# In a school, The class teacher assigned

# a task to the CR(class representative) of a class,

# Given an integer array choclates[] where choclates[i] denotes the color of

# the ith choclate.

# Teacher instructed to the CR to distribute the choclates by

# sharing p continuous(consecutive) choclates among the students.

# But CR wants to preserve as many colors of choclates as possible.

# Print the max count of unique colors of choclates that CR preserved

# after distribution.

# Input Format:

# -------------

# Line-1: Two integers L and p (space separated), L number of choclates[], and p value.

# Line-2: L number of space separated colors of choclates.

# Output Format:

# --------------

# Print the max count of unique colors.

# Sample Input-1:

# ---------------

# 6 3

# 3 3 4 5 4 1

# Sample Output-1:

# ----------------

# 3

# Explanation:

# ------------

# CR can distribute choclates from index 0 to 2 with colors 3,3,4

# Now remaining unique colors are 5,4,1 hence print 3.

# Sample Input-2:

# ---------------

# 6 2

# 9 9 9 9 7 7

# Sample Output-2:

# ----------------

# 2

# Explanation:

# ------------

# CR can distribute choclates from index 3 to 4 with colors 9,7.

# The preserved choclates are 9,9,9,7.

# unique colors are 9,7 hence print 2.

# Else, CR can distribute choclates with colors 9,9

# and preserve the choclates with colors 9,9,7,7.

# Sample Input-3:

# ---------------

# 3 0

# 4 6 8

# Sample Output-3:

# ----------------

# 3

# Explanation:

# ------------

# CR do not have to distribute any choclates.

# can preserve the choclates with colors 4,6,8.

# unique colors are 4,6,8 hence print 3.

n,m=list(map(int,input().split()))

l=list(map(int,input().split()))

d=dict()

s=set(l)

maxx=0

lenn=len(s)

for i in range(n-m+1):

sss={x for x in l[0:i]+l[i+m:]}

maxx=max(maxx,len(sss))

print(maxx)

Birbal is creating a new binary code system BBC (Birbal Binary Code) as follows:

I f(I)

-------

0 ""

1 "0"

2 "1"

3 "00"

4 "01"

5 "10"

6 "11"

7 "000"

You are given an integer value I, where I is positive number.

Your task is to find BBC representation of the given number I.

Input Format:

-------------

An integer I, 0 <= I <= 10^9

Output Format:

--------------

Print the BBC representation of I.

Sample Input-1:

---------------

23

Sample Output-1:

----------------

1000

Sample Input-2:

---------------

45

Sample Output-2:

----------------

01110

“I am getting out of memory for this code”

import java.util.\*;

public class Main{

static ArrayList<String> al=new ArrayList<>();

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

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

al.add("0");

al.add("1");

// int num=(int)Math.sqrt(n);

// int num=Integer.parseInt(Integer.toBinaryString(num))

String nums=Integer.toBinaryString(n);

int num=nums.length();

ArrayList<String> ll=Strobo(al,num);

// System.out.println(res);

System.out.println(res.get(n-1));

}

public static ArrayList<String> Strobo(ArrayList<String> al,int n){

// if(n==0){

// return ;

// }

if(n==1){

res.addAll(al);

return al;

}

ArrayList<String> l=Strobo(al,n-1);

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

for (String i : l){

ln.add("0"+i);

}

for (String i : l){

ln.add("1"+i);

}

res.addAll(ln);

return ln;

}

}

The shortcut for this code for the given input add one number and convert it into binary and remove the 1st number

import java.util.\*;

public class Main{

static ArrayList<String> al=new ArrayList<>();

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

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String nums=Integer.toBinaryString(n+1);

System.out.println(nums.substring(1));

}

}

In a school Students are given four integers which defines minimumLength,

maximumLength, oneGroup and zeroGroup related to strings

A binary string is good if it satisfies the following conditions:

-- The length of the string is in the range [minimumLength, maximumLength].

-- The size of each block of consecutive 1's is a multiple of oneGroup.

For example in a binary string 00110111100 sizes of each block of

consecutive ones are [2,4].

-- The size of each block of consecutive 0's is a multiple of zeroGroup.

For example, in a binary string 00110111100 sizes of each block of

consecutive zeros are [2,1,2].

Return the number of good binary strings. Since the answer may be too large,

return it modulo 10^9 + 7.

Note that 0 is considered a multiple of all the numbers.

Input Format

------------

read min, max, onegroup, zerogroup four integers

output Format

------------

An integer which is no of good 01strings

Sample Input-1:

---------------

2 3 1 2

Sample Output-1:

----------------

5

Explanation:

------------

There are 5 good binary strings in this example: "00", "11", "001", "100", and "111".

