T1 奇怪题

可以发现只要找出长方体的所有边界就可以求出长方体体积，即长方体内部点数，因为所有点不重合，所以与给出点数不同就不形成长方体。

T2 博弈题

问题一：求所有先手必胜的节点的编号

可以先考虑暴力

可以知道 对于每个节点 若不往回走 则只要它边上有必败点 那么它就是必胜点

若是以某个节点为根节点的一个节点是叶节点 那么不往回走它必败

所以只有 $n=1$ 时没有必胜点 注意特判 那就可以对每个节点跑一遍 dfs 得到答案

然后考虑如何优化成 $O(n)$

在做过一次暴力的 dfs 后 可以发现

每个节点（除了选定的根节点）都只有一个来自与其相连的节点的信息没有转移 那就是它的父节点考虑如何转移

转移方向肯定是自根节点而下 那就假设要处理的节点的父节点已经转移好了 因为这个父节点的状态去掉这个子节点转移到它身上的状态就是这个父节点不经过这个节点的状态 那就很容易了

把记录的状态改成每个节点旁边的必败节点数 如果它旁边没有必败节点那么它必败 否则它必胜

在第二次转移时 子节点的状态是在父节点的转移之后的 所以可以直接相减子节点的状态是不是必败节点 若减完之后父节点是必败节点 那子节点旁边的必败节点数加一

注意根节点不用转移

问题二：求节点的每步双方都是最优移法的走法数

同样先考虑暴力

同问题一 我们要求出对于每个节点不往回走的状态转移方程

为了保证移法最优 若转移的子节点必败 只能向父节点的胜利数转移 否则父节点的失败数转移

先把叶节点的必败数加一

那就可以对每个节点跑一遍 dfs 得到答案

时间复杂度 $O(n^2)$

然后考虑如何优化成 $O(n)$

同问题一 只要把子节点**成功**转移的转移数减掉 就向这个子节点可以转移状态数

并且可以发现加一的叶节点的必败数对必胜数没有影响

代码

```
#include<iostream>
#include<cstdio>
using namespace std;
struct modein{
    modein operator >>(int &x){
        x=0;char c=getchar();
        while(c<'0'||c>'9')c=getchar();
        while(c>='0'&&c<='9')x=(x<<1)+(x<<3)+(c^48),c=getchar();
```

```
        return (modein){};
    }
}qin;
struct modeout{
    void wrte(int x){
        if(x>9)wrte(x/10);
        putchar(x%10^48);
    }
    modeout operator <<(const int &x){
        wrte(x),putchar(' ');
        return (modeout){};
    }
}qout;
struct edge{
    int next,to;
}e[2000010];
int head[1000010],n,cnt,k;
int dp[1000010][2],goin[1000010];
void add(int a,int b){
    goin[b]++;
    e[++cnt].to=b;
    e[cnt].next=head[a];
    head[a]=cnt;
}
bool dfs(int p,int f){
    for(int i=head[p];i;i=e[i].next){
        if(e[i].to!=f){
            if(dfs(e[i].to,p))dp[p][1]+=dp[e[i].to][0];
            else dp[p][0]+=dp[e[i].to][1];
        }
    }
    return !dp[p][1];
}
void cfs(int p,int f){
    if(dp[f][1]==((dp[p][1])?0:dp[p][0]))dp[p][1]+=dp[f][0]-dp[p][1];
    else dp[p][0]+=dp[f][1]-((dp[p][1])?0:dp[p][0]);
    for(int i=head[p];i;i=e[i].next)
        if(e[i].to!=f)cfs(e[i].to,p);
}
int main(){
    qin>>n>>k;if(n==1)return !puts("NO!");
    for(int i=1,a,b;i<n;i++)qin>>a>>b,add(a,b),add(b,a);
    for(int i=1;i<=n;i++)if(goin[i]==1)dp[i][0]=1;
    dfs(1,1);for(int i=head[1];i;i=e[i].next)cfs(e[i].to,1);
    if(k==1){for(int i=1;i<=n;i++)if(dp[i][1])qout<<i;
    }else for(int i=1;i<=n;i++)if(dp[i][1])qout<<dp[i][1];
    return 0;
}
```

