NOI练习赛 解题报告

276f280b19d8e63bd6f992d88bbf96b2

1 一氯取代

1.1 标准算法

首先以某个点为根建立有根树，考虑以x为根的树产生的一氯取代物种类数，就是所有不同子树的种类数求和。判断子树是否相同可以用括号序列最小表示法。

然而这样做是有问题的，因为如果根是在某个同构位置上就会答案出错。所以选择一个不会产生同构的位置（比如树的中心）作为根就可以了。

2 不开心

在以下的复杂度分析当中，由于q 与n 同阶，故所有的q 都被换成了n。

2.1 测试点1 和3

树是一条链。

注意，一条链的情况当中，由于约定了节点的父亲编号一定小于自己的编号，

所以节点i 的父亲一定是$i-1$，不需要再进行搜索。

这样，就变成了经典的单点修改，区间求和问题。可以用线段树、树状数组等数据结构轻松解决。

时间复杂度$O(n\log n)$，空间复杂度$O(n)$。

2.2 测试点2

树是随机生成的，且每次询问都是询问除了子树的根的整个子树。

可以直接处理出答案。对于每次修改，将这个节点到根的路径（除它本身以外）上的每个节点的答案加上修改值，对于每次询问直接输出答案。由于树是随机生成的，树的深度是$O(\log n)$的。

时间复杂度$O(n\log n)$，空间复杂度$O(n)$。

2.3 测试点4

每次询问都是询问除了子树的根的整个子树。

于是变成了类似poj3321 Apple Tree 的经典的树上询问问题。

对树求出其dfs 序，设以节点i 为根的子树的dfs 序区间为$[l_i, r_i]$，则原问题转化为：修改：修改a_{l_i}；查询：求$a_{l_i+1}, \cdots, a_{r_i}$的区间和。可以用线段树、树状数组等数据结构轻松解决。

时间复杂度$O(n\log n)$，空间复杂度$O(n)$。

2.4 测试点5 和6

树是随机生成的。

有至少三种思路可以解决。

第一种是直接统计答案的思路。在每个节点上维护一棵线段树，对于每次修改还是从节点往根走，设走了i 步，那么就应该在当前节点的线段

树的$[i,+\infty)$区间加上答案。对于每次询问，直接在线段树上查询。

线段树需要动态开节点。时空复杂度$O(n\log^2 n)$，很可能会MLE。

第二种是利用dfs 序的思路。对于每次修改，直接修改$a[dep_i][l_i]$；对于每次查询，设这次查询的是x 到y 级下属，即是求$a[dep_i+x\cdots dep_i+y][l_i\cdots r_i]$的区间和。由于$dep_i$是$O(\log n)$的，所以我们只在第二维开线段树维护，而第一维用暴力解决。

时间复杂度$O(n\log^2 n)$，空间复杂度$O(n\log n)$。

如果第一维也用线段树维护，渐进意义上时间复杂度会变为$O(n\log n\log\log n)$，但是实际效果不会很好。

第三种是利用bfs 序的思路。对于每次修改，直接修改其在bfs 序列上的位置；对于每次查询，易得每个子树的每一层都是bfs 序列上的连续一段，则对于每一层都查询一段，最后合并答案。由于树是随机生成的，层数是$O(\log n)$的。时间复杂度$O(n\log^2 n)$，空间复杂度$O(n)$。

2.5 测试点7 ～10

没有其他特征。

利用刚才关于dfs 序的思路，我们把原问题转化成了这样一个问题：给出平面上n 个点，点上有权，每次可以修改一个点的权值，或者询问一个矩形区域（与坐标轴平行）内的点权之和。

最裸的方法是用二维线段树直接维护，需要动态开节点，时空复杂度$O(n\log^2 n)$，很可能会MLE。二维树状数组的做法同理。

一种不会MLE 的方法是先套一层分治来，然后就变成了经典问题，相信会分治的同学们都会这个经典方法。时间复杂度$O(n\log^2 n)$，空间复杂度$O(n)$。

但是对于只会线段树，而从未听说过这种分治方法的同学们，这道题也有一些别的方法。

四分树！

没错，就是那个我们第一次遇到二维区间问题第一个想到的那个东西，

后来因为它各种退化被我们抛弃的那个几乎跟OI 中的数据结构扯不上关系的四分树！

像线段树一样，四分树每次把当前的矩形区间分成大致等大的4 个子矩形，然后递归处理。单点修改复杂度$O(\log n)$，区间求和复杂度……很遗憾，是$O(n)$。

那么我们不让他退化！我们的四分树是动态开节点的，而且节点十分稀疏，如果想要往里走的时候，节点还没有被开出来，就不往里面走。

对于链上的操作，可以证明其复杂度为$O(\log n)$。然而，对于其他数据，尤其是当树是随机生成的时候，同一深度的节点会非常多，会造成很大的退化现象。

那么我们不让他退化！对于深度相同的节点，我们人为地将其错开，就能大大减少退化现象，但是仍然能有特定数据针对特定代码能够构造出退化情况。那么我们在错开深度相同的节点时加上随机化，虽然最坏情况仍然是$O(n)$，但是几乎无法构造出这样的数据了。

最终，这种做法的时间复杂度没有证明，但是比较快；空间复杂度$O(n \log n)$。

3 打字机

3.1 暴力算法

首先暴力处理出所有的A_i和B_j。

枚举i，枚举j，暴力统计出现次数，时间复杂度$O(n^4)$。

枚举i，枚举j，使用KMP算法统计出现次数，时间复杂度$O(n^3)$。

视暴力程度期望得分20～70分。

3.2 所有字母均为a的算法

字母都是a，那么一个字符串只有长度有意义。我们统计所有A_i, B_j的

长度为a_i, b_j，那实际上就是在求$\sum_i \sum_j ij \max(b_j - a_i + 1, 0)$。

这个式子怎么求都可以，排序乱搞或者直接乱搞都能方便地求出来。

3.3 不太暴力的算法1

首先暴力求出所有的A_i。而对于B我们不暴力求，而是建出一棵Trie。

枚举i，考虑$O(n)$求出所有的出现次数。我们只需用KMP算法预处理A_i，然后再Trie上求出Trie的每个节点运行KMP算法时得到的匹配次数和匹配位置就可以了。

总的时间复杂度是$O(n^2)$，期望得分70分。

3.4 不太暴力的算法2

首先暴力求出所有的B_i。而对于A我们不暴力求，而是建出一棵Trie，然后求出AC自动机。

枚举j，考虑$O(n)$求出所有的出现次数。这个似乎不是很好求，但是我们只需要求$\sum ik$就可以了。只需拿着串B_j去AC自动机里走一次，这个值还是很好维护的。

总的时间复杂度是$O(n^2)$，期望得分70分。

3.5 标准算法

上面的两个平方级别的算法，一个是KMP+Trie上递推，一个是AC自动机+暴力。

只需把两种方法结合起来变成AC自动机+Trie上递推就可以了。期望得分100分。