1.K-th smallest element

Code:

```
import java.util.Random;

public class KthSmallestElement {
    public static void main(String[] args) {
        int[] arr = {7, 10, 4, 3, 20, 15};
        int k = 3;

        int kthSmallest = findKthSmallest(arr, k);
        System.out.println("K-th smallest element: " + kthSmallest);
    }

    public static int findKthSmallest(int[] arr, int k) {
        return quickSelect(arr, 0, arr.length - 1, k - 1);
    }

    private static int quickSelect(int[] arr, int left, int right, int k) {
        if (left == right) return arr[left];

        Random random = new Random();
        int pivotIndex = left + random.nextInt(right - left + 1);

        pivotIndex = partition(arr, left, right, pivotIndex);

        if (k == pivotIndex) {
            return arr[k];
        } else if (k < pivotIndex) {
            return quickSelect(arr, left, pivotIndex - 1, k);
        } else {
            return quickSelect(arr, pivotIndex + 1, right, k);
        }
    }
```

```
    private static int partition(int[] arr, int left, int right, int pivotIndex) {
        int pivotValue = arr[pivotIndex];
        swap(arr, pivotIndex, right);
        int storeIndex = left;

        for (int i = left; i < right; i++) {
            if (arr[i] < pivotValue) {
                swap(arr, storeIndex, i);
                storeIndex++;
            }
        }

        swap(arr, storeIndex, right);
        return storeIndex;
    }

    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

Output:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS\OneDrive\Desktop\c4>javac  KthSmallestElement.java

C:\Users\ASUS\OneDrive\Desktop\c4>java  KthSmallestElement
K-th smallest element: 7

C:\Users\ASUS\OneDrive\Desktop\c4>
```

Time complexity:O(n)

