

Ranjith R 22IT085 Day 4 DSA Practice

1.Kth smallest element

PROGRAM:

```
package JavaPractice;
```

```
import java.util.PriorityQueue;
```

```
import java.util.Scanner;
```

```
public class Array {
```

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

```
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
```

```
        for (int num : arr) {
```

```
            minHeap.add(num);
```

```
        }
```

```
        for (int i = 0; i < k - 1; i++) {
```

```
            minHeap.poll();
```

```
        }
```

```
        return minHeap.peek();
```

```
    }
```

```

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 = kthSmallest(arr, k);

    System.out.println("The " + k + "-th smallest element is: " +
result);
    }
}

```

Time Complexity: $O(n \log n)$

OUTPUT:

The screenshot shows an IDE with a package explorer on the left containing 'ArrayList1', 'BookingSystem2', 'JavaPractice', 'Main', and 'TicketBooking'. The main editor displays the following Java code:

```
1 package JavaPractice;
2
3 import java.util.PriorityQueue;
4 import java.util.Scanner;
5
6 public class Array {
7
8     public static int kthSmallest(int[] arr, int k) {
9         PriorityQueue<Integer> minHeap = new PriorityQueue<>();
10
11         for (int num : arr) {
12             minHeap.add(num);
13         }
14         for (int i = 0; i < k - 1; i++) {
15             minHeap.poll();
16         }
17
18         return minHeap.peek();
19     }
20
21     public static void main(String[] args) {
22         Scanner scanner = new Scanner(System.in);
23
24         System.out.print("Enter the number of elements in the array: ");
25         int n = scanner.nextInt();
26         int[] arr = new int[n];
27     }
```

The console output at the bottom shows the program's execution:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 11:54:00 pm - 11:54:34 pm) [pid: 4908]
Enter the number of elements in the array: 6
Enter the elements of the array:
7 10 4 3 20 15
Enter the value of k: 3
The 3-th smallest element is: 7
```

2. Minimise the Heights II

```
package JavaPractice;
```

```
import java.util.Arrays;
```

```
import java.util.Scanner;
```

```
public class Array {
```

```
    public int getMinDiff(int[] arr, int n, int k) {
```

```
        Arrays.sort(arr);
```

```
        int result = arr[n - 1] - arr[0];
```

```
        int minHeight = arr[0] + k;
```

```
        int maxHeight = arr[n - 1] - k;
```

```

    for (int i = 1; i < n; i++) {
        if (arr[i] < k) continue;

        int minElement = Math.min(minHeight, arr[i] - k);
        int maxElement = Math.max(maxHeight, arr[i - 1] + k);
        result = Math.min(result, maxElement - minElement);
    }

    return result;
}

```

```

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();

    Array sol = new Array();
    int result = sol.getMinDiff(arr, n, k);
}

```

```

        System.out.println("The minimum possible difference is: " +
result);

        scanner.close();

    }

}

```

Time Complexity: $O(n \log n)$

OUTPUT:

The screenshot shows an IDE with a Java project. The code in `Array.java` is as follows:

```

19
20     return result;
21 }
22
23 public static void main(String[] args) {
24     Scanner scanner = new Scanner(System.in);
25     System.out.print("Enter the number of towers: ");
26     int n = scanner.nextInt();
27
28     int[] arr = new int[n];
29     System.out.println("Enter the heights of the towers:");
30     for (int i = 0; i < n; i++) {
31         arr[i] = scanner.nextInt();
32     }
33
34     System.out.print("Enter the value of K: ");
35     int k = scanner.nextInt();
36
37     Array sol = new Array();
38     int result = sol.getMinDiff(arr, n, k);
39
40     System.out.println("The minimum possible difference is: " + result);
41     scanner.close();
42 }
43 }
44

```

The console output shows the program's execution:

```

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 11:57:11 pm - 11:57:40 pm) [pid: 12976]
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 is: 11

```

3. Parentheses Checker

```
package JavaPractice;
```

```
import java.util.Stack;
```

```
import java.util.Scanner;
```

```
public class Array {
```

```
    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) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the expression: ");

        String s = scanner.nextLine();

