树上问题

Kloze

2023 年 12 月 14 日

- 树分治

- 树分治
- 树链剖分、长链剖分

- 树分治
- 树链剖分、长链剖分
- 树上莫队、树分块

- 树分治
- 树链剖分、长链剖分
- 树上莫队、树分块
- 生成树

[Ynoi2004] rpmtdq

给定一棵有边权的无根树，需要回答一些询问。

定义 $dist(i,j)$ 代表树上点 i 和点 j 之间的距离。

对于每一组询问，会给出 l,r，你需要输出 $min\ (dist(i,j))$ 其中 $l \leq i < j \leq r$。

$n \leq 2 \times 10^5, q \leq 10^6$

[Ynoi2004] rpmtdq

- 计算所有点对间的距离，做二维偏序

[Ynoi2004] rpmtdq

- 计算所有点对间的距离，做二维偏序
- 排除掉一些无用的点，当 $x_1 \leq x_2 \leq y_2 \leq y_1,, dist(x_1, y_1) > dist(x_2, y_2)$ 时，发现 (x_1, y_1) 是没有用的，我们不妨称 (x_2, y_2) 支配 (x_1, y_1)

[Ynoi2004] rpmtdq

- 计算所有点对间的距离，做二维偏序
- 排除掉一些无用的点，当 $x_1 \le x_2 \le y_2 \le y_1,, dist(x_1, y_1) > dist(x_2, y_2)$ 时，发现 (x_1, y_1) 是没有用的，我们不妨称 (x_2, y_2) 支配 (x_1, y_1)
- 称支配点对为不被任何点对支配的点对，一个支配点对 (i, j) 出现的必要条件时在分治中心 rt 处 i 是最大的满足 $i < j, dist(rt, i) \le dist(rt, j)$ 的点或最小的满足 $i > j, dist(rt, i) \le dist(rt, j)$ 的点。

[Ynoi2004] rpmtdq

- 计算所有点对间的距离，做二维偏序
- 排除掉一些无用的点，当 $x_1 \le x_2 \le y_2 \le y_1,, dist(x_1, y_1) > dist(x_2, y_2)$ 时，发现 (x_1, y_1) 是没有用的，我们不妨称 (x_2, y_2) 支配 (x_1, y_1)
- 称支配点对为不被任何点对支配的点对，一个支配点对 (i, j) 出现的必要条件时在分治中心 rt 处 i 是最大的满足 $i < j, dist(rt, i) \le dist(rt, j)$ 的点或最小的满足 $i > j, dist(rt, i) \le dist(rt, j)$ 的点。
- 在点分治时寻找这样的点对，由于每个 j 只会加入常数个 (i, j)，因此总点对数量是 $O(n \log n)$ 的，可以按编号排序后单调栈求解。

[Ynoi2004] rpmtdq

- 计算所有点对间的距离，做二维偏序
- 排除掉一些无用的点，当 $x_1 \le x_2 \le y_2 \le y_1,, dist(x_1, y_1) > dist(x_2, y_2)$ 时，发现 (x_1, y_1) 是没有用的，我们不妨称 (x_2, y_2) 支配 (x_1, y_1)
- 称支配点对为不被任何点对支配的点对，一个支配点对 (i, j) 出现的必要条件时在分治中心 rt 处 i 是最大的满足 $i < j, dist(rt, i) \le dist(rt, j)$ 的点或最小的满足 $i > j, dist(rt, i) \le dist(rt, j)$ 的点。
- 在点分治时寻找这样的点对，由于每个 j 只会加入常数个 (i, j)，因此总点对数量是 $O(n \log n)$ 的，可以按编号排序后单调栈求解。
- 二维数点即可，时间复杂度 $O(n \log^2 n + q \log n)$

CodeForces1019E Raining season

有 n 个结点的一棵树，每条边有两个权值 a, b，第 t 天经过第 i 条边花费时间 $a_i t + b_i$，给定 m，求 $t = 0, 1, 2, \ldots, m-1$ 时，最长的路径长度。

$n \leq 100,000,\ m \leq 1,000,000,\ 0 \leq a_i \leq 10^5,\ 0 \leq b_i \leq 10^9$

CodeForces1019E Raining season

- 一次函数权值：维护凸包

CodeForces1019E Raining season

- 一次函数权值：维护凸包
- 点分治合并凸包太复杂，考虑边分治，每次只需合并两个凸包并考虑两个子树路径相接

CodeForces1019E Raining season

- 一次函数权值：维护凸包
- 点分治合并凸包太复杂，考虑边分治，每次只需合并两个凸包并考虑两个子树路径相接
- 路径的权值是 $\sum a_i t + \sum b_i$ 的形式，dfs 即可得到两棵子树到根的所有路径权值函数，考虑如何合并路径

CodeForces1019E Raining season

- 一次函数权值：维护凸包
- 点分治合并凸包太复杂，考虑边分治，每次只需合并两个凸包并考虑两个子树路径相接
- 路径的权值是 $\sum a_i t + \sum b_i$ 的形式，dfs 即可得到两棵子树到根的所有路径权值函数，考虑如何合并路径
- 分别求出凸包后对位相加即可，然后合并两棵子树的凸包

CodeForces1019E Raining season

- 一次函数权值：维护凸包
- 点分治合并凸包太复杂，考虑边分治，每次只需合并两个凸包并考虑两个子树路径相接
- 路径的权值是 $\sum a_i t + \sum b_i$ 的形式，dfs 即可得到两棵子树到根的所有路径权值函数，考虑如何合并路径
- 分别求出凸包后对位相加即可，然后合并两棵子树的凸包
- 如何合并凸包：斜率递增，按斜率顺序从小到大归并，遇到线段相交求交点

