树

模 拟 讲 评 & 树 题 目 选 讲

主　讲：李天宇

日　期：2024.07.03

目录

CONTENTS

模拟讲评

- ✓ 简要介绍简要介绍

最浅二叉树（T1）

【题目描述】

二叉树的特点是树中的每个结点最多只有两个孩子结点。现在有一个二叉树，提供给你三个整数a,b,c，分别代表着有关于这一棵二叉树的三个信息：

1）该二叉树中有且仅有a个结点有两个孩子结点

2）该二叉树中有且仅有b个结点有一个孩子结点

3）该二叉树中有且仅有c个结点没有孩子结点

请找出满足这三个条件的所有二叉树中，**深度最小**的二叉树的深度为多少。本题中深度为：我们定义树中两个顶点之间的距离是它们之间**最短路径的边数**，树的高度就是在所有顶点到树根顶点的距离中取**最大距离**。

特别的，如果没有任何一个二叉树可以同时满足这三个条件，那么输出“-1”。

注意深度定义

【输入格式】

从文件T1.in中读入数据。

第一行输入一个正整数T，表示数据组数。

接下来T行，每行3个整数a,b,c，代表一棵二叉树的三个信息。

【输出格式】

输出到文件T1.out中。

共T行，每行一个整数，表示满足三个信息的二叉树的最小深度，若没有任何二叉树能够满足三个信息，则输出-1。

【样例1输入】

```
10
2 1 3
0 0 1
0 1 1
1 0 2
1 1 3
3 1 4
8 17 9
24 36 48
1 0 0
0 3 1
```

【样例1输出】

```
2
0
1
1
-1
3
6
-1
-1
3
```

测试点	T	a+b+c	特殊性质
1-3	<=100	<=20	
4-5	<=10^4	<=3*10^5	a=0
6-7	<=10^4	<=3*10^5	b=0
8-10	<=10^4	<=3*10^5	

子任务2（a=0）：

1）判断可行：

没有度为2的结点，树呈一个链状，可以有若干个度为1的结点，但只有一个叶子结点（度为0）。

2）输出最小深度：

输出b

子任务3（b=0）：

1）判断可行：

通过各种方法得到一个结论a+1=c时，可行。各种方法包括但不限于：枚举后归纳（枚举a=1时，发现c=2；a=2时，c=3；a=3时，c=4……）；理论分析（每多一个度为2的点，就需要多一个叶子结点）等等。

2）输出最小深度：

容易想到贪心。从上至下，尽可能的让每一层都填满，需要优先放置度为2的点。

100%数据：

1）判断可行：

通过子问题2/3得到的结论，利用子问题1的小规模数据验证，发现只要c=a+1,即具有可行性。或者聪明的你可能早就知道二叉树的这条性质。

2）输出最小深度：

依旧贪心。从上至下，尽可能的让每一层都填满，需要优先放置度为2的点，接着放置度为1的点，最后填充叶子结点。

```
p[0]=1;
for(int i=1;i<=20;++i){
    p[i]=p[i-1]*2;
}
t=re();
while(t--){
    a=re();b=re();c=re();
    if(a*2+b+1!=a+b+c){//树的性质
        cout<<"-1"<<"\n";
        continue;
    }
    if(a==0){
        cout<<b<<"\n";
        continue;
    }
    h=0;
    while(a>=p[h]){//放满的层
        a-=p[h];++h;
    }
    n=p[h]+a;//接下来每层都只能放这么多
    sum=a+b+c-p[h];//该行填满
    h+=sum/n+(sum%n?1:0);//还需要多少行
    cout<<h<<"\n";
}
```

考察点：树的基本性质，贪心

回顾：二叉树的其他性质。

（1）二叉树的第i层上至多有2^{i-1}个结点（i≥1）。
（2）深度为k的二叉树至多有2^k-1个结点（k≥1）。
……

拓展：若题目为k叉树，给出k+1条关于不同度结点数量的信息，如何解决？

管道运输（T2）

【题目描述】

油国有N个城市，通过N-1条管道来在各个城市之间运输油，任意两座城市之间都通过管道连通。

油国一共有Q条运输油的路线，第i条路线从城市ui运输到vi。一条运输路线会给它的起止城市以及中间途径的所有城市都带来ki的运输压力。油国需要知道每个城市的运输压力。

【输入格式】

从文件T2.in中读入数据。

第一行一个正整数n，代表油国城市个数。

第2~n行，每行两个整数ui，vi，表示一根连接ui，vi两个城市的管道。

第n+1行一个正整数Q，代表运输路线的数量。

接下来q行，每行三个整数si ti ki，表示有一条从城市si到城市ti运输压力为ki 的运输线路。

【输出格式】

输出到文件T2.out中。

输出共n行，每行一个整数，按照编号从1~n的顺序依次输出每个城市的运输压力。

测试点	n<=	q<=	k<=	特殊性质
1	20	100	10^5	
2-3	3000	3000	10^9	
4-5	$2*10^4$	$2*10^5$	10^9	第i条管道形如(i,i+1)
6-7	$2*10^5$	$2*10^5$	10^9	保证有一个城市是所有运输线路的起点或者终点
8-10	$2*10^5$	$2*10^5$	10^9	

【样例1输入】

```
5
3 4
1 5
4 2
5 4
10
5 4 1
5 4 1
3 5 1
4 3 1
4 3 1
1 3 1
3 5 1
5 4 1
1 5 1
```

【样例1输出】

```
2
0
6
9
7
```

子任务1

暴力模拟

时间复杂

权，

```
void bfs(){
    queue<int> q;
    q.push(s);vis[s]=true;
    while(!q.empty()){
        int xx=q.front();q.pop();
        if(xx==t)   break;
        for(int i=head[xx];i;i=nxt[i]){
            int yy=ver[i];
            if(vis[yy]) continue;
            q.push(yy);vis[yy]=true;pre[yy]=xx;
        }
    }
    int cur=t;
    while(cur!=s){
        p[cur]+=tw;
        cur=pre[cur];
    }
    p[s]+=tw;
    return;
}
```

注意链的各种形容：排列...ui,vi...只有两条边连接...

