树上水题选讲

lizongru

2018 年 10 月 19 日

- 由于讲课人水平有限，所以本次讲课题目较水，而且并没有超纲知识。

- 由于讲课人水平有限，所以本次讲课题目较水，而且并没有超纲知识。
- 所以请大家多多秒题，可以更快讲完。

- 由于讲课人水平有限，所以本次讲课题目较水，而且并没有超纲知识。
- 所以请大家多多秒题，可以更快讲完。
- 欢迎大家举报身边的同学。

CF916E

- 给出一棵树以及每个结点的权值，初始时根为 1。有 3 种操作：

CF916E

- 给出一棵树以及每个结点的权值，初始时根为 1。有 3 种操作：
- 1、是将根换为 x；

CF916E

- 给出一棵树以及每个结点的权值，初始时根为 1。有 3 种操作：
- 1、是将根换为 x；
- 2、是给出两个节点 u,v，把这两个点的 LCA 的子树中每个节点权值加上 x；

CF916E

- 给出一棵树以及每个结点的权值，初始时根为 1。有 3 种操作：
- 1、是将根换为 x；
- 2、是给出两个节点 u,v，把这两个点的 LCA 的子树中每个节点权值加上 x；
- 3、是查询以 u 为根的权值和。

CF916E

- 给出一棵树以及每个结点的权值，初始时根为 1。有 3 种操作:
- 1、是将根换为 x;
- 2、是给出两个节点 u,v，把这两个点的 LCA 的子树中每个节点权值加上 x;
- 3、是查询以 u 为根的权值和。
- $1 \leq n \leq 10^5, 1 \leq q \leq 10^5$

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。
- 对于第一种操作：我们直接将现在的根记录一下。

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。
- 对于第一种操作：我们直接将现在的根记录一下。
- 对于第二种操作：设现在的根为 rt，可以发现 u,v 在现在这棵树中的 lca 总是原树 lca(rt, u), lca(rt, v), lca(u, v) 中的最小值。然后修改即可。

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。
- 对于第一种操作：我们直接将现在的根记录一下。
- 对于第二种操作：设现在的根为 rt，可以发现 u,v 在现在这棵树中的 lca 总是原树 lca(rt, u), lca(rt, v), lca(u, v) 中的最小值。然后修改即可。
- 对于第三种操作：

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。
- 对于第一种操作：我们直接将现在的根记录一下。
- 对于第二种操作：设现在的根为 rt，可以发现 u,v 在现在这棵树中的 lca 总是原树 lca(rt, u), lca(rt, v), lca(u, v) 中的最小值。然后修改即可。
- 对于第三种操作：
- 1、若现在的根不在 u 的自树中，则没有影响，直接输出即可。

Solution

- 没有换根的情况，就是一个裸的 dfs 序 + 线段树 +lca，而这种换根的题，套路都是分类讨论。
- 对于第一种操作：我们直接将现在的根记录一下。
- 对于第二种操作：设现在的根为 rt，可以发现 u,v 在现在这棵树中的 lca 总是原树 lca(rt, u), lca(rt, v), lca(u, v) 中的最小值。然后修改即可。
- 对于第三种操作：
- 1、若现在的根不在 u 的自树中，则没有影响，直接输出即可。
- 2、否则设现在的根到 u 的路径上最近的节点为 b，答案为所有节点总权值减去 b 的子树的总权值。

BZOJ 3991

- 小 B 最近正在玩一个寻宝游戏，这个游戏的地图是一棵树。

BZOJ 3991

- 小 B 最近正在玩一个寻宝游戏，这个游戏的地图是一棵树。
- 游戏开始时，玩家可以任意选择一个点，瞬间转移到这个村庄，然后可以任意在地图的道路上行走，找到所有宝物。

BZOJ 3991

- 小 B 最近正在玩一个寻宝游戏，这个游戏的地图是一棵树。
- 游戏开始时，玩家可以任意选择一个点，瞬间转移到这个村庄，然后可以任意在地图的道路上行走，找到所有宝物。
- 小 B 希望评测一下这个游戏的难度，因此他需要知道玩家找到所有宝物需要行走的最短路程。有时某个村庄中会突然出现或消失宝物，因此小 B 需要不断地更新数据，但是小 B 太强了，不愿意自己计算，也不愿意麻烦超级计算机，因此他向你求助。