CodeForces1019E Raining season

- 一次函数权值：维护凸包
- 点分治合并凸包太复杂，考虑边分治，每次只需合并两个凸包并考虑两个子树路径相接
- 路径的权值是 $\sum a_i t + \sum b_i$ 的形式，dfs 即可得到两棵子树到根的所有路径权值函数，考虑如何合并路径
- 分别求出凸包后对位相加即可，然后合并两棵子树的凸包
- 如何合并凸包：斜率递增，按斜率顺序从小到大归并，遇到线段相交求交点
- 当然也可以维护 (a_i, b_i) 的上凸壳。

城市

有 n 座城市分布在平面上，城市 i 坐落在点 $(\cos\frac{2\pi i}{n}, \sin\frac{2\pi i}{n})$ 上。每个城市和最近的两个城市有一条直线段的道路。

此外还有 $n-3$ 条道路，这些路之间不会相交，也不会和原有的道路重合。

通过每条道路均要花费 1 的时间。

每次给两个城市，问从一个城市到另一个城市最快需要多久。

$n \leq 100000$

城市

- 这是一个三角剖分图，可以将其转化为对偶图考虑，且每个点度数不会超过 3。

城市

- 这是一个三角剖分图，可以将其转化为对偶图考虑，且每个点度数不会超过 3。
- 对这棵树分治，u, v 至少经过一条把圆环切成两段，且 u, v 各在一段的路径，从每个分治点在原图上 BFS 并更新答案。

树

给定一棵 n 个点的带权树和正整数 K，求每个点到其它所有点到距离中第 K 大的数值。

$n \leq 50000, w_i \leq 10000$

树

- 对每个点 i 二分答案 w，问题转化为求到点 i 距离 $\leq w$ 的点有几个。

树

- 对每个点 i 二分答案 w，问题转化为求到点 i 距离 $\leq w$ 的点有几个。
- 构建点分树，设 $f[x]$ 表示以 x 为根的点分树中点到 x 的距离从小到大排序，设 $g[x]$ 表示以 x 为根的点分树

树

- 对每个点 i 二分答案 w，问题转化为求到点 i 距离 $\leq w$ 的点有几个。
- 构建点分树，设 $f[x]$ 表示以 x 为根的点分树中点到 x 的距离从小到大排序，设 $g[x]$ 表示以 x 为根的点分树
- 中点到 $fa[x]$ 的距离从小到大排序

树

- 对每个点 i 二分答案 w，问题转化为求到点 i 距离 $\leq w$ 的点有几个。
- 构建点分树，设 $f[x]$ 表示以 x 为根的点分树中点到 x 的距离从小到大排序，设 $g[x]$ 表示以 x 为根的点分树
- 中点到 $fa[x]$ 的距离从小到大排序
- 对于 i，访问其在点分树上的所有祖先，容斥后求和即可（该点新增 = 该点满足条件-父亲满足条件）

Codeforces757G Can Bash Save the Day

给出一棵 n 个点的带边权树，和一个排列 p_i，有 q 次操作，操作如下：

1. 给定 l, r, x，求 $\sum_{i=l}^{r} dist(p_i, x)$
2. 给定 x，交换 p_x 和 p_{x+1}

$n, q \leq 2 \times 10^5$，强制在线

Codeforces757G Can Bash Save the Day

- 考虑多次询问一个点到其它点的距离和怎么做，点分树上维护分治中心 x 的子树到 x 的距离和以及到 x 点分树上父亲的距离和，容斥计算即可。

Codeforces757G Can Bash Save the Day

- 考虑多次询问一个点到其它点的距离和怎么做，点分树上维护分治中心 x 的子树到 x 的距离和以及到 x 点分树上父亲的距离和，容斥计算即可。
- 对于区间查询，可以转变成 $1-(l-1)$ 和 $1-r$ 两个区间，并用主席树维护前缀的信息，每次加入点 p_i 的信息到点分树上的祖先。

Codeforces757G Can Bash Save the Day

- 考虑多次询问一个点到其它点的距离和怎么做，点分树上维护分治中心 x 的子树到 x 的距离和以及到 x 点分树上父亲的距离和，容斥计算即可。
- 对于区间查询，可以转变成 $1-(l-1)$ 和 $1-r$ 两个区间，并用主席树维护前缀的信息，每次加入点 p_i 的信息到点分树上的祖先。
- 考虑交换操作，由于是交换相邻两个，因此除第 x 棵线段树以外的线段树均不会改变，重新从第 $x-1$ 棵线段树继承信息，加入 p_{x+1} 即可。

Codeforces757G Can Bash Save the Day

- 考虑多次询问一个点到其它点的距离和怎么做，点分树上维护分治中心 x 的子树到 x 的距离和以及到 x 点分树上父亲的距离和，容斥计算即可。
- 对于区间查询，可以转变成 $1-(l-1)$ 和 $1-r$ 两个区间，并用主席树维护前缀的信息，每次加入点 p_i 的信息到点分树上的祖先。
- 考虑交换操作，由于是交换相邻两个，因此除第 x 棵线段树以外的线段树均不会改变，重新从第 $x-1$ 棵线段树继承信息，加入 p_{x+1} 即可。
- 复杂度 $O(n\log^2 n)$。

Codeforces757G Can Bash Save the Day

- 考虑多次询问一个点到其它点的距离和怎么做，点分树上维护分治中心 x 的子树到 x 的距离和以及到 x 点分树上父亲的距离和，容斥计算即可。
- 对于区间查询，可以转变成 $1-(l-1)$ 和 $1-r$ 两个区间，并用主席树维护前缀的信息，每次加入点 p_i 的信息到点分树上的祖先。
- 考虑交换操作，由于是交换相邻两个，因此除第 x 棵线段树以外的线段树均不会改变，重新从第 $x-1$ 棵线段树继承信息，加入 p_{x+1} 即可。
- 复杂度 $O(n\log^2 n)$。
- 当然还可以把求 lca 的深度和改成求链交，将 $1-p_x$ 路径上的点权 $+1$，查询时查询 $1-x$ 路径上的点权和即可。

