树的直径重心 基环树

华南理工大学 江熠玲

2020 年 2 月 1 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。

引言：基本概念

引言：基本概念

图论〔Graph Theory〕是数学的一个分支。它以图为研究对象。图论中的图是由若干给定的点及连接两点的线所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用点代表事物，用连接两点的线表示相应两个事物间具有这种关系。

目录

目录

树的直径

树的直径

树是连通无环图，树上任意两点之间的路径是唯一的。定义树上任意两点u, v的距离为u到v路径上边权的和。树的直径MN为树上最长路径，即点M和N是树上距离最远的两个点。

树的直径

树的直径

算法：两次*dfs*或*bfs*。

① 从树上任意一点出发P，找到与其距离最大的点M

② 从点M出发，找到与其距离最大的点N

③ MN即为树的直径

树的直径

算法：两次dfs或bfs。

① 从树上任意一点出发P，找到与其距离最大的点M

② 从点M出发，找到与其距离最大的点N

③ MN即为树的直径

时间复杂度：$O(n)$，n为结点个数。

树的直径

树的直径

如何理解？

树的直径

如何理解？

只需证M是树的直径的一个端点，如果M是直径的一个端点，由定义知MN为树的直径。

树的直径

如何理解？

只需证M是树的直径的一个端点，如果M是直径的一个端点，由定义知MN为树的直径。

反证法：假设M不是直径的一个端点，AB是树的直径。

树的直径

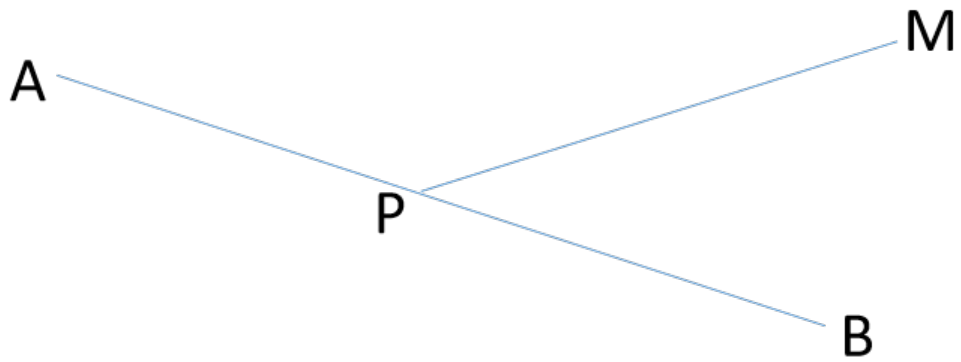
如何理解？

只需证M是树的直径的一个端点，如果M是直径的一个端点，由定义知MN为树的直径。

反证法：假设M不是直径的一个端点，AB是树的直径。

① 如果P是直径上的点，如图，

$PM > PB$则$AP + PM > AP + PB = AB$ 这与AB是直径矛盾。



树的直径

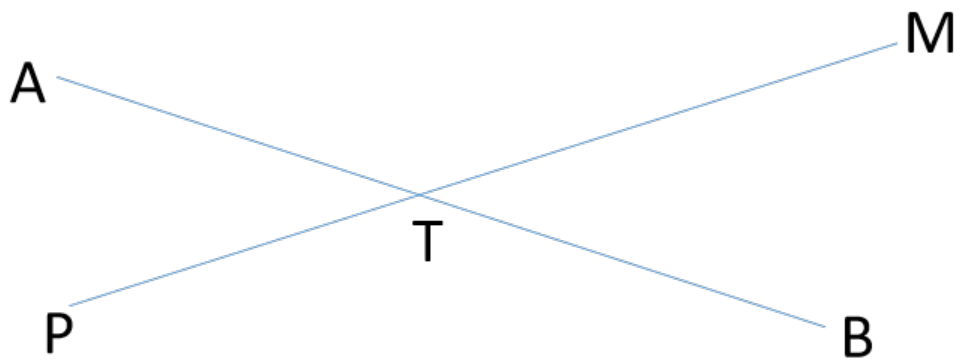
树的直径

若P不在直径上：

②-1 P到M路径与A到B路径有公共结点T

$PT + TM > PT + TB$，则$TM > TB$，

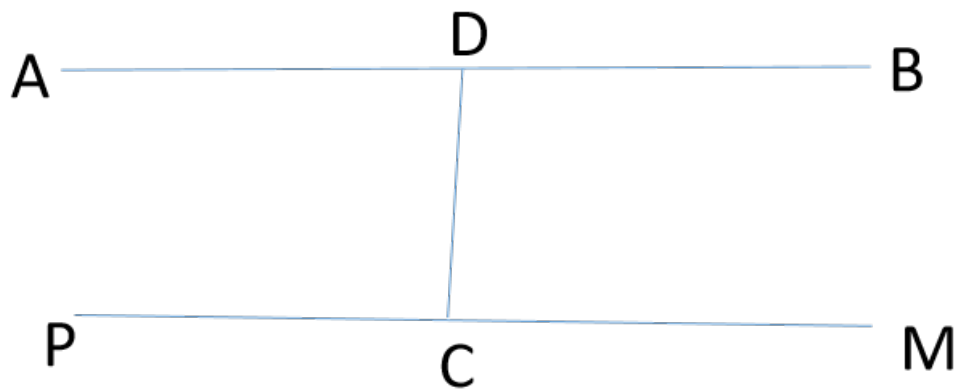
故$AT + TM > AT + TB = AB$，矛盾



树的直径

树的直径

②-2 P到M的路径与A到B的路径无公共结点$PC + CM > PC + CD + BD$，则$CM > CD + BD$，$CM + CD > BD$ 故$CM + CD + AD > BD + AD = AB$，矛盾。



树的直径

树的直径

```
void bfs(int s)
{
