

[比赛链接](#)

题解

C. Cover the Paths

题意

给定一棵树和若干路径，要求选出最小的点集，使得每条路径上至少有一个点。

题解

强制以 1 为根，根据每条路径两端点的 lca 深度从大到小排序。

对于 lca 深度最大的路径，显然必须选择一个点，由于其他路径的 lca 深度都不小于该路径，显然选择该路径的 lca 最优。

然后再考虑深度第二大的点，如果该路径上已经有点被选就不再选点，否则再选一个 lca 不断贪心。

贪心正确性是由于该决策顺序下路径 lca 深度递增，所以之前选的点深度越小越好，因此前面的路径的点只选 lca 且尽量少选。

```
const int MAXN=1e5+5;
#define lowbit(x) ((x)&(-x))
namespace Tree{
    int c[MAXN];
    void add(int pos){
        while(pos<MAXN){
            c[pos]++;
            pos+=lowbit(pos);
        }
    }
    int query(int pos){
        int ans=0;
        while(pos){
            ans+=c[pos];
            pos-=lowbit(pos);
        }
        return ans;
    }
    int query(int L,int R){
        return query(R)-query(L-1);
    }
}
struct Edge{
    int to,next;
```

```
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int d[MAXN],sz[MAXN],f[MAXN],dfn[MAXN],dfs_t;
int h_son[MAXN],mson[MAXN],p[MAXN];
void dfs_1(int u,int fa,int depth){
    sz[u]=1,f[u]=fa,d[u]=depth,mson[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs_1(v,u,depth+1);
        sz[u]+=sz[v];
        if(sz[v]>mson[u]){
            h_son[u]=v;
            mson[u]=sz[v];
        }
    }
}
void dfs_2(int u,int top){
    dfn[u]=++dfs_t,p[u]=top;
    if(mson[u])
        dfs_2(h_son[u],top);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==f[u]||v==h_son[u])
            continue;
        dfs_2(v,v);
    }
}
int query(int u,int v){
    int ans=0;
    while(p[u]!=p[v]){
        if(d[p[u]]<d[p[v]])
            swap(u,v);
        ans+=Tree::query(dfn[p[u]],dfn[u]);
        u=f[p[u]];
    }
    if(d[u]>d[v])
        swap(u,v);
    ans+=Tree::query(dfn[u],dfn[v]);
    return ans;
}
int lca(int u,int v){
    while(p[u]!=p[v]){
        if(d[p[u]]<d[p[v]])swap(u,v);
        u=f[p[u]];
    }
}
```

```

        return d[u]<d[v]?u:v;
    }
    struct Node{
        int u,v,p;
        bool operator < (const Node b)const{
            return d[p]>d[b.p];
        }
    }node[MAXN];
    int main(){
        int n=read_int();
        _for(i,1,n){
            int u=read_int(),v=read_int();
            Insert(u,v);
            Insert(v,u);
        }
        dfs_1(1,0,0);
        dfs_2(1,1);
        int m=read_int();
        _for(i,0,m){
            int u=read_int(),v=read_int(),p=lca(u,v);
            node[i]=Node{u,v,p};
        }
        sort(node,node+m);
        vector<int> ans;
        _for(i,0,m){
            if(query(node[i].u,node[i].v)==0){
                ans.push_back(node[i].p);
                Tree::add(dfn[node[i].p]);
            }
        }
        enter(ans.size());
        for(int u:ans)
            space(u);
        return 0;
    }

```

G. Berland Post

题意

给定若干条形如 $x_v + b_i \leq x_u + T$ 的限制，求最小的 T 以及这种情况下的一组合法解 $(x_1, x_2, \dots, x_n)$

其中，某些 x_i 的值已经固定。

题解

将方程转化为 $x_v \leq x_u + T - b_i$ 考虑二分 T 然后差分约束求解。

设超级源点为 s 关于点权的限制，对于无限制的 x_i 连边 $(s, i, \infty), (i, s, -\infty)$ 表示 $-\infty \leq x_i \leq \infty$

对于固定的 x_i 连边 $(s, i, v), (i, s, -v)$ 表示 $v \leq x_i \leq v$ 时间复杂度 $O(nm \log V)$

```
const int MAXN=1e3+5,MAXM=4e3+5,SPV=2e5;
const double inf=1e15,infv=1e7,eps=1e-7,eps2=1e-9;
int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(c==' '||c=='\r'||c=='\n')c=getchar();
    if(c=='?')return SPV;
    if(c=='-')sign=true,c=getchar();
    while(isdigit(c)){
        t=t*10+c-'0';
        c=getchar();
    }
    return sign?-t:t;
}
struct Edge{
    int to,next;
    double w;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v,double w){
    ++edge_cnt;
    edge[edge_cnt].to=v;
    edge[edge_cnt].w=w;
    edge[edge_cnt].next=head[u];
    head[u]=edge_cnt;
}
namespace SPFA{
    double dis[MAXN];
    int len[MAXN];
    bool inque[MAXN];
    bool solve(int src,int n){
        queue<int>q;
        _rep(i,1,n){
            inque[i]=false;
            len[i]=0;
            dis[i]=inf;
        }
        dis[src]=0;len[src]=1;
        q.push(src);
        inque[src]=true;
        while(!q.empty()){
            int u=q.front();q.pop();
            inque[u]=false;
            for(int i=head[u];i;i=edge[i].next){

