动态规划

清华大学 张华清

6813 「MCOI-02」Glass 玻璃

题目描述

首先给定一棵树，每个点上有 V_i 个玻璃，每条边上有权值 W_i。

每次操作小 S 可以选择两个节点 $u, v (u \neq v)$，从节点 u 到节点 v 的唯一路径上，**边和** 为路径上所有边的权值和，即 $\sum W_i$，**点和** 为路径上所有点（包括 u, v）的玻璃数和，即 $\sum V_i$。小 S 将可以获得 **边和** 和 **点和** 的乘积那么分数，即 $\sum W_i \times \sum V_i$。

任意两次操作不能完全相同，(u, v) 和 (v, u) 被看作是两种操作。

但是有时候这颗树太过庞大，小 S 需要你的帮助。他需要你告诉他，经过 $N(N-1)$ 次操作后，总共能得到多少分。结果可能很大，你只需要输出答案对 998244353 取模的结果。

6813 「MCOI-02」Glass 玻璃

- 引入问题：求树上$N(N-1)$条路径的长度和。
- 对每条边拆贡献。
- 对于一条边(u,v)，它的长度会被统计多少次？——断开这条边形成的两棵子树大小的乘积。

6813 「MCOI-02」Glass 玻璃

- 回到原问题，一条边(u,v)会被贡献多少次?
- 断开(u,v)，把u/v作为两棵子树的根。记$Val[i] = siz[i] * V[i]$，即i子树大小和点权的乘积。
- 那么贡献即为u子树内val的和×v子树大小+ u子树大小*v子树内val的和。

6813 「MCOI-02」Glass 玻璃

- 随便选一个点当根。
- 当断开(u,v)后，假设v是u的儿子。v的子树内val的和好求，事先dfs一遍即可轻松求出。
- u的子树？此时为整棵树挖掉v的子树。需要换根DP。
- 先从f[u]中挖去f[v]的贡献，再处理u子树大小发生的变化。同时，u的爸爸现在成了u的儿子，把父亲子树带来的贡献加上。

```
1  int get_ans(int X,int Y){
2      up[X]=sub(f[X],f[Y]);//还要把爸爸那些匹配上
3      up[X]=sub(up[X],1ll*V[X]*siz[X]%p);//考虑一下自己子树的变化
4      up[X]=add(up[X],1ll*V[X]*(n-siz[Y])%p);
5      if(fa[X])up[X]=add(up[X],up[fa[X]]);
6      return add(1ll*up[X]*siz[Y]%p,1ll*f[Y]*(n-siz[Y])%p);
7  }
8
```

4284 [SHOI2014] 概率充电器

题目描述

复制Markdown 展开

著名的电子产品品牌 SHOI 刚刚发布了引领世界潮流的下一代电子产品——概率充电器：

“采用全新纳米级加工技术，实现元件与导线能否通电完全由真随机数决定！SHOI 概率充电器，您生活不可或缺的必需品！能充上电吗？现在就试试看吧！”

SHOI 概率充电器由 $n-1$ 条导线连通了 n 个充电元件。进行充电时，每条导线是否可以导电以概率决定，每一个充电元件自身是否直接进行充电也由概率决定。随后电能可以从直接充电的元件经过通电的导线使得其他充电元件进行间接充电。

作为 SHOI 公司的忠实客户，你无法抑制自己购买 SHOI 产品的冲动。在排了一个星期的长队之后终于入手了最新型号的 SHOI 概率充电器。你迫不及待地将 SHOI 概率充电器插入电源——这时你突然想知道，进入充电状态的元件个数的期望是多少呢？

对于 100% 的数据，$n \leq 5 \times 10^5$，$0 \leq p, q_i \leq 100$。

4284 [SHOI2014] 概率充电器

- 答案为每个点被点亮的概率的和。
- 我们反过来算每个点不被点亮的概率。

4284 [SHOI2014] 概率充电器

- 考虑1号点不被点亮的概率怎么算。 把1号点当根。
- 设f[x]为x不被x及子树内的点点亮的概率。（或者说，点亮是有方向的，只能儿子到父亲）
- $f[x] = (1 - q[x]) \prod_{v \in son(x)} (1 - (1 - f[v]) * p[v])$，其中q[x]是x被点亮的概率，p[v]是v的父亲边被点亮的概率。
- 那么$f[1]$即为1号点不被点亮的概率。

4284 [SHOI2014] 概率充电器

- 我们自然可以把每个点当根，分别DP一次，复杂度$O(n^2)$。怎么优化?
- 换根DP!

4284 [SHOI2014] 概率充电器

- 设g[x]为x不被点亮的概率（包括从子树内也包括被父亲点亮）。
- $g[x] = f[x] * \Big(1 - P(x被fa点亮)\Big)$。
- 而$P(x被fa点亮) = P(x的父亲边是亮的) * \Big(1 - P(fa是不亮的)\Big)$...吗?
- $P(x被fa点亮) = P(x的父亲边是亮的) * \Big(1 - P(fa是不亮的 | fa没有被x点亮)\Big)$!
- $P(fa是不亮的 | fa没有被x点亮) = \frac{g[fa]}{1-p[x](1-f[x])}$
- 则$g[x] = f[x] * (1 - p[x] * (1 - \frac{g[fa]}{1-p[x](1-f[x])}))$

APIO2014 连珠线

题目描述

复制Markdown 展开

在达芬奇时代，有一个流行的儿童游戏称为连珠线。当然，这个游戏是关于珠子和线的。线是红色或蓝色的，珠子被编号为 1 到 n。这个游戏从一个珠子开始，每次会用如下方式添加一个新的珠子：

`Append(w, v)`：一个新的珠子 w 和一个已经添加的珠子 v 用红线连接起来。

`Insert(w, u, v)`：一个新的珠子 w 插入到用红线连起来的两个珠子 u, v 之间。具体过程是删去 u, v 之间红线，分别用蓝线连接 u, w 和 w, v。

每条线都有一个长度。游戏结束后，你的最终得分为蓝线长度之和。

给你连珠线游戏结束后的游戏局面，只告诉了你珠子和链的连接方式以及每条线的长度，没有告诉你每条线分别是什么颜色。

你需要写一个程序来找出最大可能得分。即，在所有以给出的最终局面结束的连珠线游戏中找出那个得分最大的，然后输出最大可能得分。

APIO2014 连珠线

- 当最开始的唯一结点为根时，所有的蓝色边必须满足：两条蓝边为一组，每一组都是直上直下的。
- 假设1号结点即为最开始的唯一节点，进行树形DP：
- 设f[x][0/1]为x子树内，x是否是某个蓝边组的中间节点。
- $f[x][0] = \sum \max(f[v][0], f[v][1] + w_v)$，其中$w_v$表示v的父亲边的边权。
- $f[x][1] = \max_{v \in son_x} f[v][0] + w_v + \sum_{u \in son_x, u!=v} \max(f[u][0], f[u][1] + w_u).$
  即枚举蓝边组底下的点v是谁。

