Mapping File Blocks on Disk Platter (Free Disk Space Management):

- Files are stored on disks and require allocation of disk space.
- The operating system maintains a Disk Allocation Table (DAT) to track free disk space.
- When a file is created, the OS searches the free space list and allocates space for the file.
- When a file is deleted, its space is added back to the free space list.
- DAT helps to manage disk space efficiently and avoid disk fragmentation.

Linked Free Space List:

- Free blocks are linked together using a linked list of disk blocks.
- Each disk block holds a maximum number of free blocks that can fit.
- Free blocks are used to store the free list.
- During allocation, the first block in the free space list is allocated.
- No disk fragmentation occurs.
- Reading each block can be time-consuming.
- Pointers used in the linked list consume disk space.

Bitmap or Bit Vector:

- A bitmap is a memory allocation technique that represents each block on a storage device with a single bit.
- A value of 1 represents a free block, and a value of 0 represents an allocated block.
- A disk with n blocks requires a bitmap with n bits.
- Bitmaps require less space and are simple and efficient.
- However, they may require hardware support to perform bit operations.

File System Performance:

- Disk access is slower than memory access due to seek time and latency time.
- Disk performance is worse than memory performance.
- Disk performance can be improved by using techniques such as block caching, block read-ahead, and reducing disk arm motion.

MULTIPLE CHOICE QUESTIONS

Data Structure

1. **What is the difference between a primitive data type and an abstract data type in programming?**
   A. Primitive data types are basic and cannot be changed, while abstract data types can be modified.
   B. Primitive data types are simple, while abstract data types are complex.
   C. Primitive data types are defined by the programming language, while abstract data types are defined by the programmer.
   D. Primitive data types are stored in memory, while abstract data types are stored on disk.
2. **Which of the following is a linear data structure?**
   A. Graph B. Queue
   C. Tree D. Hash Table
3. **In the big O notation, O (1) represents:**
   A. A constant time algorithm
   B. A linear time algorithm
   C. A logarithmic time algorithm
   D. An exponential time algorithm
4. **What does big O notation represent in algorithm analysis?**
   A. The worst-case time complexity of an algorithm
   B. The best-case time complexity of an algorithm
   C. The average time complexity of an algorithm
   D. The exact time complexity of an algorithm
5. **What does a big O notation of O(n) indicate about an algorithm?**
   A. The algorithm takes a constant amount of time regardless of the size of the input.
   B. The algorithm takes a linear amount of time proportional to the size of the input.
   C. The algorithm takes a logarithmic amount of time relative to the size of the input.
   D. The algorithm takes an exponential amount of time based on the size of the input.
6. **What is the big O notation for a binary search algorithm?**
   A. O(1) B. O(n)
   C. O(log n) D. O(n log n)
7. **Which of the following algorithms has a big O notation of O(n^2)?**
   A. Binary search B. Quick sort
   C. Bubble sort D. Linear search
8. **What does big O notation tell us about the efficiency of an algorithm?**
   A. It gives us an exact measurement of the time complexity of an algorithm.
   B. It gives us an approximation of the time complexity of an algorithm, taking into account the worst-case scenario.
   C. It gives us an estimate of the space complexity of an algorithm.
   D. It gives us a comparison of the time complexity of different algorithms.

9. **What is a linear data structure?**
   A. A data structure that stores data in a sequential manner
   B. A data structure that stores data in a hierarchical manner
   C. A data structure that stores data randomly
   D. A data structure that stores data in a circular manner

10. **Which of the following is an example of a linear data structure?**
   A. Tree
   B. Stack
   C. Graph
   D. Hash Table

11. **What is the difference between a stack and a queue in terms of data storage and retrieval?**
   A. A stack stores and retrieves data in a first-in, first-out (FIFO) manner, while a queue stores and retrieves data in a last-in, first-out (LIFO) manner.
   B. A stack stores and retrieves data in a last-in, first-out (LIFO) manner, while a queue stores and retrieves data in a first-in, first-out (FIFO) manner.
   C. A stack stores and retrieves data randomly, while a queue stores and retrieves data in a sequential manner.
   D. A stack stores and retrieves data in a hierarchical manner, while a queue stores and retrieves data in a circular manner.

12. **What is the time complexity of inserting or deleting an element from the beginning of a singly linked list?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n log n)

13. **What is the time complexity of inserting or deleting an element from the end of a linked list?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n log n)

14. **What is a data structure?**
   A. A collection of related data values
   B. A collection of algorithms
   C. A way of organizing and storing data in a computer so that it can be accessed and manipulated efficiently
   D. A way of organizing and storing data in a database

15. **What are the two main types of data structures?**
   A. Linear and non-linear
   B. Hierarchical and non-hierarchical
   C. Primitive and non-primitive
   D. Structured and unstructured

16. **What is an array in data structures?**
   A. A collection of data elements stored in a sequential manner
   B. A collection of data elements stored in a random manner
   C. A collection of data elements stored in a hierarchical manner
   D. A collection of data elements stored in a circular manner

17. **What is a linked list in data structures?**
   A. A collection of data elements stored in a sequential manner
   B. A collection of data elements stored in a random manner
   C. A collection of data elements stored in a hierarchical manner
   D. A linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list

18. **What is a stack in data structures?**
   A. A collection of data elements stored in a sequential manner
   B. A collection of data elements stored in a random manner
   C. A collection of data elements stored in a hierarchical manner
   D. A collection of data elements stored in a last-in, first-out (LIFO) manner

19. **Which of the following is/are the application of stack?**
   A. Backtracking
   B. Evaluation of arithmetic expression
   C. Function calls
   D. All of the above

20. **What is the time complexity of pushing an element onto a stack?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n log n)

21. **What is the time complexity of popping an element from a stack?**
   A. O(1)
   B. O(n)
   C. O(log n)
   D. O(n log n)

22. **What is the use of a stack in computer science?**
   A. To store and retrieve data in a first-in, first-out (FIFO) manner
   B. To store and retrieve data in a last-in, first-out (LIFO) manner
   C. To store and retrieve data randomly
   D. To store and retrieve data in a circular manner

23. **What is an example of a problem that can be solved using a stack?**
   A. Sorting a list of numbers
   B. Finding the shortest path between two points
   C. Balancing parentheses in a mathematical expression
   D. Searching for a specific element in a list

24. **What is the difference between a stack and a queue?**
   A. A stack stores and retrieves data in a last-in, first-out (LIFO) manner, while a queue stores and retrieves data in a first-in, first-out (FIFO) manner.
   B. A stack stores and retrieves data randomly, while a queue stores and retrieves data in a sequential manner.
   C. A stack is a linear data structure, while a queue is a non-linear data structure.
   D. A stack is a dynamic data structure, while a queue is a static data structure.

25. **What is the advantage of using a stack over other data structures to solve certain problems?**
   A. Stacks are more efficient in terms of time and space complexity.
   B. Stacks are easier to implement and use than other data structures.
   C. Stacks allow for efficient retrieval of data in a last-in, first-out (LIFO) manner, making them well-suited for certain problems.
   D. Stacks provide better performance in terms of memory usage compared to other data structures.

26. **What is a stack overflow error in computer programming?**
   A. An error that occurs when a stack exceeds its maximum size.
   B. An error that occurs when a stack is not properly initialized.
   C. An error that occurs when a stack is not properly deallocated.
   D. An error that occurs when a stack is not properly aligned in memory.

27. **What is the difference between a stack and a heap in computer programming?**
   A. A stack stores data in a last-in, first-out (LIFO) manner, while a heap stores data in a random manner.
   B. A stack is a static data structure, while a heap is a dynamic data structure.
   C. A stack is used for short-term storage of data, while a heap is used for long-term storage of data.
   D. A stack is used for program execution, while a heap is used for memory allocation.

