Multiple Choice Q & A

Introduction to Data Structures:3

1. What is a data structure?

   A. A way to store data for efficient search and retrieval.

   B. A programming language.

   C. A type of software.

   D. A computer hardware component.

   Answer: a. A way to store data for efficient search and retrieval.

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

   A. Graph

   B. Tree

   C. Array

   D. Hash Table

   Answer: Array

3. What is the time complexity of accessing an element in an array?

   A. O(1)

   B. O(n)

C. O(log n)

D. O(n^2)

Answer: O(1)

4. Which data structure uses LIFO (Last In, First Out) principle?

A. Queue

B. Stack

C. LinkedList

D. BinaryTree

Answer: Stack

5. What advantage does a linked list have over an array?

A. Fixed size

B. Better locality of reference

C. Ease of resizing

D. Faster access time

Answer: Ease of resizing

Abstract data types:

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

A. A specific implementation of a data structure in a programming language.

B. A mathematical model for data types, specifying the behavior but not the implementation.

C. A low-level system programming language.

D. A hardware device used for data storage.

Answer: B) A mathematical model for data types, specifying the behavior but not the implementation.

7. Which of the following is an example of an ADT?

A. Integer

B. Stack

C. Python list

D. C++ Vector

Answer: Stack

8. What does the ADT Stack primarily support?

A. Retrieving the least recently added element.

B. Random access to any element.

C. Adding and removing elements from the end only.

D. Sorting elements in ascending order.

Answer: C) Adding and removing elements from the end only.

9. Which operation is not typically supported by the Queue ADT?

A. Enqueue

B. Dequeue

C. Peek

D. Random access

Answer: D) Random access

10. What characteristic differentiates an ADT from a Data Structure?

    A. An ADT defines the logical form of the data type without specifying how it is implemented.

    B. An ADT can only be implemented in high-level programming languages.

    C. Data structures do not allow operations on data.

    D. Data structures are theoretical concepts with no practical application.

    Answer: A) An ADT defines the logical form of the data type without specifying how it is implemented.

Linear list – singly linked list implementation:

11. In a singly linked list, what is the role of a "node"?

    A. It represents the entire list.

    B. It stores the data element and a reference to the previous node.

    C. It stores the data element and a reference to the next node.

    D. It determines the size of the list.

    Answer: C) It stores the data element and a reference to the next node.

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

    A. O(1)

    B. O(n)

    C. O(log n)

    D. O(n^2)

    Answer: O(1)

13. Which operation is used to add a new element at the end of a singly linked list?

    A. Append

    B. Prepend

    C. Insert

    D. Delete

    Answer: Append

14. What happens if you attempt to traverse a singly linked list that contains a "null" or "nullptr" reference?

    A. It results in a runtime error.

    B. It continues to the next node.

    C. It stops the traversal and terminates the program.

    D. It goes into an infinite loop.

    Answer: A) It results in a runtime error.

15. What is the advantage of a singly linked list over an array when it comes to dynamic memory allocation?

    A. Singly linked lists use less memory.

    B. Singly linked lists have faster random access.

    C. Singly linked lists have a fixed size.

    D. Singly linked lists cannot be resized.

    Answer: A) Singly linked lists use less memory.

Insertion:

16. What is the time complexity of inserting an element at the end of an array with

    n elements?

    A. O(1)

    B. O(n)

    C. O(log n)

    D. O(n^2)

    Answer: A) O(1)

17. In a linked list, which operation allows you to insert a new element at the beginning of the list?

    A. Append

    B. Prepend

C. Delete

D. Search

Answer: B) Prepend.

18. When inserting an element into a binary search tree (BST), where would you place it if it is greater than the current node but smaller than its right child?

    A. As the right child of the current node.

    B. As the left child of the current node.

    C. As a sibling of the current node.

    D. At the root of the BST.

    Answer: A) As the right child of the current node.

