集训队互测 解题报告

长沙市长郡中学 朱屹帆

2023 年 10 月

1 题目

1.1 题目大意

小 Z 有一棵 n 个节点构成的树,这棵树以点 1 为根,边权均为 1 ,每个节点 x 有点权 a_x ,初始所有节点的权值为 0 .

小 Z 想对这棵树进行总共 m 次操作,并且 小 Z 希望能够支持以下四种类型的操作:

1. 给定 x,y,k,v ,对满足节点 i 到树上唯一的最短路径 (x,y) 的最短距离 $\leq k$ 的所有节点 i 的权值增加 v ,即 $a_i \leftarrow a_i+v$.
2. 给定 x,v ,将点 x 的子树内的所有节点 i 的权值增加 v ,即 $a_i \leftarrow a_i+v$.
3. 给定 x,y,k ,查询所有满足节点 i 到树上唯一的最短路径 (x,y) 的最短距离 $\leq k$ 的所有节点 i 的权值的和.
4. 给定 x ,查询点 x 的子树内的所有节点 i 的权值和.

由于答案会比较大,输出对 2^{64} 取模即可.

1.2 输入格式

第一行为两个正整数 n,m .

接下来第 2 行到第 n 行每行两个正整数 u,v 来描述树上的一条边.

接下来 m 行,每行先给出一个操作类型 op .

当 $op=1$ 时给定 x,y,k,v 来描述操作一;

当 $op=2$ 时给定 x,v 来描述操作二;

当 $op=3$ 时给定 x,y,k 来描述操作三;

当 $op=4$ 时给定 x 来描述操作四.

1.3 输出格式

对于每个操作 3 和操作 4 ,输出其对应的答案对 2^{64} 取模的结果.

1.4 数据范围

数据范围: $1 \leq n, m \leq 2 \times 10^5, 1 \leq x, y \leq n, 1 \leq v \leq 10^9, 0 \leq k \leq 3$.

子任务 1 (10分) : $1 \leq n, m \leq 5 \times 10^3$.

子任务 2 (10分) : $k = 0$.

子任务 3 (5分) : 保证树形态构成一条形如 1 连接 2 连接 ... 连接 n 的链(所有边均可以表示为 i 连接 $i + 1$ 的形式).

子任务 4 (10分) : 保证 $k = 1$,操作类型只包含操作一和操作三,并且操作一和操作三满足 $x = y$,即修改和查询的链是单点.

子任务 5 (10分) : 保证操作类型只包含操作一和操作三,并且操作一和操作三满足 $x = y$,即修改和查询的链是单点.

子任务 6 (15分) : 保证操作一和操作三满足 $x = y$,即修改和查询的链是单点.

子任务 7 (15分) : $k \leq 1$.

子任务 8 (25分) : 无特殊限制.

1.5 时空限制

时间限制: 7s .

空间限制: 1GB .

2 解题过程

定义 fa_x 表示以点 1 为根时点 x 在原树上的父节点, dep_x 表示以点 1 为根时点 x 与点 1 的最短距离, $\mathrm{dist}(x, y)$ 表示树上点 x 到点 y 的最短距离.

2.1 算法 1

暴力模拟修改过程,每次将满足条件的范围内的点进行处理即可.

时间复杂度 $O(nm)$,可以通过子任务 1 ,期望得分 10 分.

2.2 算法 2

当 $k=0$ 时可以发现我们需要支持对原树的链加,子树加,链查询,子树查询这四种操作,可以使用轻重链剖分进行维护.

时间复杂度 $O(m\log^2 n)$,可以通过子任务 2 ,与算法 1 结合后期望得分 20 分.

2.3 算法 3

树形态为链时,可以发现操作一和操作三的修改范围最终会体现在一条链上,具体的修改范围是可以简单计算的,使用线段树或树状数组维护区间加(修改)和区间和(查询)即可.

时间复杂度 $O(m\log n)$,可以通过子任务 3 ,与算法 1 和算法 2 结合后期望得分 25 分.

2.4 算法 4

当操作只包含操作一和操作三并且操作一和操作三满足 $k=1$ 且 $x=y$ 时,其修改范围最终会体现在一个点的邻域上.

使用根号分治,将度数 $\leq\sqrt{n}$ 的点称为白点,将度数 $>\sqrt{n}$ 的点称为黑点,分别进行处理.

