'''

Prabhath is working on words. He used to take out every letter that was repeated

in the word.

A word W is given to Prabhath. His objective is to eliminate every duplicate

letter from W such that the resultant word R contains every unique letter from W

and did not contain any duplicate letters.

And R should be at the top of the dictionary order.

NOTE:

-----

He is not allowed to shuffle the relative order of the letters of the word W to

form the word R.

Input Format:

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

A String, the word W.

Output Format:

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

Print a String, resultant word R.

Sample Input-1:

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

cbadccb

Sample Output-1:

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

adcb

Sample Input-2:

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

silicosis

Sample Output-2:

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

ilcos

'''

s=input()

st=[]

for i in range(0,len(s)):

if(len(st)==0):

st.append(s[i])

if (s[i] not in st):

while st and (s[i]<st[-1] and (st[-1] in s[i+1:len(s)])):

st.pop()

st.append(s[i])

# elif(s[i]>st[-1] and( s[i] not in s[i+1:len(s)])):

# st.append(s[i])

print("".join(st))

# s1=[]

# for i in s:

# s1.append(i)

# d=dict()

# for i in s:

# if i not in d:

# l=[]

# l.append(s1.index(i))

# d[i]=l

# else:

# d[i].append(s1.index(i))

# s1[s1.index(i)]=""

# print(d)

# ans=""

# for i in d.values():

# if(len(i)==1):

# ans+=s[i[0]]

# print(ans)

# for i in d.values():

# if(len(i)==1):

# continue

# l=[]

# for j in i:

# l.append(ans[0:j]+s[j]+ans[j:len(ans)])

# print(l)

# l.sort()

# print(l)

# ans=l[0]

# print(ans)

A game show hosted by the famous anchor Suma in which the celebrities compete

with each other. The game show format is changed a bit, There are N celebrities

participated now. And all of them answered correctly for a given question.

Now everyone got a chance to play the money board game.

Where i-th player standing at i-th index position of the top,

and value of 'i' is 0 <= i < n.

The money board is of size M\*N, and N players has coins to drop from

top of the money board. The money board is opened at top and bottom.

Each cell in the board has a diagonal bar connecting two corners of the cell,

that can redirect the coin to the rightside or to the leftside.

- The bar that redirects the coin to the right,

connected from the top-left corner to the bottom-right corner, indicated by 1.

- The bar that redirects the coin to the left,

connected from the top-right corner to the bottom-left corner, indicated by -1.

The players drop their coins at the top of board. Each coin can get stuck

in the board or fall out of the bottom. A coin gets stuck if it hits a "V"

shaped pattern between two bars or if a bar redirects the coin into

either wall of the board.

Your task is to find and return an array result[] of size N,

where result[i] is the index that the coin falls out of at the bottom

after dropping the coin from the i-th index at the top,

or -1 if the coin gets stuck in the board.

Input Format:

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

Line-1: Two space separated integer M and N, size of the board.

Next M lines: N space separated integers

Output Format:

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

Print an array result.

Sample Input-1:

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

5 5

1 1 1 -1 -1

-1 -1 -1 1 -1

1 1 1 1 -1

-1 -1 -1 -1 1

-1 -1 -1 1 -1

Sample Output-1:

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

[-1, 0, -1, -1, -1]

Sample Input-2:

