Name: Jai Darshan S

Dept: AI&DS

Date: 13/11/2024

Practice set 4:

1. Kth Smallest Element

   Java code:

```
public class KthSmallestElement {
  public static int kthsmall(int[] arr,int k){
    int n=arr.length;
    for (int i=0;i<n;i++){
      for (int j=0;j<n-i-1;j++){
        if (arr[j]>arr[j+1]){
          int temp=arr[j];
          arr[j]=arr[j+1];
          arr[j+1]=temp;
        }
      }
    }
    return arr[k-1];
  }
  public static void main(String[] args) {
    int[] arr1={7, 10, 4, 3, 20, 15};
    int k1=3;
    int[] arr2={2, 3, 1, 20, 15};
    int k2=4;
    System.out.println(kthsmall(arr1, k1));
    System.out.println(kthsmall(arr2, k2));
  }
}
```

   Output:

```
ma\AppData\Roaming\Code\User\workspaceStorage\4707c18a080e585dbb0771255f61f210\
redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'KthSmallestElement'
7
15
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

   Time complexity: O(n^2)
   Space complexity: O(1)

2. Minimize the Height II:

   Java code:

```
import java.util.Arrays;

public class MinimizeTheHeight {
  public static int height(int[] arr, int k) {
    int n=arr.length;
    if (n==1){
      return 0;
    }
    Arrays.sort(arr);
    int r=arr[n-1]-arr[0];
    int m=arr[n-1]-k;
    int na=arr[0]+k;
    for (int i=0;i<n-1;i++){
      int nn=Math.min(na,arr[i+1]-k);
      int nm=Math.max(m,arr[i]+k);
      if (nn<0){
        continue;
      }
      r=Math.min(r,nm-nn);
    }
    return r;
  }
  public static void main(String[] args) {
    int[] arr1={1, 5, 8, 10};
    int[] arr2={3, 9, 12, 16, 20};
    int k1=2,k2=3;
    System.out.println(height(arr1, k1));
    System.out.println(height(arr2, k2));
  }
}
```

Output:

```
ma\AppData\Roaming\Code\User\workspaceStorage\4707c18a080e585dbb0771255f61f210\
redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'MinimizeTheHeight'
5
11
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(n log n)
Space complexity: O(1)

3. Parenthesis Checker:

Java code:

```
import java.util.Stack;

public class ParanthesisChecker {
```

```
    public static boolean check(String s){
        if (s.length()%2==1){
            return false;
        }
        Stack<Character> st=new Stack<>();
        for (int j=0;j<s.length();j++){
            char i=s.charAt(j);
            if (i=='{' || i=='(' || i=='['){
                st.push(i);
            }else if(i==')' || i=='}' || i==']'){
            if (st.isEmpty() || (i==')' && st.peek()!='(') || (i=='}' && st.peek()!='{') || (i==']' &&
st.peek()!='[')){
                return false;
            }
            st.pop();
        }
        }
        return st.isEmpty();
    }
    public static void main(String[] args) {
        System.out.println(check("{([])}"));
        System.out.println(check("()"));
        System.out.println(check("([]"));
    }
}
```

Output:

```
User\workspaceStorage\4707c18a080e585dbb0771255f61f210\redhat.java\jdt_ws\Pract
ice 2_68869b4c\bin' 'ParanthesisChecker'
true
true
false
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(n)
Space complexity: O(n)

4. Equilibrium point:

Java code:

```
import java.util.Arrays;

public class EquilibriumPoint {
    public static int result(int arr[]) {
        if (arr.length==1){
            return 1;
        }
```

```
        int t=Arrays.stream(arr).sum();
        int l=0;
        for (int i=0;i<arr.length;i++){
            t-=arr[i];
            if (t==l){
                return i+1;
            }
            l+=arr[i];
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] arr1={1, 3, 5, 2, 2};
        int[] arr2={1};
        int[] arr3={1,2,3};
        System.out.println(result(arr1));
        System.out.println(result(arr2));
        System.out.println(result(arr3));
    }
}
```

Output:

```
redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'EquilibriumPoint'
3
1
-1
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(n)
Space complexity: O(1)

5. Binary Search:

Java code:

```
import java.util.Arrays;

public class BinarySearch {
    public static int search(int[] arr,int k){
        Arrays.sort(arr);
        int n=arr.length;
        int low=0,high=n-1;
        while (low<=high){
            int mid=(low+(high-low)/2);
            if (arr[mid]==k){
                return mid;
            }else if(arr[mid]<k){
                low=mid+1;
```

```
        }else{
          high=mid-1;
        }
      }
      return -1;
    }
    public static void main(String[] args) {
      int[] arr1={1,2,3,4,5,6},arr2={4,7,6,2,3};
      int k1=3,k2=9;
      System.out.println(search(arr1, 3));
      System.out.println(search(arr2, k2));
    }
  }
```

Output:

```
User\workspaceStorage\4707c18a080e585dbb0771255f61f210\redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'BinarySearch'
2
-1
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(log n)
Space complexity: O(1)

6. Next Greater Element:

Java code:

```
import java.util.ArrayList;

public class NextGreaterElement {
  public static ArrayList<Integer> next(int[] arr) {
    ArrayList<Integer> r=new ArrayList<>();
    for (int i=0;i<arr.length-1;i++){
      int m=-1;
      for (int j=i+1;j<arr.length;j++){
        if (arr[j]>arr[i]){
          m=arr[j];
          break;
        }
      }
      r.add(m);
    }
    r.add(-1);
    return r;
  }
  public static void main(String[] args) {
```

```
    int[] arr1={1, 3, 2, 4};
    int[] arr2={6, 8, 0, 1, 3};
    int[] arr3={50, 40, 30, 10};
    int[] arr4={10, 20, 30, 50};
    System.out.println(next(arr1));
    System.out.println(next(arr2));
    System.out.println(next(arr3));
    System.out.println(next(arr4));
  }
}
```

Output:

```
ma\AppData\Roaming\Code\User\workspaceStorage\4707c18a080e585dbb0771255f61f210
redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'NextGreaterElement'
[3, 4, 4, -1]
[8, -1, 1, 3, -1]
[-1, -1, -1, -1]
[20, 30, 50, -1]
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(n^2)
Space complexity: O(n)

7. Union of Two Arrays with Duplicate Elements:

Java code:

```
import java.util.ArrayList;

public class UnionOfTwoArrays {
  public static int union(int[] a,int[] b){
    ArrayList<Integer> arr=new ArrayList<>();
    for (int i=0;i<a.length;i++){
      if (!arr.contains(a[i])){
        arr.add(a[i]);
      }
    }
    for (int j=0;j<b.length;j++){
      if (!arr.contains(b[j])){
        arr.add(b[j]);
      }
    }
    return arr.size();
  }
  public static void main(String[] args) {
    int[] a1={1, 2, 3, 4, 5};
    int[] b1={1, 2, 3};
```

```
        int[] a2={85, 25, 1, 32, 54, 6};
        int[] b2={85, 2};
        int[] a3={1, 2, 1, 1, 2};
        int[] b3={2, 2, 1, 2, 1};
        System.out.println(union(a1, b1));
        System.out.println(union(a2, b2));
        System.out.println(union(a3, b3));
    }
}
```

Output:

```
ma\AppData\Roaming\Code\User\workspaceStorage\4707c18a080e585dbb0771255f61f210\
redhat.java\jdt_ws\Practice 2_68869b4c\bin' 'UnionOfTwoArrays'
5
7
2
PS C:\Users\nirma\OneDrive\Documents\Practice 2>
```

Time complexity: O(nxm)
Space complexity: O(n+m)