APIO2014 连珠线

- $f[1][0]$即为以1结点初始点时的答案。
- 于是我们有一个简单的$O(n^2)$做法，即分别枚举每个点当初始点，进行如上的DP。
- 如何优化？当然还是换根DP。

APIO2014 连珠线

- 我们设g[x][0/1]为以x为根时，x原来的爸爸（现在成为了x的儿子）的子树内的最大值，当fa[x]不是/是某个蓝色组的中间点。
- 进行第二遍dfs。当dfs到x时，g[x][0/1]已经在fa[x]处求出。那么以x为根的答案即$f[x][0]+\max(g[x][0],g[x][1]+w_x)$。
- 重要任务：求出x的各个儿子v的g[v][0/1]值。

APIO2014 连珠线

- g的求法和前面求[illegible]
- $g[v][0] = f[x][0$[illegible]$][1] + w_x)$
- $g[v][1]$同样需要[illegible]个不是v的儿子，或x的父[illegible]
- 预处理最大值和[illegible]否则用次大值。

```
void dp2(int x,int ff,int fe){
    ans=max(ans,f[x][0]+max(g[x][0],g[x][1]));//因为x是根。所以都不能延伸

    int mx=-0x3f3f3f3f,sec=-0x3f3f3f3f,pos=0;//第二种转移。要算上根的。记录最大和次大
    int sum= f[x][0]+max(g[x][0],g[x][1]);
    if(sum-max(g[x][0],g[x][1])+g[x][0]+fe>mx)mx=sum-max(g[x][0],g[x][1])+g[x][0]+fe,pos=ff;
    for(int i=frm[x];i;i=edge[i].nxt){
        int v=edge[i].to;if(v==ff)continue;
        int V=sum-max(f[v][0],f[v][1])+f[v][0]+edge[i].w;
        if(V>mx)sec=mx,mx=V,pos=v;
        else if(V>sec)sec=V;
    }

    for(int i=frm[x];i;i=edge[i].nxt){
        int v=edge[i].to;if(v==ff)continue;
        g[v][0]=f[x][0]-max(f[v][0],f[v][1])+max(g[x][0],g[x][1]);
        if(v!=pos){
            g[v][1]=mx-max(f[v][0],f[v][1])+edge[i].w;
        }else g[v][1]=sec-max(f[v][0],f[v][1])+edge[i].w;
        dp2(v,x,edge[i].w);
    }
}
```

2664 树上游戏

题目描述

复制Markdown 展开

lrb 有一棵树，树的每个节点有个颜色。给一个长度为 n 的颜色序列，定义 $s(i,j)$ 为 i 到 j 的颜色数量。以及

$$sum_i = \sum_{j=1}^{n} s(i,j)$$

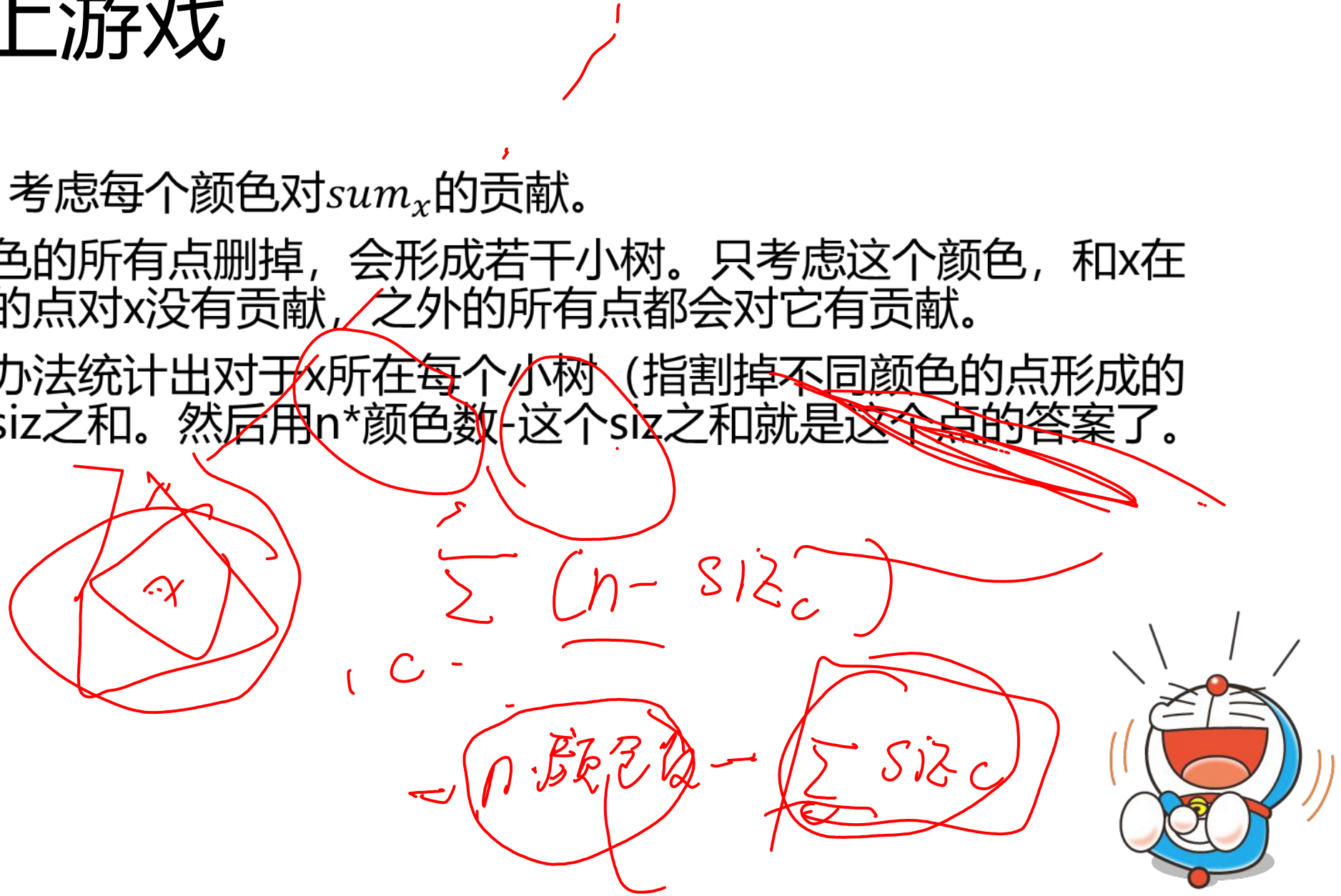
现在他想让你求出所有的 sum_i。

对于 40% 的数据，$n \leq 2000$。

对于 100% 的数据，$1 \leq n, c_i \leq 10^5$。

2664 树上游戏

- 对于一个点x，考虑每个颜色对sum_x的贡献。
- 考虑把这个颜色的所有点删掉，会形成若干小树。只考虑这个颜色，和x在同一个小树里的点对x没有贡献，之外的所有点都会对它有贡献。
- 所以我们想个办法统计出对于x所在每个小树（指割掉不同颜色的点形成的所有小树）的siz之和。然后用n*颜色数-这个siz之和就是这个点的答案了。