我们不妨考虑修改对查询的贡献:

$k=0$ 的修改对 $k=0$ 的查询:维护每个点的权值即可.

$k=0$ 的修改对 $k=1$ 的查询:单点加邻域查,可以注意到对一个点加之后,其自身和邻域的所有点的邻域的权值都会增加,因此可以转化成邻域加单点查.

$k=1$ 的修改对 $k=0$ 的查询:邻域加单点查,我们对每个黑点维护其邻域加的标记,当修改的点是黑点时直接修改其标记,当修改的点是白点时扫描其所有邻域,并将邻域所有点权值直接更新;查询时计算一个点邻域中的所有黑点标记的和,加上自己的权值即可.

$k=1$ 的修改对 $k=1$ 的查询,设修改点为 x ,查询点为 y ,我们分成以下四个部分进行处理:

$\mathrm{dist}(x,y)=0$ 时,此时有 $x=y$,直接维护贡献即可.

$\mathrm{dist}(x,y)=1$ 时,此时两个相邻点的邻域的交的大小恰好为 2 ,因此可以转化成邻域加单点查.

$\mathrm{dist}(x,y)=2$ 时,此时两个点的邻域的交大小恰好为 1 ,并且只有三种可能的点对 (x,y) 树上关系,对三种情况分别进行简单计算即可.

$\mathrm{dist}(x,y)>2$ 时,修改和查询的范围之间并没有影响.

时间复杂度 $O(n\sqrt{n})$,可以通过子任务 4 ,与算法 1 ,算法 2 和算法 3 结合后期望得分 35 分.

2.5 算法 5

当操作只包含操作一和操作三并且操作一和操作三满足 $x=y$ 时,即操作一和操作三作用在单点上.

我们尝试继续分析算法 4 ,此时由于修改范围不再是邻域,并不太适合使用根号分治,但注意到计算修改对询问的贡献似乎是比较可行的,我们尝试对这个方向进行分析.

使用根号重构,类似于对操作序列进行分块,对于操作序列中的一个块,在此之前的块的修改的贡献可以通过分层转移的方式对应到每个单点上.

此时对于块内,我们考虑一个修改对一个查询的贡献.

首先处理出来以点 x 为根时与点 x 距离 $\leq k$ 的点的数量,设为 $f_{x,k}$,这是可以通过换根 dp 简单处理的.

然后处理出来以点 x 为根时去掉点 x 邻域中的点 y 对应的子树之后距离 $\leq k$ 的点的数量,设为 $g_{(x,y),k}$,这是可以通过类似分层转移得到的,即 $g_{(x,y),k}=f_{x,k}-g_{(y,x),k-1}$.

考虑一次修改对一次查询的贡献,设修改点为 x ,查询点为 y ,修改的范围为 $a(0\leq a\leq 3)$,查询的范围为 $b(0\leq b\leq 3)$.

当 $\operatorname{dist}(x,y)>a+b$ 时,修改和查询的范围之间并没有影响.

当 $\operatorname{dist}(x,y)\leq a+b$ 时,由于 $a+b$ 较小,因此可以将点 x 和点 y 之间的路径直接取出来,并枚举其中的点 z ,可以发现去掉这条链上的边之后,点 z 可达的范围内并且在修改和查询的范围内的点恰好为与点 z 的距离 $\leq \min(a-\operatorname{dist}(x,z),b-\operatorname{dist}(y,z))$ 的点,即去掉至多两个子树之后的与点 z 距离 $\leq$ 一个值的点的数量,这个可以通过我们预处理出来的信息快速计算.

计算出将所有的 z 的贡献的和,即可计算出两个操作之间的范围的交,进而可以计算出一次修改对一次查询的贡献.

经过分析可以发现这个算法的本质是在计算一次修改范围和一次查询范围的交的大小,计算出修改对查询的贡献.

时间复杂度 $O(nk\sqrt{n})$,可以通过子任务 4,5 ,与算法 1 ,算法 2 和算法 3 结合后期望得分 45 分.

2.6 算法 6

当操作一和操作三均满足 $x=y$ 时,操作一和操作三均作用在单点上.

我们可以尝试在算法 5 的基础上添加上对子树修改和查询的贡献的分析.

使用根号重构,此时对于操作序列的一个块,在此之前的块的修改的贡献可以通过树上差分和分层转移的方式对应到每个单点上.

此时对于块内,我们考虑补充一些种类的修改对一些种类的查询的贡献.

