差分及树上差分

华南理工大学　江熠玲

2020 年 2 月 3 日

前言

前言

今天讲课的内容相对基础，当作巩固。

课件中题目若未特殊声明，时限1s，内存128M。

欢迎踊跃回答。

目录

目录

差分基础

差分基础

给出n个数，再给出个q询问，每个询问给出l_i, r_i, x，要求你在$[l_i, r_i]$上每一个值都加上x，输出最终序列。

差分基础

给出n个数，再给出个q询问，每个询问给出l_i, r_i, x，要求你在$[l_i, r_i]$上每一个值都加上x，输出最终序列。

假如要在l_i和r_i上全都加一个x,很显然为$O(n)$。

差分基础

给出n个数，再给出个q询问，每个询问给出l_i, r_i, x，要求你在$[l_i, r_i]$上每一个值都加上x，输出最终序列。

假如要在l_i和r_i上全都加一个x,很显然为$O(n)$。

考虑将$O(nq)$变为$O(n+q)$。

差分基础

差分基础

我们定义$a[i]$ 为原数列，$c[i] = a[i] - a[i-1]$,显然$a[n] = \sum_{i=1}^{n} c[i]$

。

我们定义$a[i]$ 为原数列，$c[i] = a[i] - a[i-1]$,显然$a[n] = \sum_{i=1}^{n} c[i]$

。

若想要将区间$[l, r]$ 的数全部$+v$ 则只需要将$c[l] + v, c[r+1] - v$即可。

差分基础

我们定义$a[i]$ 为原数列，$c[i]=a[i]-a[i-1]$,显然$a[n]=\sum_{i=1}^{n} c[i]$ 。

若想要将区间$[l,r]$ 的数全部$+v$ 则只需要将$c[l]+v, c[r+1]-v$即可。

差分基础

差分基础

$$
\begin{aligned}
\sum_{i=1}^{n} a[i] &= (c[1]) + (c[1] + c[2]) + \cdots + (c[1] + c[2] + \cdots + c[n]) \\
&= n \times c[1] + (n-1) \times c[2] + \cdots + c[n] \\
&= n \times (c[1] + c[2] + \cdots + c[n]) \\
&\quad - (0 \times c[1] + 1 \times c[2] + \cdots + (n-1) \times c[n])
\end{aligned} \tag{1}
$$

$$\begin{aligned}\sum_{i=1}^{n} a[i] &= (c[1]) + (c[1]+c[2]) + \cdots + (c[1]+c[2]+\cdots+c[n]) \\ &= n \times c[1] + (n-1) \times c[2] + \cdots + c[n] \\ &= n \times (c[1]+c[2]+\cdots+c[n]) \\ &\quad - (0 \times c[1] + 1 \times c[2] + \cdots + (n-1) \times c[n])\end{aligned} \tag{1}$$

所以，我们维护一个数组$c_2[i] = (i-1) \times c[i]$

差分基础

$$
\begin{aligned}
\sum_{i=1}^{n} a[i] &= (c[1]) + (c[1] + c[2]) + \cdots + (c[1] + c[2] + \cdots + c[n]) \\
&= n \times c[1] + (n-1) \times c[2] + \cdots + c[n] \\
&= n \times (c[1] + c[2] + \cdots + c[n]) \\
&\quad - (0 \times c[1] + 1 \times c[2] + \cdots + (n-1) \times c[n])
\end{aligned} \tag{1}
$$

所以，我们维护一个数组$c_2[i] = (i-1) \times c[i]$

在将区间$[l, r]$ 的数全部$+v$ 则还需同时

将$c_2[l] + v \times (i-1), c_2[r+1] + (-v) \times r$ 。

差分基础

$$
\begin{aligned}
\sum_{i=1}^{n} a[i] &= (c[1]) + (c[1]+c[2]) + \cdots + (c[1]+c[2]+\cdots+c[n]) \\
&= n \times c[1] + (n-1) \times c[2] + \cdots + c[n] \\
&= n \times (c[1]+c[2]+\cdots+c[n]) \\
&\quad - (0 \times c[1] + 1 \times c[2] + \cdots + (n-1) \times c[n])
\end{aligned} \tag{1}
$$