BZOJ 3991

- 小 B 最近正在玩一个寻宝游戏，这个游戏的地图是一棵树。
- 游戏开始时，玩家可以任意选择一个点，瞬间转移到这个村庄，然后可以任意在地图的道路上行走，找到所有宝物。
- 小 B 希望评测一下这个游戏的难度，因此他需要知道玩家找到所有宝物需要行走的最短路程。有时某个村庄中会突然出现或消失宝物，因此小 B 需要不断地更新数据，但是小 B 太强了，不愿意自己计算，也不愿意麻烦超级计算机，因此他向你求助。
- 为了简化问题，我们认为最开始时所有村庄内均没有宝物。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。
- 树链的并也有一个性质：树链按 dfs 序排列，它们的并就是每个树链的长度-相邻两个树链的 LCA 到根的路径的长度。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。
- 树链的并也有一个性质：树链按 dfs 序排列，它们的并就是每个树链的长度-相邻两个树链的 LCA 到根的路径的长度。
- 这个大概自己脑补一下就差不多了。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。
- 树链的并也有一个性质：树链按 dfs 序排列，它们的并就是每个树链的长度-相邻两个树链的 LCA 到根的路径的长度。
- 这个大概自己脑补一下就差不多了。
- 所以我们用 set 维护 dfs 序，每加入一个点就找出它在 DFS 序上的前驱后继，计算树链的并的变化长度，删除时类似。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。
- 树链的并也有一个性质：树链按 dfs 序排列，它们的并就是每个树链的长度-相邻两个树链的 LCA 到根的路径的长度。
- 这个大概自己脑补一下就差不多了。
- 所以我们用 set 维护 dfs 序，每加入一个点就找出它在 DFS 序上的前驱后继，计算树链的并的变化长度，删除时类似。
- 不过由于可以从任意一个节点出发，所以总长度应该减去所有点的 LCA 到根的路径长度（也就是 dfs 序最小的和最大的点的 LCA），答案就是总长度 $\times 2$。

Solution

- DFS 序有一个性质: 两点间 LCA 的深度 = 两点在 dfs 序上的区间中深度的最小值。
- 树链的并也有一个性质：树链按 dfs 序排列，它们的并就是每个树链的长度-相邻两个树链的 LCA 到根的路径的长度。
- 这个大概自己脑补一下就差不多了。
- 所以我们用 set 维护 dfs 序，每加入一个点就找出它在 DFS 序上的前驱后继，计算树链的并的变化长度，删除时类似。
- 不过由于可以从任意一个节点出发，所以总长度应该减去所有点的 LCA 到根的路径长度（也就是 dfs 序最小的和最大的点的 LCA），答案就是总长度 $\times 2$。
- 思想和我们前几天考的那个开荒差不多。

BZOJ3714

- 万兽之王的桌子上有 n 个杯子排成一行，编号为 $1, 2, \ldots, n$，其中某些杯子底下藏有一个小球，如果你准确地猜出是哪些杯子，你就可以获得万兽之王的青睐。

BZOJ3714

- 万兽之王的桌子上有 n 个杯子排成一行，编号为 $1, 2, \ldots, n$，其中某些杯子底下藏有一个小球，如果你准确地猜出是哪些杯子，你就可以获得万兽之王的青睐。
- 花费 c_ij 元，Fe****d**r 就会告诉你杯子 $i, i+1, \ldots, j$ 底下藏有球的总数的奇偶性。

BZOJ3714

- 万兽之王的桌子上有 n 个杯子排成一行，编号为 $1, 2, \ldots, n$，其中某些杯子底下藏有一个小球，如果你准确地猜出是哪些杯子，你就可以获得万兽之王的青睐。
- 花费 c_{ij} 元，Fe****d**r 就会告诉你杯子 $i, i+1, \ldots, j$ 底下藏有球的总数的奇偶性。
- 采取最优的询问策略，你至少需要花费多少元，才能保证得到万兽之王的赏识？

Solution

- 假设第 i 位前缀和是 $s[i]$

Solution

- 假设第 i 位前缀和是 $s[i]$
- 相当于要知道所有的 $s[i]-s[0]$ 的值

Solution

- 假设第 i 位前缀和是 $s[i]$
- 相当于要知道所有的 $s[i]-s[0]$ 的值
- 我们询问一次 i 到 j 的奇偶性，即连一条从 $i-1$ 到 j 的边，因此，当所有的点构成联通块时，即可完成要求。

Solution

- 假设第 i 位前缀和是 $s[i]$
- 相当于要知道所有的 $s[i]-s[0]$ 的值
- 我们询问一次 i 到 j 的奇偶性，即连一条从 $i-1$ 到 j 的边，因此，当所有的点构成联通块时，即可完成要求。
- 因此，最小生成树就是答案

CF 935E

- 给出一个算式，由括号、小于 10 的正整数和问号组成，问号是算式中的符号，现给出原式中加号的个数 p 和减号的个数 m，对于所有填放方式对应的结果，求最大值。

CF 935E

- 给出一个算式，由括号、小于 10 的正整数和问号组成，问号是算式中的符号，现给出原式中加号的个数 p 和减号的个数 m，对于所有填放方式对应的结果，求最大值。
- 算式长度 $\leq 10^4$ $0 \leq p, m \leq 100$

Solution

- 可以把这个算式化成一个二叉树，叶子节点是一个数，非叶子节点表示一个符号，当一个非叶子结点填加号时，就需要最大化左右子树的值，当一个非叶子结点填减号时，就需要最大化左子树值，最小化右子树值。