        System.out.println(isBalanced(s));

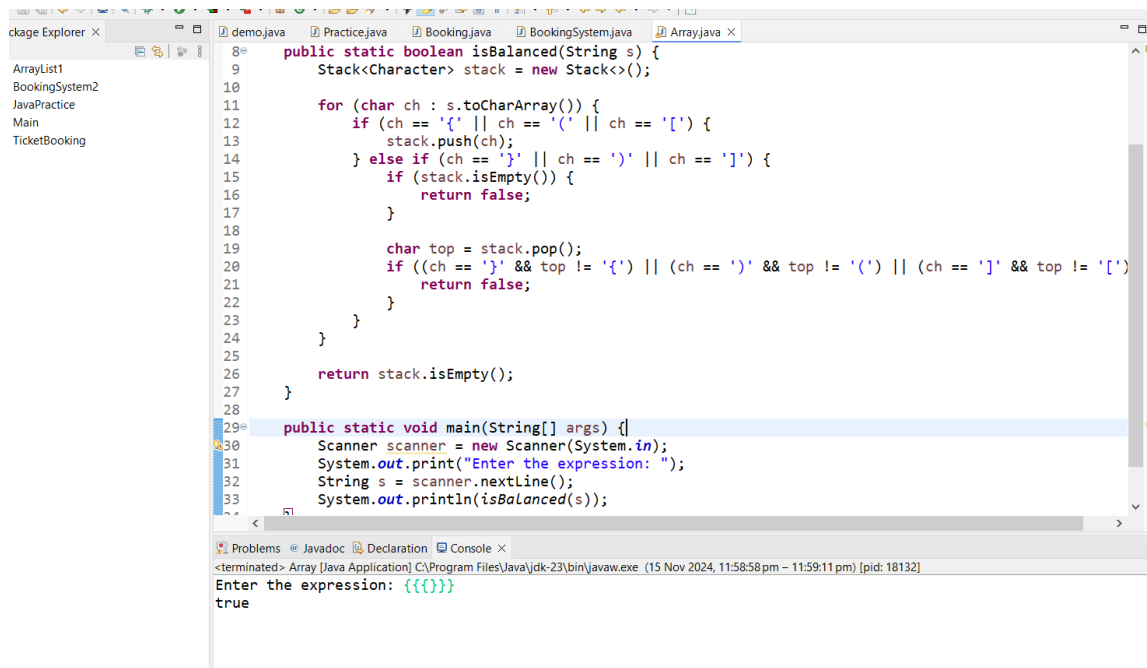
    }

}

```

Time Complexity: $O(n)$

OUTPUT:



The screenshot shows an IDE with a project named 'TicketBooking'. The 'Array.java' file is open, displaying the following code:

```

8= public static boolean isBalanced(String s) {
9     Stack<Character> stack = new Stack<>();
10
11     for (char ch : s.toCharArray()) {
12         if (ch == '{' || ch == '(' || ch == '[') {
13             stack.push(ch);
14         } else if (ch == '}' || ch == ')' || ch == ']') {
15             if (stack.isEmpty()) {
16                 return false;
17             }
18             char top = stack.pop();
19             if ((ch == '}' && top != '{') || (ch == ')' && top != '(') || (ch == ']' && top != '[')) {
20                 return false;
21             }
22         }
23     }
24     return stack.isEmpty();
25 }
26
27 public static void main(String[] args) {
28
29     Scanner scanner = new Scanner(System.in);
30     System.out.print("Enter the expression: ");
31     String s = scanner.nextLine();
32     System.out.println(isBalanced(s));
33 }

```

The console output at the bottom shows:

```

Enter the expression: {{{}}}
true

```

4. Equilibrium Point

```
package JavaPractice;
```

```
import java.util.Scanner;
```

```
public class Array {
```

```
    public static int findEquilibriumPoint(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 + 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 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();
}

