

Ranjith R 22IT085 Day 4 DSA Practice

1)String Anagram

Given two strings S1 and S2 . Return "1" if both strings are anagrams otherwise return "0" .

Note: An anagram of a string is another string with exactly the same quantity of each character in it, in any order.

Example 1:

Input: S1 = "cdbkdub" , S2 = "dsbkcsdn"

Output: 0

Explanation: Length of S1 is not same
as length of S2.

Program:

```
package JavaPractice;

import java.util.Scanner;

class Solution {
    static int areAnagram(String S1, String S2) {
        if (S1.length() != S2.length()) {
            return 0;
        }

        int[] arr = new int[26];
        for (char ch : S1.toCharArray()) {
```

```

        arr[ch - 'a']++;
    }
    for (char ch : S2.toCharArray()) {
        arr[ch - 'a']--;
    }
    for (int n : arr) {
        if (n != 0) {
            return 0;
        }
    }
    return 1;
}
}

```

```

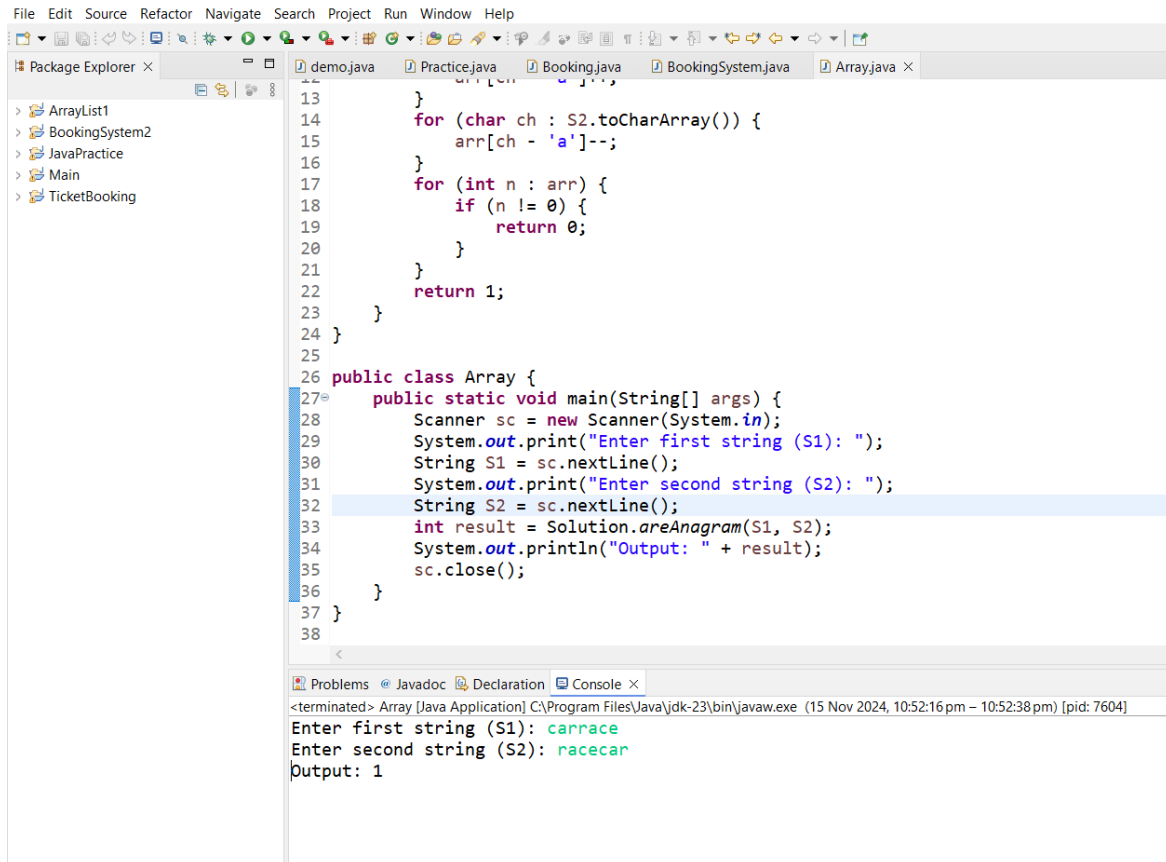
public class Array {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first string (S1): ");
        String S1 = sc.nextLine();
        System.out.print("Enter second string (S2): ");
        String S2 = sc.nextLine();
        int result = Solution.areAnagram(S1, S2);
        System.out.println("Output: " + result);
        sc.close();
    }
}

```

}

Time Complexity: $O(n)$

OUTPUT:



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

```
13     }
14     for (char ch : S2.toCharArray()) {
15         arr[ch - 'a']--;
16     }
17     for (int n : arr) {
18         if (n != 0) {
19             return 0;
20         }
21     }
22     return 1;
23 }
24 }
25
26 public class Array {
27     public static void main(String[] args) {
28         Scanner sc = new Scanner(System.in);
29         System.out.print("Enter first string (S1): ");
30         String S1 = sc.nextLine();
31         System.out.print("Enter second string (S2): ");
32         String S2 = sc.nextLine();
33         int result = Solution.areAnagram(S1, S2);
34         System.out.println("Output: " + result);
35         sc.close();
36     }
37 }
38
```

The console output at the bottom shows the execution results:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 10:52:16 pm - 10:52:38 pm) [pid: 7604]
Enter first string (S1): carrace
Enter second string (S2): racecar
Output: 1
```

2)Row with max 1s

You are given a 2D array consisting of only 1's and 0's, where each row is sorted in non-decreasing order. You need to find and return the index of the first row that has the most number of 1s. If no such row exists, return -1.

Note: 0-based indexing is followed.

Examples:

Input: arr[][] = [[0, 1, 1, 1],

[0, 0, 1, 1],

[1, 1, 1, 1],

[0, 0, 0, 0]]

Output: 2

Explanation: Row 2 contains 4 1's.

PROGRAM:

```
package JavaPractice;
```

```
class Solution {
```

```
    public int rowWithMax1s(int arr[][]) {
```

```
        int n = arr.length;
```

```
        if (n == 0) return -1;
```

```
        int m = arr[0].length;
```

```
        int maxRowIndex = -1;
```

```
        int j = m - 1;
```

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

```
            while (j >= 0 && arr[i][j] == 1) {
```

```
                maxRowIndex = i;
```

```
                j--;
```

```
            }
```

```
        }
```

```
        return maxRowIndex;
```

```

    }
}

public class Array {
    public static void main(String[] args) {
        int[][] arr = {
            {0, 1, 1, 1},
            {0, 0, 1, 1},
            {1, 1, 1, 1},
            {0, 0, 0, 0}
        };
        Solution solution = new Solution();
        System.out.println(solution.rowWithMax1s(arr));
    }
}

```

TIME COMPLEXITY: $O(m+n)$

OUTPUT:

The screenshot shows an IDE with the following components:

- Package Explorer:** Contains a package named `JavaPractice` with files `ArrayList1`, `BookingSystem2`, `Main`, and `TicketBooking`.
- Editor:** Displays the code for `Array.java`. The code defines a `Solution` class with a `rowWithMax1s` method and a `Array` class with a `main` method. The `main` method creates an instance of `Solution` and prints the result of `rowWithMax1s` on a 2D array.
- Console:** Shows the output of the program, which is the number `2`.

```
6      int m = arr[0].length;
7      int maxRowIndex = -1;
8      int j = m - 1;
9      for (int i = 0; i < n; i++) {
10         while (j >= 0 && arr[i][j] == 1) {
11             maxRowIndex = i;
12             j--;
13         }
14     }
15     return maxRowIndex;
16 }
17
18 public class Array {
19     public static void main(String[] args) {
20         int[][] arr = {
21             {0, 1, 1, 1},
22             {0, 0, 1, 1},
23             {1, 1, 1, 1},
24             {0, 0, 0, 0}
25         };
26         Solution solution = new Solution();
27         System.out.println(solution.rowWithMax1s(arr));
28     }
29 }
30
31
```

<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 10:53:14 pm - 10:53:15 pm) [pid: 13052]
2

3. Longest consecutive subsequence

Given an array of non-negative integers. Find the length of the longest subsequence such that elements in the subsequence are consecutive integers, the consecutive numbers can be in any order.

Examples:

Input: `arr[] = [2, 6, 1, 9, 4, 5, 3]`

Output: 6

Explanation: The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsequence.

PROGRAM:

```
package JavaPractice;

import java.util.HashSet;

import java.util.Scanner;
```

```

public class Array {
    public static int findLongestConseqSubseq(int[] arr) {
        HashSet<Integer> set = new HashSet<>();

        for (int num : arr) {
            set.add(num);
        }

        int longestStreak = 0;

        for (int num : arr) {
            if (!set.contains(num - 1)) {
                int currentNum = num;
                int currentStreak = 1;

                while (set.contains(currentNum + 1)) {
                    currentNum++;
                    currentStreak++;
                }

                longestStreak = Math.max(longestStreak,
currentStreak);
            }
        }

        return longestStreak;
    }
}

```

```

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 = findLongestConseqSubseq(arr);