Solution

- 可以把这个算式化成一个二叉树，叶子节点是一个数，非叶子节点表示一个符号，当一个非叶子结点填加号时，就需要最大化左右子树的值，当一个非叶子结点填减号时，就需要最大化左子树值，最小化右子树值。
- 树形 dp 可以解决这个问题，可以用 $dp[i][j][k][0/1]$ 表示第 i 个点及其子树中填了 j 个加号，k 个负号所能得到的最大/最小值，但是这样开数组开不下，但其实加号和减号数量可以互相转换，所以去掉加号或减号中的一维即可。

csa array coloring

- 给定一个长度为 n 的序列，最开始所有的颜色都是 0 号颜色。

csa array coloring

- 给定一个长度为 n 的序列，最开始所有的颜色都是 0 号颜色。
- 总共进行 m 次操作: 选择一个非空的连续子串并将它涂为一个 1 到 m 之间的颜色，每个颜色能且仅能出现一次，最后，不能有 0 号颜色出现。

csa array coloring

- 给定一个长度为 n 的序列，最开始所有的颜色都是 0 号颜色。
- 总共进行 m 次操作: 选择一个非空的连续子串并将它涂为一个 1 到 m 之间的颜色，每个颜色能且仅能出现一次，最后，不能有 0 号颜色出现。
- 定义涂色次数最多的位置的涂色次数为这个序列的价值，问最大价值是多少。

csa array coloring

- 给定一个长度为 n 的序列，最开始所有的颜色都是 0 号颜色。
- 总共进行 m 次操作: 选择一个非空的连续子串并将它涂为一个 1 到 m 之间的颜色，每个颜色能且仅能出现一次，最后，不能有 0 号颜色出现。
- 定义涂色次数最多的位置的涂色次数为这个序列的价值，问最大价值是多少。
- $1 \leq n, m \leq 10^5$

Solution

- 首先，我们可以发现在最终序列中没有出现的颜色可以忽略，最后计算价值时直接加上其数量即可。

Solution

- 首先，我们可以发现在最终序列中没有出现的颜色可以忽略，最后计算价值时直接加上其数量即可。
- 接下来，发现当有位置 i_1 i_2 i_3 i_4 满足 $i_1 < i_2 < i_3 < i_4$，i_1 及 i_3 涂为颜色 A 且 i_2 及 i_4 涂为颜色 B 时，肯定无解。

Solution

- 首先，我们可以发现在最终序列中没有出现的颜色可以忽略，最后计算价值时直接加上其数量即可。
- 接下来，发现当有位置 i_1 i_2 i_3 i_4 满足 $i_1 < i_2 < i_3 < i_4$，i_1 及 i_3 涂为颜色 A 且 i_2 及 i_4 涂为颜色 B 时，肯定无解。
- 我们可以定义一个颜色的范围为其在序列中第一次出现的位置及最后一次出现的位置。如果有解，则任意两个颜色的范围或为不相交关系，或为包含关系。

Solution

- 根据这些关系，我们可以建出很多有根树，其中一个节点的父亲节点为包含其的最短范围的颜色。

Solution

- 根据这些关系，我们可以建出很多有根树，其中一个节点的父亲节点为包含其的最短范围的颜色。
- 现在对于某个涂为 A 颜色的位置，其祖先肯定会在其之前被涂上，可以加上一的贡献。

Solution

- 根据这些关系，我们可以建出很多有根树，其中一个节点的父亲节点为包含其的最短范围的颜色。
- 现在对于某个涂为 A 颜色的位置，其祖先肯定会在其之前被涂上，可以加上一的贡献。
- 我们可以考虑剩下的情况，对于 X 有一个祖先 T，而 T 有一个儿子 Y，则若 T 在 X 与 Y 中没有出现，我们可以让 Y 涂到 X 的范围中，增加 1 的贡献。

Solution

- 根据这些关系，我们可以建出很多有根树，其中一个节点的父亲节点为包含其的最短范围的颜色。
- 现在对于某个涂为 A 颜色的位置，其祖先肯定会在其之前被涂上，可以加上一的贡献。
- 我们可以考虑剩下的情况，对于 X 有一个祖先 T，而 T 有一个儿子 Y，则若 T 在 X 与 Y 中没有出现，我们可以让 Y 涂到 X 的范围中，增加 1 的贡献。
- 将每个位置的这些贡献相加，chkmax 即可得到答案。

[ZJOI2008] 骑士

- 给定一个基环树森林，每个点有点权。

[ZJOI2008] 骑士

- 给定一个基环树森林，每个点有点权。
- 你需要选出一些点，使得不存在一条边连接的两个点都被选出。

[ZJOI2008] 骑士

- 给定一个基环树森林，每个点有点权。
- 你需要选出一些点，使得不存在一条边连接的两个点都被选出。
- 在此基础上，需要使选出点的点权和最大。

[ZJOI2008] 骑士

- 给定一个基环树森林，每个点有点权。
- 你需要选出一些点，使得不存在一条边连接的两个点都被选出。
- 在此基础上，需要使选出点的点权和最大。
- $n \leq 10^6$