28. **What is a stack frame in computer programming?**
   A. A unit of memory that stores data for a single function call in a stack.
   B. A unit of memory that stores data for a single data structure in a stack.
   C. A unit of memory that stores data for multiple function calls in a stack.
   D. A unit of memory that stores data for multiple data structures in a stack.

29. **What is the push operation in a stack data structure?**
   A. A operation to remove an element from the top of the stack
   B. A operation to add an element to the bottom of the stack
   C. A operation to add an element to the top of the stack
   D. A operation to remove an element from the bottom of the stack

30. **What is the pop operation in a stack data structure?**
   A. A operation to remove an element from the top of the stack
   B. A operation to add an element to the bottom of the stack
   C. A operation to add an element to the top of the stack
   D. A operation to remove an element from the bottom of the stack

31. **What is stack overflow in a stack data structure?**
   A. An error that occurs when a stack exceeds its maximum size
   B. An error that occurs when a stack is not properly initialized
   C. An error that occurs when a stack is not properly deallocated
   D. An error that occurs when a stack is not properly aligned in memory

32. **What is stack underflow in a stack data structure?**
   A. An error that occurs when a stack is empty and an attempt is made to remove an element from it
   B. An error that occurs when a stack is full and an attempt is made to add an element to it
   C. An error that occurs when a stack is improperly aligned in memory
   D. An error that occurs when a stack is not properly initialized

33. **What is a queue in data structures?**
   A. Sequential collection with insertion at one end and deletion at the other end
   B. Random collection
   C. Hierarchical collection
   D. Circular collection with pointers

34. **What is the time complexity of inserting an element into a queue?**
   A. O(1) B. O(n)
   C. O(log n) D. O(n log n)

35. **What is the time complexity of removing an element from a queue?**
   A. O(1) B. O(n)
   C. O(log n) D. O(n log n)

36. **What is the use of a queue in computer science?**
   A. To store and retrieve data in a last-in, first-out (LIFO) manner
   B. To store and retrieve data in a first-in, first-out (FIFO) manner
   C. To store and retrieve data randomly
   D. To store and retrieve data in a circular manner

37. **What is an example of a problem that can be solved using a queue?**
   A. Sorting a list of numbers
   B. Breadth-first search in a graph
   C. Balancing parentheses in a mathematical expression
   D. Searching for a specific element in a list

38. **What is the minimum number of fields in each node of a doubly linked list?**
   A. 2
   B. 3
   C. 4
   D. None of the above

39. **What is a graph referred to as when all vertices have equal degree?**
   A. Complete graph B. Regular graph
   C. Multi graph D. Simple graph

40. **What is a vertex with in-degree zero in a directed graph called?**
   A. Root vertex
   B. Isolated vertex
   C. Sink
   D. Articulation point

41. **What conditions must a graph meet to be considered a tree?**
   A. It must be a directed graph
   B. It must contain no cycles
   C. It must be planar
   D. It must be completely connected

42. **Where are the elements of a linked list stored?**
   A. In a structure
   B. In an array
   C. Anywhere the computer has space for them
   D. In contiguous memory locations

43. **What data structure would be best suited for implementing a parentheses checker program?**
   A. List
   B. Queue
   C. Stack
   D. Any of the above

44. **What data structure is required to perform level-order traversal on a binary tree?**
   A. Hash table B. Queue
   C. Binary search tree D. Stack

45. **What data structure is required to convert an arithmetic expression in infix to its equivalent postfix notation?**
   A. Queue
   B. Linked list
   C. Binary search tree
   D. None of above

46. **What type of binary tree has all levels except the last filled with the maximum number of nodes and all nodes in the last level having only one child, which is its left child?**
   A. Threaded tree
   B. Complete binary tree
   C. M-way search tree
   D. Full binary tree

47. What data structure is most appropriate for implementing quick sort iteratively?
A. Deque B. Queue
C. Stack D. Priority queue

48. What is the number of edges in a complete graph of n vertices?
A. n(n+1)/2 B. n(n-1)/2
C. n2/2 D. n

49. What is the space complexity of bubble sort?
A. O(1)
B. O(n^2)
C. O(2)
D. None of the above

50. What are two trees called if they have the same structure and node content?
A. Synonyms trees B. Joint trees
C. Equivalent trees D. Similar trees

51. What is the process of finding the location of a given i tem in a collection of items called?
A. Discovering B. Finding
C. Searching D. Mining

52. What is the time complexity of quicksort?
A. O(n) B. O(logn)
C. O(n^2) D. O(n logn)

53. What is quick sort also known as?
A. Merge sort
B. Tree sort
C. Shell sort
D. Partition and exchange sort

54. What sorting method is good for alphabetizing a large list of names?
A. Merge B. Heap
C. Radix D. Bubble

55. What is the total number of comparisons in a bubble sort?
A. O(n logn) B. $O(2_n)$
C. O(n^2) D. $O_{(n)}$

56. What form of access is used to add and remove nodes from a queue?
A. LIFO, Last In First Out
B. FIFO, First In First Out
C. Both A and B
D. None of these

57. Enqueue is the process of .....
A. Removing data from queue.
B. Adding item into a queue
C. Searching item in queue
D. None of the above

58. What is the term "push and pop" related to?
A. Array B. Lists
C. Stacks D. Trees

59. What is an application of stack?
A. Finding factorial
B. Tower of Hanoi
C. Infix to postfix
D. All of the above

60. What is the operation of processing each element in a list called?
A. Sorting B. Merging
C. Inserting D. Traversal

61. What is the situation when in a linked list, START=NULL called?
A. Underflow B. Overflow
C. Houseful D. Saturated

62. What are two-way lists?
A. Grounded header list
B. Circular header list
C. Linked list with header and trailer nodes
D. List traversed in two directions

63. What is the pointer associated with the availability list?
A. FIRST B. AVAIL
C. TOP D. REAR

64. What data structure cannot store non-homogeneous data elements?
A. Arrays B. Records
C. Pointers D. Stacks

65. What is a non-linear data structure?
A. Stacks B. List
C. Strings D. Trees

66. What data structure is suitable for representing hierarchical relationships between elements?
A. Dequeue B. Priority
C. Tree D. Graph

67. What data structure allows deletions at both ends of the list but only allows insertion at one end?
A. Input restricted dequeue
B. Output restricted queue
C. Priority queues
D. Stack

68. What is the order of visiting nodes in a pre-order tree traversal?
A. Root, Left, Right
B. Left, Root, Right
C. Right, Root, Left
D. Left, Right, Root

69. What is the order of visiting nodes in a post-order tree traversal?
A. Root, Left, Right
B. Left, Root, Right
C. Right, Left, Root
D. Left, Right, Root

70. What is the order of visiting nodes in an in-order tree traversal?
A. Root, Left, Right
B. Left, Root, Right
C. Right, Root, Left
D. Left, Right, Root

71. What is the average time complexity of searching for an element in a binary search tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

72. What is the worst-case time complexity of searching for an element in a binary search tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

73. What is the average time complexity of inserting an element into a binary search tree?
A. O (1) B. O(n)
C. O (log n) D. O (n log n)

74. In a linked list, what is the time complexity of inserting an element at the beginning of the list?
A. O (1) B. O(n)
C. O (log n) D. O (n log n)

75. What is the time complexity of deleting an element from the middle of a linked list?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

76. What is a circular linked list?
A. A linked list where the last node points to the first node
B. A linked list where the first node points to the last node
C. A linked list where the last node points to the second node
D. A linked list where the first node points to the second node

77. What is a doubly linked list?
A. A linked list where each node points to the next node
B. A linked list where each node points to the previous node
C. A linked list where each node points to the next and previous nodes
D. A linked list where the first and last nodes are connected

78. What is the advantage of using a doubly linked list over a singly linked list?
A. A doubly linked list requires less memory
B. A doubly linked list is faster for inserting and deleting elements
C. A doubly linked list allows for traversal in both directions
D. A doubly linked list has better cache performance

