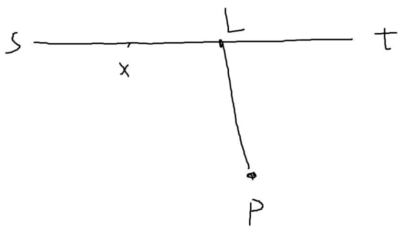
树形DP及其他

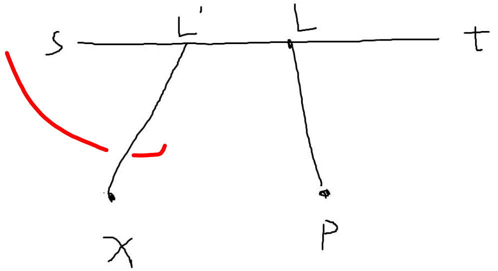
清华大学 张华清

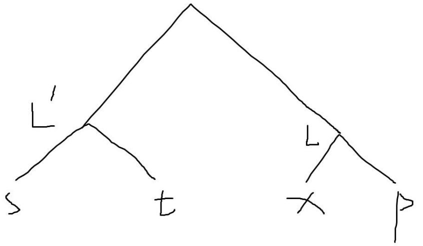
树的直径

dis(L',t) < dis(L,t) < dis(L,p)

- 定义：树上任意两点间的最长路径称为**树的直径**
- 注意：树的两个端点可能并不重合。
- 求法一（仅适用于非负边权）：
  - 从任意点x出发，用dfs求出树上到x距离最远的点p。则p必为树的某条直径的端点。
  - 再从p出发，用dfs求出树上到p距离最远的点q。
  - 则(p,q)即为树的直径。
- *简要证明：若p不是树直径的某个端点。设真实直径是(s,t)。
  - 若x在(s,t)上，dis(L,p)>dis(L,t)，则dis(s,p)>dis(s,t)，矛盾。
  - 若(x,p)和(s,t)有交，类似。
  - 若(x,p)和(s,t)无交，则dis(L,p)>dis(L,t)，则dis(L’,t)<dis(x,p)，则dis(s,p)>dis(s,t)，矛盾。







树的直径

- 求法二（适用于负边权）：树上递推。
- 树上每条链(s,t)都可以拆成两条直链(s,lca(s,t)) 和 (t,lca(s,t))
- 我们枚举每个点x，求出以x为lca的最长链。两条直链分别即为从x向下，不经过同一个儿子，最长和次长的两条链。

```
void dfs(int u,int fa){
    for(int i=frm[u];i;i=edge[i].nxt){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
        if(d1[v]+1>d1[u]){
            d2[u]=d1[u];
            d1[u]=d1[v]+1;
        }
        else{
            if(d1[v]+1>d2[u]){
                d2[u]=d1[v]+1;
            }
        }
    }
}

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n-1;i++){
    dfs(1,1);
    for(int i=1;i<=n;i++){
        ans=max(ans,d1[i]+d2[i]);
    }
```