Solution

- 如果没有环的话，就是一个上司的舞会。

Solution

- 如果没有环的话，就是一个上司的舞会。
- 考虑这个环的情况怎么处理：

Solution

- 如果没有环的话，就是一个上司的舞会。
- 考虑这个环的情况怎么处理：
- 对于每个联通块，有且仅有一个环，所以暴力找出这个环并从中断开，对于端点分别做一次 dp，设 $dp[u][0]$ 为不选 u，$dp[u][1]$ 为选 u，

Solution

- 如果没有环的话，就是一个上司的舞会。
- 考虑这个环的情况怎么处理：
- 对于每个联通块，有且仅有一个环，所以暴力找出这个环并从中断开，对于端点分别做一次 dp，设 $dp[u][0]$ 为不选 u，$dp[u][1]$ 为选 u，
- 那么这个联通块答案就是 $max(dp[u1][0], dp[u2][0])$。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。
- 2. 在满足条件 1 的情况下 max(x,y) 最小。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。
- 2. 在满足条件 1 的情况下 max(x,y) 最小。
- 3. 在满足条件 1 和 2 的情况下 min(x,y) 最小。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。
- 2. 在满足条件 1 的情况下 max(x,y) 最小。
- 3. 在满足条件 1 和 2 的情况下 min(x,y) 最小。
- 4. 在满足条件 1、2 和 3 的情况下 x>=y。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。
- 2. 在满足条件 1 的情况下 max(x,y) 最小。
- 3. 在满足条件 1 和 2 的情况下 min(x,y) 最小。
- 4. 在满足条件 1、2 和 3 的情况下 x>=y。
- 如果不存在满足条件 1 的 x、y，输出-1 -1。

[Poi2012] Rendezvous

- 给定一个 n 个顶点的有向图，每个顶点有且仅有一条出边。
- 对于顶点 i，记它的出边为 (i, a[i])。
- 再给出 q 组询问，每组询问由两个顶点 a、b 组成，要求输出满足下面条件的 x、y：
- 1. 从顶点 a 沿着出边走 x 步和从顶点 b 沿着出边走 y 步后到达的顶点相同。
- 2. 在满足条件 1 的情况下 max(x,y) 最小。
- 3. 在满足条件 1 和 2 的情况下 min(x,y) 最小。
- 4. 在满足条件 1、2 和 3 的情况下 x>=y。
- 如果不存在满足条件 1 的 x、y，输出-1 -1。
- $1 \leq n, q \leq 500000$

Solution

- 由于给出的是个基环树森林，所以我们考虑如下几种情况：

Solution

- 由于给出的是个基环树森林，所以我们考虑如下几种情况：
- 1. 最终不会走到一个环上，-1。

Solution

- 由于给出的是个基环树森林，所以我们考虑如下几种情况：
- 1. 最终不会走到一个环上，-1。
- 2. 还没走到环上就相遇，那么我们用倍增，当成树上 LCA 来处理即可。

Solution

- 由于给出的是个基环树森林，所以我们考虑如下几种情况：
- 1. 最终不会走到一个环上，-1。
- 2. 还没走到环上就相遇，那么我们用倍增，当成树上 LCA 来处理即可。
- 3. 走到环上才相遇，那么相遇点一定是两人刚走到环上时的两个点中的一个，判一下即可。

CF 280C

- 给定一个以 1 为根的有根树，每个点初始为白色，每次随机选择一个白色点，把它的子树全部染黑。

CF 280C

- 给定一个以 1 为根的有根树，每个点初始为白色，每次随机选择一个白色点，把它的子树全部染黑。
- 求所有点被染黑的期望操作数。

CF 280C

- 给定一个以 1 为根的有根树，每个点初始为白色，每次随机选择一个白色点，把它的子树全部染黑。
- 求所有点被染黑的期望操作数。
- $n \leq 10^5$

Solution

- 我们随机点一个人来讲讲这题。

Solution

- 我们随机点一个人来讲讲这题。
- redbag!

Solution

- 我们随机点一个人来讲讲这题。
- redbag!
-

Solution

- 我们随机点一个人来讲讲这题。
- redbag!
-
- 一个点被染黑有 dep_i 种方案，其中 dep_i 为 i 到根路径上的点数。

Solution

- 我们随机点一个人来讲讲这题。
- redbag!
-
- 一个点被染黑有 dep_i 种方案，其中 dep_i 为 i 到根路径上的点数。
- 其中只有选定这个点这种情况，这个点才会对答案产生贡献。

Solution

- 我们随机点一个人来讲讲这题。
- redbag!
-
- 一个点被染黑有 dep_i 种方案，其中 dep_i 为 i 到根路径上的点数。
- 其中只有选定这个点这种情况，这个点才会对答案产生贡献。
- 所以答案就是 $\sum \frac{1}{dep_i}$

CF 294E

- 给定一棵树，边有边权。

CF 294E

- 给定一棵树，边有边权。
- 你需要选一条边断掉，同时用这条边把两棵子树在任意位置连起来。

CF 294E

- 给定一棵树，边有边权。
- 你需要选一条边断掉，同时用这条边把两棵子树在任意位置连起来。
- 需要使得任意两点路径长度之和最小。

CF 294E

- 给定一棵树，边有边权。
- 你需要选一条边断掉，同时用这条边把两棵子树在任意位置连起来。
- 需要使得任意两点路径长度之和最小。
- $n \leq 5000$