19. In a stack data structure, what happens when you attempt to insert an element when the stack is full?

    A. The element is inserted at the top of the stack.

    B. The element is inserted at the bottom of the stack.

    C. An error occurs, and the stack remains unchanged.

    D. The last element in the stack is automatically removed.

    Answer: C) An error occurs, and the stack remains unchanged.

20. In a balanced binary search tree (AVL tree), what rotation(s) should be performed when inserting a new node that causes the tree to become unbalanced with a height difference of 2 between the left and right subtrees?

    A. Left-Left Rotation

B. Right-Right Rotation

C. Left-Right Rotation

D. Right-Left Rotation

Answer: C) Left-Right Rotation

Deletion and searching operations on linear list:

21. What is the time complexity of searching for an element in an unsorted linear list with n elements using linear search?

A. O(1)

B. O(log n)

C. O(n)

D. O(n^2)

Answer: C) O(n)

22. In a singly linked list, what operation allows you to remove an element from the end of the list?

A. Append

B. Prepend

C. Delete

D. Search

Answer: C) Delete

23. When searching for an element in a binary search tree (BST), what is the typical time complexity for finding the element?

A. O(1)

B. O(log n)

C. O(n)

D. O(n^2)

Answer: B) O(log n)

24. In a stack data structure, what happens when you attempt to remove an element when the stack is empty?

A. The element is successfully removed.

B. An error occurs, and the stack remains unchanged.

C. The last element in the stack is automatically pushed onto it.

D. The element is removed from the bottom of the stack.

Answer: B) An error occurs, and the stack remains unchanged.

25. In a doubly linked list, when deleting a node that is not the head or tail node, what operations are required to maintain the list's integrity?

A. Update the previous node's next pointer.

B. Update the next node's previous pointer.

C. Update both the previous and next nodes' pointers.

D. Delete the node without any updates.

Answer: C) Update both the previous and next nodes' pointers.

Stacks-Operations:

26. What operation is used to add an element to the top of a stack?

    A. Push

    B. Pop

    C. Peek

    D. Insert

    Answer: A) Push

27. In a stack data structure, what does the "pop" operation do?

    A. Adds an element to the top of the stack.

    B. Removes the element at the bottom of the stack.

    C. Removes the element from the top of the stack.

    D. Checks if the stack is empty.

    Answer: C) Removes the element from the top of the stack.

28. What is the primary purpose of the "peek" operation in a stack?

    A. To push an element onto the stack.

    B. To remove the top element from the stack.

    C. To check if the stack is empty.

    D. To view the top element without removing it.

    Answer: D) To view the top element without removing it.

29. In a stack implemented using an array, what happens when you attempt to push an element when the stack is full?

A. The element is added, and the array is automatically resized.

B. An error occurs, and the stack remains unchanged.

C. The last element in the stack is automatically removed.

D. The element is added at the bottom of the stack.

Answer: B) An error occurs, and the stack remains unchanged.

30. Consider a postfix expression evaluation using a stack. When encountering an operator, what is the correct order of operations for evaluating it?

A. Push operands onto the stack, then push the operator.

B. Push the operator onto the stack, then push operands.

C. Pop operands from the stack, apply the operator, and push the result.

D. Pop the operator from the stack, then pop operands.

Answer: C) Pop operands from the stack, apply the operator, and push the result.

Array and linked representations of stacks:

31. In an array-based representation of a stack, which index is typically used to access the top element of the stack?

A. 0

B. 1

C. -1

D. n (where n is the number of elements in the stack)

Answer: C) -1

32. What is the advantage of using a linked list representation over an array-based representation for a stack?

A. Linked lists have a fixed size.

B. Linked lists have faster random access.

C. Linked lists allow for dynamic resizing.

D. Linked lists consume less memory.

Answer: C) Linked lists allow for dynamic resizing.

