DSA

Hitarth Patel

150096724046

Jensen Huang

Questions on Circular Linked List

—----------------------------------------------------------------------------------------------------------------------------

Q1: What is a Circular Linked List (CLL), and how is it different from a Singly Linked List?

→

A Circular Linked List (CLL) is a type of linked list where the last node points back to the first node, forming a circle. This is in contrast to a Singly Linked List, where the last node points to null. In a CLL, there is no distinct starting or ending point, and we can start traversing from any node.

—----------------------------------------------------------------------------------------------------------------------------

Q2: What are the real-world applications of Circular Linked Lists?

→

Some real-world applications of Circular Linked Lists include:

* Traffic Light Control System: We can use a CLL to manage the sequence of traffic lights.
* CPU Scheduling: CLL can be used to implement the Round Robin scheduling algorithm.
* Memory Management: CLL can be used to manage memory allocation and deallocation.

—----------------------------------------------------------------------------------------------------------------------------

Q3: What are the advantages of using a Circular Linked List?

→

The advantages of using a Circular Linked List include:

* Efficient Use of Memory: CLL can be used to implement queues and stacks, which can be more memory-efficient than arrays.
* Faster Insertion and Deletion: CLL allows for faster insertion and deletion of nodes compared to arrays.
* Improved Cache Performance: CLL can exhibit better cache locality compared to arrays.

—----------------------------------------------------------------------------------------------------------------------------

Q4: How do you traverse a Circular Linked List starting from the head?

→

To traverse a Circular Linked List starting from the head, you can use the following steps:

1. Start at the head node.
2. Traverse the list until you reach the node that points back to the head node.
3. Use a loop to iterate through the nodes, and use a conditional statement to break the loop when you reach the starting node again.

—----------------------------------------------------------------------------------------------------------------------------

Q5: How do you check if a given value exists in a Circular Linked List?

→

To check if a given value exists in a Circular Linked List, you can use the following steps:

1. Start at the head node.
2. Traverse the list until you reach the node that points back to the head node.
3. Inside the loop, check if the current node's value matches the given value.
4. If a match is found, return true. If the loop completes without finding a match, return false.

—----------------------------------------------------------------------------------------------------------------------------

Q6: What is a doubly Linked List?

→

A doubly linked list is a type of linked list where each node has two pointers: one pointing to the next node (like in a singly linked list) and another pointing to the previous node. This allows for efficient traversal in both forward and backward directions.

—----------------------------------------------------------------------------------------------------------------------------

Q7: Write the algorithm for insertion at the beginning in a doubly linked list?

→

Algorithm:  
1. Create a new node with the given data.  
2. If the list is empty:  
 - Set the head to the new node.  
 - Make the new node’s next and previous point to itself.  
3. Otherwise:  
 - Find the last node.  
 - Set the new node’s next to the head.  
 - Set the new node’s prev to last.  
 - Update the last node next to the new node.  
 - Update head’s prev to new node.  
 - Set the head to the new node.

—----------------------------------------------------------------------------------------------------------------------------

Q8: How to create a node in a doubly linked list?

→

To create a node in a doubly linked list, you can use the following steps:

1. Define a structure/class with:  
 - An integer data field.  
 - A pointer to the next node.  
 - A pointer to the previous node.  
 2. Initialize a node with given data.  
 3. Set the next and prev pointers to NULL.

—----------------------------------------------------------------------------------------------------------------------------