Solution

- 首先枚举删掉的是哪条边，删掉这条边之后会使得其变成两棵树。

Solution

- 首先枚举删掉的是哪条边，删掉这条边之后会使得其变成两棵树。
- 假设断掉的是 u,v 之间的边。

Solution

- 首先枚举删掉的是哪条边，删掉这条边之后会使得其变成两棵树。
- 假设断掉的是 u,v 之间的边。
- ans=u 的树中任意两点距离和 +v 的树中任意两点的距离和 +u 树中任意点到 u 的距离和 *v 的节点数 +v 树中任意点到 v 的距离和 *u 的节点数 +u 的节点数 *v 的节点数 *uv 之间的距离

Solution

- 首先枚举删掉的是哪条边，删掉这条边之后会使得其变成两棵树。
- 假设断掉的是 u,v 之间的边。
- ans=u 的树中任意两点距离和 +v 的树中任意两点的距离和 +u 树中任意点到 u 的距离和 *v 的节点数 +v 树中任意点到 v 的距离和 *u 的节点数 +u 的节点数 *v 的节点数 *uv 之间的距离
- 设 dp[u][0] 表示 u 子树中节点数量，dp[u][1] 表示子树每个孩子到这个点的距离,dp[u][2] 表示子树之间两两距离之和，树形 dp 维护一下即可。

csa Expected Tree Degrees

- 用以下算法生成一棵有 n 个节点的树：

csa Expected Tree Degrees

- 用以下算法生成一棵有 n 个节点的树：
- 1、从标号为 0 的节点开始。

csa Expected Tree Degrees

- 用以下算法生成一棵有 n 个节点的树：
- 1、从标号为 0 的节点开始。
- 2、按顺序加上剩下的节点，并在节点 i 与 $rand()\ mod\ i$ 之间加上一条边。

csa Expected Tree Degrees

- 用以下算法生成一棵有 n 个节点的树：
- 1、从标号为 0 的节点开始。
- 2、按顺序加上剩下的节点，并在节点 i 与 $rand()\ mod\ i$ 之间加上一条边。
- 定义树的价值为 $\sum_{i=0}^{n-1} degree_i^2$。计算树的期望价值。

csa Expected Tree Degrees

- 用以下算法生成一棵有 n 个节点的树：
- 1、从标号为 0 的节点开始。
- 2、按顺序加上剩下的节点，并在节点 i 与 $rand()\ mod\ i$ 之间加上一条边。
- 定义树的价值为 $\sum_{i=0}^{n-1} degree_i^2$。计算树的期望价值。
- $1 \leq n \leq 10^6$ 精确到小数点后 6 位。

Solution

- 首先很容易想到一个 $O(n^2)$ 的 dp: 设 $dp[i][j]$ 表示在总共有 i 个节点的树中度数为 j 的点数的期望，所以 dp 方程为 $dp[i][j] = dp[i-1][j] + \frac{dp[i-1][j-1]-dp[i-1][j]}{i-1}$。

Solution

- 首先很容易想到一个 $O(n^2)$ 的 dp: 设 $dp[i][j]$ 表示在总共有 i 个节点的树中度数为 j 的点数的期望，所以 dp 方程为 $dp[i][j] = dp[i-1][j] + \frac{dp[i-1][j-1]-dp[i-1][j]}{i-1}$。
- 考虑如何优化这个 dp，可以发现一棵树当 n 非常大时，度数超过一定值的点数期望非常小，即使将其忽略也不会影响我们的答案。

Solution

- 首先很容易想到一个 $O(n^2)$ 的 dp: 设 $dp[i][j]$ 表示在总共有 i 个节点的树中度数为 j 的点数的期望，所以 dp 方程为 $dp[i][j] = dp[i-1][j] + \frac{dp[i-1][j-1]-dp[i-1][j]}{i-1}$。
- 考虑如何优化这个 dp，可以发现一棵树当 n 非常大时，度数超过一定值的点数期望非常小，即使将其忽略也不会影响我们的答案。
- 对于 $n = 10^6$，$dp[n][40]$ 已经足够小，影响不到我们的答案。因此，我们只需要计算 j 比较小的值，复杂度就被降至 $O(n \times c)$，其中 c 为一个小常数。

Solution

- 但是有没有可以精确计算而不 T 的方法呢。

Solution

- 但是有没有可以精确计算而不 T 的方法呢。
- 当然是有的。

Solution

- 但是有没有可以精确计算而不 T 的方法呢。
- 当然是有的。
- 下面给出一个式子:

Solution

- 但是有没有可以精确计算而不 T 的方法呢。
- 当然是有的。
- 下面给出一个式子:
- $ans = 6 \times (n-1) - 4 \times (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n-1})$