33. In an array-based stack, what happens when you attempt to push an element when the stack is full?

A. The element is added at the top of the stack.

B. An error occurs, and the stack remains unchanged.

C. The last element in the stack is automatically removed.

D. The element is added at the bottom of the stack.

Answer: B) An error occurs, and the stack remains unchanged.

34. Consider a linked list-based implementation of a stack. Which operation is used to add an element to the top of the stack?

A. Append

B. Prepend

C. Delete

D. Search

Answer: B) Prepend

35. In an array-based stack, how can you efficiently check if the stack is empty?

A. By comparing the top index to -1.

B. By comparing the top index to 0.

C. By comparing the top index to the stack size.

D. By checking the entire array for empty elements.

Answer: A) By comparing the top index to -1.

Stack Applications:

36. Which of the following is a common application of a stack data structure?

A. Implementing a queue.

B. Sorting elements in ascending order.

C. Evaluating postfix expressions.

D. Representing a directed graph.

Answer: C) Evaluating postfix expressions.

37. In a web browser, which data structure is commonly used to implement the "Back" button functionality?

A. Stack

B. Queue

C. Linked List

D. Array

Answer: A) Stack

38. In a text editor, what is the primary application of the undo feature?

A. Reverting to the previous saved version.

B. Copying and pasting text.

C. Searching for text within the document.

D. Creating a new document.

Answer: A) Reverting to the previous saved version.

39. In a recursive algorithm, what data structure is often used to store intermediate results during recursion?

A. Stack

B. Queue

C. Heap

D. Hash Table

Answer: A) Stack

40. In a compiler, which stack-related data structure is used to manage function calls and local variables?

A. Expression Stack

B. Call Stack

C. Queue

D. Priority Queue

Answer: B) Call Stack

Queues-operations:

41. Which operation is used to add an element to the rear (end) of a queue?

    A. Dequeue

    B. Peek

    C. Enqueue

    D. Pop

    Answer: C) Enqueue

42. In a queue data structure, which operation is used to remove an element from the front of the queue?

    A. Enqueue

    B. Peek

    C. Dequeue

    D. Push

    Answer: C) Dequeue

43. Consider a circular queue. What is the advantage of using a circular queue over a regular queue?

A. Faster enqueue and dequeue operations.

B. Smaller memory footprint.

C. No need for a rear pointer.

D. It can only store a fixed number of elements.

Answer: A) Faster enqueue and dequeue operations.

44. In a priority queue, what is the primary factor used to determine the order in which elements are removed?

A. The order in which they were added.

B. Their position in the queue.

C. Their priority value.

D. Their size.

Answer: C) Their priority value.

45. In a multithreaded environment, what synchronization mechanism is often used to implement a thread-safe queue?

A. Mutex

B. Semaphore

C. Barrier

D. Spinlock

Answer: A) Mutex

Array and linked representations:

46. In an array-based representation of a queue, which index is used to access the front element of the queue?

A. 0

B. 1

C. -1

D. n (where n is the number of elements in the queue)

Answer: A) 0

47. In a linked list-based representation of a queue, which operation is used to add an element to the rear (end) of the queue?

A. Enqueue

B. Dequeue

C. Peek

D. Push

Answer: A) Enqueue

48. In an array-based queue, what happens when you attempt to enqueue an element when the queue is full?

A. The element is added at the rear of the queue.

B. An error occurs, and the queue remains unchanged.

C. The first element in the queue is automatically removed.

D. The element is added at the front of the queue.

Answer: B) An error occurs, and the queue remains unchanged.

49. In a linked list-based queue, what is the time complexity of the dequeue operation (removing the front element)?

A. O(1)

B. O(log n)

C. O(n)

D. O(n^2)

Answer: A) O(1)

50. In an array-based queue, what is the advantage of using a circular queue (circular buffer) over a regular queue?

A. Smaller memory footprint.

B. Faster enqueue and dequeue operations.

C. Simplicity of implementation.

D. No need for a rear pointer.

Answer: B) Faster enqueue and dequeue operations.

