While folks are joining

Get you laptops ready and login to www.crio.do.
We will be coding away in the session!

DSA-1

Session 9

What’s for this session?

- Linked List
  - Introduction
  - CRUD operations
- Problems
  - Insert into Linked List
  - Split the Linked List

Library for Linked List?

- Linked List doesn’t have a library since its a primitive data structure like array and is used to implement other complex data structures (Stack ,Queue etc.).
- You will have to create this library of Insert, Delete, Search, Update operations by yourself.
  - It is very important that you know how to write these methods.
  - That is what we will do in this session.

Linked List

- **Chain with links -** If one link is broken, we cannot get to any of the others
- **Properties** (compare to array)
  - Non Sequential Memory
  - Size is flexible, can grow easily
  - Uses more memory than array(store pointers)
  - Insert and Delete are easy (don’t have to move elements)
  - No POSITIONal access
  - Cannot directly lookup a value (Search is sequential)
  - Cannot lookup previous element in Singly Linked List
- **When to use**
  - When you don’t know the number of items
  - Need fast Insert and Delete

Linked List - Applications

- Implement DS like Stacks, Queues and Trees
- Implement Graphs
    - Adjacency list representation of Graph
- Implement Hash Tables
    - Each Bucket of the hash table can be a linked list (Open chain hashing)

Linked List - Frequently asked problems

- Reverse a Linked List
- Detect a cycle in a Linked List
- Find the middle node in a Linked List
- Insert an element into a sorted Circular List
- Remove Duplicates from a Linked List
- Merge two sorted Lists
- Delete Kth to last element in a Linked List
- Partition a list into multiple lists
- Rotate a Linked List
- Implement a Doubly Linked Circular List

Node Structure

- What is Node in a Linked List?
  - Building Block
  - Node itself is not a linked list

JAVA

```
public class Node {
  public int val;
  public Node next;

  public Node(int _val) {
    val = _val;
    next = null;
  }
}
```

C++

```
class Node {
public:
  int val;
  Node* next;

  Node(int _val) {
    val = _val;
    next = NULL;
  }
};
```

Python

```
class Node:
    def __init__(self, _val):
        self.val = _val
        self.next = None
```

JavaScript

```
class Node{

  constructor(val){

    this.val = val;

    this.next= null;

  }

}
```

CRUD operations

- Create
  - Insert the first node
  - Insert a node at the beginning, middle and end
- Read
  - Traverse and read values
- Update
  - Traverse and update values
- Delete
  - Remove the only node
  - Remove a node at the beginning, middle and end
- Reverse

Note: Linked List will be covered in more detail in a pack of its own, later on

How to Approach Problems?

For any given problem, following these milestones will help you solve the problem systematically:

- **Milestone 1** - Understand the problem statement and confirm your understanding with some examples or test cases, including edge cases.
- **Milestone 2** - Think about approaches and select the best one you know. Explain your approach to a 10 year old. Write the pseudocode with function breakdown.
- **Milestone 3** - Expand pseudocode to code
- **Milestone 4** - Demonstrate that the solution works

Activity 1 - Insert into Linked List

- With DSA problems, you will be given the node structure that you can use and solve the problems.
- You won’t see the main() method in the code stubs on the platform. That is in a separate file that you need not worry about.

Activity 2 - Split the Linked List

Other types of Linked Lists

- Doubly Linked List
- Circular Linked List

Questions?

Take home exercises

- Remove from Linked List
- Search a Linked List

To be solved before the next session on Tuesday, 7:30 PM

Feedback

Thank you for joining in today. We’d love to hear your thoughts and feedback.

https://bit.ly/dsa-nps

Thank you