「CTSC2018」暴力写挂

有两棵带边权的树，求
$depth_1(x)+depth_1(y)-(depth_1(LCA_1(x,y))+depth_2(LCA_2(x,y)))$
的最大值。

$n \leq 366666, |v| \leq 2017011328$

「CTSC2018」暴力写挂

- 边分治，由于分治边两端的连通块有一侧更靠近根，所以设靠近根的一侧为 A，另一侧为 B，则对 A 中的点 x，其与 B 中任意点 y 的 LCA 都相同，于是可以确定 $depth(x) - depth(LCA(x, y))$。

「CTSC2018」暴力写挂

- 边分治，由于分治边两端的连通块有一侧更靠近根，所以设靠近根的一侧为 A，另一侧为 B，则对 A 中的点 x，其与 B 中任意点 y 的 LCA 都相同，于是可以确定 $depth(x) - depth(LCA(x, y))$。
- 考虑剩下的 $depth_2(y) - depth_2(lca_2(x, y))$ 如何确定，可以在第二棵树上建虚树。我们对每个点 x 要求 $min_y depth_2(y) - depth_2(lca_2(x, y))$。

「CTSC2018」暴力写挂

- 边分治，由于分治边两端的连通块有一侧更靠近根，所以设靠近根的一侧为 A，另一侧为 B，则对 A 中的点 x，其与 B 中任意点 y 的 LCA 都相同，于是可以确定 $depth(x) - depth(LCA(x, y))$。
- 考虑剩下的 $depth_2(y) - depth_2(lca_2(x, y))$ 如何确定，可以在第二棵树上建虚树。我们对每个点 x 要求 $min_y depth_2(y) - depth_2(lca_2(x, y))$。
- 只需要从下往上、从上往下两次 dp 即可。建虚树的过程可以在开始时排一遍序，LCA 可以用倍增 $O(n \log n)$ 预处理然后 $O(1)$ 求。

「CTSC2018」暴力写挂

- 边分治，由于分治边两端的连通块有一侧更靠近根，所以设靠近根的一侧为 A，另一侧为 B，则对 A 中的点 x，其与 B 中任意点 y 的 LCA 都相同，于是可以确定 $depth(x) - depth(LCA(x, y))$。
- 考虑剩下的 $depth_2(y) - depth_2(lca_2(x, y))$ 如何确定，可以在第二棵树上建虚树。我们对每个点 x 要求 $min_y depth_2(y) - depth_2(lca_2(x, y))$。
- 只需要从下往上、从上往下两次 dp 即可。建虚树的过程可以在开始时排一遍序，LCA 可以用倍增 $O(n \log n)$ 预处理然后 $O(1)$ 求。
- 时间复杂度 $O(n \log n)$

[WC2018] 通道

有三棵带边权的树，求：

$\max_{u,v} dist_1(u,v) + dist_2(u,v) + dist_3(u,v)$

$n \leq 10^5$

[WC2018] 通道

- 边分治第一棵树，距离变成 $d_1(u)+d_1(v)+dist_2(u,v)+dist_3(u,v)+w_{edge}$，$u,v$ 要求在分治边两侧

[WC2018] 通道

- 边分治第一棵树，距离变成 $d_1(u) + d_1(v) + dist_2(u, v) + dist_3(u, v) + w_{edge}$，$u, v$ 要求在分治边两侧
- 考虑分治边两边的点，在第二棵树上建虚树，复杂度允许我们在虚树上 dfs

[WC2018] 通道

- 边分治第一棵树，距离变成 $d_1(u)+d_1(v)+dist_2(u,v)+dist_3(u,v)+w_{edge}$，$u,v$ 要求在分治边两侧
- 考虑分治边两边的点，在第二棵树上建虚树，复杂度允许我们在虚树上 dfs
- 枚举 u,v 在第二棵树上的 $lca=w$，则距离变为 $d_1(u)+d_1(v)+d_2(u)+d_2(v)-2d_2(w)+dist_3(u,v)+w=val_u+val_v+w'+dist_3(u,v)$

[WC2018] 通道

- 边分治第一棵树，距离变成 $d_1(u)+d_1(v)+dist_2(u,v)+dist_3(u,v)+w_{edge}$，$u,v$ 要求在分治边两侧
- 考虑分治边两边的点，在第二棵树上建虚树，复杂度允许我们在虚树上 dfs
- 枚举 u,v 在第二棵树上的 $lca=w$，则距离变为 $d_1(u)+d_1(v)+d_2(u)+d_2(v)-2d_2(w)+dist_3(u,v)+w=val_u+val_v+w'+dist_3(u,v)$
- 注意到这个式子实际上是在 T_3 上求直径，可以在 dfs T_2 的时候实现直径合并即可，注意考虑端点颜色需要不同

[WC2018] 通道

- 边分治第一棵树，距离变成 $d_1(u)+d_1(v)+dist_2(u,v)+dist_3(u,v)+w_{edge}$，$u,v$ 要求在分治边两侧
- 考虑分治边两边的点，在第二棵树上建虚树，复杂度允许我们在虚树上 dfs
- 枚举 u,v 在第二棵树上的 $lca=w$，则距离变为 $d_1(u)+d_1(v)+d_2(u)+d_2(v)-2d_2(w)+dist_3(u,v)+w=val_u+val_v+w'+dist_3(u,v)$
- 注意到这个式子实际上是在 T_3 上求直径，可以在 dfs T_2 的时候实现直径合并即可，注意考虑端点颜色需要不同
- 复杂度 $O(n\log^2 n)$

「JOISC 2020 Day4」首都城市

有一棵 n 个点的树，共 k 个颜色，点有颜色。可以进行若干次操作，将所有颜色为 a 的点变为颜色 b，要求出存在某种颜色使得该种颜色的所有点在树上构成联通块的最小操作次数，