51. What is a dictionary in computer programming?

A. A collection of words and their meanings

B. A data structure that stores data in key-value pairs

C. A list of values indexed by number

D. A programming language

Answer: B) A data structure that stores data in key-value pairs

52. In a dictionary, what must keys be?

A. Only string type

B. Only numeric type

C. Immutable data type

D. Any data type

Answer: C) Immutable data type

53. What is the primary reason for using a dictionary over a list?

A. Dictionaries allow for faster element deletion.

B. Dictionaries store elements in ordered sequence.

C. Dictionaries can store elements of different data types.

D. Dictionaries provide faster access to elements using keys.

Answer: D) Dictionaries provide faster access to elements using keys.

54. How does a dictionary handle duplicate keys?

A. The dictionary keeps all duplicate keys.

B. The dictionary updates the value of the existing key with the last value assigned.

C. The dictionary raises an error when a duplicate key is added.

D. The dictionary creates a list of values for the duplicate key.

Answer: B) The dictionary updates the value of the existing key with the last value assigned.

55. Can dictionaries be nested in Python?

A. Yes, dictionaries can contain other dictionaries as values.

B. No, dictionaries cannot contain other dictionaries.

C. Yes, but only up to two levels deep.

D. No, dictionaries can only contain lists, not other dictionaries.

Answer: A) Yes, dictionaries can contain other dictionaries as values.

Linear list representation:

56. What is a linear list?

A. A collection of elements where each element is connected to two other elements except for the first and last.

B. A data structure where elements are arranged in sequence and each element is connected to its predecessor and successor.

C. A collection of elements stored in non-sequential memory locations.

D. A data structure where elements can be accessed in a circular manner.

Answer: B) A data structure where elements are arranged in sequence and each element is connected to its predecessor and successor.

57. Which of the following operations is not typically supported by a linear list data structure?

A. Insertion

B. Deletion

C. Searching

D. Sorting

Answer: D) Sorting

58. How is a linear list different from an array?

A. A linear list can only store items of the same data type, whereas an array can store items of different data types.

B. An array is a collection of items stored at contiguous memory locations, while a linear list does not require memory to be contiguous.

C. A linear list has a fixed size, whereas an array size can be dynamically altered.

D. There is no difference; linear list and array are terms used interchangeably.

Answer: B) An array is a collection of items stored at contiguous memory locations, while a linear list does not require memory to be contiguous.

59. In the context of linear list operations, what does 'traversal' mean?

A. Removing an element from the list.

B. Accessing each element of the list exactly once to perform some operation.

C. Searching for a specific element within the list.

D. Rearranging the elements of the list into a specific order.

Answer: B) Accessing each element of the list exactly once to perform some operation.

60. What is the time complexity of inserting an element at the beginning of a dynamic array-based list?

A. O(1)

B. O(log n)

C. O(n)

D. O(n^2)

Answer: C) O(n)

Skip list representation:

61. What basic structure does a skip list extend to allow for faster search operations?

A. Array

B. Binary tree

C. Linked list

D. Hash table

Answer: C) Linked list

62. Which of the following best describes the purpose of the multiple levels in a skip list?

A. To ensure that the list remains sorted

B. To decrease the storage space required

C. To allow for faster traversal by skipping over elements

D. To make it easier to implement

Answer: C) To allow for faster traversal by skipping over elements

63. How does a skip list improve search time compared to a standard linked list?

A. By using a binary search algorithm

B. By randomly accessing elements

C. By using multiple pointers to skip over nodes

D. By compressing the list to reduce the number of elements

Answer: C) By using multiple pointers to skip over nodes

64. What is the average time complexity of searching for an element in a skip list?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: B) O(log n)

65. In skip lists, how are the levels above the base level populated?

A. By inserting every nth element into the next level up

B. By randomly deciding, with a fixed probability, whether to insert an element into the next level up

C. By selecting only the maximum or minimum element for the next level

D. By duplicating every element in the base level to all upper levels

Answer: B) By randomly deciding, with a fixed probability, whether to insert an element into the next level up