    int i,j,now;
    queue<int> q;
    memset(v,0,sizeof(v));
    memset(dis,0,sizeof(dis));
    q.push(s);v[s]=1;
    while(!q.empty())
    {
        now=q.front();q.pop();
        for(i=last[now];i;i=e[i].next)
            if(!v[e[i].to])
            {
                v[e[i].to]=true;
                dis[e[i].to]=dis[now]+e[i].v;
                q.push(e[i].to);
            }
    }
    for(i=1,ans=0;i<=n;i++)
        if(dis[i]>ans)
        {
            ans=dis[i];
            point=i;
        }
}
int main()
{
    bfs(s);bfs(point);
}
```

树的重心

树的重心

树的重心可以通过简单的两次搜索求出，第一遍搜索求出每个结点的子结点数量$size[x]$，第二遍搜索找出$mx[x]$表示删去x之后的最大子树结点数(若有多个则取编号最小的那个)。

实际上这两步操作可以在一次遍历中解决。

树的重心

树的重心

```
void dp(int x,int fa)
{
    size[x]=1;
    for(int i=0;i<e[x].size();i++)
    {
        int y=e[x][i];
        if(y==fa)continue;
        dp(y,x);
        size[x]+=size[y];
        mx[x]=max(mx[x],size[y]);
    }
    mx[x]=max(mx[x],n-size[x]);
    if(mx[x]<mx[ans])ans=x;
    if(mx[x]==mx[ans]&&x<ans)ans=x;
}
```

树的重心

树的重心

还有一种方法，虽然不能保证求出来的一定是重心，但是能找到一个结点，而这个以这个结点为根的话，所有的子树的结点数量都不会超过总结点数的一半。

树的重心

还有一种方法，虽然不能保证求出来的一定是重心，但是能找到一个结点，而这个以这个结点为根的话，所有的子树的结点数量都不会超过总结点数的一半。

同样是先搜索出$size[u]$，从某个假设的根开始向下找，如果有子结点v，使$size[v] > size[u]/2$，就以v作为新的根，重复执行，直到没有满足条件的子结点。