2664 树上游戏

- 自然，我们想知道每
- 当小树是以x号结点
- 我们只需要对每个x
- cols[x]等于x子树大小
- “**子树内**”这个限制
- 我们用一个额外数组sumcs[C]表示已经dfs过的，删C颜色形成小树的大小和。
- 特别地，1号点的爸爸看做有全部的颜色，特殊处理即可。

```
1  void dfs(int x,int ff) {
2      int C=col[ff];
3      int pres=sumcs[C];
4      sumcs[col[x]]+=1;
5      siz[x]=1;
6      for(int i=frm[x]; i; i=edge[i].nxt) {
7          int v=edge[i].to;
8          if(v==ff)continue;
9          dfs(v,x);
10         siz[x]+=siz[v];
11     }
12     if(x!=1) {
13         cols[x]=siz[x]-(sumcs[C]-pres);
14         sumcs[C]+=cols[x];
15     }
16 }
```

那么此时删除的颜色

为cols[x]。

有小树的大小之和。

的和，减进入x子树前

2664 树上游戏

- 知道cols[x]后，怎么得到x所在每个小树（指割掉不同颜色的点形成的所有小树）的siz之和?
- 只需要维护当前小树的siz之和，走到x这个点，假设它父亲的颜色是C，说明你离开了上一个C子树，进入了一个新的C子树，只需要减去原来的加上现在的就行了。这个过程中需要维护一个qwq[c]表示当前点所在颜色为c小子

```
1  void get_ans(int x,int ff) {
2      int C=col[ff];
3      int preC=qwq[C];
4      if(x!=1) {
5          sums-=qwq[C];
6          qwq[C]=cols[x];
7          sums+=qwq[C];
8          ans[x]=1ll*n*n-sums+qwq[col[x]];//可看做n种颜色，只是在1号点子树内（就是整个树）都没有出现过，整个树都是一个联通块
9      }
10     for(int i=frm[x]; i; i=edge[i].nxt) {
11         int v=edge[i].to;
12         if(v==ff)continue;
13         get_ans(v,x);
14     }
15     if(x!=1)sums-=qwq[C],qwq[C]=preC,sums+=qwq[C];
16 }
```

5666 [CSP-S2019] 树的重心

题目描述

复制Markdown 展开

小简单正在学习离散数学，今天的内容是图论基础，在课上他做了如下两条笔记：

1. 一个大小为 n 的树由 n 个结点与 $n-1$ 条无向边构成，且满足任意两个结点间**有且仅有**一条简单路径。在树中删去一个结点及与它关联的边，树将分裂为若干个子树；而在树中删去一条边（保留关联结点，下同），树将分裂为**恰好**两个子树。
2. 对于一个大小为 n 的树与任意一个树中结点 c，称 c 是该树的**重心**当且仅当在树中删去 c 及与它关联的边后，分裂出的所有子树的大小均**不超过** $\lfloor \frac{n}{2} \rfloor$（其中 $\lfloor x \rfloor$ 是下取整函数）。对于包含至少一个结点的树，它的重心只可能有 1 或 2 个。

课后老师给出了一个大小为 n 的树 S，树中结点从 $1 \sim n$ 编号。小简单的课后作业是求出 S 单独删去每条边后，分裂出的两个子树的重心编号和之和。即：

$$\sum_{(u,v)\in E} \left(\sum_{\substack{1\le x\le n \\ \text{且}x\text{号点是}S'_u\text{的重心}}} x + \sum_{\substack{1\le y\le n \\ \text{且}y\text{号点是}S'_v\text{的重心}}} y \right)$$

上式中，E 表示树 S 的边集，(u,v) 表示一条连接 u 号点和 v 号点的边。S'_u 与 S'_v 分别表示树 S 删去边 (u,v) 后，u 号点与 v 号点所在的被分裂出的子树。

小简单觉得作业并不简单，只好向你求助，请你教教他。

【数据范围】

测试点编号	$n=$	特殊性质
$1 \sim 2$	7	无
$3 \sim 5$	199	无
$6 \sim 8$	1999	无
$9 \sim 11$	49991	A
$12 \sim 15$	262143	B
16	99995	无
$17 \sim 18$	199995	无
$19 \sim 20$	299995	无

5666 [CSP-S2019] 树的重心

- 法一：
- 利用一棵树的重心一定在根节点所在的重链上的性质，不断跳重儿子，直到符合条件（最靠上的，2*重儿子大小<=整个子树大小）即为重心。
- 如何优化这个跳的过程？
- 倍增啊！预处理p[x][i]表示x沿着重儿子（即重链）走2^i步会走到哪，然后倍增地跳即可。
- 当我们断掉<u,v>这条边（v是u的儿子），那么以v为根的子树可以用p[x][i]数组向下跳，但以u为根的子树，我们不会向上跳啊...？
- 换根！

5666 [CSP-S2019] 树的重心

- 考虑现在断掉<u,v>这条边，发现u的儿子们发生了这样的变化：少了v子树，多了个大小为n-siz[u]的子树。
- 我们可以轻松找出u新的重儿子。
- 咋求倍增数组?
- 注意到走到<u,v>时，u的父亲们都是已经换好根了的，也即他们的倍增数组都可以看做是以u为根的。直接重新求一次即可。
- 注意dfs回溯时，要把各种信息还原。