例题

- 给定一棵树，树上每点有权值a_x。找出两点x,y，使得$dis(x,y)\cdot \max(a_x,a_y)$最大。n<=1e5

$a_x > a_y$

x_0, y_0

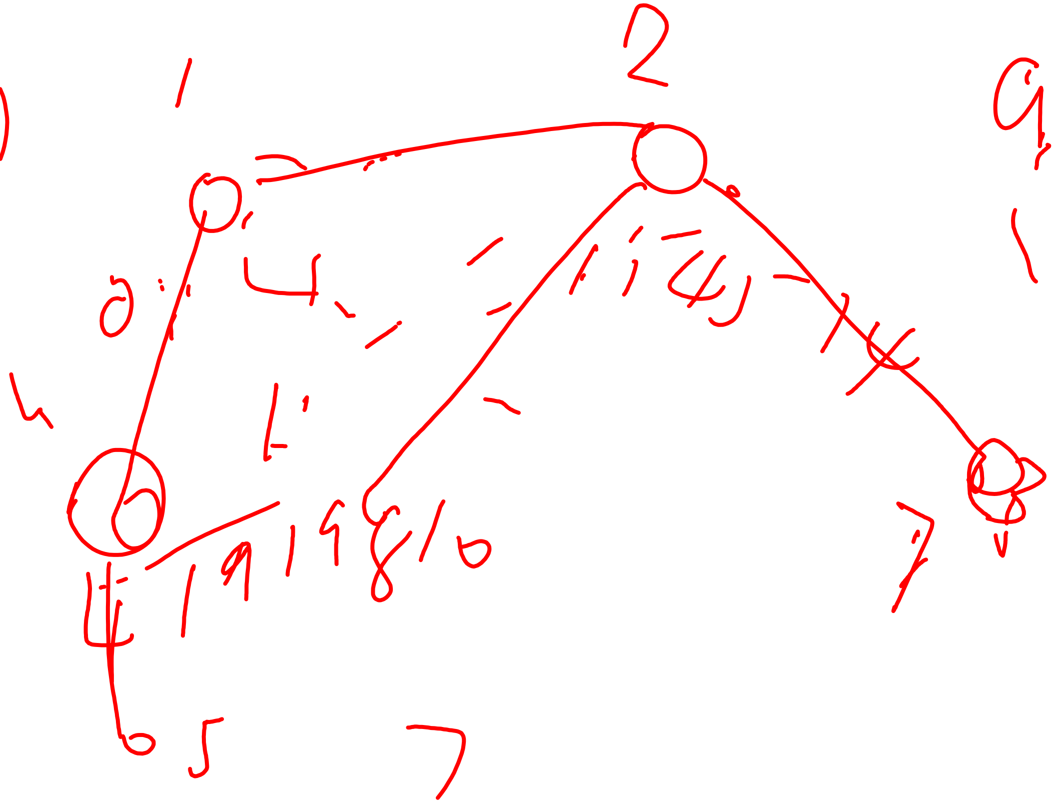
$a_{x_0} \Rightarrow a_{y_0}$

max

min

$a_{x_0} \cdot dis(x_0, y)$

$a_{x_0} \cdot dis(x_0, z_0)$



例题

- $dis(x,y)\cdot\max(a_x,a_y)=\max\left(a_x dis(x,y),a_y dis(x,y)\right)$
- 我们枚举其中一个点x，之后只需要求距离他最远的点y，取$a_x * dis(x,y)$的最大值即可
  - 但a_x可能不是a_x和a_y中较大的那个啊？
  - 没关系，反正求最大值。

$a_y \cdot dis(y,z)$

- 结论：设树的任意一条直径为(s,t)，则距离x最远的点一定为s或t中的某一个。（求法一中已证）

$a_x < a_y$

- 则求出树的直径后，从s和t分别dfs即可求出每个点x到s和t的距离。

$a_x \cdot dis(x,y) \neq \max(a_x,a_y)\cdot dis(x,y)$

树的直径

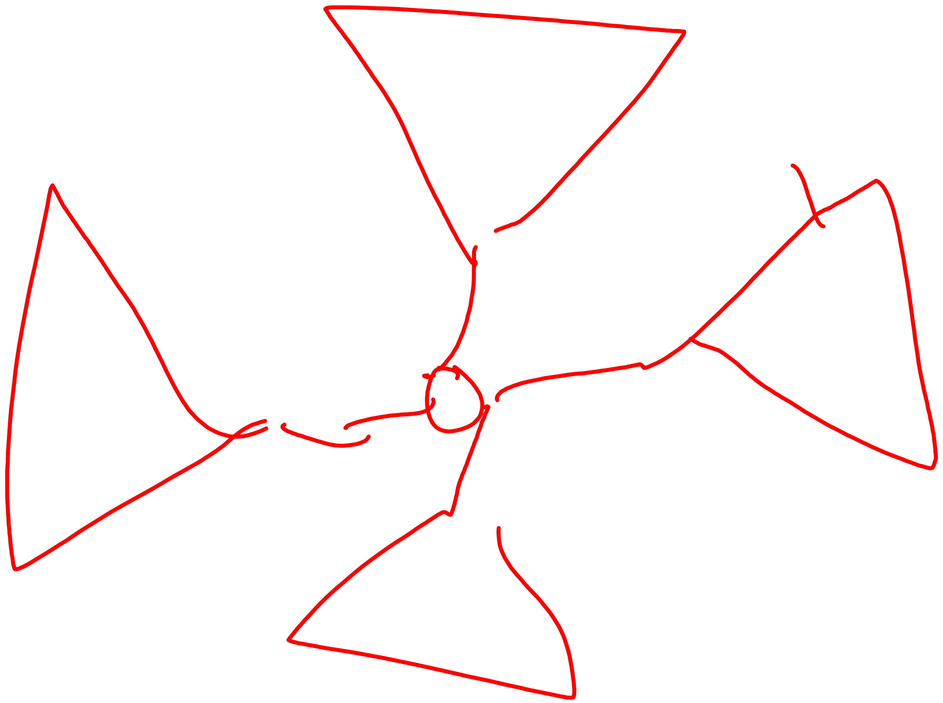
- 补充直径的性质：
- 设树的任意一条直径为(s,t)，则距离x最远的点一定为s或t中的某一个。
- 树的所有直径有交，且中点重合。
- 设树1的一条直径为(s1,t1)，树2的一条直径为(s2,t2)，两树通过一条边合并成一棵大树，那么大树的直径的两个端点必在s1,t1,s2,t2中取。只需要检验六种情况。

树的重心

- 对无根树上每个点x定义其**子树**为删去x后所形成的各个连通块。
- 最大子树最小的点称为树的重心。

t[l, r]

dis(u,v)



树的重心

- 性质：
- 重心最多只有两个。若有两个重心，则它们相邻。
- 重心的最大子树大小不超过总点数的一半。
- 重心是树上到所有点距离和最小的点。
- 插入或删除一个点，树的重心的位置最多移动一个点。
- 若添加一条边连接两棵树，那么新树的重心一定在原来两棵树的重心的路径上。
- 一棵树的重心一定在根节点所在的重链上。