Solution

- 但是有没有可以精确计算而不 T 的方法呢。
- 当然是有的。
- 下面给出一个式子:
- $ans = 6 \times (n-1) - 4 \times (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n-1})$
- 先自行证明一下。

Solution

- proven by $\frac{1}{4}$

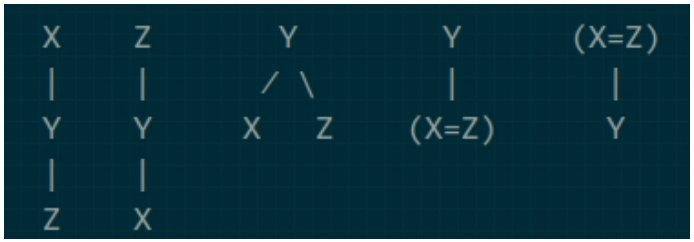
Solution

- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。

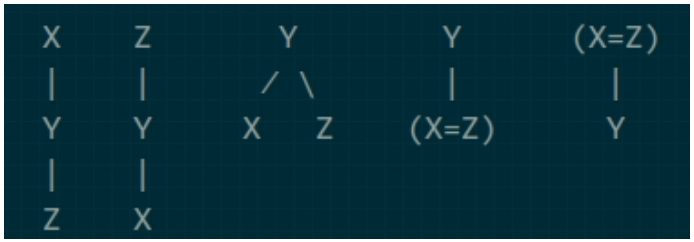
Solution

- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。
- 考虑 x, y, z 的形态：

Solution

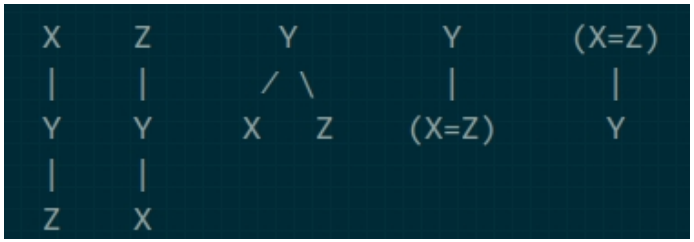
- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。
- 考虑 x, y, z 的形态：
- 

Solution

- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。
- 考虑 x, y, z 的形态：
- 

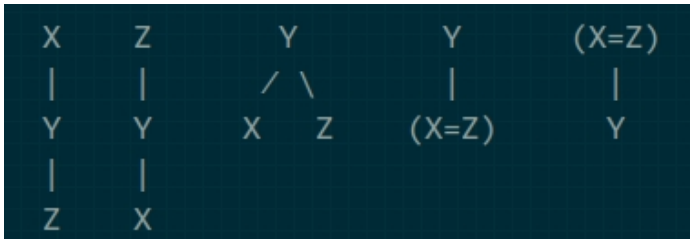
- 1&2: $2 \times \sum_1^{n-1} \frac{i-1}{i}$

Solution

- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。
- 考虑 x, y, z 的形态：
- 

- 1&2: $2 \times \sum_1^{n-1} \frac{i-1}{i}$
- 4&5: $2 \times (n-1)$

Solution

- proven by $\frac{1}{4}$
- 考虑转化一下这个问题，这棵树的价值，即找出 x, y, z 的三元组的数量，使 x, y, z 在树上相连。
- 考虑 x, y, z 的形态：
- 

- 1&2: $2 \times \sum_1^{n-1} \frac{i-1}{i}$
- 4&5: $2 \times (n-1)$
- 3: 令 $X < Z$: $\sum_{x=1}^{n-1} \sum_{z=x+1}^{n-1} \frac{1}{z} = \sum_{z=1}^{n-1} \frac{z-1}{z}$

CF 600E

- 给出一棵树，每个节点有一个颜色 x，若当前子树中没有颜色的数量超过了 x，则称 x 是这棵子树中的主导颜色.

CF 600E

- 给出一棵树，每个节点有一个颜色 x，若当前子树中没有颜色的数量超过了 x，则称 x 是这棵子树中的主导颜色.
- 现在我们要求每个节点为根的子树的主导颜色的颜色编号的和.

Solution

- 这道题我们先考虑暴力，我们可以从根节点开始扫描，把每一棵子树的答案都暴力得出，时间复杂度 $O(n^2)$.

Solution

- 这道题我们先考虑暴力，我们可以从根节点开始扫描，把每一棵子树的答案都暴力得出，时间复杂度 $O(n^2)$.
- 我们换一种思路暴力，我们可以遍历一遍树，当遍历到一个节点时，把它子树的贡献都计入答案，得出答案后再将所有节点的贡献删除.

Solution

- 这道题我们先考虑暴力，我们可以从根节点开始扫描，把每一棵子树的答案都暴力得出，时间复杂度 $O(n^2)$.
- 我们换一种思路暴力，我们可以遍历一遍树，当遍历到一个节点时，把它子树的贡献都计入答案，得出答案后再将所有节点的贡献删除.
- 可是我们可以很显然的看出，其实每个节点最后遍历的儿子子树是不用删除贡献的，所以这样可以做到优化，不过最坏时间复杂度依然是 $O(n^2)$.

