1. Disks piled up one above the other represent a

Answer Answer Option

Queue

Linked List

Array

Stack

In a Stack, the elements are added or removed from the top of the stack. The topmost element is the only one that can be accessed, and new elements can be added on top of the existing ones.

Similarly, in a disk pile, new disks are added to the top of the stack, and only the top disk can be accessed or removed. Hence, a disk pile represents a Stack data structure.

1. In a stack, if a user tries to remove an element from empty stack it is called \_\_\_\_\_\_\_\_

Answer Answer Option

Empty collection

Garbage Collection

Overflow

Underflow

The answer to your question is Underflow.

When a user tries to remove an element from an empty stack, it results in an error called "underflow." This is because the stack is empty and has no element to remove. It is a runtime error that occurs when we try to pop an element from an empty stack.

Overflow, on the other hand, is a different type of error that occurs when we try to add an element to a full stack. It happens when the stack is already full, and we cannot add any more elements.

1. What does the following fragment of code do with a linked list?

current = first;

while(current!=null){current=current.next;}

Answer Answer Option

traversal

Search

sort

all of the above

The given code fragment iterates through a linked list starting from the first node and going to the last node. It accomplishes this by setting the current node to the first node of the linked list and traversing the list by moving the current node to the next node until it reaches the end of the list (i.e., current node becomes null).

Therefore, this code performs a linked list traversal, which is a process of visiting and processing all the nodes of a linked list. It is a common operation performed on a linked list to access its elements sequentially.

1. tacks can be used by an operating system to perform a function call.

Answer Answer Option

True

False

The answer to your question is True.

Stacks can be used by an operating system to perform a function call. When a program calls a function, the computer stores the return address (i.e., the address of the instruction to execute after the function call) and the function's local variables and parameters on the stack. The function's code is then executed, and when it's finished, the return address is retrieved from the stack, and the program continues from where it left off.

This process of storing the return address and local variables on the stack is known as a stack frame, and it's a fundamental concept in computer science. In fact, many programming languages and operating systems use stacks to implement function calls and manage memory.

1. Consider the following operation performed on a stack of size 5.

Push(20);

Pop();

Push(50);

Push(10);

Pop();

Push(70);

Pop();

Push(60);

After the completion of all operation, the number of elements present in stack is?

1

2

0

3

After performing the given operations on the stack, the number of elements present in the stack will be 2.

Let's walk through the operations step by step:

Push(20): The stack now contains one element, 20.

Pop(): The top element (20) is removed from the stack. The stack is now empty.

Push(50): The stack now contains one element, 50.

Push(10): The stack now contains two elements, 10 and 50.

Pop(): The top element (10) is removed from the stack. The stack now contains one element, 50.

Push(70): The stack now contains two elements, 70 and 50.

Pop(): The top element (70) is removed from the stack. The stack now contains one element, 50.

Push(60): The stack now contains two elements, 60 and 50.

So, after all these operations, the number of elements present in the stack is 2.

1. How many pointers are necessarily changed for the insertion at middle point in a singly Linked List?

For the insertion of a node at the middle point of a singly linked list, we need to modify two pointers: the pointer of the node before the insertion point and the pointer of the node at the insertion point.

Let's say we want to insert a new node with value x between two nodes with values a and b. To do this, we need to perform the following steps:

Create a new node with value x.

Traverse the linked list until we find the node with value a.

Modify the next pointer of the node with value a to point to the new node.

Modify the next pointer of the new node to point to the node with value b.

Therefore, we only need to change the two nodes’ pointers immediately before and after the insertion point, and no other pointers need to be modified.

1. How many pointers are necessarily changed for the insertion at middle point in a singly Linked List?

Option

One

Four

Three

Two

For the insertion of a node at the middle point of a singly linked list, we need to modify two pointers: the pointer of the node before the insertion point and the pointer of the node at the insertion point.

Therefore, the answer to your question is Two.

1. What is the time complexity to count the number of elements in the linked list?

Option

O(logn)

O(n)

O(n2)

O(1)

The time complexity to count the number of elements in a linked list is O(n), where n is the number of nodes in the linked list.

To count the number of elements in a linked list, we need to traverse the entire list and count the number of nodes. In the worst case, we need to visit all the nodes of the linked list to count its length, so the time complexity of this operation is proportional to the number of nodes in the list, which is O(n).

Therefore, the answer to your question is O(n).

1. In linked lists, there are no NULL links/reference in a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Option

I don’t’ know

doubly linked list

Single linked list

Circular linked list

In a circular linked list, there are no NULL links/references.

A circular linked list is a linked list where the last node points to the first node, forming a loop. This means that there is no end to the linked list, and any traversal of the list will eventually return to the starting node. Therefore, in a circular linked list, there are no NULL pointers or references, as every node has a valid next pointer that points to another node in the list.

In contrast, in a singly or doubly linked list, the last node has a NULL next pointer, indicating the end of the list.

1. Linked lists are best suited\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Option

none of above

for both of above situation

for relatively permanent collections of data

for the size of the structure and the data in the structure are constantly changing

Linked lists are best suited for the situation where the size of the structure and the data in the structure are constantly changing.

Linked lists are dynamic data structures that can grow or shrink as needed, making them ideal for situations where the size of the data being stored is not fixed. They are particularly useful when we need to add or remove elements frequently, as these operations can be performed in constant time (O(1)) by adjusting pointers.

In contrast, arrays are better suited for relatively permanent collections of data, where the size of the data is fixed and known in advance. This is because arrays are typically implemented as contiguous blocks of memory, so adding or removing elements can be expensive in terms of time and memory.

Therefore, the answer to your question is "for the size of the structure and the data in the structure are constantly changing".

1. \_\_\_\_\_\_\_\_\_\_ linked list is a two way list.

Option

both 1 and 2

Circular

doubly

none

ADoubly linked list is a two way list.

In a doubly linked list, each node contains two pointers, one pointing to the next node in the list and the other pointing to the previous node. This allows for efficient traversal of the list in both forward and backward directions, making doubly linked lists useful in situations where we need to traverse the list in both directions.

Therefore, the answer to your question is "doubly".