树的重心

- 求解：树上递推求子树大小即可。
- 这就是我们前面说的，把无根树钦定一个根，当作有根树处理。

```
int siz[x];//以x为根的子树大小
int maxp[x];//x的最大子树大小
void getrt(int x,int fa){
    siz[x]=1,maxp[x]=0;
    for(int i=frm[x];i;i=edge[i].nxt){
        int v=edge[i].to;
        if(v!=fa){
            getrt(v,x);
            siz[x]+=siz[v];
            maxp[x]=max(maxp[x],siz[v]);
        }
    }
    maxp[x]=max(maxp[x],sum-siz[x]);
    if(maxp[x]<maxp[rt])rt=x;
}
```

n

Luogu 5536 【XR-3】核心城市

题目描述 复制Markdown 展开

X 国有 n 座城市，$n-1$ 条长度为 1 的道路，每条道路连接两座城市，且任意两座城市都能通过若干条道路相互到达，显然，城市和道路形成了一棵树。

X 国国王决定将 k 座城市钦定为 X 国的核心城市，这 k 座城市需满足以下两个条件：

1. 这 k 座城市可以通过道路，在不经过其他城市的情况下两两相互到达。
2. 定义某个非核心城市与这 k 座核心城市的距离为，这座城市与 k 座核心城市的距离的最小值。那么所有非核心城市中，与核心城市的距离最大的城市，其与核心城市的距离最小。你需要求出这个最小值。

Luogu 5536 【XR-3】核心城市

- 首先，当k=1时，选择的城市为直径的中点
  - 假设选的点为x，树上到x最远的点一定是直径的两个端点中的一个，而$dis(S,x)+dis(T,x)\geq dis(S,T)$（注：树具有三角形不等式），所以$\max\left(dis(S,x),dis(T,x)\right)\geq\frac{1}{2}dis(S,T)$。等号在$x$是$S,T$中点时才取到
- 其次，当$k\geq 1$时，直径的中点也一定要选择。
  - 上面的反证法仍成立，即，即使选了大于1个点，但没有选择直径中点，那么S或T到他们任意一个的距离都>1/2dis(S,T)。

Luogu 5536 【XR-3】核心城市

- 那么我们先选择直径中点，然后提根。
- 下一个选谁？选根的儿子中，子树最深的那个
  - 如果它没有被选，那么它的儿子们也不能被选，那么到已选城市的距离最大值始终不会减小。
- 以此类推即可。

[NOIP2018 提高组] 保卫王国（O(nm)

题目描述

Z 国有 n 座城市，$(n-1)$ 条双向道路，每条双向道路连接两座城市，且任意两座城市都能通过若干条道路相互到达。

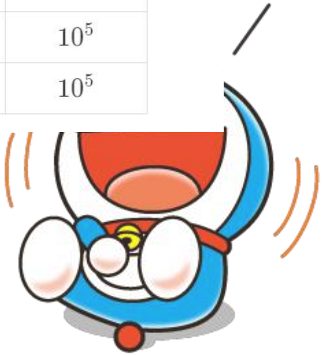
Z 国的国防部长小 Z 要在城市中驻扎军队。驻扎军队需要满足如下几个条件：

- 一座城市可以驻扎一支军队，也可以不驻扎军队。
- 由道路直接连接的两座城市中至少要有一座城市驻扎军队。
- 在城市里驻扎军队会产生花费，在编号为 i 的城市中驻扎军队的花费是 p_i。

小 Z 很快就规划出了一种驻扎军队的方案，使总花费最小。但是国王又给小 Z 提出了 m 个要求，每个要求规定了其中两座城市是否驻扎军队。小 Z 需要针对每个要求逐一给出回答。具体而言，如果国王提出的第 j 个要求能够满足上述驻扎条件（不需要考虑第 j 个要求之外的其它要求），则需要给出在此要求前提下驻扎军队的最小开销。如果国王提出的第 j 个要求无法满足，则需要输出 -1。现在请你来帮助小 Z。

数据规模与约定

测试点编号	type	$n=m=$
$1,2$	A3	10
$3,4$	C3	10
$5,6$	A3	100
7	C3	100
$8,9$	A3	2×10^3
$10,11$	C3	2×10^3
$12,13$	A1	10^5
$14,15,16$	A2	10^5
17	A3	10^5
$18,19$	B1	10^5
$20,21$	C1	10^5
22	C2	10^5
$23,24,25$	C3	10^5



[NOIP2018 提高组] 保卫王国（O(nm)

- 每次询问DP一次即可。设f[x][0/1]为x子树内，x是否放军队的最小花费。
- $f[x][0]=\sum_{v\in son(x)} f[v][1]$
- $f[x][1]=\sum_{v\in son(x)} \min(f[v][0],f[v][1])+a[x]$
- 若x必须/不允许放军队，只需要把f[x][0]/f[x][1]修改为-INF即可。
- 也可以把a_x的花费改为-INF/+INF即可（当然若为-INF，最后要加回来。）