```

```

        int v=edge[i].to;
        if(dis[v]>dis[u]+edge[i].w+eps2){
            dis[v]=dis[u]+edge[i].w;
            len[v]=len[u]+1;
            if(len[v]>n)
                return false;
            if(!inque[v]){
                q.push(v);
                inque[v]=true;
            }
        }
    }
    return true;
}

struct Edge2{
    int u,v,w;
};
vector<Edge2> edges;
int a[MAXN];
bool check(int n,double T){
    int s=++n;
    edge_cnt=0;
    _rep(i,1,n)
        head[i]=0;
    _for(i,1,n){
        if(a[i]==SPV){
            Insert(s,i,infv);
            Insert(i,s,infv);
        }
        else{
            Insert(s,i,a[i]);
            Insert(i,s,-a[i]);
        }
    }
    for(Edge2 e:edges)
        Insert(e.v,e.u,T-e.w);
    return SPFA::solve(s,n);
}

int main(){
    int n,m;
    while(cin>>n>>m){
        edges.clear();
        _rep(i,1,n)
            a[i]=read_int();
        while(m--){
            int u=read_int(),v=read_int(),w=read_int();
            edges.push_back(Edge2{u,v,w});
        }
        double lef=0,rig=infv;
    }
}

```

```
while(rig-lef>eps){
    double mid=(lef+rig)/2;
    if(check(n,mid))
        rig=mid;
    else
        lef=mid;
}
printf("%.6lf\n",lef);
check(n,lef);
_rep(i,1,n)
printf("%.6lf ",SPFA::dis[i]);
puts("");
}
return 0;
}
```

J. Subsequence Sum Queries

题意

给定一个长度为 n 的序列，接下来 q 个询问，每次询问区间 $[l,r]$ 有多少个子序列满足元素之和整除 m

题解

考虑分治处理询问。设当前维护区间为 $[lef,rig]$ $mid=\frac{lef+rig}{2}$ 对 $[l,r]$ 在 $[lef,mid],[mid+1,rig]$ 的询问直接递归到左右区间处理。

对于跨 mid 的询问，提前 $O((rig-lef)m)$ 处理出区间 $[i,mid](lef\leq i\leq mid),[mid+1,j](mid+1\leq j\leq rig)$ 的答案。

然后对每个询问相当于背包合并，时间复杂度 $O(qm^2)$ 于是总时间复杂度 $O(nm\log n+qm^2)$

```
const int MAXN=2e5+5,MAXM=20,mod=1e9+7;
int ans[MAXN],a[MAXN],m;
struct query{
    int lef,rig,id;
};
int s[MAXN][MAXM],temp[MAXM<<1];
void solve(int lef,int rig,vector<query> b){
    int mid=lef+rig>>1;
    if(lef==rig){
        _for(i,0,b.size())
            ans[b[i].id]=1+(a[mid]==0);
        return;
    }
```

```

    mem(s[mid],0);
    s[mid][0]++;
    s[mid][a[mid]]++;
    for(int i=mid-1;i>=lef;i--){
        _for(j,0,m)
            s[i][(a[i]+j)%m]=(s[i+1][(a[i]+j)%m]+s[i+1][j])%mod;
    }
    mem(s[mid+1],0);
    s[mid+1][0]++;
    s[mid+1][a[mid+1]]++;
    for(int i=mid+2;i<=rig;i++){
        _for(j,0,m)
            s[i][(a[i]+j)%m]=(s[i-1][(a[i]+j)%m]+s[i-1][j])%mod;
    }
    vector<query>b1,b2;
    _for(i,0,b.size()){
        if(b[i].rig<=mid)
            b1.push_back(b[i]);
        else if(b[i].lef>mid)
            b2.push_back(b[i]);
        else{
            mem(temp,0);
            _for(j,0,m)_for(k,0,m)
                temp[j+k]=(temp[j+k]+1LL*s[b[i].lef][j]*s[b[i].rig][k])%mod;
            ans[b[i].id]=(temp[0]+temp[m])%mod;
        }
    }
    solve(lef,mid,b1);
    solve(mid+1,rig,b2);
}
int main()
{
    int n=read_int();
    m=read_int();
    _rep(i,1,n)a[i]=read_LL()%m;
    int q=read_int();
    vector<query> b;
    _for(i,0,q){
        int l=read_int(),r=read_int();
        b.push_back(query{l,r,i});
    }
    solve(1,n,b);
    _for(i,0,q)
        enter(ans[i]);
    return 0;
}