79. What is the time complexity of reversing a linked list?
A. O (1) B. O(n)
C. O (log n) D. O (n log n)

80. What is the time complexity of finding an element in a binary tree?
A. O (1)
B. O(n)
C. O (log n)
D. O (n log n)

81. What is the time complexity of inserting an element in a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

82. What is the time complexity of deleting an element in a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

83. What is the time complexity of searching for the minimum element in a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

84. What is the time complexity of searching for the maximum element in a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

85. What is the time complexity of performing an in-order traversal of a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

86. What is the time complexity of performing a pre-order traversal of a binary tree?
A. O(1) B. O(n)
C. O(log n) D. O(n log n)

87. What is the advantage of using the array data structure among the following options?
A. The entities of mixed type of data types can be easily stored
B. Access to elements in the array is relatively easy
C. The index number of the first element always starts from 1
D. None of the above

88. Which among the following is an application of the queue data structure?
A. When a resource is shared among multiple users
B. Load balancing
C. Asynchronous data transfer
D. All of the above

89. Which type of data structure can be used to implement queues among the following options?
A. Linked List
B. Array
C. Stack
D. All of the above

90. Which type of data structure allows for insertion and deletion from both ends?
A. String B. Queue
C. Stack D. Dequeue

91. Which sorting algorithm is used to achieve the best time complexity in the worst-case scenario?
A. Bubble sort B. Selection sort
C. Quick sort D. Merge sort

92. What term is used to describe the scenario when a user tries to remove an element from an empty stack?
A. Underflow
B. Empty collection
C. Overflow
D. Garbage Collection

93. What is the maximum number of non-zero values that can exist in an adjacency matrix of a simple graph with n vertices?
A. (n * (n - 1)) / 2 B. (n * (n + 1)) / 2
C. n * (n - 1) D. n * (n + 1)

94. What is the time complexity of the selection sort algorithm in the best and worst cases?
A. Best: O(n^2), Worst: O(n)
B. Best: O(n log n), Worst: O(n^2)
C. Best: O(n), Worst: O(n^2)
D. Best: O(n^2), Worst: O(n^2)

95. What type of sorting algorithm is the bubble sort algorithm?
A. Linear B. Non-linear
C. Logarithmic D. Exponential

96. What is the space complexity of the merge sort algorithm?
A. O(n) B. O(1)
C. O(n log n) D. O(n^2)

97. What is the time complexity of the quick sort algorithm in the average case?
A. O(n) B. O(n^2)
C. O(n log n) D. O(1)

98. Which sorting algorithm is used to sort linked lists?
A. Insertion sort B. Quick sort
C. Merge sort D. Bubble sort

99. Which sorting algorithm is most efficient for small data sets?
A. Quick sort B. Bubble sort
C. Insertion sort D. Merge sort

100. What is the time complexity of the insertion sort algorithm in the best case?
A. O(n^2) B. O(n log n)
C. O(n) D. O(1)

101. What type of sorting algorithm is the radix sort algorithm?
A. Non-comparison based
B. Comparison based
C. Logarithmic
D. Exponential

102. Which sorting algorithm is used to sort arrays in place?
A. Merge sort B. Quick sort
C. Insertion sort D. Bubble sort

103. What is the time complexity of the shell sort algorithm in the average case?
A. O(n) B. O(n log n)
C. O(n^2) D. O(1)

104. The time complexity of Insertion Sort is:
A. O(n) B. O(n log n)
C. O(n^2) D. O(log n)

105. The time complexity of Merge Sort is:
A. O(n)
B. O(n log n)
C. O(n^2)
D. O(log n)

106. The time complexity of Quick Sort is:

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

107. The time complexity of Heap Sort is:

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

108. The time complexity of Bubble Sort is:

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

109. Which sorting algorithm is used to sort linked lists?

A. Insertion Sort B. Merge Sort
C. Quick Sort D. Heap Sort

110. Which sorting algorithm is used for sorting large data sets?

A. Insertion Sort B. Merge Sort
C. Quick Sort D. Heap Sort

111. Which sorting algorithm is used for sorting arrays with random order?

A. Insertion Sort B. Merge Sort
C. Quick Sort D. Heap Sort

112. Which sorting algorithm is used for sorting arrays with sorted or nearly sorted data?

A. Insertion Sort B. Merge Sort
C. Quick Sort D. Heap Sort

113. Which sorting algorithm is used for sorting arrays with large number of elements?

A. Insertion Sort B. Merge Sort
C. Quick Sort D. Heap Sort

114. Which of the following sorting algorithms is considered to be the fastest for small data sets?

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

115. Which of the following sorting algorithms is a divide-and-conquer algorithm?

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

116. Which of the following sorting algorithms has the best average case time complexity?

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

117. Which of the following sorting algorithms has a worst case time complexity of O(n^2)?

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

118. Which of the following sorting algorithms is an in-place sorting algorithm?

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

119. Which of the following sorting algorithms has the best space complexity?

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

120. Which of the following sorting algorithms is suitable for sorting linked lists?

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

121. Which of the following sorting algorithms is used for sorting data in a specialized manner like sorting data in a particular range or sorting data by a specific field?

A. Bubble sort
B. Insertion sort
C. Quick sort
D. Radix sort

122. What is the time complexity of linear search?

A. O(n) B. O(logn)
C. O(1) D. O(nlogn)

123. What is the time complexity of binary search?

A. O(n) B. O(logn)
C. O(1) D. O(nlogn)

124. What is the main advantage of using binary search over linear search?

A. More efficient in large data sets
B. Can be used only on sorted data sets
C. Can only be used on unsorted data sets
D. Not as efficient as linear search in small data sets

125. What is the basic idea behind linear search?

A. To search for an element by comparing it with every element in the data set
B. To search for an element by dividing the data set into two parts and comparing the target element with the middle element
C. To search for an element by comparing it with the first and last elements of the data set
D. To search for an element by comparing it with the last element of the data set

126. What is the basic idea behind binary search?

A. To search for an element by comparing it with every element in the data set
B. To search for an element by dividing the data set into two parts and comparing the target element with the middle element
C. To search for an element by comparing it with the first and last elements of the data set
D. To search for an element by comparing it with the last element of the data set

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

A. O(n) B. O(logn)
C. O(1) D. O(nlogn)

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

A. O(n)
B. O(logn)
C. O(1)
D. O(nlogn)

129. What type of data set is required to perform binary search?

A. Unsorted data set
B. Sorted data set
C. Randomly ordered data set
D. Any type of data set

130. What is the main disadvantage of using linear search over binary search?

A. More efficient in large data sets
B. Can be used only on sorted data sets
C. Not as efficient as binary search in large data sets
D. Can only be used on unsorted data sets

