'''

In an international cricket stadium before starting of the game

players stand in a row with tshirt numbers 0's and 1's randomly.

But the players with tshirt-number 1 wants to stand together.

Your task is to print the least number of interchages of places to group all

players with tshirt number 1.

Input Format:

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

Line-1: Read an integer R, number of players.

Line-2: Read R tshirt-numbers standing in a row.

Output Format:

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

Print an integer result.

Sample Input-1

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

5

1 0 1 0 1

Sample Output-1

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

1

Explanation

-----------

There are 3 ways to group all 1's together

1 1 1 0 0 after 1 interchange

0 1 1 1 0 after 2 interchanges

0 0 1 1 1 after 1 interchange

The least is 1.

Sample Input-2

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

7

1 0 1 0 1 0 1

Sample Output-2

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

2

Explanation

-----------

There are 4 ways to group all 1's together

1 1 1 1 0 0 0 after 2 interchanges

0 0 0 1 1 1 1 after 2 interchanges

0 1 1 1 1 0 0 after 2 interchanges

0 0 1 1 1 1 0 after 2 interchanges

The minimum is 2.

Sample Input-3

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

6

0 1 0 0 0 0

Sample Output-3

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

0

Explanation

-----------

No interchange required.

'''

n=int(input())

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

count=0

for i in l:

if(i==1):

count+=1

min=len(l)

for i in range(0,len(l)-count+1):

cn=0

for j in range(i,i+count):

if(l[j]==0):

cn+=1

if(cn<min):

min=cn

print(min)

AlphaCipher is a string formed from another string by rearranging its letters

You are given a string S.

Your task is to check, can any one of the AlphaCipher is a palindrome or not.

Input Format:

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

A string S

Output Format:

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

Print a boolean value.

Sample Input-1:

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

carrace

Sample Output-1:

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

true

Sample Input-2:

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

code

Sample Output-2:

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

false

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.next();

int res=0;

// int bit=0;

for(char i:s.toCharArray()){

int mask=1<<(i-'a');

res^=mask;

// System.out.println(res);

}

System.out.println((res & (res-1))==0);

// if(res==0 || res>='a' && res<='z'|| res>='A' && res<='Z'){

// System.out.println(true);

// }

// else{

// System.out.println(false);

// }

}

}

Given an array of integer elements, ele[], YOur task is to find and print

the number of the subarrays, which gives you an even value after multiplication

of the elements in it.

Input Format:

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

Line-1: An integer N, number of elements.

Line-2: Space separated elements.

Output Format:

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

Print number of possible subarrays.

Sample Input-1:

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

4

5 8 9 17

Sample Output-1:

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

6

Explanation:

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

5 x 8 = 40

5 x 8 x 9 = 360

5 x 8 x 9 x 17 = 6120

8 = 8

8 x 9 = 72

8 x 9 x 17 = 1224

Hence we got 6 even subarray multiplications.

Sample Input-2:

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

3

5 7 11

Sample Output-2:

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

0

Explanation:

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

No subarrays which will give you even value after multiplication.

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int oc=0;

int ec=0;

int[] arr=new int[n];

int start=0;

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

arr[i]=sc.nextInt();

if(arr[i]%2==0 ){

int c=i-start;

oc+=(((c)\*((c)+1)))/2;

start=i+1;

}

}

if(start!=n){

int c=n-start;

oc+=(((c)\*((c)+1)))/2;

}

// ec=n-oc;

int ans=((n\*(n+1))/2)-oc;

System.out.println(ans);

}

}

There are n cartons of binary string of lenght n, where carton[i] is '0' if the ith

carton is empty and '1' if it contains one product.

In one operation, you can move one product from a carton to an adjacent carton.

cartoon i is adjacent to carton j if abs(i - j) == 1. Note that after doing so, there

may be more than one product in some cartons.

Return an array answer of size n, where answer[i] is the minimum number of operations

needed to move all the products to the ith carton

Each answer[i] is calculated considering the initial state of the cartons.

Input Format

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

A string of zeroes and ones

Output Format

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

An integer array which is minimum no of operations

Sample Input-1:

===============

110

Sample Output-1:

================

1 1 3

Explanation:

============

The answer for each cartoon is as follows:

1) First carton: you will have to move one product from the second carton to the first

carton in one operation.

2) Second carton: you will have to move one product from the first carton to the second

carton in one operation.

3) Third carton: you will have to move one product from the first carton to the third carton

in two operations, and move one product from the second product to the third carton in

one operation.

Sample Input-2:

===============

001011

Sample Output-2:

================

11 8 5 4 3 4

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.next();

ArrayList<Integer> l=new ArrayList<>();

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

if(s.charAt(i)=='1'){

l.add(i);

}

}

int[] arr=new int[s.length()];

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

// if(l.contains(i)){

// arr[i]=1;

// }

// else{

// arr[i]=0;

// }

// }

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

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

for(int j=0;j<l.size();j++){

if(i!=j)

// arr[i]+=l.get(j)+(Math.abs(j-1));

System.out.print("");

arr[i]+=Math.abs(l.get(j)-i);

}

// arr[i]+=Integer.parseInt(String.valueOf(s.charAt(i)));

}

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

}

}