
C. O(V^2)
D. O(E log V)

Answer: A

Q439. How can you detect a cycle in a directed graph?

A. By counting edges in the graph
B. Using DFS and checking for back edges
C. Using BFS traversal only
D. By finding the shortest path first

Answer: B

Q440. What is the Union-Find (Disjoint Set Union) data structure used for?

A. Finding shortest paths between graph vertices
B. Storing key-value pairs with fast lookup
C. Tracking set membership and merging sets efficiently
D. Sorting elements into ordered groups

Answer: C

Q441. What is the time complexity of the dynamic programming solution for the 0/1 knapsack problem with n items and capacity W?

A. O(n log n)
B. O(2^n)
C. O(n × W)
D. O(n)

Answer: C

Q442. What is the Longest Common Subsequence (LCS) problem?

A. Finding the longest subsequence present in both sequences
B. Finding the longest increasing subsequence in an array
C. Finding the longest common substring between two strings
D. Finding the longest common prefix of two strings

Answer: A

Q443. What is the greedy choice property?

A. Always choosing the most expensive option first
B. A locally optimal choice leads to global optimum
C. A property that all algorithms must have
D. A property unique to dynamic programming only

Answer: B

Q444. What is the coin change problem (minimum coins)?

A. Finding minimum coins to make a target amount
B. Sorting coins by their face value order
C. Counting total coins in a collection
D. Finding the maximum number of coins used

Answer: A

Q445. What is the time complexity of the recursive Fibonacci without memoization?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(2^n)

Answer: D

Q446. How does memoization improve the Fibonacci computation?

A. It reduces time from O(2^n) to O(n) by caching
B. It makes the computation run in O(log n) time
C. It does not improve it at all
D. It makes the computation run in O(1) time

Answer: A

Q447. What is the N-Queens problem?

A. A graph traversal problem with N vertices
B. Sorting N queens by their rank and position
C. Finding N queens in a given dataset collection
D. Placing N queens so no two attack each other

Answer: D

Q448. What is the difference between top-down and bottom-up dynamic programming?

A. Top-down is always faster in practice than the bottom-up approach overall
B. Top-down uses recursion with memoization; bottom-up uses tabulation
C. Bottom-up always uses significantly more memory than the top-down approach
D. They are exactly the same fundamental approach overall

Answer: B

Q449. What is Huffman coding an example of?

A. Greedy algorithm
B. Backtracking
C. Dynamic programming
D. Divide and conquer

Answer: A

Q450. What is the Longest Increasing Subsequence (LIS) problem's time complexity with DP?

A. O(n)
B. O(2^n)
C. O(n log n)
D. O(n^2)

Answer: D

Q451. What is path compression in Union-Find?

A. Making every node in find path point to root
B. Compressing the storage used by the structure
C. Shortening paths between nodes in a graph
D. Compressing the data stored in each node

Answer: A

Q452. What is union by rank in Union-Find?

A. Attaching shorter tree under taller tree during union
B. Sorting all elements by their assigned rank
C. Ranking all elements before performing union
D. Using ranked hash functions for set operations

Answer: A

Q453. What is the nearly constant amortized time complexity of Union-Find with path compression and union by rank?

A. O(1) constant time
B. O(log log n) iterated log
C. O(log n) logarithmic
D. O(; †â'' –àverse Ackermann

Answer: D

Q454. What is a Fibonacci heap?

A. A heap with better amortized decrease-key and insert
B. A Fibonacci sequence number generator structure
C. A binary heap variant with extra child pointers
D. A heap that stores Fibonacci numbers only

Answer: A

Q455. What is a suffix array?

A. An array of string prefixes
B. A compressed trie structure
C. A sorted array of all string suffixes
D. An array suffix pointer

Answer: C

Q456. What is a sparse table?

A. A table with many empty cells throughout
B. A compressed hash table for sparse data
C. A sparse matrix representation structure
D. A structure for O(1) static range minimum queries

Answer: D

Q457. What is a k-d tree?

A. A space-partitioning tree for k-dimensional data
B. A tree with k children per node always
C. A tree of exactly depth k levels
D. A k-ary heap data structure

Answer: A

Q458. What is a circular buffer?

A. A buffer shaped like a circle visually
B. A buffer for circular linked list nodes
C. A double-ended buffer with two pointers
D. A fixed-size buffer that wraps around when full

Answer: D

Q459. What is an interval tree?

A. A tree for querying overlapping intervals efficiently
B. A tree storing time intervals as keys
C. A segment tree with interval-based updates
D. A tree with regular spacing between nodes

Answer: A

Q460. What is the time complexity of inserting into a trie for a word of length L?

A. O(n)
B. O(log n)
C. O(L)
D. O(1)

Answer: C

Q461. What is the KMP (Knuth-Morris-Pratt) algorithm?

A. A string compression algorithm for reducing text size
B. A hash-based search method for substring matching tasks
C. A pattern matcher using failure function to skip redundancy
D. A comparison-based sorting algorithm for string arrays

Answer: C

Q462. What is the time complexity of the KMP algorithm?

A. O(n + m)
B. O(m log n)
C. O(n × m)
D. O(n^2)

Answer: A

Q463. What is the Rabin-Karp algorithm?

A. A divide and conquer string matching algorithm
B. A string matching algorithm using rolling hashes
C. A string sorting algorithm by character values
D. A tree-based matching algorithm using tries

Answer: B

Q464. What is a rolling hash in Rabin-Karp?

A. A hash that updates efficiently as the window slides
B. A multi-level hash with cascading computation
C. A hash that changes randomly each call
D. A hash stored in a circular buffer structure

Answer: A

Q465. What is the longest palindromic substring problem?

A. Finding the longest string in a collection
B. Reversing a string to find palindromes
C. Finding the longest contiguous palindrome substring
D. Finding all palindromes in a given string

Answer: C

Q466. What is the time complexity of finding the longest palindromic substring using dynamic programming?

A. O(n^3)
B. O(n)
C. O(n^2)
D. O(n log n)

Answer: C

Q467. What is string hashing used for?

A. Efficiently comparing strings via hash values
B. Encrypting strings for secure storage
C. Compressing strings to save disk space
D. Sorting strings into alphabetical order

Answer: A

Q468. What is the Boyer-Moore algorithm?

A. A pattern matcher using bad character and good suffix rules
B. A brute force matching algorithm for strings
C. A string sorting algorithm by character frequency
D. A compression algorithm for text data files

Answer: A

Q469. What is the Z-array of a string?

A. An array where Z[i] is longest prefix match starting at i
B. The reverse of the original string as array
C. An array of character frequencies for the string
D. An array of zeros for initialization

Answer: A

Q470. How is the Z-algorithm used for pattern matching?

A. By hashing the pattern and comparing values
B. By using a suffix tree of the text string
C. By sorting the text characters first
D. By concatenating pattern and text then computing Z-array

Answer: D

Q471. What is an NP-complete problem?

A. A problem that cannot be solved at all
B. A problem with no known algorithm existing
C. A problem in NP as hard as any NP problem
D. A problem that runs in constant time always

Answer: C

Q472. What is an NP-hard problem?

A. A problem with a known polynomial time solution
B. A problem that is always in the class P
C. A problem that is easy to solve efficiently
D. A problem at least as hard as NP-complete problems

Answer: D

Q473. What is memoization's space-time tradeoff?

A. It uses no extra space at all
B. It uses extra space to avoid redundant computation
C. It increases time to save storage space
D. It reduces both space and time simultaneously

Answer: B

Q474. What is the relationship between P and NP?

A. P and NP are completely disjoint sets
B. P = NP has been proven true conclusively
C. NP "† P has been proven true conclusively
D. P "† NP, but whether P = NP is open

Answer: D

Q475. What is a polynomial reduction?

A. Reducing the degree of a polynomial expression
B. Transforming one problem into another in polynomial time
C. Reducing the time complexity of an algorithm
D. Simplifying an algorithm's code implementation

Answer: B

Q476. What is tail recursion optimization?

A. Adding a tail data structure to the recursion stack
B. Removing all recursive calls from the entire program
C. Making recursive call the last operation for iteration optimization
D. Making the recursion run in reverse backward order

Answer: C

Q477. What is loop unrolling?

A. Removing all loops from the source code base entirely
B. Making loops run in the reverse backward execution order
C. Executing multiple iterations per loop cycle to reduce overhead
D. Converting all iterative loops to recursion-based calls

Answer: C

Q478. What is cache-friendly code?

A. Code with no memory access at all
B. Code that runs entirely in cache memory
C. Code that uses caching library APIs
D. Code that maximizes CPU cache hit rates

Answer: D

Q479. What is the difference between best case, worst case, and average case complexity?

A. Best is minimum time, worst is maximum, average is expected
B. Worst case is purely theoretical and never practical
C. They are always the same value for any algorithm
D. Best case only applies to small input sizes

Answer: A

Q480. Why is O(n log n) considered efficient for sorting?

A. It works only for small input sizes efficiently
B. It uses the least memory of all algorithms
C. It matches the comparison-based sorting lower bound
D. It is the fastest possible for all sorting

Answer: C

Q481. What is the time complexity of T(n) = T(n-1) + n?

A. O(n)
B. O(2^n)
C. O(n log n)
D. O(n^2)

Answer: D

Q482. Which notation gives both upper and lower asymptotic bounds?

A. Small-o notation
B. Big-O notation
C. Theta notation
D. Omega notation

Answer: C

Q483. What is the space complexity of a recursive function with depth n?

A. O(n) due to call stack frames
B. O(1) constant space always
C. O(n^2) due to nested recursion
D. O(log n) due to stack halving

Answer: A

Q484. Which recurrence relation represents binary search?

A. T(n) = 2T(n/2) + n
B. T(n) = 2T(n-1) + 1
C. T(n) = T(n-1) + 1
D. T(n) = T(n/2) + 1

Answer: D

Q485. What is amortized analysis used for?

A. Measuring best-case time for a single run
B. Finding worst case of each single operation
C. Calculating space used by a single function
D. Computing the average cost over many operations

Answer: D

Q486. What is tail recursion?

A. Recursion where the call is the last operation
B. Recursion that uses an explicit stack structure
C. Recursion with no defined base case at all
D. Recursion with multiple simultaneous recursive calls

Answer: A

Q487. Which complexity is better for large inputs: O(n log n) or O(n^2)?

A. O(n^2) is always better for large inputs
B. O(n log n) is better for large inputs
C. Neither is better; they alternate in speed
D. Both have identical growth for large inputs

Answer: B

Q488. What is the difference between an algorithm and a heuristic?

A. Algorithms guarantee solutions; heuristics may not
B. Algorithms use hardware; heuristics use software
C. Heuristics guarantee correctness; algorithms do not
D. Heuristics are faster; algorithms are always slower

Answer: A

Q489. What does the substitution method prove in recurrence solving?

A. It proves by guessing and verifying via induction
B. It proves by converting recurrence to iteration
C. It proves by drawing a complete recursion tree
D. It proves by applying the Master Theorem formula

Answer: A

Q490. What is the recurrence for merge sort?

A. T(n) = 2T(n/2) + n
B. T(n) = T(n/2) + n
C. T(n) = 2T(n-1) + 1
D. T(n) = T(n-1) + n

Answer: A

Q491. What is the advantage of a circular queue over a linear queue?

A. It stores more elements in the same memory space
B. It allows random access to any element directly
C. It efficiently reuses vacated front space in array
D. It automatically sorts elements during insertion

Answer: C

Q492. How does a doubly linked list differ from a singly linked list?

A. Each node has pointers to both next and previous
B. It stores two data fields in each single node
C. It uses half the memory of a singly linked list
D. It can only be traversed in the forward direction

Answer: A

Q493. What is the time complexity of searching in an unsorted linked list?

A. O(1)
B. O(n log n)
C. O(log n)
D. O(n)

Answer: D

Q494. What is a sentinel node in a linked list?

A. A dummy node used to simplify boundary operations
B. A node that stores the list's total length value
C. A special node that stores metadata about types
D. A node that points to the middle of the list

Answer: A

Q495. How is a stack implemented using two queues?

A. Push to both queues; pop alternates between the two
B. Push to Q2 always; pop alternates and merges both queues
C. Push and pop both operate on Q1; Q2 stores overflow
D. Push to Q1 always; for pop, move Q1 to Q2 and dequeue

Answer: D

Q496. What is the time complexity of inserting at a specific position in an array?

A. O(n)
B. O(n^2)
C. O(log n)
D. O(1)

Answer: A

Q497. What problem does a circular linked list solve?

A. It automatically maintains elements in sorted order
B. It reduces memory usage by half compared to singly
C. It allows continuous traversal without null termination
D. It enables O(1) search for any element in list

Answer: C

Q498. Which data structure is best for implementing a browser's back button?

A. Deque
B. Array
C. Stack
D. Queue

Answer: C

Q499. What is the disadvantage of an array-based stack?

A. It requires O(n) time for every push operation
B. It cannot store primitive data type values at all
C. It has a fixed capacity that can cause overflow
D. It does not support push and pop operations well

Answer: C

Q500. What is the amortized cost of appending to a dynamic array?

A. O(n^2)
B. O(log n)
C. O(1)
D. O(n)

Answer: C

Q501. What is the maximum number of nodes in a binary tree of height h?

A. 2^(h+1) - 1
B. h^2 + 1
C. 2^h
D. 2h + 1

Answer: A

Q502. What is the difference between adjacency list and adjacency matrix?

A. Lists cannot represent weighted graphs but matrices can
B. Lists use O(V+E) space; matrices use O(V^2) space
C. Both always use exactly the same amount of space
D. Lists use O(V^2) space; matrices use O(V+E) space

Answer: B

Q503. What is a binary search tree (BST) property?

A. Nodes are arranged in ascending order left to right
B. Left child is less; right child is greater than parent
C. All nodes have exactly two children always
D. Every level must be completely filled with nodes

Answer: B

Q504. What is a full binary tree?

A. A tree where each node has exactly one child node
B. A tree where all leaf nodes are at the same level
C. A tree where every node has zero or two children
D. A tree with the maximum possible number of nodes

Answer: C

Q505. How is a graph different from a tree?

A. Trees have weighted edges; graphs have unweighted ones
B. Graphs must be connected; trees can be disconnected
C. Trees can have cycles; graphs cannot have any cycles
D. Graphs can have cycles; trees are always acyclic

Answer: D

Q506. What is a self-loop in a graph?

A. An edge shared between two separate graph components
B. A path that returns to the starting vertex via others
C. A cycle that passes through every vertex exactly once
D. An edge that connects a vertex to itself directly

Answer: D

Q507. What is the space complexity of an adjacency matrix for V vertices?

A. O(V)
B. O(V + E)
C. O(E)
D. O(V^2)

Answer: D

Q508. What is a balanced binary tree?

A. A tree where left and right subtree heights differ by at most one
B. A tree where all nodes have exactly two children always
C. A tree where all leaf nodes store the same data value
D. A tree that contains equal number of left and right nodes

Answer: A

Q509. What is a directed acyclic graph (DAG)?

A. An undirected graph with at least one cycle present
B. A tree with multiple root nodes and directed edges
C. A graph where all edges are bidirectional and acyclic
D. A directed graph with no cycles between any vertices

Answer: D

Q510. What is the depth of a node in a tree?

A. The maximum number of nodes at that particular level
B. The total number of nodes in the entire tree structure
C. The number of edges from the root to that specific node
D. The number of children that the node currently has

Answer: C

Q511. How does interpolation search improve over binary search?

A. It estimates position based on value distribution
B. It uses extra memory to cache previous search results
C. It works on unsorted arrays unlike binary search
D. It always guarantees O(1) time for any input data

Answer: A

Q512. What is the average-case time complexity of interpolation search on uniform data?

A. O(log n)
B. O(log log n)
C. O(sqrt(n))
D. O(n)

Answer: B

Q513. What is exponential search used for?

A. Finding range for binary search on unbounded arrays
B. Searching in a hash table with open addressing
C. Finding an element in an unsorted linked list
D. Searching in a balanced binary search tree structure

Answer: A

Q514. What is the time complexity of searching in a balanced BST?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(1)

Answer: B

Q515. What is ternary search?

A. A search that compares three elements at each single step
B. A search that uses three different search algorithms together
C. A search that requires three sorted arrays simultaneously
D. A search that divides the search space into three equal parts

Answer: D

Q516. What is jump search and what is its optimal block size?

A. Search every other element; block size is always two
B. Jump to random positions; block size is chosen randomly
C. Skip logarithmic blocks; block size is always log of n
D. Skip fixed blocks then linear search; block size is sqrt(n)

Answer: D

Q517. What is the space complexity of recursive binary search?

A. O(1)
B. O(n log n)
C. O(log n)
D. O(n)

Answer: C

Q518. How does Fibonacci search determine the split point?

A. It always splits the array exactly at the golden ratio
B. It uses Fibonacci numbers to divide the search space
C. It uses random Fibonacci-like sequences for splitting
D. It hashes elements using Fibonacci number as the key

Answer: B

Q519. What is the key advantage of binary search over linear search?

A. It can find multiple elements in a single search
B. It has O(log n) time versus O(n) on sorted data
C. It uses less memory than linear search algorithm
D. It works on any type of data without restrictions

Answer: B

Q520. What happens if binary search is applied to an unsorted array?

A. It will automatically sort the array first then search
B. It will always return the correct result anyway
C. It will convert to linear search as a fallback mode
D. It may skip the target and return incorrect result

Answer: D

Q521. What is the time complexity of merge sort in all cases?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(log n)

Answer: B

Q522. What is the worst-case pivot selection in quick sort?

A. Randomly selecting the pivot element each time
B. Always picking the median as pivot element
C. Always picking smallest or largest as pivot element
D. Picking the middle index element as the pivot

Answer: C

Q523. What is the space complexity of merge sort?

A. O(n)
B. O(log n)
C. O(n^2)
D. O(1)

Answer: A

Q524. How does counting sort achieve O(n+k) time complexity?

A. It counts occurrences of each value and computes positions
B. It divides elements into buckets based on digit values
C. It recursively splits and merges the input array halves
D. It compares pairs of elements to find their order

Answer: A

Q525. Why is quick sort preferred over merge sort in practice?

