wjyyy
HB老年弱鸡OIer 欢迎访问www.wjyyy.top

洛谷 P2056 [ZJOI2007]捉迷藏 题解【点分治】【堆】【图论】

动态点分治入门题?

题目描述

Jiajia和Wind是一对恩爱的夫妻，并且他们有很多孩子。某天，Jiajia、Wind和孩子们决定在家里玩捉迷藏游戏。他们的家很大且构造很奇特，由 N 个屋子和 $N-1$ 条双向走廊组成，这 $N-1$ 条走廊的分布使得任意两个屋子都互相可达。

游戏是这样进行的，孩子们负责躲藏，Jiajia负责找，而Wind负责操纵这 N 个屋子的灯。在起初的时候，所有的灯都没有被打开。每一次，孩子们只会躲藏在没有开灯的房间中，但是为了增加刺激性，孩子们会要求打开某个房间的电灯或者关闭某个房间的电灯。为了评估某一次游戏的复杂性，Jiajia希望知道可能的最远的两个孩子的距离（即最远的两个关灯房间的距离）。

我们将以如下形式定义每一种操作：

- `C(hange) i` 改变第 i 个房间的照明状态，若原来打开，则关闭；若原来关闭，则打开。
- `G(ame)` 开始一次游戏，查询最远的两个关灯房间的距离。

输入输出格式

输入格式：

第一行包含一个整数 N，表示房间的个数，房间将被编号为 $1,2,3,\ldots,N$ 的整数。

接下来 $N-1$ 行每行两个整数 a,b，表示房间 a 与房间 b 之间有一条走廊相连。

接下来一行包含一个整数 Q，表示操作次数。接着 Q 行，每行一个操作，如上文所示。

输出格式：

对于每一个操作 `Game`，输出一个非负整数到 `hide.out`，表示最远的两个关灯房间的距离。若只有一个房间是关着灯的，输出0；若所有房间的灯都开着，输出-1。

输入输出样例

输入样例#1：

```
8
1 2
2 3
3 4
3 5
3 6
6 7
6 8
7
G
C 1
G
C 2
G
C 1
G
```

输出样例：

```
4
3
3
4
```

说明

对于20%的数据，$N \leq 50, M \leq 100$ ；

对于60%的数据，$N \leq 3000, M \leq 10000$ ；

对于100%的数据，$N \leq 100000, M \leq 500000$ 。

题解：

看起来动态点分治由于维护了一棵树高最多为 $\log n$ 的点分树，每次修改操作的次数是 $O(\log n)$ ，但是处理父子关系还是很难维护的。

这个题第一眼看上去（如果不带修的话）是有点分治的思路在里面的。但是每次询问的图都在改变，因此就有了动态点分治。

由于我们需要找出树上最远的两个关灯的点，点的状态是动态的。而点分治每次都是在找重心，因此把每一次的重心分层，并两两“连边”，就形成了点分树。在点分树上的儿子所管辖的点数一定小于父亲所管辖的点数的一半，所以树高是 $O(\log n)$ 。

> 注：因此下文的“分治子树”指点分树上的子树。分治重心指分治子树的根节点。

点分治的核心是在子树重心处统计过重心的路径，而重点是不能在重心处统计同一子树内的答案。

本题要我们找最长关灯点对，因此我们需要找每个点 x 作为重心时的分治子树内到当前点 x 距离，

并合并两个不同的子树中点的信息。由于只需要查询最大值，所以我们用一个堆来维护每一个分治子树中的信息。

又因为分治子树上的点到分治重心 k 的距离与当前点的距离不是线性关系（点 x 和点 k 不一定相邻，此时 $x \to k$ 路径上的点就没有方便计算的途径），所以这个信息是子树 k 内的点到点 x 的距离。记最大值为 mx_k。

然后对于 x，任意的 $\langle x,y\rangle \in$ 点分树，可以更新答案为 $\max_{\langle x,i\rangle\in\text{点分树},\langle x,j\rangle\in\text{点分树},i\neq j} mx_i + mx_j$。此时我们发现，由于 $i \neq j$，所以只需要取最大的两个 mx_k 就可以了。而这个信息也是动态的，所以应该再开一个堆。