4516 [JSOI2018] 潜入行动

- n<=1e5,m<=100

题目描述

复制Markdown 展开

外星人又双叒叕要攻打地球了，外星母舰已经向地球航行！这一次，JYY 已经联系好了黄金舰队，打算联合所有 JSOIer 抵御外星人的进攻。

在黄金舰队就位之前，JYY 打算事先了解外星人的进攻计划。现在，携带了监听设备的特工已经秘密潜入了外星人的母舰，准备对外星人的通信实施监听。

外星人的母舰可以看成是一棵 n 个节点、 $n-1$ 条边的**无向树**，树上的节点用 $1,2,\cdots,n$ 编号。 JYY 的特工已经装备了隐形模块，可以在外星人母舰中不受限制地活动，可以神不知鬼不觉地在节点上安装监听设备。

如果在节点 u 上安装监听设备，则 JYY 能够监听与 u **直接相邻**所有的节点的通信。换言之，如果在节点 u 安装监听设备，则对于树中每一条边 (u,v) ，节点 v 都会被监听。特别注意**放置在节点 u 的监听设备并不监听 u 本身的通信**，这是 JYY 特别为了防止外星人察觉部署的战术。

JYY 的特工一共携带了 k 个监听设备，现在 JYY 想知道，有多少种不同的放置监听设备的方法，能够使得母舰上**所有节点**的通信都被监听？为了避免浪费，**每个节点至多只能安装一个监听设备，且监听设备必须被用完**。

4516 [JSOI2018] 潜入行动

- 状态设计
- f[x][i]：x子树内，放了i个监听设备的方案数
- 额外信息?
  - x点是否被监听了
  - x点是否有设备
  - 子树内其他点的信息无需被记录。
- f[x][i][0/1][0/1]：x子树内，放了i个监听设备，其中x是否有设备，x是否被监听的方案数

4516 [JSOI2018] 潜入行动

- 转移：一个子树一个子树地合并。假设现在合并x的v子树。
- $tmp[i+j][0][0] = \sum f[x][i][0][0] * f[v][j][0][1]$：x没有设备也没被监听，那么v不能有设备并且v必须被监听了。
- $tmp[i+j][1][0] = \sum f[x][i][1][0] * (f[v][j][0][0] + f[v][j][0][1])$：x有设备，但没被监听，那么v不能有设备。
- $tmp[i+j][0][1] = \sum f[x][i][0][1] * (f[v][j][0][1] + f[v][j][1][1]) + f[x][i][0][0] * f[v][j][1][1]$：x没设备，但需要被监听。如果x之前已经被监听了，那么v不必有设备，但v自己必须被监听。如果x之前没被监听，那就必须被v处的设备监听，所以v必须有设备。
- $tmp[i+j][1][1] = \sum f[x][i][1][1] * (f[v][j][0][0] + f[v][j][0][1] + f[v][j][1][0] + f[v][j][1][1]) + f[x][i][1][0] * (f[v][j][1][0] + f[v][j][1][1])$：f有设备，也需要被监听。如果之前已经被监听了，那么对v没有任何要求。如果x之前没被监听，那就必须被v处的设备监听。
- *说明：转移方程中，左边的f[x]是合并后的f，右边的f[x]是合并前的f。实现时，需要新建一个tmp数组。*

4516 [JSOI2018] 潜入行动

- 复杂度同树形背包。
- 朴素的分析：O(nm^2)。
- 较为细致的分析：O(n^2)！
  - 考虑这样的枚举：for(int i=0;i<=已经合并的子树大小;i++)
    for(int j=0;j<=siz[v];j++)
  - 总复杂度是O(n^2)，因为树上每对点只会在LCA处合并一次。
- 更细致的分析：O(nm)！
  - 其中m是枚举的上界，即i+j<=m。

NOI2020 命运

形式化的：给定一棵树 $T=(V,E)$ 和点对集合 $Q\subseteq V\times V$，满足对于所有 $(u,v)\in Q$，都有 $u\neq v$，并且 u 是 v 在树 T 上的祖先。其中 V 和 E 分别代表树 T 的结点集和边集。求有多少个不同的函数 $f:E\to\{0,1\}$（将每条边 $e\in E$ 的 $f(e)$ 值置为 0 或 1），满足对于任何 $(u,v)\in Q$，都存在 u 到 v 路径上的一条边 e 使得 $f(e)=1$。由于答案可能非常大，你只需要输出结果对 $998,244,353$（一个素数）取模的结果。

测试点编号	n	m	T 为完全二叉树
$1\sim4$	≤ 10	≤ 10	否
5	≤ 500	≤ 15	否
6	$\le 10^4$	≤ 10	否
7	$\le 10^5$	≤ 16	否
8	$\le 5\times10^5$	≤ 16	否
9	$\le 10^5$	≤ 22	否
10	$\le 5\times10^5$	≤ 22	否
11	≤ 600	≤ 600	否
12	$\le 10^3$	$\le 10^3$	否
$13\sim14$	$\le 2\times10^3$	$\le 5\times10^5$	否
$15\sim16$	$\le 5\times10^5$	$\le 2\times10^3$	否
$17\sim18$	$\le 10^5$	$\le 10^5$	是
19	$\le 5\times10^4$	$\le 10^5$	否
20	$\le 8\times10^4$	$\le 10^5$	否