所以，我们维护一个数组$c_2[i] = (i-1) \times c[i]$

在将区间$[l, r]$ 的数全部$+v$ 则还需同时

将$c_2[l] + v \times (i-1), c_2[r+1] + (-v) \times r$ 。

结论： $\sum_{i=1}^{n} a[i] = n \times \sum_{i=1}^{n} c[i] - \sum_{i=1}^{n} c_2[i]$

目录

目录

树上差分

树上差分

数的两个性质：

1 任意两个节点之间有且只有一条路径。

树上差分

数的两个性质：

1. 任意两个节点之间有且只有一条路径。
2. 根节点确定时，一个节点只有一个父亲节点。

树上差分

树的两个性质：

1. 任意两个节点之间有且只有一条路径。
2. 根节点确定时，一个节点只有一个父亲节点。

树上差分：点的差分

树上差分：点的差分

在一棵n个结点的树中，形容从s_i走到到t_i的要求,求这条路径上的点被经过的次数。

树上差分：点的差分

在一棵n个结点的树中，形容从s_i走到到t_i的要求,求这条路径上的点被经过的次数。

显然，我们需要找到他们的LCA（中转点）。

树上差分：点的差分

在一棵n个结点的树中，形容从s_i走到到t_i的要求,求这条路径上的点被经过的次数。

显然，我们需要找到他们的LCA（中转点）。

我们需要让$cnt[s]++$,让$cnt[t]++$，而让他们的$cnt[lca]--$，$cnt[faher[lca]]--$;

树上差分：点的差分

在一棵n个结点的树中，形容从s_i走到到t_i的要求,求这条路径上的点被经过的次数。

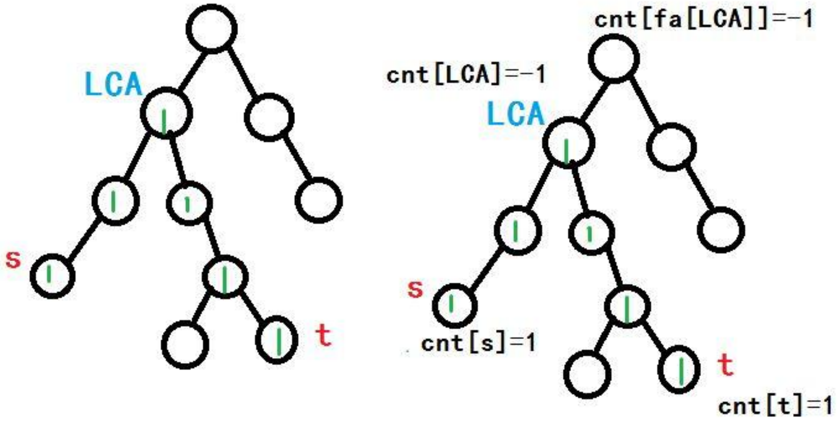
显然，我们需要找到他们的LCA（中转点）。

我们需要让$cnt[s]++$,让$cnt[t]++$，而让他们的$cnt[lca]--$，$cnt[faher[lca]]--$;

