树上简单求和 解题报告

题目大意

给定两棵 n 个节点的树，共用点权，m 次操作对第一棵树链加，对第二棵树链求和。

数据范围

$1 \leq n, m \leq 2 \times 10^5$。

解题过程

子任务 1

在第一棵树上暴力跳链修改，第二棵树上暴力跳链查询，时间复杂度 $O(mn)$，获得 5 分。

子任务 4

A 性质保证了第二棵树在所有无根树中均匀随机生成，直径是期望 $O(\sqrt{n})$ 的。

将第一棵树的链修改视为四次到根的链修改，如果进行单点查询则在第一棵树上进行子树查询。

具体地，记录第一棵树的 dfs 序，修改时转成 x 到根的链修改，然后对 x 单点加，查询时在第二棵树上暴力跳链，对 y 单点查询时只需要查询节点 y 在第一棵树的子树内的标记和即可。

使用 $O(\sqrt{n}) - O(1)$ 的分块可以做到 $O(m\sqrt{n})$，使用树状数组的 $O(m\sqrt{n}\log n)$ 的做法也能通过，获得 14 分。

子任务 6

C 性质保证了第一棵树按 dfs 序给出，第二棵树是 1 到 n 的链。

对 1 到 n 的序列建线段树，对第一棵树重链剖分。

对于线段树的一个节点，其包含区间 $[l, r]$，我们考虑区间内出现过的链顶编号最小的重链上的所有节点，记录其在区间 $[l, r]$ 的出现次数。

容易发现，对于一次重链上的区间修改 $[l, r]$（l, r 表示链顶和链底的编号），在线段树上递归到的 $O(\log n)$ 个区间，如果区间包含链上的节点，那么这个区间所维护的重链一定是当前重链，直接打标记即可。

查询直接在线段树上查询，因为线段树上每个节点依旧可以维护区间和，标记下传时判一下两个儿子维护的重链和自己是否一样。

时间复杂度 $O(n + m\log^2 n)$，获得 19 分。

子任务 5

B 性质保证了两棵树都是随机生成的链，我们可以将它们视为序列。

对两个序列都分块，对第一个序列做 $O(\sqrt{n}) - O(1)$ 的区间修改，区间查询分块。

对第二个序列的散块，直接在第一个序列上 $O(1)$ 查询。

对第二个序列的整块，可以每块离线下来，对第一个序列建立前缀和数组 c_i，表示 $[1, i]$ 中有 c_i 个节点在这个块中出现。

于是我们可以 $O(1)$ 算出来第一个序列的区间 $[l,r]$ 修改对这个整块和的影响，从而维护整块的值。

时间复杂度 $O((n+m)\sqrt{n})$，获得 17 分。

验题人的子任务题解：

在这一档部分分中，保证两棵树是随机生成的两条链。可以考虑将两条链视作两个序列 a,b，并对序列 a 重标号为 1 到 n。

考虑对于两个序列分别进行序列分块，块长为 B。我们先来考虑修改操作。对于散块，我们可以进行暴力修改，复杂度为 $O(B)$。

对于整块而言，如果进行暴力的修改，那么我们需要遍历块内的所有元素，并找到其在序列 b 上的位置进行修改。

定义 pos_i 表示下标 i 所属的块的标号，有 $pos_i=\frac{i-1}{B}+1$。

定义 p_x 表示**数值** x 在序列 b 中的位置。

定义一次修改操作为 (x,k)，表示序列 a 中的**下标** x 加上了 k。

两个修改操作 $(x_1,k_1),(x_2,k_2)$ 是**可合并的，当且仅当** $pos_{x_1}=pos_{x_2}$ 且 $pos_{p_{x_1}}=pos_{p_{x_2}}$，也就是两个修改操作的影响都分别局限于序列 a,b 中的同一块。

注意到序列随机生成，猜想将所有可合并的修改操作合并后，剩余的修改操作应该会很少，实际也是如此，打表发现块长取到 $[2\times 10^3,2.5\times 10^3]$ 时理论较优。可以对这些修改操作进行根号分治，减小程序的常数。

考虑到内存访问顺序，应该使用邻接表来存储这一些修改，剩余的则是经典的分块操作，复杂度 $O((B+\frac{n}{B}+K)m)$，其中 B 为块长， K 为块长为 B 时本质不同的修改操作个数。

验题人的程序块长取 2.3×10^3 时最优，此时本质不同的修改操作只有约 7569 种，可以通过这道题。这种写法做到了在线，同时运行效率比较可观。

子任务 2,3

留给时空常数或复杂度较高的做法，例如时间 $O(m\sqrt{n}\log n)$ 或空间 $O(n\sqrt{n})$ 的做法。

子任务 7

对第二棵树进行树分块，找到 $O(\sqrt{n})$ 个关键点，使所有点到最近的关键点祖先距离不超过 $O(\sqrt{n})$。

对第一棵树的 dfs 序分块，使用子任务 4 的方法可以做到 $O(1)$ 查询。

查询时可以差分成三次到根的链查询，从当前节点向上跳到第一个关键点，中途的点直接 $O(1)$ 查询即可，于是我们现在需要对于关键点到根的路径和动态维护，可以分别离线下来。

对于每一个关键点到根的路径，将其标为黑色，我们在第一棵树上记录 c_x 表示节点 x 到根的路径上黑色节点的个数，预处理每次询问的 LCA 可以做到 $O(1)$ 查询每次修改对第二棵树关键点到根路径和的影响。

于是时间复杂度 $O((n+m)\sqrt{n})$ 空间 $O(n+m)$，可以通过。

验题人提出的更快的做法：

对两棵树求出括号序，那么我们可以将一条链视为括号序上的一段区间和 LCA 的单点。

对两棵树的括号序列分块，预处理块前缀之间的贡献，这部分的空间是 $O(n)$ 的，通过多次前缀和优化可以做到时间 $O(n\sqrt{n})$。

对于一次修改操作，散块和 LCA 使用 $O(1) - O(\sqrt{n})$ 的分块。

注意到这里需要差分成两个前缀，因为我们只能快速处理前缀块之间的贡献。

找到所对应的 $O(1)$ 个前缀块，$O(\sqrt{n})$ 枚举第二棵树的前缀块，更新贡献。

查询同理，散块使用子任务 4 的方法 $O(1)$ 查询，整块对整块的贡献已经解决，我们只需要考虑散块对整块的贡献。

但因为可以处理每一个点所对应的块编号，然后在修改时 $O(1)$ 打标记，所以这也是平凡的。

因为这些做法大多数只涉及序列操作，所以常数较小。

两种做法都能通过。

特别感谢

感谢杨岱谦同学对题目初步思路提供的帮助。

感谢钟雨霖、周桓毅同学的验题以及分别提出的子任务做法，更快的正解。