Operations - insertion:

66. What does the insertion operation do in data structures?

A. Removes an element from the data structure

B. Searches for an element within the data structure

C. Adds a new element to the data structure

D. Replaces an existing element within the data structure

Answer: C) Adds a new element to the data structure

67. Where can elements be inserted in a queue?

A. Only at the beginning

B. Anywhere in the queue

C. Only at the end

D. Both at the beginning and the end

Answer: C) Only at the end

68. In a binary search tree, where is a new node inserted?

A. As the new root node of the tree

B. Based on the value of the node, to maintain the binary search tree property

C. At the leftmost position of the tree

D. At the rightmost position of the tree

Answer: B) Based on the value of the node, to maintain the binary search tree property

69. In a linked list, how is a new node inserted at a specific position?

A. By updating the head pointer to point to the new node

B. By replacing the node currently at that position with the new node

C. By adjusting the pointers of the adjacent nodes to include the new node

D. By deleting the last node and adding the new node at the specified position

Answer: C) By adjusting the pointers of the adjacent nodes to include the new node

70. What is the time complexity of inserting an element into a hash table?

A. O(1), assuming no collisions occur

B. O(log n), as elements are inserted in a sorted manner

C. O(n), due to the necessity of traversing the entire structure

D. O(n log n), as a rehashing of all elements might be required

Answer: A) O(1), assuming no collisions occur

Deletion and searching.

71. What is the result of a successful search operation in a data structure?

A. The new data structure without the searched element

B. The position or index of the searched element

C. A boolean value indicating the success of the operation

D. The value of the searched element

Answer: B) The position or index of the searched element

72. Which data structure allows for the fastest deletion of an element, assuming its position is known?

   A. Array

   B. Linked list

   C. Binary search tree

   D. Hash table

   Answer: B) Linked list

73. In binary search trees, what is a challenge when deleting a node with two children?

   A. Finding a replacement for the deleted node that maintains the binary search tree property

   B. Deleting both children nodes as well

   C. Rebalancing the entire tree

   D. Finding the node to delete

   Answer: A) Finding a replacement for the deleted node that maintains the binary search tree property

74. When searching for an element in a hash table with chaining, what is the worst-case time complexity?

   A. O(1)

   B. O(log n)

   C. O(n)

D. O(n log n)

Answer: Answer: C) O(n)

75. What must be done after deleting a node from a red-black tree to maintain its properties?

A. The tree must be completely rebuilt.

B. No action is required; the tree automatically maintains its properties.

C. Specific cases must be handled to re-balance the tree and restore its properties.

D. All remaining nodes must be re-inserted into the tree.

Answer: C) Specific cases must be handled to re-balance the tree and restore its properties.

Hash functions:

76. What is the primary purpose of a hash function in computing?

A. To encrypt data

B. To compress files into a smaller size

C. To map data of arbitrary size to data of fixed size

D. To sort data in ascending or descending order

Answer: C) To map data of arbitrary size to data of fixed size

77. What is a desirable property of a good hash function?

A. It produces the same hash value for different inputs.

B. It produces different hash values for similar inputs.

C. It allows easy retrieval of the original input from its hash value.

D. It minimizes the number of collisions.

Answer: D) It minimizes the number of collisions.

78. What happens when two different inputs produce the same output in a hashing process?

A. This is called a hashing overflow.

B. This situation is impossible in good hash functions.

C. This is known as a collision.

D. The hash function automatically rehashes one of the inputs.

Answer: C) This is known as a collision.

79. In the context of hash tables, what is chaining used for?

A. To extend the size of the hash table when it's full

B. To add additional security by encrypting the hash values

C. To handle collisions by linking entries with the same hash value

D. To chain multiple hash tables together for increased capacity

Answer: C) To handle collisions by linking entries with the same hash value.