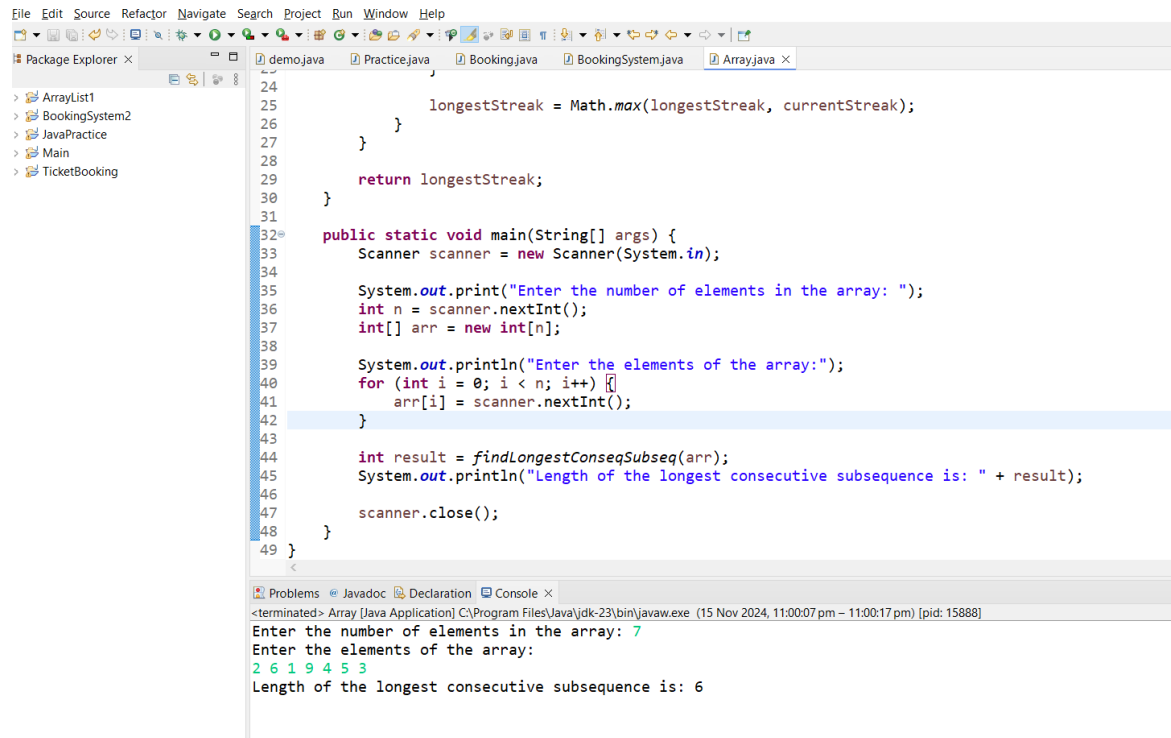
    System.out.println("Length of the longest consecutive
subsequence is: " + result);

    scanner.close();
}
}

```

Time Complexity: $O(n)$

OUTPUT:



```
File Edit Source Refactor Navigate Search Project Run Window Help
demo.java Practice.java Booking.java BookingSystem.java Array.java x
Package Explorer x
> ArrayList1
> BookingSystem2
> JavaPractice
> Main
> TicketBooking

24
25         longestStreak = Math.max(longestStreak, currentStreak);
26     }
27 }
28
29     return longestStreak;
30 }
31
32 public static void main(String[] args) {
33     Scanner scanner = new Scanner(System.in);
34
35     System.out.print("Enter the number of elements in the array: ");
36     int n = scanner.nextInt();
37     int[] arr = new int[n];
38
39     System.out.println("Enter the elements of the array:");
40     for (int i = 0; i < n; i++) {
41         arr[i] = scanner.nextInt();
42     }
43
44     int result = findLongestConseqSubseq(arr);
45     System.out.println("Length of the longest consecutive subsequence is: " + result);
46
47     scanner.close();
48 }
49 }

Problems Javadoc Declaration Console x
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 11:00:07 pm - 11:00:17 pm) [pid: 15888]
Enter the number of elements in the array: 7
Enter the elements of the array:
2 6 1 9 4 5 3
Length of the longest consecutive subsequence is: 6
```

4. Longest palindrome in a string

Given a string S, find the longest palindromic substring in S. Substring of string S: $S[i \dots j]$ where $0 \leq i \leq j < \text{len}(S)$. Palindrome string: A string which reads the same backwards. More formally, S is palindrome if $\text{reverse}(S) = S$. In case of conflict, return the substring which occurs first (with the least starting index).

Example 1:

Input: S = "aaaabbaa"

Output: aabbaa

Explanation: The longest palindrome string present in the given string is "aabbaa".

PROGRAM:

```
package JavaPractice;
```

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