子任务2（所有点构成一条链）：

树上问题变为区间问题。每次选取[u,v]，使区间内每个点权+k。

区间修改单点查询：线段树，时间复杂度O((q+n) log n)

更简单的：区间差分

数组	1	4	6	3	10	2
差分	1	3	2	-3	7	-8
差分的前缀和	1	4	6	3	10	2
前缀和	1	5	11	14	24	26
前缀和的差分	1	4	6	3	10	2

子任务2（所有

```
for(int i=1;i<n;++i){
    x=re();y=re();
    adde(x,y);
    adde(y,x);
    if(min(x,y)!=i || max(x,y)!=i+1)
        subtask2=false;
}
k=re();
if(subtask2){
    for(int i=1;i<=k;++i){
        s=re();t=re();tw=re();
        d[min(s,t)]+=tw;
        d[max(s,t)+1]-=tw;
    }
    for(int i=1;i<=n;++i){
        p[i]=p[i-1]+d[i];
    }
}
```

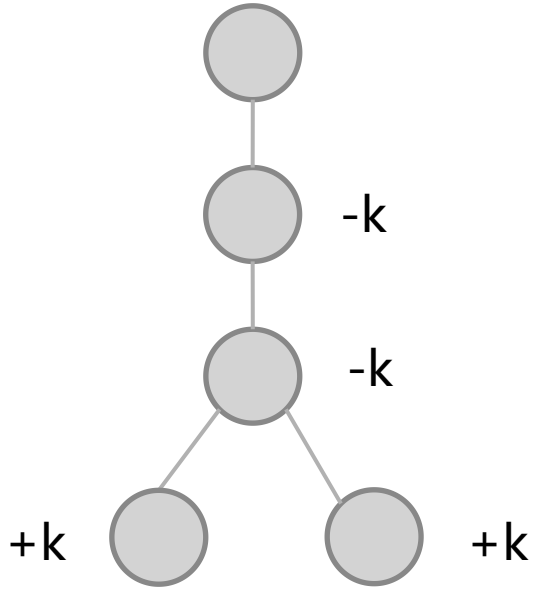
子任务3（保证有一个城市是所有运输线路的起点或者终点）：

以该城市作为根节点，则变为若干个区间问题，每一个叶子结点到根节点的路径都构成一个区间。（稍微复杂一点：两个城市具备祖先-后代关系）

简化版的树上差分：在另一个结点处标记差分值，每一个结点的点权由所有路径的差分求和得到（dfs遍历一遍子树）。

100%数据：

树上差分



点差分：
路径上的所有点权加x
两点u,v之间的路径即u先到lca(u,v)，然后再从lca(u,v)到v
令u,v点权+k，lca(u,v)点权-k（否则会重复计算），lca(u,v)的父亲要再-k。

每个点的点权通过子树内的差分和得到

```
void dfs(int x){
    vis[x]=true;
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(vis[y])
            continue;
        fa[y][0]=x;
        for(int p2=1;p2<=20;++p2){
            fa[y][p2]=fa[fa[y][p2-1]][p2-1];
        }
        d[y]=d[x]+1;
        dfs(y);
    }
    return ;
}
```

```
int lca(int x,int y){
    if(d[x]<d[y])
        swap(x,y);
    for(int i=20;i>=0;--i){
        if(d[x]-(1<<i)>=d[y])
            x=fa[x][i];
    }
    if(x==y)
        return y;
    for(int i=20;i>=0;--i){
        if(fa[x][i]!=fa[y][i]){
            x=fa[x][i];
            y=fa[y][i];
        }
    }
    return fa[x][0];
}
```

```
void dfs2(int x,int xfa){
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(y==xfa)  continue;
        dfs2(y,x);
        p[x]+=p[y];
    }
    return ;
}
```

```
d[1]=1;
dfs(1);
k=re();
for(int i=1;i<=k;++i){
    s=re();t=re();tw=re();
    int l=lca(s,t);
    p[s]+=tw;p[t]+=tw;p[l]-=tw;p[fa[l][0]]-=tw;
}
dfs2(1,0);
```

考察点：树上倍增，树上差分，最近公共祖先

一、树上倍增法求LCA：

预处理出x节点向上走2^k步到达的节点，存在F(x,k)中。

①设d[x]表示x的深度，不妨令d[x]>=d[y]

②把x向上调整到与y同一深度（通过倍增调整）

③若此时x=y，则LCA(x,y)=y

④否则继续向上调整，从大到小枚举步长2^i,保持调整的大小一致，并且两者不相会

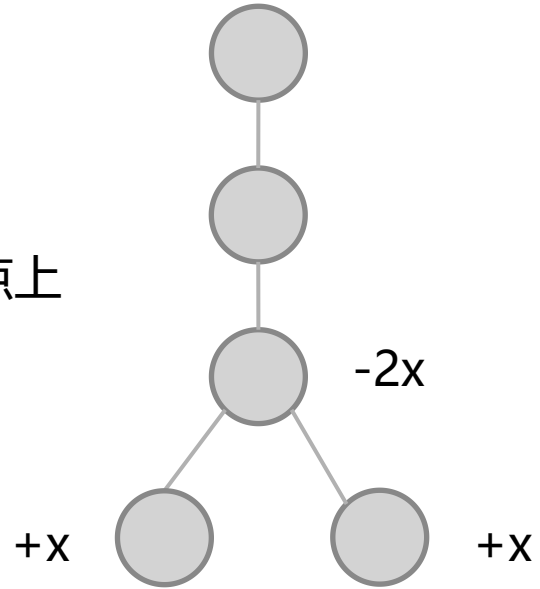
⑤若x,y只差一步就相会了，它们的父亲节点F(x,0)就是LCA

二、树上差分：边差分，路径上的所有边权加x

考察点：树上倍增，树上差分，最近公共祖先

二、树上差分：

边差分：
路径上的所有边权加x
将边权转化为点权->依附于深度更深的点上



松鼠家的灯（T3）

【题目描述】

松鼠的家非常大，一共有n个房间。这n个房间的构成可以看作是一棵树，其中1号房间是树根。每个房间内都有一盏灯，每盏灯最初的时候可能亮着，也可能灭着。每个房间内也有一个按钮，按下按钮后，以该房间为子树根的子树内的所有房间的灯的状态都会发生变化，即原本亮着的会灭掉，原本灭着的会亮起来。松鼠会依次进行q次操作，操作分为两种：