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

5 5

1 1 1 1 1

-1 -1 -1 -1 -1

1 1 1 1 1

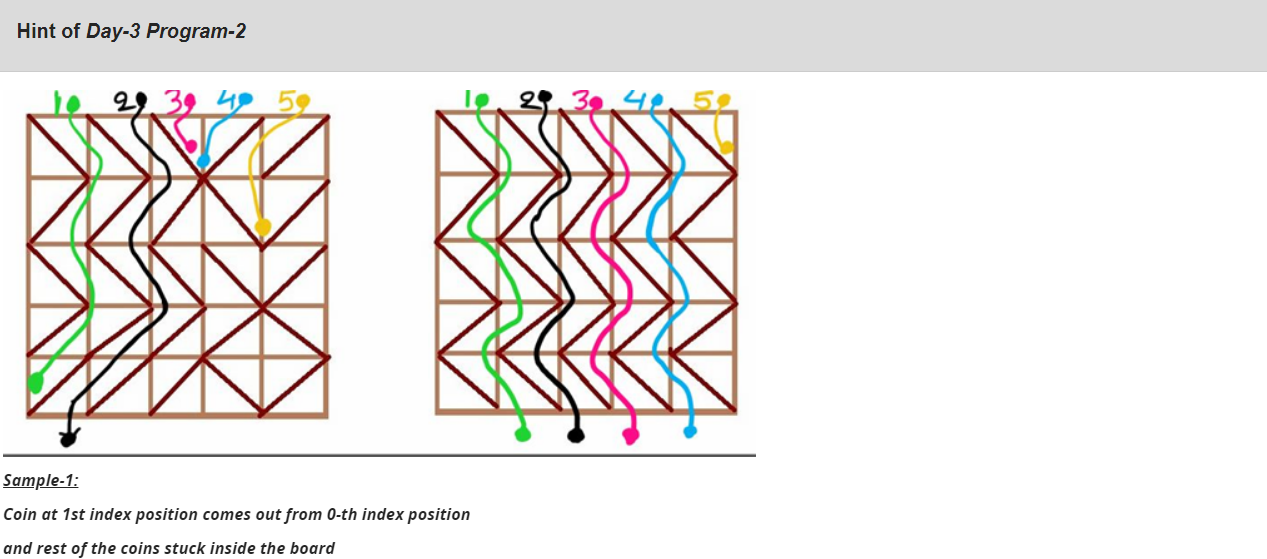
-1 -1 -1 -1 -1

1 1 1 1 1

Sample Output-2:

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

[1, 2, 3, 4, -1]

  
import java.util.\*;

public class Main{

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();

}

}

int[] res=new int[c];

// bool

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

res[i]=dfs(arr,0,i);

}

System.out.println(Arrays.toString(res));

}

public static int dfs(int[][] arr,int r,int c){

if(c+1<arr[0].length){

if(arr[r][c]==1 && arr[r][c+1]==-1){

return -1;

}

}

if(c-1>=0 ){

if(arr[r][c]==-1 && arr[r][c-1]==1){

return -1;

}

}

if(r==arr.length-1){

if(c==0 && arr[r][c]==-1){

return -1;

}

if(c==arr[0].length-1 && arr[r][c]==1){

return -1;

}

if(arr[r][c]==1){

return c+1;

}

else{

return c-1;

}

}

if((arr[r][c]==-1 && c==0)||(arr[r][c]==1 && c+1==arr[0].length)){

return -1;

}

if(r+1<arr.length){

// return dfs(arr,r+1,c);

if(arr[r][c]==1){

return dfs(arr,r+1,c+1);

}

else{

return dfs(arr,r+1,c-1);

}

}

return -1;

}

}

There are N computers in a network, all the computers are connected as tree

structure. And one new connection is added in the Network. The computers in

the network are identified with their IDs, the IDs are numbered between 1 to N.

The connections in the network is given as coonection[i] = [comp-A, comp-B],

there is a connection between comp-A and comp-B.

Your task is to remove a connection in the network and print it, so that

all the computers are connected as tree structure. If there are multiple

options to remove, remove the connection that occurs last in the input.

Input Format:

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

Line-1: Two space separated integers N, number of computers.

Next N lines: Two space separated integers, comp-A & comp-B.

Output Format:

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

Print the connection which is removed.

Sample Input-1:

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

6

1 2

3 4

3 6

4 5

5 6

2 3

Sample Output-1:

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

5 6

Sample Input-2:

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

4

1 2

2 3

3 4

2 4

Sample Output-2:

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

2 4

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[][] arr=new int[n][2];

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

arr[i][0]=sc.nextInt();

arr[i][1]=sc.nextInt();

}

int[] parents=new int[n+1];

int a1=-1;

int a2=-1;

for(int i=0;i<n+1;i++){

parents[i]=i;

}

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

if(find(parents,arr[i][0])==find(parents,arr[i][1])){

a1=arr[i][0];

a2=arr[i][1];

}

union(parents,arr[i][0],arr[i][1]);

}

System.out.println(a1+" "+a2);

}

public static void union(int[] arr,int x,int y){

int p1=find(arr,x);

int p2=find(arr,y);

arr[p1]=p2;

}

public static int find(int [] arr,int x){

if(arr[x]==x){

return x;

}

return find(arr,arr[x]);

}

}