T3 政治题

$Subtask-1$：暴力 $O(n\times m^t)$，20分。

只要把不是第 0 天的出去方式减一就可以了，注意一下它只能转移到自己的情况。

$Subtask-2/3$：一个性质，（10+10）分

对于一个环来说，无论是哪天，在某个节点上的概率肯定都相等，所以答案应为所有节点的幸福值的平均数。

而所有节点的幸福值相等的话，答案肯定是这个幸福值，刚好等于所有节点的幸福值的平均数。

$Subtask-4$ dp $O((n+m)\times t+t^2)$ 30分

先求可以往回走的情况。

先考虑一条链上的情况。

发现对于一条长为 $t+1$ 的单向链，设第一个节点为 0，则到节点 i 的概率是 $C_t^ip^i(1-p)^{t-i}$

再考虑有根单向树（根节点是 0）（方向是向深度大的连且所有的叶节点的深度都是 t）对于一个节点，其子节点的概率肯定要除以该节点的**出度**。

于是对于某个节点 i，它的概率是 $\dfrac{C_t^ip^i(1-p)^{t-i}}{\prod\limits_{j\in(i\to 0]}(du_j-1)}$（$(i\to 0]$ 是 i 到 0 的路径，不包括 i 但包括 0）。

那么就可以 dp 了，设 dp_i^j 是第 i 个节点是在单向链第 j 个时（注意不是第 j 天），单向链上第 $t\to j$ 个节点的答案之和。

则 $dp_i^j=V_iC_t^jp^j(1-p)^{t-j}+\dfrac{\sum\limits_{k\in son_i}dp_k^{j+1}}{du_i}$。

然后求不可以往回走的情况。

因为原来做法是根据有根单向树得出的，所以对于有根单向树上的某个节点 i，它的概率还是 $\dfrac{C_t^ip^i(1-p)^{t-i}}{\prod\limits_{j\in(i\to 0]}(du_j-1)}$。

设 dp_i^j 是第 i 个节点是在单向链第 j 个时（注意不是第 j 天），单向链上第 $t\to j$ 个节点的答案之和。

那是不是 $dp_i^j=V_iC_t^jp^j(1-p)^{t-j}+\dfrac{\sum\limits_{k\in son_i}dp_k^{j+1}}{du_i}$ 还成立呢?

当然不是，显然上一轮从 dp_a^{c+1} 转移到 dp_b^c 的部分是不能转移到 dp_a^{c-1} 的（因为算法考虑的是在单向链上的情况，所以不用考虑 c 的差超过 1 的情况）。

那是不是 $dp_i^j=V_iC_t^jp^j(1-p)^{t-j}+\dfrac{\sum\limits_{k\in son_i}(\frac{du_k\times(dp_k^{j+1}-V_kC_t^{j+1}p^{j+1}(1-p)^{t-j-1})-dp_i^{j+2}}{du_k-1}+V_kC_t^{j+1}p^{j+1}(1-p)^{t-j-1})}{du_i}$ 就成立呢?

虽然这看起来已经复杂得看起来像正解，但它不是。

可以从上方的 $Subtask-1$发现，一度点不是 0 号点时，它只能转移到自己，也就是在长度是 $t+1$ 的单向链上它的后方全是它：

$0\to1\to2\to3\to3\to3$ （3 是一度点）

那就是 $dp_i^j=V_iC_t^jp^j(1-p)^{t-j}+dp_i^{j+1}$

但这样还是不对 ————

可以发现如果是一度点从出来，那么这时一度点一定是是 $t+1$ 的单向链的 0 号点。

如果从与一度点相连的**两度点**出来（不回到这个一度点），那么这个两度点一定是是 $t+1$ 的单向链的 0 号点或 1 号点。

于是可以发现，如果这个图上有一条**孤立**的链（即这条链上只有两度点和一度点），设这条链的一度点是 0，

那么从 0 往外走的限制是第 ≤0 天，第 ≤1 天，第 ≤2 天，第 ≤3 天 .....

