DATE:12–11 -2024 **PRACTICE SET – 2**

DSA

1.0-1 knapsack problem

```
import java.util.Scanner;

public class Knapsack {
  public static int knapsack(int[] weights, int[] values, int capacity) {
    int n = weights.length;
    int[][] dp = new int[n + 1][capacity + 1];

    for (int i = 1; i <= n; i++) {
      for (int w = 0; w <= capacity; w++) {
        if (weights[i - 1] <= w) {
          dp[i][w] = Math.max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);
        } else {
          dp[i][w] = dp[i - 1][w];
        }
      }
    }
    return dp[n][capacity];
  }

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

    // Get weights as a single line of space-separated integers
    System.out.print("Enter weights (space-separated): ");
    String[] weightsInput = scanner.nextLine().split(" ");
    int[] weights = new int[weightsInput.length];
    for (int i = 0; i < weightsInput.length; i++) {
      weights[i] = Integer.parseInt(weightsInput[i]);
    }

    // Get values as a single line of space-separated integers
    System.out.print("Enter values (space-separated): ");
    String[] valuesInput = scanner.nextLine().split(" ");
    int[] values = new int[valuesInput.length];
    for (int i = 0; i < valuesInput.length; i++) {
      values[i] = Integer.parseInt(valuesInput[i]);
```

```
    }

    // Get capacity as a single integer
    System.out.print("Enter knapsack capacity: ");
    int capacity = scanner.nextInt();

    // Calculate the maximum value for the knapsack
    int maxValue = knapsack(weights, values, capacity);
    System.out.println("Maximum value in Knapsack = " + maxValue);

    scanner.close();
  }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Knapsack.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Knapsack
Enter weights (space-separated): 5 6 7
Enter values (space-separated): 10 20 30
Enter knapsack capacity: 8
Maximum value in Knapsack = 30
```

2.Floor in Sorted Array

```
import java.util.Scanner;

public class Sort {
  public static int findFloor(int[] arr, int target) {
    int left = 0, right = arr.length - 1;
    int floor = -1;

    while (left <= right) {
      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {
        return arr[mid];
      } else if (arr[mid] < target) {
        floor = arr[mid];
        left = mid + 1;
      } else {
        right = mid - 1;
      }
    }
    return floor;
```

```
    }

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

        // Get array elements from user as a single line
        System.out.print("Enter sorted array elements (space-separated): ");
        String[] input = scanner.nextLine().split(" ");
        int[] arr = new int[input.length];
        for (int i = 0; i < input.length; i++) {
            arr[i] = Integer.parseInt(input[i]);
        }

        // Get target value
        System.out.print("Enter target value: ");
        int target = scanner.nextInt();

        // Find and print the floor of the target in the array
        System.out.println("Floor of " + target + " is: " + findFloor(arr, target));

        scanner.close();
    }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Sort.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Sort
Enter sorted array elements (space-separated): 1 2 10 10 3 10
Enter target value: 10
Floor of 10 is: 10
```

3.Check equal arrays

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

public class Equal {
    public static boolean areArraysEqual(int[] arr1, int[] arr2) {
        // Check if both arrays are of the same length and contain the same elements
        return Arrays.equals(arr1, arr2);
    }

    public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);

    // Get the first array from the user
    System.out.print("Enter elements of the first array (space-separated): ");
    String[] input1 = scanner.nextLine().split(" ");
    int[] arr1 = new int[input1.length];
    for (int i = 0; i < input1.length; i++) {
      arr1[i] = Integer.parseInt(input1[i]);
    }

    // Get the second array from the user
    System.out.print("Enter elements of the second array (space-separated): ");
    String[] input2 = scanner.nextLine().split(" ");
    int[] arr2 = new int[input2.length];
    for (int i = 0; i < input2.length; i++) {
      arr2[i] = Integer.parseInt(input2[i]);
    }

    // Check if the arrays are equal
    if (areArraysEqual(arr1, arr2)) {
      System.out.println("The arrays are equal.");
    } else {
      System.out.println("The arrays are not equal.");
    }

    scanner.close();
  }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Equal.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Equal
Enter elements of the first array (space-separated): 1 2 3 4 5 6
Enter elements of the second array (space-separated): 1 2 3 4 5 6
The arrays are equal.

