/\*

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

For example,

Given [0,1,0,2,1,0,1,3,2,1,2,1], return 6.

For example,

Given [0,1,0,2,1,0,1,3,2,1,2,1], return 6.

^

3| ■ □: water

2| ■ □ □ □ ■ ■ □ ■ ■: elevation map

1| ■ □ ■ ■ □ ■ ■ ■ ■ ■ ■

————————————————————————>

The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

题目的意思就是：每一个数字代表一个树立的木块，合起来可以盛多少水

方法：递归。

1、找最高的那个，下标为i

2、i的左右分别找最高

3、往左右迭代

\*/

class Solution {

public:

int findleft(vector<int>& height,int flag,int sum)

{

int ff;

int max=0;

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

{

if(height[i]>max)

{

max=height[i];

ff=i;

}

}

if(max==0)

return sum;

for(int i=ff+1;i<flag;i++)

sum+=height[ff]-height[i];

sum=findleft(height,ff,sum);

return sum;

}

int findright(vector<int>& height,int flag,int sum)

{

int ff;

int max=0;

for(int i=flag+1;i<height.size();i++)

{

if(height[i]>max)

{

max=height[i];

ff=i;

}

}

if(max==0)

return sum;

for(int i=flag+1;i<ff;i++)

sum=sum+height[ff]-height[i];

sum=findright(height,ff,sum);

return sum;

}

int trap(vector<int>& height)

{

//way-1

/\*

int max=0;

int flag=-1;

for(int i=0;i<height.size();i++)

{

if(height[i]>max)

{

max=height[i];

flag=i;

}

}

int sum=0;

sum=findleft(height,flag,sum);

sum=findright(height,flag,sum);

return sum;

\*/

//way-2

int n=height.size();

int l=0,r=n-1,res=0,minh;

while(l<r)

{

minh=min(height[l],height[r]);

if(height[l]==minh)

{

while(++l<r && height[l]<minh)

{

res+=minh-height[l];

}

}

else

{

while(l<--r && height[r]<minh)

{

res+=minh-height[r];

}

}

}

return res;

}

};