131. Which of the following is faster, linear search or binary search?

A. Linear search
B. Binary search
C. Both are equally fast
D. It depends on the size of the data set

132. What is the purpose of a hash function in a hash table?

A. To convert a key into an index for the table

B. To sort the elements in the table

C. To compare elements in the table

D. To store the elements in the table

133. What is a hash table in computer science?

A. A data structure that stores elements in a sorted manner

B. A data structure that stores elements in a random order

C. A data structure that stores elements using a key-value pair

D. A data structure that stores elements using an array

134. What is collision resolution in hash tables?

A. The process of adding elements to a hash table

B. The process of removing elements from a hash table

C. The proces s of resolving conflicts when two keys hash to the same index

D. The process of searching for a specific element in a hash table

135. What are some common collision resolution techniques in hash tables?

A. Linear probing

B. Binary search

C. Chaining

D. All of the above

136. What is linear probing in hash tables?

A. A technique that resolves collisions by searching through the table sequentially

B. A technique that resolves collisions by dividing the table into smaller sections

C. A technique that resolves collisions by using a linked list to store elements at the same index

D. A technique that resolves collisions by using binary search to find the correct index

137. What is chaining in hash tables?

A. A technique that resolves collisions by searching through the table sequentially

B. A technique that resolves collisions by dividing the table into smaller sections

C. A technique that resolves collisions by using a linked list to store elements at the same index

D. A technique that resolves collisions by using binary search to find the correct index

138. What is the time complexity of searching in a hash table using chaining?

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

C. O(n) D. O(n log n)

139. What is the main purpose of a hash function in a hash table?

A. To sort the elements in the table

B. To find the index of an element in the table

C. To compare the elements in the table

D. To store the elements in the table

140. What is the most common way to resolve collisions in a hash table?

A. Chaining

B. Open Addressing

C. Binary Search

D. Linear Search

141. What is the time complexity of searching an element in a hash table using the hash function?

A. O(n) B. O(logn)

C. O(1) D. O(nlogn)

142. What is the main advantage of using a hash table over a traditional array for storing elements?

A. Faster access time

B. Better sorting

C. More memory efficient

D. Better searching

143. What is the purpose of a collision resolution technique in a hash table?

A. To ensure that each element is stored at a unique index in the table

B. To sort the elements in the table

C. To compare the elements in the table

D. To store the elements in the table

144. What is the main objective of a minimum spanning tree algorithm?

A. To find the shortest path between two vertices

B. To find the minimum weight of a tree that connects all vertices

C. To find the longest path between two vertices

D. To find the maximum weight of a tree that connects all vertices

145. What is the time complexity of Prim's algorithm for finding a minimum spanning tree?

A. O(n^2) B. O(nlogn)

C. O(n) D. O(logn)

146. What is the time complexity of Kruskal's algorithm for finding a minimum spanning tree?

A. O(n^2) B. O(nlogn)

C. O(n) D. O(logn)

147. What is the difference between Prim's and Kruskal's algorithm for finding a minimum spanning tree?

A. Prim's algorithm starts with a single vertex and adds vertices to the tree while Kruskal's algorithm starts with all vertices and adds edges to the tree.

B. Prim's algorithm starts with all vertices and adds edges to the tree while Kruskal's algorithm starts with a single vertex and adds vertices to the tree.

C. Both algorithms start with a single vertex and add edges to the tree.

D. Both algorithms start with all vertices and add vertices to the tree.

148. What is the Round-Robin algorithm used for?

A. To find the minimum spanning tree

B. To schedule processes in a computer system

C. To sort a list of elements

D. To find the longest path between two vertices

149. What is the main advantage of using a minimum spanning tree algorithm?

A. To find the shortest path between two vertices

B. To find the minimum weight of a tree that connects all vertices

C. To find the longest path between two vertices

D. To find the maximum weight of a tree that connects all vertices

150. What is the use of a Round-Robin algorithm in a computer system?

A. To find the minimum spanning tree

B. To schedule processes in a computer system

C. To sort a list of elements

D. To find the longest path between two vertices

151. What is the main disadvantage of using a Kruskal's algorithm compared to Prim's algorithm for finding a minimum spanning tree?
A. Kruskal's algorithm is less efficient than Prim's algorithm
B. Kruskal's algorithm is more difficult to implement than Prim's algorithm
C. Kruskal's algorithm is less reliable than Prim's algorithm
D. Kruskal's algorithm is less accurate than Prim's algorithm

152. What is the use of a hash function in a hash table data structure?
A. To generate a unique key for each element in the table
B. To sort the elements in the table
C. To find the shortest path between two vertices
D. To find the longest path between two vertices

153. What is the main idea behind the greedy algorithm for solving the shortest path problem?
A. To find the shortest path by visiting all nodes in the graph
B. To find the shortest path by selecting the closest node at each step
C. To find the shortest path by selecting the most distant node at each step
D. To find the shortest path by using dynamic programming

154. What is the time complexity of Dijkstra's algorithm for finding the shortest path in a graph?
A. O(n^2)
B. O(nlogn)
C. O(n)
D. O(log n)

155. What is the difference between Dijkstra's algorithm and Bellman-Ford algorithm for finding the shortest path in a graph?
A. Dijkstra's algorithm is faster than Bellman-Ford algorithm
B. Bellman-Ford algorithm can handle negative weight edges while Dijkstra's algorithm cannot
C. Dijkstra's algorithm is simpler than Bellman-Ford algorithm
D. Bellman-Ford algorithm is more efficient than Dijkstra's algorithm

156. What is the use of the relaxation step in Dijkstra's algorithm for finding the shortest path?
A. To update the distance of each node from the source node
B. To find the shortest path between two nodes
C. To keep track of the visited nodes
D. To find the minimum distance between two nodes

157. Can Dijkstra's algorithm be used to find the shortest path in a graph with negative weight edges?
A. Yes, it can be used
B. No, it cannot be used

158. What is the main difference between greedy algorithm and dynamic programming for solving the shortest path problem?
A. Greedy algorithm gives the optimal solution while dynamic programming gives the suboptimal solution
B. Dynamic programming gives the optimal solution while greedy algorithm gives the suboptimal solution
C. Both greedy algorithm and dynamic programming give the optimal solution
D. Both greedy algorithm and dynamic programming give the suboptimal solution

159. What is the use of the priority queue in Dijkstra's algorithm for finding the shortest path?
A. To keep track of the visited nodes
B. To find the shortest path between two nodes
C. To update the distance of each node from the source node
D. To find the node with the minimum distance from the source node

160. Which of the following statements is true about undirected graphs?
A. The edges in an undirected graph have a direction.
B. The edges in an undirected graph have no direction.
C. The edges in an undirected graph are always weighted.
D. An undirected graph cannot have cycles.

161. Which of the following data structures is commonly used to represent a graph?
A. Linked list
B. Queue
C. Stack
D. Array

162. Which of the following is the best algorithm to find the transitive closure of a graph?
A. Dijkstra's algorithm
B. Bellman-Ford algorithm
C. Warshall's algorithm
D. Prim's algorithm

163. Which of the following algorithms is used to find the shortest path in a weighted graph?
A. Depth-first search
B. Breadth-first search
C. Dijkstra's algorithm
D. Kruskal's algorithm

164. What is the time complexity of Warshall's algorithm?
A. O(V+E)
B. O(V^2)
C. O(E^2)
D. O(E log V)

165. Which traversal algorithm is used to traverse a graph starting from a particular vertex and exploring as far as possible along each branch before backtracking?
A. Depth-first traversal
B. Breadth-first traversal
C. Topological sorting
D. Kruskal's algorithm

166. Which traversal algorithm is used to traverse a graph by exploring all the vertices at the same level before moving on to the next level?
A. Depth-first traversal
B. Breadth-first traversal
C. Topological sorting
D. Kruskal's algorithm

167. What is the time complexity of Breadth-first traversal of a graph?
A. O(V+E)
B. O(V^2)
C. O(E^2)
D. O(E log V)

168. Which of the following algorithms can be used to find a topological ordering of a directed acyclic graph (DAG)?
A. Depth-first traversal
B. Breadth-first traversal
C. Topological sorting (DFS)
D. Topological sorting (BFS)

169. Which of the following is true about a DAG?
A. It can have cycles.
B. It can have multiple sources.
C. It can have multiple sinks.
D. It can be disconnected.