5666 [CSP-S2019] 树的重心

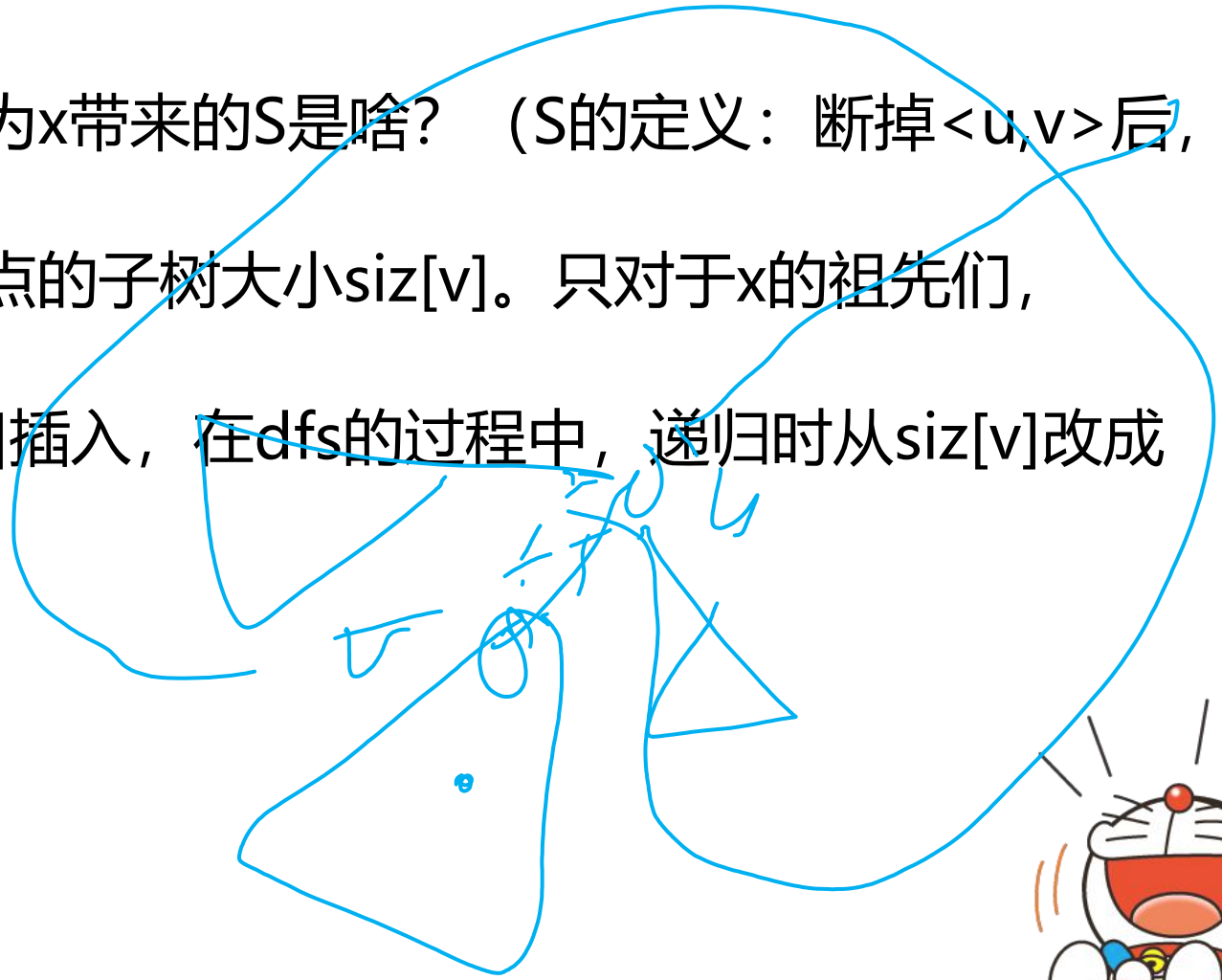
- 法二：
- 思路：考虑每个点的贡献，即对于每个点，统计有多少种断边方案使其成为重心。
- 首先，我们求出整棵树的一个重心rt，把它当作根。
  - 可能不容易想到。我们往往会随便选个1号点当根。
- 对于点x，考虑割哪些边E能让点x成为重心？
- 首先E显然不能在x子树内，这是重心为根所保证的。
- 其次，设割掉E剩下的两个子树中，不含x的那个子树大小为S，那么要求：
- $2mxn[x] \leq n - S$
- $2(n - S - siz[x]) \leq n - S$，其中$siz[x]$是x为根的子树大小，mxn[x]是x的各个子树的大小最大值。

5666 [CSP-S2019] 树的重心

- 整理一下，$n-2siz[x] \le S \le n-2mxn[x]$。
- 左右的界之和x有关，中间的S是和断的边E有关。考虑dfs整棵树，过程中用数据结构（如权值树状数组/权值线段树）把每条边带来的S记下，然后对[n-2siz[x],n-2mxn[x]]区间求和即可。