此时我们每个点维护了两个堆

1. 大根堆 $\{q_x\}$ 维护以 x 为根的分治子树中到 fa_x（指 x 在分治子树上的父亲）的距离。
2. 大根堆 $\{q'_x\}$ 维护点分树上 x 的每个儿子 k 的 $\max\{q_k\}$。

接下来考虑如何开关灯。

我们每次只修改了一个点，并且一个点的信息只可能在它点分树上的祖先节点出现，有 $O(\log n)$ 个，我们在构造点分树的时候可以预处理出每个点 x 到它第 i 个父亲的距离，记作 $d_{i,x}$，那个父亲记为 $f_{i,x}$，特殊地，每个点的直接父亲记为 fa_x。

此时考虑每次修改点 x 对第 i 个父亲的影响，看到上面两个堆的意义，还需要分类讨论。

- 当关灯时，$u = f_{i,x}$ 子树中多了一个距离为 $d_{i,x}$ 的关灯点。需要在 $\{q_u\}$ 中插入 $d_{i+1,x}$。看是否 $d_{i+1,x}$ 成为了 $\{q_u\}$ 中最大的元素，如果是，则把 $\{q'_{fa_u}\}$ 中原来的 u 答案删掉，更新为这个答案。
- 当开灯时，u 的子树中少了一个距离为 $d_{i,x}$ 的关灯点，则需要在 $\{q_u\}$ 中删除相应的元素，如果删除了最大的，再拿此时最大的补上去。

这时需要统计答案了。发现答案是 $\max_{i=1}^{n} \max\{q'_i\}$，仍然是类似的堆操作。至此我们整道题维护了3种堆，届时输出最后一种堆的堆顶即可。

当子树内只有一个或没有关灯点的时候贡献都是 0，要输出 -1 的情况可以在外面判。一个点的时候还要存一下答案是否在最后一种堆中…因此边界情况会比较多。堆的删除是用懒惰删除法，@Dew 教了我一种神仙的结构体写法非常赞。