1）get x，询问以x房间作为子树根的子树内一共有多少盏灯是亮着的

2）pow x，按下x房间的按钮

你需要输出所有get操作的答案。

【输入格式】

从文件 T3.in 中读入数据。

第一行一个整数n，代表房间总数量。

第二行共n-1个整数，第i个数xi表示在树中xi房间是i+1房间的父亲结点

第三行给出每个房间灯的初始状态，1代表开着，0代表灭着。

第四行一个整数q，接下来q行，每行一次操作。

【输出格式】

按照顺序输出所有get询问操作的答案，一行一个。

测试点编号	n<=	q<=	特殊性质
1	50	50	
2	500	500	
3-4	3000	3000	
5-8	$2*10^5$	$2*10^5$	所有的get都在所有的pow之后
9-12	$2*10^4$	$2*10^5$	$X_i=i$
13-20	$2*10^5$	$2*10^5$	

【样例1输入】

```
4
1 1 1
1 0 0 1
9
get 1
get 2
get 3
get 4
pow 1
get 1
get 2
get 3
get 4
```

【样例1输出】

```
2
0
0
1
2
1
1
0
```

子任务1（n,q

暴力模拟，对

```
void dfs(int x,int fa){
    dep[x]=dep[fa]+1;
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(y!=fa)   dfs(y,x);
    }
    return ;
}
void dfs2(int x,int v){
    tt[x]=a[x]=a[x]^v;
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(dep[y]>dep[x])   dfs2(y,v);
        tt[x]+=tt[y];
    }
    return ;
}
```

```
void sol3(){
    bool firstget=true;
    for(int i=1;i<=m;++i){
        cin>>str;u=re();
        if(str=="get"){
            if(firstget){
                firstget=false;dfs
            }
            cout<<tt[u]<<"\n";
        }
        else{
            d[u]^=1;
        }
    }
    return;
}
```

```
void dfs3(int x,int fa){
    a[x]^=d[x];
    tt[x]=a[x];
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(y==fa)   continue;
        d[y]^=d[x];
        dfs3(y,x);
        tt[x]+=tt[y];
    }
    return ;
}
```

子任务3　(所有结点呈现一条链状):

区间修改，区间查询=>线段树

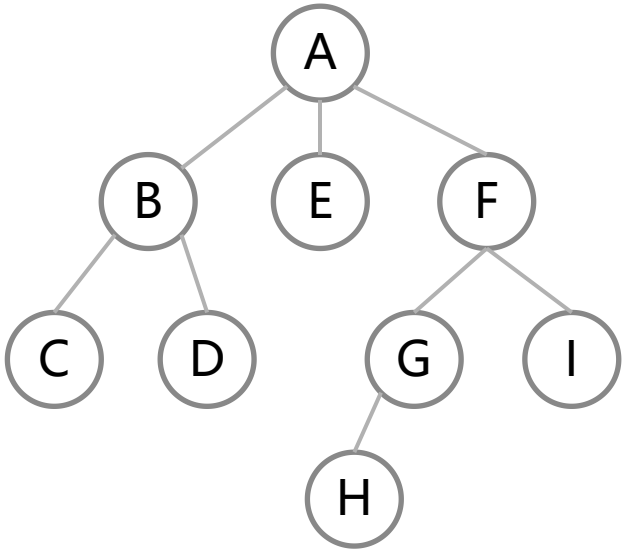
100%数据：

DFS序+线段树。

通过子任务3，发现如果能够把一棵子树内的结点修改与查询看作是一段区间内的修改与查询，则可以用线段树解决。

DFS序：按照深搜的顺序对一棵树上的结点进行编号，把树形结构映射成线性结构，然后用树状数组、线段树这样的数据结构维护区间信息。

欧拉序是在DFS序的基础上进行的拓展，加入回溯等信息。有两种形式：
1）第一次到达结点记录一次，每访问完该结点的一棵子树就再记录一次，共2n-1个编号。
2）每个结点入栈和出栈分别记录一次，一共2n个编号。



DFS序: A-B-C-D-E-F-G-H-I

欧拉序1：A-B-C-B-D-B-A-E-A-F-G-H-G-F-I-F-A

欧拉序2：A-B-C-C-D-D-B-E-E-F-G-H-H-G-I-I-F-A

```
void dfs(int x){//确定dfs序
    st[x]=
    for(in
        in
        if
        df
    }
    ed[x]=
    return
}
```

```
void build(int x,int l,int r){
    if(
}
int
bui
bui
pus
ret
}
```

```
void update(int x,int tl,int tr,int ul,int ur){
    if(ul<=tl&&ur>=tr){
        t[x].ltag^=1;
        t[x].sum=t[x].cnt-t[x].sum;
        return ;
    }
    int mid=(tl+tr)>>1;
    pushdown(x);
    if(ul<=mid) update(lc(x),tl,mid,ul,ur);
    if(ur>mid)  update(rc(x),mid+1,tr,ul,ur);
    pushup(x);
    return ;
}
```

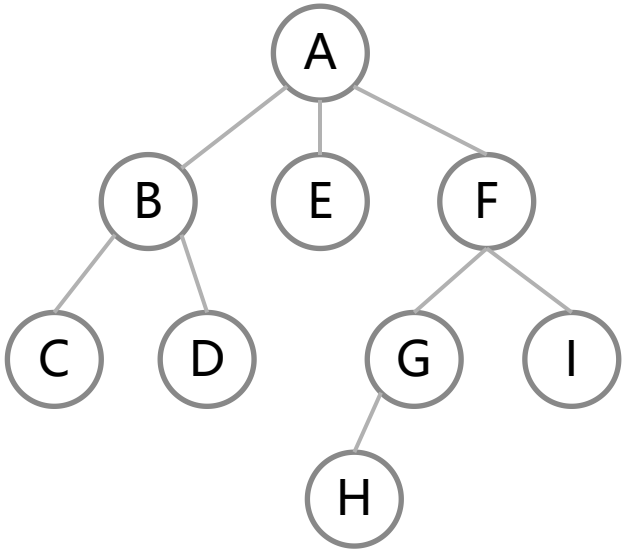
之间（
操作。
填充位）

考察点：DFS序

拓展：若T2也是分为两种操作，对路径上的所有点加权值，查询某个点的权值，如何解决?