可以发现其他点没有这样的限制。

特判一下这样的向外的边就好了。

$Subtask-5$ 组合数优化 $O((n+m)\times t)$ 30分

优化一下就好了。

代码

```
#include<iostream>
#include<cstdio>
using namespace std;
struct modein{
    modein operator >>(int &x){
        x=0;char c=getchar();
        while(c<'0'||c>'9')c=getchar();
        while(c>='0'&&c<='9')x=(x<<1)+(x<<3)+(c^48),c=getchar();
        return (modein){};
    }
}qin;
const long long mod=998244353;
struct edge{
    int next,to,v[2];bool usd[2];
    int &operator [](int x){return x<2?v[x]:to;}
    bool &operator ()(bool x){return usd[x];}
}e[1000010];
int head[1000010],v[1000010],is[1000010],cnt=1,n,m,t;bool now;
long long inv[1000010],onv[1000010],dp[1000010][2],res;
long long o[2],u[2],goin[1000010],cs[1000010][2];
void add(int a,int b){
    goin[a]++,goin[b]++;
```

```
    e[++cnt][2]=b;
    e[cnt].next=head[a];
    is[head[a]=cnt]=t;
    e[++cnt][2]=a;
    e[cnt].next=head[b];
    is[head[b]=cnt]=t;
}
long long qk(long long a,long long b){
    long long ret=1;
    while(b){
        if(b&1)ret=ret*a%mod;
        a=a*a%mod,b>>=1;
    }
    return ret;
}
long long c(int a,int b){
    return onv[b]*inv[a]%mod*inv[b-a]%mod;
}
long long got(int i,int k){
    return c(k,t)*v[i]%mod*qk(u[0],k)%mod*qk(u[1],t-k)%mod;
}
void dfs(int p,int f,int c){
    if(goin[p]>2)return;
    for(int i=head[p];i;i=e[i].next)
        if(e[i][2]!=f)is[i]=c,dfs(e[i][2],p,c+1);
}
int main(){
    qin>>n>>m>>t,scanf("%lld/%lld",&o[0],&o[1]);
    u[0]=o[0]*qk(o[1],mod-2)%mod,u[1]=(o[1]-o[0])*qk(o[1],mod-2)%mod;
    for(int i=1;i<=n;i++)qin>>v[i];onv[0]=1;
    for(int i=1,a,b;i<=m;i++)qin>>a>>b,add(a,b);
    for(int i=1;i<=t;i++)onv[i]=onv[i-1]*i%mod;
    inv[t]=qk(onv[t],mod-2);
    for(int i=t-1;~i;i--)inv[i]=inv[i+1]*(i+1)%mod;
    for(int i=1;i<=n;i++)if(goin[i]==1)dfs(i,i,0);
    for(int k=t-1;~k;k--,now=!now){
        for(int i=1;i<=n;i++)cs[i][now]=dp[i][now]=0;
        for(int i=1;i<=cnt;i++)e[i](now)=false;
        for(int i=1;i<=n;i++)for(int _=head[i];_;_=e[_].next){
            if(is[_]>=k){
                if(e[_^1](!now)){
                    e[_][now]=(dp[e[_][2]][!now]*cs[e[_][2]][!now]%mod-e[_^1]
[!now]+mod)%mod;
                    e[_][now]=(e[_][now]*qk(cs[e[_][2]][!now]-1,mod-
2)%mod+got(e[_][2],k+1))%mod;
                }else e[_][now]=(dp[e[_][2]][!now]+got(e[_][2],k+1))%mod;
                dp[i][now]=(dp[i][now]+e[_][now])%mod,cs[i][now]++,e[_]
(now)=true;
            }else if(goin[i]==1)dp[i][now]=(dp[i][now]+dp[i]
[!now]+got(i,k+1))%mod,cs[i][now]=1;
        }
        for(int i=1;i<=n;i++)dp[i][now]=dp[i][now]*qk(cs[i][now],mod-2)%mod;
    }
    for(int i=1;i<=n;i++)res=(res+dp[i][!now]+got(i,0))%mod;
    return !(cout<<res*qk(n,mod-2)%mod);
}
```

T4 地理题

寻宝 题解

暴力

可以用 bitset 记一下走到过的节点然后就好了，复杂度我猜是 $O(nmk)$，但好像不是。

代码

```
#include<iostream>
#include<cstdio>
#include<queue>
#include<bitset>
#include<algorithm>
using namespace std;
struct modein{
    template<class T>
    modein operator >>(T &x){
        x=0;char c=getchar();
        while(c<'0'||c>'9')c=getchar();
        while(c>='0'&&c<='9')x=(x<<1)+(x<<3)+(c^48),c=getchar();
        return (modein){};
    }
}qin;
struct edge{
    int next,to,v;
}e[1000010];
int head[1000010],cnt;
void add(int a,int b,int c){
    e[++cnt].to=b;
    e[cnt].next=head[a];
    e[cnt].v=c;
    head[a]=cnt;
}
int n,m,T,k;
queue<pair<pair<int,int>,bitset<110> > >q;
bitset<110>dis[110];
int main(){
    qin>>n>>m>>k>>T;
    for(int i=1,a,b,c;i<=m;i++)qin>>a>>b>>c,add(a,b,c);
    q.push({{k,1},(bitset<110>)(0)});
    for(pair<pair<int,int>,bitset<110> >p;!q.empty();){
        p=q.front(),q.pop();
        for(int i=head[p.first.second];i;i=e[i].next){
            if(p.first.first-e[i].v<0)continue;
            if(!p.second[e[i].to])dis[e[i].to][p.first.first-e[i].v]=true;
            q.push({{p.first.first-e[i].v,e[i].to},p.second|(((bitset<110>)1)
<<e[i].to)});
        }
    }
    for(int a;T--;)
        qin>>a,cout<<dis[a].count()<<endl;
    return 0;
}
```

