Maha Gowri S **11/11/2024**

1.0/1 Knapsack Problem

Code:

```
import java.util.Scanner;

class KnapsackProblem {

    static int knapSack(int W, int wt[], int val[], int n) {
        if (n == 0 || W == 0)
            return 0;

        if (wt[n - 1] > W)
            return knapSack(W, wt, val, n - 1);

        return Math.max(knapSack(W, wt, val, n - 1),
                        val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1));
    }

    public static void main(String args[]) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of items: ");
        int n = scanner.nextInt();

        int[] profit = new int[n];
        int[] weight = new int[n];

        System.out.println("Enter the profits of the items:");
        for (int i = 0; i < n; i++) {
```

```
            profit[i] = scanner.nextInt();
        }

        System.out.println("Enter the weights of the items:");
        for (int i = 0; i < n; i++) {
            weight[i] = scanner.nextInt();
        }

        System.out.print("Enter the capacity of the knapsack: ");
        int W = scanner.nextInt();

        System.out.println("Maximum profit: " + knapSack(W, weight, profit, n));

        scanner.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac KnapsackProblem.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java KnapsackProblem
Enter the number of items: 3
Enter the profits of the items:
60 100 120
Enter the weights of the items:
10 20 30
Enter the capacity of the knapsack: 50
Maximum profit: 220

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time Complexity: O(2^N)

2.Floor in a sorted array

Code:

```
import java.util.Scanner;

class sorted {

    static int floorSearch(int arr[], int n, int x) {
        if (x >= arr[n - 1])
            return n - 1;

        if (x < arr[0])
            return -1;

        for (int i = 1; i < n; i++)
            if (arr[i] > x)
                return (i - 1);

        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();

        int arr[] = new int[n];
        System.out.println("Enter " + n + " sorted elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
```

```
        }

        System.out.print("Enter the number to find its floor: ");
        int x = scanner.nextInt();

        int index = floorSearch(arr, n, x);
        if (index == -1)
            System.out.println("Floor of " + x + " doesn't exist in the array.");
        else
            System.out.println("Floor of " + x + " is " + arr[index]);
            scanner.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac sorted.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java sorted
Enter the size of the array: 7
Enter 7 sorted elements of the array:
1 2 4 6 10 12 14
Enter the number to find its floor: 7
Floor of 7 is 6

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time Complexity: O(log n)

3. Check equal arrays

Code:

```
import java.util.Arrays;
import java.util.Scanner;

class Equalarrays {

    public static boolean areEqual(int arr1[], int arr2[]) {
        int N = arr1.length;
        int M = arr2.length;

        if (N != M)
            return false;

        Arrays.sort(arr1);
        Arrays.sort(arr2);

        for (int i = 0; i < N; i++)
            if (arr1[i] != arr2[i])
                return false;

        return true;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the first array: ");
        int n = scanner.nextInt();
        int arr1[] = new int[n];
```

```
        System.out.println("Enter " + n + " elements for the first array:");
        for (int i = 0; i < n; i++) {
            arr1[i] = scanner.nextInt();
        }

        System.out.print("Enter the size of the second array: ");
        int m = scanner.nextInt();
        int arr2[] = new int[m];
        System.out.println("Enter " + m + " elements for the second array:");
        for (int i = 0; i < m; i++) {
            arr2[i] = scanner.nextInt();
        }

        if (areEqual(arr1, arr2))
            System.out.println("Yes");
        else
            System.out.println("No");
            scanner.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac Equalarrays.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java Equalarrays
Enter the size of the first array: 5
Enter 5 elements for the first array:
3 5 2 5 2
Enter the size of the second array: 5
Enter 5 elements for the second array:
2 3 5 5 2
Yes

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time Complexity: O(N*log(N))

4.Palindrome linked list

Code:

```
import java.util.Scanner;


class Node {
    int data;
    Node next;
    Node(int d) {
        data = d;
        next = null;
    }
}

class Linkedlist {

    static Node reverseList(Node head) {
        Node prev = null;
        Node curr = head;
        Node next;
        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        return prev;
    }

    static boolean isIdentical(Node n1, Node n2) {
```

```
    while (n1 != null && n2 != null) {
        if (n1.data != n2.data)
            return false;
        n1 = n1.next;
        n2 = n2.next;
    }
    return true;
}

static boolean isPalindrome(Node head) {
    if (head == null || head.next == null)
        return true;