170. Which of the following is used to determine whether a graph is acyclic or not?
A. Depth-first traversal
B. Breadth-first traversal
C. Topological sorting (DFS)
D. Topological sorting (BFS)

171. What is the time complexity of Depth-first traversal of a graph?
A. O(V+E) B. O(V^2)
C. O(E^2) D. O(E log V)

172. Which of the following algorithms is used to find the minimum spanning tree of a graph?
A. Depth-first traversal
B. Breadth-first traversal
C. Dijkstra's algorithm
D. Kruskal's algorithm

173. Which of the following is not a sorting algorithm?
A. Bubble sort B. Heap sort
C. Quick sort D. Binary search

174. Which of the following data structures is used to implement heap sort?
A. Stak B. Queue
C. Heap D. Linked list

175. What is the worst-case time complexity of heap sort?
A. O(n) B. O(n log n)
C. O(n^2) D. O(log n)

176. Which of the following is not a step in heap sort?
A. Building a heap
B. Sorting the heap
C. Swapping the first and last elements
D. Inserting elements into the heap

177. What is the space complexity of heap sort?
A. O(1) B. O(log n)
C. O(n) D. O(n log n)

178. What is the minimum number of elements that can be sorted using heap sort?
A. 0 B. 1
C. 2 D. 3

179. Which of the following is not a property of a heap?
A. It is a complete binary tree.
B. The root is always the maximum or minimum element.
C. The left child is always greater than the right child.
D. The height of the tree is log(n).

180. What is the time complexity of building a heap?
A. O(n) B. O(n log n)
C. O(log n) D. O(n^2)

181. Which of the following is not a type of heap?
A. Binary heap B. Fibonacci heap
C. Ternary heap D. Quad heap

182. Which of the following is a disadvantage of heap sort?
A. It is not an in-place algorithm.
B. It cannot handle duplicate elements.
C. It has a worst-case time complexity of O(n^2).
D. It is not stable.

183. Who invented Shell sort?
A. Donald Knuth
B. Edsger Dijkstra
C. Stephen Cole Kleene
D. Donald Shell

184. Which of the following sorting algorithm is based on the insertion sort algorithm?
A. Shell sort B. Heap sort
C. Merge sort D. Quick sort

185. In Shell sort, what is the initial gap size between elements that are compared?
A. 1 B. 2
C. n/2 D. log(n)

186. What is the time complexity of the best-case scenario for Shell sort?
A. O(n) B. O(n log n)
C. O(n^2) D. O(log n)

187. Which of the following is the primary advantage of Shell sort over other sorting algorithms?
A. It has a worst-case time complexity of O(n^2).
B. It is an in-place algorithm.
C. It is easy to implement.
D. It can sort large datasets more efficiently than other sorting algorithms.

188. What is the time complexity of the worst-case scenario for Shell sort?
A. O(n)
B. O(n log n)
C. O(n^2)
D. O(log n)

189. Which of the following is the final step in Shell sort?
A. Shifting elements to the left or right based on their value.
B. Swapping elements that are out of order.
C. Repeating the process with a smaller gap size.
D. Partitioning the array into subarrays.

190. Which of the following is a disadvantage of Shell sort?
A. It is not an in-place algorithm.
B. It is not stable.
C. It can only sort arrays of a certain size.
D. It is difficult to understand.

191. Which of the following is the time complexity of the average case for Shell sort?
A. O(n) B. O(n log n)
C. O(n^2) D. O(log n)

192. Which of the following is the time complexity of the space required for Shell sort?
A. O(1) B. O(n)
C. O(n log n) D. O(n^2)

193. What is the main difference between internal and external sorting?
A. Internal sorting is used for small datasets and external sorting is used for large datasets.
B. Internal sorting is performed entirely in memory and external sorting involves writing data to disk.
C. Internal sorting is a stable sorting algorithm and external sorting is an unstable sorting algorithm.
D. Internal sorting is a sorting algorithm that uses comparison-based operations and external sorting uses non-comparison operations.

194. Which of the following is an example of an internal sorting algorithm?
A. Merge sort
B. Quick sort
C. Radix sort
D. External sort

195. Which of the following is an example of an external sorting algorithm?

A. Bubble sort B. Selection sort

C. Insertion sort D. Merge sort

196. What is the time complexity of an external sort?

A. O(n) B. O(n log n)

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

197. Which of the following is true about external sorting?

A. It is used to sort small datasets.

B. It involves reading and writing data to and from external storage.

C. It is always faster than internal sorting.

D. It is a comparison-based sorting algorithm.

198. Which of the following is a disadvantage of external sorting?

A. It requires a large amount of memory.

B. It is not scalable.

C. It is difficult to implement.

D. It is only suitable for certain types of data.

199. Which of the following is a disadvantage of internal sorting?

A. It requires a large amount of memory.

B. It is not suitable for large datasets.

C. It is slow for small datasets.

D. It is not a stable sorting algorithm.

200. What is the time complexity of an internal sort?

A. O(n)

B. O(n log n)

C. O(n^2)

D. O(log n)

201. Which of the following is an advantage of external sorting?

A. It is faster than internal sorting.

B. It can handle larger datasets.

C. It requires less memory than internal sorting.

D. It is easier to implement than internal sorting.

202. Which of the following is a disadvantage of external sorting?

A. It is not stable.

B. It requires more disk I O than internal sorting.

C. It is not a comparison-based sorting algorithm.

D. It is only suitable for datasets with a small number of distinct values.

203. Which of the following sorting algorithms is based on the idea of selecting the smallest element in the unsorted part of an array and swapping it with the first element in the unsorted part of the array?

A. Bubble sort B. Insertion sort

C. Selection sort D. Quick sort

204. Which of the following sorting algorithms is based on the idea of partitioning an array into two subarrays, with elements in the left subarray being smaller than the pivot and elements in the right subarray being larger than the pivot?

A. Bubble sort B. Insertion sort

C. Selection sort D. Quick sort

205. In Insertion sort, which of the following is true about the sorted part of the array after the first iteration?

A. The first element is always the smallest element in the array.

B. The first element is always the largest element in the array.

C. The first element is always in its correct sorted position.

D. The first element is always in its correct sorted position relative to the second element.

206. What is the time complexity of Insertion sort in the worst-case scenario?

A. O(n) B. O(n log n)

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

207. In Selection sort, how many comparisons are made in the first iteration of the outer loop?

A. n B. n-1

C. n-2 D. log(n)

208. What is the time complexity of Selection sort in the best-case scenario?

A. O(n) B. O(n log n)

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

209. Which of the following is an advantage of Insertion sort over Selection sort?

A. It is faster for small datasets.

B. It requires fewer comparisons.

C. It has a better worst-case time complexity.

D. It is easier to implement.

210. Which of the following is an advantage of Selection sort over Insertion sort?

A. It requires less memory.

B. It has a better worst-case time complexity.

C. It is stable.

D. It is more efficient for large datasets.

211. What is the time complexity of Insertion sort in the best-case scenario?

A. O(n)

B. O(n log n)

C. O(n^2)

D. O(log n)

212. Which of the following is a disadvantage of Selection sort?

A. It is not stable.

B. It requires more memory than Insertion sort.

C. It has a worse worst-case time complexity than Insertion sort.

D. It is difficult to implement.

213. What is Warshall's algorithm used for?

A. To find the shortest path between two vertices in a graph

B. To find the transitive closure of a graph

C. To find the minimum spanning tree of a graph

D. To find the maximum flow in a network

214. Warshall's algorithm is based on which concept?

A. Depth-first search

B. Breadth-first search

C. Matrix multiplication

D. Greedy algorithm

215. What is the time complexity of Warshall's algorithm?

A. O(n^2)

B. O(nlogn)

C. O(n^3)

D. O(2^n)

216. What is a priority queue?

A. A queue with elements ordered based on their priority

B. A queue with elements ordered based on their frequency of use

C. A queue with elements ordered based on their age

D. A queue with elements ordered randomly

217. What is the main difference between a min heap and a max heap?

A. Min heap has elements ordered in ascending order, while max heap has elements ordered in descending order

B. Min heap has elements ordered in descending order, while max heap has elements ordered in ascending order

C. Min heap is a complete binary tree, while max heap is not

D. Min heap is not a complete binary tree, while max heap is

218. What is the time complexity of inserting an element into a heap?

A. O(n) B. O(logn)
C. O(1) D. O(nlogn)

9. What is the time complexity of deleting the minimum element from a min heap?

A. O(n) B. O(logn)
C. O(1) D. O(nlogn)

. What is the time complexity of heap sort?