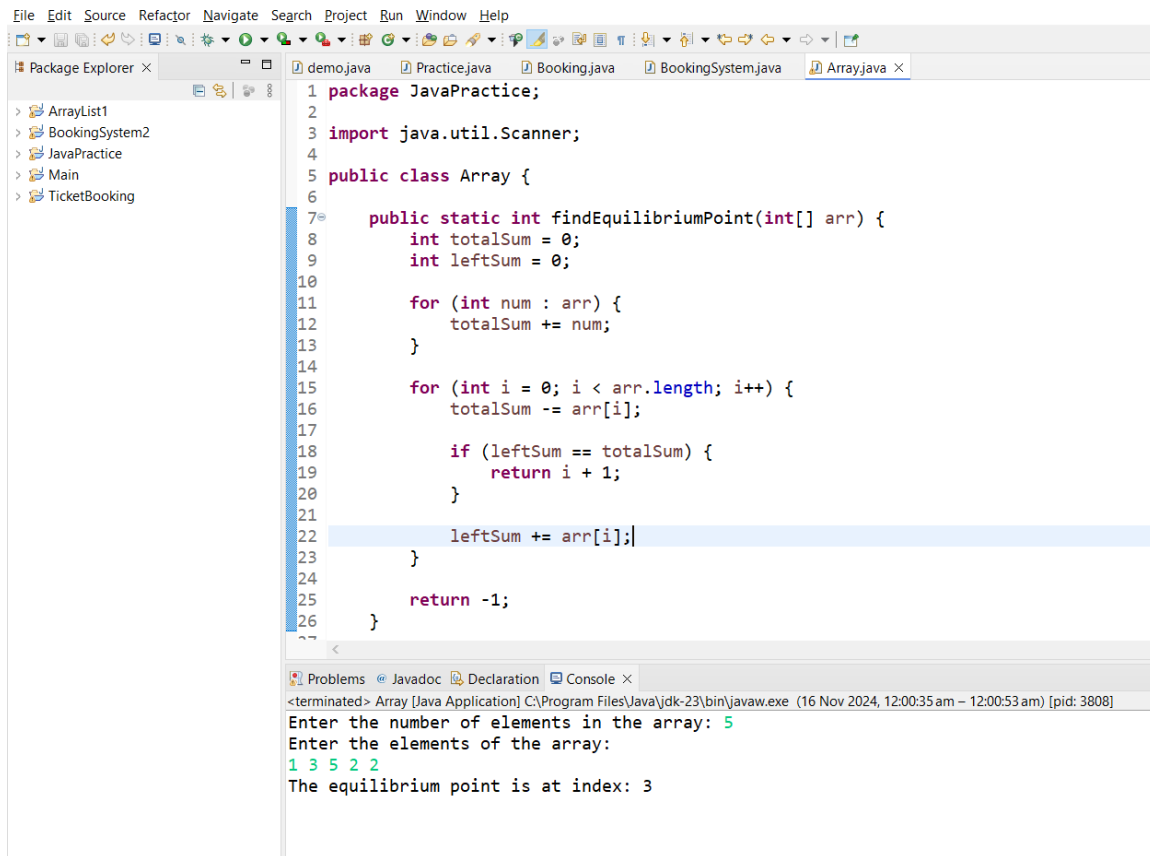
int result = findEquilibriumPoint(arr);

if (result == -1) {
    System.out.println("No equilibrium point found");
} else {
    System.out.println("The equilibrium point is at index: " +
result);
}
}

```

Time Complexity: $O(n)$

OUTPUT:



5. Binary Search

```
package JavaPractice;
```

```
import java.util.Scanner;
```

```
public class Array {
```

```
    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) {
            right = mid - 1;
        } else {
            left = 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 target element to search: ");

    int target = scanner.nextInt();

    int result = binarySearch(arr, target);

    System.out.println(result);