正解

引理

这个图性质：

从 1 号点（即起点）出发可以到达其他所有地点，两个地点之间至多有一条单向边，从一个地点出发到另一个地点简单路径至多只有一条，且没有自环。

可以说明这个图是一个所有环上的边都是一个朝向的仙人掌，这个应该可以直接看出来。

证明：

先求出以起点出发每个点的 dfs 序。

如果一个点有多条从 dfs 序比它小的点向它连的入边，那么从起点出发到它有多条简单路径。

所以一个点有且仅有一条从 dfs 序比它小的点向它连的入边（除了起点）（记为 1 类边）。

可以把所有 1 类边取出建一棵树，起点为根节点。

剩下的边就是从 dfs 序比它大的点向它连的边（记为 2 类边）。

如果有 2 类边 $b \to a$，a 不是 b 的在树上的祖先，那么有 $1 \to b \to a$ 和原来的 $1 \to a$ 两条简单路径。

所以所有 2 类边 $b \to a$ 满足 a 是 b 的在树上的祖先，即存在 $a \to b$ 的 1 类边简单路径。

设一个边权 $check$。

对于所有 2 类边 $b \to a$，把 $a \to b$ 经过的 1 类边的 $check++$。

如果有 1 类边 $a \to b$ 的 $check > 1$，那么 $b \to a$ 有多条简单路径。

所以所有 1 类边的 $check \leq 1$。

可以发现几个连续的 $check = 1$ 的 1 类边和对应的一条 2 类边成一个边都是一个朝向的环。

因为 $check \leq 1$ 所以每条边最多只在一个环里，而且可以有一个点在多个环里，即仙人掌。

所以这个图是一个所有环上的边都是一个朝向的仙人掌。

转化

因为从起点出发到一个点简单路径至多只有一条，即必须经过的路径。

于是到这个点的最多剩余钱数是经过这条路径的钱数。

可以经过环多花钱，环的经过次数无上限。

而对于环组成的联通块（即一棵树）（由一个点在多个环形成）一个环被选中就要使得它的父亲环也被选中。

根节点是起点出发到终点的简单路径经过的节点所在的环，但是因为终点只能经过一次所以终点所在环不能选。

暴力做法

可以状压枚举环被选中的状态 check 一下再跑同余最短路，复杂度 $O(n + T2^n n^2 logn)$。

正确做法

因为只有根节点可以直接无限制的选，所以同余最短路的开始基数使是所有根节点的环权大小最小值。

假如求出了一个节点的所有子节点的答案，可以直接对它们取 min。

然后因为必须这个节点选了它的子节点才可以选，所以同余最短路应该从这个节点的值开始，然后以所有子节点的答案取 min 和这个节点的值为普通同余最短路的初始值跑同余最短路就可以得到这个节点的答案。

再把所有树合并就可以了。

复杂度 $O(n+Tn^3logn)$，可以用同余最短路转圈法去掉 log 但是各种卡不满，常数极小，再加上数据有点弱，所以可以过。

常数小证明：因为仙人掌环数是小于 $\frac{n-1}{2}$ 的而且环的大小小于 100，所以复杂度实际上是和 100^3T 差不了多少的，再加上终点的限制会使得很多点都大大地跑不满，而且每个点最多询问一次，会使得常数变成几百分之一，所以可以过。