NOI2020 命运

- 显然考虑树形DP。
- 子树内的限制一定在子树里处理完了，完全在子树外的限制肯定也和这个子树无关。
- 我们关心的其实是V在x子树内，U在x子树外，当前还没有被标记的（即还没有一个边为1）的限制中，U最靠下的那个。因为U最靠下的被满足了其他的也一定满足。
- 具体来说：设f[x][i]为考虑x子树，当前没被标记上1的限制中，dep[U]的最大值为i时的方案数。

O(n^2)

NOI2020 命运

- 转移：
- 一个子树一个子树地合并。假设现在合并u的v子树。
- v的父亲u处考虑<u,v>这条边标记0还是1。
- 枚举i<dep[x],j<dep[v]。
- 为0：$tmp[\max(i,j)] = tmp[\max(i,j)] + f[u][i] * f[v][j]$。
- 为1：$tmp[i] = tmp[i] + f[u][i] * f[v][j]$。
- 直接做为O(n^3)。使用前缀和优化可做O(n^2)。

一个题

边权为1.

- 问树中有几个联通块的直径介于[L,R]之间。n<=~~1e5~~1e3

i, j

直径为

一个题

- 我们可能冲上去会写这样的DP：
- 设f[x][d][h]为在x子树内选，并且钦定x一定选，直径为d，距x最远的点（即到x最长链）的距离为h，的连通块个数。然后就可以转移了。
  - 设计思路：
  - 连通块问题常见的DP设计即f[x]为x子树内并且x一定选...
  - 我们要求直径在[L,R]内，我们对每个d求就把直径为d的方案数。为此，我们需要记录已知的最长链；为了求之后可能的直径长度，还需要额外记录到x的最长链（想想如何用DP求树的直径）。
- 复杂度有那么一点炸裂。如何优化？

f[x]

$d_1,$

一个题

[L,R]

$\le R$

- 做差分。求直径介于[0,R]的个数-[0,L-1]的答案。
  - 看到区间，都要想想能不能差分——前缀一定不会比区间难。
- 以[0,R]为例。显然设f[x]为x子树内，钦定x一定选，并且直径不超过R的连通块方案数。可能形成新的直径，发现再记个到x的最长链就行了。
- 于是记f[x][h]为为x子树内，钦定x一定选，到x最长链长度为h并且直径不超过R的方案数。
- 转移：$tmp[\max(hu,hv+1)] += f[u][hu]*f[v][hv]$，$hu+hv+1\le R$。复杂度$O(n^2)$
- 把限制放到DP的外面，在转移时时刻满足这一限制即可 *（Luogu1156垃圾陷阱）*。

THUPC2023初赛 大富翁

size(x)
-dep(x)
-w(x)

题目背景

复制Markdown 展开

有一天，小 W 和小 H 在玩大富翁。

题目描述

这版大富翁的游戏规则比较独特。它的地图是一棵 n 个节点的有根树，其中 1 号节点为根。树上每个节点都有一个价格，第 x 号节点的价格记为 w_x。

对于树上两个不同的节点 x, y，若 x 是 y 的祖先节点（即，x 在 1 号点到 y 号点的简单路径上），则称 x **支配** y。

游戏过程中，小 W 和小 H 轮流**购买**树上的一个未被人购买过的节点，直到树上的 n 个节点都被小 W 或小 H 购买。（游戏开始前，树上的所有节点都没有被购买。）

对于一次购买，假设买方购买了 x 号节点，那么他首先要向系统支付 w_x 个游戏币。假设此时 x 支配着 n_1 个已被买方的对手购买了的节点，同时又被 n_2 个已被对手购买了的节点支配。若 $n_1 > n_2$，那么对手要向买方支付 $n_1 - n_2$ 个游戏币，若 $n_1 < n_2$，那么买方要向对手支付 $n_2 - n_1$ 个游戏币。

小 W 和小 H 都是绝顶聪明的人，他们都会在游戏中采用最优策略，来使自己赚到尽量多的游戏币。现在，小 W 想考考你：如果他先手，他最终能赚到多少个游戏币？（即，在整个游戏过程中，小 W 从小 H 手中获得的游戏币个数减去他支付给系统和小 H 的游戏币个数。你可以认为，游戏开始前，小 H 和小 W 手中都有足够数量的游戏币。注意：答案可能为负数。）

子任务

对于所有测试数据，$1 \le n \le 2 \times 10^5$，$0 \le w_x \le 2 \times 10^5$。保证输入的图为一棵以 1 号节点为根的有根树。

THUPC2023初赛 大富翁

- 最优化题目，别忘记试试贪心。
- 容易发现：只和最终局面有关，和具体操作顺序无关。

THUPC2023初赛 大富翁

- 容易发现收益和只和最终局面有关，而和点的选取过程无关。具体来说，设W为最终局面中先手选取的点的集合，H为最终局面中后手选取的点的集合，则先手总收益为：
- $\sum_{x\in W}\sum_{y\in H}[x支配y]-\sum_{x\in W}\sum_{y\in H}[y支配x]-\sum_{x\in W}w(x)$
- 而这等于：
- $\sum_{x\in W}(\sum_{y\in H}[x支配y]+\sum_{y\in W}[x支配y])-\sum_{x\in W}(\sum_{y\in H}[y支配x]\sum_{y\in W}[y支配x])-\sum_{x\in W}w(x)$
- 即
- $\sum_{x\in W}\sum_{y=1}^{n}[x支配y]-\sum_{x\in W}\sum_{y=1}^{n}[y支配x]-\sum_{x\in W}w(x)$
- $=\sum_{x\in W}siz[x]-dep[x]-w(x)$，其中siz[x]为x的子树大小（不含自己），dep[x]为x的深度。
- 同理，后手总收益为$\sum_{x\in H}siz[x]-dep[x]-w(x)$
- 由上式可看出，每个点对收益的贡献是独立的。双方的最优策略即按照siz[x]-dep[x]-w[x]从大到小的顺序依次选取结点。