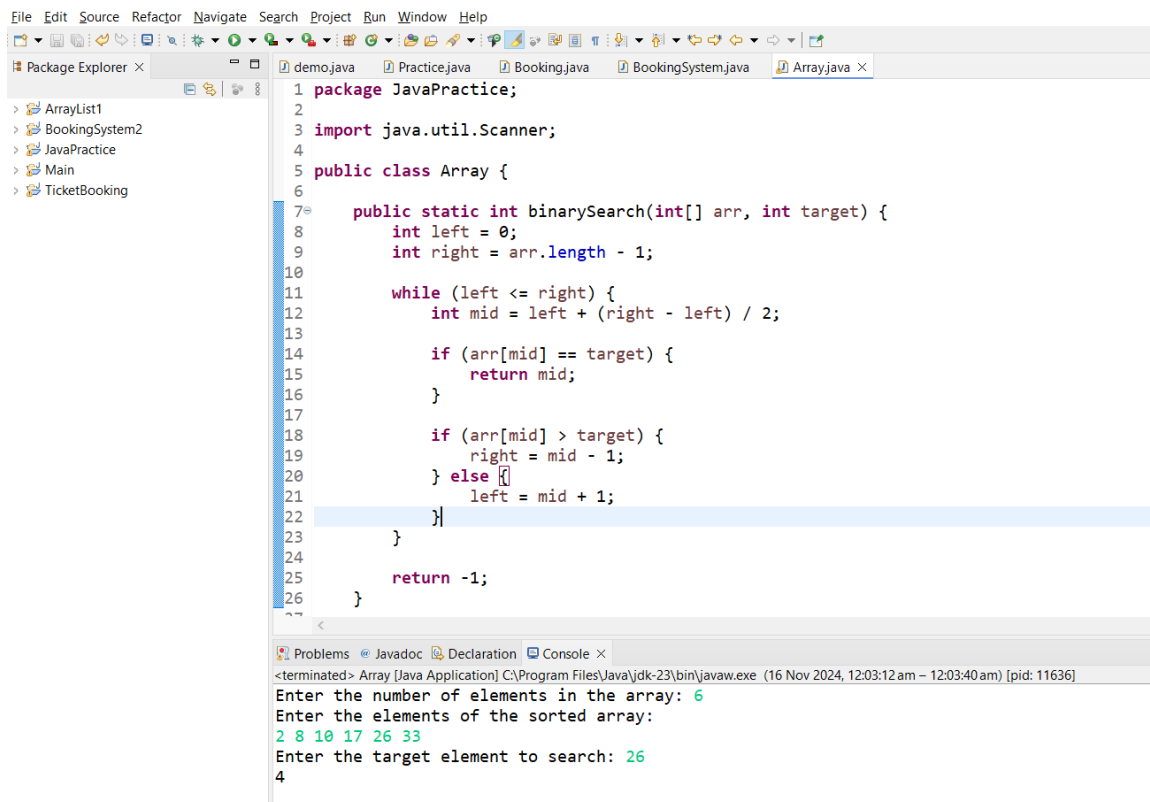
}

}

```

Time Complexity: $O(\log n)$

OUTPUT:



The screenshot shows an IDE with a Java project named 'JavaPractice'. The 'Package Explorer' on the left lists several packages, including 'Main'. The main editor displays the 'Array.java' file, which contains a 'binarySearch' method. The console at the bottom shows the execution output, including prompts for the number of elements, the array elements, and the target element, along with the resulting index.

```

1 package JavaPractice;
2
3 import java.util.Scanner;
4
5 public class Array {
6
7     public static int binarySearch(int[] arr, int target) {
8         int left = 0;
9         int right = arr.length - 1;
10
11         while (left <= right) {
12             int mid = left + (right - left) / 2;
13
14             if (arr[mid] == target) {
15                 return mid;
16             }
17
18             if (arr[mid] > target) {
19                 right = mid - 1;
20             } else {
21                 left = mid + 1;
22             }
23         }
24
25         return -1;
26     }
27 }

```

```

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (16 Nov 2024, 12:03:12 am - 12:03:40 am) [pid: 11636]
Enter the number of elements in the array: 6
Enter the elements of the sorted array:
2 8 10 17 26 33
Enter the target element to search: 26
4

```

6. Next greater element

```
package JavaPractice;

import java.util.Stack;
import java.util.*;
public class Array {
    public static void printNGE(int[] arr) {
        int n = arr.length;
        int[] nge = 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();
            }
            nge[i] = stack.isEmpty() ? -1 : stack.peek();
            stack.push(arr[i]);
        }

        for (int i = 0; i < n; i++) {
            System.out.println(arr[i] + " -> " + nge[i]);
        }
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the size of the array : ");
        int n=sc.nextInt();
```