A. Quick sort has better cache performance and low overhead
B. Quick sort uses more memory which speeds up sorting
C. Quick sort has better worst-case time complexity
D. Quick sort is always stable unlike merge sort algorithm

Answer: A

Q526. What is the time complexity of heap sort?

A. O(log n)
B. O(n log n)
C. O(n^2)
D. O(n)

Answer: B

Q527. What is radix sort and when is it used?

A. A comparison sort that divides data by value ranges
B. A recursive sort that always selects median as pivot
C. A non-comparison sort processing digits from least significant
D. A sorting method that uses hashing for element placement

Answer: C

Q528. What is the key operation in the partition step of quick sort?

A. Merging two sorted halves into a single sorted array
B. Finding the minimum element in the unsorted portion
C. Rearranging elements around a chosen pivot value point
D. Comparing every pair of elements and swapping if needed

Answer: C

Q529. Is selection sort a stable algorithm?

A. It is stable only when elements are all distinct
B. It is stable only for arrays under a certain size
C. No, it is not stable due to long-range swapping
D. Yes, it is always stable for all possible inputs

Answer: C

Q530. What is shell sort?

A. A generalized insertion sort using decreasing gap sizes
B. A comparison sort based on the shell command utility
C. A variant of merge sort using shell-like structures
D. A sorting algorithm that uses a shell data structure

Answer: A

Q531. What is open addressing in hash tables?

A. Storing colliding elements in a separate overflow area
B. Using open-source hash functions for all key types
C. Finding alternative slots within the same table for collisions
D. Leaving table slots open and unoccupied permanently

Answer: C

Q532. What is linear probing?

A. Checking slots at quadratically increasing intervals
B. Checking sequential slots one by one after collision
C. Using a second hash function to find alternate slots
D. Probing in a linear linked list attached to each slot

Answer: B

Q533. What is the primary clustering problem in linear probing?

A. Occupied slots form long contiguous runs over time
B. Clusters form in the linked lists at each slot
C. All keys hash to the same single slot in table
D. Keys cluster at the end of the hash table only

Answer: A

Q534. How does quadratic probing reduce clustering?

A. It uses a secondary hash function for probing step
B. It probes at quadratically increasing intervals from hash
C. It uses random slot selection for each collision
D. It sorts the table after each insertion operation

Answer: B

Q535. What is double hashing?

A. Using the same hash function twice on the same key
B. Applying one hash function to the result of another
C. Hashing both the key and value separately for storage
D. Using two different hash functions for probe sequence

Answer: D

Q536. When should a hash table be resized?

A. When the first collision occurs in the table structure
B. When all slots in the table are completely occupied
C. When the load factor exceeds a certain threshold value
D. When the hash function starts producing negative values

Answer: C

Q537. What is the multiplication method of hashing?

A. Multiplying the key by the table size directly
B. Multiplying the hash by the load factor for index
C. Multiplying key by constant, extracting fractional part
D. Multiplying all keys together to get a single hash

Answer: C

Q538. What is rehashing?

A. Removing all elements and starting with an empty table
B. Rebuilding the hash table with a new size or hash function
C. Reversing the hash function to retrieve original keys
D. Recomputing hash values using the same table and function

Answer: B

Q539. What is the birthday paradox's relevance to hashing?

A. It explains why hash functions must use prime table sizes
B. It shows collisions are more likely than intuition suggests
C. It demonstrates that larger tables always prevent collisions
D. It proves that hash tables never experience collisions

Answer: B

Q540. What is the difference between a hash map and a hash set?

A. Hash maps are faster; hash sets use more memory space
B. Hash maps store key-value pairs; hash sets store keys only
C. Hash maps use chaining; hash sets use open addressing
D. Hash sets allow duplicates; hash maps do not allow them

Answer: B

Q541. What is an AVL tree?

A. A multi-way tree used primarily for database indexing
B. A self-balancing BST with height difference at most one
C. A BST where all leaves are at the exact same level
D. A tree where each node stores at most two keys only

Answer: B

Q542. What rotation is performed for a left-left imbalance in an AVL tree?

A. Left rotation
B. Right-left rotation
C. Right rotation
D. Left-right rotation

Answer: C

Q543. What is a red-black tree?

A. A binary tree that alternates red and black at each level
B. A BST where nodes are colored red or black with rules
C. A tree using red and black to indicate data priority
D. A tree where red nodes are leaves and black are internal

Answer: B

Q544. What is the maximum height of an AVL tree with n nodes?

A. O(n^2)
B. O(n)
C. O(log n)
D. O(sqrt(n))

Answer: C

Q545. How does deletion work in a BST when the node has two children?

A. Simply remove the node and leave children as orphan nodes
B. Delete both children and reconnect their subtrees together
C. Swap with the root node and then delete the root node
D. Replace with in-order successor or predecessor, then delete it

Answer: D

Q546. What is the purpose of a B-tree?

A. Minimizing disk I/O for database and file system access
B. Compressing tree data for network data transmission
C. Storing data in a binary format for fast processing
D. Balancing workload across multiple processor threads

Answer: A

Q547. What is a segment tree used for?

A. Segmenting data into equal-sized portions for sorting
B. Creating visual segments for graphical tree rendering
C. Answering range queries and updates on an array efficiently
D. Storing segments of text for string matching algorithms

Answer: C

Q548. What is a binary heap?

A. A binary tree where all nodes have the exact same value
B. A sorted binary tree with elements in ascending order
C. A complete binary tree satisfying the heap property order
D. A binary tree where leaves store minimum values always

Answer: C

Q549. What is the time complexity of building a heap from an array?

A. O(n log n)
B. O(n)
C. O(n^2)
D. O(log n)

Answer: B

Q550. How is a node inserted into an AVL tree?

A. Insert as in BST, then rotate to fix any imbalance
B. Always insert at the root and push other nodes down
C. Insert at the deepest level regardless of key value
D. Insert at a random position and then sort the tree

Answer: A

Q551. What does Dijkstra's algorithm find?

A. All possible paths between two specific graph vertices
B. Shortest paths from a single source to all vertices
C. Minimum spanning tree of a weighted graph structure
D. Maximum flow through a network with capacity constraints

Answer: B

Q552. What is the time complexity of Dijkstra's algorithm with a binary heap?

A. O(V + E)
B. O(E * V)
C. O((V+E) log V)
D. O(V^2)

Answer: C

Q553. What does Kruskal's algorithm compute?

A. Minimum spanning tree using edge sorting approach
B. Shortest path between two specific vertices only
C. All connected components of an undirected graph
D. Topological ordering of a directed acyclic graph

Answer: A

Q554. What does Prim's algorithm compute?

A. Minimum spanning tree by growing from a vertex
B. Shortest paths from all sources to all destinations
C. Strongly connected components of a directed graph
D. Maximum matching in a bipartite graph structure

Answer: A

Q555. What is topological sorting?

A. Arranging vertices in a circle for visual graph display
B. Sorting edges by weight for minimum spanning tree building
C. Sorting graph vertices by their degree in ascending order
D. Linear ordering of vertices so each edge goes forward

Answer: D

Q556. Can topological sort be performed on a graph with cycles?

A. Yes, but the result may not be unique or deterministic
B. No, topological sort requires a directed acyclic graph
C. No, but it can be performed on undirected cyclic graph
D. Yes, cycles are handled by ignoring back edges only

Answer: B

Q557. What limitation does Dijkstra's algorithm have?

A. It only works on undirected graphs not directed ones
B. It requires the graph to be a complete graph always
C. It cannot handle graphs with more than 100 vertices
D. It fails on graphs with negative weight edge values

Answer: D

Q558. What is the Union-Find data structure used for?

A. Storing vertex and edge data in compressed format
B. Searching for elements in a sorted graph structure
C. Finding unions and intersections of sorted arrays
D. Tracking connected components and detecting cycles

Answer: D

Q559. What is the Bellman-Ford algorithm used for?

A. Finding maximum flow in a network with capacities
B. Finding minimum spanning trees in undirected graphs
C. Finding shortest paths even with negative edge weights
D. Finding all bridges and articulation points in graphs

Answer: C

Q560. What is the time complexity of the Floyd-Warshall algorithm?

A. O(E^2)
B. O(V^3)
C. O(V^2)
D. O(V * E)

Answer: B

Q561. What is the difference between memoization and tabulation?

A. Memoization uses tables; tabulation uses recursive calls
B. Both are identical approaches with different naming only
C. Memoization is bottom-up; tabulation is top-down approach
D. Memoization is top-down; tabulation is bottom-up approach

Answer: D

Q562. What are the two key properties for dynamic programming applicability?

A. Optimal substructure and overlapping subproblems present
B. Greedy choice property and matroid structure correctness
C. Logarithmic depth and polynomial branching factor only
D. Divide and conquer with non-overlapping subproblem cases

Answer: A

Q563. What is the time complexity of the 0/1 knapsack problem using DP?

A. O(n * W)
B. O(n log n)
C. O(n^2)
D. O(2^n)

Answer: A

Q564. What is the longest common subsequence (LCS) problem?

A. Finding the longest substring that appears in one string
B. Finding the longest path between two nodes in a graph
C. Finding the longest sorted subsequence in one sequence
D. Finding the longest subsequence common to two sequences

Answer: D

Q565. What is the activity selection problem?

A. Selecting activities based on their priority level values
B. Selecting maximum non-overlapping activities by finish time
C. Selecting activities that require the minimum total cost
D. Selecting the activity with the longest duration possible

Answer: B

Q566. What is the rod cutting problem?

A. Finding the minimum number of cuts to divide a rod evenly
B. Cutting a rod into equal pieces for uniform distribution
C. Cutting a rod into pieces to maximize total sale revenue
D. Cutting rods from a sheet to minimize material waste cost

Answer: C

Q567. How does branch and bound differ from backtracking?

A. Branch and bound uses bounds to prune unpromising branches
B. Backtracking uses bounds; branch and bound does not use any
C. Branch and bound only works on graph problems specifically
D. They are identical algorithms with different names only

Answer: A

Q568. What is the coin change problem using dynamic programming?

A. Finding minimum coins needed to make a given amount value
B. Sorting coins by value for efficient change distribution
C. Finding the total value of all coins combined together
D. Counting the total number of coins in a given collection

Answer: A

Q569. What is the fractional knapsack problem?

A. A problem of dividing a knapsack into fractional parts
B. A knapsack problem where items must be taken completely
C. A knapsack problem where items can be taken partially
D. A problem of distributing items into fractional groups

Answer: C

Q570. What is the time complexity of the LCS problem using DP?

A. O(m + n)
B. O(m * n)
C. O(2^(m+n))
D. O(m^2 + n^2)

Answer: B

Q571. What is a Fibonacci heap?

A. A min-heap that stores only Fibonacci numbers as values
B. A binary heap where the size follows Fibonacci sequence
C. A heap with amortized O(1) insert and decrease-key time
D. A heap based on Fibonacci numbers for index calculation

Answer: C

Q572. What is a suffix array?

A. An array storing the frequency of each character in string
B. An array of all prefixes of a string sorted in order
C. An array of all suffixes of a string sorted lexicographically
D. An array of indices where a specific pattern is found

Answer: C

Q573. What is path compression in Union-Find?

A. Compressing file paths for efficient storage in memory
B. Shortening the physical path between two graph vertices
C. Making each node point directly to the root during find
D. Compressing data along the path in a tree structure

Answer: C

Q574. What is union by rank in Union-Find?

A. Attaching the shorter tree under the root of taller tree
B. Ranking all elements before performing any union operation
C. Unioning sets based on alphabetical rank of elements
D. Sorting elements by rank before adding to disjoint sets

Answer: A

Q575. What is a trie's space complexity for n strings of average length m?

A. O(m)
B. O(n^2)
C. O(n * m)
D. O(n)

Answer: C

Q576. What is an interval tree?

A. A binary tree with nodes at regular interval positions
B. A tree that stores intervals for overlap and point queries
C. A tree where node values differ by a constant interval
D. A tree that stores time intervals for scheduling queries

Answer: B

Q577. What is the time complexity of decrease-key in a Fibonacci heap?

A. O(n)
B. O(n log n)
C. O(1) amortized
D. O(log n)

Answer: C

Q578. What is a k-d tree used for?

A. Dividing a tree into k balanced subtrees for search
B. Storing k different data types in a single tree node
C. Organizing points in k-dimensional space for queries
D. Creating k copies of a binary tree for redundancy

Answer: C

Q579. What is the advantage of a balanced BST over a hash table?

A. It provides faster average-case lookup than hashing does
B. It has simpler implementation than hash table structures
C. It maintains elements in sorted order for range queries
D. It uses less memory than a hash table for same elements

Answer: C

Q580. What is a binomial heap?

A. A binary heap with binomial distribution of element values
B. A heap that performs binomial search for extract min
C. A collection of binomial trees satisfying heap property
D. A heap using binomial coefficients for priority values

Answer: C

Q581. What is the KMP algorithm?

A. A string sorting algorithm using key-median partitioning
B. A pattern matching algorithm using a prefix failure table
C. A string compression algorithm using key-mapped patterns
D. A substring search using kernel-mapped positioning index

Answer: B

Q582. What is the time complexity of the KMP algorithm?

A. O(n^2)
B. O(m^2)
C. O(n * m)
D. O(n + m)

Answer: D

Q583. What is the Rabin-Karp algorithm?

A. A text compression algorithm using repeated substrings
B. A string sorting algorithm based on radix computation
C. A pattern matching algorithm using rolling hash function
D. A string encryption algorithm using the Karp cipher key

Answer: C

Q584. What is the failure function in KMP?

A. A table mapping each character to its failure position
B. A table storing longest proper prefix that is also suffix
C. A function that detects when the algorithm has failed
D. A function that counts the number of failed comparisons

Answer: B

Q585. What is a rolling hash?

A. A hash that changes completely for each new substring
B. A hash that rotates character positions in the input string
C. A hash computed by rolling through all possible hash functions
D. A hash updated incrementally by adding and removing characters

Answer: D

Q586. What is the longest palindromic substring problem?

A. Finding the longest string that contains a palindrome inside
B. Finding the longest contiguous palindrome within a string
C. Generating the longest possible palindrome from characters
D. Finding all palindromes in a string and returning longest

Answer: B

Q587. What is the Boyer-Moore algorithm's key idea?

A. Comparing only the first and last characters of pattern
B. Comparing every other character for faster matching speed
C. Comparing pattern from right to left with shift heuristics
D. Comparing characters from left to right systematically

Answer: C

Q588. What is the Z-algorithm used for in string processing?

A. Sorting strings in reverse lexicographic Z to A order
B. Compressing strings using Z-encoding for smaller storage
C. Encrypting strings using the Z-cipher transformation
D. Computing Z-array for pattern matching in linear time

Answer: D

Q589. What is string hashing used for?

A. Sorting strings in hash-based order for faster access
B. Compressing strings to reduce storage space requirements
C. Efficiently comparing strings and finding pattern matches
D. Encrypting strings for secure network communication only

Answer: C

Q590. What is the time complexity of checking if a string is a palindrome?

A. O(n log n)
B. O(1)
C. O(n^2)
D. O(n)

Answer: D

Q591. What is the difference between best, average, and worst case?

A. They describe input scenarios giving minimum, expected, and maximum cost
B. They represent different programming languages and their speeds
C. They measure performance for small, medium, and large inputs
D. They describe memory, disk, and network usage respectively

Answer: A

Q592. What is the time complexity of an algorithm with three sequential loops of O(n)?

A. O(3n)
B. O(n)
C. O(n^2)
D. O(n^3)

Answer: B

Q593. What is the space-time tradeoff?

A. Trading code readability for faster execution time always
B. Trading algorithm correctness for improved speed results
C. Trading disk space for network bandwidth in algorithms
D. Trading space complexity for time complexity or vice versa

Answer: D

Q594. What is the complexity of matrix multiplication of two n x n matrices?

A. O(n^2)
B. O(n^3)
C. O(n log n)
D. O(n)

Answer: B

Q595. What is loop unrolling as an optimization technique?

A. Converting recursive loops into iterative ones for speed
B. Removing all loops from the algorithm entirely always
C. Unrolling nested loops into a single flat loop structure
D. Reducing loop overhead by executing multiple iterations per cycle

Answer: D

Q596. What is memoization's effect on the Fibonacci sequence computation?

A. It changes complexity from O(n^2) to O(n log n) time
B. It has no effect on the time complexity of Fibonacci
C. It changes complexity from O(n) to O(1) constant time
D. It changes complexity from O(2^n) to O(n) linear time

Answer: D

Q597. What is the significance of polynomial vs exponential complexity?

A. Polynomial algorithms are considered efficient; exponential are not
B. Exponential algorithms are always faster than polynomial ones
C. Both are equally efficient for all practical input sizes
D. Polynomial algorithms are impractical; exponential are efficient

Answer: A

Q598. What is cache-friendly code?

A. Code that disables caching for more predictable performance
B. Code that accesses memory in patterns that maximize cache hits
C. Code that uses a custom cache data structure for lookups
D. Code that stores all data in the CPU cache permanently

Answer: B

Q599. What is the complexity class P?

A. Problems solvable in polynomial time by deterministic machine
B. Problems that have no known solution in any time bound
C. Problems solvable in polynomial time by nondeterministic machine
D. Problems solvable only in exponential time by any machine

Answer: A

Q600. What is tail call optimization?

A. Removing the last call in a function for faster returns
B. Optimizing function calls at the tail of a linked list
C. Optimizing the last element of an array for faster access
D. Reusing the current stack frame for a tail recursive call

Answer: D

Q601. What is the time complexity of T(n) = 4T(n/2) + n using the Master Theorem?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(n^2 log n)

Answer: C

Q602. What is the difference between recursion and iteration in terms of space usage?

A. Recursion always uses less space
B. Iteration typically uses less space because it doesn't require a call stack
C. They always use the same amount of space
D. Iteration always uses more space due to loop variables

Answer: B

Q603. What is the solution to the recurrence T(n) = T(n/2) + O(1)?