    Node slow = head, fast = head;
    while (fast.next != null && fast.next.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

    Node head2 = reverseList(slow.next);
    slow.next = null;
    boolean ret = isIdentical(head, head2);
    head2 = reverseList(head2);
    slow.next = head2;

    return ret;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter the number of elements in the linked list: ");
        int n = scanner.nextInt();

        if (n <= 0) {
            System.out.println("Linked list cannot be empty");
            return;
        }

        System.out.print("Enter the elements of the linked list: ");
        Node head = new Node(scanner.nextInt());
        Node current = head;

        for (int i = 1; i < n; i++) {
            current.next = new Node(scanner.nextInt());
            current = current.next;
        }

        boolean result = isPalindrome(head);

        if (result)
            System.out.println("true");
        else
            System.out.println("false");
            scanner.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac Linkedlist.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java Linkedlist
Enter the number of elements in the linked list: 5
Enter the elements of the linked list: 1 2 3 2 1
true

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time complexity: O(n)

5.Balanced tree check

Code:

```
import java.util.Scanner;


class Node {
    int data;
    Node left, right;
    Node(int d) {
        data = d;
        left = right = null;
    }
}


class BinaryTree {
    Node root;


    boolean isBalanced(Node node) {
        if (node == null)
            return true;
```

```
    int lh = height(node.left);
    int rh = height(node.right);

    if (Math.abs(lh - rh) <= 1 && isBalanced(node.left) && isBalanced(node.right))
        return true;

    return false;
}

int height(Node node) {
    if (node == null)
        return 0;

    return 1 + Math.max(height(node.left), height(node.right));
}

public static void main(String args[]) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of nodes: ");
    int n = scanner.nextInt();

    if (n <= 0) {
        System.out.println("Tree cannot be empty");
        return;
    }

    BinaryTree tree = new BinaryTree();

    System.out.println("Enter node values:");
```

```
    tree.root = new Node(scanner.nextInt());

    for (int i = 1; i < n; i++) {
        insertNode(tree.root, scanner.nextInt());
    }

    if (tree.isBalanced(tree.root))
        System.out.println("Tree is balanced");
    else
        System.out.println("Tree is not balanced");
}

static void insertNode(Node root, int data) {
    Node newNode = new Node(data);
    Node current = root;
    while (true) {
        if (data < current.data) {
            if (current.left == null) {
                current.left = newNode;
                break;
            }
            current = current.left;
        } else {
            if (current.right == null) {
                current.right = newNode;
                break;
            }
            current = current.right;
        }
    }
```

```
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac BinaryTree.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java BinaryTree
Enter the number of nodes: 6
Enter node values:
1 2 3 4 5 8
Tree is not balanced

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time Complexity:O(n^2)

6.Triplet sum in array

Code:

```
import java.util.Scanner;


public class TripletSum {


    static boolean find3Numbers(int[] arr, int sum) {
        int n = arr.length;


        for (int i = 0; i < n - 2; i++) {
            for (int j = i + 1; j < n - 1; j++) {
                for (int k = j + 1; k < n; k++) {
                    if (arr[i] + arr[j] + arr[k] == sum) {
                        System.out.println("Triplet is " + arr[i] + ", " + arr[j] + ", " + arr[k]);
                        return true;
                    }
```

```
            }
        }
    }
    return false;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of elements in the array: ");
    int n = scanner.nextInt();

    int[] arr = new int[n];
    System.out.println("Enter the elements of the array:");
    for (int i = 0; i < n; i++) {
        arr[i] = scanner.nextInt();
    }

    System.out.print("Enter the sum to find the triplet for: ");
    int sum = scanner.nextInt();

    boolean result = find3Numbers(arr, sum);

    if (!result) {
        System.out.println("No triplet found with the given sum.");
    }
}
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>javac TripletSum.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>java TripletSum
Enter the number of elements in the array: 6
Enter the elements of the array:
1 4 45 6 10 8
Enter the sum to find the triplet for: 22
Triplet is 4, 10, 8

C:\Users\gowri\OneDrive\Desktop\Practice\Set 2>
```

Time Complexity:O(n^2)