最终统计：$cnt[i]+=\sum_{j\in childI} cnt[j]$。

树上差分：点的差分

树上差分：点的差分



树上差分：边的差分

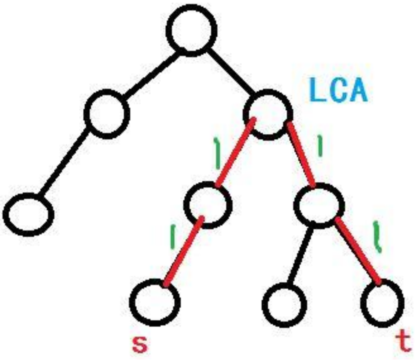
树上差分：边的差分

边进行差分需要把边塞给点,但是,这里的标记并不是同点差分一样。

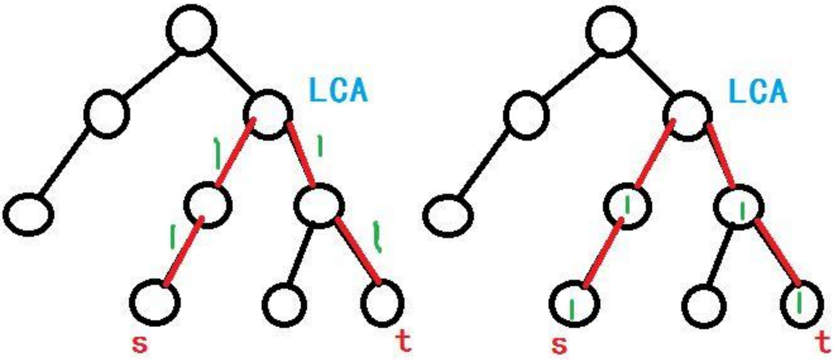
把边塞给点的话,是塞给这条边所连的深度较深的节点（儿子节点）。

树上差分：边的差分

树上差分：边的差分



树上差分：边的差分



$$cnt[s]++, cnt[t]++, cnt[LCA]-=2$$

树上差分

树上差分

树上差分重点在于思想的运用。

课题很少但与其他算法的结合考察还是比较多的，并且几乎与LCA成对出现，因为主要针对的是树上的路径问题。

目录

目录

例题

例题

[BZOJ 4390]Max Flow

[BZOJ 4390]Max Flow

给定一棵有N个点的树，所有节点的权值都为0。有K次操作，每次指定两个点s, t，将s到t路径上所有点的权值都加一。请输出K次操作完毕后权值最大的那个点的权值。

[BZOJ 4390]Max Flow

[BZOJ 4390]Max Flow

树上关于点的差分。对于差分数组标记后从下往上累加，求最大值即可。

[NOIP 2012]借教室

[NOIP 2012]借教室

我们需要处理接下来n天的借教室信息，其中第i天学校有r_i个教室可供租借。共有m份订单，每份订单用三个正整数描述，分别为d_j, s_j, t_j，表示某租借者需要从第s_j天到第t_j天租借教室（包括第s_j天和第t_j天），每天需要租借d_j个教室。

借教室的原则是先到先得，也就是说我们要按照订单的先后顺序依次为每份订单分配教室。如果在分配的过程中遇到一份订单无法完全满足，则需要停止教室的分配，通知当前申请人修改订单。

输出需通知修改的订单编号。

[NOIP 2012]借教室

[NOIP 2012]借教室

二分能满足的订单数。

差分数组，对于二分的一个值，先差分到当前订单，扫描维护前缀和即为当前借的教室数与d作比较即可。

[EOJ 3631]Delivery Service

[EOJ 3631]Delivery Service

给出一颗树，树的边上有边权。然后q次询问，每次询问u, v的最短距离，在询问前，可以用魔法来换任意两个边，可以使用任意次魔法，问所有询问和的最小值是多少。

$1 \leq n, m \leq 2 \times 10^5$。

[EOJ 3631]Delivery Service

[EOJ 3631]Delivery Service

可以使用任意次魔法，就是可以交换任意边，那么一定把用的最多次数的边变成最小的，以此类推。

树上差分统计路径覆盖次数，从大到小贪心赋边权即可。

[POJ 3417]Network

[POJ 3417]Network

一棵有N个点的树，再往里面加入M条新边，现在要破坏其中的两条边，要求一条是原来树中的边，一条是新边，使其不连通。求方案的数量。

$1 \leq N \leq 100000), 1 \leq M \leq 100000)$。

[POJ 3417]Network

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1 若这条边被覆盖0次，

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1. 若这条边被覆盖0次，则可以任意再切断一条附加边。

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1. 若这条边被覆盖0次，则可以任意再切断一条附加边。
2. 若这条边被覆盖1次，

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1. 若这条边被覆盖0次，则可以任意再切断一条附加边。
2. 若这条边被覆盖1次，那么只能再切断唯一的一条附加边。

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1. 若这条边被覆盖0次，则可以任意再切断一条附加边。
2. 若这条边被覆盖1次，那么只能再切断唯一的一条附加边。
3. 若这条边被覆盖2次及以上，