A. O(n)
B. O(n log n)
C. O(log n)
D. O(1)

Answer: C

Q604. If f(n) = 5n^3 + 2n^2 + 100n + 42, what is the Big-O classification?

A. O(n^2)
B. O(5n^3)
C. O(n^3)
D. O(n^4)

Answer: C

Q605. What does it mean if an algorithm is said to run in polynomial time?

A. Its running time is O(2^n)
B. Its running time can be expressed as O(n^k) for some constant k
C. Its running time is always O(n)
D. It cannot be analyzed using Big-O notation

Answer: B

Q606. What is the recursion tree method used for?

A. Sorting elements in a tree
B. Visualizing the work done at each level of a recurrence to find the total cost
C. Converting iterative code to recursive code
D. Building binary search trees

Answer: B

Q607. Which of the following is NOT a valid step in the substitution method for solving recurrences?

A. Guess the form of the solution
B. Use mathematical induction to prove the guess
C. Assume the solution holds for smaller values
D. Differentiate the recurrence with respect to n

Answer: D

Q608. What is the space complexity of an algorithm that creates a new array of size n within a loop that runs n times?

A. O(n)
B. O(n^2)
C. O(log n)
D. O(1)

Answer: A

Q609. What does little-o notation o(f(n)) signify?

A. The growth rate is exactly f(n)
B. The growth rate is strictly less than f(n) asymptotically
C. The growth rate is at least f(n)
D. The growth rate is strictly greater than f(n)

Answer: B

Q610. How does the accounting method of amortized analysis work?

A. It counts the total number of function calls
B. It assigns different charges to operations so cheap operations subsidize expensive ones
C. It measures the actual runtime on a specific machine
D. It ignores expensive operations in the analysis

Answer: B

Q611. How can you implement a min-queue that supports enqueue, dequeue, and getMin all in amortized O(1)?

A. Use a single array with sorting
B. Use two stacks, each augmented to track minimums
C. Use a linked list sorted by value
D. Use a binary search tree

Answer: B

Q612. What is the time complexity of inserting an element at a specific position i in an array of size n?

A. O(1)
B. O(i)
C. O(n - i)
D. O(n)

Answer: D

Q613. What is the advantage of using a circular buffer over a linear array for a queue?

A. Circular buffers use less memory
B. They avoid the need to shift elements after dequeue operations
C. They support random access faster
D. They can store more elements

Answer: B

Q614. What happens when a dynamic array doubles its size upon reaching capacity?

A. Elements are sorted into the new array
B. A new array of double the size is allocated and all elements are copied over
C. The old array is extended in place without copying
D. Only the new elements are placed in the new array

Answer: B

Q615. How can you use a stack to check if parentheses in an expression are balanced?

A. Count opening and closing parentheses
B. Push opening brackets onto the stack and pop when a matching closing bracket is found
C. Sort the brackets and compare halves
D. Use two stacks, one for each type

Answer: B

Q616. What is the time complexity of reversing a singly linked list using iteration?

A. O(n^2)
B. O(n log n)
C. O(1)
D. O(n)

Answer: D

Q617. What is a priority queue and how does it differ from a regular queue?

A. A priority queue is the same as a regular queue but faster
B. Elements are dequeued based on priority rather than insertion order
C. A priority queue only stores integers
D. A priority queue reverses the order of insertion

Answer: B

Q618. What is the space overhead of a singly linked list compared to an array for storing n integers?

A. No extra space
B. O(1) extra space
C. O(n) extra space for storing next pointers
D. O(n^2) extra space

Answer: C

Q619. How does a deque (double-ended queue) differ from a standard queue?

A. A deque is slower than a queue
B. A deque allows insertion and deletion at both the front and rear
C. A deque only allows deletion from both ends
D. A deque can only store two elements

Answer: B

Q620. What is the time complexity of finding the middle element of a singly linked list in one pass?

A. O(n^2)
B. O(n)
C. O(log n)
D. O(1)

Answer: B

Q621. What is the time complexity of inserting an element into a binary search tree in the average case?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: B

Q622. What is the relationship between the number of edges and vertices in a complete graph K_n?

A. n edges
B. n(n-1)/2 edges
C. n^2 edges
D. 2n edges

Answer: B

Q623. What is the advantage of an adjacency list representation over an adjacency matrix for sparse graphs?

A. Adjacency lists use O(V^2) space regardless of edges
B. Adjacency lists use O(V + E) space, which is much less than O(V^2) for sparse graphs
C. Adjacency lists allow O(1) edge lookup
D. Adjacency lists are always slower

Answer: B

Q624. What is the time complexity of deleting the minimum element from a min-heap with n elements?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: B

Q625. What is a perfect binary tree?

A. A tree where every node has exactly one child
B. A tree where all internal nodes have two children and all leaves are at the same depth
C. A tree where nodes are sorted
D. A tree with the minimum possible height for n nodes

Answer: B

Q626. What is the difference between a tree and a graph?

A. Trees can have cycles but graphs cannot
B. A tree is a connected acyclic graph, while a graph can have cycles and disconnected components
C. Graphs always have fewer edges than trees
D. Trees can only store numbers

Answer: B

Q627. How is a binary heap typically stored in memory?

A. As a linked list of nodes
B. As a contiguous array using index formulas for parent and children
C. As a hash table
D. As a 2D matrix

Answer: B

Q628. What is the in-degree of a vertex in a directed graph?

A. The number of edges going out from the vertex
B. The total number of edges in the graph
C. The number of edges coming into the vertex
D. The number of self-loops at the vertex

Answer: C

Q629. What is the sum of degrees of all vertices in an undirected graph with E edges?

A. E
B. 2E
C. E^2
D. V + E

Answer: B

Q630. What is a degenerate (skewed) binary tree?

A. A tree where every node has exactly two children
B. A tree where every node has at most one child, resembling a linked list
C. A tree with all leaves at the same level
D. A tree stored using an adjacency matrix

Answer: B

Q631. What is the time complexity of jump search on a sorted array of n elements?

A. O(n)
B. O(log n)
C. O(" n)
D. O(n log n)

Answer: C

Q632. How does exponential search work?

A. It searches all elements exponentially faster
B. It finds the range where the target lies by doubling the index, then applies binary search within that range
C. It multiplies each element by 2
D. It sorts the array first, then searches linearly

Answer: B

Q633. What is the worst-case time complexity of ternary search?

A. O(n)
B. O(log3 n), which is O(log n)
C. O(" n)
D. O(n log n)

Answer: B

Q634. What is the key condition that makes interpolation search perform better than binary search?

A. The array must be in descending order
B. The data must be uniformly distributed
C. The array size must be a power of 2
D. The array must contain only integers

Answer: B

Q635. How can binary search be modified to find the leftmost (first) occurrence of a duplicate element?

A. Return the first match found
B. When the target is found, continue searching in the left half instead of returning immediately
C. Sort the array again after finding the element
D. Use linear search after finding any occurrence

Answer: B

Q636. What is the Fibonacci search technique?

A. Searching for Fibonacci numbers in an array
B. A search method that divides the array using Fibonacci numbers instead of halving
C. A recursive search that calls itself twice
D. A hash-based search technique

Answer: B

Q637. What is the time complexity of searching in an unsorted hash table versus a sorted array using binary search?

A. Hash table: O(n), Binary search: O(log n)
B. Hash table: O(1) average, Binary search: O(log n)
C. Both are O(log n)
D. Both are O(1)

Answer: B

Q638. What is a search tree and what operation does it optimize?

A. A tree used for sorting elements
B. A tree data structure that organizes data to optimize search operations to O(log n)
C. A tree that stores only search queries
D. A tree with a fixed number of nodes

Answer: B

Q639. How does binary search on answer (parametric search) work?

A. It searches for the answer in a sorted array of answers
B. It binary searches over the answer space using a feasibility check function
C. It sorts the answers before searching
D. It applies binary search twice

Answer: B

Q640. What is the primary advantage of exponential search over binary search for searching in an unbounded or infinite list?

A. It uses less memory
B. It does not require knowing the array size beforehand
C. It is always faster
D. It does not require the array to be sorted

Answer: B

Q641. What is the recurrence relation for the time complexity of merge sort?

A. $T(n) = T(n-1) + n$
B. $T(n) = 2T(n/2) + O(n)$
C. $T(n) = T(n/2) + O(1)$
D. $T(n) = 4T(n/2) + O(n)$

Answer: B

Q642. Why is quicksort generally faster than merge sort in practice despite having the same average complexity?

A. Quicksort has a lower worst-case complexity
B. Quicksort has better cache performance and smaller constant factors due to in-place operations
C. Quicksort uses less stack space
D. Quicksort is always stable

Answer: B

Q643. What pivot selection strategy helps avoid quicksort's worst case?

A. Always picking the first element
B. Always picking the last element
C. Choosing the median of the first, middle, and last elements
D. Picking the largest element

Answer: C

Q644. What is counting sort's key limitation?

A. It cannot sort integers
B. It requires O(k) extra space where k is the range of input values, making it impractical for large ranges
C. It is always slower than quicksort
D. It cannot sort arrays larger than 100 elements

Answer: B

Q645. How does radix sort achieve linear time complexity?

A. By using comparison-based sorting internally
B. By sorting digit by digit from least significant to most significant using a stable sort like counting sort
C. By randomly partitioning the array
D. By using a binary search tree

Answer: B

Q646. What is the time complexity of heap sort and is it stable?

A. O(n^2) and stable
B. O(n log n) and stable
C. O(n log n) and not stable
D. O(n) and stable

Answer: C

Q647. What is shell sort and how does it improve insertion sort?

A. It sorts only the outer shell of the array
B. It insertion-sorts elements that are far apart first, reducing the gap progressively
C. It uses a shell data structure
D. It sorts the array into a shell shape

Answer: B

Q648. Is merge sort preferred over quicksort for sorting linked lists? Why?

A. No, quicksort is always preferred
B. Yes, because merge sort does not require random access and linked list merge is O(1) extra space
C. No, because merge sort is slower on linked lists
D. Yes, but only for doubly linked lists

Answer: B

Q649. What is the purpose of the partition step in quicksort?

A. To merge two sorted subarrays
B. To place the pivot in its correct sorted position with smaller elements before it and larger after
C. To find the median of the array
D. To count the number of elements

Answer: B

Q650. What is the difference between LSD (Least Significant Digit) and MSD (Most Significant Digit) radix sort?

A. LSD processes from the least significant digit and requires a stable sort; MSD processes from the most significant digit and can use recursion
B. They are identical algorithms
C. LSD is faster for all inputs
D. MSD cannot sort strings

Answer: A

Q651. How does quadratic probing resolve collisions?

A. By probing at positions h(k)+1, h(k)+2, h(k)+3, ...
B. By probing at positions h(k)+1^2, h(k)+2^2, h(k)+3^2, ...
C. By using a second hash function
D. By rehashing the entire table

Answer: B

Q652. What is secondary clustering in hash tables?

A. When all elements hash to the first slot
B. When keys that hash to the same slot follow the same probe sequence
C. When the hash table is divided into two clusters
D. When the load factor exceeds 2

Answer: B

Q653. What is the expected number of probes for a successful search in a hash table with open addressing at load factor ; ð

A. 1/(1-; •
B. (1/; ’ ¢ Æâƒ òƒ Ó±))
C. ;
D. log(; •

Answer: B

Q654. Why is it recommended to use a prime number for hash table size when using the division method?

A. Prime numbers are faster to compute
B. It helps distribute keys more uniformly by reducing patterns from common factors
C. Prime numbers use less memory
D. It is not recommended; powers of 2 are always better

Answer: B

Q655. What is the typical threshold load factor at which a hash table should be resized?

A. 0.25
B. 0.5
C. 0.75
D. 1.0

Answer: C

Q656. What is the time complexity of rehashing a hash table with n elements?

A. O(1)
B. O(log n)
C. O(n)
D. O(n^2)

Answer: C

Q657. How does double hashing avoid clustering?

A. It uses two hash tables
B. It uses a second hash function to determine the probe step size, making each key's probe sequence different
C. It hashes the value twice before inserting
D. It doubles the table size on each collision

Answer: B

Q658. What is the advantage of open addressing over chaining?

A. Open addressing is always faster
B. Open addressing has better cache performance because all data is stored in the table array itself
C. Open addressing handles higher load factors better
D. Open addressing never needs resizing

Answer: B

Q659. After approximately how many insertions into a hash table with m slots does the probability of at least one collision exceed 50%?

A. m insertions
B. m/2 insertions
C. About " (2m) insertions
D. About log(m) insertions

Answer: C

Q660. What is the difference between separate chaining and open addressing?

A. Separate chaining uses linked lists at each slot while open addressing probes for the next empty slot
B. They are the same technique with different names
C. Open addressing uses linked lists while chaining uses arrays
D. Separate chaining never has collisions

Answer: A

Q661. What is the balance factor in an AVL tree?

A. The number of nodes in the tree
B. The difference in heights of the left and right subtrees of a node
C. The total height of the tree
D. The number of rotations performed

Answer: B

Q662. When is a right-left (RL) rotation performed in an AVL tree?

A. When a node is inserted in the left subtree of the left child
B. When a node is inserted in the right subtree of the left child
C. When a node is inserted in the left subtree of the right child
D. When a node is inserted in the right subtree of the right child

Answer: C

Q663. How is a node with two children deleted from a BST?

A. Simply remove the node
B. Replace it with its in-order successor or predecessor, then delete that replacement node
C. Swap it with the root and remove the root
D. Delete both children first, then remove the node

Answer: B

Q664. What is Morris traversal and what is its space complexity?

A. A traversal using two stacks with O(n) space
B. An in-order traversal using threading to achieve O(1) space without a stack or recursion
C. A traversal that visits nodes randomly in O(n) space
D. A BFS traversal using O(n) space

Answer: B

Q665. What is a Red-Black tree's maximum height in terms of the number of nodes n?

A. Exactly log2(n)
B. At most 2 * log2(n + 1)
C. Exactly n
D. At most n/2

Answer: B

Q666. What is a segment tree's time complexity for a range sum query after O(n) construction?

A. O(n)
B. O(log n)
C. O(1)
D. O(" n)

Answer: B

Q667. How can you verify if a given binary tree is a valid BST?

A. Check if each node is greater than its children
B. Perform in-order traversal and verify the result is strictly increasing
C. Check if the root is the median value
D. Count the number of nodes and compare with height

Answer: B

Q668. Why is bottom-up heap construction O(n) rather than O(n log n)?

A. Because it skips half the elements
B. Because most nodes are near the leaves and require very few sift-down operations, forming a convergent series
C. Because it uses insertion sort internally
D. Because it only processes the root node

Answer: B

Q669. What is a B-tree primarily used for?

A. In-memory sorting
B. Maintaining sorted data in database and file systems with minimal disk reads
C. Network routing
D. String pattern matching

Answer: B

Q670. What is the LCA (Lowest Common Ancestor) of two nodes in a binary tree?

A. The root of the tree
B. The deepest node that is an ancestor of both given nodes
C. The parent of the deeper node
D. The node at the midpoint between the two nodes

Answer: B

Q671. What is the time complexity of Dijkstra's algorithm using a Fibonacci heap?

A. O(V^2)
B. O(V log V + E)
C. O(E log V)
D. O(V * E)

Answer: B

Q672. How does Kruskal's algorithm find a minimum spanning tree?

A. By starting from a vertex and adding the cheapest edge to a neighbor
B. By sorting all edges by weight and adding them if they don't create a cycle, using Union-Find
C. By finding the shortest path from source to all vertices
D. By removing the most expensive edges

Answer: B

Q673. What is topological sorting and on which type of graph is it performed?

A. Sorting vertices by degree on any graph
B. A linear ordering of vertices such that for every directed edge (u,v), u comes before v; only on DAGs
C. Sorting edges by weight in an undirected graph
D. Ordering vertices alphabetically

Answer: B

Q674. How does the Bellman-Ford algorithm differ from Dijkstra's algorithm?

A. Bellman-Ford is always faster
B. Bellman-Ford can handle negative edge weights and detect negative cycles, while Dijkstra cannot
C. Dijkstra handles negative weights but Bellman-Ford does not
D. They produce different results on the same graph

Answer: B

Q675. What is the time complexity of Prim's algorithm using a binary heap?

A. O(V^2)
B. O(E log V)
C. O(V * E)
D. O(V + E)

Answer: B

Q676. How can you detect a cycle in an undirected graph using DFS?

A. Check if any vertex has degree > 2
B. If DFS visits an already-visited vertex that is not the parent of the current vertex, a cycle exists
C. Count the number of edges; a cycle exists if E > V
D. Use topological sort

Answer: B

Q677. What does the Union-Find path compression optimization do?

A. It compresses the data stored at each node
B. It flattens the tree by making every node point directly to the root during find operations
C. It removes unused paths from the graph
D. It compresses the graph into fewer vertices

Answer: B

Q678. What is the Floyd-Warshall algorithm used for?

A. Finding the minimum spanning tree
B. Finding shortest paths between all pairs of vertices
C. Topological sorting
D. Detecting bipartite graphs

Answer: B

Q679. What is the time complexity of finding connected components in an undirected graph?

A. O(V^2)
B. O(V + E)
C. O(V * E)
D. O(E^2)

Answer: B

Q680. What is a back edge in DFS and what does it indicate?

A. An edge going from a node to its parent
B. An edge from a node to an ancestor in the DFS tree, indicating a cycle in directed graphs
C. An edge that was removed during DFS
D. An edge connecting two different components

Answer: B

Q681. What is the time complexity of the dynamic programming solution for the Longest Common Subsequence?

A. O(n)
B. O(n * m) where n and m are the lengths of the two sequences
C. O(n^3)
D. O(2^n)

Answer: B

Q682. Why does the greedy approach fail for the 0/1 Knapsack problem but work for the fractional Knapsack?

A. The greedy approach is always incorrect
B. In 0/1 Knapsack, items cannot be divided, so a locally optimal choice may block a better global solution
C. The fractional Knapsack has fewer items
D. Both can be solved greedily

Answer: B

Q683. How does the N-Queens backtracking solution work?