80. What characteristic is most critical for a cryptographic hash function?

A. It must be reversible.

B. It must be fast to compute for any given input.

C. It must be infeasible to generate the original input given the hash output.

D. It must produce fixed-size output for any size of input data.

Answer: C) It must be infeasible to generate the original input given the hash output.

Collision resolution-separate chaining:

81. What is separate chaining in the context of hash tables?

A. A method to extend the hash table when it is full

B. A technique to keep all keys in a single linked list

C. A collision resolution technique that uses linked lists to store multiple elements with the same hash value

D. A method to compress the hash table size

Answer: C) A collision resolution technique that uses linked lists to store multiple elements with the same hash value.

82. What is a primary advantage of using separate chaining for collision resolution in a hash table?

A. It eliminates the need for rehashing

B. It can theoretically accommodate an unlimited number of collisions

C. It guarantees constant time operations

D. It reduces the size of the hash table

Answer: B) It can theoretically accommodate an unlimited number of collisions

83. How does separate chaining affect the search operation in a hash table?

A. It makes the search operation faster by directly accessing the element.

B. It can slow down the search operation if many elements are stored in the same chain.

C. It has no effect on the search operation time.

D. It makes the search operation unnecessary.

Answer: B) It can slow down the search operation if many elements are stored in the same chain.

84. In separate chaining, what factor significantly influences the efficiency of hash table operations?

A. The size of the data elements

B. The load factor and the quality of the hash function

C. The number of hash tables

D. The length of the key

Answer: B) The load factor and the quality of the hash function.

85. Which of the following strategies can help improve the performance of a hash table using separate chaining?

A. Increasing the number of chains beyond the number of elements in the table

B. Keeping the length of each chain short by maintaining a low load factor

C. Using a single, large chain for all elements

D. Decreasing the size of the hash table

Answer: B) Keeping the length of each chain short by maintaining a low load factor

86. What is open addressing in the context of hash tables?

A. A method where collisions are resolved by storing elements outside of the table.

B. A technique where each cell of a hash table stores a single key-value pair.

C. A collision resolution strategy that involves finding another slot within the table.

D. A method to increase the capacity of the hash table dynamically.

Answer: C) A collision resolution strategy that involves finding another slot within the table.

87. How does linear probing resolve collisions?

A. By linking all elements with the same hash into a list.

B. By placing the collided item in the next empty slot found in a circular manner.

C. By rehashing the key with a secondary hash function.

D. By storing collided elements in a separate data structure.

Answer: B) By placing the collided item in the next empty slot found in a circular manner.

88. What is a primary disadvantage of linear probing?

A. It requires additional memory outside of the hash table.

B. It cannot handle deletions efficiently.

C. It leads to clustering of entries, which can degrade performance.

D. It is incompatible with fixed-size hash tables.

Answer: C) It leads to clustering of entries, which can degrade performance.

89. In the context of linear probing, what is meant by the term "clustering"?

A. A group of keys that hash to the same value.

B. The tendency of consecutive slots to become occupied, leading to increased collision resolution times.

C. The process of grouping similar keys together for faster access.

D. A technique for reducing the size of the hash table.

Answer: B) The tendency of consecutive slots to become occupied, leading to increased collision resolution times.

90. How can the issue of primary clustering in linear probing be mitigated?

A. By using a larger hash table.

B. By implementing a double hashing technique instead of linear probing.

C. By periodically rehashing all keys in the table.

D. By ensuring that all keys are unique.

Answer: B) By implementing a double hashing technique instead of linear probing.

Quadratic probing:

91. How does quadratic probing differ from linear probing when resolving collisions in a hash table?

A. Quadratic probing uses a linear function to find the next slot, while linear probing uses a quadratic function.

B. Quadratic probing places the collided item in the next available slot without following any specific sequence.

C. Quadratic probing uses a quadratic function of the form i2 (where i2 is the attempt number) to find the next slot, whereas linear probing simply moves to the next slot.

