Problem Solving 13/11/2024

1. Kth smallest Element
   Time complexity: O(n log k)
   Space complexity: O(k)

```java
package program13thNov;
import java.util.*;
public class KthSmallestElement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size of array");
        int n = scanner.nextInt();
        System.out.println("Enter the values");
        int[] arr = new int[n];
        for (int i =0 ; i < n ; i++) {
            arr[i] = scanner.nextInt();
        }
        System.out.println("Enter the kth value");
        int k = scanner.nextInt();
        PriorityQueue<Integer> priorityQueue = new PriorityQueue<Integer>((a,b)->b-a);
        for ( int i = 0 ; i < n ; i++) {
            priorityQueue.offer(arr[i]);
            if (priorityQueue.size()>k) {
                priorityQueue.poll();
            }
        }
        System.out.println(priorityQueue.peek());
        scanner.close();
    }
}

```

```
<terminated> KthSmallestElement [Java Application] C:\
Enter the size of array
6
Enter the values
7 10 4 3 15 20
Enter the kth value
3
7
```

2. Minimize the Height II

   Time complexity: O(n log n)

   Space complexity: O(1)

```java
package program13thNov;
import java.util.*;
public class MinimiseTheHeights2 {
    static int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        Arrays.sort(arr);
        int res = arr[n - 1] - arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] - k < 0)
                continue;
            int minH = Math.min(arr[0] + k, arr[i] - k);
            int maxH = Math.max(arr[i - 1] + k, arr[n - 1] - k);
            res = Math.min(res, maxH - minH);
        }
        return res;
    }
    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 value of k: ");
        int k = scanner.nextInt();
        int result = getMinDiff(arr, k);
        System.out.println(result);

        scanner.close();
    }
}

```

```
<terminated> MinimiseTheHeights2 [Java Application] C:\Program Files\Java\bin\javaw.ex
Enter the number of elements in the array: 6
Enter the elements of the array:
12 6 4 15 17 10
Enter the value of k: 6
8
```

3. Parenthesis Checker

   Time complexity: O(n)

   Space complexity: O(n)

```java
package program13thNov;
import java.util.*;
public class ParanthesisChecker {
    public static boolean isBalanced(String s) {
        Stack<Character> stk = new Stack<>();
        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (ch == '(' || ch == '{' || ch == '[') {
                stk.push(ch);
            } else {
                if (!stk.empty() &&
                    ((stk.peek() == '(' && ch == ')') ||
                     (stk.peek() == '{' && ch == '}') ||
                     (stk.peek() == '[' && ch == ']'))) {
                    stk.pop();
                } else {
                    return false;
                }
            }
        }
        return stk.empty();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string of brackets to check: ");
        String input = scanner.nextLine();
        if (isBalanced(input)) {
            System.out.println("Balanced");
        } else {
            System.out.println("Not Balanced");
        }
        scanner.close();
    }
}

```

```
<terminated> ParanthesisChecker [Java Application] C:\Program Files\Java\bin\javaw.
Enter a string of brackets to check: ({[[{()}}]]
Not Balanced
```

4. Equilibrium Point

   Time complexity: O(n)

   Space complexity: O(1)

```java
package program13thNov;
import java.util.Scanner;
public class EqulibriumPoint {
    public static int findEquilibriumPoint(long[] arr) {
        int n = arr.length;
        int left = 0, pivot = 0, right = 0;
        for (int i = 1; i < n; i++) {
            right += arr[i];
        }
        while (pivot < n - 1 && right != left) {
            pivot++;
            right -= arr[pivot];
            left += arr[pivot - 1];
        }
        return (left == right) ? pivot + 1 : -1;
    }
    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();
        long[] arr = new long[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextLong();
        }
        int result = findEquilibriumPoint(arr);
        System.out.println("Equilibrium index: " + result);
        scanner.close();
    }
}
```

```
<terminated> EqulibriumPoint [Java Application] C:\Program Files\Java\bin\javaw.exe (1
Enter the number of elements in the array: 7
Enter the elements of the array:
-7 1 5 2 -4 3 0
Equilibrium index: 4
```