操作一对操作三的贡献在算法 5 中有较为详细的讲解.

考虑计算操作二对操作四的贡献,由于子树可以用 dfs 序的区间来描述,因此交的大小是容易计算的.

由于操作一对操作四的贡献和操作二对操作三的贡献中,都是计算与一个点距离 $\leq k$ 的所有点和一个点的子树的交,因此我们可以使用相同的计算方式.

设操作与一个点距离 $\leq k$ 的这个点为 x ,操作范围为 a ,操作一个点的子树的这个点为 y ,按照算法 5 先预处理出来 f 和 g 的信息.

当点 x 不在点 y 的子树内时,可以发现交的大小恰好为点 y 子树中与点 y 距离 $\leq a-\mathrm{dist}(x,y)$ 的点的数量,即为 $g_{(y,fa_y),a-\mathrm{dist}(x,y)}$.

当点 x 在点 y 的子树内时,可以发现交的大小恰好为所有与点 x 距离 $\leq a$ 的点的数量减去超出点 y 子树的点的数量,当 $x=y$ 时为 $g_{(y,fa_y),a}$,否则为 $f_{x,a}-\left(f_{y,a-\mathrm{dist}(x,y)}-g_{(y,fa_y),a-\mathrm{dist}(x,y)}\right)$.

时间复杂度 $O(nk\sqrt{n})$,可以通过子任务 4,5,6 ,与算法 1 ,算法 2 和算法 3 结合后期望得分 60 分.

2.7 算法 7

注意到算法 6 的核心思想在与计算操作对询问的贡献,即计算两者之间范围的交的大小,我们尝试换一种思路,当 $k=1$ 时类似树链剖分的直接维护.

考虑到当 $k=0$ 时,我们可以依据轻重链剖分的性质,由于至多跳 $\log n$ 条轻边,因此在 $\log n$ 条重链上进行处理,重链的标号具有连续性,因此可以快速进行修改和查询,这启发我们尝试在保持轻重链剖分在 $\log n$ 条重链上进行处理的性质的基础上,通过稍微修改重链的标号,使得链上邻域修改的标号大致连续.

我将这种在保持树链剖分部分性质的基础上通过调整标号顺序得到更多性质的方式称为"重标号树剖",这里介绍一种标号方式:

按照重链中深度最小的点的 dfs 顺序来处理每一条重链,考虑对于一条重链,先将这条重链上的所有点按照深度从小到大排序进行标号,然后将这条重链上的所有点按照深度从小到大处理每个点,处理到点 x 时将点 x 的所有轻儿子进行标号,注意到有可能处理一条重链时其深度最小的点已经被标号,因此已经被标号的点不会再进行标号.

可以发现这样的结构仍然具有树链剖分的许多性质,比如仍然保证了至多在 $\log n$ 条重链上进行处理的性质,并且这样标号使得邻域信息具有了一些连续性.

考虑一次链邻域修改操作:我们需要处理出对于同一条重链上修改点 x 到点 y 路径邻域信息,设 $dep_x \leq dep_y$,可以注意到路径邻域正是点 x 邻域开始标号的编号到点 y 邻域结束标号的编号,由于编号的连续性,我们可以使用数据结构快速维护更新,而对于同一条重链上修改点 x 到点 y 路径上的信息,则可以直接类似树链剖分那样维护,但是注意到重链中深度最小的点的编号并不一定满足连续性,这个点可能需要为了保证其父节点所在重链的编号的连续性而不与自己所在重链编号连续,因此对重链顶部的节点进行特殊处理,其余重链上的点快速维护更新即可;注意到对新的重链进行处理的时候可能会导致原来的重链的信息被重复计算,因此需要去掉重复部分.

考虑一次链邻域查询操作:可以发现和修改操作是类似的,对于一条重链,路径邻域信息可以通过编号的连续性快速得到,路径上的点的编号除去重链中深度最小的点之外,其他的点的编号满足连续性,因此对重链顶部的节点进行特殊处理之后,可以快速得到重链上路径的点的信息.

子树修改和子树查询是简单的,由于我们是按照 dfs 顺序来处理的每一条重链,因此除去当前节点以外其子树内的节点一定满足编号连续,因此可以快速处理出来,当前节点特殊处理即可.

时间复杂度 $O(m\log^2 n)$,常数较大,可以通过子任务 2,4,7 ,与算法 1 和算法 3 结合后期望得分 50 分.