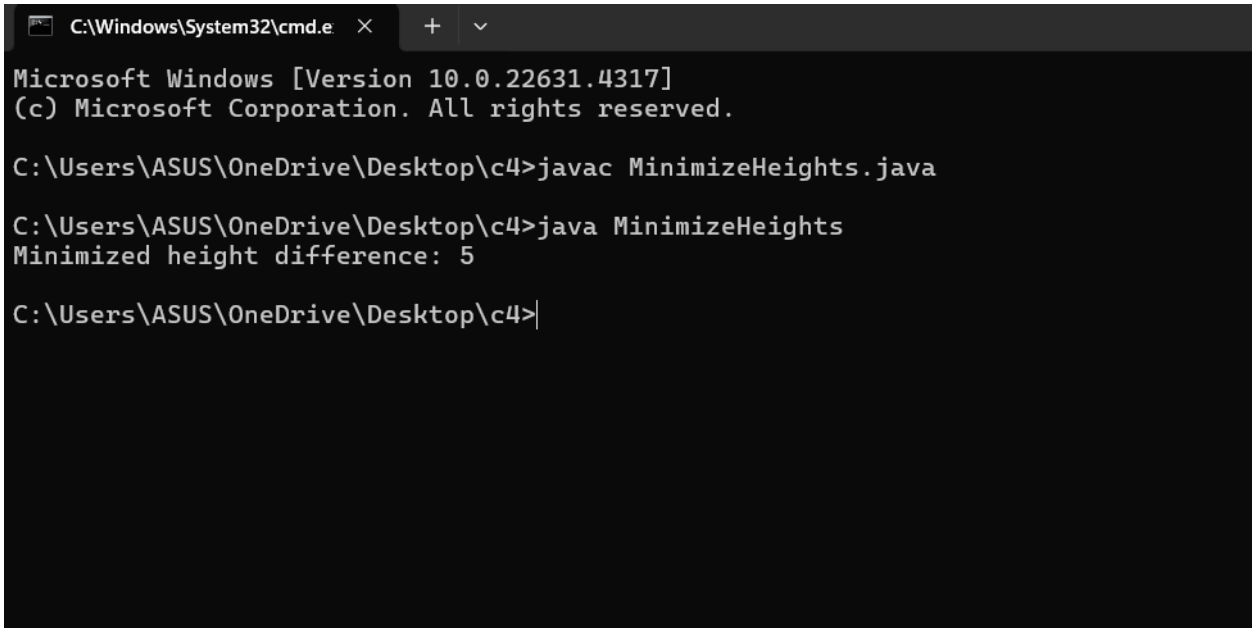
2)Minimize the heights-II

Code:

```
import java.util.Arrays;

public class MinimizeHeights {
    public static void main(String[] args) {
        int[] arr = {1, 5, 8, 10};
        int k = 2;
        System.out.println("Minimized height difference: " +
minimizeHeightDifference(arr, k));
    }

    public static int minimizeHeightDifference(int[] arr, int k) {
        Arrays.sort(arr);
        int n = arr.length;
```

```
        int result = arr[n - 1] - arr[0];
        int small = arr[0] + k;
        int big = arr[n - 1] - k;
        if (small > big) {
            int temp = small;
            small = big;
            big = temp;
        }
        for (int i = 1; i < n - 1; i++) {
            int add = arr[i] + k;
            int sub = arr[i] - k;
            if (sub >= small || add <= big) {
                continue;
            }
            if (big - sub <= add - small) {
                small = sub;
            } else {
                big = add;
            }
        }
        return Math.min(result, big - small);
    }
}
```

Output:



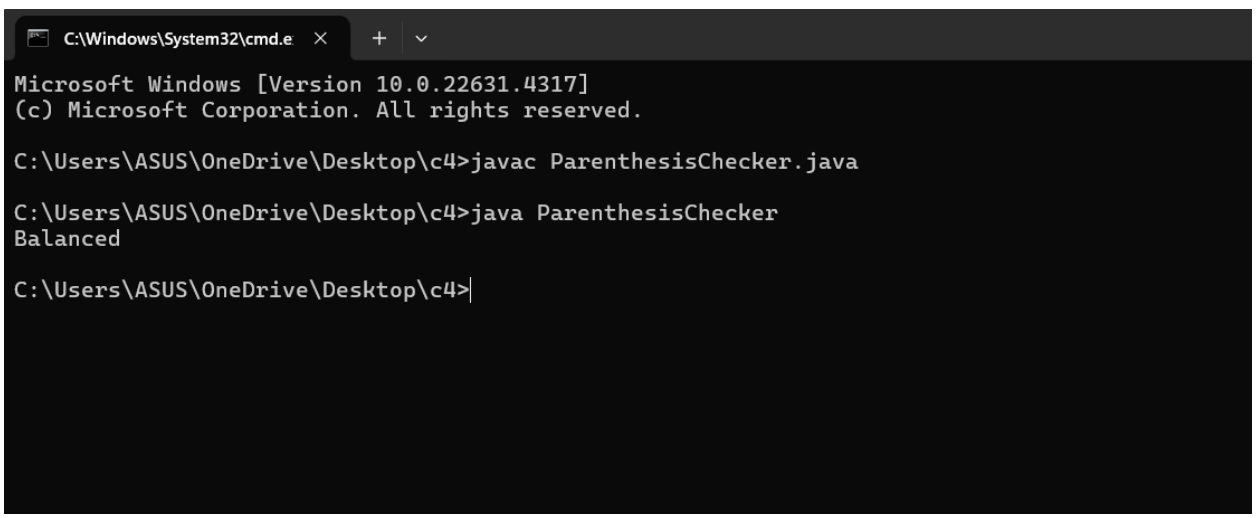
Time complexity:O(n logn)

3).Parenthesis checker

Code;

```
import java.util.Stack;

public class ParenthesisChecker {
    public static void main(String[] args) {
        String expression = "{[]}";
        if (isBalanced(expression)) {
            System.out.println("Balanced");
        } else {
            System.out.println("Not Balanced");
        }
    }

    public static boolean isBalanced(String expression) {
        Stack<Character> stack = new Stack<>();
        for (char ch : expression.toCharArray()) {
            if (ch == '(' || ch == '{' || ch == '[') {
```

```
            stack.push(ch);
        } else if (ch == ')' && !stack.isEmpty() && stack.peek() == '(') {
            stack.pop();
        } else if (ch == '}' && !stack.isEmpty() && stack.peek() == '{') {
            stack.pop();
        } else if (ch == ']' && !stack.isEmpty() && stack.peek() == '[') {
            stack.pop();
        } else {
            return false;
        }
    }
    return stack.isEmpty();
  }
}
```



