Kth SMALLEST

CODE

```
import java.util.*;

public class KthSmallestElement {

    public static int kthSmallest(int[] arr, int k) {

        for (int i = 0; i < k; i++) {

            int minIndex = i;

            for (int j = i + 1; j < arr.length; j++) {

                if (arr[j] < arr[minIndex]) {

                    minIndex = j;

                }

            }

            int temp = arr[i];

            arr[i] = arr[minIndex];

            arr[minIndex] = temp;

        }

        return arr[k - 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 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();
        if (k > 0 && k <= n) {
            System.out.println("The " + k + "th smallest element is: " + kthSmallest(arr, k));
        } else {
            System.out.println("Invalid input for k.");
        }
    }
}
```

OUTPUT

```
G:\java programs\hw>java KthSmallestElement
Enter the size of the array: 6
Enter the elements of the array:
7 10 4 3 20 15
Enter the value of k: 3
The 3th smallest element is: 7
```

TIME COMPLEXITY

O(n)

MINIMIZE THE HEIGHT(II)

CODE

```
import java.util.*;

class MinHeight {

    public static int minimizeHeightDifference(int[] arr, int k) {

        int n = arr.length;

        if (n == 1) {

            return 0;

        }

        Arrays.sort(arr);

        int maxDifference = arr[n - 1] - arr[0];

        int smallest = arr[0] + k;

        int largest = arr[n - 1] - k;

        if (smallest > largest) {

            int temp = smallest;

            smallest = largest;

            largest = temp;

        }

        for (int i = 1; i < n - 1; i++) {

            int addK = arr[i] + k;

            int subtractK = arr[i] - k;

            if (subtractK >= smallest || addK <= largest) {

                continue;

            }

            if (largest - subtractK <= addK - smallest) {
                smallest = subtractK;
            } else {
                largest = addK;
            }
        }
        return Math.min(maxDifference, largest - smallest);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of towers: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter the heights of the towers:");
        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 = minimizeHeightDifference(arr, k);
        System.out.println("The minimum possible difference after modifications is: " + result);
    }
}
```

OUTPUT

```
G:\java programs\hw>java MinHeight
Enter the number of towers: 5
Enter the heights of the towers:

3 9 12 16 20

Enter the value of K: 3

The minimum possible difference after modifications is: 11
```

TIME COMPLEXITY

O(n)

UNION OF TWO ARRAY

CODE

```
import java.util.*;

public class UnionCount {

    public static int countUnion(int[] a, int[] b) {

        HashSet<Integer> set = new HashSet<>();

        for (int num : a) {

            set.add(num);

        }

        for (int num : b) {

            set.add(num);

        }

        return set.size();

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements in array a: ");

        int n = scanner.nextInt();

        int[] a = new int[n];

        System.out.println("Enter elements of array a:");

        for (int i = 0; i < n; i++) {

            a[i] = scanner.nextInt();

        }

        System.out.print("Enter the number of elements in array b: ");

        int m = scanner.nextInt();
        int[] b = new int[m];
        System.out.println("Enter elements of array b:");
        for (int i = 0; i < m; i++) {
            b[i] = scanner.nextInt();
        }
        System.out.println("The number of distinct elements in the union is: " + countUnion(a, b));
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java UnionCount
Enter the number of elements in array a: 5
Enter elements of array a:
1 2 3 4 5
Enter the number of elements in array b: 3
Enter elements of array b:
1 2 3
The number of distinct elements in the union is: 5
```

TIME COMPLEXITY

O(n)

PARENTHESIS CHECKER

CODE

```
import java.util.*;

public class Parantheses {

    public static boolean isBalanced(String s) {

        Stack<Character> stack = new Stack<>();

        for (char ch : s.toCharArray()) {

            if (ch == '{' || ch == '(' || ch == '[') {

                stack.push(ch);

            } else if (ch == '}' || ch == ')' || ch == ']') {

                if (stack.isEmpty() || !isMatchingBracket(stack.peek(), ch)) {

                    return false;

                }

                stack.pop();

            }

        }

        return stack.isEmpty();

    }

    private static boolean isMatchingBracket(char opening, char closing) {

        return (opening == '(' && closing == ')') ||

               (opening == '{' && closing == '}') ||

               (opening == '[' && closing == ']');

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string containing brackets: ");
        String input = scanner.nextLine();
        if (isBalanced(input)) {
            System.out.println("True");
        } else {
            System.out.println("False");
        }
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java Parantheses
Enter a string containing brackets: }
False

G:\java programs\hw>java Parantheses
Enter a string containing brackets: {
False
```

TIME COMPLEXITY

O(n)

EQUILIBRIUM POINT

CODE

```
import java.util.*;

class EquilibriumPoint {

    static int equilibriumPoint(int arr[]) {

        int n = arr.length;

        int totalSum = 0;

        for (int i = 0; i < n; i++) {

            totalSum += arr[i];

        }

        int leftSum = 0;

        for (int i = 0; i < n; i++) {

            totalSum -= arr[i];

            if (leftSum == totalSum) {

                return i + 1;

            }

            leftSum += arr[i];

        }

        return -1;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");

        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();
        }
        int result = equilibriumPoint(arr);
        if (result == -1) {
            System.out.println("No equilibrium point exists.");
        } else {
            System.out.println("The equilibrium point is at index " + result);
        }
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java EquilibriumPoint
Enter the number of elements: 5
Enter the elements of the array:
1 3 5 2 2
The equilibrium point is at index 3
```

TIME COMPLEXITY

O(n)

BINARY SEARCH

CODE

```
import java.util.Scanner;

class BinarySearchTree {

    public static boolean isValidInorder(int[] arr) {

        for (int i = 1; i < arr.length; i++) {

            if (arr[i] <= arr[i - 1]) {

                return false;

            }

        }

        return true;

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");

        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();

        }

        boolean result = isValidInorder(arr);

        if (result) {

            System.out.println("True");

        } else {
            System.out.println("False");
        }
        scanner.close();
    }
}
```

OUTPUT

```
G:\java programs\hw>java BinarySearchTree
Enter the number of elements: 7
Enter the elements of the array:
35 86 44 78 65 90 2
False
```

TIME COMPLEXITY

O(n)

NEXT GREATER ELEMENT

CODE

```
import java.util.Stack;

import java.util.Scanner;

public class Nextgreater {

    public static int[] nextGreaterElement(int[] arr) {

        int[] result = new int[arr.length];

        for (int i = 0; i < arr.length; i++) {

            result[i] = -1;

        }

        Stack<Integer> stack = new Stack<>();

        for (int i = 0; i < arr.length; i++) {

            while (!stack.isEmpty() && arr[i] > arr[stack.peek()]) {

                int index = stack.pop();

                result[index] = arr[i];

            }

            stack.push(i);

        }

        return result;

    }

    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();
        }
        int[] result = nextGreaterElement(arr);
        System.out.println("Next Greater Elements:");
        for (int num : result) {
            System.out.print(num + " ");
        }
    }
}
```

OUTPUT

```
G:\java programs\hw>java Nextgreater
Enter the size of the array:
4
Enter the elements of the array:
10 20 30 50
Next Greater Elements:
20 30 50 -1
```

TIME COMPLEXITY

O(n)