2.8 算法 8

定义范围阈值 $K=3$,可以发现算法 7 十分具有拓展性,考虑对标号方式进行改进:

按照重链中深度最小的点的 dfs 顺序来处理每一条重链,考虑对于一条重链,先将这条重链上的所有点按照深度从小到大排序进行标号,然后将这条重链上的所有点按照深度从小到大处理每个点,处理到点 x 时将点 x 的所有轻儿子子树中的与点 x 距离 $=1$ 的所有点进行标号;然后将这条重链上的所有点按照深度从小到大处理每个点,处理到点 x 时将点 x 的所有轻儿子子树中的与点 x 距离 $=2$ 的所有点进行标号;然后将这条重链上的所有点按照深度从小到大处理每个点,处理到点 x 时将点 x 的所有轻儿子子树中的与点 x 距离 $=3$ 的所有点进行标号;注意到有可能处理一条重链时与重链顶部节点的距离 ≤ 3 的节点已经被标号,因此已经被标号的点不会再进行标号.

可以发现这样的结构在算法 7 的基础上需要对重链上与重链顶部节点距离 ≤ 3 的所有节点进行特殊处理.

对于每一个节点 x ,考虑维护线段集合 $s0_{x,k}$ 表示与点 x 距离 $=k$ 的在点 x 的子树内的所有节点编号区间(当 $k=4$ 时描述的是距离 >4 的点),线段集合 $s1_{x,k}$ 表示与点 x 距离 $=k$ 的在点 x 的轻儿子的子树内的所有节点编号区间(当 $k=4$ 时描述的是距离 >4 的点),可以发现转移是简单的.

根据标号时编号的连续性,可以发现 $s1_{x,k}(0\leq k\leq 4)$ 均满足线段数量是 $O(1)$ 的,具体来说均满足线段数量 ≤ 2 ,因为一个节点的与其距离 ≤ 3 的距离相同的轻儿子一定会在同一时刻被连续标号,但由于可能并不是处理这个节点所在重链时对其的标号(即与其距离 <3 的祖先方向的节点与其并不在一条重链上,并且会将这一层的节点进行标号),因此这个节点的重儿子的编号可能会将连续编号划分成两段,因此线段数量 ≤ 2 ;而当距离 ≥ 4 时显然编号是连续的,因此线段数量 ≤ 1 .

同时也可以发现 $s0_{x,k}(0 \le k \le 4)$ 均满足线段数量是 $O(K)$ 的,具体来说满足线段数量 ≤ 4 ,因为一个节点的与其距离 ≤ 3 的距离相同的所有节点中,对于相同的重儿子来说,每一层一定都是连续的,因此即使每一层之间都是独立的,线段数量仍然满足 ≤ 4 .

因此如果我们需要对于单点修改其 $\le k$ 的轻儿子子树的范围内的所有点,则线段总数为 $O(K)$,如果我们需要对于单点修改其 $\le k$ 的子树范围内的所有点,则线段总数为 $O(K^2)$.

考虑一次链邻域修改操作:可以发现和算法 7 的链邻域修改操作是类似的,考虑每次对同一条重链上的点 x 和点 y 之间的信息进行处理,首先对重链上的顶部的至多三个节点进行特殊处理,需要对其轻儿子子树信息进行处理;重链的尾部节点需要对其子树信息进行处理;中间的部分可以通过编号的连续性进行处理;重链贡献重复部分对其子树信息进行处理;总共需要处理 $O(\log n)$ 条重链,因此该部分时间复杂度为 $O(K^2 \log^2 n)$;最后需要对最近公共祖先之上的部分进行处理;因此单次修改总时间复杂度为 $O(K^2 \log^2 n)$.

考虑一次链邻域查询操作:可以发现和修改操作是类似的,同样对重链顶部上至多三个节点进行特殊处理;重链尾部节点进行子树处理;中间部分通过编号连续性来处理;重链贡献重复部分进行子树处理;最后对最近公共祖先之上的部分进行处理;因此单次查询总时间复杂度为 $O(K^2 \log^2 n)$.

子树修改和子树查询是简单的,由于我们按照 dfs 顺序给每条重链标号,因此我们可以扫描节点 x 的子树信息进行处理,单次修改和查询的总时间复杂度为 $O(K^2 \log n)$.

时间复杂度 $O(mK^2 \log^2 n)$,常数较大,可以通过所有子任务,期望得分 100 分.

3 参考资料

暂无.