基环树

基环树

基环树是一种图，它由一个环组成，环上每个点都是一棵树点树根，所以称为基环树。当然，一棵树上连一条边也会变成基环树。

基环树

基环树是一种图，它由一个环组成，环上每个点都是一棵树点树根，所以称为基环树。当然，一棵树上连一条边也会变成基环树。

基环树一般分成环和树来分别处理（显然环的处理较为麻烦），首先得找到环。

基环树

基环树

```
void get_loop(int u) {
    vis[u] = ++vs;
    for (int i = head[u]; ~i; i = edge[i].nxt) {
        int v = edge[i].to;
        if(v == fa[u]) continue;
        if(vis[v]) {
            if(vis[v] < vis[u]) continue;
            loop[++cnt] = v;
            for ( ;v != u; v = fa[v]) {
                loop[++cnt] = fa[v];
            }
        }
        else fa[v] = u, get_loop(v);
    }
}
```

基环树

基环树

基环内向树：有向图，在基环树的基础上每个节点的出度为1。

基环外向树：有向图，在基环树的基础上每个节点的入度为1。

目录

目录

[POJ 1985]Cow Marathon

[POJ 1985]Cow Marathon

有n个农场，这n个农场有一些边连着，然后要你找出两个点，使得这一对点的路径长度最大，输出这个最大的长度。

$1 \leq n \leq 40000$。

[POJ 1985]Cow Marathon

[POJ 1985]Cow Marathon

树的直径。

[POJ 1849]Two

[POJ 1849]Two

有一颗n个结点的带权的无向树，在s结点放两个机器人，这两个机器人会把树的每条边都走一遍，但是最后机器人不要求回到出发点。问你两个机器人走的路总长之和的最小值是多少?

[POJ 1849]Two

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？$2 \times \sum w_i -$(从出发点所能到达的最远距离)。即沿着最远距离走，过程中每个分叉走两遍。

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？$2 \times \sum w_i -$(从出发点所能到达的最远距离)。即沿着最远距离走，过程中每个分叉走两遍。

假设只有2个机器人遍历树,且要求回到原点, 它最少需要走多少路?

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？$2 \times \sum w_i -$(从出发点所能到达的最远距离)。即沿着最远距离走，过程中每个分叉走两遍。

假设只有2个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

[POJ 1849]Two

假设只有1个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？$2 \times \sum w_i -$(从出发点所能到达的最远距离)。即沿着最远距离走，过程中每个分叉走两遍。

假设只有2个机器人遍历树,且要求回到原点, 它最少需要走多少路?$2 \times \sum w_i$。

若不用回到原点？$2 \times \sum w_i -$(树的直径)。

[POJ 1849]Two

[POJ 1849]Two

考虑从一个结点遍历整个树再回到原点需要把每个边计算两遍，这里机器人不用回到出发点，所以两个机器人到达的点越远越好。

要使路程最近，若起点在树的直径上，则两辆车往不同的方向走，直径上的边只用走一遍，其他的要走两遍。

若起点不在直径上，则两人一起走到直径上，再往不同的方向走。

这样最优解就是所有边$\times 2-$直径，因为直径只走了一次，而其他边必走两遍。

[POJ 1655]Balancing Act

[POJ 1655]Balancing Act

（多组数据）给出一棵树，求出这颗树的重心以及重心子树中节点数最多的子树的节点数。

[POJ 1655]Balancing Act

[POJ 1655]Balancing Act

模板。

网络会议

网络会议

在一棵节点数为n 的树上寻找一个点使得其他点到它的距离之和最小。

$1 \leq n \leq 50000$。

网络会议

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

假设我们已经知道了所有点到点i 的距离的总和，点j 和点i 相连，那么我们如何求出所有点到点j 的距离的总和呢？

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

假设我们已经知道了所有点到点i 的距离的总和，点j 和点i 相连，那么我们如何求出所有点到点j 的距离的总和呢？

设$s[i]$ 表示i 子树的节点个数，$d[i]$ 表示以i 为会议地点的答案。

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

假设我们已经知道了所有点到点i 的距离的总和，点j 和点i 相连，那么我们如何求出所有点到点j 的距离的总和呢？

设$s[i]$ 表示i 子树的节点个数，$d[i]$ 表示以i 为会议地点的答案。

那么通过观察，我们会发现一个这样的转移式子:

$$d[j] = d[i] + (n - s[j]) - s[j] = d[i] + (n - 2 \times s[j])$$

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

假设我们已经知道了所有点到点i 的距离的总和，点j 和点i 相连，那么我们如何求出所有点到点j 的距离的总和呢？

设$s[i]$ 表示i 子树的节点个数，$d[i]$ 表示以i 为会议地点的答案。

那么通过观察，我们会发现一个这样的转移式子:

$$d[j] = d[i] + (n - s[j]) - s[j] = d[i] + (n - 2 \times s[j])$$

这样当我们知道$d[i]$ 即可通过转移得到$d[j]$，时间复杂度$O(n)$。

网络会议

这时候我们考虑一下，能不能用手上已知的答案，通过一些转移，得到另一些未知的相邻点的答案。

假设我们已经知道了所有点到点i 的距离的总和，点j 和点i 相连，那么我们如何求出所有点到点j 的距离的总和呢？

设$s[i]$ 表示i 子树的节点个数，$d[i]$ 表示以i 为会议地点的答案。

那么通过观察，我们会发现一个这样的转移式子:

$$d[j] = d[i] + (n - s[j]) - s[j] = d[i] + (n - 2 \times s[j])$$

这样当我们知道$d[i]$ 即可通过转移得到$d[j]$，时间复杂度$O(n)$。

大范围数据用dfs 实现算法的话可能会栈溢出，因此想通过全部数据点，需要使用bfs 实现算法。

网络会议

网络会议

考虑到所有边权均为1。

网络会议

考虑到所有边权均为1。

$max\{siz[x]-1, n-siz[x]\}$ 最小的那一个即为答案，即树的重心。

[HDU 5886]Tower Defence

[HDU 5886]Tower Defence

给出一棵树，边上有权值，现在毁掉任意一条边，分成两部分，求这两部分中最远的两点距离期望，答案$\times(n-1)$

[HDU 5886]Tower Defence

[HDU 5886]Tower Defence

题目要求乘以$n-1$，所以直接把概率消掉了，所以只要求割去每条边后的最长链长度即可。

[HDU 5886]Tower Defence

题目要求乘以$n-1$，所以直接把概率消掉了，所以只要求割去每条边后的最长链长度即可。

1 最长链没有被拆开，

[HDU 5886]Tower Defence

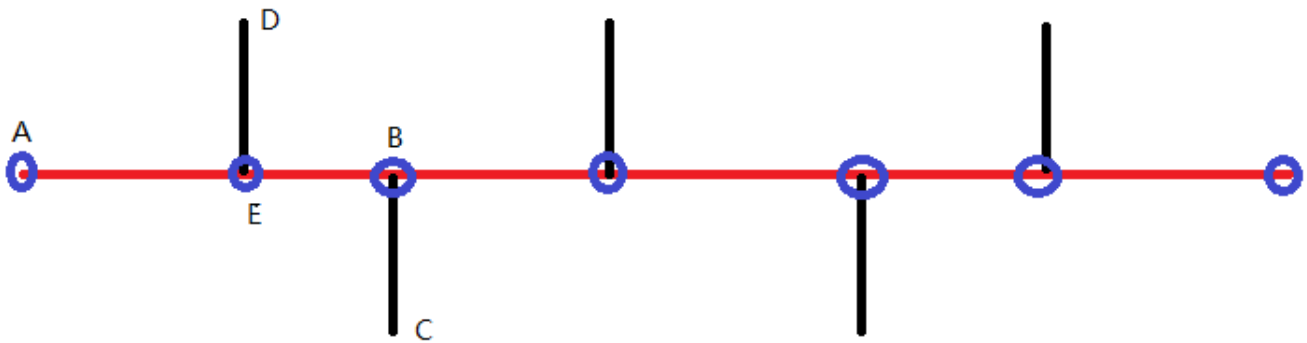
题目要求乘以$n-1$，所以直接把概率消掉了，所以只要求割去每条边后的最长链长度即可。