代码

```
#include<iostream>
#include<cstdio>
#include<queue>
#include<vector>
using namespace std;
#define long long long
struct modein{
    template<class T>
    modein operator >>(T &x){
        x=0;char c=getchar();
        while(c<'0'||c>'9')c=getchar();
        while(c>='0'&&c<='9')x=(x<<1)+(x<<3)+(c^48),c=getchar();
        return (modein){};
    }
}qin;
struct edge{
    int next,to,v;
}e[100010];
int head[100010],in[100010],to[100010],cnt,n,m,T;
void add(int a,int b,int c){
    e[++cnt].to=b;
    e[cnt].next=head[a];
    e[cnt].v=c;
    head[a]=cnt;
}
int pt[100010],low[100010],dfn[100010],idx;
int pv[100010],dis[100010],hav[100010],tot,cs[100010];
int heap[100010],st[100010],top,now;long k,ans,ret;
int dp[1010][5010][2],sk[100010],tok;bool ban[100010],usd[100010];
priority_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > >q;
void app(int a,int b){
    if(!a||!b)return;
    e[++cnt].to=b;
    e[cnt].next=heap[a];
```

```
    heap[a]=cnt;
    e[++cnt].to=a;
    e[cnt].next=heap[b];
    heap[b]=cnt;
}
void make(int p,int f,int w){
    hav[++tot]=w;
    for(int i=p;i!=f;i=pt[i])in[i]=tot,hav[tot]+=pv[i],cs[tot]++;
    to[tot]=f;
}
void dfs(int p){
    dfn[p]=++idx;
    for(int i=head[p];i;i=e[i].next){
        if(!dfn[e[i].to]){
            dis[e[i].to]=dis[p]+e[i].v;
            pt[e[i].to]=p,pv[e[i].to]=e[i].v;
            dfs(e[i].to);
        }else make(p,e[i].to,e[i].v);
    }
}
void cfs(int p){
    usd[p]=true;
    for(int i=heap[p];i;i=e[i].next){
        if(!ban[e[i].to]&&!usd[e[i].to]){
            cfs(e[i].to);
            for(int j=0;j<now;j++)dp[p][j][1]=min(dp[p][j][1],dp[e[i].to][j][0]);
        }
    }
    dp[p][hav[p]%now][1]=min(dp[p][hav[p]%now][1],hav[p]),tok=0;
    for(int i=1;i<now;i++)if(dp[p][i][1]<1e9)sk[++tok]=dp[p][i][1];
    while(!q.empty())q.pop();dp[p][hav[p]%now]
[0]=hav[p],q.push({hav[p],hav[p]%now});
    for(pair<int,int>t;!q.empty();){
        t=q.top(),q.pop();
        if(t.first!=dp[p][t.second][0])continue;
        for(int i=1;i<=tok;i++){
            if(t.first+sk[i]<dp[p][(t.first+sk[i])%now][0]){
                dp[p][(t.first+sk[i])%now][0]=t.first+sk[i];
                q.push({dp[p][(t.first+sk[i])%now][0],(t.first+sk[i])%now});
            }
        }
    }
}
int main(){
    qin>>n>>m>>k>>T;
    for(int i=1,a,b,c;i<=m;i++)qin>>a>>b>>c,add(a,b,c);
    dfs(1);
    for(int i=1;i<=tot;i++)app(in[to[i]],i);
    for(int a;T--;){
        qin>>a,top=0,now=1e9,ret=k-dis[a];
        for(int i=1;i<=tot;i++)usd[i]=ban[i]=false;ban[in[a]]=true;
        for(int i=1;i<=tot;i++)if(to[i]==a)ban[i]=true;
        for(int i=a;i;i=pt[i]){
            if(in[i]&&!ban[in[i]]&&!usd[in[i]])st[++top]=in[i],usd[in[i]]=true;
            for(int
j=1;j<=tot;j++)if(to[j]==i&&!ban[j]&&!usd[j])st[++top]=j,usd[j]=true;
```

```
        }
        for(int i=1;i<=top;i++)now=min(now,hav[st[i]]);
        if(now<1e9){
            for(int i=0;i<=tot;i++)for(int j=0;j<now;j++)dp[i][j][0]=dp[i][j]
[1]=1e9;
            for(int i=1;i<=top;i++)cfs(st[i]);tok=0;
            for(int i=1;i<=top;i++)for(int j=1;j<now;j++)dp[0][j][1]=min(dp[0][j]
[1],dp[st[i]][j][0]);
            for(int i=1;i<now;i++)if(dp[0][i][1]<1e9)sk[++tok]=dp[0][i][1];
            while(!q.empty())q.pop();dp[0][0][0]=0,q.push({0,0});
            for(pair<int,int>t;!q.empty();){
                t=q.top(),q.pop();
                if(t.first!=dp[0][t.second][0])continue;
                for(int i=1;i<=tok;i++){
                    if(t.first+sk[i]<dp[0][(t.first+sk[i])%now][0]){
                        dp[0][(t.first+sk[i])%now][0]=t.first+sk[i];
                        q.push({dp[0][(t.first+sk[i])%now][0],
(t.first+sk[i])%now});
                    }
                }
            }
            for(int i=0;i<now;i++)if(dp[0][i][0]<=ret&&dp[0][i][0]!=1e9)ans+=
(ret-dp[0][i][0])/now+1;
            cout<<ans<<endl,ans=0;
        }else cout<<(ret>=0)<<endl;
    }
    return 0;
}
```