1）对子树内所有点操作->[start(x),end(x)]，只操作其中的start()
2）对单个点操作->[start(x),start(x)]
3）对某段路径(u,v)操作->依然分为两段，以u到lca(u,v)为例，操作[start(lca(u,v)) , start(u)]中的所有点，对end()取start()的相反操作。若在深搜过程中，lca(u,v)先访问了其他子树，然后才到u，则其他子树的start()与end()都会出现在区间内，从而相互抵消，而路径上的点只会出现start()，不会出现end()。

[树上操作]



欧拉序2：A-B-C-C-D-D-B-E-E-F-G-H-H-G-I-I-F-A

例如现在要修改B~G路径上的值，分为B~A,A~G，单看A~G对应区间为[A,G]，不难发现A~G路径上的点A、F、G都只有start点(标红），其他的B、C、D、E则不然。

大雪封路（T4）

【题目描述】

雪国有N个城市，通过N-1条公路相互连通，即可以将雪国看作是一棵N个结点N-1条边的树。每条公路都有自己的长度。两个城市之间的距离dist(u,v)定义为：从u到v的简单路径边权和。特别的，我们规定dist(u,u)=0。

雪国目前正在遭受暴雪。现在有q个时刻，每个时刻过后，都会有一条公路因为暴雪的原因无法通行。显然每次有公路无法通行，都会使得城市之间的连通状况发生改变，仍能够相互连通的城市看作是一个连通块。你需要求出每个时刻后每个连通块内距离最远两个点的距离的最大值。形式化的，每次操作后，你要求出

$$\max_{c \in C} \{ \max_{u,v \in c} dist(u,v) \}$$

其中，C为当前所有的连通块构成的集合。

【输入格式】

从文件 T4.in 中读取数据。第一行一个整数T，表示数据组数，对每组数据：
第一行两个整数n,q，依次表示雪国的城市数和出现大雪封路的时刻数量。
接下来n-1行，每行三个整数u,v,w，表示上有一条连接城市u和城市v的长度为w的公路。
接下来 q行，每行一个整数ei，表示该时刻后，第ei条公路被大雪封住无法通行，保证每组数据内，ei不会重复出现。

【输出格式】

输出到文件 T4.out 中。
对每组数据输出q行，每行一个整数，依次表示每个时刻后所求的答案。

测试点编号	T<=	每组q<n<=	特殊性质
1-2	100	100	
3-4	10	2000	
5-8	10	10^4	所有城市构成一条链
9-12	5	$3*10^4$	
13-20	5		

所有的w<=10^5。对测试点13-20，保证$\sum n$<=$3*10^5$。

【样例1输入】

```
2
4 2
1 2 1
2 3 2
3 4 3
2
3
12 2
1 2 1
2 3 1
1 4 3
2 5 4
5 6 3
5 7 2
7 8 1
8 9 1
9 10 1
7 11 5
8 12 3
4
6
```

【样例1输出】

```
3
1
10
9
```

```
void sol1(){
    for(int i=1;i<=q;++i){
        curans=0;
        e[i]=re();
        cut[e[i]*2]=cut[e[i]*2+1]=true;//要断开
        for(int j=1;j<=n;++j){
            vis[j]=false;
        }
        for(int j=1;j<=n;++j){
            if(vis[j])  continue;
            sol1x=j;
            dis[j]=0;
            dfs(j,0);
            dis[sol1x]=0;
            dfs(sol1x,0);
        }
        printf("%lld\n",curans);
    }
    return ;
}
```

```
void dfs(int x,int fa){
    vis[x]=true;
    for(int i=head[x];i;i=nxt[i]){
        if(cut[i])  continue;
        int y=ver[i];
        if(y==fa)   continue;
        dis[y]=dis[x]+w[i];
        dfs(y,x);
    }
    if(dis[x]>dis[sol1x]){
        sol1x=x;curans=max(curans,dis[x]);
    }
    return;
}
```

```
void sol2(){
    vec.push_back(1);
 for(int i=1;i<=q;++i){
     e[i]=re();
     int l=f[ver[e[i]*2]],r=f[ver[e[i]*2+1]];
     if(l>r) swap(l,r);//定区间，相差1
     vector<int>::iterator itl,itr=lower_bound(vec.begin(),vec.end(),r);
     itl=itr;--itl;
     ms.erase(ms.find(dis[(*itr)-1]-dis[*itl]));

     ms.insert(dis[(*itr)-1]-dis[r]);
     ms.insert(dis[l]-dis[*itl]);
     multiset<long long>::iterator msit=ms.end();
     vec.insert(itr,r);
     printf("%lld\n",*(--msit));
 }
 return ;
    }
    ms.insert(dis[n]);
```

子任务3 ：

或许可以延用子任务2的思路，每次边断开后，不需要扫描所有连通块内的直径，只需要对当前边连接的两侧的连通块重新扫描直径并记录即可。每次对连通块求直径都是O(k)的，k为连通块大小。
极端情况下，连通块每次都分为一个极小的连通块和一个极大的连通块，此时时间复杂度过高；随机情况下，连通块大小分配会比较均匀，时间复杂度可能能通过。

总时间复杂度$O(Tn^2 \sim Tnlogn)$

100%数据 ：

一个比较套路的思路：正难则反，例：星球大战

按照题目描述去做，比较烦的地方：每次都要重新求得一个连通块内的直径，哪怕它也许并没有变，但我们并不确定。
由于没有强制在线，我们可以“时光倒流”，先看作大雪封住了若干条道路，然后随着逆向的时间，这些道路一一解封。解封一条道路，意味着两个连通块合并在一起，就可以用并查集来操作,然后计算新的连通块直径。这样的优点在哪呢?