1. 最长链没有被拆开，那么答案就是最长链;
2. 最长链被拆开了，

[HDU 5886]Tower Defence

题目要求乘以$n-1$，所以直接把概率消掉了，所以只要求割去每条边后的最长链长度即可。

1. 最长链没有被拆开，那么答案就是最长链;
2. 最长链被拆开了，那么答案一定在最长链的端点到某个叶子结点上。



[HDU 5886]Tower Defence

[HDU 5886]Tower Defence

有了这个结论，这个题就好做了，首先预处理出最长链，数组存下来，然后对于每个最长链上的结点预处理出它的路径权值最大的叶子结点（不在最长链上），剩下的就是处理最长链左边和右边的拆开后的最大值了，左右扫一遍，两个数组L和R存下来，最后扫一遍最长链，取L和R中最大值得到答案。

[NOIP2018 D2T1]旅行

[NOIP2018 D2T1]旅行

一个n个点，m条边的连通图。可以从任意一个点出发，前往任意一个相邻的未访问的结点，或沿着上一次来这个点的边返回。需要遍历每一个点。每经过一个新的结点，就将这个结点写下来。最终可以得到一个序列。求字典序最小的序列。

$n \leq 5000, m \leq n$。

[NOIP2018 D2T1]旅行

[NOIP2018 D2T1]旅行

$n < m$：对于树的情况，显然从1出发，每次从字典序最小的相邻结点*DFS*即可。

[NOIP2018 D2T1]旅行

$n < m$：对于树的情况，显然从1出发，每次从字典序最小的相邻结点DFS即可。

$n = m$: 对于有环的情况，由于环只有一个，我们可以将环找出来，枚举删掉环上的每一条边，然后按树的情况求解即可。

[NOIP2018 D2T1]旅行

$n < m$：对于树的情况，显然从1出发，每次从字典序最小的相邻结点DFS即可。

$n = m$: 对于有环的情况，由于环只有一个，我们可以将环找出来，枚举删掉环上的每一条边，然后按树的情况求解即可。

时间复杂度$O(n^2)$。

[BZOJ 1791][IOI 2008]Island

[BZOJ 1791][IOI 2008]Island

有一个基环树和树组成的森林，共有n个结点，边有边权。试求其中所有联通块的直径之和。$2 \leq n \leq 10^6$

[BZOJ 1791][IOI 2008]Island

[BZOJ 1791][IOI 2008]Island

答案为各部分直径之和。

[BZOJ 1791][IOI 2008]Island

答案为各部分直径之和。

如何求基环树直径？

[BZOJ 1791][IOI 2008]Island

答案为各部分直径之和。

如何求基环树直径？

找出基环树的环，求出以环上每个点为根节点的树的直径以及深度，然后在环上求边权前缀和，枚举j, 那么答案就是$sum[j] - sum[i] + deep[j] + deep[i]$，

[BZOJ 1791][IOI 2008]Island

答案为各部分直径之和。

如何求基环树直径？

找出基环树的环，求出以环上每个点为根节点的树的直径以及深度，然后在环上求边权前缀和，枚举j, 那么答案就是$sum[j] - sum[i] + deep[j] + deep[i]$，用单调队列维护$max\{deep[i] - sum[i]\}$即可。

[CodeForces 835F]Roads in the Kingdom

[CodeForces 835F]Roads in the Kingdom

有一棵n个结点的基环树，边有边权。需要从环上删去一条边，以最小化直径。输出最小化后的直径。

$n \leq 2 \times 10^5$。

[CodeForces 835F]Roads in the Kingdom

[CodeForces 835F]Roads in the Kingdom

枚举删掉的是哪一条边。

[CodeForces 835F]Roads in the Kingdom

枚举删掉的是哪一条边。

路径长度是$sum_j + dep_j - sum_i + dep_i$。我们用两个set维护所有$sum_i + dep_i$和$-sum_i + dep_i$。这样就能求直径只要从两个set 中分别取出最大值就好了。当然，还要特判两个最大值是同一个结点的情况。