其他：注意 x, y 分别指父子的时候不要搞混了…

点分治+堆所以时间复杂度为 $O((n+m)\log^2 n)$。

Code:

```
#include<cstdio>
#include<cstring>
#include<queue>
using std::priority_queue;
int read()
{
    int x=0;
```

```
    char ch=getchar();
    while(ch<'0'||ch>'9')
        ch=getchar();
    while(ch>='0'&&ch<='9')
    {
        x=x*10+ch-'0';
        ch=getchar();
    }
    return x;
}
struct edge
{
    int n,nxt;
    edge(int n,int nxt)
    {
        this->n=n;
        this->nxt=nxt;
    }
    edge(){}
}e[200000];
int head[100100],ecnt=-1;
void add(int from,int to)
{
    e[++ecnt]=edge(to,head[from]);
    head[from]=ecnt;
    e[++ecnt]=edge(from,head[to]);
    head[to]=ecnt;
}
bool used[100100];
int fa[100100],f[100100],rt,tot=0;
int sz[100100];
void dfs(int x,int from)
{
    sz[x]=1;
    f[x]=0;
    for(int i=head[x];~i;i=e[i].nxt)
        if(e[i].n!=from&&!used[e[i].n])
        {
            dfs(e[i].n,x);
            sz[x]+=sz[e[i].n];
            f[x]=f[x]>sz[e[i].n]?f[x]:sz[e[i].n];
        }
    f[x]=f[x]>tot-sz[x]?f[x]:tot-sz[x];
    rt=f[x]<f[rt]?x:rt;
}

int d[18][100100],cnt[100100],dpt=1;

void Dfs(int x,int from)
{
    d[++cnt[x]][x]=dpt;
    ++dpt;
    for(int i=head[x];~i;i=e[i].nxt)
        if(e[i].n!=from&&!used[e[i].n])
        Dfs(e[i].n,x);
    --dpt;
}
```

```
void divide(int x,int from)//仅初始化块
{
    rt=0;
    tot=sz[x];
    dfs(x,x);
    used[x=rt]=1;
    fa[x]=from;
    for(int i=head[x];~i;i=e[i].nxt)
        if(!used[e[i].n])
            divide(e[i].n,x);
    for(int i=head[x];~i;i=e[i].nxt)
        if(!used[e[i].n])
            Dfs(e[i].n,x);
    used[x]=0;
}

bool col[100100];
int sum=0;

struct heap
{
    priority_queue<int> q;
    priority_queue<int> p;
    void maintain()
    {
        while(!p.empty()&&p.top()==q.top())
        {
            p.pop();
            q.pop();
        }
    }
    inline void POP(int x)
    {p.push(x);}
    inline void PUSH(int x)
    {q.push(x);}
    int TOP()
    {
        maintain();
        return q.top();
    }
    inline int sz()
    {return (q.size()-p.size());}
}q[100100],q_[100100],Q;
//q表示来源于自己子树中的
//q_表示存它爹的
int ans[100100];
bool gg[100100];
void upd(int x)
{
    if(q[x].sz()==1)
    {
        if(!gg[x])
            Q.POP(ans[x]);
        ans[x]=q[x].TOP();
        gg[x]=1;
    }
    else if(!q[x].sz())
```

```
    {
        if(!gg[x])
            Q.POP(ans[x]);
        gg[x]=0;
        ans[x]=0;
        Q.PUSH(0);
    }
    else
    {
        if(!gg[x])
            Q.POP(ans[x]);
        gg[x]=0;
        int g=q[x].TOP();
        q[x].POP(g);
        ans[x]=g+q[x].TOP();
        q[x].PUSH(g);
        Q.PUSH(ans[x]);
    }
}
void change(int x)
{
    int y=x,tmp=0;
    if(col[x])
    {
        col[x]=0;
        --sum;
        while(fa[y])
        {
            ++tmp;
            //先考虑y对fa[y]的原贡献
            if(q_[y].TOP()==d[tmp][x])
            {
                q_[y].POP(d[tmp][x]);
                //要删除一些元素了
                q[fa[y]].POP(d[tmp][x]);
                if(q_[y].sz())
                    q[fa[y]].PUSH(q_[y].TOP());
                upd(fa[y]);
            }
            else
                q_[y].POP(d[tmp][x]);
            y=fa[y];
        }
        q[x].POP(0);
        upd(x);
    }
    else
    {
        col[x]=1;
        ++sum;
        q[x].PUSH(0);
        upd(x);
        while(fa[y])
        {
            ++tmp;
            if(!q_[y].sz()||d[tmp][x]>q_[y].TOP())
            {
                if(q_[y].sz())
```

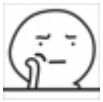
```
                    q[fa[y]].POP(q_[y].TOP());
                q[fa[y]].PUSH(d[tmp][x]);
                upd(fa[y]);
            }
            q_[y].PUSH(d[tmp][x]);
            y=fa[y];
        }
    }
}

int main()
{
    memset(head,-1,sizeof(head));

    f[0]=1e9;
    int n,u,v;
    n=read();
    for(int i=1;i<n;++i)
    {
        u=read();
        v=read();
        add(u,v);
    }
    sz[1]=n;
    divide(1,0);
    for(int i=1;i<=n;++i)
        Q.PUSH(0);
    for(int i=1;i<=n;++i)
        change(i);
    char s[100];
    int m;
    m=read();
    while(m--)
    {
        scanf("%s",s);
        if(s[0]=='G')
        {
            if(!sum)
                puts("-1");
            else if(sum==1)
                puts("0");
            else
                printf("%d\n",Q.TOP());
        }
        else
        {
            u=read();
            change(u);
        }
    }
    return 0;
}
```

分类: 解题报告 , 数据结构 , 图论

标签: 点分治 , 解题报告 , 堆 , 图论



wjyyy
关注 - 1
粉丝 - 10
+加关注

« 上一篇: 洛谷 P3244 / loj 2115 [HNOI2015] 落忆枫音 题解【拓扑排序】【组合】【逆元】
» 下一篇: 洛谷 P3285 / loj 2212 [SCOI2014] 方伯伯的 OJ 题解【平衡树】【线段树】

posted @ 2019-03-07 22:00 wjyyy 阅读(62) 评论(0) 编辑 收藏

software_orang:

厉害

刷新评论 刷新页面 返回顶部

发表评论

感谢您的回复:) 服务器端执行耗时77毫秒

编辑 预览

提交 退出 订阅评论

感谢您的回复:) 服务器端执行耗时77毫秒

[Ctrl+Enter快捷键提交]

相关博文:
· 【bzoj1095】[ZJOI2007]Hide 捉迷藏 动态点分治+堆
· [bzoj1095][ZJOI2007]Hide 捉迷藏 点分树，动态点分治
· 【BZOJ1095】[ZJOI2007]Hide 捉迷藏 动态树分治+堆
· BZOJ_1095_[ZJOI2007]Hide捉迷藏_动态点分治+堆
· 【ZJOI2007】捉迷藏
» 更多推荐...