A. O(n) B. O(logn)
C. O(nlogn) D. O(n^2)

What is topological sorting?

A. A method for sorting nodes in a directed acyclic graph (DAG) in a linear order such that for every directed edge (u, v), vertex u comes before vertex v

3. A method for finding the shortest path in a directed graph

. A method for finding the longest path in a directed graph

A method for finding a cycle in a directed graph

222. What is the main difference between depth-first topological sorting and breadth-first topological sorting?

A. Depth-first topological sorting sorts nodes based on their depth, while breadth-first topological sorting sorts nodes based on their breadth

B. Depth-first topological sorting sorts nodes based on their breadth, while breadth-first topological sorting sorts nodes based on their depth

C. Depth-first topological sorting uses a stack, while breadth-first topological sorting uses a queue

D. Depth-first topological sorting uses a queue, while breadth-first topological sorting uses a stack

223. What is the time complexity of depth-first topological sorting?

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

224. What is the time complexity of breadth-first topological sorting?

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

225. What is the main operation performed on a linear queue?

A. Insertion B. Deletion
C. Both A. and B. D. Sorting

226. What is the main advantage of using a circular queue over a linear queue?

A. Avoids overflow issues
B. Easy implementation
C. Faster than linear queue
D. Better memory utilization

227. What is the time complexity of inserting an element into a circular queue?

A. O(1) B. O(logn)
C. O(n) D. O(nlogn)

228. What is the time complexity of deleting an element from a circular queue?

A. O(1) B. O(logn)
C. O(n) D. O(nlogn)

229. What is the main difference between linear queue and circular queue in terms of handling overflow?

A. Linear queue handles overflow by increasing its size dynamically, while circular queue handles overflow by making the front and rear pointers point to the first position

B. Linear queue handles overflow by making the front and rear pointers point to the first position, while circular queue handles overflow by increasing its size dynamically

C. Linear queue handles overflow by using a linked list, while circular queue handles overflow by making the front and rear pointers point to the first position

D. Linear queue handles overflow by making the front and rear pointers point to the first position, while circular queue handles overflow by using a linked list

230. What is the basic syntax for checking if a stack is full in C?

A. if(top == MAX_SIZE-1)
B. if(top >= MAX_SIZE)
C. if(top == MAX_SIZE)
D. if(top > MAX_SIZE-1)

231. What is the basic syntax for checking if a stack is empty in C?

A. if(top == -1) B. if(top <= -1)
C. if(top == 0) D. if(top < 0)

232. What is the basic syntax for handling overflow in a stack in C?

A. printf("Stack overflow\n");
B. return -1;
C. exit(1);
D. Throw an exception

233. What is the basic syntax for handling underflow in a stack in C?

A. printf("Stack underflow\n");
B. return -1;
C. exit(1);
D. Throw an exception

234. What is the basic syntax for checking if a queue is full in C?

A. if(rear == MAX_SIZE-1)
B. if(rear >= MAX_SIZE)
C. if(rear == MAX_SIZE)
D. if(rear > MAX_SIZE-1)

235. What is the basic syntax for checking if a queue is empty in C?

A. if(front == rear)
B. if(front > rear)
C. if(front >= rear)
D. if(front == -1)

236. What is the basic syntax for handling overflow in a queue in C?

A. printf("Queue overflow\n");
B. return -1;
C. exit(1);
D. Throw an exception

237. What is the basic syntax for handling underflow in a queue in C?

A. printf("Queue underflow\n");
B. return -1;
C. exit(1);
D. Throw an exception

238. What is the basic syntax for checking if a circular queue is full in C?

A. if((rear+1)%MAX_SIZE == front)

B. if(rear == front)

C. if(rear == MAX_SIZE-1)

D. if(rear >= MAX_SIZE)

239. What is the basic syntax for checking if a circular queue is empty in C?

A. if(front == rear)

B. if(front > rear)

C. if(front >= rear)

D. if(front == -1)

240. What is the basic syntax for handling overflow in a circular queue in C?

A. printf("Circular queue overflow\n");

B. return -1;

C. exit(1);

D. Throw an exception

241. What is the basic syntax for handling underflow in a circular queue in C?

A. printf("Circular queue underflow\n");

B. return -1;

C. exit(1);

D. Throw an exception

242. What is the balancing factor of a node in an AVL tree?

A. The difference in height between the left and right subtrees of the node

B. The sum of the heights of the left and right subtrees of the node

C. The height of the node's parent

D. The number of children the node has

243. What is the range of the balancing factor in an AVL tree?

A. -1 to 1

B. -2 to 2

C. -3 to 3

D. 0 to 2

244. What does a balancing factor of -1 indicate in an AVL tree?

A. The left subtree is taller than the right subtree

B. The right subtree is taller than the left subtree

C. The left and right subtrees are equal in height

D. The node is a leaf node

245. What does a balancing factor of 0 indicate in an AVL tree?

A. The left and right subtrees are equal in height

B. The left subtree is taller than the right subtree

C. The right subtree is taller than the left subtree

D. The node is a leaf node

246. What does a balancing factor of 2 indicate in an AVL tree?

A. The left subtree is taller than the right subtree by 2 levels

B. The right subtree is taller than the left subtree by 2 levels

C. The left subtree is taller than the right subtree by 1 level

D. The right subtree is taller than the left subtree by 1 level

247. What is the maximum possible height difference between the left and right subtrees of a node in an AVL tree?

A. 1 level

B. 2 levels

C. 3 levels

D. 4 levels

248. How is the balancing factor used to maintain balance in an AVL tree?

A. Whenever a node is inserted or deleted, the balancing factors of all nodes in the affected subtree are updated, and if any node has a balancing factor outside the range of -1 to 1, rotations are performed to rebalance the tree.

B. Whenever a node is inserted or deleted, the balancing factors of all nodes in the tree are updated, and if any node has a balancing factor outside the range of -2 to 2, rotations are performed to rebalance the tree.

C. Whenever a node is inserted or deleted, the balancing factors of all nodes in the affected subtree are updated, and if any node has a balancing factor outside the range of 0 to 2, rotations are performed to rebalance the tree.

D. The balancing factor is not used to maintain balance in an AVL tree.

249. What is the correct postfix notation for the infix expression "3 + 4 * 2 / (1 - 5) ^ 2 ^ 3"?

A. 3 4 2 * 1 5 - 2 3 ^ ^ / +

B. 3 4 2 * 1 5 - 3 2 ^ ^ / +

C. 3 4 2 * 1 5 - 2 ^ 3 ^ / +

D. 3 4 2 * 1 5 - 2 3 ^ ^ / +

250. What is the correct infix notation for the prefix expression "* + 3 / 2 1 - 5 ^ 2 3"?

A. 3 + 2 / 1 * (5 ^ 2 - 3)

B. + 3 * / 2 1 - ^ 5 2 3

C. + 3 / 2 1 * (5 ^ 2 - 3)

D. * + 3 / 2 1 - ^ 5 2 3

251. What is the correct prefix notation for the postfix expression "5 2 ^ 3 * 4 5 / + 6 -"?

A. - + ^ 5 2 * 3 / 4 5 6

B. - + * ^ 5 2 3 / 4 5 6

C. - + * 5 ^ 2 3 / 4 5 6

D. - + ^ 5 2 3 * / 4 5 6

252. Which notation is the most commonly used by humans to express arithmetic expressions?

A. Postfix notation

B. Infix notation

C. Prefix notation

D. All notations are equally popular

253. Which notation is the easiest to parse and evaluate using a stack-based algorithm?

A. Postfix notation

B. Infix notation

C. Prefix notation

D. All notations can be parsed and evaluated equally easily

254. Which notation is the easiest to convert to machine code or assembly language?

A. Postfix notation

B. Infix notation

C. Prefix notation

D. All notations can be easily converted to machine code or assembly language

255. What is the advantage of using prefix or postfix notation over infix notation?

A. Prefix and postfix notation can eliminate the need for parentheses

B. Prefix and postfix notation are more human-readable

C. Prefix and postfix notation are more compact than infix notation

D. Prefix and postfix notation are more intuitive than infix notation

256. **Which of the following is NOT an application of stacks?**

A. Implementing undo-redo functionality in text editors

B. Evaluating arithmetic expressions

C. Implementing backtracking algorithms

D. Storing data in a hash table

257. **What is the application of a stack in implementing a web browser's back button?**

