Assignment 2 Report

直方图中最大的矩形面积问题

曹旭 大数据学院 16307110230

一、题目描述与分析

题目给出 n 个非负整数代表直方图每个矩形条的高度，其中每个条形的宽度为 1，找到直方图中最大矩形的区域。

该问题可以用数组存储每个条形的高度，然后通过分析各个条形之间的关系找出计算方法。假如不考虑时间复杂度的话，很容易就能想到$O(n^2)$的常规解决思路，但是我测试了$O(n^2)$的算法后，发现从第四个到第十个样例会报错，说明改题目不能单纯用暴力法解决。于是我联想到了以前看过的水容器计算的算法，打算使用栈来解决这个问题。

二、使用栈的解决方法

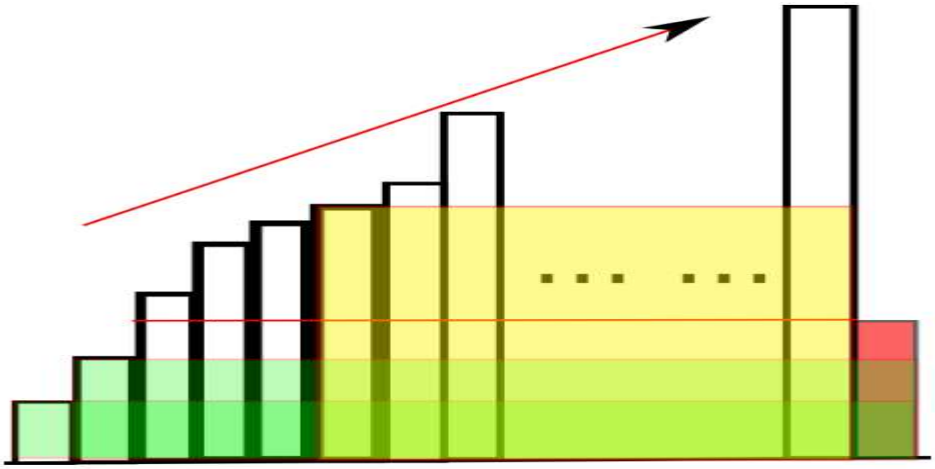

图 1 使用栈的方法图解

我们假设所有的条形都存在数组 a[n]中，首先从 a[0]开始遍历整个数组，先将 a[0]压入栈中，栈存储对应条形的高度和位置。自 a[0]开始，找出所有的后一个比前一个高于或等于的条形，将其压入栈中，直到遇到一个高度下降的条形 a[k]。然后开始从栈顶逐个 pop 出，同时计算矩形的面积，计算完的面积和最大面积比较，如果大于最大面积就更新最大面积，直到栈顶元素的高度小于等于 a[k]为止，继续从 a[k]往后遍历数组，重复这一过程。从这里我们可以发现我们维护了一个递增的堆栈，栈内元素一定要比当前位置 a[i]的元素小。

下面举一个例子来说明一些细节问题：

假设输入 5 个条形高度，input: 8 9 8 7 6

output:30

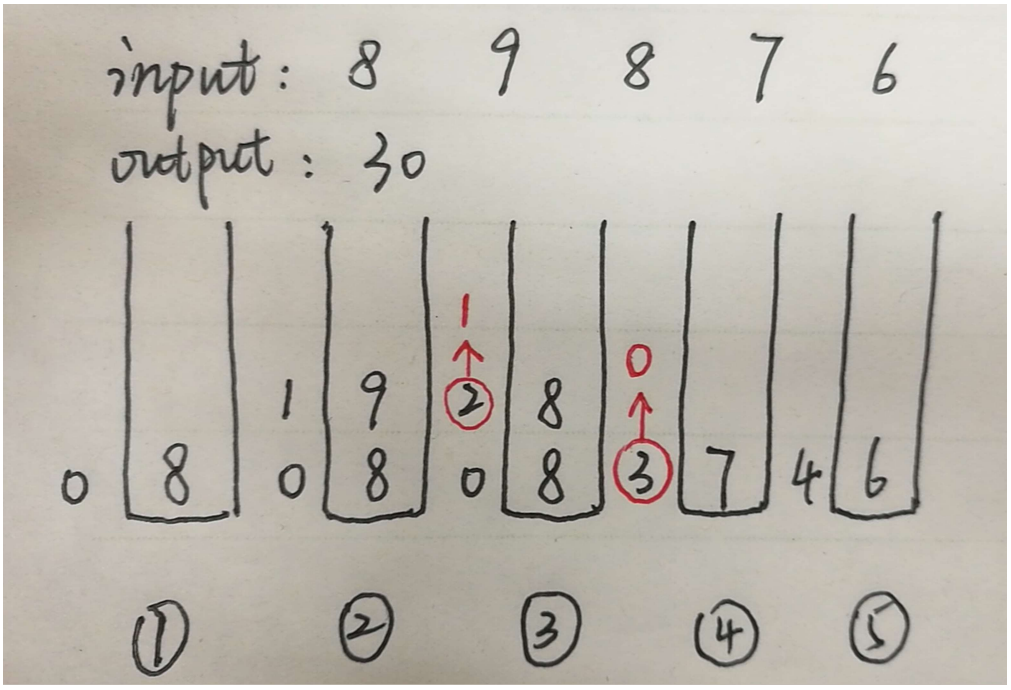

图 2 示例

从 a[0]==8 开始，因为 9 比 8 大，将 8、9 逐个压入栈中，因为 a[2]==8，所以 9 出栈，此时计算最大面积为 9*(2-1)=9。然后 8 入栈，a[3]==7，两个 8 依次出栈，计算最大面积为 8*(3-0)=24。7 入栈，a[4]==6,7 出栈，计算最大面积为 7*(4-0)=28，这里。6 入栈，全部 push 出栈，计算得到最大面积为 6*(5-0)=30。

关于边缘情况，也就是数组最后一个元素 a[n-1]如何进栈的问题，需要先判断最后一个元素是否高于前一个元素，倘若高于，则 push 进栈，然后全部 pop 出，计算面积。如果小于前一个元素，则先 pop 最后一个元素前的元素，直到符合“栈内元素一定要比当前位置 a[i]的元素小”这一条件后将 a[n-1]push 进栈，然后再 pop。

这样处理下来稳健性很好，在处理复杂的柱形图（增减不一）时该程序能够快速给出正确的解答，如下图:

该方法的平均时间复杂度和最坏时间复杂度均为O(n)，最坏空间复杂度为O(n)。

最后，在写完这部分代码和报告后我又想到了也许也可以用分治法来解决这个问题，在每个条形图形成的山峰的位置一分为二迭代求解，但这个方法的时间复杂度肯定大于O(n)，因此栈依旧是这一问题最佳的解决方案。