最新 IT 新闻:
· 从《隐秘的角落》热播，看视频平台的突围之路
· 国美app上架京东自营百货，京东提供物流及售后
· B站“上车”，为车企和年轻人搭桥
· 溃败之后社区团购再成风口，可惜这注定是属于巨头的游戏
· 梁建章一人撑起了一台春晚，但携程的产品还要更努力
» 更多新闻...

公告

昵称: wjyyy
园龄: 2年1个月
粉丝: 10
关注: 1
+加关注

搜索

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

最新随笔

1.快速沃尔什变换 FWT 学习笔记【多项式】
2.HBTS(HBOI) 2019 真实退役记
3.杜教筛瞎推 学习笔记【杜教筛】【数学】
4.CF1139E Maximize Mex 题解【二分图】
5.CF1139D Steps to One 题解【莫比乌斯反演】【枚举】【DP】
6.洛谷 P2480 [SDOI2010]古代猪文 题解【欧拉定理】【CRT】【Lucas定理】
7.loj 6433 「PKUSC2018」最大前缀和 题解【DP】【枚举】【二进制】【排列组合】
8.九省联考 2018 Day 1 复现
9.CF1012C Hills 题解【DP】
10.洛谷 P4774 / loj 2721 [NOI2018] 屠龙勇士 题解【同余】【exgcd】【CRT】

我的标签

解题报告(35)
DP(11)
字符串(9)
贪心(8)
学习笔记(7)
二进制(4)
数学(4)
前缀和(4)
KMP(4)
构造(3)
更多

随笔分类

Codeforces(1)
倍增(1)
递推(1)
动态规划(11)
多项式(1)
分块(1)
构造(2)
计算几何(1)

随笔档案

友情链接

dew
wjyyy

最新评论

1. Re:Codeforces Round #545 (Div. 2) 题解

@ huangda1我比赛的时候写的 n^2 的但是第二维是没用的。比赛的时候还是求稳吧 XD...

--wjyyy

2. Re:Codeforces Round #545 (Div. 2) 题解

@ wjyyy没事没事。。 我自己当时证明了下，你的是对的。。而且我还试了一下d-i+b||i+c的情况，发现有问题。。感觉 你的方法我在比赛的时候肯定想不到...

--huangda1

3. Re:Codeforces Round #545 (Div. 2) 题解

@ huangda1这个是说 0 1 或 1 0 已经多到无法抵消了，自己这边必须承担一些，此时就不合法了。（抱歉回复晚了）...

--wjyyy

4. Re:Codeforces Round #545 (Div. 2) 题解

请问这个代码if(d-i+c>h || i+b>h) 是什么意思?

--huangda1

5. Re:NOI 2019 网络同步赛 游记

造成恐慌, 竟赛三年(x

--rvalue

阅读排行榜

1. NOIWC 2019 冬眠记【游记】(562)
2. 【DP】+【贪心】【前缀和】洛谷P2893 [USACO08FEB]修路Making the Grade 题解(433)
3. Codeforces Round #545 (Div. 2) 题解(428)
4. 【AC自动机】【字符串】【字典树】AC自动机 学习笔记(387)
5. 【数学】【筛素数】Miller-Rabin素性测试 学习笔记(375)

评论排行榜

1. Codeforces Round #545 (Div. 2) 题解(4)
2. NOIWC 2019 冬眠记【游记】(4)
3. NOI 2019 网络同步赛 游记(1)

推荐排行榜

1. NOIWC 2019 冬眠记【游记】(3)
2. 【AC自动机】【字符串】【字典树】AC自动机 学习笔记(1)
3. Codeforces Round #545 (Div. 2) 题解(1)
4. CF1012C Hills 题解【DP】(1)
5. HBTS(HBOI) 2019 真实退役记(1)