[POJ 3417]Network

对于新加的一条边来说，肯定会与之前的树形成一个环，而此时环内的树上边和新加的这条边一同删除就会是一种方案。

而这道题是将所有新边都加入后的情况，那么我们看每条边，如果没有与它形成环的情况，那么这条边删除肯定会使得图不连通，即情况就会加M，也就是和新加的M条边任意组合都可以。

因而我们每次读入一条附加边，就给x到y的路径上的所有主要边记录上“被覆盖一次”，对于我们想要切割的一条主要边，有以下3种情况

1. 若这条边被覆盖0次，则可以任意再切断一条附加边。
2. 若这条边被覆盖1次，那么只能再切断唯一的一条附加边。
3. 若这条边被覆盖2次及以上，没有可行的方案。

[BZOJ 4326][NOIP 2015]运输计划

[BZOJ 4326][NOIP 2015]运输计划

L国有n个星球，还有$n-1$条双向航道，连通了所有星球。

小P掌管一家物流公司，该公司有很多个运输计划，每个运输计划形如：有一艘物流飞船需要从u_i号星球沿最快的宇航路径飞行到v_i号星球去。显然，飞船驶过一条航道是需要时间的，对于航道i，任意飞船驶过它所花费的时间为t_i，并且任意两艘飞船之间不会产生任何干扰。允许小P把某一条航道改造成虫洞，飞船驶过虫洞不消耗时间。在虫洞建设完成后，有m个运输计划会同时开始，所有飞船一起出发。如果小P可以自由选择将哪一条航道改造成虫洞，试求出小P的物流公司完成所有计划需要的最短时间是多少？

[BZOJ 4326][NOIP 2015]运输计划

[BZOJ 4326][NOIP 2015]运输计划

利用lca 求出每条路径的长度，二分答案ans，对于所有路径$> ans$的路径都至少需要删掉一条边。

[BZOJ 4326][NOIP 2015]运输计划

利用lca 求出每条路径的长度，二分答案ans，对于所有路径$> ans$的路径都至少需要删掉一条边。

最优方案一定是删去这些路径交集的最长边，即$> ans$的路径都经过的一条最长边。

[BZOJ 4326][NOIP 2015]运输计划

利用lca 求出每条路径的长度，二分答案ans，对于所有路径$> ans$的路径都至少需要删掉一条边。

最优方案一定是删去这些路径交集的最长边，即$> ans$的路径都经过的一条最长边。

树上差分：$s[i]$ 表示点i 到$fa[i]$ 这一条边经过的路径数，对于一条从u 到v 的路径，将$s[u]$ 和$s[v]$ 均$+1, s[lca(u,v)]-2$向上求和即可。

[BZOJ 4326][NOIP 2015]运输计划

利用lca 求出每条路径的长度，二分答案ans，对于所有路径$> ans$ 的路径都至少需要删掉一条边。

最优方案一定是删去这些路径交集的最长边，即$> ans$的路径都经过的一条最长边。

树上差分：$s[i]$ 表示点i 到$fa[i]$ 这一条边经过的路径数，对于一条从u 到v 的路径，将$s[u]$ 和$s[v]$ 均$+1, s[lca(u, v)] - 2$向上求和即可。

再判断删去这条边后能否满足所二分的答案即可。

[BZOJ 4326][NOIP 2015]运输计划

[BZOJ 4326][NOIP 2015]运输计划

```cpp
bool check(int lim)
{
    int i,j,cnt=0,t=0;
    for(i=1;i<=n;i++) sum[i]=0;
    for(i=1;i<=m;i++)
        if(d[i]>lim)
        {
            cnt++;
            t=max(t,d[i]-lim);
            sum[a[i]]++;sum[b[i]]++;
            sum[lca[i]]-=2;
        }
    work(1);
    for(i=1;i<=n;i++)
        if(sum[i]==cnt&&v[i]>=t) return true;
    return false;
}
```

```cpp
void work(int now)
{
    int i,j;
    for(i=last[now];i;i=e[i].next)
        if(e[i].to!=fa[now])
        {
            work(e[i].to);
            sum[now]+=sum[e[i].to];
        }
}
```