时间复杂度$O(nlogn)$。

[BZOJ 1040][ZJOI 2008]骑士

[BZOJ 1040][ZJOI 2008]骑士

n个点n条边的图，每个点都有点权，要求找到一个点集，点集中的点相互之间不能有边相连，最大化点集的权值和。

$1 \leq n \leq 10^6$

[BZOJ 1040][ZJOI 2008]骑士

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

如果是一棵树该怎么做？

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

如果是一棵树该怎么做？DP。

$f[i][0] = \sum max(f[son[i]][0], f[son[i][1])$

$f[i][1] = \sum f[son[i]][0]$

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

如果是一棵树该怎么做？DP。

$f[i][0] = \sum max(f[son[i]][0], f[son[i][1])$

$f[i][1] = \sum f[son[i]][0]$

在每一棵基环树的环上枚举一条边，记它的两个端点为u和v，然后删掉这条边做树形dp即可。

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

如果是一棵树该怎么做？DP。

$f[i][0] = \sum max(f[son[i]][0], f[son[i][1])$

$f[i][1] = \sum f[son[i]][0]$

在每一棵基环树的环上枚举一条边，记它的两个端点为u和v，然后删掉这条边做树形dp即可。

从该边的两个端点出发选择：

1. 强制不选u，v任意，环的贡献为以u做DP的$f[u][0]$。
2. 强制不选v，u任意，环的贡献为以v做DP的$f[v][0]$。

[BZOJ 1040][ZJOI 2008]骑士

如果联通的话，就是一个基环树了，否则为基环树森林。这道题可以简单的抽象为：基环树的最大独立集。

如果是一棵树该怎么做？DP。

$f[i][0] = \sum max(f[son[i]][0], f[son[i][1])$

$f[i][1] = \sum f[son[i]][0]$

在每一棵基环树的环上枚举一条边，记它的两个端点为u和v，然后删掉这条边做树形dp即可。

从该边的两个端点出发选择：

1. 强制不选u，v任意，环的贡献为以u做DP的$f[u][0]$。
2. 强制不选v，u任意，环的贡献为以v做DP的$f[v][0]$。

[BZOJ 2878] 迷失乐园

[BZOJ 2878] 迷失乐园

有一棵n个结点的基环树，边有边权。当你从一个结点出发后，你每次会等概率地选择下一个未被访问过的结点，并走到那个结点，直到与当前点相邻的结点都被访问过为止。试求等概率选择出发点，所走的路径长度的期望值。

[BZOJ 2878] 迷失乐园

[BZOJ 2878] 迷失乐园

先考虑如何对树做这个问题。我们设$down[i]$为从结点i开始，只向孩子结点走的路径的期望长度。

[BZOJ 2878] 迷失乐园

先考虑如何对树做这个问题。我们设$down[i]$为从结点i开始，只向孩子结点走的路径的期望长度。

按题意可以得到：$down[i] = \frac{\sum_{j \in chl} dis[i][j] + down[j]}{son[i]}$。

[BZOJ 2878] 迷失乐园

先考虑如何对树做这个问题。我们设$down[i]$为从结点i开始，只向孩子结点走的路径的期望长度。

按题意可以得到：$down[i] = \frac{\sum_{j \in chl} dis[i][j] + down[j]}{son[i]}$。

同理设$up[i]$为从结点i开始，只向上走的路径的期望长度。

有两种方案：

1. 节i走到父亲f，然后再走到f的子树内部(除去i自身的子树)的其余叶子节点；总长度为：
   $down[f] \times son[f] - (down[i] + dis[i][f])$
2. 节点i走到父亲f，然后从f继续往上走，那么这种情况只有一种：$up[f]$

共有$(son[f] - 1) + 1$种情况，于是我们可以得到转移$up[i] = dis[i][f] + \frac{son[f] - down[i] - dis[i][f] + up[f]}{(son[f] - 1) + 1}$