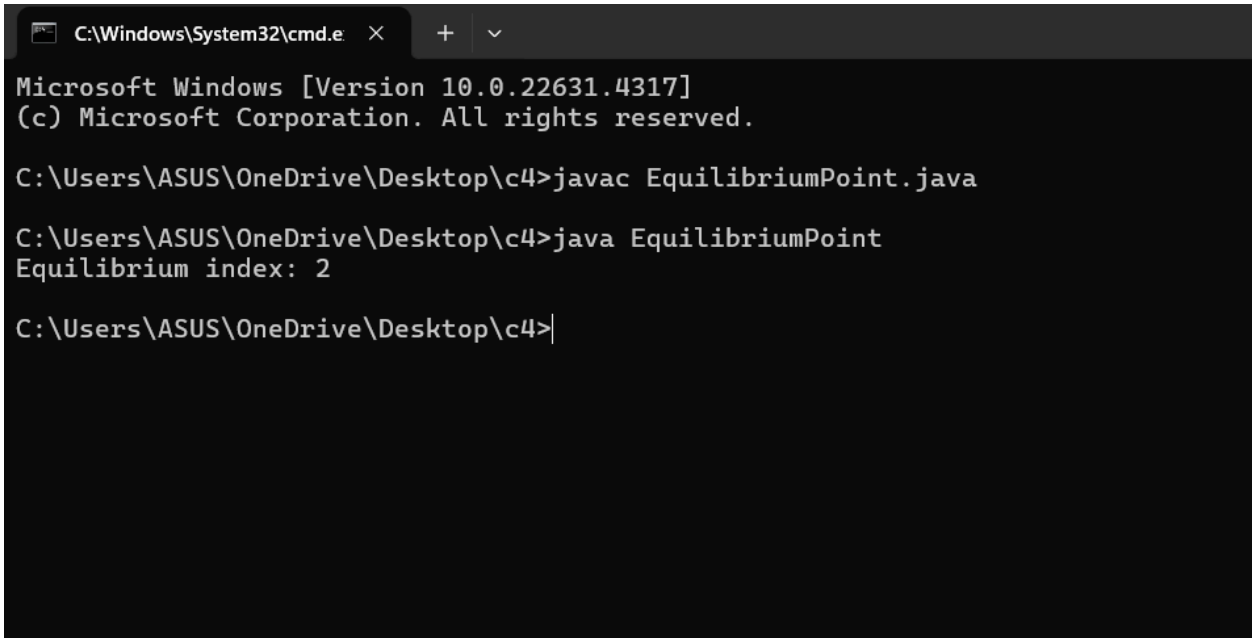
Time complexity:O(n)

4)Equilibrium point

Code:

```
public class EquilibriumPoint {
  public static void main(String[] args) {
    int[] arr = {1, 3, 5, 2, 2};
```

```
        int equilibriumIndex = findEquilibrium(arr);
        System.out.println("Equilibrium index: " + equilibriumIndex);
    }

    public static int findEquilibrium(int[] arr) {
        int totalSum = 0;
        int 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;
    }
}
```