A. It places all queens randomly and checks if the placement is valid
B. It places queens one row at a time, backtracking when a placement conflicts with existing queens
C. It sorts the board positions and assigns queens greedily
D. It uses dynamic programming to find all placements

Answer: B

Q684. What is the Longest Increasing Subsequence (LIS) problem's optimal time complexity?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(2^n)

Answer: B

Q685. What is Huffman coding and which algorithm design paradigm does it use?

A. A sorting algorithm using divide and conquer
B. An optimal prefix-free encoding algorithm using the greedy approach
C. A string matching algorithm using dynamic programming
D. A graph algorithm using backtracking

Answer: B

Q686. What is the difference between the 0/1 Knapsack and the fractional Knapsack problem?

A. 0/1 Knapsack allows fractions while fractional does not
B. In 0/1 Knapsack items are taken whole or not at all; in fractional Knapsack items can be divided
C. They are the same problem
D. Fractional Knapsack is always harder

Answer: B

Q687. What is the rod cutting problem in dynamic programming?

A. Cutting a graph into components
B. Finding the maximum revenue by cutting a rod of length n into pieces with given prices for each length
C. Dividing an array into equal parts
D. Cutting a string at specific positions

Answer: B

Q688. What is branch and bound and how does it differ from backtracking?

A. They are the same algorithm
B. Branch and bound uses bounds (upper/lower) to prune branches that cannot yield better solutions than the current best
C. Branch and bound only works on trees
D. Backtracking uses bounds while branch and bound does not

Answer: B

Q689. What is the space optimization technique for the 0/1 Knapsack DP solution?

A. Use a 3D array instead of 2D
B. Use only two 1D arrays (or one array traversed in reverse) since each row only depends on the previous row
C. Use a hash table
D. Use a linked list

Answer: B

Q690. What is the activity selection problem and which approach solves it optimally?

A. Selecting the minimum number of activities; solved by DP
B. Selecting the maximum number of non-overlapping activities; solved by a greedy algorithm that sorts by finish time
C. Selecting activities with maximum total weight; solved by brute force
D. Assigning activities to workers; solved by backtracking

Answer: B

Q691. What is the amortized time complexity of Union-Find operations with both path compression and union by rank?

A. O(1)
B. O(log n)
C. O(; †â’Â v†W&R ± is the inverse Ackermann function
D. O(n)

Answer: C

Q692. What is a Fibonacci heap's key advantage over a binary heap?

A. Simpler implementation
B. O(1) amortized decrease-key operation instead of O(log n)
C. Less memory usage
D. Faster extract-min

Answer: B

Q693. What is a suffix array and what problem does it help solve?

A. An array of suffix sums for range queries
B. A sorted array of all suffixes of a string, enabling efficient pattern matching and string analysis
C. An array storing the suffixes of Fibonacci numbers
D. A compressed version of a string

Answer: B

Q694. What is a sparse table used for?

A. Storing sparse matrices
B. Answering static range minimum/maximum queries in O(1) after O(n log n) preprocessing
C. Building sparse graphs
D. Compressing data

Answer: B

Q695. How does a k-d tree determine which dimension to split on at each level?

A. It always splits on the first dimension
B. It cycles through the dimensions at each level (depth mod k)
C. It randomly selects a dimension
D. It splits on the dimension with the most variance

Answer: B

Q696. What is the time complexity of merge operation in a binomial heap?

A. O(n)
B. O(log n)
C. O(1)
D. O(n log n)

Answer: B

Q697. What is an interval tree used for?

A. Storing time intervals for scheduling
B. Finding all intervals that overlap with a given query interval efficiently
C. Measuring time intervals
D. Sorting intervals by length

Answer: B

Q698. What is the space complexity of a trie storing n strings with average length m over an alphabet of size <3ð

A. O(n)
B. O(n * m * <2•
C. O(n * m)
D. O(<2•

Answer: B

Q699. What is a balanced BST's advantage over a hash table for ordered operations?

A. Balanced BSTs have O(1) lookup
B. Balanced BSTs support ordered operations like range queries, in-order traversal, and predecessor/successor in O(log n)
C. Hash tables cannot store strings
D. Balanced BSTs use less memory

Answer: B

Q700. What is a circular buffer and where is it commonly used?

A. A buffer that stores data in a circle shape on disk
B. A fixed-size buffer that wraps around, commonly used in streaming, producer-consumer patterns, and I/O buffering
C. A buffer that can grow infinitely
D. A buffer used only for circular linked lists

Answer: B

Q701. What is the failure function (partial match table) in the KMP algorithm?

A. A table indicating which characters to skip in the text
B. A table storing the length of the longest proper prefix that is also a suffix for each prefix of the pattern
C. A table of hash values for the pattern
D. A table mapping characters to their positions

Answer: B

Q702. How does the Rabin-Karp algorithm handle spurious matches?

A. It ignores them completely
B. It performs a character-by-character comparison when hash values match to verify the actual match
C. It uses a different hash function each time
D. It rehashes the entire text

Answer: B

Q703. What is the Z-array for a string S?

A. An array of zeros
B. Z[i] = length of the longest substring starting at i that matches a prefix of S
C. An array sorting characters alphabetically
D. The reverse of the string stored as an array

Answer: B

Q704. What is the key idea behind the Boyer-Moore algorithm?

A. Searching from left to right in the pattern
B. Comparing the pattern from right to left and using bad character and good suffix heuristics to skip positions
C. Using hash values for comparison
D. Building a trie of all substrings

Answer: B

Q705. What is the time complexity of computing the longest palindromic substring using dynamic programming?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(n^3)

Answer: C

Q706. What is a rolling hash and how does it enable efficient string matching?

A. A hash that rolls dice for randomness
B. A hash function that can compute the hash of the next window in O(1) by removing the leading character and adding the trailing character
C. A hash that is computed once and never changes
D. A hash function specific to number sequences

Answer: B

Q707. How does the KMP algorithm achieve O(n + m) time complexity?

A. By sorting the text first
B. By never re-comparing characters that have already been matched, using the failure function to resume matching
C. By using hash values instead of character comparison
D. By dividing the text into blocks

Answer: B

Q708. What is the time complexity of checking if one string is a rotation of another?

A. O(n^2)
B. O(n) by concatenating the string with itself and searching for the other string
C. O(n log n)
D. O(2^n)

Answer: B

Q709. In polynomial string hashing, what values are typically chosen for the base p and modulus M?

A. p = 1 and M = 10
B. p greater than the alphabet size and M a large prime to minimize collisions
C. Both should be powers of 2
D. p = 0 and M = the string length

Answer: B

Q710. What is the best-case time complexity of the Boyer-Moore algorithm for pattern matching?

A. O(n * m)
B. O(n + m)
C. O(n / m)
D. O(n)

Answer: C

Q711. What is the relationship between the complexity classes P and NP?

A. P is a strict subset of NP and P "` NP is proven
B. P "† NP (P is contained in NP), but whether P = NP is unknown
C. NP ", P (NP is a strict subset of P)
D. P and NP are disjoint sets

Answer: B

Q712. What is the time complexity of Strassen's matrix multiplication algorithm?

A. O(n^3)
B. O(n^2)
C. O(n^(log2 7)) "H O(n^2.807)
D. O(n^2 log n)

Answer: C

Q713. Which of the following is a well-known NP-complete problem?

A. Binary search
B. Sorting an array
C. The Boolean Satisfiability Problem (SAT)
D. Finding the minimum in an array

Answer: C

Q714. What is the difference between time complexity and space complexity?

A. They always have the same value
B. Time complexity measures the number of operations as a function of input size; space complexity measures the memory used
C. Space complexity is always larger
D. Time complexity measures clock time in seconds

Answer: B

Q715. What is the significance of the O(n log n) barrier for comparison-based sorting?

A. It can be beaten by any sorting algorithm
B. No comparison-based sorting algorithm can have a worst case better than :'†â Æör â•
C. O(n log n) is the average case but O(n) is achievable in the worst case
D. Only merge sort achieves this bound

Answer: B

Q716. How does loop unrolling improve performance at the hardware level?

A. By reducing the total number of operations
B. By reducing branch overhead and enabling better instruction pipelining and cache utilization
C. By converting the loop to recursion
D. By eliminating the need for variables

Answer: B

Q717. What is the time-space tradeoff in memoization?

A. Memoization uses more time to save space
B. Memoization uses extra space to store computed results, reducing time by avoiding redundant computation
C. Memoization has no tradeoff
D. Memoization reduces both time and space

Answer: B

Q718. What is the complexity of adding two n-digit numbers?

A. O(1)
B. O(n)
C. O(n^2)
D. O(log n)

Answer: B

Q719. What is cache-oblivious algorithm design?

A. Algorithms that don't use any cache
B. Algorithms designed to efficiently use the memory hierarchy without knowing cache parameters
C. Algorithms that ignore memory constraints
D. Algorithms that clear the cache before running

Answer: B

Q720. What is the difference between worst-case and average-case complexity?

A. They are always the same
B. Worst-case gives the maximum time over all inputs of size n; average-case gives the expected time over a distribution of inputs
C. Average-case is always worse than worst-case
D. Worst-case only considers the best input

Answer: B

Hard Questions

360 questions

Q721. What is the time complexity of the recurrence T(n) = 3T(n/4) + n log n?

A. O(n^(log_4(3)))
B. O(n^2)
C. O(n log^2 n)
D. O(n log n)

Answer: D

Q722. Which of the following recurrences cannot be solved by the Master Theorem?

A. T(n) = T(n-1) + n
B. T(n) = 2T(n/2) + n log n
C. T(n) = 2T(n/2) + n
D. T(n) = 4T(n/2) + n^2

Answer: A

Q723. What is the amortized cost of insertion in a dynamic array that doubles its size when full?

A. O(1)
B. O(n^2)
C. O(log n)
D. O(n)

Answer: A

Q724. Using the substitution method, what is the solution to T(n) = T(n-1) + n with T(1) = 1?

A. O(2^n)
B. O(n log n)
C. O(n)
D. O(n^2)

Answer: D

Q725. What is the lower bound for any comparison-based sorting algorithm?

A. O(n log n)
B. O(n^2)
C. O(log n)
D. O(n)

Answer: A

Q726. In the potential method of amortized analysis, the amortized cost is defined as:

A. Actual cost × potential function
B. Actual cost + change in potential
C. Actual cost - change in potential
D. Actual cost / potential function

Answer: B

Q727. What complexity class does the problem of finding the median of an unsorted array belong to?

A. O(n^2) minimum brute force
B. O(n log n) using sort first
C. O(n) using deterministic selection
D. O(log n) using binary search

Answer: C

Q728. What is the space complexity of a recursive algorithm with depth d and O(1) space per call?

A. O(d)
B. O(2^d)
C. O(d^2)
D. O(1)

Answer: A

Q729. Which of the following is true about the relationship between time and space complexity?

A. Space complexity can exceed time complexity
B. Space complexity cannot exceed time complexity
C. Time complexity is always greater than space
D. Time and space complexity are always equal

Answer: B

Q730. What is the time complexity of T(n) = 2T(n/2) + n/log n?

A. O(n)
B. O(n log n)
C. O(n log log n)
D. O(n (log n)^2)

Answer: C

Q731. What is the amortized time complexity of enqueue and dequeue operations when implementing a queue with two stacks?

A. O(1) amortized for both
B. O(1) enqueue, O(n) dequeue
C. O(n) enqueue, O(1) dequeue
D. O(n) for both operations

Answer: A

Q732. How can you detect a cycle in a linked list efficiently?

A. By sorting the list nodes first in order
B. By converting the list into an array form
C. Using two nested loops over all nodes
D. Using Floyd's tortoise and hare algorithm

Answer: D

Q733. What is the time complexity of reversing a singly linked list iteratively?

A. O(1)
B. O(n^2)
C. O(n)
D. O(n log n)

Answer: C

Q734. In a skip list, what is the expected time complexity of search?

A. O(n)
B. O(1)
C. O(log n)
D. O(n log n)

Answer: C

Q735. What is an XOR linked list?

A. A circular linked list variant with XOR pointers
B. A linked list that stores only odd-valued elements
C. A linked list using XOR for data encryption purposes
D. A doubly linked list using XOR of addresses to save memory

Answer: D

Q736. How do you find the starting node of a cycle in a linked list using Floyd's algorithm?

A. Use a hash table to store all visited node addresses
B. Count all nodes in the list and compute the start
C. Reset one pointer to head and move both one step
D. Reverse the entire list and then detect the cycle

Answer: C

Q737. What is the minimum number of stacks needed to implement a deque?

A. 1
B. 3
C. 2
D. 4

Answer: C

Q738. What is the space overhead per node in a doubly linked list compared to a singly linked list?

A. One extra data field
B. No extra overhead
C. Two extra pointers
D. One extra pointer

Answer: D

Q739. What is the time complexity of merging two sorted linked lists into one sorted list?

A. O(n × m)
B. O(n + m)
C. O(n log m)
D. O(n^2)

Answer: B

Q740. What is a self-organizing list?

A. A list that reorders based on access patterns
B. A list that balances itself like a tree structure
C. A list that maintains sorted order automatically
D. A list that automatically deletes duplicate items

Answer: A

Q741. What is the time complexity of building a heap from an unsorted array of n elements?

A. O(n log n)
B. O(n)
C. O(log n)
D. O(n^2)

Answer: B

Q742. What is a threaded binary tree?

A. A tree where null pointers become in-order links
B. A binary tree with multiple root nodes
C. A tree with thread-safe atomic operations
D. A tree used in multithreading concurrency

Answer: A

Q743. What is the minimum number of nodes in an AVL tree of height h?

A. N(h) = N(h-1) + N(h-2) + 1
B. 2^(h+1) - 1
C. 2^h
D. h + 1

Answer: A

Q744. In a directed graph with V vertices, what is the maximum number of edges?

A. V^2
B. V
C. V(V-1)
D. V(V-1)/2

Answer: C

Q745. What is a B-tree of order m?

A. A tree data structure with m distinct roots
B. A self-balancing tree with at most m children
C. A binary tree with exactly m total levels
D. A graph with m connected components total

Answer: B

Q746. What is a Red-Black tree's maximum height for n nodes?

A. n/2
B. n
C. log n
D. 2 log(n+1)

Answer: D

Q747. What is a bipartite graph?

A. A graph divisible into two disjoint vertex sets
B. A fully connected complete graph structure
C. A graph with exactly two vertices in total
D. A graph with exactly two connected components

Answer: A

Q748. What is the time complexity of deleting the root from a max-heap of n elements?

A. O(n log n)
B. O(n)
C. O(1)
D. O(log n)

Answer: D

Q749. How many distinct BSTs can be formed with n distinct keys?

A. 2^n (powers of two)
B. n^2 (square of n)
C. C(n) the Catalan number
D. n! (factorial of n)

Answer: C

Q750. What is the difference between a strongly connected and weakly connected directed graph?

A. Strongly connected means directed paths exist between all pairs; weakly only without directions
B. There is no meaningful difference between strongly and weakly connected directed graphs
C. Strongly connected graphs always contain more edges than weakly connected directed graphs
D. Weakly connected graphs always have more vertices and edges than strongly connected ones

Answer: A

Q751. What is the worst-case time complexity of interpolation search?

A. O(n)
B. O(log n)
C. O(log log n)
D. O(" n)

Answer: A

Q752. What is Fibonacci search?

A. A search on Fibonacci heap data structures
B. A recursive search with Fibonacci-like recurrence
C. Searching for Fibonacci numbers in an array
D. A search using Fibonacci numbers to divide array

Answer: D

Q753. What is the time complexity of searching in a balanced binary search tree?

A. O(n log n)
B. O(n)
C. O(log n)
D. O(1)

Answer: C

Q754. How does binary search handle finding the first occurrence of a duplicate element in a sorted array?

A. Standard binary search always finds the first occurrence automatically
B. Binary search cannot handle duplicate elements in sorted arrays
C. By using linear search after binary search locates any occurrence
D. By modifying binary search to continue searching left after finding it

Answer: D

Q755. What is the order-statistic problem?

A. Counting elements in a range
B. Finding the median of sorted data
C. Finding the k-th smallest element
D. Sorting all elements into order

Answer: C

Q756. What is the time complexity of the Quickselect algorithm in the average case?

A. O(n log n)
B. O(n^2)
C. O(log n)
D. O(n)

Answer: D

Q757. How can you search in a sorted and rotated array?

A. Interpolation search handles rotation natively
B. Only linear search works for this case
C. Modified binary search identifies the sorted half
D. Ternary search is the only viable approach

Answer: C

Q758. What is the time complexity of searching in a skip list?

A. O(log n) expected
B. O(1) constant
C. O(n) linear
D. O(n log n) total

Answer: A

Q759. What is two-pointer technique in searching?

A. Using two separate arrays for searching data
B. Searching for two distinct elements at once
C. Binary search implemented with two mid points
D. Using two pointers from different positions to converge

Answer: D

Q760. What is the time complexity of finding the median of two sorted arrays of sizes m and n?

A. O((m + n) log(m + n))
B. O(m × n)
C. O(m + n)
D. O(log(min(m, n)))

Answer: D

Q761. What is radix sort's time complexity for n integers with d digits?

A. O(n × d^2)
B. O(d × (n + k)) where k is the base
C. O(n^2)
D. O(n log n)

Answer: B

Q762. What is the worst-case time complexity of randomized quick sort?

A. O(n)
B. O(n^2)
C. O(n log^2 n)
D. O(n log n)

Answer: B

Q763. What is bucket sort?

A. A sort that only works on string data
B. A sort distributing elements into buckets then merging
C. A merge sort variant with fixed partitions
D. A comparison sort using memory buckets

Answer: B

Q764. When is bucket sort most efficient?

A. When data is in reverse sorted order
B. When the array is extremely large in size
C. When input is uniformly distributed over a range
D. When data has many duplicate values present

Answer: C

Q765. What is introsort?

A. A merge sort variant with optimizations
B. A pure insertion sort variant
C. A hybrid starting with quicksort switching to heapsort
D. A non-comparison sort using radix method

Answer: C

Q766. What is TimSort?

