Name: Jai Darshan S

Dept: AI&DS

Date: 12/11/24

Practice set 3:

1. Anagram

Java code:

public class Anagram {

    public static boolean checkanagram(String s1,String s2){

        if (s1.length()!=s2.length()){

            return false;

        }

        int[] freq=new int[26];

        for (int i=0;i<s1.length();i++){

            freq[s1.charAt(i)-'a']++;

        }

        for (int j=0;j<s2.length();j++){

            freq[s2.charAt(j)-'a']--;

        }

        for (int k:freq){

            if (k!=0){

                return false;

            }

        }

        return true;

    }

    public static void main(String[] args) {

        String s1="geeks",s2="kseeg";

        String s3="allergy",s4="allergic";

        System.out.println(checkanagram(s1, s2));

        System.out.println(checkanagram(s3, s4));

    }

}

A black screen with white text

Description automatically generatedOutput:

Time complexity: O(n)

Space complexity: O(1)

1. Row With Max 1’s:

Java code:

public class RowWithMaxOnes {

    public static int row(int[][] arr){

        int s=0;

        int r=-1;

        for (int i=0;i<arr.length;i++){

            int c=0;

            for (int j=0;j<arr[0].length;j++){

                if (arr[i][j]==1){

                    c+=1;

                }

            }

            if (c>s){

                s=c;

                r=i;

            }

        }

        return r;

    }

    public static void main(String[] args) {

        int[][] arr1={

            {0, 1, 1, 1},

            {0, 0, 1, 1},

            {1, 1, 1, 1},

            {0, 0, 0, 0}

        };

        int[][] arr2={

            {0,0},

            {1,1}

        };

        System.out.println(row(arr1));

        System.out.println(row(arr2));

    }

}

A black screen with blue text

Description automatically generatedOutput:

Time complexity: O(mxn)

Space complexity: O(1)

1. Longest consequtive subsequence:

Java code:

import java.util.HashSet;

import java.util.Arrays;

public class LongestConsecutiveSubsequence {

    // Function to return length of longest subsequence of consecutive integers.

    public static int find(int[] arr) {

        // code here

        if (arr.length==0){

            return 0;

        }

        HashSet<Integer> set=new HashSet<>();

        for (int i:arr){

            set.add(i);

        }

        Integer[] ar=set.toArray(new Integer[0]);

        Arrays.sort(ar);

        int r=1;

        int c=1;

        for (int i=1;i<ar.length;i++){

            if (ar[i]-ar[i-1]==1){

                c++;

            }else{

                r=Math.max(r,c);

                c=1;

            }

        }

        r=Math.max(r,c);

        return r;

    }

    public static void main(String[] args) {

        int[] arr1={2, 6, 1, 9, 4, 5, 3};

        int[] arr2={1, 9, 3, 10, 4, 20, 2};

        int[] arr3={15, 13, 12, 14, 11, 10, 9};

        System.out.println(find(arr1));

        System.out.println(find(arr2));

        System.out.println(find(arr3));

    }

}

A black screen with blue text

Description automatically generatedOutput:

Time complexity: O(n log n)

Space complexity: O(n)

1. Longest Palindromic Substring:

Java code:

public class LongestPalindromicSubString {

    public static boolean palindrome(String s,int low,int high){

        while (low<high) {

            if (s.charAt(low)!=s.charAt(high)){

                return false;

            }

            low++;

            high--;

        }

        return true;

    }

    public static String longest(String s){

        int m=1;

        int st=0;

        for (int i=0;i<s.length();i++){

            for (int j=i;j<s.length();j++){

                if (palindrome(s,i,j) && (j-i+1)>m){

                    st=i;

                    m=(j-i+1);

                }

            }

        }

        return s.substring(st, st+m);

    }

    public static void main(String[] args) {

        String s1="aaaabbaa";

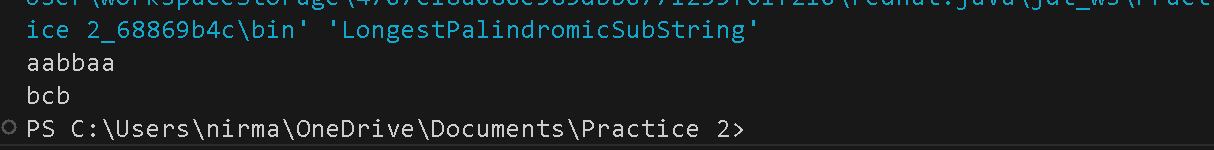
        String s2="abcbd";

        System.out.println(longest(s1));

        System.out.println(longest(s2));

    }

}

Output:

Time complexity: O(n^3)

Space complexity: O(1)

1. Rat in a Maze Problem

Java code:

import java.util.ArrayList;

public class RatOnAMaze {

    static String direction = "DLRU";

    static int[] dr = { 1, 0, 0, -1 };

    static int[] dc = { 0, -1, 1, 0 };

    static boolean isValid(int row, int col, int n, int[][] maze)

    {

        return row >= 0 && col >= 0 && row < n && col < n

            && maze[row][col] == 1;

    }

    static void findPath(int row, int col, int[][] maze,

                         int n, ArrayList<String> ans,

                         StringBuilder currentPath)

    {

        if (row == n - 1 && col == n - 1) {

            ans.add(currentPath.toString());

            return;

        }

        maze[row][col] = 0;

        for (int i = 0; i < 4; i++) {

            int nextrow = row + dr[i];

            int nextcol = col + dc[i];

            if (isValid(nextrow, nextcol, n, maze)) {

                currentPath.append(direction.charAt(i));

                findPath(nextrow, nextcol, maze, n, ans,

                         currentPath);

                currentPath.deleteCharAt(

                    currentPath.length() - 1);

            }

        }

        maze[row][col] = 1;

    }

    public static void main(String[] args)

    {

        int[][] maze1 = {

            { 1, 0, 0, 0 },

            { 1, 1, 0, 1 },

            { 1, 1, 0, 0 },

            { 0, 1, 1, 1 }

        };

        int n1 = maze1.length;

        ArrayList<String> result = new ArrayList<>();

        StringBuilder currentPath = new StringBuilder();

        if (maze1[0][0] != 0 && maze1[n1 - 1][n1 - 1] != 0) {

            findPath(0, 0, maze1, n1, result, currentPath);

        }

        if (result.size() == 0)

            System.out.println(-1);

        else

            for (String path : result)

                System.out.println(path + " ");

        System.out.println();

    }

}

A black background with white text

Description automatically generatedOutput:

Time complexity: O(3^(mxn))

Space complexity: O(mxn)