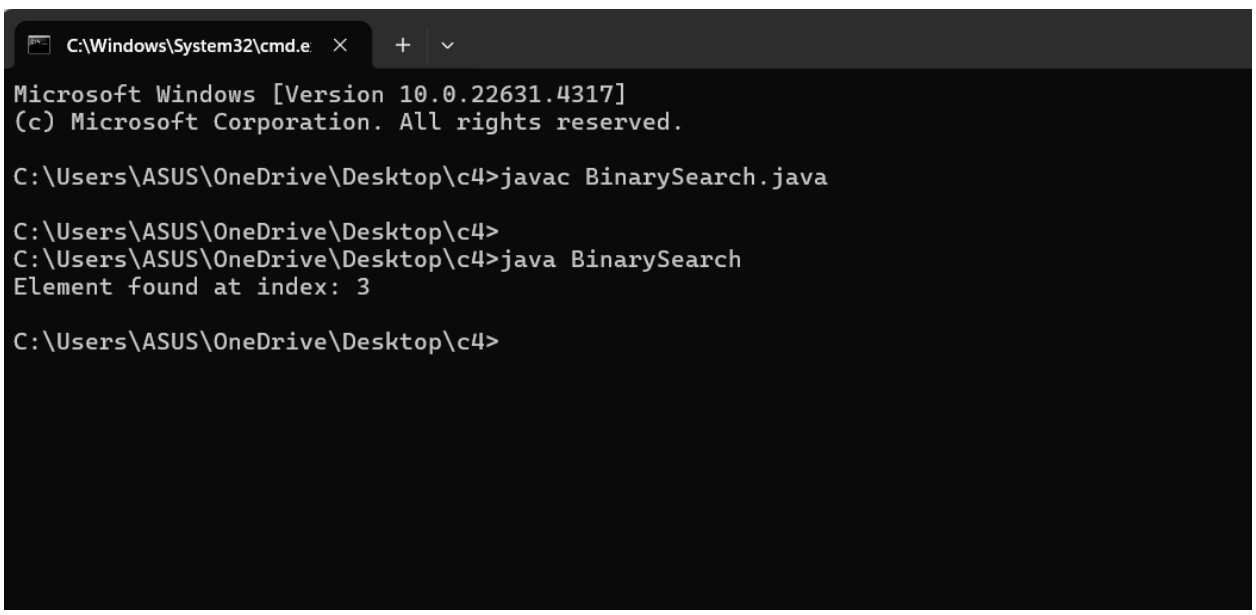
Time complexity:O(n)

5)Binary Search

Code:

```
public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = {1, 3, 5, 7, 9, 11, 13};
        int target = 7;
        int result = binarySearch(arr, target);
        if (result == -1) {
            System.out.println("Element not found");
        } else {
            System.out.println("Element found at index: " + result);
        }
    }

    public static int binarySearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
```

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

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

        return -1;
    }
}
```



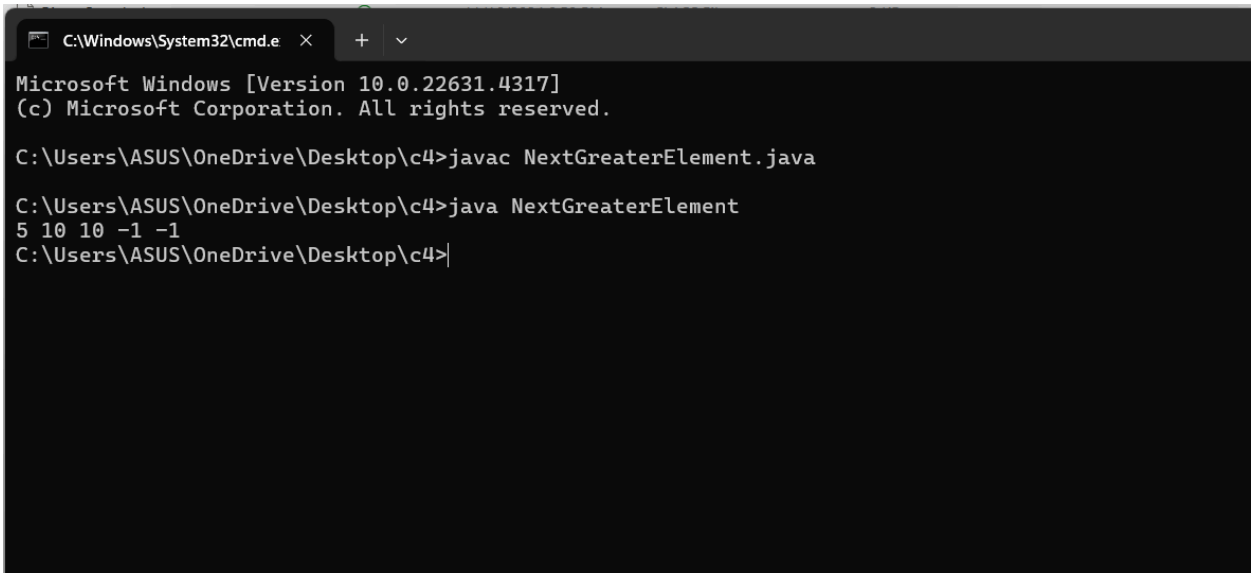
Time complexity:O(logn)

6).Next greater element

Code;