$n, k \leq 10^5$

「JOISC 2020 Day4」首都城市

- 考虑颜色 i 的点构成的虚树，如果虚树某条边上有其它颜色 j，建边 $i->j$

「JOISC 2020 Day4」首都城市

- 考虑颜色 i 的点构成的虚树，如果虚树某条边上有其它颜色 j，建边 $i->j$
- 在新图上求强连通分量，输出最小的分量大小即可

「JOISC 2020 Day4」首都城市

- 考虑颜色 i 的点构成的虚树，如果虚树某条边上有其它颜色 j，建边 $i->j$
- 在新图上求强连通分量，输出最小的分量大小即可
- 用树链剖分优化连边过程

一棵 n 个点的树，节点 i 有权值 v_i，有 m 次操作，操作如下：

1. 把 v_x 改为 y

2. 选择连通块使得点权和最大（也可以不选择），输出最大的点权和

$n, m \leq 10^5, |v| \leq 1000$

- 暴力 $f_x = v_x + \sum_{y \in son_x} max\{f_y, 0\}$

- 暴力 $f_x = v_x + \sum_{y \in son_x} max\{f_y, 0\}$
- 一条链怎么做？线段树合并区间信息，ls, rs 表示从左边界开始最大和，和从右边界开始最大和

- 暴力 $f_x = v_x + \sum_{y \in son_x} max\{f_y, 0\}$
- 一条链怎么做？线段树合并区间信息，ls, rs 表示从左边界开始最大和，和从右边界开始最大和
- 树链剖分上树，需要合并所有轻儿子的 dp 值

- 暴力 $f_x = v_x + \sum_{y \in son_x} max\{f_y, 0\}$
- 一条链怎么做？线段树合并区间信息，ls, rs 表示从左边界开始最大和，和从右边界开始最大和
- 树链剖分上树，需要合并所有轻儿子的 dp 值
- 维护修改，更新 fa[top]，值为新贡献-旧贡献

LOJ6289. 花朵

有一棵 n 个点的树，每个点有点权 b_i，一种选择方法合法定义为选出的点集 S 大小恰好为 M，且 S 是一个独立集。

选择方案的权重为 $\prod_{i\in S} b_i$，求出所有合法的选择方案权值之和，答案对 998244353 取模。

$M \leq N \leq 8 \times 10^4$

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.
- 当 x 与其儿子合并时就是一个卷积的形式：
  $f[x][i] * g[son_x][j] \to f'[x][i+j]$

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.
- 当 x 与其儿子合并时就是一个卷积的形式：
  $f[x][i] * g[son_x][j] \to f'[x][i+j]$
- 初值条件：$f[x][1] = b_x, g[x][0] = 1$

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.
- 当 x 与其儿子合并时就是一个卷积的形式：
  $f[x][i] * g[son_x][j] \to f'[x][i+j]$
- 初值条件：$f[x][1] = b_x, g[x][0] = 1$
- 如果树退化成链，分治 NTT 即可解决。

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.
- 当 x 与其儿子合并时就是一个卷积的形式：
  $f[x][i] * g[son_x][j] \to f'[x][i+j]$
- 初值条件：$f[x][1] = b_x, g[x][0] = 1$
- 如果树退化成链，分治 NTT 即可解决。
- 链扩展到树：树链剖分，对于一条重链来说，先求出该重链中每个点轻儿子为根的多项式 f, g，然后对于重链中每个点都将其轻儿子与该点合并.

LOJ6289. 花朵

- 先考虑暴力，令 $f[x][i], g[x][i]$ 分别表示点 x 选/不选的情况下 S 大小为 i 的权值乘积之和.
- 当 x 与其儿子合并时就是一个卷积的形式：
  $f[x][i] * g[son_x][j] \to f'[x][i+j]$
- 初值条件：$f[x][1] = b_x, g[x][0] = 1$
- 如果树退化成链，分治 NTT 即可解决。
- 链扩展到树：树链剖分，对于一条重链来说，先求出该重链中每个点轻儿子为根的多项式 f, g，然后对于重链中每个点都将其轻儿子与该点合并.
- 合并轻边和重链上的信息用分治 NTT 解决。

icpc2017 西安 D Islands

给一个 n 个点的树，每条边可以断开，也可以不断开，求断边方案数使得和 1 联通的联通块大小为 $k = 1, 2, 3, \ldots, n$ 的方案，答案对 1811939329 取模。

- 同理，令 $f[k][i]$ 表示 k 向父亲所在对联通块贡献的节点数。

- 同理，令 $f[k][i]$ 表示 k 向父亲所在对联通块贡献的节点数。
- 有多项式 $f_k[x^n]$，$f_k[x^0] = 2^{sz_k-1}$

- 同理，令 $f[k][i]$ 表示 k 向父亲所在对联通块贡献的节点数。
- 有多项式 $f_k[x^n]$，$f_k[x^0] = 2^{sz_k-1}$
- 同样分治 NTT 进行轻边信息的合并，对于重链，分治进行一个类似矩阵乘法的转移即可。

POI2014 Hotel

给一棵树，求有多少组点满足 a, b, c 互不相同，

$dist_{a,b} = dist_{b,c} = dist_{a,c}$

$n \leq 10^5$

POI2014 Hotel

- 存在一个中心点使得每个点到它的距离相同。

POI2014 Hotel

- 存在一个中心点使得每个点到它的距离相同。
- 先考虑其中两个点 a, b 的 lca，设它们到 lca 的距离为 i，则记录 $f[x][i]$ 表示 x 的子树中距 x 为 i 的点的个数。

POI2014 Hotel

- 存在一个中心点使得每个点到它的距离相同。
- 先考虑其中两个点 a, b 的 lca，设它们到 lca 的距离为 i，则记录 $f[x][i]$ 表示 x 的子树中距 x 为 i 的点的个数。
- 考虑合并后把第三个点加上去：$g[x][i]$ 表示现在在 x 点，在其子树中已经完成了 a, b 的选取，现在还需要找出一个距离 x 为 i 的节点。