A. A real-time sorting algorithm for streaming data
B. A sort that runs in constant time for all input
C. A sort named after its time complexity class
D. A hybrid stable sort using merge and insertion sort

Answer: D

Q767. Is quick sort a stable sorting algorithm?

A. Only with Lomuto partition scheme
B. Only when using random pivots
C. Yes, always stable
D. No, in its standard implementation

Answer: D

Q768. What is the time complexity of sorting n strings of maximum length s using radix sort?

A. O(s log n)
B. O(n^2 × s)
C. O(n × s)
D. O(n log n)

Answer: C

Q769. What is the Dutch National Flag problem in the context of sorting?

A. Coloring a graph with three distinct colors
B. Sorting elements of three different colors
C. Sorting using three different algorithms
D. Three-way partitioning around a pivot value

Answer: D

Q770. What is external sorting?

A. Sorting using network-based remote requests
B. Sorting data too large for memory using disk
C. Sorting using external third-party libraries
D. Sorting on external hardware devices only

Answer: B

Q771. What is universal hashing?

A. Hashing all elements to the same single bucket
B. Using the same hash function across all applications
C. Randomly selecting a hash function from a family of functions
D. A single hash function that works for all data types

Answer: C

Q772. What is perfect hashing?

A. A hash function that always distributes keys evenly
B. A hash function with no collisions for any input set
C. A hashing technique that requires no table at all
D. A two-level scheme guaranteeing O(1) worst-case lookup

Answer: D

Q773. What is a Bloom filter?

A. A type of hash table with chaining
B. A comparison-based sorting algorithm
C. A filter for removing duplicate elements
D. A probabilistic structure testing set membership

Answer: D

Q774. What is cuckoo hashing?

A. A technique using random hash function selection
B. A technique using two hash functions with displacement
C. A hashing technique inspired by bird nesting
D. A type of perfect hashing for static keys

Answer: B

Q775. What is consistent hashing?

A. Using the same hash function consistently always
B. A collision-free hashing technique for all inputs
C. A hashing that always produces the same output
D. A scheme remapping only a fraction of keys on change

Answer: D

Q776. What is the expected maximum chain length in a hash table with n elements and n slots using chaining?

A. O(log n / log log n)
B. O(" n)
C. O(1)
D. O(n)

Answer: A

Q777. What is Robin Hood hashing?

A. A two-level hashing scheme with perfect balance
B. A charitable hashing distribution across buckets
C. Stealing hash values from other tables entirely
D. Open addressing where longer probes displace shorter ones

Answer: D

Q778. What is the purpose of a cryptographic hash function?

A. To speed up hash table lookups efficiently
B. To compress files to smaller size format
C. To produce collision-resistant irreversible fixed output
D. To sort data in a deterministic order

Answer: C

Q779. What is hopscotch hashing?

A. An open addressing scheme with bounded neighborhoods
B. A random displacement hashing for load balance
C. A hashing game-based approach to distribution
D. A chaining variant using skip list buckets

Answer: A

Q780. What is the false positive probability of a Bloom filter with m bits, k hash functions, and n inserted elements?

A. (1 - e^(-kn/m))^k approx
B. n / m ratio
C. 1 / 2^k value
D. k / m ratio

Answer: A

Q781. What is a splay tree?

A. A B-tree variant used in database indexing
B. A self-adjusting BST that moves accessed nodes to root
C. A tree that spreads elements across multiple arrays
D. A tree that is always perfectly balanced

Answer: B

Q782. What is the time complexity of range query and update in a segment tree?

A. O(n) for both operations
B. O(n) query, O(1) update
C. O(1) query, O(n) update
D. O(log n) for both operations

Answer: D

Q783. What is a Fenwick tree (Binary Indexed Tree)?

A. A balanced BST variant with extra pointers
B. A structure for O(log n) prefix sums and updates
C. A tree used in networking protocols
D. A binary tree indexed by key values

Answer: B

Q784. What is lazy propagation in a segment tree?

A. Deferring updates to children until they are queried
B. Lazily inserting elements one at a time
C. Skipping unnecessary queries for performance
D. Delaying the tree construction until needed

Answer: A

Q785. What is a treap?

A. A BST-heap hybrid with keys and random priorities
B. A tree-shaped trap data structure
C. A triple-ended queue data structure
D. A tree with array-like access properties

Answer: A

Q786. What is the Euler tour technique for trees?

A. Linearizing a tree by recording DFS visit times
B. A graph coloring technique for tree nodes
C. Touring all leaf nodes in level order
D. Finding Euler paths in tree structures

Answer: A

Q787. What is Heavy-Light Decomposition (HLD) of a tree?

A. Decomposing into chains for efficient path queries
B. Separating heavy and light weight nodes
C. Removing heavy edges from the tree structure
D. Balancing an unbalanced tree by restructuring

Answer: A

Q788. What is the time complexity of LCA using binary lifting (sparse table)?

A. O(n^2) per query with no preprocessing needed
B. O(n) per query with no preprocessing
C. O(1) per query after O(n^2) preprocessing
D. O(log n) per query after O(n log n) preprocessing

Answer: D

Q789. What is a persistent segment tree?

A. A segment tree preserving all versions via path copying
B. A segment tree that never changes after construction
C. A segment tree stored on disk permanently
D. A compressed segment tree with reduced memory use

Answer: A

Q790. How does a B+ tree differ from a B-tree?

A. B+ tree is not a balanced tree structure
B. B+ tree has more levels than a B-tree
C. B+ tree stores all data at leaves with linked leaf nodes
D. B+ tree has fewer keys per node overall

Answer: C

Q791. What is the Floyd-Warshall algorithm?

A. An all-pairs shortest path DP algorithm
B. A minimum spanning tree algorithm only
C. A single-source shortest path algorithm
D. A graph coloring assignment algorithm

Answer: A

Q792. What is the time complexity of Floyd-Warshall algorithm?

A. O(V^3)
B. O(V^2)
C. O(V × E)
D. O(V + E)

Answer: A

Q793. What are Strongly Connected Components (SCCs) in a directed graph?

A. Maximal subgraphs with mutual reachability
B. Components with no cycles at all
C. Components with the most vertices total
D. Components with the most edges overall

Answer: A

Q794. Which algorithms can find Strongly Connected Components?

A. Dijkstra's shortest path algorithm
B. Prim's spanning tree algorithm
C. Kosaraju's and Tarjan's algorithms
D. Floyd-Warshall all-pairs algorithm

Answer: C

Q795. What is an articulation point (cut vertex) in a graph?

A. A vertex with no edges connected to it
B. The vertex with the smallest key value
C. A vertex whose removal disconnects the graph
D. A vertex with the maximum degree overall

Answer: C

Q796. What is a bridge in a graph?

A. An edge with the maximum weight assigned
B. An edge whose removal disconnects the graph
C. The longest edge in the graph by weight
D. An edge connecting two separate components

Answer: B

Q797. What is the maximum flow problem?

A. Finding the maximum number of vertices
B. Finding maximum flow from source to sink
C. Finding the maximum edge weight in graph
D. Sorting edges by their flow capacity

Answer: B

Q798. What is the Ford-Fulkerson method for maximum flow?

A. An iterative method finding augmenting paths until none remain
B. A divide and conquer approach for solving network flows
C. A greedy algorithm for computing the minimum cut value
D. A dynamic programming approach for computing max flow

Answer: A

Q799. What does the Max-Flow Min-Cut theorem state?

A. Maximum flow equals the number of all paths
B. Maximum flow equals minimum edge weight
C. Maximum flow equals the minimum cut capacity
D. Minimum cut equals the total number of edges

Answer: C

Q800. What is the time complexity of Dijkstra's algorithm with a Fibonacci heap?

A. O(V × E)
B. O((V + E) log V)
C. O(V log V + E)
D. O(V^2)

Answer: C

Q801. What is the matrix chain multiplication problem's time complexity using DP?

A. O(n^3)
B. O(n!)
C. O(2^n)
D. O(n^2)

Answer: A

Q802. What is the Traveling Salesman Problem (TSP)?

A. Finding the shortest path between two specific cities
B. Finding the fastest route avoiding all toll roads
C. Finding minimum cost Hamiltonian cycle visiting all cities
D. A minimum spanning tree problem for city networks

Answer: C

Q803. What is the time complexity of solving TSP using dynamic programming (Held-Karp)?

A. O(n^3)
B. O(n^2 × 2^n)
C. O(n!)
D. O(2^n)

Answer: B

Q804. What is branch and bound?

A. A hashing technique for collision resolution
B. A type of comparison-based sorting method
C. An optimization technique pruning suboptimal branches
D. A graph traversal method for shortest paths

Answer: C

Q805. What is the edit distance (Levenshtein distance) problem?

A. The minimum single-character edits between two strings
B. The difference in the lengths of two strings
C. The number of common characters in two strings
D. The distance between two arrays in memory

Answer: A

Q806. What is the difference between the 0/1 knapsack and fractional knapsack?

A. 0/1 uses greedy approach; fractional uses DP approach
B. Fractional knapsack is computationally harder overall
C. They are exactly the same problem formulation
D. 0/1 requires whole items (DP); fractional allows fractions (greedy)

Answer: D

Q807. What is a randomized algorithm?

A. An algorithm with random bugs in code
B. An algorithm using random numbers in its decisions
C. An algorithm that shuffles input before sorting
D. An algorithm that produces random wrong output

Answer: B

Q808. What is an approximation algorithm?

A. An algorithm that gives approximate time complexity
B. An algorithm finding near-optimal solutions in polynomial time
C. An imprecise algorithm with unreliable output
D. An algorithm that estimates the input data size

Answer: B

Q809. What is the subset sum problem?

A. Finding the sum of all array elements
B. Summing consecutive elements in an array
C. Finding the maximum valued subset overall
D. Determining if a subset sums to a target value

Answer: D

Q810. What is the meet-in-the-middle technique?

A. A two-pointer approach for sorted arrays
B. A median-finding technique using partitioning
C. Dividing the array in half for sorting
D. Splitting input in half and combining to reduce complexity

Answer: D

Q811. What is a suffix tree?

A. A tree that stores all string prefixes
B. A compressed trie of all suffixes for O(m) matching
C. An array of suffix pointers with sorting
D. A binary tree of sorted suffix indices

Answer: B

Q812. What is a van Emde Boas tree?

A. A tree with O(log log U) operations for integer keys
B. A tree with variable branching factor per level
C. A cache-oblivious tree for external memory
D. A self-balancing BST with color properties

Answer: A

Q813. What is a rope data structure?

A. A linked list variant for characters
B. A hash map for string key-value pairs
C. A graph connecting strings with edges
D. A balanced binary tree for efficient string operations

Answer: D

Q814. What is a count-min sketch?

A. A data structure for exact element counting
B. A compressed hash table with min values
C. A min-heap variant tracking element counts
D. A probabilistic structure for approximate frequency counts

Answer: D

Q815. What is a link-cut tree?

A. A tree where links can be physically cut
B. A doubly linked tree with removable nodes
C. A dynamic forest supporting link, cut, and path queries
D. A tree with removable edges but fixed vertices

Answer: C

Q816. What is a wavelet tree?

A. A tree for signal processing operations
B. A frequency-based heap for priority ordering
C. A Fourier transform data structure for arrays
D. A tree for rank, select, and range queries on sequences

Answer: D

Q817. What is a persistent data structure?

A. A data structure saved to disk permanently
B. A data structure that cannot ever be deleted
C. A data structure that persists across program runs
D. A structure preserving all previous versions on modification

Answer: D

Q818. What is a binomial heap?

A. A heap with binomial probability distribution
B. A heap used in statistical analysis methods
C. A binary heap with extra pointer fields added
D. A collection of binomial trees with O(log n) merge

Answer: D

Q819. What is an order-statistic tree?

A. An augmented BST supporting O(log n) rank and select
B. A tree for statistical analysis of data sets
C. A tree that orders elements statistically
D. A heap that tracks insertion order of elements

Answer: A

Q820. What is a succinct data structure?

A. A brief textual description of a data structure
B. A structure using near information-theoretic minimum space
C. A small fixed-capacity array with limited size
D. A compressed hash table designed for very dense data

Answer: B

Q821. What is the Aho-Corasick algorithm?

A. A multi-pattern matcher building a trie-based automaton
B. A compression algorithm for pattern libraries
C. A single pattern matching algorithm only
D. A string sorting algorithm for multiple strings

Answer: A

Q822. What is Manacher's algorithm?

A. A string sorting algorithm for arrays
B. A pattern matching algorithm for text
C. A string compression algorithm for storage
D. An O(n) algorithm finding all palindromic substrings

Answer: D

Q823. What is the time complexity of building a suffix array using the DC3/skew algorithm?

A. O(n log^2 n)
B. O(n log n)
C. O(n^2)
D. O(n)

Answer: D

Q824. What is the LCP (Longest Common Prefix) array?

A. An array storing longest common prefix lengths between consecutive sorted suffixes
B. An array storing character frequencies found within a given string
C. An array storing the lengths of all strings within a collection
D. An array storing prefix sums computed over numeric data elements

Answer: A

Q825. What is Burrows-Wheeler Transform (BWT)?

A. A string matching algorithm for patterns
B. A reversible transformation aiding text compression
C. A string encryption method for security
D. A hash function designed for string data

Answer: B

Q826. What is a suffix automaton?

A. An automaton that generates all suffixes
B. A pattern matching machine for any text
C. A minimal DFA recognizing all suffixes of a string
D. A suffix tree variant with compressed nodes

Answer: C

Q827. What is the time complexity of the Boyer-Moore algorithm in the best case?

A. O(n/m)
B. O(n × m)
C. O(n)
D. O(n + m)

Answer: A

Q828. What is polynomial string hashing?

A. Hashing using polynomial long division
B. Computing hash as sum of characters times powers of a base
C. Hashing that produces polynomial-length hash output
D. Using polynomial time algorithms for hash computation

Answer: B

Q829. What is the longest repeated substring problem's optimal solution?

A. O(n^3) brute force approach
B. O(n^2) using dynamic programming
C. O(n log^2 n) using rolling hashing
D. O(n) using suffix array with LCP array

Answer: D

Q830. What is the number of distinct substrings of a string of length n?

A. n(n+1)/2 minus sum(LCP)
B. 2^n (exponential count)
C. n^2 (quadratic count)
D. n! (factorial of n)

Answer: A

Q831. What is the complexity class PSPACE?

A. Problems solvable using polynomial space amount
B. Problems requiring exponential space to solve
C. Problems solvable in constant space only
D. Problems with no space requirement at all

Answer: A

Q832. What is Cook's theorem?

A. P equals NP has been conclusively proven
B. Every NP problem is fundamentally unsolvable
C. Every problem is solvable in polynomial time
D. The Boolean satisfiability problem (SAT) is NP-complete

Answer: D

Q833. What is the difference between a decision problem and an optimization problem?

A. Decision has yes/no answer; optimization seeks best solution
B. They are exactly the same type of problem
C. Optimization problems are always solvable in P
D. Decision problems are always computationally harder

Answer: A

Q834. What is pseudo-polynomial time?

A. An algorithm that pretends to be efficient
B. Time polynomial in input value rather than input size
C. False polynomial time that does not exist
D. The same as polynomial time in all cases

Answer: B

Q835. What is the 3-SAT problem?

A. A Boolean satisfiability problem with 3 literals per clause
B. A problem with exactly 3 valid solutions
C. A sorting problem with 3 distinct elements
D. Satisfying exactly 3 linear equations

Answer: A

Q836. What is the vertex cover problem?

A. Coloring all vertices with minimum colors
B. Finding all connected vertices in a graph
C. Finding smallest vertex set covering all edges
D. Covering all vertices with color assignments

Answer: C

Q837. What is the approximation ratio of the 2-approximation algorithm for vertex cover?

A. The solution is at most twice the optimal
B. The solution is at most 1.5 times optimal
C. The solution is at most 3 times the optimal
D. The solution always equals the optimal exactly

Answer: A

Q838. What is the class co-NP?

A. The class whose complement problems are in NP
B. The complement of NP problems
C. Problems that are harder than NP class
D. Problems that are not in NP at all

Answer: A

Q839. What is a parameterized algorithm?

A. An algorithm that dynamically changes its parameters at runtime
B. An algorithm whose complexity depends on input size and a fixed parameter
C. A tunable sorting algorithm that uses an adjustable pivot element
D. An algorithm that has many user-tunable parameter values

Answer: B

Q840. What is the time hierarchy theorem?

A. All problems can be solved in the same time
B. Time complexity is always fixed and unchanging
C. More time allows a Turing machine to solve more problems
D. There is no hierarchy in time complexity classes

Answer: C

Q841. What is the time complexity of T(n) = 3T(n/4) + n log n by Master Theorem?

A. O(n log^2 n)
B. O(n^2 log n)
C. O(n log n)
D. O(n^(log_4 3))

Answer: C

Q842. Which method handles recurrences not solvable by the Master Theorem?

A. Simple iteration counting method
B. Akra-Bazzi generalized method
C. Direct formula application method
D. Basic substitution guess method

Answer: B

Q843. What is the complexity of T(n) = 2T(n/2) + n log n?

A. O(n^2 log n)
B. O(n log n)
C. O(n log^2 n)
D. O(n^2)

Answer: C

Q844. In amortized analysis, what is the potential method?

A. Assigning actual cost to each operation directly
B. Assigning a potential energy function to data structure
C. Counting total operations divided by total time
D. Measuring memory usage across all operations run

Answer: B

Q845. What is the significance of the P vs NP problem?

A. It asks if space complexity equals time complexity always
B. It asks if all algorithms can run in constant time always
C. It asks if polynomial verification implies polynomial solving
D. It asks if every problem has a brute force solution path

Answer: C

Q846. What does the recursion tree method visualize?

A. The sorted order of elements after each comparison
B. The memory allocation pattern of the algorithm run
C. The branching cost at each level of the recurrence
D. The input-output mapping of the algorithm function

Answer: C

Q847. What is the time complexity of T(n) = T(n/3) + T(2n/3) + n?

A. O(n)
B. O(n log^2 n)
C. O(n^2)
D. O(n log n)

Answer: D

Q848. When does the Master Theorem not apply directly?