新的连通块的直径只有两种可能：经过修复的边或者不经过修复的边。若不经过，则一定为原连通块中的某条直径；若经过，则求出两个端点分别到原连通块内的最长距离，拼在一起即为新的直径。

100%数据 ：

直径的性质：从树上任意一点出发的最长简单路径的另一个端点，一定是直径的两个端点之一。

因此找最长距离的时候不需要搜索一遍连通块，只需要求得该点到直径两个端点的距离即可。如何快速求到直径两个端点的距离？

路径：利用LCA+倍增来求，时间复杂度O(log n)

考察点：并查集，树的直径，树上倍增，LCA

树的直径：
1）两遍DFS法，从一点出发找最远点，最远点一定为直径端点之一，再从直径端点之一出发，找另一个最远点。
2）树形DP法。

考察点：并查集，树的直径，树上倍增，LCA

树的直径的性质：
（1）对树上的任意一点而言，树上与它距离最远的点一定为树的直径的两个端点的其中之一。
（2）直径的两端点一定是两个叶子结点。
（3）对于两棵树，如果第一棵树的直径两端点为（u，v），另一棵树的直径为（x，y），用一条边将两棵树连接，那么新树的直径的两端点一定是u，v，x，y中的两点。
（4）对于一棵树，如果在一个点上接一个叶子结点，那么最多会改变直径的一个端点。
（5）若一棵树存在多条直径，那么这些直径交于一段，若交于一点，则该点为这些直径的中点。

考察点：并查集，树的直径，树上倍增，LCA

树上倍增：

可以存向上走2^k的结点编号、距离和、最大/最小权值……

题目选讲

- ✓ 简要介绍简要介绍

数据结构：并查集、树的孩子兄弟表示法、二叉堆、树状数组、线段树、字典树、笛卡尔树、平衡树（AVL、treap、splay）

算法：树的重心、树的直径、DFS序与欧拉序、树上差分、子树、树上倍增、最近公共祖先

树形DP

种树【CSPS2023T4】

【题目描述】

你是一个森林养护员，有一天，你接到了一个任务：在一片森林内的地块上种树，并养护至树木长到指定的高度。森林的地图有 n 片地块，其中 1 号地块连接森林的入口。共有 n−1 条道路连接这些地块，使得每片地块都能通过道路互相到达。最开始，每片地块上都没有树木。

你的目标是：在每片地块上均种植一棵树木，并使得 i 号地块上的树的高度生长到不低于ai米。你每天可以选择一个未种树且与某个已种树的地块直接邻接（即通过单条道路相连）的地块，种一棵高度为 0 米的树。如果所有地块均已种过树，则你当天不进行任何操作。特别地，第 1 天你只能在 1 号空地种树。

对每个地块而言，从该地块被种下树的当天开始，该地块上的树每天都会生长一定的高度。由于气候和土壤条件不同，在第 x 天，i 号地块上的树会长高 $\max\ (bi + x * ci,\ 1)$ 米。注意这里的 x 是从整个任务的第一天，而非种下这棵树的第一天开始计算。

你想知道：最少需要多少天能够完成你的任务？

【输入格式】

第一行包含一个正整数n，表示森林的地块数量。
接下来n行，每行三个整数ai,bi,ci，分别描述一片地块，含义如题目中所描述。
接下来n-1行，每行两个正整数ui，vi，表示一条连接地块ui和vi的道路。

【输出格式】

一个正整数，表示完成任务所需的最少天数。

【样例1输入】

```
4
12 1 1
2 4 -1
10 3 0
7 10 -2
1 2
1 3
3 4
```

【样例1输出】

```
5
```

测试点编号	$n \leq$	特殊性质
1	20	A
2 ~ 4		无
5 ~ 6	500	A
7 ~ 8	10^5	
9 ~ 10		B
11 ~ 13		C
14 ~ 16		D
17 ~ 20		无

特殊性质A：ci=0
特殊性质B：ui=i,vi=i+1
特殊性质C：与任何地块直接相连的道路不超过2条
特殊性质D：ui=1

【解析】

子问题B：按照1~n顺序的链（10pts）

种树只能按照1、2、3……n这样的顺序种，因此可以求得每棵树最少多久能长成目标高度（涉及数学运算），取max。

【解析】
子问题D：菊花图（15pts）

种完根节点后，其他结点都可以种植。但讨论种植的先后顺序非常困难。

一个重要的思想步骤：答案具有单调性，若最早能在t天能完成任务，则t往后的时间都可以，t之前都不可以。因此二分结束时间答案，同时确定了结束时间后，每个结点最晚种植时间也能确定，那就可以按照最晚种植时间排序后进行分配。

设f[t]为在t时间及之前就必须要种植的数量，只要f[t]<=t，则二分的答案可以保留。

【解析】
子问题C：链（15pts）

根结点两端的点可以在第2个时刻种下，相距2的点最早可以在第3个时刻种下……因此不仅要考虑由于树的高度需求计算出的最晚时间，还要考虑到这个结点按照顺序能够种植的最早时间；同时若一个结点距离根节点距离为s，最晚种植时间是t，那么同一侧距离根节点距离为s-1的结点，最晚t-1就要种植……

【解析】

将C、D结论拓展到一棵正常的树，同样需要考虑子问题C的两件事情，最后再利用子问题D的方法判断能否满足。

树的重心【CSPS2019】

【题目描述】

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~ 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：

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

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。

【输入格式】

第一行一个整数T表示数据组数。

接下来依次给出每组输入数据，对于每组数据：

第一行一个整数n表示树S的大小。

接下来 n−1 行，每行两个以空格分隔的整数ui,vi，表示树中的一条边(ui,vi)

【输出格式】

共 T 行，每行一个整数，第 i 行的整数表示：第 i 组数据给出的树单独删去每条边后，分裂出的两个子树的重心编号和之和。

测试点编号	$n=$	特殊性质
$1\sim2$	7	无
$3\sim5$	199	无
$6\sim8$	1999	无
$9\sim11$	49991	A
$12\sim15$	262143	B
16	99995	无
$17\sim18$	199995	无
$19\sim20$	299995	无

