Maha Gowri S **13/11/2024**

1.K-th smallest element

Code:

```
import java.util.PriorityQueue;
import java.util.Scanner;

public class KthSmallest {
    public static int kthSmallest(int[] arr, int k) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>((a, b) -> b - a);
        for (int i : arr) {
            maxHeap.add(i);
            if (maxHeap.size() > k) {
                maxHeap.poll();
            }
        }
        return maxHeap.peek();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter the value of k:");
```

```
        int k = sc.nextInt();
        System.out.println("K-th smallest element is " + kthSmallest(arr, k));
        sc.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>javac KthSmallest.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java KthSmallest
Enter the number of elements:
8
Enter the elements:
34 56 76 63 29 44 111 99
Enter the value of k:
5
K-th smallest element is 63

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>
```

Time Complexity:O(nlogk)

2.Minimize the heights-II

Code:

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


public class MinimizeHeights {
    public static int getMinDiff(int[] arr, int k) {
        Arrays.sort(arr);
        int n = arr.length;
        int result = arr[n - 1] - arr[0];


        int smallest = arr[0] + k;
        int largest = arr[n - 1] - k;
```

```
        for (int i = 0; i < n - 1; i++) {
            int min = Math.min(smallest, arr[i + 1] - k);
            int max = Math.max(largest, arr[i] + k);
            result = Math.min(result, max - min);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter the value of k:");
        int k = sc.nextInt();
        System.out.println("Minimum difference is " + getMinDiff(arr, k));
        sc.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java MinimizeHeights
Enter the number of elements:
8
Enter the elements:
23 44 67 89 34 12 97 58
5
Minimum difference is 75

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java MinimizeHeights
Enter the number of elements:
6
Enter the elements:
12 6 4 15 17 10
Enter the value of k:
6
Minimum difference is 8

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>
```

Time Complexity: O(n logn)

3.Parenthesis checker

Code:

```
import java.util.Scanner;
import java.util.Stack;


public class ParenthesisChecker {
    public static boolean isBalanced(String expr) {
        Stack<Character> stack = new Stack<>();
        for (char ch : expr.toCharArray()) {
            if (ch == '(' || ch == '{' || ch == '[') {
                stack.push(ch);
            } else {
                if (stack.isEmpty()) return false;
                char last = stack.pop();
                if ((ch == ')' && last != '(') || (ch == '}' && last != '{') || (ch == ']' && last != '[')) {
```

```
                return false;
            }
        }
    }
    return stack.isEmpty();
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the expression:");
    String expr = sc.nextLine();
    System.out.println("Is the expression balanced? " + isBalanced(expr));
    sc.close();
}
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>javac ParenthesisChecker.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java ParenthesisChecker
Enter the expression:
((()){{{}}}[][]
Is the expression balanced? true

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java ParenthesisChecker
Enter the expression:
(((()){}{}}
Is the expression balanced? false

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>
```

Time Complexity: O(n)

4.Equilibrium point

Code:

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

```
public class EquilibriumPoint {
    public static int findEquilibriumPoint(int[] arr) {
        int totalSum = 0, leftSum = 0;
        for (int num : arr) totalSum += num;

        for (int i = 0; i < arr.length; i++) {
            totalSum -= arr[i];
            if (leftSum == totalSum) return i;
            leftSum += arr[i];
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        int result = findEquilibriumPoint(arr);
        if (result != -1) {
            System.out.println("Equilibrium point is at index " + result);
        } else {
            System.out.println("No equilibrium point found.");
        }
```

```
        sc.close();
    }
}
```

Output:

```
Enter the number of elements:
7
Enter the elements:
34 46 28 67 31 17 91
No equilibrium point found.

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java EquilibriumPoint
Enter the number of elements:
5
Enter the elements:
1 3 5 2 2
Equilibrium point is at index 2
```

Time Complexity: O(n)

5.Binary Search

Code:

```
import java.util.Scanner;


public class BinarySearch {
    public static int binarySearch(int[] arr, int key) {
        int left = 0, right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == key) return mid;
            if (arr[mid] < key) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the sorted elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.println("Enter the key to search:");
        int key = sc.nextInt();
        int result = binarySearch(arr, key);
        System.out.println(result != -1 ? "Element found at index " + result : "Element not found");
        sc.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>javac BinarySearch.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java BinarySearch
Enter the number of elements:
9
Enter the sorted elements:
3 4 5 6 7 45 56 87 109
Enter the key to search:
87
Element found at index 7

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>
```

Time Complexity:O(log n)

6.Next greater element

Code:

```
import java.util.Scanner;

public class NextGreaterElement {
    static void printNGE(int arr[], int n) {
        int next;
        for (int i = 0; i < n; i++) {
            next = -1;
            for (int j = i + 1; j < n; j++) {
                if (arr[i] < arr[j]) {
                    next = arr[j];
                    break;
                }
            }
            System.out.println(arr[i] + " -- " + next);
        }
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        printNGE(arr, n);
```

```
        sc.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>javac NextGreaterElement.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java NextGreaterElement
Enter the number of elements:
5
Enter the elements:
13 45 32 21 7
13 -- 45
45 -- -1
32 -- -1
21 -- -1
7 -- -1

C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java NextGreaterElement
Enter the number of elements:
4
Enter the elements:
4 5 2 25
4 -- 5
5 -- 25
2 -- 25
25 -- -1
```

Time Complexity:O(n^2)

7.Union of two arrays with duplicate elements

Code:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class UnionOfArrays {
    public static ArrayList<Integer> findUnion(int[] arr1, int[] arr2) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : arr1) map.put(num, map.getOrDefault(num, 0) + 1);
```

```
    for (int num : arr2) map.put(num, map.getOrDefault(num, 0) + 1);

    return new ArrayList<>(map.keySet());
}

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

    System.out.println("Enter the number of elements in the first array:");
    int n1 = sc.nextInt();
    int[] arr1 = new int[n1];
    System.out.println("Enter the elements of the first array:");
    for (int i = 0; i < n1; i++) {
        arr1[i] = sc.nextInt();
    }

    System.out.println("Enter the number of elements in the second array:");
    int n2 = sc.nextInt();
    int[] arr2 = new int[n2];
    System.out.println("Enter the elements of the second array:");
    for (int i = 0; i < n2; i++) {
        arr2[i] = sc.nextInt();
    }

    ArrayList<Integer> union = findUnion(arr1, arr2);
    System.out.println("Union of the two arrays:");
    for (int num : union) {
        System.out.print(num + " ");
    }
```

```
        sc.close();
    }
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java UnionOfArrays
Enter the number of elements in the first array:
4
Enter the elements of the first array:
4 5 2 25
Enter the number of elements in the second array:
6
Enter the elements of the second array:
13 24 35 46 57 68
Union of the two arrays:
2 35 4 68 5 24 25 57 13 46
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>java UnionOfArrays
Enter the number of elements in the first array:
4
Enter the elements of the first array:
1 2 34 56
Enter the number of elements in the second array:
4
Enter the elements of the second array:
2 56 3 44
Union of the two arrays:
1 2 34 3 56 44
C:\Users\gowri\OneDrive\Desktop\Practice\Set 4>
```

Time Complexity:O(n+m)