It can be proven that there are only 5 good strings satisfying all conditions.

Sample Input-2:

---------------

4 4 4 3

Sample Output-2:

----------------

1

Explanation:

------------

There is only 1 good binary string in this example: "1111".

It can be proven that there is only 1 good string satisfying all conditions.

Only 62.5% test cases passed don’t know why didn’t even understood the logic

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int minl=sc.nextInt();

int maxl=sc.nextInt();

int oneg=sc.nextInt();

int zg=sc.nextInt();

int[] dp=new int[maxl+1];

dp[0]=1;

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

if(i-oneg<0 && i-zg<0){

dp[i]=0;

}

else if(i-oneg<0 ){

dp[i]=dp[i-zg];

}

else if(i-zg<0){

dp[i]=dp[i-oneg];

}

else{

dp[i]=dp[i-oneg]+dp[i-zg];

}

}

// System.out.println(Arrays.toString(dp));

int sum=0;

for(int i=minl;i<=maxl;i++){

sum+=dp[i];

}

System.out.println(sum);

}

}

Given a matrix of 0's and 1's of dimension rows X cols.

We could traverse from one element(row, col) to any other element(row + 1, col) or (row, col + 1).

Print 'true' if there is any path from first element(0,0) of the matrix to the last element(rows - 1, cols - 1)

of the matrix that visits the same number of 1's and 0's.

Print 'false' otherwise.

Input Format

-------------

Line-1: Read two integers rows and cols(space separated).

Line-2: Read the matrix of dimension rows X cols.

Output Format

--------------

If the required path exist print 'true' otherwise 'false'.

Sample input-1:

---------------

3 4

0 1 1 0

0 0 0 0

0 0 1 0

Sample output-1:

----------------

true

Explanation: In the given matrix there is a path that is having three 0's and three 1's.

Sample input-2:

---------------

4 3

0 0 1

1 0 0

0 0 0

0 0 1

Sample output-2:

----------------

false

Explanation: There is NO path that is having same number of 0's and 1's.

// import java.util.\*;

// public class Main{

// static boolean flag=false;

// public static void main(String[] args){

// Scanner sc=new Scanner(System.in);

// int r=sc.nextInt();

// int c=sc.nextInt();

// int[][] arr=new int[r][c];

// for(int i=0;i<r;i++){

// for(int j=0;j<c;j++){

// arr[i][j]=sc.nextInt();

// }

// }

// boolean[][] vis=new boolean[r][c];

// vis[0][0]=true;

// if(arr[0][0]==0){

// dfs(arr,vis,0,0,0,1,r-1,c-1);

// }

// else{

// dfs(arr,vis,0,0,1,0,r-1,c-1);

// }

// System.out.println(flag);

// }

// static boolean dfs(int[][] arr,boolean[][] vis,int r,int c,int co,int cz,int destr,int destc){

// if(r==destr && c==destc){

// if(cz==co){

// // flag=true;

// // }

// return true;

// } }

// vis[r][c]=true;

// if( r+1<destr && !vis[r+1][c]){

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

// if(arr[r+1][c]==0){

// dfs(arr,vis,r+1,c,co,cz+1,destr,destc);

// }

// else{

// dfs(arr,vis,r+1,c,co+1,cz,destr,destc);

// }

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

// }

// if( c+1<destc && !vis[r][c+1]){

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

// if(arr[r][c+1]==0){

// dfs(arr,vis,r,c+1,co,cz+1,destr,destc);

// }

// else{

// dfs(arr,vis,r,c+1,co+1,cz,destr,destc);

// }

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

// }

// vis[r][c]=false;

// return false;

// }

// }

import java.util.\*;

public class Main

{

public static boolean dfs(int x, int y, int[][] grid, int[][] visited, int zeros, int ones)

{

int rows = grid.length;

int columns = grid[0].length;

if (x < 0 || x >= rows || y < 0 || y >= columns || visited[x][y] == 1)

{

return false;

}

visited[x][y] = 1;

if (grid[x][y] == 0)

{

zeros++;

}

else

{

ones++;

}

if (x == rows - 1 && y == columns - 1 && ones == zeros)

{

return true;

}

if (dfs(x + 1, y, grid, visited, zeros, ones) || dfs(x, y + 1, grid, visited, zeros, ones))

{

return true;

}

visited[x][y] = 0;

return false;

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int m = sc.nextInt();

int[][] grid = new int[n][m];

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

{

for (int j = 0; j < m; j++){

grid[i][j] = sc.nextInt();

}

}

int[][] visited = new int[n][m];

boolean res = dfs(0, 0, grid, visited, 0, 0);

System.out.println(res);

}

}