13/11/2024

DSA-practice-4

1.KthSmallestElement:

Coding:

```
import java.util.PriorityQueue;
import java.util.Scanner;
public class KthSmallestElement {
    public static int findKthSmallest(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int num : arr) minHeap.add(num);
        for (int i = 1; i < k; i++) minHeap.poll();
        return minHeap.poll();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of elements: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];
        System.out.println("Enter elements:");
        for (int i = 0; i < n; i++) arr[i] = scanner.nextInt();
        System.out.print("Enter k: ");
        int k = scanner.nextInt();
        System.out.println("The " + k + "th smallest element is " + findKthSmallest(arr, k));
        scanner.close();
    }
}
```

Solution:

```
C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>javac KthSmallestElement.java

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java KthSmallestElement
Enter number of elements: 7
Enter elements:
7
10
4
3
20
15
1
Enter k: 3
The 3th smallest element is 4
```

2.Minimize the heights-II

Coding:

```
import java.util.Arrays;
public class MinimizeHeightDifference {
    public static int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        if (n == 1) return 0;
        Arrays.sort(arr);
        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 minHeight = Math.min(smallest, arr[i + 1] - k);
            int maxHeight = Math.max(largest, arr[i] + k);
            result = Math.min(result, maxHeight - minHeight);
        }
        return result;
    }
    public static void main(String[] args) {
        int[] arr = {1, 5, 8, 10};
        int k = 2;
        System.out.println("Minimum difference is " + getMinDiff(arr, k));
    }
}
```

Solution:

```
C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>javac MinimizeHeightDifference.java

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java MinimizeHeightDifference
Minimum difference is 5
```

3.Parenthesis checker:

Coding:

```
import java.util.Stack;
public class BalancedBrackets {
    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()) return false;
            char top = stack.pop();
            if ((ch == ')' && top != '(') ||
                (ch == '}' && top != '{') ||
                (ch == ']' && top != '[')) {
                return false;
            }
        }
    }
    return stack.isEmpty();
  }
  public static void main(String[] args) {
    String s = "{([])}";
    System.out.println("Is the expression balanced? " + isBalanced(s));
  }
}
```

Solution:

```
C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>javac BalancedBrackets.java

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java BalancedBrackets
Is the expression balanced? true
```

4.Equilibrium point:

Coding:

```
import java.util.*;
public class EquilibriumPoint {
  public static int findEquilibriumPoint(int[] arr) {
    int totalSum = 0;
    for (int num : arr) totalSum += num;
    int leftSum = 0;
    for (int i = 0; i < arr.length; i++) {
      totalSum -= arr[i];
      if (leftSum == totalSum)
        return i + 1;
      leftSum += arr[i];
    }
    return -1;
  }
  public static void main(String[] args) {
    int[] arr = {1, 3, 5, 2, 2};
    System.out.println("Equilibrium Point: " + findEquilibriumPoint(arr));
  }
}
```

Solution:

```
C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>javac EquilibriumPoint.java

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java EquilibriumPoint
Equilibrium Point: 3
```

5.Binary Search::

```
import java.io.*;

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;
    }
}
```

6.Next greater element::

```
import java.io.*;

class Main{
    static void func(int arr[], int n) {
        int i, j;
        int next = -1;
        for (i = 0; i < n; i++) {
            for (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) {
        int arr[] = { 11, 22, 33, 3 };
        int n = arr.length;
        func(arr, n);
    }
}
```