D. Quadratic probing decreases the search space by half after each collision, while linear probing does not.

Answer: C) Quadratic probing uses a quadratic function of the form i2 (where i2 is the attempt number) to find the next slot, whereas linear probing simply moves to the next slot.

92. What is a primary benefit of quadratic probing over linear probing?

A. It requires less memory.

B. It eliminates the possibility of collisions.

C. It reduces the problem of primary clustering.

D. It guarantees to find an empty slot for insertion if the table is not full.

Answer: C) It reduces the problem of primary clustering.

93. What challenge does quadratic probing face that linear probing does not?

A. It can result in secondary clustering.

B. It is unable to handle deletions.

C. It requires a secondary hash function.

D. It cannot be used in fixed-size hash tables.

Answer: A) It can result in secondary clustering.

94. What characterizes double hashing as a collision resolution technique in hash tables?

A. Using a single hash function to determine the probe sequence

B. Using two hash functions to determine the initial position and the probe sequence step size

C. Inserting all elements into a single linked list to avoid collisions

D. Doubling the size of the hash table upon each collision

Answer: B) Using two hash functions to determine the initial position and the probe sequence step size

95. How does double hashing mitigate the clustering problem observed in linear and quadratic probing?

A. By ensuring that the probe sequence is the same for all keys

B. By using a variable step size, derived from a second hash function, to distribute entries more uniformly

C. By rehashing all keys upon detecting a collision

D. By limiting the hash table to store a maximum number of elements

Answer: B) By using a variable step size, derived from a second hash function, to distribute entries more uniformly

96. Why is double hashing considered more efficient in avoiding collisions than linear or quadratic probing?

A. Because it uses a complex mathematical algorithm to completely eliminate collisions

B. Because the step size varies for each key, reducing the chance of secondary clustering

C. Because it automatically resizes the hash table for each new insertion

D. Because it places all collided elements into a separate buffer

Answer: B) Because the step size varies for each key, reducing the chance of secondary clustering

97. What is the main purpose of rehashing in hash tables?

A. To decrease the size of the hash table

B. To increase the load factor beyond 1

C. To improve the distribution of elements and reduce collisions by resizing the hash table and reapplying a hash function

D. To convert the hash table into another data structure

Answer: C) To improve the distribution of elements and reduce collisions by resizing the hash table and reapplying a hash function

98. When is rehashing typically performed in a hash table?

A. When the hash table is empty to optimize future operations

B. After every insertion to ensure maximum efficiency

C. When the load factor reaches or exceeds a certain threshold, indicating the table is becoming too crowded

D. At fixed intervals, regardless of the number of elements in the table

Answer: C) When the load factor reaches or exceeds a certain threshold, indicating the table is becoming too crowded

99. What is a potential downside of rehashing in a hash table?

A. It can temporarily decrease the search speed for elements.

B. It can lead to a permanent loss of data if not done correctly.

C. It requires additional memory for the rehashing process, which can be computationally expensive.

D. It reduces the load factor too much, making the hash table inefficient.

Answer: C) It requires additional memory for the rehashing process, which can be computationally expensive.

100. What is the main feature of extendible hashing?

A. The hash table size doubles whenever the load factor exceeds a certain threshold.

B. The hash function changes dynamically based on the current size of the hash table.

C. It uses a fixed-size hash table to store all key-value pairs.

D. It allows the hash table to expand or contract dynamically, adjusting the directory size without rehashing all keys.

Answer: D) It allows the hash table to expand or contract dynamically, adjusting the directory size without rehashing all keys.

101. What property must all Binary Search Trees satisfy?

A. Each node has exactly two children.

B. The left child of a node must have a greater value than the node.

C. The value of a left child is less than its parent, and the value of a right child is greater.

D. Every node must have at least one child.

Answer: C) The value of a left child is less than its parent, and the value of a right child is greater.

