12/11/2024

DSA(Practice-3)

1.Anagram:

Coding:

```
import java.util.Arrays;
public class AnagramCheck {
    public static boolean areAnagrams(String s1, String s2) {
        if (s1.length() != s2.length()) return false;
        char[] str1 = s1.toCharArray();
        char[] str2 = s2.toCharArray();
        Arrays.sort(str1);
        Arrays.sort(str2);
        return Arrays.equals(str1, str2);
    }

    public static void main(String[] args) {
        String s1 = "geeks";
        String s2 = "kseeg";
        System.out.println(areAnagrams(s1, s2));
    }
}
```

Output:

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

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java AnagramCheck
true
```

Time complexity:O(N log N)

2. row with max 1s':

Coding:

```
import java.util.Scanner;
public class Max1sRowFinder {
    public static int rowWithMax1s(int[][] matrix) {
        int maxRow = -1;
        int rows = matrix.length;
        int cols = matrix[0].length;
        int j = cols - 1;
        for (int i = 0; i < rows; i++) {
```

```
            while (j >= 0 && matrix[i][j] == 1) {
                j--;
                maxRow = i;
            }
        }
        return maxRow;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = scanner.nextInt();
        int[][] matrix = new int[rows][cols];
        System.out.println("Enter the matrix values (0s and 1s): ");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
        int result = rowWithMax1s(matrix);
        if (result != -1) {
            System.out.println("Row with the maximum number of 1s: " + result);
        } else {
            System.out.println("No 1s in the matrix.");
        }

        scanner.close();
    }
}
```

Output:

```
C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java Max1sRowFinder
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix values (0s and 1s):
0
1
0
0
0
1
0
0
1
Row with the maximum number of 1s: 1
```

Time complexity:o(1)

3.

Coding:

```
import java.util.HashSet;
public class LongestConsecutiveSubsequence {
    public static int findLongestConsecutiveSubsequence(int[] arr) {
        if (arr.length == 0) return 0;
        HashSet<Integer> set = new HashSet<>();
        for (int num : arr) {
            set.add(num);
        }
        int longestStreak = 0;
        for (int num : set) {
            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) {
        int[] arr = {100, 4, 200, 1, 3, 2};
        System.out.println("Length of the longest consecutive subsequence is: " +
findLongestConsecutiveSubsequence(arr));
    }
}
```

Output:

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

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java LongestConsecutiveSubsequence
Length of the longest consecutive subsequence is: 4
```

Time complexity:o(1)

4.longest palindrome in a string:

Coding:

```
import java.util.*;
public class LongestPalindromeSubstring {
    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; // Length of palindrome
    }
    public static void main(String[] args) {
        String s = "babad";
        System.out.println("Longest palindromic substring is: " + longestPalindrome(s));
    }
}
```

Output:

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

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java LongestPalindromeSubstring
Longest palindromic substring is: aba
```

Time complexity:o(n^2)

5.rat in a maze problem:

Coding:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class RatInMaze {
    private static void findPaths(int[][] mat, int x, int y, String path, List<String> paths) {
        int n = mat.length;
        if (x < 0 || y < 0 || x >= n || y >= n || mat[x][y] == 0) {
            return;
        }
        if (x == n - 1 && y == n - 1) {
```

```
            paths.add(path);
            return;
        }
        mat[x][y] = 0;
        findPaths(mat, x + 1, y, path + "D", paths);
        findPaths(mat, x, y - 1, path + "L", paths);
        findPaths(mat, x, y + 1, path + "R", paths);
        findPaths(mat, x - 1, y, path + "U", paths);
        mat[x][y] = 1;
    }
    public static List<String> findPath(int[][] mat) {
        List<String> paths = new ArrayList<>();
        if (mat[0][0] == 1) {
            findPaths(mat, 0, 0, "", paths);
            Collections.sort(paths);
        }
        return paths;
    }
    public static void main(String[] args) {
        int[][] mat = {
            {1, 0, 0, 0},
            {1, 1, 0, 1},
            {1, 1, 0, 0},
            {0, 1, 1, 1}
        };
        List<String> result = findPath(mat);
        if (result.isEmpty()) {
            System.out.println("-1");
        } else {
            System.out.println(String.join(" ", result));
        }
    }
}
```

Solution:

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

C:\Users\Sadhasivam v\OneDrive\Desktop\java practice>java RatInMaze
DDRDRR DRDDRR
```