A：树是一条链

B：树是完美二叉树，即$1\le i\le (n-1)/2$，存在两条边$(p_i,p_{2i}),(p_i,p_{2i+1})$

【样例1输入】

```
2
5
1 2
2 3
2 4
3 5
7
1 2
1 3
1 4
3 5
3 6
6 7
```

【样例1输出】

```
32
56
```

【解析】

子问题小数据(40pts):

暴力切断每条边后，求各自的重心，每次求重心O(n)，总时间复杂度O(n^2)

```
void sol1(){
    for(int i=1;i<n;++i){
        for(int j=1;j<=n;++j){
            f[j]=0;//子树大小
        }
        cut[i*2]=cut[i*2+1]=1;
        sum=0;//结点总个数
        dfs(ver[i*2],0);dfs2(ver[i*2],0);//统计+dp
        sum=0;
        dfs(ver[i*2+1],0);dfs2(ver[i*2+1],0);
        cut[i*2]=cut[i*2+1]=0;
    }
    cout<<ans<<"\n";
    return;
}
```

```
void dfs(int x,int fa){
    ++sum;
    for(int i=head[x];i;i=nxt[i]){
        if(cut[i])  continue;
        int y=ver[i];
        if(y==fa)   continue;
        dfs(y,x);
    }
    return;
}
void dfs2(int x,int fa){
    f[x]=1;
    int maxc=0;
    for(int i=head[x];i;i=nxt[i]){
        if(cut[i])  continue;
        int y=ver[i];
        if(y==fa)   continue;
        dfs2(y,x);
        maxc=max(maxc,f[y]);
        f[x]+=f[y];
    }
    maxc=max(maxc,sum-f[x]);
    if(maxc<=sum/2){
        ans+=x;
    }
    return;
}
```

【解析】
子问题C(15pts):链

从一端向另一端尝试切边，两棵树的重心都会向另一端偏移。
假设一条链上的点为1-n：
1号点只有在切断(1,2),(2,3)时，可能做重心
2号点只有在切断(2,3),(3,4),(4,5)时，可能做重心
3号点只有在切断(4,5),(5,6),(6,7)时，可能做重心
……
对称的n号点,n-1号点
特别的，mid点只会在切断(1,2),(n-1,n)时，可能做重心
于是得出数学公式，只要找到端点&mid点即可计算，时间复杂度O(n)

【解析】

子问题C(15pts):链

```
void sol2(){
    ans=((long long)n+1)*n/2*3;
    int id=0,pre=0;
    for(int i=1;i<=n;++i){
        if(du[i]==1)
            id=i,ans-=i;
    }
    for(int j=1;j<=n/2;++j){
        for(int i=head[id];i;i=nxt[i]){
            int y=ver[i];
            if(y==pre)  continue;
            pre=id;id=y;break;
        }
    }
    ans-=id;
    cout<<ans<<"\n";
    return ;
}
```

【解析】
子问题B(20pts):完全二叉树

262143=2^{18}-1，即有18层，根据对称性，每层的结点做重心的次数相同。
我们把每个边归在深度更深的结点上，则每个结点在相应的边切断后，会做一次重心。问题在于剩下的一个重心。
另一个重心一定会向根节点另一个孩子方向偏移，并且只会偏移到根节点的另一个孩子，为什么？

另一个子树的根假设为x，由于x的子树未被破坏，因此x的两个子树都没有超过结点的一半；同时由于删去了至少一个结点，x与根节点相连的子树，也不会超过结点的一半。可以确定删1个结点时，根为原根root与x，除此之外，根均只能为x。
找到原二叉树的根，以及他的左右孩子，则通过数学公式可以计算得出。时间复杂度O(n)。

```
void sol3(){//2**18-1，即有18层，每层的结点做重心的次数相同
    //根节点做2**17次，左右孩子2**17次，其他结点1次
    sum=n;
    dfs2(1,0);
    int root=ans;
    ans=(long long)root*((1<<17)-1);
    for(int i=head[root];i;i=nxt[i]){
        int y=ver[i];
        ans+=(long long)y*((1<<17)-1);
    }
    ans+=(long long)n*(1+n)/2;
    cout<<ans<<"\n";
    return;
}
```

考场上扔了T2来打这题的部分分，然后没看到数据范围是等号，不知道怎么判完全二叉树然后40分滚粗.....

【解析】

两个子问题对我们的启示？如何将O(n^2)的算法转变成O(n)的？

通过运算、分析，找到每个结点能做根节点的次数，避免每次断边找根的过程。

首先找到原始重心root，①对于点x≠root，若x成为割掉某条边后的重心，则这条边一定不在x的子树内。设割掉一条边后，另外一棵树的大小为S，$g_x = \max_{y \in son(x)} \{s_y\}$，即x的重儿子。由于x为重心，则根据重心性质：