A. The URLs of the visited web pages are pushed onto a stack

B. The URLs of the visited web pages are popped off a stack

C. The URLs of the visited web pages are stored in a queue

D. The URLs of the visited web pages are stored in a hash table

258. **Which data structure can be used to implement a stack?**

A. Array

B. Linked list

C. Both array and linked list

D. Neither array nor linked list

259. **What is the application of a stack in evaluating arithmetic expressions?**

A. The operands are pushed onto the stack, and the operators are evaluated in order

B. The operators are pushed onto the stack, and the operands are evaluated in order

C. The operands and operators are both pushed onto the stack in random order

D. The operands and operators are stored in separate stacks

260. **What is the application of a stack in implementing function calls in programming languages?**

A. The return address and parameters of a function call are pushed onto the stack, and the function is executed

B. The function itself is pushed onto the stack, and the return address and parameters are evaluated in order

C. The parameters of a function call are pushed onto the stack, and the function is executed without using a return address

D. The return address and parameters of a function call are stored in a queue

261. **What is the application of a stack in implementing undo-redo functionality in text editors?**

A. The previous states of the document are pushed onto a stack, and the user can undo or redo by popping from the stack

B. The current state of the document is pushed onto a stack, and the user can undo or redo by popping from the stack and reapplying the changes

C. The current and previous states of the document are stored in separate stacks, and the user can switch between them

D. The undo-redo functionality is not implemented using a stack, but using a hash table

262. **What is the application of a stack in implementing backtracking algorithms?**

A. The possible moves are pushed onto a stack, and the algorithm backtracks by popping from the stack

B. The previous states of the algorithm are pushed onto a stack, and the algorithm backtracks by popping from the stack

C. The possible moves and previous states of the algorithm are stored in separate stacks, and the algorithm switches between them

D. Backtracking algorithms do not use a stack

263. **What is the time complexity of quicksort on this array?**

A. O(1) B. O(n)

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

264. **What is a singly linked list?**

A. A linked list in which each node has two pointers, one to the previous node and one to the next node

B. A linked list in which each node has only one pointer, to the next node

C. A linked list in which each node has only one pointer, to the previous node

D. A linked list in which each node has two pointers, one to the previous node and one to the next node, and the nodes are arranged in a circular fashion

265. **What is the syntax for creating a new node in a singly linked list?**

A. new_node = Node(data, prev, next)

B. new_node = Node(data, next)

C. new_node = Node(data, prev)

D. new_node = Node(datA.

266. **What is a doubly linked list?**

A. A linked list in which each node has two pointers, one to the previous node and one to the next node

B. A linked list in which each node has only one pointer, to the next node

C. A linked list in which each node has only one pointer, to the previous node

D. A linked list in which each node has two pointers, one to the previous node and one to the next node, and the nodes are arranged in a circular fashion

267. **What is the syntax for creating a new node in a doubly linked list?**

A. new_node = Node(data, prev, next)

B. new_node = Node(data, next)

C. new_node = Node(data, prev)

D. new_node = Node(datA.

268. **What is a circular linked list?**

A. A linked list in which each node has two pointers, one to the previous node and one to the next node

B. A linked list in which each node has only one pointer, to the next node

C. A linked list in which each node has only one pointer, to the previous node

D. A linked list in which each node has two pointers, one to the previous node and one to the next node, and the nodes are arranged in a circular fashion

269. **What is the syntax for creating a new node in a circular linked list?**

A. new_node = Node(data, prev, next)

B. new_node = Node(data, next)

C. new_node = Node(data, prev)

D. new_node = Node(datA.

270. **What is the advantage of using a doubly linked list over a singly linked list?**

A. Doubly linked lists use less memory

B. Doubly linked lists are easier to implement

C. Doubly linked lists can be sorted more efficiently

D. Doubly linked lists allow for traversal in both directions

271. What is the advantage of using a circular linked list?

A. Circular linked lists use less memory

B. Circular linked lists are easier to implement

C. Circular linked lists can be sorted more efficiently

D. Circular linked lists can be used to model certain real-world problems, such as scheduling algorithms

272. What is the disadvantage of using a circular linked list?

A. Circular linked lists are slower than other types of linked lists

B. Circular linked lists cannot be used to model certain real-world problems, such as scheduling algorithms

C. Circular linked lists are more difficult to implement than other types of linked lists

D. Circular linked lists can lead to infinite loops if not implemented carefully

273. How do you traverse a circular linked list?

A. Using a for loop

B. Using a while loop

C. Using a do-while loop

D. Using recursion

274. How do you insert a node at the beginning of a circular linked list?

A. Set the next pointer of the new node to the head node, and set the next pointer of the last node to the new node

B. Set the next pointer of the new node to the head node, and set the next pointer of the head node to the new node

C. Set the next pointer of the new node to the last node, and set the next pointer of the head node to the new node

D. Set the next pointer of the new node to the head node, and set the next pointer of the new node to the last node

275. How do you delete a node from a circular linked list?

A. Set the next pointer of the previous node to the next node, and set the next pointer of the current node to null

B. Set the next pointer of the previous node to the next node, and set the next pointer of the current node to the previous node

C. Set the previous pointer of the previous node to the next node, and set the next pointer of the current node to null

D. Set the previous pointer of the previous node to the next node, and set the previous pointer of the current node to the previous node

276. What is overflow in a stack?

A. When the stack is full and a push operation is attempted

B. When the stack is empty and a pop operation is attempted

C. When the stack is partially full and a push operation is attempted

D. When the stack is partially empty and a pop operation is attempted

277. What is the syntax for detecting overflow in a stack?

A. if (top == MAXSIZE)

B. if (top > MAXSIZE)

C. if (top == 0)

D. if (top < 0)

278. What is underflow in a stack?

A. When the stack is empty and a pop operation is attempted

B. When the stack is full and a push operation is attempted

C. When the stack is partially full and a push operation is attempted

D. When the stack is partially empty and a pop operation is attempted

279. What is the syntax for detecting underflow in a stack?

A. if (top == 0)

B. if (top > MAXSIZE)

C. if (top == MAXSIZE)

D. if (top < 0)

280. What is an empty stack?

A. A stack with no elements

B. A stack with one element

C. A stack with some elements but no top element

D. A stack with some elements but no bottom element

281. What is the syntax for checking if a stack is empty?

A. if (top == -1)

B. if (top > MAXSIZE)

C. if (top == MAXSIZE)

D. if (top < 0)

282. What is a full stack?

A. A stack with no elements

B. A stack with one element

C. A stack with some elements but no top element

D. A stack with no more room for new elements

283. What is the syntax for checking if a stack is full?

A. if (top == 0)

B. if (top > MAXSIZE)

C. if (top == MAXSIZE-1)

D. if (top < 0)

284. Which of the following is an example of a nonlinear data structure that allows access to its elements in a random manner?