[Codeforces 739B]Alyona and a tree

[Codeforces 739B]Alyona and a tree

一棵根节点为1的树，每个点都有点值$a[i]$，每条边也有权值，$dist(v, u)$表示从v到u边权和。现在给出“控制”的定义：对一个点u，设点v在其子树上，且$dis(u, v) \le a_v$，则称u控制v。要求求出每个点控制了多少个点。

$1 \le n \le 2 \times 10^5, 1 \le a_i \le 10^9$。

[Codeforces 739B]Alyona and a tree

[Codeforces 739B]Alyona and a tree

如果v可以控制u,那么从v到u的路上的所有结点都可以控制u，因为从v到u路上的$dist(v,u)$是递减的。

[Codeforces 739B]Alyona and a tree

如果v可以控制u,那么从v到u的路上的所有结点都可以控制u，因为从v到u路上的$dist(v,u)$是递减的。

可以每次遍历一个点的时候，二分找出根节点到当前点i路径上点，找出$dist(j,i)$刚好大于$a[i]$的点，树上差分统计这条路径。

[Codeforces 739B]Alyona and a tree

如果v可以控制u,那么从v到u的路上的所有结点都可以控制u，因为从v到u路上的$dist(v,u)$是递减的。

可以每次遍历一个点的时候，二分找出根节点到当前点i路径上点，找出$dist(j,i)$刚好大于$a[i]$的点，树上差分统计这条路径。

而后遍历当前点i的所有儿子结点k，$cnt[i]+=cnt[k]$。

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

一个结点数为n的树上，m个人同时从s_i到t_i(第0时刻开始),每个点有个人蹲在那睡觉,第w_i秒的时候睁眼看看,然后他近视(只能看到当前点)。求每个人能看到多少人。

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

测试点编号	n	m	约定
1	$=991$	$=991$	所有人的起点等于自己的终点，即$S_i=T_i$
2			
3	$=992$	$=992$	$W_j=0$
4			
5	$=993$	$=993$	无
6	$=99994$	$=99994$	树退化成一条链，其中1与2有边，2与3有边，…，$n-1$与n有边
7			
8			
9	$=99995$	$=99995$	所有的$S_i=1$
10			
11			
12			
13	$=99996$	$=99996$	所有的$T_i=1$
14			
15			
16			
17	$=99997$	$=99997$	无
18			
19			
20	$=299998$	$=299998$	

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

对于25%的数据,有$1 \le n, m \le 993$。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于25%的数据,有$1 \leq n, m \leq 993$。

考虑此时n很小，可以对于每条路径上暴力模拟，经过某个点时可以看一下当前时刻，是否跟经过的点的w相等，如果相等，则贡献加一。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于25%的数据,有$1 \leq n, m \leq 993$。

考虑此时n很小，可以对于每条路径上暴力模拟，经过某个点时可以看一下当前时刻，是否跟经过的点的w相等，如果相等，则贡献加一。

对于45%的数据,有$1 \leq n, m \leq 993$或者$1 \leq n, m \leq 99995, S_i = 1$。

对于25%的数据,有$1 \le n, m \le 993$。

考虑此时n很小，可以对于每条路径上暴力模拟，经过某个点时可以看一下当前时刻，是否跟经过的点的w相等，如果相等，则贡献加一。

对于45%的数据,有$1 \le n, m \le 993$或者$1 \le n, m \le 99995, S_i = 1$。

注意到测试点$9-12$时，保证m条路径的出发点都是1，那么我们可以考虑如果将1作为树根，那么一条路径怎样才能对于它经过的点产生贡献。不难看出对于一个点i，若$deep[i] = w[i]$，就有贡献。对于一条路径，标记$sum[i] = 1$,之后做一遍DFS，对于i的所有子结点，$sum[i] += sum[j]$，传递标记，于是$sum[i]$此时便表示在$deep[i]$时刻经过的结点数。