$$2 * (n - S - s_x) \leq n - S$$

$$2 * g_x \leq n - S$$

即我们要找到对于一个x，多少条割掉的边能满足：

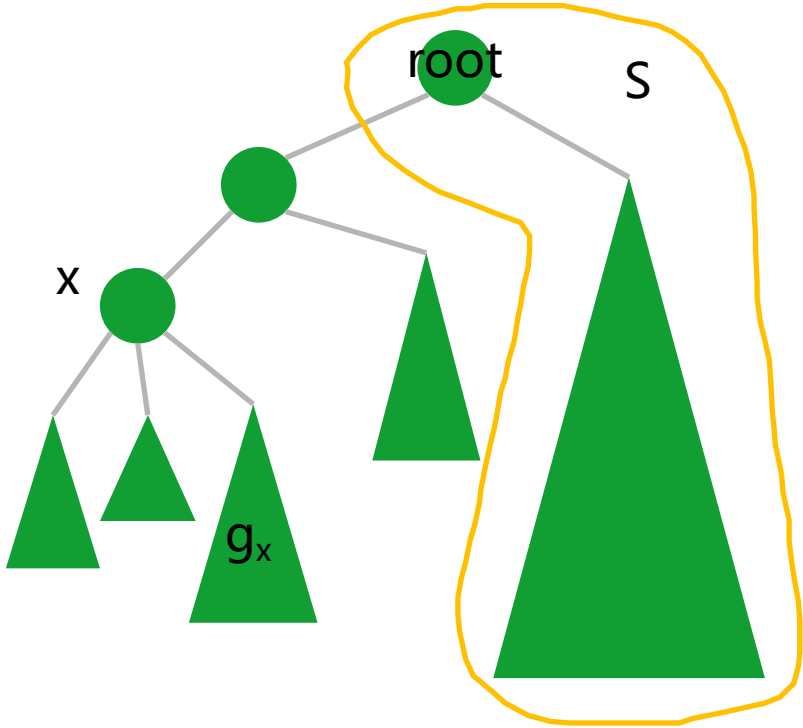
1、$n - 2s_x \leq S \leq n - 2g_x$

2、边不在x的子树内

1、$n-2s_x \le S \le n-2g_x$
2、边不在x的子树内

如果只有条件1，可以在dfs时，拿一个树状数组动态维护割掉当前点和其父亲的边之后S的值有多少个，每个点的询问就是一个区间求和。如何加上条件2？

再拿一个树状数组按照dfs序动态维护经过的所有点的S的值有多少个，对于一个点，在子树内可以割掉的边就是回溯与进入时区间求和后的差。（根到x的路径）



```
void dfs4(int x,int fa){
    update(n-f[x],1,t);
    update(f[fa],-1,t);
    if(x!=root){
        ans+=(long long)x*(query(n-2*g[x],t)-query(n-2*f[x]-1,t));
        ans+=(long long)x*(query(n-2*g[x],tt)-query(n-2*f[x]-1,tt));
        if(inu[fa]) inu[x]=true;
        if(inu[x]&&2*f[v]<=n-f[x])  ans+=root;
        if(!inu[x]&&2*f[u]<=n-f[x]) ans+=root;
    }
    update(f[x],1,tt);
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(y==fa)   continue;
        dfs4(y,x);
    }
    if(x!=root){
        ans-=(long long)x*(query(n-2*g[x],tt)-query(n-2*f[x]-1,tt));//去掉子树内的
    }
    update(n-f[x],-1,t);
    update(f[fa],1,t);
    return;
}
```

方便理解树状数组作用的链接

【解析②换根倍增法】

赛道修建【NOIP2018】

【题目描述】

C 城将要举办一系列的赛车比赛。在比赛前，需要在城内修建m条赛道。

C 城一共有n个路口，这些路口编号为1~n，有n-1条适合于修建赛道的双向通行的道路，每条道路连接着两个路口。其中，第i条道路连接的两个路口编号为ai和bi，该道路的长度为li 。借助这n-1条道路，从任何一个路口出发都能到达其他所有的路口。

一条赛道是一组互不相同的道路e1,e2....ek，满足可以从某个路口出发，依次经过道路e1,e2....ek（每条道路经过一次，不允许调头）到达另一个路口。一条赛道的长度等于经过的各道路的长度之和。为保证安全，要求每条道路至多被一条赛道经过。

目前赛道修建的方案尚未确定。你的任务是设计一种赛道修建的方案，使得修建的m条赛道中长度最小的赛道长度最大（即m条赛道中最短赛道的长度尽可能大）

【输入格式】

输入文件第一行包含两个由空格分隔的正整数n,m，分别表示路口数及需要修建的 赛道数。接下来 n-1行，第i 行包含三个正整数ai,bi,li，表示第i 条适合于修建赛道的道 路连接的两个路口编号及道路长度。保证任意两个路口均可通过这n-1条道路相互到达。每行中相邻两数之间均由一个空格分隔。

【输出格式】

一个数，表示最小赛道长度的最大值。

【样例1输入】

```
9 3
1 2 6
2 3 3
3 4 5
4 5 10
6 2 4
7 2 9
8 4 7
9 4 4
```

【样例1输出】

```
15
```

测试点编号	n	m	$a_i = 1$	$b_i = a_i + 1$	分支不超过 3
1	≤ 5	$= 1$	否	否	是
2	≤ 10	$\leq n-1$	否	是	是
3	≤ 15	$\leq n-1$	是	否	否
4	$\leq 10^3$	$= 1$	否	否	是
5	$\leq 3 \times 10^4$	$= 1$	是	否	否
6	$\leq 3 \times 10^4$	$= 1$	否	否	否
7	$\leq 3 \times 10^4$	$\leq n-1$	是	否	否
8	$\leq 5 \times 10^4$	$\leq n-1$	是	否	否
9	$\leq 10^3$	$\leq n-1$	否	是	是
10	$\leq 3 \times 10^4$	$\leq n-1$	否	是	是
11	$\leq 5 \times 10^4$	$\leq n-1$	否	是	是
12	≤ 50	$\leq n-1$	否	否	是
13	≤ 50	$\leq n-1$	否	否	是
14	≤ 200	$\leq n-1$	否	否	是
15	≤ 200	$\leq n-1$	否	否	是
16	$\leq 10^3$	$\leq n-1$	否	否	是
17	$\leq 10^3$	$\leq n-1$	否	否	否
18	$\leq 3 \times 10^4$	$\leq n-1$	否	否	否
19	$\leq 3 \times 10^4$	$\leq n-1$	否	否	否
20	$\leq 5 \times 10^4$	$\leq n-1$	否	否	否

【解析】
子问题1：m=1（20pt）

即找到一条最长的路径->树的直径

【解析】
子问题2：ai=1（20pt）

菊花图，路径最多经过两条边。
先排序,若2m>n,最大的几条独成一条赛道,直到2m<=n后,最大的配最小。若2m<=n，取前2m个最大配最小。

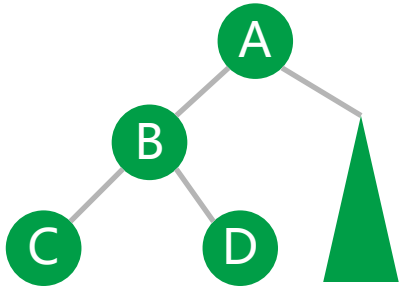
【解析】
子问题3：bi=ai+1（20pt）

线性，二分路径长度mid，从起点开始贪心的选择，直到区间长度达标后断开，以下一个点作为起点继续找下一个区间。

【解析】

综上，已经可以拿到55pt。考虑子问题4：分支不超过3，即二叉树。

一个显而易见的想法，首先依然二分答案，问题：如何验证二分的答案？



情况1：(B,C),(B,D)两条边都满足，则无需再考虑

情况2：(B,C),(B,D)其中一条边满足，则剩下一条边考虑和(A,B)结合。(A,B)如果带上它满足，可以带上它；(A,B)带上它仍不满足，则再加上一段，总之不是在B这里要考虑的事情了。