```

    int[] arr = new int[n];

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

    for (int i = 0; i < n; i++)
    {
        arr[i]=sc.nextInt();
    }

    printNGE(arr);
}
}

```

Time Complexity: $O(n)$

OUTPUT:

```

1 package JavaPractice;
2
3 import java.util.Stack;
4 import java.util.*;
5 public class Array {
6     public static void printNGE(int[] arr) {
7         int n = arr.length;
8         int[] nge = new int[n];
9         Stack<Integer> stack = new Stack<>();
10
11         for (int i = n - 1; i >= 0; i--) {
12             while (!stack.isEmpty() && stack.peek() <= arr[i]) {
13                 stack.pop();
14             }
15             nge[i] = stack.isEmpty() ? -1 : stack.peek();
16             stack.push(arr[i]);
17         }
18
19         for (int i = 0; i < n; i++) {
20             System.out.println(arr[i] + " -> " + nge[i]);
21         }
22     }
23
24     public static void main(String[] args) {
25         Scanner sc=new Scanner(System.in);

```

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (16 Nov 2024, 12:07:11 am - 12:07:28 am) [pid: 8540]
 Enter the size of the array :
 5
 Enter the elements of the array :
 2 4 5 10 11
 2 -> 4
 4 -> 5
 5 -> 10
 10 -> 11
 11 -> -1

7.Union of two arrays with duplicate elements

```
package JavaPractice;

import java.util.HashSet;
import java.util.Scanner;

public class Array {
    public static int findUnion(int[] a, int[] b) {
        HashSet<Integer> uList = new HashSet<>();
        for (int num : a) {
            uList.add(num);
        }
        for (int num : b) {
            uList.add(num);
        }
        return uList.size();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of elements in array 1: ");
        int n1 = sc.nextInt();
        int[] arr1 = new int[n1];
        System.out.println("Enter elements of array 1:");
        for (int i = 0; i < n1; i++) {
            arr1[i] = sc.nextInt();
        }
    }
}
```

```

    }

    System.out.print("Enter number of elements in array 2: ");
    int n2 = sc.nextInt();
    int[] arr2 = new int[n2];
    System.out.println("Enter elements of array 2:");
    for (int i = 0; i < n2; i++) {
        arr2[i] = sc.nextInt();
    }

    int unionSize = findUnion(arr1, arr2);

    System.out.println("Union size (unique elements): " +
unionSize);

    sc.close();
}
}

```

Time Complexity: $O(n+m)$

OUTPUT:

The screenshot shows an IDE with the following components:

- Package Explorer:** Shows a project structure with folders: ArrayList1, BookingSystem2, JavaPractice, Main, and TicketBooking.
- Editor:** Displays the code for `Array.java`. The code defines a `findUnion` method using a `HashSet` to find the union of two integer arrays and a `main` method to take user input and print the result.
- Console:** Shows the execution output, including the number of elements entered for each array, the elements themselves, and the final union size.

```
3 import java.util.HashSet;
4 import java.util.Scanner;
5
6 public class Array {
7     public static int findUnion(int[] a, int[] b) {
8         HashSet<Integer> uList = new HashSet<>();
9         for (int num : a) {
10             uList.add(num);
11         }
12         for (int num : b) {
13             uList.add(num);
14         }
15         return uList.size();
16     }
17
18     public static void main(String[] args) {
19         Scanner sc = new Scanner(System.in);
20
21         System.out.print("Enter number of elements in array 1: ");
22         int n1 = sc.nextInt();
23         int[] arr1 = new int[n1];
24         System.out.println("Enter elements of array 1:");
25         for (int i = 0; i < n1; i++) {
26             arr1[i] = sc.nextInt();
27         }
28     }
29 }
```

Console Output:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (16 Nov 2024, 12:10:23 am – 12:10:38 am) [pid: 7604]
Enter number of elements in array 1: 5
Enter elements of array 1:
1 2 3 4 5
Enter number of elements in array 2: 3
Enter elements of array 2:
1 2 3
Union size (unique elements): 5
```