A. When b is greater than a in the recurrence
B. When the recurrence has exactly two subproblems
C. When subproblems have different unequal sizes
D. When f(n) is a simple polynomial function

Answer: C

Q849. What is the aggregate method in amortized analysis?

A. Using a potential function to track stored credits
B. Assigning different costs to different operation types
C. Counting only the most expensive operation each time
D. Summing total cost of n operations then dividing by n

Answer: D

Q850. What is the difference between deterministic and nondeterministic algorithms?

A. Deterministic ones use randomness; nondeterministic do not
B. Deterministic ones are slower; nondeterministic are faster
C. Nondeterministic ones always find the optimal solution path
D. Nondeterministic ones can explore multiple paths at once

Answer: D

Q851. How does the XOR linked list reduce memory usage?

A. It stores XOR of next and previous addresses per node
B. It stores only even-indexed nodes and skips odd ones
C. It removes half the nodes and links remaining ones
D. It compresses data values using XOR encoding scheme

Answer: A

Q852. What is the time complexity of reversing a singly linked list iteratively?

A. O(1)
B. O(n^2)
C. O(n log n)
D. O(n)

Answer: D

Q853. How can you detect a cycle in a linked list in O(1) space?

A. Use Floyd's cycle detection with two pointer speeds
B. Sort the list and check for duplicate node values
C. Use a hash set to store all visited node addresses
D. Store visited flags inside each node data field

Answer: A

Q854. What is the worst-case time for dynamic array resizing?

A. O(log n) due to binary copy optimization
B. O(1) per single resize operation performed
C. O(n) to copy all elements to new array
D. O(n^2) due to nested copy and shift steps

Answer: C

Q855. How is a min-stack implemented with O(1) getMin?

A. Store min value only at the bottom of the stack
B. Sort the stack after every push operation call
C. Scan the entire stack each time getMin is called
D. Maintain an auxiliary stack tracking current minimums

Answer: D

Q856. What is the skip list and how does it improve linked list search?

A. A multi-level linked list with express lane pointers
B. A balanced tree structure embedded in a linked list
C. A sorted array overlay that replaces the linked list
D. A hash table combined with a singly linked list node

Answer: A

Q857. What is an unrolled linked list?

A. A linked list where each node stores an array of elements
B. A linked list with randomized pointer assignments only
C. A linked list where nodes are stored in reverse order
D. A linked list that has been converted into a flat array

Answer: A

Q858. How do you find the intersection point of two linked lists efficiently?

A. Reverse both lists and compare from the tail forward
B. Hash all elements of both lists and compare buckets
C. Sort both lists and perform a binary search on each
D. Compute lengths, align starts, then traverse together

Answer: D

Q859. What is the time complexity of merging k sorted linked lists using a min-heap?

A. O(nk log k)
B. O(nk^2)
C. O(nk)
D. O(n log k)

Answer: A

Q860. What is a gap buffer and where is it used?

A. An array with a movable gap used in text editors
B. A circular array used for network packet buffering
C. A stack-based buffer used for function call storage
D. A linked list variant used in database indexing

Answer: A

Q861. What is a bipartite graph?

A. A graph whose vertices can be split into two disjoint independent sets
B. A directed graph with edges going in exactly two directions only
C. A graph where every vertex has exactly two edges connected to it
D. A graph that contains exactly two connected components in total

Answer: A

Q862. How many edges does a tree with n vertices have?

A. n + 1
B. n
C. n - 1
D. n / 2

Answer: C

Q863. What is the worst-case height of an unbalanced BST with n nodes?

A. O(sqrt(n))
B. O(n log n)
C. O(log n)
D. O(n)

Answer: D

Q864. What is a strongly connected component in a directed graph?

A. A subgraph where every vertex is reachable from every other vertex
B. A subgraph containing exactly one vertex with no outgoing edges
C. A subgraph where all vertices have the same in-degree value
D. A maximal set of vertices with no edges between any of them

Answer: A

Q865. What is the Euler's formula for connected planar graphs?

A. V + E - F = 2
B. V + E + F = 2
C. V - E - F = 2
D. V - E + F = 2

Answer: D

Q866. What is a threaded binary tree?

A. A binary tree where null pointers are replaced with in-order links
B. A binary tree where each node stores a thread identifier
C. A binary tree stored across multiple threads of execution
D. A binary tree that requires multi-threaded access for traversal

Answer: A

Q867. What is a hypergraph?

A. A graph where edges can connect any number of vertices
B. A three-dimensional representation of a standard graph
C. A graph where vertices are organized in a grid pattern
D. A graph where each edge connects exactly two vertices

Answer: A

Q868. What is the chromatic number of a graph?

A. The maximum degree of any single vertex in the graph
B. The minimum number of colors needed to color all vertices
C. The number of connected components in the given graph
D. The total count of edges present in the entire graph

Answer: B

Q869. What is the maximum number of edges in a simple undirected graph with n vertices?

A. n^2
B. n(n-1)
C. n(n+1)/2
D. n(n-1)/2

Answer: D

Q870. What is a k-ary tree?

A. A tree with k root nodes and shared leaf structure
B. A tree where each node has exactly k children always
C. A tree where each node has at most k children nodes
D. A tree that has exactly k levels from root to leaves

Answer: C

Q871. What is the worst-case time complexity of interpolation search?

A. O(n)
B. O(sqrt(n))
C. O(log log n)
D. O(log n)

Answer: A

Q872. How does exponential search combine with binary search?

A. It uses exponentiation to compute hash for direct access
B. It finds a range by doubling then binary searches within
C. It applies binary search on exponentially growing subarrays
D. It sorts the array exponentially then searches linearly

Answer: B

Q873. What is the time complexity of search in a skip list?

A. O(log n) expected with O(n) worst case
B. O(n log n) expected and O(n^2) worst case
C. O(n) in both average and worst cases
D. O(1) in all cases due to direct access

Answer: A

Q874. What is the lower bound for comparison-based searching in sorted arrays?

A. O(log n)
B. O(1)
C. O(sqrt(n))
D. O(n)

Answer: A

Q875. How does search work in a van Emde Boas tree?

A. It exploits universe size for O(log log u) operations
B. It uses comparison-based binary search on tree nodes
C. It performs breadth-first search on a balanced binary tree
D. It hashes each element for constant time access always

Answer: A

Q876. What is a fractional cascading technique?

A. Dividing search space into fractional geometric sections
B. Cascading hash functions for multi-level collision handling
C. Sharing elements between catalogs to speed up searches
D. Splitting arrays into fractions for parallel search

Answer: C

Q877. What is the time complexity of searching in an optimal binary search tree?

A. O(1) since the tree is precomputed optimally
B. O(n) for every possible search query
C. O(n^2) due to the overhead of optimal structure
D. O(log n) or better based on access frequencies

Answer: D

Q878. What is the two-pointer search technique?

A. Using two separate arrays for parallel search
B. Using two recursive calls to divide the search space
C. Using two hash tables to resolve search collisions
D. Using two indices moving toward each other or same direction

Answer: D

Q879. What is the complexity of searching in a perfectly balanced k-d tree?

A. O(k * log n)
B. O(n)
C. O(log n)
D. O(n^(1-1/d) + k)

Answer: D

Q880. What is the sentinel linear search optimization?

A. Adding a sentinel at the end to eliminate bound checking
B. Using multiple sentinels to divide array into segments
C. Replacing all elements with sentinels then binary search
D. Sorting the array first and placing sentinel in middle

Answer: A

Q881. What is the lower bound for comparison-based sorting?

A. O(log n)
B. O(n)
C. O(n log n)
D. O(n^2)

Answer: C

Q882. What is the worst-case time complexity of randomized quick sort?

A. O(n log^2 n) amortized
B. O(n log n) guaranteed always
C. O(n^2) but extremely unlikely
D. O(n) with high probability

Answer: C

Q883. What is introsort and why is it used?

A. A sort using interpolation search within sorting passes
B. A sort based on introspective analysis of data patterns
C. A sort combining insertion, heap, and quick sort strategies
D. A sort that introduces randomness into merge sort steps

Answer: C

Q884. How does Tim sort combine merge sort and insertion sort?

A. It merge sorts odd indices and insertion sorts even ones
B. It finds natural runs, extends them with insertion sort, merges
C. It uses merge sort first then refines with insertion sort
D. It alternates between merge and insertion on each pass

Answer: B

Q885. What is the time complexity of bucket sort in the average case?

A. O(n + k)
B. O(n log n)
C. O(n^2)
D. O(k log k)

Answer: A

Q886. What is the median-of-three strategy in quick sort?

A. Sorting the first three elements as a preprocessing step
B. Dividing the array into three parts instead of two parts
C. Choosing the median of first, middle, and last elements
D. Running three separate quick sorts and taking the median

Answer: C

Q887. What is the worst-case time of radix sort for n d-digit numbers?

A. O(n log n)
B. O(d * n^2)
C. O(n^2)
D. O(d * n)

Answer: D

Q888. Why is merge sort preferred for sorting linked lists?

A. Merge sort automatically handles duplicate values better
B. Merge sort is simpler to implement than other sorts
C. Merge sort has better time complexity for linked lists
D. Merge sort needs no random access and merges in O(1) space

Answer: D

Q889. What is the Dutch National Flag problem in sorting?

A. A problem of sorting flags in alphabetical country order
B. Sorting elements from multiple countries by region codes
C. Arranging colored blocks using merge sort partitioning
D. Sorting elements by color: red, white, and blue efficiently

Answer: D

Q890. What is the adaptive property in sorting algorithms?

A. The algorithm automatically selects the best data type
B. The algorithm takes advantage of existing order in input
C. The algorithm adapts to different programming languages
D. The algorithm changes its memory usage based on input

Answer: B

Q891. What is universal hashing?

A. A single hash function that works for all input types
B. Using the same hash function universally in all tables
C. A hash function standardized across all programming uses
D. Randomly selecting a hash function from a designed family

Answer: D

Q892. What is perfect hashing?

A. A hash function that produces perfectly sequential values
B. A hashing method that uses perfectly balanced tree nodes
C. A hashing scheme with zero collisions for a static set
D. A hash table with load factor of exactly one point zero

Answer: C

Q893. What is a cuckoo hash table?

A. A table using bird-like migration patterns for elements
B. A table that randomly evicts elements when it gets full
C. A table using cuckoo search optimization for hash values
D. A table using two hash functions with element displacement

Answer: D

Q894. What is consistent hashing used for?

A. Maintaining sorted order of keys in the hash table
B. Ensuring hash values are always consecutive integers
C. Distributing keys across nodes with minimal reassignment
D. Guaranteeing the same hash for same input every time

Answer: C

Q895. What is a Bloom filter?

A. A sorting filter applied before inserting into hash table
B. A hash table variant that filters out duplicate key entries
C. A deterministic set membership data structure using hashing
D. A probabilistic structure that may give false positives only

Answer: D

Q896. What is Robin Hood hashing?

A. A chaining method where collisions are sorted by hash value
B. Stealing slots from rich elements to give to poor elements
C. Using a tree structure within each hash table bucket slot
D. Probing by comparing displacement distances and swapping

Answer: D

Q897. What is the expected longest probe sequence with linear probing at load factor 0.5?

A. O(1)
B. O(sqrt(n))
C. O(log n)
D. O(n)

Answer: C

Q898. What is the space complexity of a Bloom filter with n elements and false positive rate p?

A. O(n * log(1/p))
B. O(log n)
C. O(n^2)
D. O(n)

Answer: A

Q899. What is hopscotch hashing?

A. Each element must be within a fixed neighborhood of its hash
B. Elements are sorted by hop count from their hash position
C. Probing follows a randomized hopping pattern each time
D. Elements hop between two completely separate tables

Answer: A

Q900. What is the worst-case lookup time in a hash table with chaining?

A. O(1)
B. O(log n)
C. O(n log n)
D. O(n)

Answer: D

Q901. What is a splay tree?

A. A multi-way search tree used for external data storage
B. A self-adjusting BST that moves accessed nodes to root
C. A tree that splays leaves outward for balanced structure
D. A tree that spreads nodes evenly across all levels always

Answer: B

Q902. What is a treap?

A. A tree with trap-based deletion for security purposes
B. A combination of a BST and a heap using random priorities
C. A binary tree that operates as both stack and queue
D. A tree that traps duplicate elements in separate nodes

Answer: B

Q903. What is the time complexity of range query in a segment tree?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: B

Q904. What is a B+ tree and how does it differ from a B-tree?

A. B+ trees have fewer keys per node than standard B-trees
B. B+ trees store all data in leaves with linked leaf nodes
C. B+ trees do not maintain sorted order within their nodes
D. B+ trees store data only in internal nodes not in leaves

Answer: B

Q905. What is lazy propagation in a segment tree?

A. Deferring updates to child nodes until they are accessed
B. Propagating updates to all nodes immediately upon change
C. Lazily constructing the tree only when queries are made
D. Propagating values from leaves to root after every update

Answer: A

Q906. What is an Euler tour of a tree?

A. A traversal listing each node when entering and leaving
B. A tour that visits only leaf nodes of the given tree
C. A traversal that processes nodes in reverse level order
D. A path visiting every edge exactly once in entire tree

Answer: A

Q907. What is the amortized time of splay tree operations?

A. O(log n) per operation in the amortized analysis
B. O(n) per operation in the amortized analysis
C. O(1) per operation in the amortized analysis
D. O(n log n) per operation in the amortized analysis

Answer: A

Q908. What is a Fenwick tree (Binary Indexed Tree)?

A. A tree that indexes binary representations of all keys
B. A full binary tree storing prefix sums at each node
C. A binary tree indexed by Fenwick encoding of data keys
D. A tree using binary indexing for prefix sum queries

Answer: D

Q909. What is a persistent data structure in the context of trees?

A. A tree that persists on disk rather than in memory
B. A tree that cannot be modified once it is created
C. A tree stored in a persistent database management system
D. A tree that preserves all previous versions after updates

Answer: D

Q910. What is the order of a B-tree?

A. The total number of keys stored in the entire tree
B. The maximum number of children each node can have
C. The minimum number of keys in the root node only
D. The height of the tree from root to deepest leaf

Answer: B

Q911. What is Tarjan's algorithm used for?

A. Finding shortest paths in weighted directed graph
B. Finding minimum spanning tree of an undirected graph
C. Finding strongly connected components in directed graph
D. Finding maximum matching in a bipartite graph only

Answer: C

Q912. What is an articulation point in a graph?

A. A vertex that lies on the shortest path between two nodes
B. A vertex whose removal disconnects the graph components
C. A vertex with the highest degree in the entire graph
D. A vertex that connects two spanning trees together only

Answer: B

Q913. What is the Ford-Fulkerson method?

A. Computing maximum flow in a flow network iteratively
B. Finding shortest path between two vertices in a graph
C. Finding minimum cut in an unweighted undirected graph
D. Computing minimum spanning tree using edge relaxation

Answer: A

Q914. What is the max-flow min-cut theorem?

A. Maximum flow equals the minimum cut capacity value
B. Minimum flow equals the maximum cut capacity value
C. Maximum cut always exceeds the minimum flow amount
D. Maximum flow equals minimum number of edges in graph

Answer: A

Q915. What is Kosaraju's algorithm?

A. A two-pass DFS algorithm for strongly connected components
B. A dynamic programming approach for all-pairs shortest paths
C. A greedy algorithm for finding minimum spanning trees
D. A divide and conquer method for planar graph coloring

Answer: A

Q916. What is a bridge in a graph?

A. A vertex that connects two separate components together
B. An edge whose removal disconnects the graph into parts
C. The shortest edge in the entire graph by edge weight
D. An edge that forms part of every spanning tree found

Answer: B

Q917. What is the A* search algorithm?

A. A brute-force search examining all paths exhaustively
B. A random walk search with probabilistic path selection
C. An uninformed search using only edge weight for priority
D. A heuristic-guided search combining cost and estimate

Answer: D

Q918. What is the time complexity of finding all bridges in a graph?

A. O(V + E)
B. O(V * E)
C. O(E^2)
D. O(V^2)

Answer: A

Q919. What is a minimum cut in a graph?

A. The smallest set of edges whose removal disconnects source from sink
B. The edge with the minimum weight in the entire graph structure
C. The minimum number of vertices needed to cover all edges
D. The shortest path between any two vertices in graph

Answer: A

Q920. What is the Johnson's algorithm for all-pairs shortest paths?

A. Running Bellman-Ford from every vertex in the given graph
B. Using Floyd-Warshall with optimized matrix multiplication
C. Using BFS from each vertex for unweighted graph paths
D. Reweighting edges then running Dijkstra from each vertex

Answer: D

Q921. What is the matrix chain multiplication problem?

A. Multiplying all matrices in a chain from left to right
B. Multiplying chains of numbers using matrix representation
C. Finding optimal parenthesization to minimize scalar multiplications
D. Finding the chain of matrices with the largest determinant

Answer: C

Q922. What is the traveling salesman problem (TSP)?

A. Finding the fastest way to deliver mail in one city only
B. Finding the shortest route visiting all cities exactly once
C. Finding the minimum spanning tree of a city road network
D. Finding the cheapest flight between two specific airports

Answer: B

Q923. What is the time complexity of the DP solution for TSP?

A. O(n!)
B. O(n^2 * 2^n)
C. O(n^3)
D. O(2^n)

Answer: B

Q924. What is the principle of optimality in dynamic programming?

A. The first feasible solution found is always the optimal one
B. An optimal solution contains optimal solutions to subproblems
C. Every subproblem must be solved before the main problem
D. The optimal solution always uses the greedy choice at start

Answer: B

Q925. What is a randomized algorithm?

A. An algorithm that only works on randomly generated inputs
B. An algorithm that uses random numbers to guide decisions
C. An algorithm that produces random incorrect output values
D. An algorithm that randomly skips steps for faster speed

Answer: B

Q926. What is the edit distance problem?

A. Finding the distance between two nodes in an edit graph
B. Counting the number of edits made to a document over time
C. Finding minimum operations to transform one string to another
D. Measuring the physical distance between edit buttons on screen

Answer: C

Q927. What is the difference between Las Vegas and Monte Carlo algorithms?