POI2014 Hotel

- 存在一个中心点使得每个点到它的距离相同。
- 先考虑其中两个点 a, b 的 lca，设它们到 lca 的距离为 i，则记录 $f[x][i]$ 表示 x 的子树中距 x 为 i 的点的个数。
- 考虑合并后把第三个点加上去：$g[x][i]$ 表示现在在 x 点，在其子树中已经完成了 a, b 的选取，现在还需要找出一个距离 x 为 i 的节点。
- 那么答案就是
  $ans+ = f[x][i] \times g[y][i+1] + g[x][i] \times f[y][i-1]$

POI2014 Hotel

- 存在一个中心点使得每个点到它的距离相同。
- 先考虑其中两个点 a, b 的 lca，设它们到 lca 的距离为 i，则记录 $f[x][i]$ 表示 x 的子树中距 x 为 i 的点的个数。
- 考虑合并后把第三个点加上去：$g[x][i]$ 表示现在在 x 点，在其子树中已经完成了 a, b 的选取，现在还需要找出一个距离 x 为 i 的节点。
- 那么答案就是
  $ans+ = f[x][i] \times g[y][i+1] + g[x][i] \times f[y][i-1]$
- 转移就是 $g[x][i]+ = f[x][i+1] * f[y][i], f[x][i+1]+ = f[y][i], g[x][i-1]+ = g[x][i]$

POI2014 Hotel

- 暴力 $O(n^2)$，与深度有关的问题用长链剖分优化。

POI2014 Hotel

- 暴力 $O(n^2)$，与深度有关的问题用长链剖分优化。
- 对每个点选择深度最大的儿子作为重儿子，继承重儿子 dp 数组，并暴力合并所有轻儿子的信息。

POI2014 Hotel

- 暴力 $O(n^2)$，与深度有关的问题用长链剖分优化。
- 对每个点选择深度最大的儿子作为重儿子，继承重儿子 dp 数组，并暴力合并所有轻儿子的信息。
- 每个点只会在重链顶端贡献复杂度，所以复杂度 $O(n)$

POI2014 Hotel

- 暴力 $O(n^2)$，与深度有关的问题用长链剖分优化。
- 对每个点选择深度最大的儿子作为重儿子，继承重儿子 dp 数组，并暴力合并所有轻儿子的信息。
- 每个点只会在重链顶端贡献复杂度，所以复杂度 $O(n)$
- 一些长链剖分的性质：从某个点向其祖先跳，跳到的点所在长链一定大于当前长链。

Another Boring Problem

给定一棵 n 个点的树，点有颜色 c_i，每次给定 x, y, k，请你求出树上 (x, y) 路径出现次数第 k 多的颜色的出现次数。

$n, q \leq 10^5$

Another Boring Problem

- 树上莫队，求出树的 dfs 时的括号序列，在到达点、退出点时将其加入到序列。

Another Boring Problem

- 树上莫队，求出树的 dfs 时的括号序列，在到达点、退出点时将其加入到序列。
- 设每个点有两个出现位置 s_x, t_x，则查询区间相当于 $[t_x, s_y]$

Another Boring Problem

- 树上莫队，求出树的 dfs 时的括号序列，在到达点、退出点时将其加入到序列。
- 设每个点有两个出现位置 s_x, t_x，则查询区间相当于 $[t_x, s_y]$
- 注意处理一下 *lca* 的颜色即可，剩下的问题就是序列上的莫队了。

有 n 个点，点有点权 a_i，点 i 被选中会产生贡献 $f_d(a_i^2)$，点 i 和 j 同时被选中会产生贡献 $f_d(a_i a_j)$。

设 $z = \prod_i p_i^{k_i}$，定义 $f_d(z) = \prod (-1)^{k_i}[k_i \le d]$。

对每个点，求其及其子树中的点若同时被选中的贡献和。

$1 \le n \le 2 \times 10^5, 1 \le d \le 20$，$a_i$ 构成一个 1 到 n 的排列

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。
- $f_d(ij) = f(i)f(j)f_d^2(ij), f(i) := f_{\inf}(i)$

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。
- $f_d(ij) = f(i)f(j)f_d^2(ij), f(i) := f_{\inf}(i)$
- 考虑 $f_d^2(ij) = 1$ 即判断 ij 是否有大于 d 次的因子。

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。
- $f_d(ij) = f(i)f(j)f_d^2(ij), f(i) := f_{\inf}(i)$
- 考虑 $f_d^2(ij) = 1$ 即判断 ij 是否有大于 d 次的因子。
- 设 $g(k)$ 为 k 的最大的 d 次因子，则 $f_d^2(ij) = 1 \leftrightarrow g(k) = 1$

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。
- $f_d(ij) = f(i)f(j)f_d^2(ij), f(i) := f_{\inf}(i)$
- 考虑 $f_d^2(ij) = 1$ 即判断 ij 是否有大于 d 次的因子。
- 设 $g(k)$ 为 k 的最大的 d 次因子，则 $f_d^2(ij) = 1 \leftrightarrow g(k) = 1$
- 我们可以进行代换
  $[g(k) = 1] = \sum_{p|g(k)} \mu(p) = \sum_{p^{d+1}|k} \mu(p)$

- 先考虑一棵菊花树（求 $\sum_i \sum_j f_d(ij)$）怎么做
- 将 $f_d(k)$ 点值分成两部分，一部分是是否为 0，另一部分是正负号。
- $f_d(ij) = f(i)f(j)f_d^2(ij), f(i) := f_{\inf}(i)$
- 考虑 $f_d^2(ij) = 1$ 即判断 ij 是否有大于 d 次的因子。
- 设 $g(k)$ 为 k 的最大的 d 次因子，则 $f_d^2(ij) = 1 \leftrightarrow g(k) = 1$
- 我们可以进行代换
  $[g(k) = 1] = \sum_{p|g(k)} \mu(p) = \sum_{p^{d+1}|k} \mu(p)$
- 于是答案为 $\sum_i \sum_j f(i)f(j) \sum_{p^{d+1}|ij} \mu(p)$