THUPC2023初赛 大富翁

- 咳咳，应该没有人做题的时候会这么想吧！
- ->n1-n2？想成对手给我n1块钱，我给对手n2块钱。
- ->对于每一对有祖先父子关系的点x,y，如果y和x不被同一个人占领，y会给x一块钱。
- ->对于每一对有祖先父子关系的点x,y，即使y和x被同一个人占领，y也给x一块钱——左手给右手，不赚不赔。
- ->每个点带来的贡献会是siz[x]-dep[x]-w[x]
- 排序后轮流取即可。

THUPC2023初赛 大富翁

- 注:
- 这不是一道博弈论的题目
- 这也不是一道推式子的题目

一个题

【题目描述】

给定一棵有 n 个结点的有根树，根结点为 1 号点。

每个点有权值 a_i，初始时均为 0，以及花费 v_i，表示对它进行一次操作时要花费的代价。

对点 u 进行一次参数为 x 的操作，即是在树中将它子树中的所有点的权值都加上 x。

要让**每个叶子**的权值互不相同且非零，即当 i,j（$i \neq j$）为叶子时要满足 $a_i \neq 0$ 且 $a_i \neq a_j$。

你需要构造一个总花费尽量小的操作序列。

测试点编号	$n \leq$	特殊限制
$1 \sim 4$	18	无
$5 \sim 6$	2×10^5	A
$7 \sim 9$		B
$10 \sim 13$		C
$14 \sim 25$		无

【输出格式】

输出到文件 ***segtree.out*** 中。

第一行，一个数 q，表示操作序列的长度，你需要保证 $0 \leq q \leq 2n$。

接下来 q 行，每行两个数 u, x，表示对点 u 进行了一次参数为 x 的操作，你需要保证 x 是正整数且 $1 \leq x \leq 10^9$。

你需要保证每次操作合法，且总花费是所有情况中最小的。

本题使用**自定义校验器**检验你的答案是否正确，因此若有多种满足条件的方案，你只需要输出**任意一种**。

5904 [POI2014] HOT-Hotels 加强版

lca

边权为1

题目描述

给出一棵有 n 个点的树，求有多少组点 (i, j, k) 满足 i, j, k 两两之间的距离都相等。

(i, j, k) 与 (i, k, j) 算作同一组。

ch[x][i]

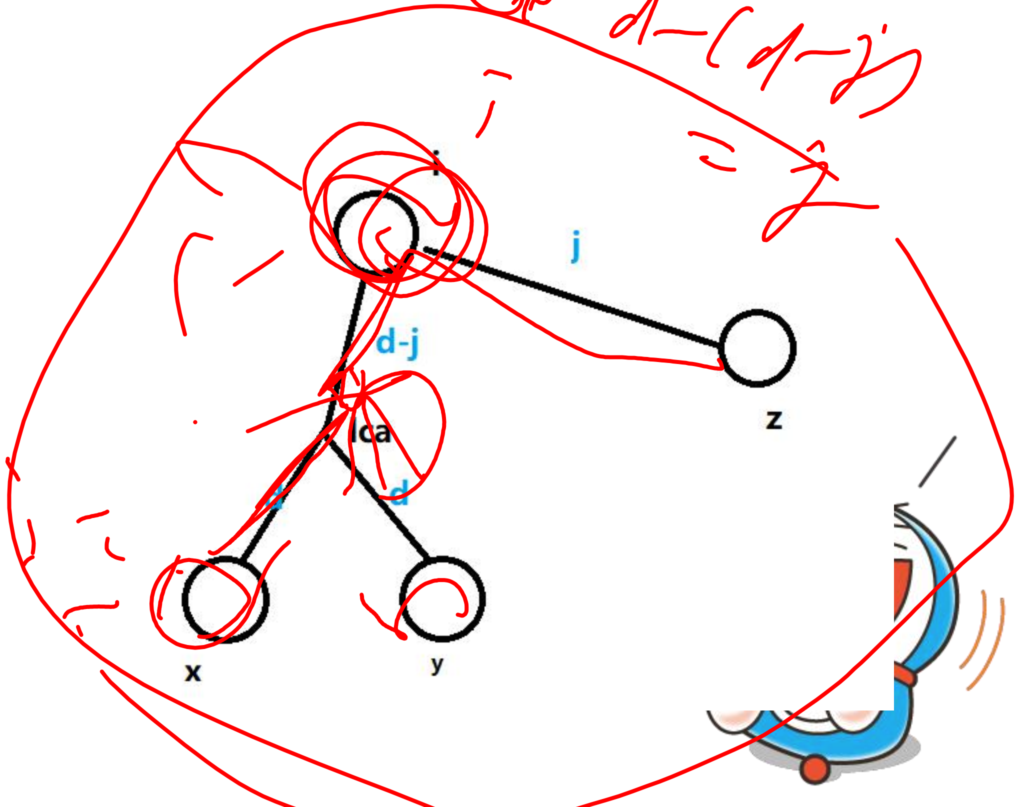
f[x][i]