A. Las Vegas always gives correct answer; Monte Carlo may not
B. Monte Carlo always gives correct answer; Las Vegas may not
C. Both may give wrong answers but differ in probability only
D. Both always give correct answers but differ in time only

Answer: A

Q928. What is the subset sum problem?

A. Finding the sum of all elements in a given set of numbers
B. Partitioning a set into subsets with equal element counts
C. Determining if a subset exists that sums to a target value
D. Finding the largest subset with maximum element sum total

Answer: C

Q929. What is the time complexity of the edit distance DP solution?

A. O(max(m, n))
B. O(m^2 * n^2)
C. O(m + n)
D. O(m * n)

Answer: D

Q930. What is the approximation ratio of a 2-approximation algorithm?

A. It runs in twice the time of the optimal algorithm speed
B. It finds a solution at most twice the optimal cost value
C. It finds a solution exactly half of the optimal cost value
D. It uses twice the memory of the optimal algorithm space

Answer: B

Q931. What is a van Emde Boas tree?

A. A BST variant with van Emde Boas balancing rotations
B. A hash tree using van Emde Boas hashing for collisions
C. A tree supporting O(log log u) operations on integers
D. A tree named after its visual Van-like branching pattern

Answer: C

Q932. What is a splay tree's working set property?

A. Elements in the working set are stored in a separate list
B. The working set of the program determines tree structure
C. Accessing recently accessed elements is faster than others
D. Recently accessed elements are always at leaf positions

Answer: C

Q933. What is a link-cut tree?

A. A binary tree where links between nodes can be toggled
B. A linked list that can be cut and reconnected efficiently
C. A tree that measures the number of cuts needed to split
D. A tree that links and cuts edges to maintain forest paths

Answer: D

Q934. What is the inverse Ackermann function's role in Union-Find?

A. It generates the sequence of ranks for union operations
B. It determines the initial size of the disjoint set array
C. It computes the Ackermann value used for hashing keys
D. It describes the amortized time per operation with optimizations

Answer: D

Q935. What is a persistent segment tree?

A. A segment tree stored persistently on disk for backup
B. A segment tree that persists in memory and never freed
C. A segment tree that preserves all versions after updates
D. A segment tree with persistent root node that never changes

Answer: C

Q936. What is a suffix tree?

A. A binary tree where values are string suffix indices
B. A tree storing all prefixes of a given input string
C. A tree that sorts strings by their suffix characters
D. A compressed trie of all suffixes of a given string

Answer: D

Q937. What is the time complexity to build a suffix array?

A. O(n log^2 n)
B. O(n log n)
C. O(n^2)
D. O(n)

Answer: D

Q938. What is a range tree?

A. A tree storing ranges of IP addresses for routing
B. A multi-level tree for orthogonal range queries on points
C. A binary tree where nodes represent value range intervals
D. A tree that stores the range of values at each level

Answer: B

Q939. What is the merge operation complexity in a binomial heap?

A. O(n log n)
B. O(log n)
C. O(1)
D. O(n)

Answer: B

Q940. What is a weight-balanced tree?

A. A tree where each node stores its subtree weight for balancing
B. A tree that assigns weights to nodes based on depth level
C. A tree balanced by the total data size in each subtree
D. A tree where all edges have equal weight values assigned

Answer: A

Q941. What is Manacher's algorithm?

A. An algorithm for computing edit distance in linear time
B. An algorithm for sorting strings in optimal O(n log n) time
C. An algorithm for building suffix trees in constant space
D. An algorithm for finding all palindromic substrings in O(n)

Answer: D

Q942. What is the Aho-Corasick algorithm?

A. A text indexing algorithm using Aho-style balanced trees
B. A single-pattern matching algorithm using dynamic tables
C. A multi-pattern matching algorithm using automaton and trie
D. A string compression algorithm using coarse approximation

Answer: C

Q943. What is the suffix automaton?

A. A pushdown automaton that processes suffix expressions
B. An automaton that generates all possible suffixes randomly
C. A finite automaton accepting all suffixes of a given string
D. A deterministic automaton that sorts suffixes alphabetically

Answer: C

Q944. What is the Burrows-Wheeler Transform used for?

A. Transforming binary data into readable text string format
B. Sorting strings using Wheeler's specialized comparison
C. Encrypting strings for secure data transmission purposes
D. Transforming strings to improve compressibility of data

Answer: D

Q945. What is the LCP array in suffix array construction?

A. An array mapping each character position to its prefix identifier
B. An array of longest common prefixes between consecutive sorted suffixes
C. An array storing the length of each suffix in the original string
D. An array of least common patterns found across all string suffixes

Answer: B

Q946. What is the time complexity of building a suffix tree using Ukkonen's algorithm?

A. O(n^2)
B. O(n^3)
C. O(n)
D. O(n log n)

Answer: C

Q947. What is the longest repeated substring problem?

A. Finding the longest substring that occurs at least twice
B. Finding the longest string that can be formed by repeating
C. Finding any substring that repeats in consecutive positions
D. Finding the substring that appears most frequently overall

Answer: A

Q948. What is the worst-case time of Boyer-Moore algorithm?

A. O(n / m)
B. O(n + m)
C. O(n * m)
D. O(n)

Answer: C

Q949. What is the difference between Rabin-Karp and KMP algorithms?

A. Rabin-Karp is recursive; KMP is always purely iterative
B. Rabin-Karp is for numbers; KMP is only for text strings
C. Rabin-Karp uses hashing; KMP uses prefix table comparison
D. Rabin-Karp uses sorting; KMP uses dynamic programming tables

Answer: C

Q950. What is the time complexity of computing the Z-array?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(n * m)

Answer: C

Q951. What is Strassen's algorithm and what complexity does it achieve?

A. A matrix multiplication algorithm achieving O(n^2.81) time
B. A string matching algorithm achieving O(n + m) time bound
C. A graph search algorithm achieving O(V + E) complexity
D. A sorting algorithm achieving O(n log n) complexity always

Answer: A

Q952. What is an NP-complete problem?

A. A problem that can only be solved using nondeterministic steps
B. A problem that has been completely solved in polynomial time
C. A problem in NP that is at least as hard as all NP problems
D. A problem with no possible solution by any known algorithm

Answer: C

Q953. What is the significance of NP-hardness?

A. NP-hard problems can only be solved using quantum computers
B. NP-hard problems are easy to solve in polynomial time
C. NP-hard problems are at least as hard as all problems in NP
D. NP-hard problems have exactly one unique correct solution

Answer: C

Q954. What is the Cook-Levin theorem?

A. It proves that SAT is the first NP-complete problem known
B. It proves that P equals NP for all practical problems
C. It proves that all NP-hard problems are also in the class P
D. It proves that no NP problem can be solved in polynomial time

Answer: A

Q955. What is the complexity class PSPACE?

A. Problems where space equals time complexity for all inputs
B. Problems requiring exponential space for any solution found
C. Problems solvable in polynomial time with unlimited space
D. Problems solvable using polynomial amount of memory space

Answer: D

Q956. What is an FPT (Fixed-Parameter Tractable) algorithm?

A. An algorithm with fixed running time for all parameter values
B. An algorithm where exponential cost depends only on parameter k
C. An algorithm using fixed-point arithmetic for optimization
D. An algorithm that fixes parameters to reduce space complexity

Answer: B

Q957. What is the gap between P and NP problems practically?

A. P problems are deterministic; NP problems are always randomized
B. P problems use less memory; NP problems use more memory only
C. P problems are solved in seconds; NP takes millennia for large input
D. There is no practical gap; all NP problems run fast in practice

Answer: C

Q958. What is a polynomial-time reduction?

A. Reducing the polynomial degree of a time complexity function
B. Transforming one problem into another in polynomial time steps
C. Converting exponential algorithms into polynomial ones always
D. Reducing the running time of any algorithm to polynomial time

Answer: B

Q959. What is the complexity of the best known matrix multiplication algorithm?

A. O(n^2) matching the lower bound for matrix multiplication
B. O(n^3) with no improvement possible over standard method
C. O(n^2.373) using advanced algebraic technique approaches
D. O(n^2.5) using a combination of Strassen and standard methods

Answer: C

Q960. What is the role of reductions in proving NP-completeness?

A. Reductions simplify NP-complete problems to make them solvable
B. Reductions convert NP-complete problems into polynomial ones
C. Reductions eliminate the need to prove problem membership in NP
D. Reductions show a known NP-complete problem transforms to new one

Answer: D

Q961. What is the solution to the recurrence T(n) = 2T(n/4) + " n?

A. O(n)
B. O(" n log n)
C. O(n log n)
D. O(" n)

Answer: B

Q962. What is the Akra-Bazzi method used for?

A. Sorting elements in an array
B. Solving recurrences with unequal subproblem sizes that the Master Theorem cannot handle
C. Finding shortest paths in graphs
D. Balancing binary search trees

Answer: B

Q963. In amortized analysis, what does the potential function :b &W &W6VçCð

A. The actual cost of each operation
B. The stored-up energy in the data structure that can pay for future expensive operations
C. The maximum possible running time
D. The probability of an expensive operation occurring

Answer: B

Q964. What complexity class does an algorithm with running time O(n^(log_2 3)) belong to when compared to O(n^2)?

A. Faster than O(n^2) since log_2(3) "H 1.585 < 2
B. Slower than O(n^2) since log_2(3) > 2
C. Exactly O(n^2)
D. Incomparable to O(n^2)

Answer: A

Q965. What is the smoothed complexity analysis of an algorithm?

A. Analysis after rounding input sizes to powers of 2
B. Analysis of the algorithm on inputs with small random perturbations added to worst-case inputs
C. Analysis that ignores the first n operations
D. Analysis only on uniformly random inputs

Answer: B

Q966. What is the time complexity of T(n) = T(n-1) + T(n-2) + O(1)?

A. O(n)
B. O(n^2)
C. O(2^n)
D. O(n log n)

Answer: C

Q967. What is the difference between output-sensitive and input-sensitive algorithms?

A. Output-sensitive algorithms depend on both input and output size, while input-sensitive depend only on input size
B. They are the same thing
C. Output-sensitive algorithms are always faster
D. Input-sensitive algorithms cannot be analyzed

Answer: A

Q968. What is the complexity of T(n) = T(" n) + O(1) after the substitution m = log n?

A. O(log n)
B. O(log log n)
C. O(" n)
D. O(n)

Answer: B

Q969. What does the Extended Church-Turing thesis imply about algorithm complexity?

A. All problems can be solved in polynomial time
B. Any model of computation can be simulated by a Turing machine with at most a polynomial slowdown
C. Quantum computers can solve NP-complete problems efficiently
D. Recursion is always faster than iteration

Answer: B

Q970. What is the worst-case space complexity of a recursive algorithm with branching factor b and depth d?

A. O(b^d)
B. O(b * d)
C. O(d)
D. O(b + d)

Answer: C

Q971. What is the time complexity of sorting a stack using only one additional stack?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(2^n)

Answer: C

Q972. What is the time complexity of the LRU cache eviction using a doubly linked list and a hash map?

A. O(n) for get and O(1) for put
B. O(1) for both get and put
C. O(log n) for both get and put
D. O(n) for both get and put

Answer: B

Q973. What is a self-organizing list using the move-to-front heuristic?

A. A list that sorts elements on each access
B. A list that moves the most recently accessed element to the front
C. A list that randomly rearranges elements
D. A list that deletes rarely accessed elements

Answer: B

Q974. What is the worst-case time complexity of the skip list's insertion operation?

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)

Answer: C

Q975. How does a lock-free concurrent queue typically achieve thread safety?

A. Using mutexes on every operation
B. Using Compare-And-Swap (CAS) atomic operations
C. By making copies of the entire queue for each thread
D. By allowing only one thread to use the queue

Answer: B

Q976. What is the amortized time complexity of the multipop operation on a stack that pops k elements?

A. O(k) amortized per multipop
B. O(1) amortized per element across all operations
C. O(n) amortized per multipop
D. O(k^2) amortized per multipop

Answer: B

Q977. What problem does the 'Next Greater Element' algorithm solve using a stack?

A. Finding the maximum element in an array
B. For each element, finding the next element in the array that is strictly greater
C. Sorting the array in descending order
D. Finding the median of the array

Answer: B

Q978. What is the cache performance advantage of arrays over linked lists?

A. Arrays support faster deletion
B. Arrays store elements contiguously in memory, enabling better spatial locality and prefetching
C. Arrays use less total memory
D. Arrays have faster pointer operations

Answer: B

Q979. What is the maximum number of elements that can be stored in a circular queue of array size n?

A. n
B. n - 1
C. n + 1
D. 2n

Answer: B

Q980. What is the time complexity of flattening a multilevel doubly linked list?

A. O(n) where n is the total number of nodes across all levels
B. O(n^2)
C. O(n log n)
D. O(2^n)

Answer: A

Q981. What is the minimum number of nodes in a Red-Black tree of height h (counting black height)?

A. 2^h - 1
B. 2^(h/2)
C. h + 1
D. 2^h

Answer: A

Q982. What is the time complexity of checking if an undirected graph is bipartite?

A. O(V^2)
B. O(V + E)
C. O(V * E)
D. O(2^V)

Answer: B

Q983. How many distinct binary trees can be formed with n nodes?

A. n!
B. 2^n
C. The nth Catalan number C(2n, n)/(n+1)
D. n^2

Answer: C

Q984. What is the time complexity of building a min-heap from an array of n elements using the bottom-up approach?

A. O(n log n)
B. O(n^2)
C. O(n)
D. O(log n)

Answer: C

Q985. What is a multigraph?

A. A graph with multiple connected components
B. A graph that allows multiple edges between the same pair of vertices
C. A graph where all vertices have the same degree
D. A graph that is both directed and undirected

Answer: B

Q986. What is the constant factor in the upper bound of AVL tree height relative to log2(n)?

A. Exactly 1.0
B. Approximately 1.44
C. Approximately 2.0
D. Approximately 0.5

Answer: B

Q987. What is the minimum number of edges to remove to make a connected graph a tree?

A. V - 1
B. E - V + 1
C. E - 1
D. V

Answer: B

Q988. What property does a B-tree of order m guarantee about node occupancy?

A. Every node has exactly m children
B. Every non-root internal node has at least ceil(m/2) children
C. Every node has at most 2 children
D. Leaf nodes store no keys

Answer: B

Q989. What is the isomorphism problem for trees?

A. Checking if two trees have the same number of nodes
B. Determining if two trees are structurally identical by finding a bijection between their nodes
C. Finding the largest common subtree
D. Checking if a tree is balanced

Answer: B

Q990. What is the graph density and how is it calculated for an undirected simple graph?

A. The number of vertices divided by the number of edges
B. 2E / (V(V-1)), representing the ratio of actual edges to maximum possible edges
C. E * V
D. V^2 / E

Answer: B

Q991. What is the worst-case time complexity of the Quickselect algorithm?

A. O(n)
B. O(n log n)
C. O(n^2)
D. O(log n)

Answer: C

Q992. What is the Median of Medians algorithm used for?

A. Sorting arrays efficiently
B. Guaranteeing O(n) worst-case selection of the k-th smallest element
C. Finding the mode of an array
D. Building a balanced binary search tree

Answer: B

Q993. How can you search in a row-wise and column-wise sorted 2D matrix in optimal time?

A. Binary search every row: O(n log n)
B. Start from the top-right corner and eliminate a row or column at each step: O(n + m)
C. Linear search: O(n * m)
D. Sort the entire matrix first: O(n * m * log(n * m))

Answer: B

Q994. What is the time complexity of searching for a key in a sorted and rotated array using modified binary search?

A. O(n)
B. O(n log n)
C. O(log n)
D. O(" n)

Answer: C

Q995. What is a van Emde Boas tree's search time complexity?

A. O(log n)
B. O(log log U) where U is the universe size
C. O(1)
D. O(" n)

Answer: B

Q996. What is the lower bound on the number of comparisons needed for searching in a sorted array?

A. O(1)
B. O(log n)
C. :'†Æör â' –â F†R porst case for comparison-based algorithms
D. O(n)

Answer: C

Q997. What is the time complexity of searching in a balanced k-d tree with n points in d dimensions?

A. O(n)
B. O(log n) always
C. O(n^(1-1/d) + k) for range search returning k points
D. O(d * n)

Answer: C

Q998. What is the fractional cascading technique and what speedup does it provide?

A. It sorts fractional numbers faster
B. It speeds up searching the same key in k sorted lists from O(k log n) to O(k + log n)
C. It divides the search space into fractions
D. It cascades hash functions for faster lookup

Answer: B

Q999. What is the expected search time in a skip list with n elements?

A. O(n)
B. O(" n)
C. O(log n) with high probability
D. O(1)

Answer: C

Q1000. What is the time complexity of the A* search algorithm in the worst case?

A. O(V + E) always
B. O(b^d) where b is the branching factor and d is the solution depth
C. O(V log V)
D. O(E log V)

Answer: B

Q1001. What is the time complexity of TimSort in the worst case?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(n " n)

Answer: B

Q1002. What is the worst-case time complexity of introsort?

A. O(n^2)
B. O(n log n)
C. O(n log^2 n)
D. O(n)

Answer: B

Q1003. What is the information-theoretic lower bound for comparison-based sorting expressed in terms of comparisons?

A. n comparisons
B. n log n comparisons
C. # log2(n!)# comparisons, which is 9,†â Æör â•
D. n^2 comparisons

Answer: C

Q1004. What is the three-way partitioning (Dutch National Flag) algorithm's advantage for quicksort?

A. It eliminates the need for recursion
B. It handles arrays with many duplicate elements efficiently by partitioning into three groups: less, equal, and greater
C. It always selects the optimal pivot
D. It reduces space complexity to O(1) for merge sort

Answer: B

Q1005. What is external sorting and when is it used?

A. Sorting data stored in external variables
B. Sorting large datasets that do not fit in main memory by using disk-based merge strategies
C. Sorting data received from external APIs
D. Sorting arrays using an external library

Answer: B

Q1006. What is the time complexity of patience sorting?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(n " n)

Answer: B

Q1007. How does block sort (WikiSort) achieve O(n log n) sorting with O(1) extra memory and stability?