- 考虑 i 的贡献为 $f(i)\sum_p \mu(p)\sum_j [\frac{p^{d+1}}{(p^{d+1},i)}|j]f(j)$

- 考虑 i 的贡献为 $f(i)\sum_p \mu(p)\sum_j [\frac{p^{d+1}}{(p^{d+1},i)}|j]f(j)$
- 当 $p \nmid i$ 时，$\frac{p^{d+1}}{(p^{d+1},i)}$ 必然包含素数的 $d+1$ 次幂，即 $f(j)=0$

- 考虑 i 的贡献为 $f(i)\sum_p \mu(p)\sum_j [\frac{p^{d+1}}{(p^{d+1},i)}|j]f(j)$
- 当 $p \nmid i$ 时，$\frac{p^{d+1}}{(p^{d+1},i)}$ 必然包含素数的 $d+1$ 次幂，即 $f(j)=0$
- 因此对 p 只需要枚举所有 i 的因数即可，复杂度 $O(n\ln n)$

- 考虑 i 的贡献为 $f(i)\sum_p \mu(p)\sum_j [\frac{p^{d+1}}{(p^{d+1},i)}|j]f(j)$
- 当 $p \nmid i$ 时，$\frac{p^{d+1}}{(p^{d+1},i)}$ 必然包含素数的 $d+1$ 次幂，即 $f(j)=0$
- 因此对 p 只需要枚举所有 i 的因数即可，复杂度 $O(n\ln n)$
- 当然这个贡献取负号就是删除，用树上莫队可以做到 $O(n\ln n\sqrt{n})$ 的复杂度。

- 考虑 i 的贡献为 $f(i)\sum_p \mu(p)\sum_j [\frac{p^{d+1}}{(p^{d+1},i)}|j]f(j)$
- 当 $p \nmid i$ 时，$\frac{p^{d+1}}{(p^{d+1},i)}$ 必然包含素数的 $d+1$ 次幂，即 $f(j)=0$
- 因此对 p 只需要枚举所有 i 的因数即可，复杂度 $O(n\ln n)$
- 当然这个贡献取负号就是删除，用树上莫队可以做到 $O(n\ln n\sqrt{n})$ 的复杂度。
- 进一步优化，树链剖分（启发式合并）加入点即可，继承重子树的信息，复杂度 $O(n\ln n\log n)$

Codeforces 888G 最小异或生成树

n 个点，每个点有点权 a_i，u, v 间的边边权为 a_{uv}，求最小生成树。

$n \leq 10^5, a_i \leq 10^9$

Codeforces 888G 最小异或生成树

- 考虑 Boruvka 算法求解最小生成树的过程，每一个连通块都选择连出的最小边。

Codeforces 888G 最小异或生成树

- 考虑 Boruvka 算法求解最小生成树的过程，每一个连通块都选择连出的最小边。
- 在 trie 树上边权最小的必要条件是 LCA 最深，因此可以在 trie 树上考虑这个问题，连通块一定是 trie 的某个子树。

Codeforces 888G 最小异或生成树

- 考虑 Boruvka 算法求解最小生成树的过程，每一个连通块都选择连出的最小边。
- 在 trie 树上边权最小的必要条件是 LCA 最深，因此可以在 trie 树上考虑这个问题，连通块一定是 trie 的某个子树。
- 于是可以分别让左子树联通，让右子树联通，最后考虑合并左右子树的联通块。

Codeforces 888G 最小异或生成树

- 考虑 Boruvka 算法求解最小生成树的过程，每一个连通块都选择连出的最小边。
- 在 trie 树上边权最小的必要条件是 LCA 最深，因此可以在 trie 树上考虑这个问题，连通块一定是 trie 的某个子树。
- 于是可以分别让左子树联通，让右子树联通，最后考虑合并左右子树的联通块。
- 启发式合并，选择小的子树在大的子树上查询即可，复杂度 $O(n \log^2 n)$

Rectangle

有一个 $n \times n$ 的矩阵 A，开始时 A 的每个元素都为 0。共有 m 次操作，每次选中 A 的一个子矩阵并将其中的数都增加 w。操作后由 A 构造一个 n 个点的无向图，边 (i, j) 的边权为 $A_{i,j}$。求该图最小生成树的边权和。$1 \leq n, m \leq 100000, -10^6 \leq w \leq 10^6$

Rectangle

- 注意到 Boruvka 算法求最小生成树只需要 $O(\log n)$ 轮连边。

Rectangle

- 注意到 Boruvka 算法求最小生成树只需要 $O(\log n)$ 轮连边。
- 对于每一轮来说，线段树 + 扫描线求出权值矩阵的第 i 行，同时维护区间的最小值及与最小值不同连通块的最小值，依照该信息连边即可。

Codeforces 632F Swimmers in the Pool

给一个 $n \times n$ 的矩阵 a，需要判断矩阵是否满足以下三个条件：1.$a_{i,j} = a_{j,i}$ 2.$a_{i,j} = 0$ 3. 对任意 i, j, k，有 $a_{i,j} \leq max(a_{i,k}, a_{j,k})$ 如果满足输出 MAGIC，否则输出 NOT MAGIC。$n \leq 2.5 \times 10^3, a_{i,j} \leq 10^9$

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$
- 同时，取 $k = i$，有 $b_{i,j}$ 满足 $b_{i,j} \leq a_{i,j}$，因此即需要 $b_{i,j} = a_{i,j}$

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$
- 同时，取 $k = i$，有 $b_{i,j}$ 满足 $b_{i,j} \leq a_{i,j}$，因此即需要 $b_{i,j} = a_{i,j}$
- 可以归纳 $b_{i,j}$ 的计算过程得出 $b_{i,j}$ 即 i 到 j 的所有路径中最大边权的最小值。

