Home Works

Homeworks

CSE 2220 - Data Structures and Algorithm

Lab 1: Introduction of Data types: primitive and non-primitive, Data structure: linear and nonlinear, Abstract data type.

Lab 2: Representation of arrays, Applications of arrays, Sparse matrix and its representation.

1. **Reverse an Array**
   Reverse the elements of an array in place (i.e, without using another array).
2. **Rotate Array (Left / Right)**
   Rotate the array to the left by `k` positions (or to the right).
   Example: `[1,2,3,4,5]` left-rotated by 2 → `[3,4,5,1,2]`.
3. **Move Zeroes**
   Given an array, move all zeros to the end while maintaining the relative order of non-zero elements.
   Example: `[0,1,0,3,12]` → `[1,3,12,0,0]`.
4. **Sliding Window Maximum / Minimum**
   Given an array and a window size `k`, compute the maximum (or minimum) for each sliding window of size `k`.

Lab 3: Implementation of Traversal and Insertion Operations in a Singly Linked List

1. **Inserting a Node at the Beginning**
   Write a function to insert a node at the **beginning** of a linked list.
   - Input: LinkedList = 10 -> 20 -> 30, Inserting 5 at the beginning.
   - Expected Output: 5 -> 10 -> 20 -> 30
2. **Inserting a Node at the End**
   Implement a function to insert a node at the **end** of the linked list.
   - Input: LinkedList = 10 -> 20 -> 30, Inserting 40 at the end.
   - Expected Output: 10 -> 20 -> 30 -> 40
3. **Inserting a Node After a Given Node**
   [To insert a node after a given node, we traverse the list until we find the node with the desired value, and then adjust the pointers to insert the new node.]

Homeworks

CSE 2220 - Data Structures and Algorithm

- Input: 9 10 20 30 31
- insert_node_after_item(20, 25);
- Output: 9 10 20 25 30 31

4. **Inserting a Node before a Given Node**

   [To insert a node before a given node, we traverse the list until we find the node with the desired value, and then adjust the pointers to insert the new node.]

   - Input: 9 10 20 25 30 31
   - insert_node_before_item(30,27);
   - Output: 9 10 20 25 27 30 31
5. **Write a function to create a linked list from an array of values.**
   - Input: Array = [5, 10, 15, 20]
   - Expected Output: LinkedList = 5 -> 10 -> 15 -> 20

Lab 4: Deletion Operations in a Singly Linked List and Implementation of Doubly Linked List

1. **Write a function to delete a node with a specific value from the linked list.**
   - Input: LinkedList = `10 -> 20 -> 30 -> 40`, Delete node `20`.
   - Expected Output: `10 -> 30 -> 40`
2. **Implement a function to delete the node after a given value.**
   - Input: LinkedList = `10 -> 20 -> 30 -> 40`, Delete node after `20`.
   - Expected Output: `10 -> 20 -> 40`
3. **Write a function to search for a value in the linked list.**
   - Input: LinkedList = `10 -> 20 -> 30`, Search for `20`.
   - Expected Output: True (`20` found in the list)
4. **Write a function to delete the first node of a linked list.**
   - Input: LinkedList = `10 -> 20 -> 30`
   - Expected Output: `20 -> 30`
5. **Write a program to reverse a doubly linked list by changing the links (without using extra space).**
6. **Given a sorted DLL, insert a new node such that the list remains sorted**

Homeworks

CSE 2220 - Data Structures and Algorithm

Lab 5: Basic Concepts and Applications of Queue and Stack.

Stack:

1. **Reverse a String using Stack**
    - Write a program to reverse a string using a stack.
2. **Balanced Parentheses**
    - Given an expression string with parentheses `(` and `)`, check whether the parentheses are balanced or not using a stack.
3. **Sort a Stack**
    - Given a stack, write a program to sort its elements using recursion (you can only use stack operations like push, pop, etc.).
4. **Next Greater Element**
    - Given an array, for each element, find the next greater element. The next greater element is the first larger number to the right of the current element using a stack.
5. **Implement Two Stacks in an Array**
    - Implement two stacks in a single array without wasting space.
6. **Stock Span Problem**
    - Write a program to calculate the span of stock's price for all days using a stack. The stock span on a given day is defined as the maximum number of consecutive days the stock price was less than or equal to the current price.
7. **Min Stack**
    - Design a stack that supports `push`, `pop`, `top`, and retrieving the minimum element in constant time.

Queue:

1. **Generate Binary Numbers**
    - Given a number `n`, generate and print binary numbers from 1 to `n` using a queue.
2. **Reverse a Queue**
    - Write a program to reverse the elements of a queue.
3. **Implement Stack using Queues**
    - Implement a stack using two queues.
4. **Circular Queue Implementation**
    - Implement a circular queue with fixed size using an array.

Homeworks

CSE 2220 - Data Structures and Algorithm

5. **Interleave First Half and Second Half of a Queue**
    - Given a queue of integers, interleave the first half of the queue with the second half.
6. **First Non-Repeating Character in a Stream**
    - Given a stream of characters, find the first non-repeating character at any point in the stream using a queue.
7. **Sliding Window Maximum**
    - Given an array and an integer `k`, find the maximum element in each sliding window of size `k` using a deque (double-ended queue).
8. **Implement Deque (Double-Ended Queue)**
    - Implement a deque with operations `insertFront`, `insertRear`, `deleteFront`, `deleteRear`, and checking if the deque is empty or full.