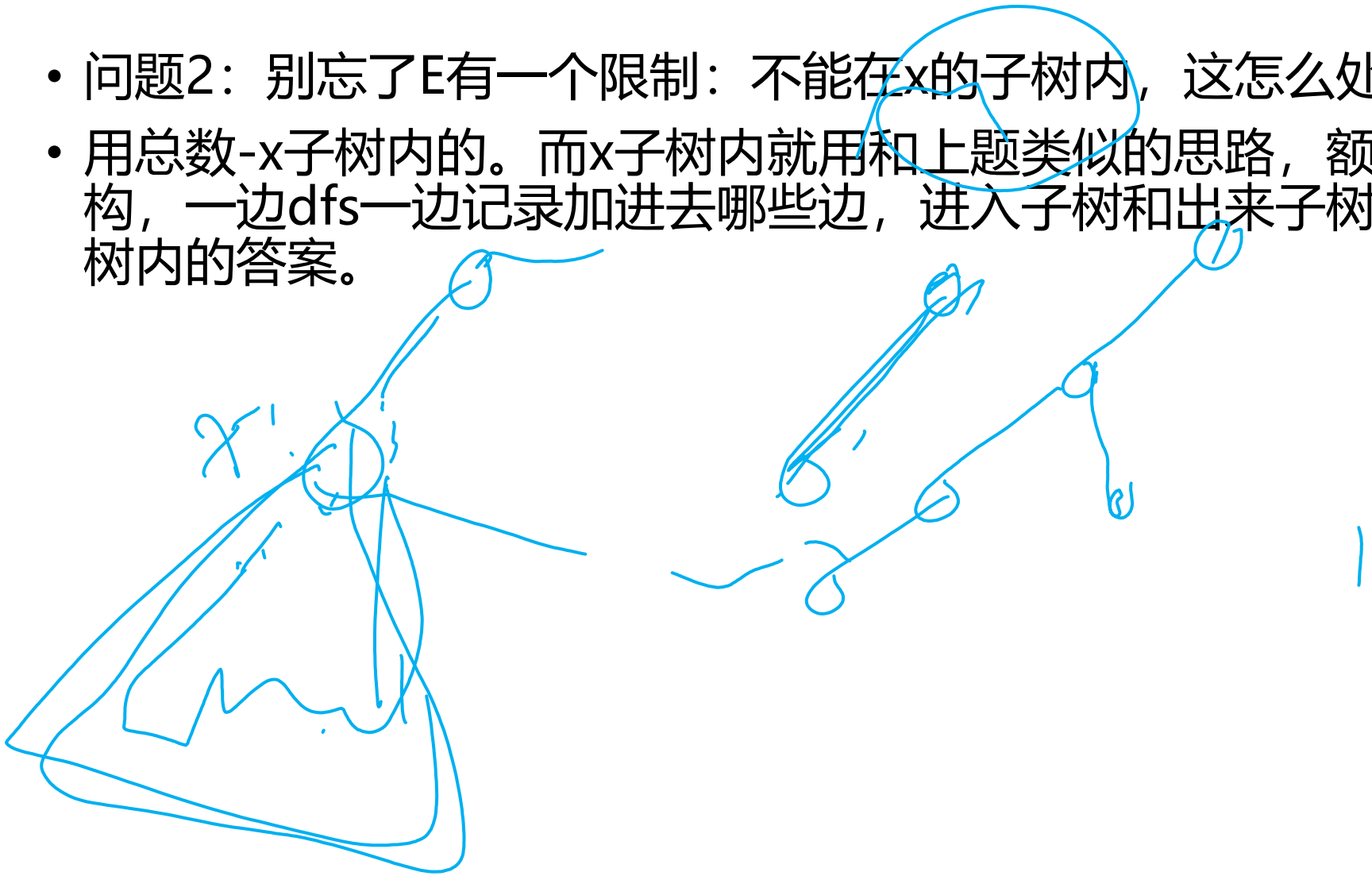
5666 [CSP-S2019] 树的重心

- 问题1：对于一条边<u,v>，它为x带来的S是啥？（S的定义：断掉<u,v>后，不含x的那个子树大小）。
- 对于大多数<u,v>，S即为较深点的子树大小siz[v]。只对于x的祖先们，S=n-siz[v]。
- 只需要先一股脑把所有S按siz[v]插入，在dfs的过程中，递归时从siz[v]改成n-siz[v]，回溯时改回来即可。



5666 [CSP-S2019] 树的重心

- 问题2：别忘了E有一个限制：不能在x的子树内，这怎么处理？
- 用总数-x子树内的。而x子树内就用和上题类似的思路，额外开一个数据结构，一边dfs一边记录加进去哪些边，进入子树和出来子树之间的差就是子树内的答案。



5666 [CSP-S2019] 树的重心

- 注：如果是在CSP/NOIP赛场遇到本题？
- 75分的暴力一定要拿下！

CTS2019 氪金手游

题目描述

复制Markdown 展开

小刘同学是一个喜欢氪金手游的男孩子。

他最近迷上了一个新游戏，游戏的内容就是不断地抽卡。现在已知：

- 卡池里总共有 N 种卡，第 i 种卡有一个权值 W_i，小刘同学不知道 W_i 具体的值是什么。但是他通过和网友交流，他了解到 W_i 服从一个分布。
- 具体地，对每个 i，小刘了解到三个参数 $p_{i,1}, p_{i,2}, p_{i,3}$，$W_i$ 将会以 $p_{i,j}$ 的概率取值为 j，保证 $p_{i,1} + p_{i,2} + p_{i,3} = 1$。

小刘开始玩游戏了，他每次会氪一元钱来抽一张卡，其中抽到卡 i 的概率为：

$$\frac{W_i}{\sum_j W_j}$$

对于全部的测试数据，保证 $N \leq 1000$，$a_{i,j} \leq 10^6$。

小刘会不停地抽卡，**直到**他手里集齐了全部 N 种卡。

抽卡结束之后，服务器记录下来了小刘**第一次**得到每张卡的时间 T_i。游戏公司在这里设置了一个彩蛋：公司准备了 $N-1$ 个二元组 (u_i, v_i)，如果对任意的 i，成立 $T_{u_i} < T_{v_i}$，那么游戏公司就会认为小刘是极其幸运的，从而送给他一个橱柜的手办作为幸运大奖。

游戏公司为了降低获奖概率，它准备的这些 (u_i, v_i) 满足这样一个性质：对于任意的 $\varnothing \neq S \subsetneq \{1, 2, \ldots, N\}$，总能找到 (u_i, v_i) 满足：$u_i \in S, v_i \notin S$ **或者** $u_i \notin S, v_i \in S$。

请你求出小刘同学能够得到幸运大奖的概率，可以保证结果是一个有理数，请输出它对 998244353 取模的结果。

CTS2019 氪金手游

- 注：本题是在所有w_x确定之后再开始游戏。

- 发现本题和SAO之间的关系了吗？/xyx
- 二元关系看作边，从u_i向v_i连边。题面最后的性质相当于说有向图弱联通。则把有向边视为无向边后得到一棵树。
- 题意即，每次以$\frac{w_i}{\sum_{x\ weibeixuan} w_x}$的概率选择i，最终得到原图拓扑序的概率。
- SAO不就是，每次以$\frac{1}{\text{没被选的个数}}$的概率选择i，求得到拓扑序的概率嘛！

CTS2019 氪金手游

- 还是先考虑只有外向边的情形。概率为$\prod_x \frac{w_x}{sumw[x]}$，其中$sumw[x]$表示x子树内所有点w值的和。
- 还是和SAO一样的容斥DP，只不过第二维从记录x为根的子树大小，变为记录x为根的子树内所有点w值的和。

Luogu 6799 「StOI-2」独立集

一棵由 n 个点组成的无根树，给定 m 条树上的路径，请求出由这 m 条路径组成的 独立集 方案总数。

由于这个答案可能很大，您只需求出它对 $998,244,353$ 取模的结果即可。

所谓 独立集 ，就是一个路径集合，满足这个集合中**不存在**一对在树上有交点的路径。特殊的，空集和只包括一条路径的集合也是独立集。

- n, m<=5e5

Luogu 6799 「StOI-2」独立集

- 先考虑序列做法。把链按右端点排序。设f[i]为考虑了前i条链的答案。转移考虑第i条链选不选。
- 不选？直接继承f[i-1]！
- 选？那就找最靠后的，与i不相交的链pos（可以二分），继承f[pos]。
- 即f[i]=f[i-1]+f[pos]

Luogu 6799 「StOI-2」独立集

- 下面我们试图把如上方法扩展到树上。
- 问题在于序列上每个点只有一个前驱，而树却又多个（多个儿子）
- 如果选择了一条链，序列情形：这段区间所有点都不能选；树上情形：链上的点不能选，但它的子树们可以选。
- 我们不再能设f[i]为考虑前i条链的答案，而应该设f[x]为考虑了x子树内的点。