```
import java.util.Stack;

public class NextGreaterElement {
    public static void main(String[] args) {
        int[] arr = {4, 5, 2, 10, 8};
        int[] result = nextGreaterElement(arr);

        for (int i : result) {
            System.out.print(i + " ");
        }
    }

    public static int[] nextGreaterElement(int[] arr) {
        int n = arr.length;
        int[] result = new int[n];
        Stack<Integer> stack = new Stack<>();

        for (int i = n - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }

            if (stack.isEmpty()) {
                result[i] = -1;
            } else {
                result[i] = stack.peek();
            }

            stack.push(arr[i]);
        }

        return result;
    }
```

}



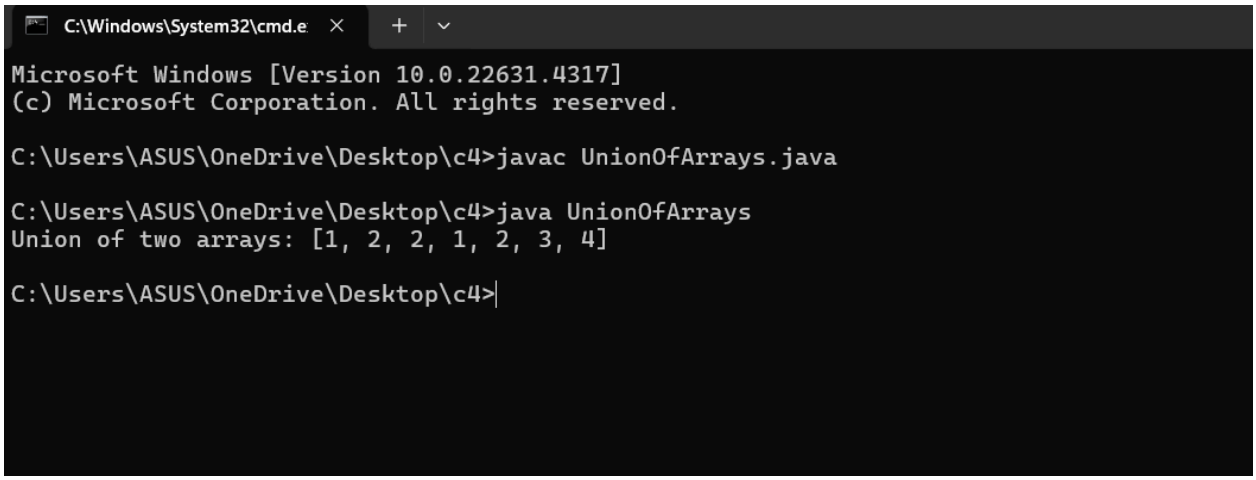
Time complexity:O(n)

7)Union of two arrays with duplicate elements:

Code:

```
import java.util.Arrays;

public class UnionOfArrays {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 2, 1};
        int[] arr2 = {2, 3, 4};
        int[] result = union(arr1, arr2);

        System.out.println("Union of two arrays: " + Arrays.toString(result));
    }

    public static int[] union(int[] arr1, int[] arr2) {
        int n1 = arr1.length;
        int n2 = arr2.length;
```

```
        int[] result = new int[n1 + n2];
        int i = 0;

        for (int num : arr1) {
            result[i++] = num;
        }

        for (int num : arr2) {
            result[i++] = num;
        }

        return result;
    }
}
```



Time complexity:O(n+m)