Note：均有可能存在更优或更优美的算法，欢迎各位联系菜鸡出题人，QQ：352912432

树上的数

做法一

按照 $O(n^2)$ 的暴力做法，每次修改暴力递归子树统计，标记每一个访问到的节点，如果遇到访问过的节点，说明该节点的整棵子树都已经被标记过了，可以直接返回。可以证明这样做的复杂度是 $O(n+m)$ 的。

做法二

相当于每次对dfs序一个区间染色，可以用并查集实现，即把相邻的已被染色的位置连到一个联通块内，时间复杂度为$O(n\alpha(n))$或$O(n\log n)$，实际效率和做法一差不多。

时代的眼泪

可以交换求和顺序，询问u就相当于考虑以u为根时每个点x的子树内有多少个点的w小于该点，记这个值为h_x,答案为所有h_i的总和。

先考虑如何求$u=1$时的答案，以1为根dfs，同时维护一个数据结构用来查询到根路径上有多少个点的a_i大于当前点。

然后通过换根dp求出每个u的答案，可以发现当根从点x变为x的儿子y时，y的子树会变为所有点，x的子树会变为不“在以1为根时y子树内”的所有点。

此做法时间复杂度$O(n\log n)$。

传统艺能

首先对于没有修改的情况，有一个显然的 dp 转移：

```
if(!last[i]) dp[i] = dp[i - 1] * 2 + 1;
if(last[i]) dp[i] = dp[i - 1] * 2 - dp[last[i] - 1];
```

其中 `dp[i]` 表示考虑前 i 位的方案数，`last[i]` 表示最近的与 `a[i]` 相同的字符的位置，如果 `a[i]` 第一次出现，则 `last[i]` 为 0

现在我们考虑矩阵。

设状态矩阵 A，其中 `A[i][j]` 表示以 i 开头，在结尾预支一个 j 的方案数，其中 i,j 为 0 时表示一个空序列。

对于一个数 i 的初始矩阵，对于对角线的元素，显然方案数为 1，而对于第 i 行，显然都为 1。

比如对于数字 1 的初始矩阵为 $\begin{bmatrix}1&0&0&0\\1&1&1&1\\0&0&1&0\\0&0&0&1\end{bmatrix}$。

容易发现两个状态矩阵的合并就是矩阵乘法，即 $C_{i,j}=\sum\limits_{k=0}^{3}A_{i,k}\times B_{k,j}$

用线段树支持查询和修改即可，时间复杂度$O(4^3m\log n)$。

铺设道路

先令$b_i=d_i-d_{i-1}$ 那么我们相当于每次选取$l<r$，令b_l-1，b_r+1，代价$(r-l)^2$，令所有b变成0。容易发现最短时间为$\sum_i max(b_i,0)$，达到最短时间需要我们每次每次选择的$b_l>0$,$b_r<0$。

考虑一种贪心策略，遍历b_i，若有$b_i>0$将其压入队列中，$b_i<0$则

若使得代价最大，与最靠左的b_l配对。

若使得代价最小，与最靠右的b_l配对。

注意到$d_i\geq 0$，所以我们一定能找到数字配对。

可以用交换法证明贪心的正确性。时间复杂度$O(n)$。