[BZOJ 2878]迷失乐园

[BZOJ 2878]迷失乐园

每个点最后的期望就是$\frac{down[i]*son[i]+up[i]}{son[i]+1}$。

然后考虑对环的处理。环上的两个结点互相到达，既可以按顺时针方向走，也可以按逆时针方向走。因此，我们枚举这个方向，就可以拆环了。

[BZOJ 2878]迷失乐园

每个点最后的期望就是$\frac{down[i]*son[i]+up[i]}{son[i]+1}$。

然后考虑对环的处理。环上的两个结点互相到达，既可以按顺时针方向走，也可以按逆时针方向走。因此，我们枚举这个方向，就可以拆环了。

环上有两种走法，一种是顺时针，一种是逆时针。

[BZOJ 2878]迷失乐园

每个点最后的期望就是$\frac{down[i]*son[i]+up[i]}{son[i]+1}$。

然后考虑对环的处理。环上的两个结点互相到达，既可以按顺时针方向走，也可以按逆时针方向走。因此，我们枚举这个方向，就可以拆环了。

环上有两种走法，一种是顺时针，一种是逆时针。

环上的点编号为$1,2,3,4,5$，那么对于1来说，顺时针走的话，走到2的概率为1，走到3的概率为$\frac{1}{son[2]+1}$，以此类推，逆时针走的话同理。

[BZOJ 2878] 迷失乐园

[BZOJ 2878] 迷失乐园

用g表示到这个点的概率(g的初值为0.5，顺时针和逆时针两种)。

[BZOJ 2878] 迷失乐园

用g表示到这个点的概率(g的初值为0.5，顺时针和逆时针两种)。

同时我们沿着环每走到一个位置就加上从这里向外向树走的期望长度，设d为在环上已经走的路径，转移的仍是用总和/总情况。

[BZOJ 2878] 迷失乐园

用g表示到这个点的概率(g的初值为0.5，顺时针和逆时针两种)。

同时我们沿着环每走到一个位置就加上从这里向外向树走的期望长度，设d为在环上已经走的路径，转移的仍是用总和/总情况。

$up[x] = \sum_{i \in ChildX} g \times \frac{(down[cir[i]]+d) \times son[cir[i]]}{siz[cir[i]]+1}$

[BZOJ 2878] 迷失乐园

用g表示到这个点的概率(g的初值为0.5，顺时针和逆时针两种)。

同时我们沿着环每走到一个位置就加上从这里向外向树走的期望长度，设d为在环上已经走的路径，转移的仍是用总和/总情况。

$up[x] = \sum_{i \in ChildX} g \times \frac{(down[cir[i]]+d) \times son[cir[i]]}{siz[cir[i]]+1}$

注意绕一圈走到头的地方与之前的不一样，因为出发点不可能经过两次

[BZOJ 2878] 迷失乐园

用g表示到这个点的概率(g的初值为0.5，顺时针和逆时针两种)。

同时我们沿着环每走到一个位置就加上从这里向外向树走的期望长度，设d为在环上已经走的路径，转移的仍是用总和/总情况。

$$up[x] = \sum_{i \in ChildX} g \times \frac{(down[cir[i]]+d) \times son[cir[i]]}{siz[cir[i]]+1}$$

注意绕一圈走到头的地方与之前的不一样，因为出发点不可能经过两次

$$up[x] = \sum_{i \in ChildX} g \times (down[cir[i]] + d)$$

[ARC 079F]Namori Grundy

[ARC 079F]Namori Grundy

有一个弱联通的有向图，含有n个结点和n条边。试问是否存在方案，赋给每个结点一个自然数权值val_i，满足对于所有结点u，$val_u = mex\{val_v|(u,v) \in E\}$。一个集合的$mex$是没有在这个集合中出现的最小自然数。

基础练习

基础练习

链接：https://vjudge.net/contest/354626

密码：HL20200201

https://www.luogu.org/problemnew/show/P5022

结束啦

结束啦

欢迎提问

Questions are welcomed!