**12/11/24**

2) row with max 1s' :

soln :

class Solution {

public int[] rowAndMaximumOnes(int[][] mat) {

int r[]=new int [2];

int n;

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

n=0;

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

if(mat[i][j] == 1)

n++;

}

if(n>r[1]){

r[1]=n;

r[0]=i;

}

}

return r;

}

}

TC : O(m X N)

SC : O(1)

1) anagram program:

soln :

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

int n = s1.length();

int m = s2.length();

if(n!=m)

return false;

char arr[] = new char[26];

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

arr[s1.charAt(i) - 'a']++;

arr[s2.charAt(i) - 'a']--;

}

for(int i:arr) {

if(i!=0)

return false;

}

return true;

}

Tc : O(n)

SC : O(n)

5) rat in a maze problem :

soln :

class Solution {

public ArrayList<String> findPath(int[][] mat) {

int n = mat.length;

int vis[][] = new int[n][n];

vis[0][0] = 1;

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

String cp = "";

if(mat[0][0] == 0 || mat[n-1][n-1] == 0){

res.add("-1");

return res;

}

rec(0,0,n,mat,res,cp,vis);

return res;

}

public static void rec(int r ,int c, int n, int mat[][],ArrayList<String> res, String cp , int vis[][]){

if(c == n-1 && r == n-1){

res.add(cp);

return;

}

if(c-1>=0 && vis[r][c-1]!=1 && mat[r][c-1] == 1){

vis[r][c-1] = 1;

rec(r , c-1 , n , mat , res , cp+"L" , vis);

vis[r][c-1] = 0;

}

if(c+1<n && vis[r][c+1]!=1 && mat[r][c+1] == 1){

vis[r][c+1] = 1;

rec(r , c+1 , n , mat , res , cp+"R" , vis);

vis[r][c+1] = 0;

}

if(r-1>=0 && vis[r-1][c]!=1 && mat[r-1][c] == 1){

vis[r-1][c] = 1;

rec(r-1 , c , n , mat , res , cp+"U" , vis);

vis[r-1][c] = 0;

}

if(r+1<n && vis[r+1][c]!=1 && mat[r+1][c] == 1){

vis[r+1][c] = 1;

rec(r+1 , c , n , mat , res , cp+"D" , vis);

vis[r+1][c] = 0;

}

}

}

TC : O(4^n)

Sc : o(n X m)

3) Longest consequtive subsequence :

soln :

class Solution {

public int longestConsecutive(int[] nums) {

int r = 0;

Set<Integer> l = new HashSet<>();

for(int i:nums)

l.add(i);

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

if(l.contains(nums[i]-1))

continue;

int c = 0;

int j = nums[i];

while(l.contains(j)){

c++;

j++;

}

if(c>r)

r = c;

}

return r;

}

}

TC : O(n);

SC : O(n);

4) longest palindrome in a string :

soln :

class Solution {

public String longestPalindrome(String s) {

int n = s.length();

if (n == 1)

return s;

String pal = "";

for (int i = 1; i < n; i++) {

int l = i;

int r = i;

while (s.charAt(r) == s.charAt(l)) {

l--;

r++;

if (l < 0 || r >= n)

break;

}

String tem = s.substring(l + 1, r);

if (tem.length() > pal.length())

pal = tem;

l = i-1;

r = i;

while (s.charAt(r) == s.charAt(l)) {

l--;

r++;

if (l < 0 || r >= n)

break;

}

tem = s.substring(l + 1, r);

if ( tem.length() > pal.length())

pal = tem;

}

return pal;

}

TC : O(n^2)

Sc : O(1)