- n<=1e5

n≤1e7

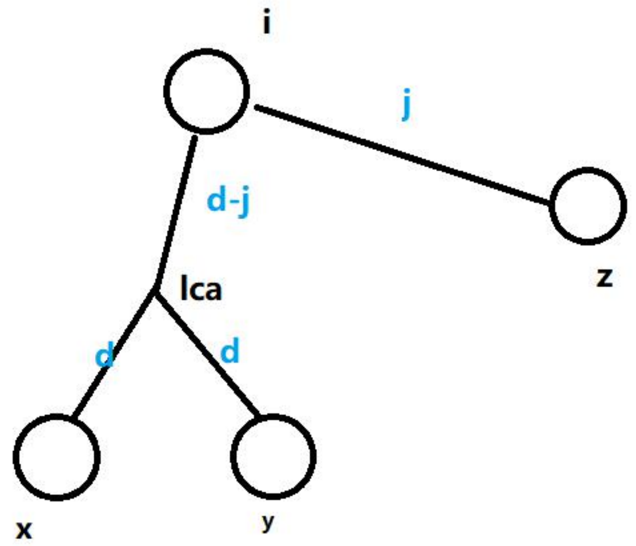
5904 [POI2014] HOT-Hotels 加强版

- 符合条件的(i,j,k)只可能是这种情况
- 我们选择在i处统计答案。需要记录其子树内有多少对点(x,y)，满足 $dis(x,lca(x,y))-dis(lca(x,y),i)= dis(y,lca(x,y))-dis(lca(x,y),i)= d$



5904 [POI2014] HOT-Hotels 加强版

- 设g[u][i]为u子树中，这样二元组的个数
- 为了求出g，我们需要额外一个数组f[u][i]表示u子树内，距离u为i的点的个数。
- ans += f[x][i]*g[v][j+1]
- ans += g[x][i]*f[v][j+1]
- g[x][i] += g[v][i+1]
- g[x][i] += f[x][i]*f[v][i-1]
- f[x][i] += f[v][i-1]
- 复杂度O(n^2)

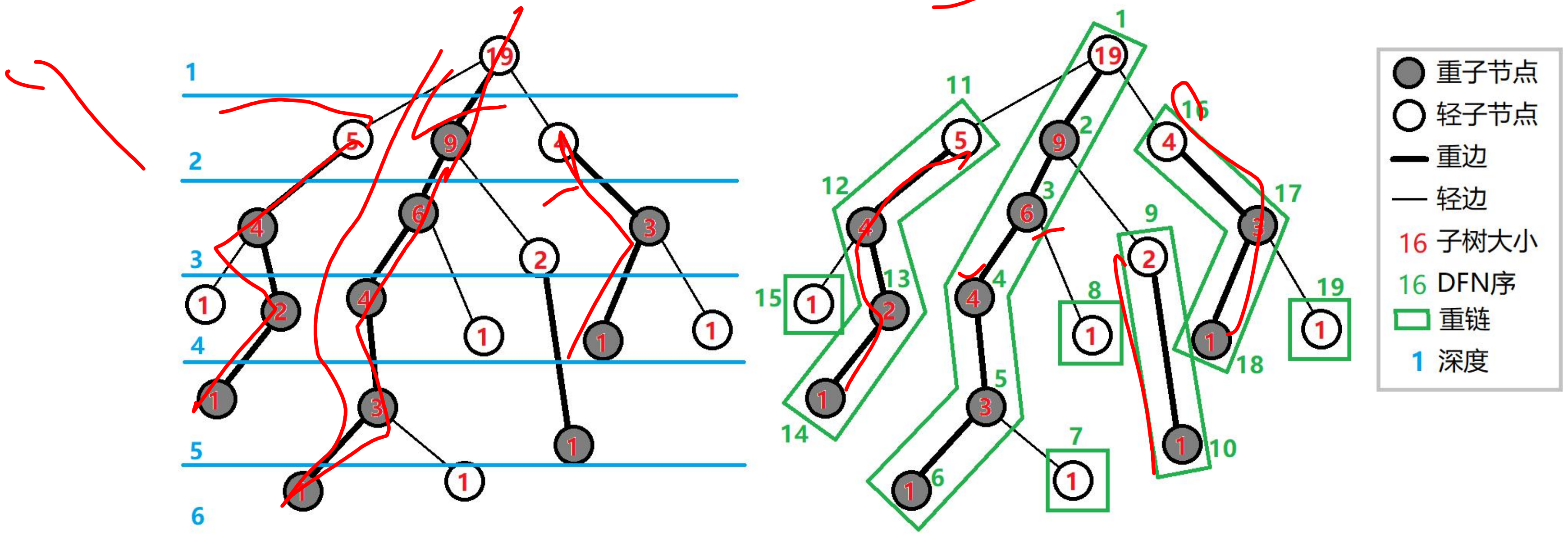


5904 [POI2014] HOT-Hotels 加强版

- 注意到DP数组的第二维和深度有关，且大小是x子树的深度，考虑使用**长链剖分**优化。

长链剖分

- 对于树上的结点x，定义其**重儿子**为其子树深度最大的子结点。其他子节点为轻儿子。
- 相应定义**重边**，**轻边**。若干首尾相连的重边称为**重链**。



