Bizon the Champion isn't just attentive, he also is very hardworking.

Bizon the Champion decided to paint his old fence his favorite color, orange. The fence is represented as *n* vertical planks, put in a row. Adjacent planks have no gap between them. The planks are numbered from the left to the right starting from one, the *i*-th plank has the width of 1 meter and the height of *ai* meters.

Bizon the Champion bought a brush in the shop, the brush's width is 1 meter. He can make vertical and horizontal strokes with the brush. During a stroke the brush's full surface must touch the fence at all the time (see the samples for the better understanding). What minimum number of strokes should Bizon the Champion do to fully paint the fence? Note that you are allowed to paint the same area of the fence multiple times.

Input

The first line contains integer *n* (1 ≤ *n* ≤ 5000) — the number of fence planks. The second line contains *n* space-separated integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 109).

Output

Print a single integer — the minimum number of strokes needed to paint the whole fence.

Example

Input

5  
2 2 1 2 1

Output

3

Input

2  
2 2

Output

2

Input

1  
5

Output

1

Note

In the first sample you need to paint the fence in three strokes with the brush: the first stroke goes on height 1 horizontally along all the planks. The second stroke goes on height 2 horizontally and paints the first and second planks and the third stroke (it can be horizontal and vertical) finishes painting the fourth plank.

In the second sample you can paint the fence with two strokes, either two horizontal or two vertical strokes.

In the third sample there is only one plank that can be painted using a single vertical stroke.

• 每块木板宽度均为1，高度为h[i]

• n块木板连接为宽度为n的栅栏

• 每次可以刷一横或一竖(上色)

• 最少刷多少次可以使得栅栏被全部上色

• 1 ≤ n ≤ 5000

首先，我们假如每次都刷一个木板，即一竖行，那么需要n次刷完，可见这是一个ans的最大值。就是最差的情况下我这样刷最多为n刷。

其次：如果我们选择一横行的刷，而n个木板中最短的为min，那么我们可以花min刷，把他们都刷成a [ i ] - min的高度，那么剩下来的栅栏又变成了开始的情况，我们可以在选择前面不为0的x个，继续按上面的方法刷，可见是一个递归调用即可。

要特别注意的是前面的条件，就是刷x个木板，最多用x刷，如果某一次求得大于x，那么取x，这样就很easy了。

其实可以归结为：其实最初的思想可以归结为优先横刷，其次竖刷（如果竖刷花费更小），注意一定要仔细。

#include<iostream>

#include<stdio.h>

using namespace std;

int h[5010];

int solve(int l,int r)

{

int len=r-l+1;

int minn=h[l];

for(int i=l;i<=r;i++)

minn=min(minn,h[i]);

for(int i=l;i<=r;i++)

h[i]-=minn;

int ans=minn;

for(int i=l;i<=r;i++)

if(h[i]>0)

{

int k=i+1;

while(h[k]>0&&k<=r) //找连续不为0的一段

k++;

ans+=solve(i,k-1);

}

if(ans>len)//如果还不如一列一列的涂

ans=len;

return ans;

}

int main()

{

int n;

cin>>n;

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

cin>>h[i];

cout<<solve(1,n);

return 0;

}