A. Tree B. Graph

C. Stack D. Queue

285. Which of the following is an example of a nonlinear data structure that is used for network modeling and analysis?

A. Tree B. Graph

C. Stack D. Queue

286. What is a directed graph?

A. A graph in which each edge has a direction from one node to another

B. A graph in which each edge has no direction

C. A graph in which each node has a weight

D. A graph in which each edge has a weight

287. What is an undirected graph?

A. A graph in which each edge has no direction

B. A graph in which each edge has a direction from one node to another

C. A graph in which each node has a weight

D. A graph in which each edge has a weight

288. What is a weighted graph?

A. A graph in which each node has a weight

B. A graph in which each edge has a weight

C. A graph in which each node and edge has a weight

D. A graph in which each node and edge has no weight

289. What is an unweighted graph?
A. A graph in which each node has a weight
B. A graph in which each edge has a weight
C. A graph in which each node and edge has a weight
D. A graph in which each node and edge has no weight

290. What is a complete graph?
A. A graph in which each node is connected to every other node
B. A graph in which each node is connected to a subset of the other nodes
C. A graph in which each node has a weight
D. A graph in which each edge has a weight

291. What is a sparse graph?
A. A graph in which there are relatively few edges compared to the maximum number of edges
B. A graph in which there are relatively many edges compared to the maximum number of edges
C. A graph in which each node has a weight
D. A graph in which each edge has a weight

292. What is a dense graph?
A. A graph in which there are relatively many edges compared to the maximum number of edges
B. A graph in which there are relatively few edges compared to the maximum number of edges
C. A graph in which each node has a weight
D. A graph in which each edge has a weight

293. What is a bipartite graph?
A. A graph in which the nodes can be partitioned into two disjoint sets such that each edge connects a node in one set to a node in the other set
B. A graph in which each node has a weight
C. A graph in which each edge has a weight
D. A graph in which each edge has a direction from one node to another

294. What is the main difference between graph traversal and tree traversal?
A. Graphs may have loops, while trees do not have loops
B. Graphs and trees have the same traversal methods
C. Trees may have loops, while graphs do not have loops
D. Trees and graphs have different traversal methods, but both have loops

295. Which of the following traversal methods is used only for trees?
A. Depth-First Search (DFS)
B. Breadth-First Search (BFS)
C. Inorder traversal
D. None of the above

296. What is the main purpose of graph traversal?
A. To find the shortest path between two nodes in a graph
B. To find the longest path between two nodes in a graph
C. To visit all the nodes in a graph exactly once
D. To remove loops from a graph

297. How many edges are there in a complete graph with n vertices?
A. n-1 B. n
C. n(n-1)/2 D. n(n+1)/2

298. What is the maximum number of edges in a simple graph with 8 vertices?
A. 7 B. 14
C. 2 D. 28

299. Which of the following algorithms solves the all-pair shortest path problem in a graph with negative edge weights?
A. Bellman-Ford algorithm
B. Dijkstra's algorithm
C. Floyd-Warshall algorithm
D. Prim's algorithm

300. What is the time complexity of the Floyd-Warshall algorithm for the all-pair shortest path problem?
A. O(V log V) B. O(V^2)
C. O(V^3) D. O(E log V)

301. What is the time complexity of the Bellman-Ford algorithm for the all-pair shortest path problem?
A. O(V log V) B. O(V^2)
C. O(V^3) D. O(E log V)

302. What is the time complexity of Dijkstra's algorithm for the all-pair shortest path problem?
A. O(V log V) B. O(V^2)
C. O(V^3) D. O(E log V)

303. Which of the following data structures is useful in traversing a given graph by breadth first search?
A. Stack B. Queue
C. Binary tree D. Linked list

304. What is the time complexity of breadth first search traversal of a graph with V vertices and E edges?
A. O(V) B. O(E)
C. O(V + E) D. O(V * E)

305. Which of the following algorithms can be used to find the shortest path between two nodes in a graph?
A. Breadth First Search (BFS)
B. Depth First Search (DFS)
C. Dijkstra's algorithm
D. All of the above

306. Which of the following algorithms is most suitable for finding the shortest path between two nodes in a graph?
A. Breadth First Search (BFS)
B. Depth First Search (DFS)
C. Dijkstra's algorithm
D. All of the above

307. Which data structure is commonly used for implementing recursive algorithms?
A. Queue B. Stack
C. Binary tree D. Linked list

308. Which of the following is an example of an infix expression?
A. A+BC
B. +A*BC
C. ABC+*
D. None of the above

309. If the elements '5', '10', '15' and '20' are added in a stack, in what order will they be removed?
A. 20, 15, 10, 5
B. 5, 10, 15, 20
C. 15, 20, 5, 10
D. None of the above

310. What is the correct way to increment the rear end in a circular queue?
A. rear = rear + 1
B. (rear + 1) % max
C. (rear % max) + 1
D. None of the above

311. Which of the following options correctly creates a new node in dynamic memory allocation for a linked list?

A. ptr = (node*)malloc(sizeof(node*))

B. ptr = (node)malloc(sizeof(node))

C. ptr = (node*)malloc(sizeof(node))

D. None of the above

ANSWER SHEET

1.C	2.B	3.A	4.A	5.B	6.C	7.C	8.B	9.A	10.B
11.B	12.B	13.A	14.C	15.A	16.A	17.D	18.D	19.D	20.A
21.A	22.B	23.C	24.A	25.C	26.A	27.D	28.A	29.C	30.A
31.A	32.A	33.A	34.A	35.A	36.B	37.B	38.B	39.B	40.A
41.B	42.C	43.C	44.B	45.D	46.B	47.C	48.B	49.A	50.C
51.C	52.D	53.D	54.C	55.C	56.B	57.A	58.C	59.D	60.D
61.A	62.D	63.B	64.A	65.D	66.C	67.A	68.A	69.C	70.B
71.C	72.B	73.C	74.A	75.B	76.A	77.C	78.C	79.B	80.C
81.C	82.C	83.B	84.B	85.B	86.B	87.B	88.D	89.D	90.D
91.D	92.A	93.C	94.D	95.A	96.A	97.C	98.A	99.C	100.C
101.A	102.C	103.B	104.C	105.B	106.B	107.B	108.C	109.A	110.B
111.C	112.A	113.D	114.B	115.D	116.C	117.A	118.C	119.B	120.B
121.D	122.A	123.B	124.A	125.A	126.B	127.C	128.B	129.B	130.C
131.B	132.A	133.C	134.C	135.D	136.A	137.C	138.C	139.B	140.A
141.C	142.A	143.A	144.B	145.A	146.B	147.A	148.B	149.B	150.B
151.A	152.A	153.B	154.B	155.B	156.A	157.B	158.A	159.D	160.B
161.A	162.C	163.C	164.B	165.A	166.B	167.A	168.C	169.B,C	170.C

171.A	172.D	173.D	174.C	175.B	176.D	177.A	178.C	179.C	180.A
181.D	182.A	183.D	184.A	185.C	186.A	187.D	188.C	189.B	190.B
191.B	192.A	193.B	194.A	195.D	196.B	197.B	198.A	199.B	200.B
201.B	202.B	203.C	204.D	205.C	206.C	207.B	208.C	209.A	210.D
211.A	212.C	213.B	214.C	215.C	216.A	217.A	218.B	219.B	220.C
221.A	222.C	223.C	224.C	225.C	226.A	227.A	228.A	229.A	230.A
231.A	232.A	233.A	234.A	235.A	236.A	237.A	238.A	239.A	240.A
241.A	242.A	243.A	244.B	245.A	246.B	247.B	248.A	249.A	250.A
251.A	252.B	253.A	254.C	255.A	256.D	257.A	258.C	259.A	260.A
261.A	262.B	263.C	264.B	265.B	266.A	267.A	268.D	269.D	270.D
271.D	272.D	273.B	274.B	275.B	276.A	277.A	278.A	279.A	280.A
281.A	282.D	283.C	284.A	285.B	286.A	287.A	288.B	289.D	290.A
291.A	292.A	293.A	294.A	295.C	296.C	297.C	298.D	299.C	300.C
301.B	302.A	303.B	304.C	305.C	306.C	307.B	308.A	309.A	310.C
311.C									