5. Binary Search

   Time complexity: O(log n)

   Space complexity: O(1)

```java
package program13thNov;
import java.util.Scanner;
public class BinarySearch {
    int binarySearch(int arr[], int x) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == x)
                return mid;
            if (arr[mid] < x)
                low = mid + 1;
            else
                high = mid - 1;
        }
        return -1;
    }
    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 sorted array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();       }
        System.out.print("Enter the element to search for: ");
        int x = scanner.nextInt();
        BinarySearch ob = new BinarySearch();
        int result = ob.binarySearch(arr, x);
        if (result == -1)
            System.out.println("Element is not present in array");
        else
            System.out.println("Element is present at index " + result);
        scanner.close();
    }
}
```

```
<terminated> BinarySearch [Java Application] C:\Program Files\Java\bin\javaw.exe
Enter the number of elements in the array: 10
Enter the elements of the sorted array:
1 2 3 4 5 6 7 8 9 10
Enter the element to search for: 2
Element is present at index 1
```

6. Next Greater Element

   Time complexity: O(n)

   Space complexity: O(n)

```java
package program13thNov;

import java.util.Scanner;

public class NextGreaterElement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size of 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();
        }
        printNGE(arr, n);
        scanner.close();
    }
    static class Stack {
        int top;
        int[] items = new int[100];
        void push(int x) {
            if (top == 99) {
                System.out.println("Stack full");
            } else {
                items[++top] = x;
            }
        }
        int pop() {
            if (top == -1) {
                System.out.println("Underflow error");
                return -1;
            } else {
                return items[top--];
            }
        }
        boolean isEmpty() {
            return top == -1;
        }
    }
    static void printNGE(int[] arr, int n) {
        Stack s = new Stack();
        s.top = -1;
        s.push(arr[0]);
        for (int i = 1; i < n; i++) {
            int next = arr[i];
            if (!s.isEmpty()) {
                int element = s.pop();
                while (element < next) {
                    System.out.println(next);
                    if (s.isEmpty()) break;
                    element = s.pop();
                }
                if (element > next) s.push(element);
            }
            s.push(next);
        }
        while (!s.isEmpty()) {
            int element = s.pop();
            System.out.println(-1);
        }
    }
}
```

```
<terminated> NextGreaterElement [Java Application] C:\Program Files\Ja
Enter the size of the array:
4
Enter the elements of the array:
4 5 2 25
5
25
25
-1
```

7. Union of Two Array

   Time complexity: O(n + m)

   Space complexity: O(n + m)

```java
package program13thNov;
import java.util.*;
public class UnionOf2Array {
    public static int findUnion(int[] a, int[] b) {
        HashSet<Integer> unionSet = new HashSet<>();
        for (int num : a) {
            unionSet.add(num);
        }
        for (int num : b) {
            unionSet.add(num);
        }
        return unionSet.size();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the first array: ");
        int n1 = scanner.nextInt();
        int[] a = new int[n1];
        System.out.println("Enter the elements of the first array:");
        for (int i = 0; i < n1; i++) {
            a[i] = scanner.nextInt();
        }
        System.out.print("Enter the number of elements in the second array: ");
        int n2 = scanner.nextInt();
        int[] b = new int[n2];
        System.out.println("Enter the elements of the second array:");
        for (int i = 0; i < n2; i++) {
            b[i] = scanner.nextInt();
        }
        int result = findUnion(a, b);
        System.out.println("The number of unique elements in the union is: " + result);
        scanner.close();
    }
}
```

```
<terminated> UnionOf2Array [Java Application] C:\Program Files\Java\bin\javaw.exe (13-
Enter the number of elements in the first array: 5
Enter the elements of the first array:
1 5 3 4 2
Enter the number of elements in the second array: 7
Enter the elements of the second array:
1 9 4 8 7 5 3
The number of unique elements in the union is: 8
```