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$
- 同时，取 $k = i$，有 $b_{i,j}$ 满足 $b_{i,j} \leq a_{i,j}$，因此即需要 $b_{i,j} = a_{i,j}$
- 可以归纳 $b_{i,j}$ 的计算过程得出 $b_{i,j}$ 即 i 到 j 的所有路径中最大边权的最小值。
- 考虑 kruskal 的过程，每次加入权值最小的边，则 i 到 j 的所有路径的最大边权的最小值一定在 kruskal 生成树中。

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$
- 同时，取 $k = i$，有 $b_{i,j}$ 满足 $b_{i,j} \leq a_{i,j}$，因此即需要 $b_{i,j} = a_{i,j}$
- 可以归纳 $b_{i,j}$ 的计算过程得出 $b_{i,j}$ 即 i 到 j 的所有路径中最大边权的最小值。
- 考虑 kruskal 的过程，每次加入权值最小的边，则 i 到 j 的所有路径的最大边权的最小值一定在 kruskal 生成树中。
- 换句话说最小生成树一定是一种最小瓶颈生成树。

Codeforces 632F Swimmers in the Pool

- 1 和 2 条件实际上保证了矩阵 a 是一个邻接矩阵，可以将问题转化到图上。
- 实际上边从小到大排序后 bitset 优化能过。
- 考虑条件 $a_{i,j} \leq \min_{k=1}^{n} max(a_{i,k}, a_{j,k}) = b_{i,j}$
- 同时，取 $k = i$，有 $b_{i,j}$ 满足 $b_{i,j} \leq a_{i,j}$，因此即需要 $b_{i,j} = a_{i,j}$
- 可以归纳 $b_{i,j}$ 的计算过程得出 $b_{i,j}$ 即 i 到 j 的所有路径中最大边权的最小值。
- 考虑 kruskal 的过程，每次加入权值最小的边，则 i 到 j 的所有路径的最大边权的最小值一定在 kruskal 生成树中。
- 换句话说最小生成树一定是一种最小瓶颈生成树。
- 求最小生成树后判断两点路径的最小边权是否等于 $a_{i,j}$ 即可，复杂度 $O(n^2 \log n)$

LOJ574. 黄金矿工

有一棵 n 个点的带边权的有向树，边的方向从父亲指向儿子，树上某些点有黄金和矿工，黄金有权值 p_x，矿工有权值 q_y，矿工 y 挖到黄金 x 的收益是 $p_x + q_y + \sum_i c_i$，其中 $\sum_i c_i$ 是树上路径的边权和。有 q 次操作，每次添加黄金或添加矿工，每次询问后求最大收益和。

$n, q \leq 10^5$，所有权值均为正整数。

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。
- 考虑贡献可以表示为 $w_x + depth_x + w_y - depth_y$，所以不妨将黄金和矿工的权看作 v_x, v_y，黄金和矿工匹配会产生 $v_x + v_y$ 的贡献。

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。
- 考虑贡献可以表示为 $w_x + depth_x + w_y - depth_y$，所以不妨将黄金和矿工的权看作 v_x, v_y，黄金和矿工匹配会产生 $v_x + v_y$ 的贡献。
- 考虑没有修改怎么做，反悔贪心（模拟费用流），提供一个反悔项（黄金反悔）$v''_x = v_x - v_y$，每次匹配后加入可并堆，从叶子到根向上合并即可，复杂度 $O(qn \log n)$

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。
- 考虑贡献可以表示为 $w_x + depth_x + w_y - depth_y$，所以不妨将黄金和矿工的权看作 v_x, v_y，黄金和矿工匹配会产生 $v_x + v_y$ 的贡献。
- 考虑没有修改怎么做，反悔贪心（模拟费用流），提供一个反悔项（黄金反悔）$v''_x = v_x - v_y$，每次匹配后加入可并堆，从叶子到根向上合并即可，复杂度 $O(qn \log n)$
- 再考虑加入矿工，跑完可并堆之后我们有了一个流过一部分流量到网络，这个时候从加入到矿工开始可以进行增广，且由于向下的边容量是 $+\inf$，于是向上走到最浅的祖先后，其子树中的黄金一定是都能匹配的，接下来是一个查询最值的问题，树剖 + 线段树维护即可。

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。
- 考虑贡献可以表示为 $w_x + depth_x + w_y - depth_y$，所以不妨将黄金和矿工的权看作 v_x, v_y，黄金和矿工匹配会产生 $v_x + v_y$ 的贡献。
- 考虑没有修改怎么做，反悔贪心（模拟费用流），提供一个反悔项（黄金反悔）$v''_x = v_x - v_y$，每次匹配后加入可并堆，从叶子到根向上合并即可，复杂度 $O(qn \log n)$
- 再考虑加入矿工，跑完可并堆之后我们有了一个流过一部分流量到网络，这个时候从加入到矿工开始可以进行增广，且由于向下的边容量是 $+\inf$，于是向上走到最浅的祖先后，其子树中的黄金一定是都能匹配的，接下来是一个查询最值的问题，树剖 + 线段树维护即可。
- 最后加入黄金，同样地，我们维护能到达 x 的最大权值，用堆维护轻儿子的最值，在重链上用线段树维护最值即可。

