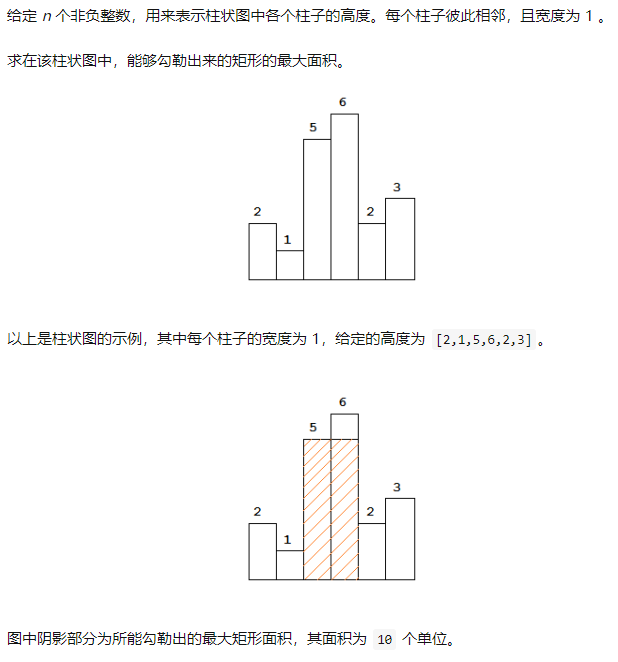
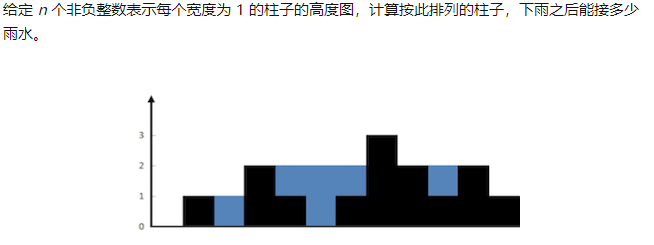
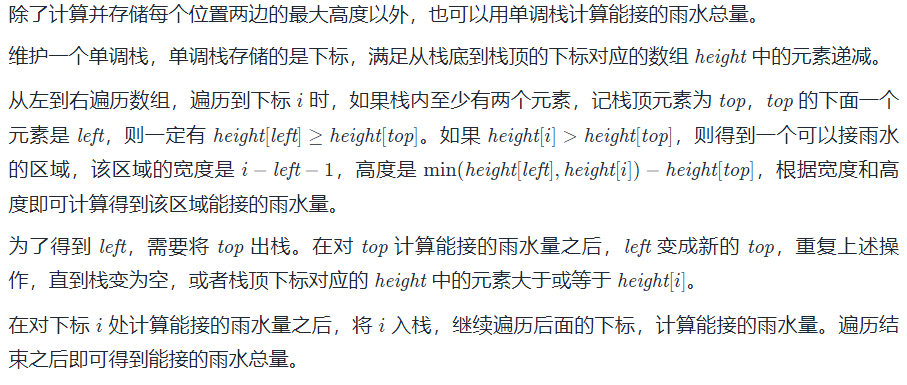
单调栈的原理，就是以某一个值为最小（最大）值，向这个值的两侧延伸，遇到大于它（小于它）的值，就将它延伸的范围扩大，当然，一般来说，要这样做的算法复杂度为o(n^2)，但是借助栈这个玩意，维护其单调增（减），就可以在o(n)的时间复杂度解决这个问题。



n<=10^6

思路：当前的柱子可以向两边拓展，遇到比自己大的就接着拓展，小的就停止，然后用自己的高度乘以拓展的宽度，每次都更新最大面积，时间复杂度同样为O(N^2)，所以我们接着借助单调栈，维护单调递增栈，对于每个柱子分两次向两边扩散，比方说向右，遇到比栈顶高的就入栈，比栈顶低的就弹出，弹出的所有值，对于他们来说，这个比他们低的值就是他们能扩展到的最右边的值，向左也是一样，最后我们就能得到每个柱子能扩展到的最左和最右，就可以求出每个柱子的位置的最大面积，然后求最大值就好了





POJ2796 区间最大参考值

给定一个数组，定义某个区间的参考值为：区间所有元素的和\*区间最小元素。求该数组中的最大参考值以及对应的区间。

算出每个数往左或者往右多少之类都是它作为最小值，比方说向右，建立一个单调递增的栈，

每次元素进入栈就pop掉顶上的值，每个进来的元素标记其能扩展的最左的点（具体见代码）。向左同理，最后用每个点乘以其最左的点到最右的点之间的区间和，区间和用前缀和求出即可

#include<iostream>

#include<cstdio>

#define ll long long

#define inf 1e18

#define mn 200005

using namespace std;

ll h[mn],l[mn],r[mn],s[mn],sum[mn],

top,ans,n,m;

int main()

{

ll te,x,y,z,i,j,k;

char ch;

while(cin>>n)

{

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

{

scanf("%lld",&h[i]);

sum[i]=h[i]+sum[i-1];

}

ll top=0;

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

{

while(top&&h[s[top]]>=h[i])top--;//此时为计算区间最小值，最大值要变号

l[i]=top?s[top]+1:1;

s[++top]=i;

}

top=0;

for(i=n;i>=1;i--)

{

while(top&&h[s[top]]>=h[i])top--;//此时为计算区间最小值，最大值要变号

r[i]=top?s[top]-1:n;

s[++top]=i;

}

ans=0;

ll lx=1,rx=1;

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

{

ll tmp=h[i]\*(sum[r[i]]-sum[l[i]-1]);

if(tmp>ans)

{

ans=tmp;

lx=l[i];

rx=r[i];

}

}

cout<<ans<<endl;

cout<<lx<<" "<<rx<<endl<<endl;

}

return 0;

}