Luogu 6799 「StOI-2」独立集

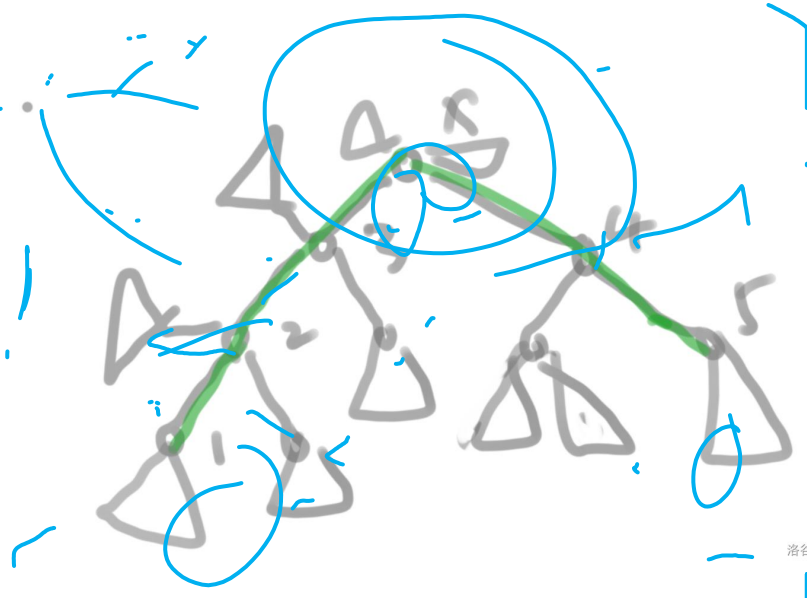
- 序列情形，我们选择按右端点对链们排序之后，在右端点考虑每一条链选不选。那树上问题，我们应在哪个位置决定这条链选不选呢？
- ——LCA处！

Luogu 6799 「StOI-2」独立集

- 设f[x][0/1]为仅考虑x子树内的链的方案数，其中x（没）有被覆盖。
- 记g[x]=f[x][0]+f[x][1]，即总方案数。
- f[x][0]?x自己不被覆盖，它的儿子们随便选。$f[x][0]=\prod_{v\in son(x)} g[v]$。
- f[x][1]?选且仅能选一条lca为x的链。加法原理，依次考虑lca为x的那些链选不选。

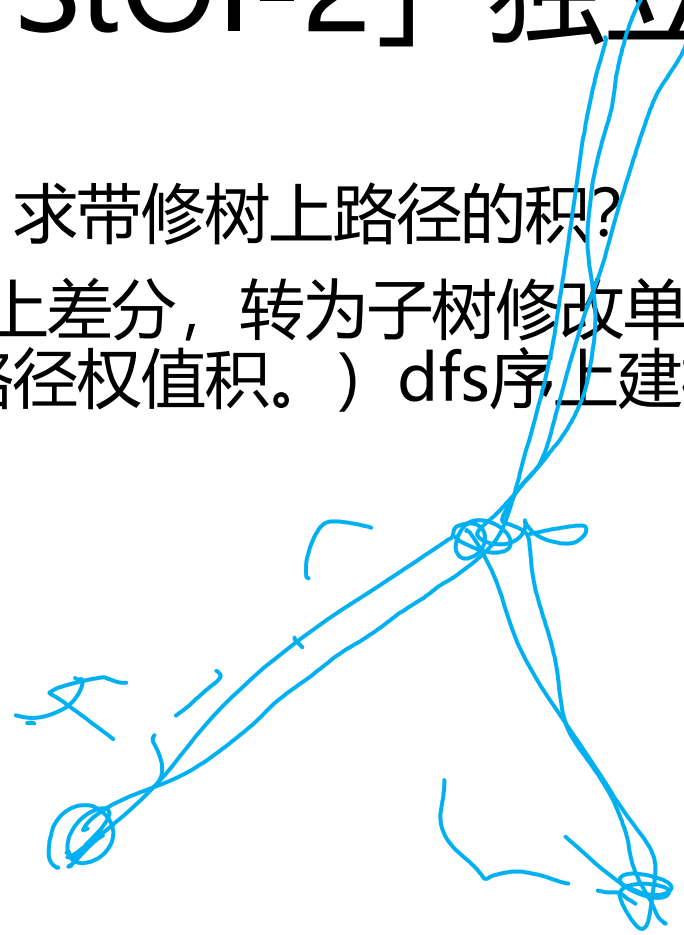
Luogu 6799 「StOI-2」独立集

- 画个图：假设选了这个链：我们发现x的大多数儿子都随便选（各个子树之间独立，直接把g乘进去），除了3和4两个在链上的点（不能被覆盖）。
- 那3子树方案数是多少？发现它大多数点也都可以随便选，除了2这个点（不能被覆盖），即f[3][0]/g[2]。
- 2子树？f[2][0]/g[1]。
- 1子树？它所有儿子随便选，即f[1][0]。
- 则本例中，
  f[x][1]=(f[x][0]/g[3]/g[4])*(f[3][0]/g[2])*(f[2][0]/g[1])*f[1][0]*(f[4][0]/g[5])*f[5][0]
- 推而广之，设h[u]=f[u][0]/g[u]，则答案为链上除了x所有点的h值乘起来，再乘上f[x][0]。



Luogu 6799 「StOI-2」独立集

- 接下来就是数据结构问题。求带修树上路径的积？
- 别树剖（两个log）！用树上差分，转为子树修改单点查询。（即，把每个点的权值的含义变为到根路径权值积。）dfs序上建树状数组即可。



Luogu 6799 「StOI-2」独立集

- 其实，还有一个问题：我们定义h[x]=f[x]/g[x]，但g[x]可能无逆元。
- DP转移式大概长这样：
  f[x][1]=(f[x][0]/g[3]/g[4])*(f[3][0]/g[2])*(f[2][0]/g[1])*f[1][0]*(f[4][0]/g[5])*f[5][0]
- 我们抛弃原来h的定义，转而定义为h[x]=f[fa[x]][0]/g[x]=$\prod_{v\in son(\mathrm{fa}[x]),v!=x} g[v]$。
- h[x]可以把fa[x]的所有儿子拉出来，求一个g的前缀和和后缀和，把x挖掉即可。
- 但f[x][0]/g[3]/g[4]就无法这么做。还得对每个点建线段树求区间和。

