# God Knows题解

crazy_cloud

October 9, 2017

我看这题题解写得比较简略，就来做点补充。

# 1 Solution

首先考虑最普通的$dp$，设$f_i$表示最后一个选的是$(i, p_i)$的最优答案。我们计算$f_i$的时候，需要枚举位置$j$ 使得两条边不相交而且$X$部在$[j+1, i-1]$之间的所有点都和$(j, p_j)$或者$(i, p_i)$之一相交。

将$(i, p_i)$看成二维平面上的一个点，那么就是要求$p_j < p_i$且两点作为角形成的矩形内部没有其他点。

考虑这些$j$实际上是什么？如果我们将其按照$p_j$排序，可以发现它们的值一定是单调递减的。更一般地，它们就是将$[1, i-1]$所有$p_j$小于$p_i$的$j$拉出来以$p_j$为关键字排序之后，按照$j$的大小形成的单调递减的单调栈。

如果我们以$p$的值作为下标，那么问题就变成要你支持修改一个下标上的数（将$p_i$上的数改为$i$），以及查询一个区间（$[1, p_i]$）所有数形成的单调栈$stack$中，$f_x(x \in stack)$的最小值。

这是一个很经典的问题（[WC2013]楼房重建），类似的还有维护区间单调栈的所有数的和之类的。考虑到可能有的同学没见过这个模型，在这里详细讲一下：

先考虑如何查询，定义函数$query([l, r], p)$表示查询时间区间$[l, r]$组成的单调栈在最后加入$p$之后最小的$f_x$，查询时显然$p = 0$调用一下就好了。

如果$l = r$，那么我们可以直接计算答案。设$mid = \left\lfloor \frac{l+r}{2} \right\rfloor$，$rmx$表示区间$[mid+1, r]$中所有数的最大值。

如果$rmx > p$，那么显然$p$不会对左半区间产生任何影响，而右边的最大值会加入到左边区间的单调栈中。

这样答案等于$\min(query([l, mid], rmx), query([mid+1, r], p))$。

否则，右边的单调栈会直接被$p$全部弹出，答案等于$query([l, mid], p)$。

如果我们一直这样递归下去，复杂度是没有保证的，怎么办呢？

令$fmin_x$表示$x$这个节点右半边区间最大值加入到左半边的单调栈之后，单调栈的最小$f_x$，即$query(left.range, right.max)$。先不考虑这个如何维护，反正我们每次修改之后线段树上每个位置的这个值都要修改过来。

假如我们的$query$操作中的区间已经是一个完整的线段树区间了，那么我们的$rmx$显然可以$O(1)$得到，$query([l, mid], rmx)$其实就是$fmin_x$，也可以$O(1)$得到。

那么这个$query$的时间复杂度如何分析呢？我们研究里面的嵌套函数调用了多少次。

首先，在我的区间还不是一个完整的线段树区间时，$rmx$需要通过

调用区间最小值来得到，调用一次的复杂度是$O(\log n)$的，而调用次数也是$O(\log n)$的，因此这个复杂度是$O(\log^2 n)$的。

其次，我们的查询操作会下放到$O(\log n)$个完整的线段树区间，观察发现接下来我们往下走的过程已经不会增加新的分支，都是一条路走到底，因此每个区间最多会向下走$O(\log n)$步，这个时间复杂度也是$O(\log^2 n)$的。

至于$fmin_x$怎么在每次修改之后重新得到？直接在$update$时调用一下$query$就好了。每次修改会调用$O(\log n)$次$update$，由于这时我的$query$都是在线段树上的完整区间上查，因此每次复杂度是$O(\log n)$ 的。

于是总的时间复杂度就是$O(n \log^2 n)$的。