```

public class Array {
    public static String longestPalindrome(String s) {
        if (s == null || s.length() < 1) return "";

        int start = 0, end = 0;

        for (int i = 0; i < s.length(); i++) {
            int len1 = expandAroundCenter(s, i, i);
            int len2 = expandAroundCenter(s, i, i + 1);
            int len = Math.max(len1, len2);

            if (len > end - start) {
                start = i - (len - 1) / 2;
                end = i + len / 2;
            }
        }

        return s.substring(start, end + 1);
    }

    private static int expandAroundCenter(String s, int left, int
right) {
        while (left >= 0 && right < s.length() && s.charAt(left) ==
s.charAt(right)) {
            left--;
            right++;
        }
        return right - left - 1;
    }
}

```

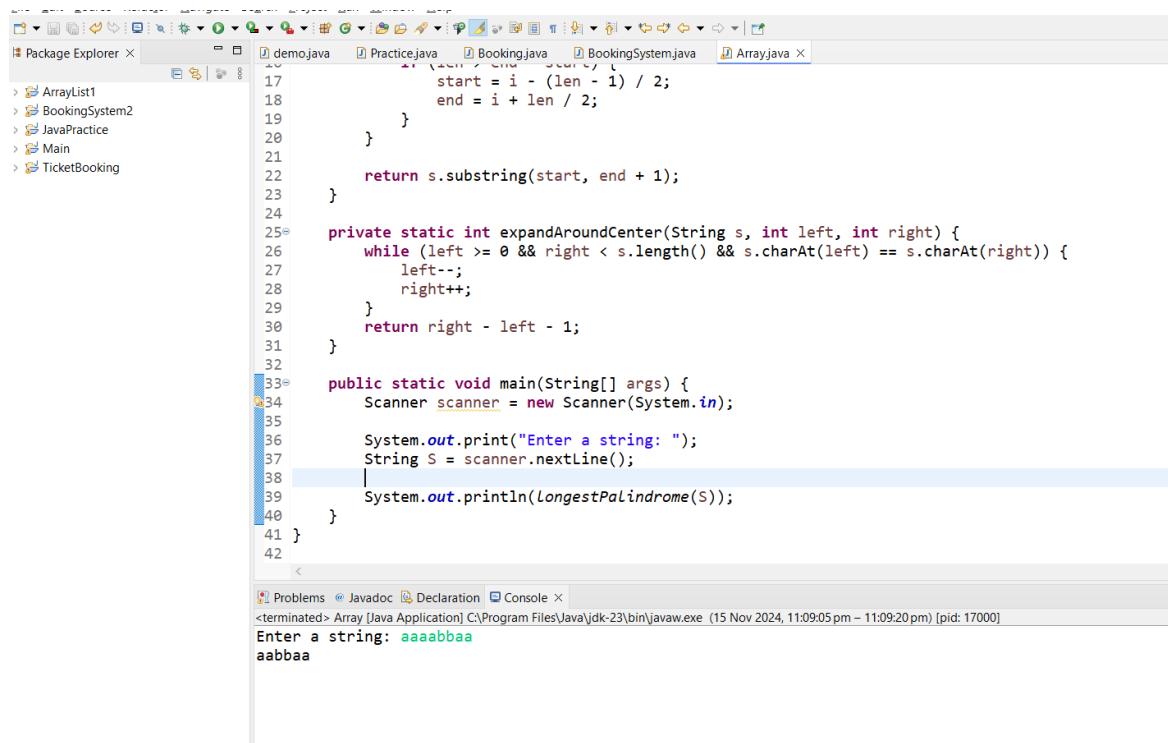
```
}
```

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.print("Enter a string: ");  
    String S = scanner.nextLine();  
  
    System.out.println(LongestPalindrome(S));  
}
```

```
}
```

Time Complexity: $O(n^2)$

OUTPUT:



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

```
17         start = i - (len - 1) / 2;  
18         end = i + len / 2;  
19     }  
20 }  
21  
22     return s.substring(start, end + 1);  
23 }  
24  
25 private static int expandAroundCenter(String s, int left, int right) {  
26     while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {  
27         left--;  
28         right++;  
29     }  
30     return right - left - 1;  
31 }  
32  
33 public static void main(String[] args) {  
34     Scanner scanner = new Scanner(System.in);  
35  
36     System.out.print("Enter a string: ");  
37     String S = scanner.nextLine();  
38     |  
39     System.out.println(LongestPalindrome(S));  
40 }  
41 }  
42
```

The console output at the bottom shows the execution of the program:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk-23\bin\javaw.exe (15 Nov 2024, 11:09:05 pm - 11:09:20 pm) [pid: 17000]  
Enter a string: aaaabbbaa  
aabbbaa
```