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

对于60%的数据,除了以上数据，新增10%，有$1 \leq n, m \leq 99995$且树退化成链。

对于60%的数据,除了以上数据，新增10%，有$1 \leq n, m \leq 99995$且树退化成链。

在链上肯定是要么往左要么往右，即$S \leq T$或者$S > T$。先只考虑$S \leq T$的情况，如果对于S到T之间的点i，要产生贡献的话，肯定满足$i - S = w[i]$，移项可得$S = i - w[i]$时才可以满足要求。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于60%的数据,除了以上数据，新增10%，有$1 \le n, m \le 99995$且树退化成链。

在链上肯定是要么往左要么往右，即$S \le T$或者$S > T$。先只考虑$S \le T$的情况，如果对于S到T之间的点i，要产生贡献的话，肯定满足$i - S = w[i]$，移项可得$S = i - w[i]$时才可以满足要求。

注意到等式右边只与i本身有关，不妨设为$K[i]$，所以题目变成了查询S到T之间$K[i]$等于S的i的数量。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于60%的数据,除了以上数据，新增10%，有$1 \le n, m \le 99995$且树退化成链。

在链上肯定是要么往左要么往右，即$S \le T$或者$S > T$。先只考虑$S \le T$的情况，如果对于S到T之间的点i，要产生贡献的话，肯定满足$i - S = w[i]$，移项可得$S = i - w[i]$时才可以满足要求。

注意到等式右边只与i本身有关，不妨设为$K[i]$，所以题目变成了查询S到T之间$K[i]$等于S的i的数量。

因为题目只涉及到首和尾，我们可以很容易联想到差分，即对于S打上$+1$标记，T打上-1标记。

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

根据上述思路，我们考虑具体做法：对于每个点i,我们很容易发现只有从一个特定的点出发才有可能对i产生贡献。我们考虑维护一个统计数组A，$A[k]$表示的是处理到当前的结点时，从k出发的路径（而且还没有走到终点）有多少条。这样对于每个点i，我们只要查询一下所对应的$A[K[i]]$就可以了，根据上面的分析，这就是我们的答案了。

[BZOJ 4719][NOIP 2016]天天爱跑步

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \leq n, m \leq 299998$。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$。

路径依靠LCA拆分成2分段。

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$**。**

路径依靠LCA拆分成2分段。

在t时从x出发:

1 从下往上走:在i点能看到需满足

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$。

路径依靠LCA拆分成2分段。

在t时从x出发:

1. 从下往上走:在i点能看到需满足$w_i = t + dep[x] - dep[i]$,
   即$w_i + dep[i] = t + dep[x]$;

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$。

路径依靠LCA拆分成2分段。

在t时从x出发:

1. 从下往上走:在i点能看到需满足$w_i = t + dep[x] - dep[i]$,
   即$w_i + dep[i] = t + dep[x]$;
2. 从上往下走:在i点能看到需满足

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$。

路径依靠LCA拆分成2分段。

在t时从x出发:

1. 从下往上走:在i点能看到需满足$w_i = t + dep[x] - dep[i]$,
   即$w_i + dep[i] = t + dep[x]$;
2. 从上往下走:在i点能看到需满足$w_i = t - dep[x] + dep[i]$,
   即$w_i - dep[i] = t - dep[x]$。

只讨论从下往上走怎么处理(因为从上往下走处理方式同理):

[BZOJ 4719][NOIP 2016]天天爱跑步

对于100%的数据,有$1 \le n, m \le 299998$。

路径依靠LCA拆分成2分段。

在t时从x出发:

1. 从下往上走:在i点能看到需满足$w_i = t + dep[x] - dep[i]$,
   即$w_i + dep[i] = t + dep[x]$;
2. 从上往下走:在i点能看到需满足$w_i = t - dep[x] + dep[i]$,
   即$w_i - dep[i] = t - dep[x]$。

只讨论从下往上走怎么处理(因为从上往下走处理方式同理):

对树上$w_i + dep[i]$的值相同的点进行差分,进出的时候$+1, -1$即可。

基础练习

基础练习

结束啦

结束啦

欢迎提问

Questions are welcomed!