credit: OI-wiki

长链剖分优化DP

- 对于每个点，如果我们能O(1)地直接处理其重儿子，再以O(lth[v])的复杂度分别处理其轻儿子v，那么整个DP的总复杂度为O(n)。其中lth[v]是v子树的深度，也即以v为顶点的长链长度。

5904 [POI2014] HOT-Hotels 加强版

- ~~ans += f[x][i]*g[v][j+1]~~
- ans += g[x][i]*f[v][j+1]
- g[x][i] += g[v][i+1]
- ~~g[x][i] += f[x][i]*f[v][i-1]~~
- f[x][i] += f[v][i-1]
- 对于本题，如何“O(1)地直接处理重儿子”？直接把它的f和g数组占为己用！
- 我们先合并重儿子，此时第一、二、四行转移不存在/本身O(1)。对于第三行和第五行，直接把重儿子的f/g数组拿过来用就好！
- 对于轻儿子们，正常转移就OK。

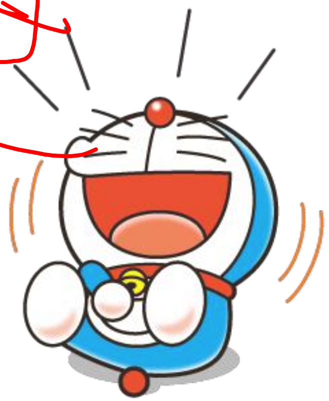
5904 [反

- 实现上，需 两倍自己
  最大深度的

```
void dfs(int x,int ff){
    f[x][0]=1;
    if(!mxs[x])return;
    f[mxs[x]]=f[x]+1,g[mxs[x]]=g[x]-1;
    dfs(mxs[x],x);
    ans+=g[mxs[x]][1];

    for(int h=frm[x];h;h=edge[h].nxt){
        int v=edge[h].to;
        if(v==ff||v==mxs[x])continue;
        f[v]=++totf;totf+=mxl[v];totg+=mxl[v],g[v]=++totg,totg+=mxl[v];
        dfs(v,x);
        for(int i=1;i<=mxl[v]+1;i++){
            ans+=1ll*g[x][i]*f[v][i-1];
            if(i!=mxl[v]+1)ans+=1ll*g[v][i]*f[x][i-1];
        }
        for(int i=1;i<=mxl[v]+1;i++){
            if(i!=mxl[v]+1)g[x][i-1]+=g[v][i];
            g[x][i]+=f[x][i]*f[v][i-1];
        }
        for(int i=1;i<=mxl[v]+1;i++)f[x][i]+=f[v][i-1];
    }
}
```

g[x]



一个题

- 问树中有几个联通块的直径介于[L,R]之间。n<=1e5

一个题

- 显然做差分。求直径介于[0,R]的个数-[0,L-1]的答案。
- 以[0,R]为例。显然设f[x]为x子树内，钦定x一定选，并且直径不超过R的连通块方案数。可能形成新的直径，发现再记个到x的最长链就行了。
- 于是记个f[x][h]为为x子树内，钦定x一定选，到x最长链长度为h并且直径不超过R的方案数。
- 转移：$f[x][\max(hu,hv+1)] += f[u][hu]*f[v][hv], hu+hv+1 \le R$。

一个题

- 同样注意到第二维和深度有关，使用长链剖分。
- 同样，先合并重儿子，直接把其f数组拿过来用就好，复杂度O(1)。
- 对于轻儿子，直接看转移式子： $f'[u][\max(hu, hv+1)] \mathrel{+}= f[u][hu] * f'[v][hv], hu + hv + 1 \le R$，我们需要同时枚举hu和hv。能不能只枚举hv?
- *我们用f[x]表示合并前的DP数组，用f’[x]表示合并后的。*

一个题

- 带着max总是不方便的，我们应该先分类讨论去掉max：
- $hu \le hv+1$：有$f'[u][hv+1] += f[v][hv] * (\sum_{hu+hv+1 \le R, du \le dv+1} f[u][du])$
- 而对于所有满足$hu > hv+1$的hu，$f'[u][du] += f[u][du] * f[v][dv]$。
- 第一种转移？区间求和，单点加！
- 第二种转移？区间乘法！
- 维护线段树。

一个题

- 更具体地，为了快速继承重儿子的DP数组，全局只建一棵线段树。在处理u时，就让线段树上[dfn[x],dfn[x]+lth[x]]这段区间的值为f[x][0]...f[x][lth[x]]。
- 还是先递归长儿子。回到父亲时，平移的任务自然就被完成。
- 复杂度O(nlogn)

谢谢大家！