情况3：(B,C),(B,D)两条边都不满足，则此时一定需要结合，首先考虑两者结合，因为如果先考虑和(A,B)结合，(B,C),(B,D)一定有一条边就用不到了，这不是一个最优的结合方法。两者结合不了，再取较长的一条边向上结合。

【解析】
子问题4的
实在没有办

部结合，

```
void dfs(int x,int fa,int mid){
    multiset<int> mst;
    f[x]=0;
    for(int i=head[x];i;i=nxt[i]){
        int y=ver[i];
        if(y==fa)   continue;
        dfs(y,x,mid);
        if(f[y]+w[i]>=mid)  ++cnt;
        else    mst.insert(f[y]+w[i]);
    }
    while(!mst.empty()){
        multiset<int>::iterator it=mst.begin();
        mst.erase(it);
        multiset<int>::iterator it1=mst.lower_bound(mid-*it);
        if(it1==mst.end()){
            f[x]=max(f[x],*it);
        }
        else{
            ++cnt;
            mst.erase(it1);
        }
    }
    return;
}
```

Promotion Counting P

【题目描述】

奶牛们又一次试图创建一家创业公司，还是没有从过去的经验中吸取教训——牛是可怕的管理者！

为了方便，把奶牛从 1~ n 编号，把公司组织成一棵树，1 号奶牛作为总裁（这棵树的根节点）。除了总裁以外的每头奶牛都有一个单独的上司（它在树上的“双亲结点”）。

所有的第 i 头牛都有一个不同的能力指数 pi，描述了她对其工作的擅长程度。如果奶牛 i 是奶牛 j 的祖先节点，那么我们我们把奶牛 j 叫做 i 的下属。

不幸地是，奶牛们发现经常发生一个上司比她的一些下属能力低的情况，在这种情况下，上司应当考虑晋升她的一些下属。你的任务是帮助奶牛弄清楚这是什么时候发生的。简而言之，对于公司的中的每一头奶牛 i，请计算其下属 j 的数量满足 pj>pi

【输入格式】
第一行n
接下来n行包括奶牛们的能力指数p1,p2...pn，保证互不相同
接下来n-1行描述了奶牛2~n的上司的编号

【输出格式】
输出n行。第i行给出有多少奶牛i的下属比i的能力高

【数据范围】
n<=1e5，pi<=1e9

【样例1输入】
```
5
804289384
846930887
681692778
714636916
957747794
1
1
2
3
```

【样例1输出】
```
2
0
1
0
0
```

【解析】
树状数组+dfs

访问x时，将p[x]加入树状数组中，由于ans[x]取决于x的子树，接下来访问x的子树时，所有子树内的p[]都会加入到树状数组中，而在访问x之前，树状数组还有其他点的p[]，则两者做一个差即可得到子树内比p[x]大的节点数量。

感 谢 观 看

主　讲：李天宇

日　期：2024.07.03

若二叉树叶子结点数为n0，度为2的结点数为n2，则一定满足n0=n2+1。

证明：

设度为1的结点数为n1，s为二叉树总的结点数，因为一个结点的度只能为0/1/2，则有s=n0+n1+n2。同时，除了根节点外，每个结点都是孩子结点，即s=孩子结点数+1，而孩子结点数=n1+2*n2，因此s=n1+2*n2+1

联立得n0+n1+n2=n1+2*n2+1
即n0=n2+1

星球大战[JSOI2008]

【题目描述】

很久以前，在一个遥远的星系，一个黑暗的帝国靠着它的超级武器统治着整个星系。

某一天，凭着一个偶然的机遇，一支反抗军摧毁了帝国的超级武器，并攻下了星系中几乎所有的星球。这些星球通过特殊的以太隧道互相直接或间接地连接。

但好景不长，很快帝国又重新造出了他的超级武器。凭借这超级武器的力量，帝国开始有计划地摧毁反抗军占领的星球。由于星球的不断被摧毁，两个星球之间的通讯通道也开始不可靠起来。

现在，反抗军首领交给你一个任务：给出原来两个星球之间的以太隧道连通情况以及帝国打击的星球顺序，以尽量快的速度求出每一次打击之后反抗军占据的星球的连通块的个数。（如果两个星球可以通过现存的以太通道直接或间接地连通，则这两个星球在同一个连通块中）。

【输入格式】
第一行n,m表示星球数目及隧道的数目，星球编号：0~n-1
接下来m行，每行x，y表示x和y之间有隧道
接下来一行一个整数k，代表遭受攻击星球的数目
接下来k行，每行一个整数，按照顺序列出攻击目标。

【输出格式】
第一行是开始时星球连通块个数。接下来k行，每行一个整数代表打击后现存的连通块个数。

100%数据，m<=2e5，n<=2m

【样例1输入】

```
8 13
0 1
1 6
6 5
5 0
0 6
1 2
2 3
3 4
4 5
7 1
7 2
7 6
3 6
5
1
6
3
5
7
```

【样例1输出】

```
1
1
1
2
3
3
```

树上操作【HAOI2015】

【题目描述】

有一棵点数为 N 的树，以点 1 为根，且树有点权。然后有 M 个操作，分为三种：
操作 1：把某个节点 x 的点权增加 a。
操作 2：把某个节点 x 为根的子树中所有点的点权都增加 a。
操作 3：询问某个节点 x 到根的路径中所有点的点权和。

【输入格式】

第一行包含两个整数 N,M。表示点数和操作数。
接下来一行 N 个整数，表示树中节点的初始权值。
接下来 N−1 行每行两个正整数 from,to， 表示该树中存在一条边 (from,to)。
再接下来 M 行，每行分别表示一次操作。其中第一个数表示该操作的种类，之后接这个操作的参数。飞往vj号星球。

【输出格式】

对于每个询问操作，输出该询问的答案。答案之间用换行隔开。