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Equal
Enter elements of the first array (space-separated): 1 2 3 4
Enter elements of the second array (space-separated): 13 5 6 7
The arrays are not equal.
```

4.Palindrome linked list

```
import java.util.Scanner;

class ListNode {
  int val;
  ListNode next;

  ListNode(int val) {
    this.val = val;
    this.next = null;
  }
}

public class Palindrome {
  public static boolean isPalindrome(ListNode head) {
    if (head == null || head.next == null) return true;

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

    ListNode secondHalf = reverseList(slow);

    ListNode firstHalf = head;
    while (secondHalf != null) {
      if (firstHalf.val != secondHalf.val) return false;
      firstHalf = firstHalf.next;
      secondHalf = secondHalf.next;
    }

    return true;
  }

  private static ListNode reverseList(ListNode head) {
    ListNode prev = null;
    while (head != null) {
      ListNode next = head.next;
      head.next = prev;
      prev = head;
```

```
        head = next;
    }
    return prev;
}

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();

    ListNode head = null;
    ListNode tail = null;

    System.out.println("Enter the elements of the linked list:");
    for (int i = 0; i < n; i++) {
        int val = scanner.nextInt();
        ListNode newNode = new ListNode(val);
        if (head == null) {
            head = newNode;
            tail = head;
        } else {
            tail.next = newNode;
            tail = tail.next;
        }
    }

    if (isPalindrome(head)) {
        System.out.println("The linked list is a palindrome.");
    } else {
        System.out.println("The linked list is not a palindrome.");
    }

    scanner.close();
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Palindrome.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Palindrome
Enter the number of elements in the linked list: 4
Enter the elements of the linked list:
1
2
2
1
The linked list is a palindrome.

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Palindrome
Enter the number of elements in the linked list: 6
Enter the elements of the linked list:
1
2
3
4
5
6
The linked list is not a palindrome.
```

}

5.Balanced tree check

```
import java.util.Scanner;

class TreeNode {
  int val;
  TreeNode left, right;

  TreeNode(int val) {
    this.val = val;
    this.left = this.right = null;
  }
}

public class Tree {
  public static boolean isBalanced(TreeNode root) {
    return checkHeight(root) != -1;
  }

  private static int checkHeight(TreeNode node) {
    if (node == null) return 0;
```

```
        int leftHeight = checkHeight(node.left);
        if (leftHeight == -1) return -1;

        int rightHeight = checkHeight(node.right);
        if (rightHeight == -1) return -1;

        if (Math.abs(leftHeight - rightHeight) > 1) return -1;

        return Math.max(leftHeight, rightHeight) + 1;
    }

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

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

        TreeNode root = null;

        System.out.println("Enter the tree nodes (value -1 for null nodes, enter in level-order):
");
        if (n > 0) {
            root = new TreeNode(scanner.nextInt());
        }

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

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

        scanner.close();
    }

    private static void addNode(TreeNode root, int val) {
        TreeNode newNode = new TreeNode(val);
        TreeNode current = root;
        while (current != null) {
```

```
        if (val < current.val) {
            if (current.left == null) {
                current.left = newNode;
                return;
            } else {
                current = current.left;
            }
        } else {
            if (current.right == null) {
                current.right = newNode;
                return;
            } else {
                current = current.right;
            }
        }
    }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac Tree.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java Tree
Enter the number of nodes in the tree: 7
Enter the tree nodes (value -1 for null nodes, enter in level-order): 1 2 3 4 5 6 7
The tree is balanced.
```

6.Triplet sum in array

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

public class TripletSum{

    public static boolean findTriplet(int[] arr, int target) {
        Arrays.sort(arr);  // Sort the array to use two-pointer technique

        for (int i = 0; i < arr.length - 2; i++) {
            int left = i + 1;
            int right = arr.length - 1;

            while (left < right) {
                int currentSum = arr[i] + arr[left] + arr[right];
                if (currentSum == target) {
                    System.out.println("Triplet found: " + arr[i] + ", " + arr[left] + ", " + arr[right]);
                    return true;
```

```
            } else if (currentSum < target) {
                left++;
            } else {
                right--;
            }
        }
    }
    return false;
  }

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

    // Get array from user as a single line of space-separated integers
    System.out.print("Enter array elements (space-separated): ");
    String[] input = scanner.nextLine().split(" ");
    int[] arr = new int[input.length];
    for (int i = 0; i < input.length; i++) {
      arr[i] = Integer.parseInt(input[i]);
    }

    // Get the target sum from user
    System.out.print("Enter target sum: ");
    int target = scanner.nextInt();

    // Check if a triplet exists
    if (!findTriplet(arr, target)) {
      System.out.println("No triplet found with the given sum.");
    }

    scanner.close();
  }
}
```

```
C:\Users\SUNITHARAJ\Downloads\new\cdc>javac TripletSum.java

C:\Users\SUNITHARAJ\Downloads\new\cdc>java TripletSum
Enter array elements (space-separated): 1 4 6 8 10 12
Enter target sum: 18
Triplet found: 4, 6, 8

C:\Users\SUNITHARAJ\Downloads\new\cdc>java TripletSum
Enter array elements (space-separated): 2 4 6 8 10
Enter target sum: 100
No triplet found with the given sum.
```