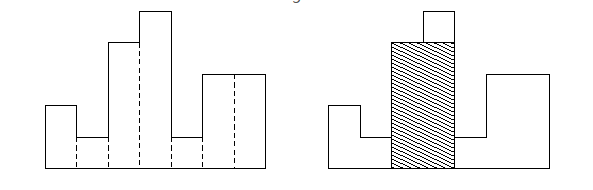
# 单调栈



给你一个这样的图像，让你求最大的矩形。

这题的策略是寻找某一高度作为最终高度的左右扩展范围  
对于i这个范围是L[i]和R[i]   
L[i]是从i往左找，找到的第一个比i高度小的位置+1  
R[i]是从i往右找，找到的第一个比i高度小的位置-1  
最后循环遍历一下，找到最大的  i的高度\*(R[i]-L[i]+1)就是结果  
这个方法是能解决问题的，但是暴力的做的话复杂度比较的高  
那么怎么办  
单调栈是一个实现的话就是一个普通的栈，只不过我们为了满足单调栈的性质添加了一些操作  
栈是后进先出的，单调性指的是严格的递增或者递减。  
单调栈有一个重要的性质：  
1、若是单调递增栈，则从栈顶到栈底的元素是严格递增的。若是单调递减栈，则从栈顶到栈底的元素是严格递减的。  
这个栈是通过人工的添加策略来实现的，并不是STL有这么一个栈，只是我们维护一个栈，他符合上述性质  
  
首先从左往右找（使用递减栈）  
那么对于这题，每次添加一个高度的时候，我们看栈顶的元素是不是比它大了，大了就pop，知道栈顶的元素是比它小的。  
然后把这个点的高度放进去，这个高度的L[i]就是栈里的那个高度所在的位置+1了。  
  
画个例子

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 直方图 | 2 | 1 | 4 | 5 | 1 | 3 | 3 |
| L[i] | 1 | 1 | 3 | 4 | 1 | 6 | 6 |
| R[i] | 1 | 7 | 4 | 4 | 7 | 7 | 7 |

注意直方图最左边的和最右边都增加一个高度无穷小的高度，用来区别  
如果你用单调栈推出了上述数据，ok，你就懂了

**while**(qq.top().val>=arry[i])qq.pop();