LOJ574. 黄金矿工

- 显然这是一个费用流模型，如何优化它是需要考虑的重点。
- 考虑贡献可以表示为 $w_x + depth_x + w_y - depth_y$，所以不妨将黄金和矿工的权看作 v_x, v_y，黄金和矿工匹配会产生 $v_x + v_y$ 的贡献。
- 考虑没有修改怎么做，反悔贪心（模拟费用流），提供一个反悔项（黄金反悔）$v''_x = v_x - v_y$，每次匹配后加入可并堆，从叶子到根向上合并即可，复杂度 $O(qn \log n)$
- 再考虑加入矿工，跑完可并堆之后我们有了一个流过一部分流量到网络，这个时候从加入到矿工开始可以进行增广，且由于向下的边容量是 $+\inf$，于是向上走到最浅的祖先后，其子树中的黄金一定是都能匹配的，接下来是一个查询最值的问题，树剖 + 线段树维护即可。
- 最后加入黄金，同样地，我们维护能到达 x 的最大权值，用堆维护轻儿子的最值，在重链上用线段树维护最值即可。
- 复杂度 $O(q \log^2 n)$

JOISC 2020 Day3 收获

IOI 农场是一个种植苹果的农场，以位于一个巨大的环形湖周边而闻名。在 IOI 农场，共有 N 个员工，从 1 到 N 标号。共有 M 棵苹果树，从 1 到 M 标号。湖的周长为 L 米。在初始时刻，第 i 位员工站在离湖最北端顺时针 A_i m 米的位置，第 j 棵苹果树在离湖最北端顺时针 B_i m 的位置。保证这 $N+M$ 个整数 A_i, B_i 互不相同。由于 IOI 农场苹果树是经过改良的特殊品种，一棵树同时只能结一个苹果。同时，如果一棵树上的苹果被摘掉了，在恰好 C 秒后会长出一个苹果。在初始时刻，每棵树上都有一个苹果，同时每个员工开始沿着顺时针方向移动。员工的移动速度是 1m/s。如果一个员工在某一时刻到达了一颗长有苹果的苹果树，他会摘掉这个苹果 (如果在到达时恰好长出苹果，员工也会摘掉)。这里我们忽略员工摘苹果的时间。有 Q 个问题，第 k 个问题的形式如下：询问前 T_k s 中，第 V_k 个员工一共收获了多少个苹果 (注意包含第 T_s 末收获的苹果)。请编写一个程序来回答这些询问。$1 \leq N, M, Q \leq 2 \times 10^5$，其余 10^9 范围

JOISC 2020 Day3 收获

- 考虑到 C 是定值，那么当第 x 个人拿了苹果后，下一个拿该苹果的人是确定的，设为 p_i，则连边 $i \to p_i$，这样可以得到一棵基环内向树，这棵树的边权 w_i 是两个人拿到苹果的时间差。

JOISC 2020 Day3 收获

- 考虑到 C 是定值，那么当第 x 个人拿了苹果后，下一个拿该苹果的人是确定的，设为 p_i，则连边 $i \to p_i$，这样可以得到一棵基环内向树，这棵树的边权 w_i 是两个人拿到苹果的时间差。
- 对于不在环上的人，考虑对每棵苹果树处理出 (v_0, t_0) 表示 t_0 时刻第 v_0 个人第一次摘下该苹果。那么这种情况的答案就是 v_k 子树中 $t_0 + dist(v_0, v_k) \le t_k$ 的苹果数量。转化一下就是 $t_0 + depth_{v_0} \le t_k + depth_{v_k}$，可以二维数点。

JOISC 2020 Day3 收获

- 考虑到 C 是定值，那么当第 x 个人拿了苹果后，下一个拿该苹果的人是确定的，设为 p_i，则连边 $i \to p_i$，这样可以得到一棵基环内向树，这棵树的边权 w_i 是两个人拿到苹果的时间差。
- 对于不在环上的人，考虑对每棵苹果树处理出 (v_0, t_0) 表示 t_0 时刻第 v_0 个人第一次摘下该苹果。那么这种情况的答案就是 v_k 子树中 $t_0 + dist(v_0, v_k) \le t_k$ 的苹果数量。转化一下就是 $t_0 + depth_{v_0} \le t_k + depth_{v_k}$，可以二维数点。
- 对于在环上的人，考虑对每棵苹果树处理出 (v_0, t_0) 表示在 t_0 时刻环上的点 v_0 第一次摘下了这个苹果，可以倍增求出。于是贡献为 $\max(0, \lfloor \frac{t - t_0 + (s_{vk} - s_{v0})}{\sum_{i \ on \ path} w_i} \rfloor) + 1$，整理可得形式 $\lfloor \frac{a_{vk} - a_{v0}}{\sum_{i \ on \ path} w_i} \rfloor = q_{vk} - q_{v0} + [r_{vk} \ge r_{v0}]$，对前半部分可以求和算出，对后半部分就是一个二维数点 (r 和 q 的大小要求)。

JOISC 2020 Day3 收获

- 考虑到 C 是定值，那么当第 x 个人拿了苹果后，下一个拿该苹果的人是确定的，设为 p_i，则连边 $i \to p_i$，这样可以得到一棵基环内向树，这棵树的边权 w_i 是两个人拿到苹果的时间差。
- 对于不在环上的人，考虑对每棵苹果树处理出 (v_0, t_0) 表示 t_0 时刻第 v_0 个人第一次摘下该苹果。那么这种情况的答案就是 v_k 子树中 $t_0 + dist(v_0, v_k) \le t_k$ 的苹果数量。转化一下就是 $t_0 + depth_{v_0} \le t_k + depth_{v_k}$，可以二维数点。
- 对于在环上的人，考虑对每棵苹果树处理出 (v_0, t_0) 表示在 t_0 时刻环上的点 v_0 第一次摘下了这个苹果，可以倍增求出。于是贡献为 $\max(0, \lfloor \frac{t - t_0 + (s_{vk} - s_{v0})}{\sum_{i \ on \ path} w_i} \rfloor) + 1$，整理可得形式 $\lfloor \frac{a_{vk} - a_{v0}}{\sum_{i \ on \ path} w_i} \rfloor = q_{vk} - q_{v0} + [r_{vk} \ge r_{v0}]$，对前半部分可以求和算出，对后半部分就是一个二维数点 (r 和 q 的大小要求)。
- 复杂度 $O(n \log n)$