[NOIP2018 提高组] 保卫王国（完整版

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：

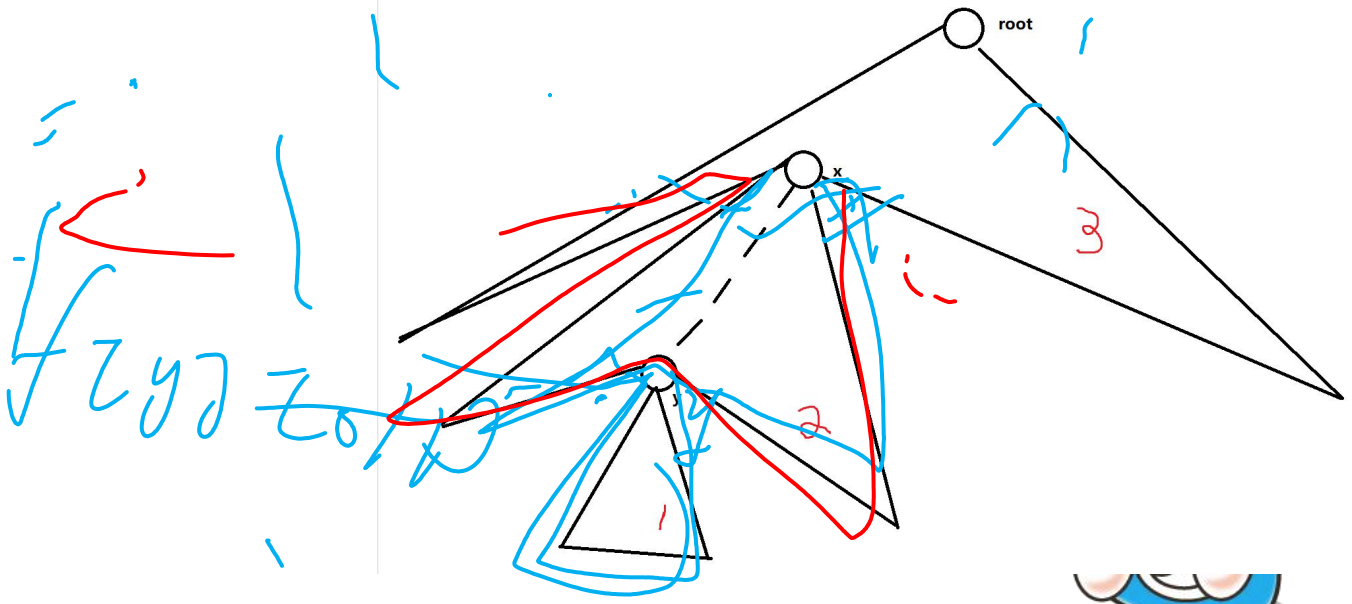
- 一座城市可以驻扎一支军队，也可以不驻扎军队。
- 由道路直接连接的两座城市中至少要有一座城市驻扎军队。
- 在城市里驻扎军队会产生花费，在编号为 i 的城市中驻扎军队的花费是 p_i。

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。

对于 100%的数据，保证 $1 \leq n, m \leq 10^5$，$1 \leq p_i \leq 10^5$，$1 \leq u, v, a, b \leq n$，$a \neq b$，$x, y \in \{0, 1\}$。

NOIP2018 保卫王国（完整版

- 假设修改了x和y。先考虑简单情形：x是y的祖先。
- 树可以分成三部分：1. y的子树　2.x的子树除去y的子树　3.整棵树除去x的子树。
- 当x和y的选取与否确定后，这三部分的花费是独立的，分别求出最小花费相加即可。



NOIP2018 保卫王国（完整版

- 第一部分很简单的，就是原来DP求出的。我们记f1[x][0/1]为以x为根的子树，x/不选时的最小花费。
- 第三部分也比较简单，我们记f2[x][0/]1为整棵树去掉以x为根的子树，当x选/不选时的最小化非。
- f2可以使用换根DP求出。
  - 我们现在求f2[v]，x是v的爸爸。
  - 当v不选取时，x必须选。f2[v][0]=f2[x][1]+(f1[x][1]-min(f1[v][0],f1[v][1]))。后面括号里是v的兄弟子树的花费。
  - 当v染色时，x可选可不选。f2[v][1]=min(f2[v][0],f2[x][0]+f1[x][0]-f1[v][1])。

NOIP2018 保卫王国（完整版

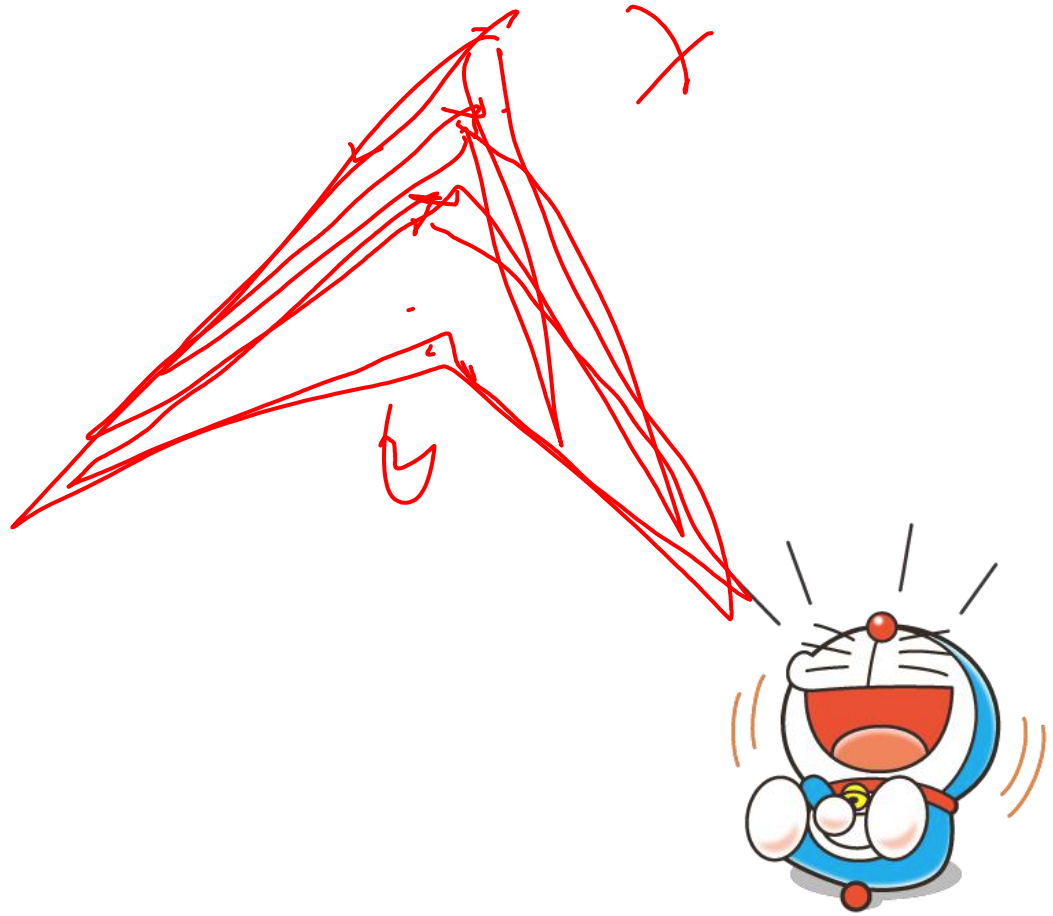
- 第三部分比较麻烦。看起来我们需要一个：f3[x][y][0/1][0/1]为x子树去掉y子树的部分，当x选/不选，y选/不选的最小花费。但显然复杂度不对劲。
- 倍增！
- 设dp[i][j][0/1][0/1]为，i的2^j级祖先的子树去掉i子树的部分，当i选/不选，当2^j级祖
- 边界：dp
- 倍增的预
可。其实

```
for(int i=1;i<=n;i++)
{
    dp[i][0][0][1]=dp[i][0][1][1]=f1[f[i][0]][1]-min(f1[i][0],f1[i][1]);//父节点染色
    dp[i][0][1][0]=f1[f[i][0]][0]-f1[i][1];//父节点不染色
}
for(int j=1;j<=19;j++)
    for(int i=1;i<=n;i++)
    {
        int fa=f[i][j-1];
        f[i][j]=f[fa][j-1];//先计算祖先
        for(int x=0;x<2;x++)//枚举当前点的状态
            for(int y=0;y<2;y++)//枚举第2^j辈祖先的状态
                for(int z=0;z<2;z++)//枚举第2^(j-1)辈祖先的状态
                    dp[i][j][x][y]=min(dp[i][j][x][y],dp[i][j-1][x][z]+dp[fa][j-1][z][y]);
    }
```