```

L. Increasing Costs

题意

给定一个无向连通图，定义源点为 1 号点。对每条边，询问删除该边会导致源点到多少个点的最短路改变。

题解

首先跑最短路，然后保留所有在最短路树上的边，同时规定每条边方向由距离近的点指向距离远的点，易知构成有向无环图。

问题转化为求有向无环图的支配边。

对任意一个点，如果该点有至少两条入边，易知所有入边支配点集均为空，否则该边的支配点集等价于该点的支配子树。

```
const int MAXN=2e5+5,MAXM=2e5+5,MAXV=22;
namespace Tree{
    struct Edge{
        int to,id,next;
    }edge[MAXN+MAXM];
    int head1[MAXN],head2[MAXN],edge_cnt;
    int deg[MAXN],f[MAXN],dep[MAXN],anc[MAXN][MAXV],lg2[MAXN];
    int deg0[MAXN];
    void Insert1(int u,int v,int id){
        edge[++edge_cnt]=Edge{v,id,head1[u]};
        head1[u]=edge_cnt;
        deg[v]++;
        deg0[v]++;
    }
    void Insert2(int u,int v){
        edge[++edge_cnt]=Edge{v,0,head2[u]};
        head2[u]=edge_cnt;
    }
    int LCA(int u,int v){
        if(dep[u]<dep[v])
            swap(u,v);
        while(dep[u]>dep[v])u=anc[u][lg2[dep[u]-dep[v]]];
        if(u==v)
            return u;
        for(int i=MAXV-1;i>=0;i--){
            if(anc[u][i]!=anc[v][i])
                u=anc[u][i],v=anc[v][i];
        }
        return anc[u][0];
    }
    int build(int n){
```

```

    lg2[1]=0;
    _for(i,2,MAXN)lg2[i]=lg2[i>>1]+1;
    int rt=n+1;
    queue<int> q;
    _rep(i,1,n){
        if(deg[i]==0){
            f[i]=rt;
            q.push(i);
        }
    }
    while(!q.empty()){
        int u=q.front();q.pop();
        dep[u]=dep[f[u]]+1;
        Insert2(f[u],u);
        anc[u][0]=f[u];
        for(int i=1;i<MAXV;i++){
            anc[u][i]=anc[anc[u][i-1]][i-1];
        }
        for(int i=head1[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(f[v]==0)
                f[v]=u;
            else
                f[v]=LCA(u,f[v]);
            deg[v]--;
            if(deg[v]==0)
                q.push(v);
        }
    }
    return rt;
}
int sz[MAXN],ans[MAXM];
void dfs(int u){
    sz[u]=1;
    for(int i=head2[u];i;i=edge[i].next){
        int v=edge[i].to;
        dfs(v);
        sz[u]+=sz[v];
    }
}
void solve(int n,int m){
    int rt=build(n);
    dfs(rt);
    _rep(u,1,n){
        for(int i=head1[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(deg0[v]==1)
                ans[edge[i].id]=sz[v];
        }
    }
    _for(i,0,m)
        enter(ans[i]);
}

```

```
    }  
}  
struct Edge{  
    int to,w,id,next;  
}edge[MAXM<<1];  
int head[MAXN],edge_cnt;  
void Insert(int u,int v,int w,int id){  
    edge[++edge_cnt]=Edge{v,w,id,head[u]};  
    head[u]=edge_cnt;  
}  
LL dis[MAXN];  
bool vis[MAXN];  
int main(){  
    int n=read_int(),m=read_int();  
    _for(i,0,m){  
        int u=read_int(),v=read_int(),w=read_int();  
        Insert(u,v,w,i);  
        Insert(v,u,w,i);  
    }  
    priority_queue<pair<LL,int> >q;  
    mem(dis,127);  
    dis[1]=0;  
    q.push(make_pair(-dis[1],1));  
    while(!q.empty()){  
        int u=q.top().second;  
        q.pop();  
        if(vis[u])continue;  
        vis[u]=true;  
        for(int i=head[u];i;i=edge[i].next){  
            int v=edge[i].to;  
            if(dis[v]>dis[u]+edge[i].w){  
                dis[v]=dis[u]+edge[i].w;  
                q.push(make_pair(-dis[v],v));  
            }  
        }  
    }  
    _rep(u,1,n){  
        for(int i=head[u];i;i=edge[i].next){  
            int v=edge[i].to;  
            if(dis[v]==dis[u]+edge[i].w)  
                Tree::Insert1(u,v,edge[i].id);  
        }  
    }  
    Tree::solve(n,m);  
    return 0;  
}
```

赛后总结

jxm 开局一个多小时后就罚坐了，疯狂写假题，下次一定先确认思路没问题再写题。

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest5

Last update: 2021/08/04 10:52