102. What is the time complexity of searching for an element in a balanced binary search tree?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: B) O(log n)

103. Which of the following operations can be performed efficiently using a Binary Search Tree?

A. Finding the minimum and maximum elements

B. Calculating the average of all elements

C. Reversing the order of elements

D. Finding the mode of the elements (the value that appears most frequently)

Answer: A) Finding the minimum and maximum elements

104. In a BST, what is the worst-case time complexity of inserting a new node?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: C) O(n), which occurs when the BST degenerates into a linked list (i.e., all elements are inserted in ascending or descending order).

105. How does deletion of a node with two children generally get handled in a Binary Search Tree?

A. By removing the node and promoting its left child as the new parent

B. By finding the in-order successor (the smallest node in the right subtree) or in-order predecessor (the largest node in the left subtree) to replace the deleted node

C. By merging the left and right subtrees into a single subtree

D. By randomly selecting one of the children to replace the deleted node.

Answer: B) By finding the in-order successor (the smallest node in the right subtree) or in-order predecessor (the largest node in the left subtree) to replace the deleted node

Implementation:

106. Which of the following data structures is primarily used to implement a FIFO (First In, First Out) queue?

A. Array

B. Linked List

C. Stack

D. Tree

Answer: B) Linked List

107. In object-oriented programming, what is encapsulation?

A. A design pattern that ensures a class has only one instance and provides a global point of access to it

B. The practice of keeping fields within a class private, then providing public methods to access and modify the values of these fields

C. The ability of an object to take on many forms

D. A technique for defining multiple methods with the same name but different parameters

Answer: B) The practice of keeping fields within a class private, then providing public methods to access and modify the values of these fields.

108. What is the primary purpose of the 'Decorator' design pattern?

A. To ensure a class has only one instance

B. To add new responsibilities to an object dynamically without altering its structure

C. To simplify complex network of classes by introducing a middle layer

D. To define new operations on objects without changing the objects' classes

Answer: B) To add new responsibilities to an object dynamically without altering its structure

109. Which algorithm is most suitable for searching for an item in a sorted array?

A. Linear search

B. Bubble sort

C. Binary search

D. Quick sort

Answer: C) Binary search

110. In a graph represented using an adjacency list, what is the time complexity of checking if there is an edge between two vertices?

A. O(1)

B. O(log n)

C. O(n)

D. O(V + E)

Answer: C) O(n)

Operations- Searching

111. Which of the following is a linear search algorithm?

A. Binary Search

B. Depth-First Search

C. Sequential Search

D. Interpolation Search

Answer: C) Sequential Search

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

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: A) O(1), occurring when the central element of the search space is the target value.

113. In which scenario is interpolation search more efficient than binary search?

A. When the elements are uniformly distributed

B. When searching for the smallest or largest element

C. When the array elements are in descending order

D. When the array has a lot of duplicate values

Answer: A) When the elements are uniformly distributed.

114. Which data structure is most efficient for implementing a Breadth-First Search (BFS) algorithm?

A. Stack

B. Queue

C. Priority Queue

D. Linked List

Answer: B) Queue

115. What is the time complexity of searching for an element in a balanced binary search tree?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: B) O(log n)

116. Which of the following data structures allows for the fastest insertion at the beginning of the structure?

A. Array

B. Linked List

C. Binary Search Tree

D. Hash Table

Answer: B) Linked List

117. In a queue data structure, where does deletion occur?

A. At the front of the queue

B. At the back of the queue

C. At any random location within the queue

D. In the middle of the queue

Answer: A) At the front of the queue

118. What is the time complexity of inserting a new node in a doubly linked list?

A. O(1), assuming you have a pointer to the position where you want to insert

B. O(log n)

C. O(n)

D. O(n log n)

Answer: A) O(1)

119. In a binary search tree, what is the time complexity of deleting a node that has two children?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: B) O(log n)

120. What is the worst-case time complexity of deleting an element from a min-heap?

A. O(1)

B. O(log n)

C. O(n)

D. O(n log n)

Answer: C) O(n)