Solution

- 这道题我们先考虑暴力，我们可以从根节点开始扫描，把每一棵子树的答案都暴力得出，时间复杂度 $O(n^2)$.
- 我们换一种思路暴力，我们可以遍历一遍树，当遍历到一个节点时，把它子树的贡献都计入答案，得出答案后再将所有节点的贡献删除.
- 可是我们可以很显然的看出，其实每个节点最后遍历的儿子子树是不用删除贡献的，所以这样可以做到优化，不过最坏时间复杂度依然是 $O(n^2)$.
- 这个时候我们可以让最后一个遍历的儿子子树尽量大，也就是选择重儿子，这样可以做到稳定 $O(nlog(n))$.

Solution

- 这道题我们先考虑暴力，我们可以从根节点开始扫描，把每一棵子树的答案都暴力得出，时间复杂度 $O(n^2)$.
- 我们换一种思路暴力，我们可以遍历一遍树，当遍历到一个节点时，把它子树的贡献都计入答案，得出答案后再将所有节点的贡献删除.
- 可是我们可以很显然的看出，其实每个节点最后遍历的儿子子树是不用删除贡献的，所以这样可以做到优化，不过最坏时间复杂度依然是 $O(n^2)$.
- 这个时候我们可以让最后一个遍历的儿子子树尽量大，也就是选择重儿子，这样可以做到稳定 $O(nlog(n))$.
- 这道题就是一个 dsu 裸题，当然也可以用线段树合并做。

BZOJ 3083

- hack 国有 n 个城市，这些城市构成了一颗树。hack 国有一个首都，我们可以把这个首都看做整棵树的根，但 hack 国比较奇怪，首都是随时有可能变为另外一个城市的。

BZOJ 3083

- hack 国有 n 个城市，这些城市构成了一颗树。hack 国有一个首都，我们可以把这个首都看做整棵树的根，但 hack 国比较奇怪，首都是随时有可能变为另外一个城市的。
- hack 国的每个城市有一个防御值来防御常数国和沙国的进攻，有些时候国王会使得某两个城市之间的路径上的所有城市的防御值都变为某个值。

BZOJ 3083

- hack 国有 n 个城市，这些城市构成了一颗树。hack 国有一个首都，我们可以把这个首都看做整棵树的根，但 hack 国比较奇怪，首都是随时有可能变为另外一个城市的。
- hack 国的每个城市有一个防御值来防御常数国和沙国的进攻，有些时候国王会使得某两个城市之间的路径上的所有城市的防御值都变为某个值。
- 国王想知道在某个时候，如果把首都看做整棵树的根的话，那么以某个城市为根的子树的所有城市的防御值最小是多少。由于国王无法解决这个问题，所以他拦住了妙想希望她能帮忙。

BZOJ 3083

- hack 国有 n 个城市，这些城市构成了一颗树。hack 国有一个首都，我们可以把这个首都看做整棵树的根，但 hack 国比较奇怪，首都是随时有可能变为另外一个城市的。
- hack 国的每个城市有一个防御值来防御常数国和沙国的进攻，有些时候国王会使得某两个城市之间的路径上的所有城市的防御值都变为某个值。
- 国王想知道在某个时候，如果把首都看做整棵树的根的话，那么以某个城市为根的子树的所有城市的防御值最小是多少。由于国王无法解决这个问题，所以他拦住了妙想希望她能帮忙。
- 但妙想还要跳 hop，所以这个重大的问题就被转交到了你的手上。

Solution

- 没有换根的话，就是树链剖分裸题

Solution

- 没有换根的话，就是树链剖分裸题
- 下面只需要考虑换根是怎样的操作。设 x 为要查询的城市，rt 为当前的根，root 为原来的根对于询问分类讨论:

Solution

- 没有换根的话，就是树链剖分裸题
- 下面只需要考虑换根是怎样的操作。设 x 为要查询的城市，rt 为当前的根，root 为原来的根对于询问分类讨论:
- x==rt，查询 [1,n]

Solution

- 没有换根的话，就是树链剖分裸题
- 下面只需要考虑换根是怎样的操作。设 x 为要查询的城市，rt 为当前的根，root 为原来的根对于询问分类讨论:
- x==rt，查询 [1,n]
- x 在 rt 的子树内，则子树没有变，直接查询

Solution

- 没有换根的话，就是树链剖分裸题
- 下面只需要考虑换根是怎样的操作。设 x 为要查询的城市，rt 为当前的根，root 为原来的根对于询问分类讨论:
- x==rt，查询 [1,n]
- x 在 rt 的子树内，则子树没有变，直接查询
- x 不在 rt 到 root 的链上，子树没有变

Solution

- 没有换根的话，就是树链剖分裸题
- 下面只需要考虑换根是怎样的操作。设 x 为要查询的城市，rt 为当前的根，root 为原来的根对于询问分类讨论:
- x==rt，查询 [1,n]
- x 在 rt 的子树内，则子树没有变，直接查询
- x 不在 rt 到 root 的链上，子树没有变
- x 在 rt 到 root 的链上，(即 x=lca(x,rt))，发现现在 x 的子树就是整棵树减去 x 往 rt 方向向下那个节点的子树，于是倍增求那个点然后整个 dfs 序就是分成两段了，即 [1,pos[v]-1], [pos[v]+sz[v]+1,n]。

BZOJ 3589

- 事件 0: 这棵树长出了一些果子, 即某个子树中的每个节点都会长出 K 个果子.