A. By using quicksort internally
B. By using block-based merging strategies that rotate and swap fixed-size blocks in-place
C. By sacrificing stability for space efficiency
D. By using hash tables for comparisons

Answer: B

Q1008. What is the optimal number of comparisons needed to sort 5 elements?

A. 5
B. 7
C. 8
D. 10

Answer: B

Q1009. What is the worst-case time complexity of the best known gap sequence for shell sort?

A. O(n^2)
B. O(n log^2 n)
C. O(n^(4/3))
D. O(n log n)

Answer: C

Q1010. What is a sorting network and what is the depth of the optimal sorting network for n inputs?

A. A neural network for sorting with O(n) depth
B. A fixed sequence of comparators; optimal depth is O(log^2 n) by the AKS network
C. A graph that represents sorting comparisons; depth is O(n)
D. A distributed system for parallel sorting; depth is O(n log n)

Answer: B

Q1011. What is the expected maximum chain length in a hash table with n elements and n slots using universal hashing?

A. O(1)
B. O(log n / log log n)
C. O(" n)
D. O(n)

Answer: B

Q1012. How does cuckoo hashing achieve O(1) worst-case lookup?

A. By using a single hash function with a large table
B. By using two hash functions and two tables, where each key is in exactly one of two possible locations
C. By sorting the keys before inserting
D. By maintaining a sorted linked list at each slot

Answer: B

Q1013. What is the false positive rate of a Bloom filter with m bits, k hash functions, and n inserted elements?

A. (1 - e^(-kn/m))^k approximately
B. k/m
C. n/m
D. 1 - (1 - 1/m)^n

Answer: A

Q1014. What is consistent hashing and why is it important in distributed systems?

A. A hash function that always returns the same value
B. A technique where adding or removing a server only requires remapping O(K/n) keys on average, where K is keys and n is servers
C. Hashing that guarantees zero collisions
D. A hash function that works across different programming languages

Answer: B

Q1015. What is the amortized insertion time of cuckoo hashing?

A. O(1) amortized
B. O(log n) amortized
C. O(n) amortized
D. O(" n) amortized

Answer: A

Q1016. What is a minimal perfect hash function?

A. A hash function with minimal computation
B. A hash function that maps n keys to n consecutive integers with no collisions
C. The simplest possible hash function
D. A hash function that only works for small inputs

Answer: B

Q1017. What is the count-min sketch data structure used for?

A. Exact counting of elements
B. Approximate frequency estimation of elements in a data stream using sub-linear space
C. Sorting elements by frequency
D. Storing minimum and maximum values

Answer: B

Q1018. What is tabulation hashing and what property does it guarantee?

A. A hash function stored in a table that guarantees O(n log n) sorting
B. A hash function that XORs values from lookup tables for each byte of the key, providing 3-wise independence
C. A hashing method that creates a truth table
D. A method of hashing that only works for table-structured data

Answer: B

Q1019. What is the space complexity of a Bloom filter achieving false positive rate ;R v—F, â VÆVÖVçG3ð

A. O(n)
B. O(n log(1/;R'•
C. O(1/;R•
D. O(n^2)

Answer: B

Q1020. What is Robin Hood hashing's key invariant?

A. Each slot has exactly one element
B. Elements with longer probe distances displace elements with shorter probe distances to equalize probe lengths
C. The hash table is always sorted
D. Each element is hashed twice

Answer: B

Q1021. What is the time complexity of LCA computation using Euler tour and sparse table after O(n log n) preprocessing?

A. O(log n) per query
B. O(1) per query
C. O(n) per query
D. O(" n) per query

Answer: B

Q1022. What is Heavy-Light Decomposition (HLD) used for?

A. Balancing binary trees
B. Decomposing a tree into chains so that any root-to-leaf path crosses O(log n) chains, enabling efficient path queries
C. Finding the heaviest subtree
D. Removing heavy nodes from the tree

Answer: B

Q1023. Without lazy propagation, what is the time complexity of a range update on a segment tree, and how does lazy propagation improve it?

A. O(n) without, O(log n) with lazy propagation
B. O(n log n) without, O(n) with lazy propagation
C. O(log n) without, O(1) with lazy propagation
D. Both are O(n)

Answer: A

Q1024. What is the amortized time complexity of splay tree operations?

A. O(1)
B. O(log n)
C. O(n)
D. O(" n)

Answer: B

Q1025. What is a Fenwick tree's advantage over a segment tree?

A. Fenwick trees support more operations
B. Fenwick trees use less memory (n elements vs 4n) and have smaller constant factors while supporting prefix operations
C. Fenwick trees have better worst-case complexity
D. Fenwick trees support range updates natively

Answer: B

Q1026. How much extra space does each update operation use in a persistent segment tree?

A. O(1)
B. O(n)
C. O(log n)
D. O(n log n)

Answer: C

Q1027. What is the Centroid Decomposition of a tree used for?

A. Finding the center of mass of weighted nodes
B. Recursively finding and removing centroids to create a decomposition tree enabling efficient path-based queries
C. Balancing unbalanced trees
D. Finding the median node

Answer: B

Q1028. What is the time complexity of range update and point query in a Fenwick tree?

A. O(n) for each
B. O(log n) for each
C. O(1) for update, O(n) for query
D. O(n log n) for each

Answer: B

Q1029. What is the time complexity of constructing an optimal BST given n keys with known access probabilities?

A. O(n)
B. O(n^2)
C. O(n^3)
D. O(2^n)

Answer: C

Q1030. How does a treap maintain both BST and heap properties simultaneously?

A. By using two separate trees
B. Each node has a key (BST-ordered) and a random priority (heap-ordered); rotations maintain both properties
C. By sorting elements twice
D. By storing elements in two arrays

Answer: B

Q1031. What is the time complexity of Tarjan's algorithm for finding strongly connected components?

A. O(V^2)
B. O(V + E)
C. O(V * E)
D. O(V^3)

Answer: B

Q1032. According to the duality principle in network flow, what is equal to the capacity of the minimum s-t cut?

A. The number of edges in the graph
B. The maximum flow from source s to sink t
C. The minimum spanning tree weight
D. The number of vertices on the shortest path

Answer: B

Q1033. What is the time complexity of the Edmonds-Karp algorithm for maximum flow?

A. O(V * E)
B. O(V * E^2)
C. O(V^2 * E)
D. O(V^3)

Answer: C

Q1034. What is an Euler path in a graph?

A. A path that visits every vertex exactly once
B. A path that traverses every edge exactly once
C. The shortest path between two vertices
D. A path with minimum total weight

Answer: B

Q1035. What is the time complexity of finding all articulation points in an undirected graph?

A. O(V^2)
B. O(V + E)
C. O(V * E)
D. O(E^2)

Answer: B

Q1036. What is Johnson's algorithm for all-pairs shortest paths?

A. Running Dijkstra from every vertex without preprocessing
B. Reweighting edges using Bellman-Ford to eliminate negative weights, then running Dijkstra from every vertex
C. Using Floyd-Warshall on a reduced graph
D. A divide-and-conquer approach for shortest paths

Answer: B

Q1037. What is a Hamiltonian path and why is finding one computationally hard?

A. A shortest path in a weighted graph, solvable in polynomial time
B. A path visiting every vertex exactly once; the decision problem is NP-complete
C. A path traversing every edge exactly once; solvable in O(V + E)
D. The longest path in a tree

Answer: B

Q1038. What is the Dinic's algorithm time complexity for maximum flow?

A. O(V * E)
B. O(V^2 * E)
C. O(V * E^2)
D. O(V^3)

Answer: B

Q1039. What is the 2-SAT problem and how is it related to graph algorithms?

A. Finding two satisfying assignments using brute force
B. A satisfiability problem solvable in polynomial time by building an implication graph and finding SCCs
C. A graph coloring problem
D. An NP-complete problem with no efficient solution

Answer: B

Q1040. What is the time complexity of finding a minimum spanning tree using Bor ÷f¶ w2 Æv÷&—F†Óð

A. O(V^2)
B. O(E log V)
C. O(V + E)
D. O(E log E)

Answer: B

Q1041. What is the time complexity of the matrix chain multiplication DP solution?

A. O(n^2)
B. O(n^3)
C. O(2^n)
D. O(n log n)

Answer: B

Q1042. How does the meet-in-the-middle technique reduce the Subset Sum time from O(2^n) to approximately O(2^(n/2))?

A. By sorting the subsets before checking
B. By splitting elements into two halves, enumerating all sums in each half, then combining using binary search or hashing
C. By using dynamic programming instead of brute force
D. By reducing the problem to a graph problem

Answer: B

Q1043. What is the difference between Las Vegas and Monte Carlo randomized algorithms?

A. Las Vegas algorithms are deterministic
B. Las Vegas algorithms always produce correct results (with variable runtime), while Monte Carlo may produce incorrect results (with bounded runtime)
C. Monte Carlo algorithms are always faster
D. They are the same type of algorithm

Answer: B

Q1044. What is the time complexity of solving the edit distance problem using DP?

A. O(n)
B. O(n * m)
C. O(n^3)
D. O(2^n)

Answer: B

Q1045. Which of the following problems violates Bellman's principle of optimality and therefore cannot be solved by standard DP?

A. Shortest path problem
B. Longest simple path in a general graph
C. Matrix chain multiplication
D. Edit distance

Answer: B

Q1046. What is the time complexity of the Held-Karp algorithm for the Traveling Salesman Problem?

A. O(n!)
B. O(n^2 * 2^n)
C. O(n^3)
D. O(2^n)

Answer: B

Q1047. What is the Knuth optimization for DP?

A. Using hash tables for memoization
B. An optimization for DP problems with the property that the optimal split point is monotone, reducing O(n^3) to O(n^2)
C. Parallelizing DP computations
D. Converting recursion to iteration

Answer: B

Q1048. What is the Convex Hull Trick optimization in dynamic programming?

A. Finding the convex hull of points
B. An optimization for DP transitions of the form dp[i] = min(dp[j] + b[j] * a[i]) using a set of lines
C. Computing convex combinations of solutions
D. A geometric algorithm for nearest neighbors

Answer: B

Q1049. What is the difference between online and offline algorithms?

A. Online algorithms require internet access
B. Online algorithms process input piece by piece as it arrives, while offline algorithms have access to the entire input upfront
C. Offline algorithms are always slower
D. Online algorithms cannot be analyzed

Answer: B

Q1050. What is the time complexity of solving the Subset Sum problem using DP?

A. O(n * S) where S is the target sum (pseudo-polynomial)
B. O(2^n)
C. O(n^2)
D. O(n log n)

Answer: A

Q1051. What is a van Emde Boas tree's space complexity?

A. O(n)
B. O(U) where U is the universe size
C. O(log U)
D. O(n log n)

Answer: B

Q1052. What is a rope data structure and what advantage does it offer over arrays for string operations?

A. A compressed string format
B. A balanced binary tree of strings that supports O(log n) concatenation, split, and insertion instead of O(n) for arrays
C. A linked list of characters
D. An encrypted string storage

Answer: B

Q1053. What is the order-statistic tree and what additional information does it maintain?

A. A tree that only stores statistics
B. An augmented BST that stores subtree sizes to support O(log n) rank and select queries
C. A tree that orders nodes by creation time
D. A tree that counts the number of queries

Answer: B

Q1054. What is a wavelet tree used for?

A. Audio signal processing
B. Answering rank, select, and range frequency queries on sequences in O(log <2’ F–ÖR v†W&R Ã is the alphabet size
C. Compressing images
D. Sorting wavelets

Answer: B

Q1055. Which underlying data structure does the link-cut tree use internally to maintain preferred paths?

A. Red-Black trees
B. AVL trees
C. Splay trees
D. B-trees

Answer: C

Q1056. How much space does a succinct data structure use relative to the information-theoretic minimum Z?

A. O(Z) bits
B. Z + o(Z) bits, meaning within a lower-order additive term of the minimum
C. Exactly Z bits
D. 2Z bits

Answer: B

Q1057. What is a suffix tree and how does it relate to a suffix array?

A. They are completely unrelated data structures
B. A suffix tree is a compressed trie of all suffixes; a suffix array stores the same suffixes sorted, and they can be converted between in O(n) time
C. A suffix array is always preferred over a suffix tree
D. A suffix tree stores prefix information while a suffix array stores suffix information

Answer: B

Q1058. What is the time complexity of the decrease-key operation in a Fibonacci heap?

A. O(log n) amortized
B. O(1) amortized
C. O(n)
D. O(log log n)

Answer: B

Q1059. What is an x-fast trie and what time complexity does it achieve for predecessor queries?

A. O(log n)
B. O(log log U) using hash tables at each level of a binary trie over the universe
C. O(1)
D. O(" n)

Answer: B

Q1060. What is the weight-balanced tree's balance condition?

A. Left and right subtrees have equal height
B. The ratio of sizes of left and right subtrees is bounded by a constant ; Â v†W&R óB &B ± "d 1-1/" 2
C. Each node has the same weight
D. The total weight equals n

Answer: B

Q1061. What is the time complexity of Manacher's algorithm for finding the longest palindromic substring?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(n^3)

Answer: C

Q1062. What is the Aho-Corasick algorithm's time complexity for matching k patterns of total length m against a text of length n?

A. O(n * k)
B. O(n + m + z) where z is the number of matches
C. O(n * m)
D. O(k * m * n)

Answer: B

Q1063. What is the suffix automaton of a string?

A. An automaton that accepts only the suffixes of the string
B. A minimal DFA that accepts exactly all substrings of the string, built in O(n) time and space
C. An automaton that generates all permutations
D. A finite automaton for sorting strings

Answer: B

Q1064. What is the Burrows-Wheeler Transform (BWT) primarily used for?

A. Encrypting strings for security
B. Rearranging a string to group similar characters together, enabling better compression
C. Sorting strings in reverse order
D. Finding palindromes in strings

Answer: B

Q1065. What key optimization technique does Ukkonen's suffix tree algorithm use to achieve linear time?

A. Hash tables for each node
B. Suffix links, implicit extensions, and the active point trick to avoid redundant traversal
C. Divide and conquer on the string
D. Binary search on suffix positions

Answer: B

Q1066. How can the number of distinct substrings of a string be computed using a suffix array and LCP array?

A. Count = n(n+1)/2
B. Count = n(n+1)/2 - sum of all LCP values
C. Count = sum of all LCP values
D. Count = n^2

Answer: B

Q1067. What is the worst-case time complexity of the Boyer-Moore algorithm?

A. O(n/m)
B. O(n + m)
C. O(n * m)
D. O(n)

Answer: C

Q1068. What is the LCP (Longest Common Prefix) array and how is it used with suffix arrays?

A. An array of string lengths
B. LCP[i] stores the length of the longest common prefix between the i-th and (i-1)-th sorted suffixes; it enables efficient pattern matching and string analysis
C. An array of character frequencies
D. An array mapping characters to positions

Answer: B

Q1069. What is the time complexity of the Kasai algorithm for constructing the LCP array from a suffix array?

A. O(n^2)
B. O(n log n)
C. O(n)
D. O(n * m)

Answer: C

Q1070. What is the palindromic tree (Eertree) data structure?

A. A binary tree of palindrome lengths
B. A data structure that stores all distinct palindromic substrings of a string in O(n) time and space
C. A trie of reversed strings
D. A balanced BST of palindromes

Answer: B

Q1071. What is Cook's theorem and its significance?

A. It proves P = NP
B. It proves that the Boolean satisfiability problem (SAT) is NP-complete, establishing the first NP-complete problem
C. It proves that all NP problems are unsolvable
D. It provides a polynomial algorithm for SAT

Answer: B

Q1072. What is pseudo-polynomial time and why is it misleading?

A. It is polynomial time disguised as exponential
B. An algorithm runs in time polynomial in the numeric value of the input but exponential in the input size (number of bits)
C. It is always faster than polynomial time
D. It means the algorithm is approximately polynomial

Answer: B

Q1073. What is the complexity class co-NP?

A. Problems whose complements are in P
B. Problems whose complements are in NP; equivalently, problems with polynomial-time disproofs
C. Problems harder than NP
D. Problems that are not in NP

Answer: B

Q1074. What fundamental consequence does the time hierarchy theorem have for the P vs NP question?

A. It proves P = NP
B. It proves that strictly more time allows solving strictly more problems, implying P "` EXPTIME
C. It shows all problems are equally hard
D. It proves NP = PSPACE

Answer: B

Q1075. What is the best known time complexity for matrix multiplication of two n×n matrices?

A. O(n^3)
B. O(n^2.807) (Strassen)
C. O(n^2.371) approximately (current best)
D. O(n^2)

Answer: C

Q1076. What is a Fixed-Parameter Tractable (FPT) algorithm?

A. An algorithm with fixed running time
B. An algorithm with time O(f(k) * n^c) where k is a parameter, f is any computable function, and c is a constant independent of k
C. An algorithm that fixes errors in parameters
D. An algorithm with O(n) time for any parameters

Answer: B

Q1077. What is the role of polynomial-time reductions in proving NP-completeness?

A. They simplify the problem to make it solvable
B. They show that if problem A reduces to problem B in polynomial time and A is NP-hard, then B is also NP-hard
C. They reduce the time complexity of algorithms
D. They convert NP problems to P problems

Answer: B

Q1078. What is the PSPACE complexity class?

A. Problems solvable in polynomial time
B. Problems solvable using polynomial space on a deterministic Turing machine
C. Problems that use no space at all
D. Problems solvable on parallel computers

Answer: B

Q1079. What is the approximation ratio and when is it used?

A. The ratio of input size to output size
B. For NP-hard optimization problems, the ratio of the algorithm's solution value to the optimal solution value, guaranteed to be within a bound
C. The ratio of time complexity to space complexity
D. The ratio of average case to worst case complexity

Answer: B

Q1080. What is the Exponential Time Hypothesis (ETH) and what does it imply?

A. All problems require exponential time
B. 3-SAT cannot be solved in 2^(o(n)) time, implying many NP-hard problems have no sub-exponential algorithms
C. P = NP is true
D. Exponential algorithms are always impractical

Answer: B