合并即

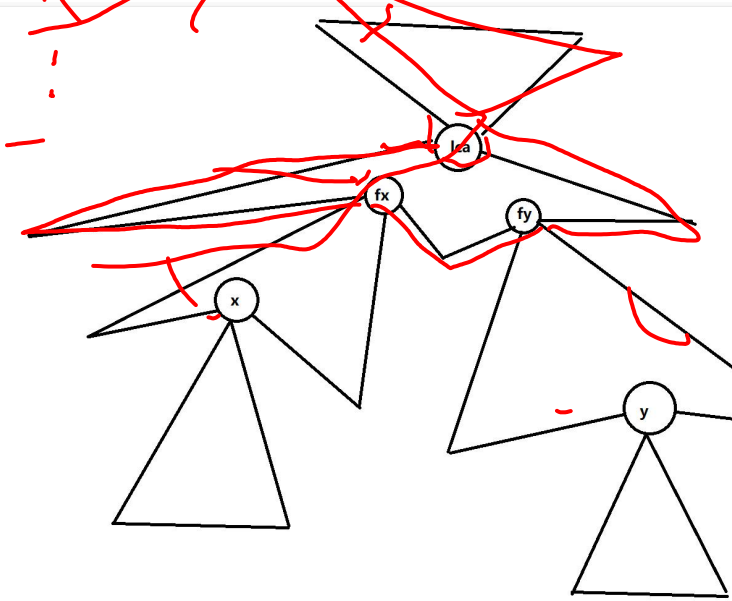
NOIP2018 保卫王国（完整版

- 预处理好DP数组后，就可倍增地求出我们想要的第三部分的答案，即x子树去掉y子树后的最小花费。
- 把这三部分的花费相加即可。



NOIP2018 保卫王国（完整版

- 再考虑一般的情况。fx/fy是子树内含有x/y的lca的儿子。
- 用上面的方法，可以求出fx子树内，当x选/不选的答案（只需要用x子树的答案+ fx子树去掉x子树的答案。后者倍增已求出）。同理可求fy子树内，当y选/不选的答案。
- 接下来，用这两个答案求出lca子树内，当x,y选/不选的答案。只需要用f1[lca][0/1]去掉原来fx,fy的贡献，再合并上这两个答案。
- 再加上整棵树去掉lca子树的答案（f2数组）即可。



CF814E An unavoidable detour for home

题意翻译

给出 n 个点，和每个点的度 d_i 让你构造出一张无向图满足以下两条性质：

- 点 1 到点 i 仅有**唯一**一条最短路。
- 点 1 到点 i 的最短路长度大于等于点 1 到点 $i-1$ 的最短路长度。

求能构成满足条件的无向图的个数?

$n \leq 50$，$2 \leq d_i \leq 3$。

CF814E An unavoidable detour for home

- 先观察性质。
- 把整张图按照到1号点的最短路长度分层。下一层的每个点会向上一层的某个点连恰好一条边
- 层之内可以连边
- 除此之外没有其他边。

CF814E An unavoidable detour for home

- 自然，我们想要DP。
- 需要知道?
- 上一层留有几个度数和下一层相连，这样就知道下一层会有几个点了。
- 这还不够，为不重复统计，需要知道上一层几个点在下一层留有1个度数（称为1接口），几个点留有2个度数（称为2接口）。
- 设f[i][x][y]为考虑了前i个点，最后一层留下x个1接口和y个2接口的方案数。
- 一个一个转移？不太行，需要同时记录两层的信息。
- 一层一层地转移!

CF814E An unavoidable detour for home

- 考虑f[i][x][y]能转移到哪?
- 下一层的点数一定为i+x+2y，即f[i][x][y]->f[i+x+2y][x’][y’]
- 这样枚举的复杂度已经n^5了。
- 如何优化？套路：**分层转移。**

CF814E An unavoidable detour for home

- 转移系数来自于两部分：层之间的连边和层内部的连边
- 当x+2y（即下一层的点数）相同时，两层之间的转移系数只和x,y相关；下一层层内部的转移系数（即连边方案数）只和x′,y′有关。二者是独立的，于是我们可以分层转移。
- 外面枚举i。
- 先枚举L=x+2y（即下一层点数）。计算层之间的转移系数$C = \sum_{x+2y=L} f[i][x][y] * \frac{(x+2y)!}{2^y}$。
- 再计算下一层内部的转移系数。设g[t][x′][y′]为[i+1,i+t]里的这些点，向下有x′个1接口，y′个2接口的方案数。简单讨论即可。
- 然后$f[i+L][x'][y'] \mathrel{+}= C * g[L][x'][y']$
- 复杂度O(n^4)。

谢谢大家！