Slide 1: Title Slide

- Title: Introduction to Data Structures and Algorithms
- Subtitle: Queue Operations, Double Linked List, Bubble Sort, and Minimum Spanning Tree
- Author: [Your Name]

Slide 2: Agenda

- Introduction
- Queue Operations
- Double Linked List
- Bubble Sort
- Minimum Spanning Tree
- Conclusion

Slide 3: Introduction

- What are Data Structures?
  - Definition
  - Importance in computer science

Slide 4: Queue Operations

- Definition: A linear data structure that follows the First In First Out (FIFO) principle

Slide 5: Basic Queue Operations

- Operations:
  - Enqueue (Insert element)
  - Dequeue (Remove element)
  - Peek (Get front element)
  - IsEmpty (Check if queue is empty)

Slide 6: Queue Operations - Java Example

- Java Example:
  Java

  Copy

```
  import java.util.LinkedList;
• import java.util.Queue;
•
• public class QueueExample {
•     public static void main(String[] args) {
•         Queue<Integer> queue = new LinkedList<>();
•         queue.add(1); // Enqueue
•         queue.add(2);
•         queue.add(3);
•
•         System.out.println("Front element: " + queue.peek()); // Peek
```

```
        System.out.println("Removed element: " + queue.remove()); // Dequeue
    }
}

```

Slide 7: Double Linked List

- Definition: A linear data structure where each element points to the next and previous elements

Slide 8: Basic Double Linked List Operations

- Operations:
  - Insert (At beginning, end, or specific position)
  - Delete (From beginning, end, or specific position)
  - Traverse (Iterate through elements)

Slide 9: Double Linked List - Java Example

- Java Example:
  Java

  Copy

```
class Node {
    int data;
    Node prev;
    Node next;

    Node(int data) {
        this.data = data;
    }
}

public class DoubleLinkedList {
    Node head;

    // Add element at the end
    public void append(int data) {
        if (head == null) {
            head = new Node(data);
            return;
        }
        Node current = head;
        •         while (current.next != null) {
            •             current = current.next;
            •         }
        •         Node newNode = new Node(data);
        •         current.next = newNode;
        •         newNode.prev = current;
        •     }
        • }
        •
```

Slide 10: Bubble Sort

- Definition: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order

Slide 11: Steps of Bubble Sort

- Steps:
  - Compare adjacent elements
  - Swap if necessary
  - Repeat until sorted

Slide 12: Bubble Sort - Java Example

- Java Example:
  Java

  Copy

```
  public class BubbleSort {
•     public static void main(String[] args) {
•         int[] arr = {5, 1, 4, 2, 8};
•         bubbleSort(arr);
•         for (int num : arr) {
•             System.out.print(num + " ");
•         }
•     }
•
•     public static void bubbleSort(int[] arr) {
•         int n = arr.length;
•         for (int i = 0; i < n - 1; i++) {
•             for (int j = 0; j < n - 1 - i; j++) {
•                 if (arr[j] > arr[j + 1]) {
•                     int temp = arr[j];
```

```
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
}

```

Slide 13: Minimum Spanning Tree

- Definition: A subset of the edges of a connected, edge-weighted graph that connects all the vertices without any cycles and with the minimum possible total edge weight

Slide 14: Minimum Spanning Tree Algorithms

- Algorithms:
  - Kruskal's Algorithm
  - Prim's Algorithm

Slide 15: Minimum Spanning Tree - Java Example

- Java Example using Kruskal's Algorithm:
  Java

  Copy

```
import java.util.*;

class Edge implements Comparable<Edge> {
    int src, dest, weight;

    public int compareTo(Edge compareEdge) {
        return this.weight - compareEdge.weight;
    }
}

class Subset {
    int parent, rank;
}

public class Kruskal {
    int V, E;
    Edge[] edge;
```

-
- // Utility methods omitted for brevity
-
- public void kruskalMST()