BZOJ 3589

- 事件 0: 这棵树长出了一些果子，即某个子树中的每个节点都会长出 K 个果子.
- 事件 1: 妙想希望你求出 l 条树枝上的果子数. 一条树枝其实就是一个从某个节点到根的路径的一段. 每次妙想会选定一些树枝，让你求出在这些树枝上的节点的果子数的和.

BZOJ 3589

- 事件 0: 这棵树长出了一些果子，即某个子树中的每个节点都会长出 K 个果子.
- 事件 1: 妙想希望你求出 l 条树枝上的果子数. 一条树枝其实就是一个从某个节点到根的路径的一段. 每次妙想会选定一些树枝，让你求出在这些树枝上的节点的果子数的和.
- 注意，树枝之间可能会重合，这时重合的部分的节点的果子只要算一次.

BZOJ 3589

- 事件 0: 这棵树长出了一些果子，即某个子树中的每个节点都会长出 K 个果子.
- 事件 1: 妙想希望你求出 l 条树枝上的果子数. 一条树枝其实就是一个从某个节点到根的路径的一段. 每次妙想会选定一些树枝，让你求出在这些树枝上的节点的果子数的和.
- 注意，树枝之间可能会重合，这时重合的部分的节点的果子只要算一次.
- $n(1 \leq n \leq 200,000), l \leq 5$

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。
- 总权值 = 单链-两两之交 + 三链之交 ...，所以状压枚举即可

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。
- 总权值 = 单链-两两之交 + 三链之交 ...，所以状压枚举即可
- 两条链的交集求法如下：

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。
- 总权值 = 单链-两两之交 + 三链之交 ...，所以状压枚举即可
- 两条链的交集求法如下：
- 1. 求两条链底的 LCA

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。
- 总权值 = 单链-两两之交 + 三链之交 ...，所以状压枚举即可
- 两条链的交集求法如下：
- 1. 求两条链底的 LCA
- 2. 若 LCA 的深度小于其中一条链的链顶深度交集为空返回 0

Solution

- 首先只考虑事件 0，即子树修改，链上查询，树链剖分的裸题
- 考虑事件 1，我们可以发现一个性质：$l \leq 5$，所以我们考虑容斥原理。
- 总权值 = 单链-两两之交 + 三链之交 ...，所以状压枚举即可
- 两条链的交集求法如下：
- 1. 求两条链底的 LCA
- 2. 若 LCA 的深度小于其中一条链的链顶深度交集为空返回 0
- 3. 返回一条链链底为 LCA 链顶为两条链顶中深度较大的那个

BZOJ 3862

- 给出一棵 n 个点的树，有三种操作。

BZOJ 3862

- 给出一棵 n 个点的树，有三种操作。
- 操作 1：把 x,y 路径上所有边反色

BZOJ 3862

- 给出一棵 n 个点的树，有三种操作。
- 操作 1：把 x,y 路径上所有边反色
- 操作 2：把 x,y 路径上所有相邻的边反色，即一个点在路径上

BZOJ 3862

- 给出一棵 n 个点的树，有三种操作。
- 操作 1：把 x,y 路径上所有边反色
- 操作 2：把 x,y 路径上所有相邻的边反色，即一个点在路径上
- 操作 3：询问 x,y 路径上黑边的个数。

BZOJ 3862

- 给出一棵 n 个点的树，有三种操作。
- 操作 1：把 x,y 路径上所有边反色
- 操作 2：把 x,y 路径上所有相邻的边反色，即一个点在路径上
- 操作 3：询问 x,y 路径上黑边的个数。
- 注意刚开始的时候所有边均为白色。

Solution

- 操作 1,3 都是基本的操作，关键就是 2.

Solution

- 操作 1,3 都是基本的操作，关键就是 2.
- 对于每个点维护这个点的轻儿子是否要反色。每次修改的时候直接区间修改即可。

Solution

- 操作 1,3 都是基本的操作，关键就是 2.
- 对于每个点维护这个点的轻儿子是否要反色。每次修改的时候直接区间修改即可。
- 两条重链相连的轻边需要特判。路径的顶点到他父亲之间的边也需要特判。

Solution

- 操作 1,3 都是基本的操作，关键就是 2.
- 对于每个点维护这个点的轻儿子是否要反色。每次修改的时候直接区间修改即可。
- 两条重链相连的轻边需要特判。路径的顶点到他父亲之间的边也需要特判。
- 每次计算答案的时候，链顶的颜色都要结合上父亲的轻儿子是否反色来计算。

The End

Thanks!