CCPC Wannafly Camp Day3

比赛链接

C. 无向图定向

题意

给定无向图，要求对每条边定向，得到 DAG 同时最小化最长路。

题解 1

易知可以当拓扑序确定时每条边方向是确定的，考虑枚举所有拓扑序的全排列然后计算最长路。

显然这样会 TLE 于是考虑随机化乱搞，~~居然过了。~~

```
const int MAXN=20,MAXM=200;
struct Edge{
    int to,next;
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
int deg[MAXN],dp[MAXN],a[MAXN];
vector<int> g[MAXN];
int cal(int n){
    int ans=0;
    queue<int> q;
    _rep(u,1,n){
        dp[u]=0;
        g[u].clear();
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(a[u]>a[v]){
                g[u].push_back(v);
                deg[v]++;
            }
        }
    }
    _rep(i,1,n){
        if(!deg[i])
        q.push(i);
    }
```

```
    while(!q.empty()){
        int u=q.front();
        q.pop();
        ans=max(ans,dp[u]);
        _for(i,0,g[u].size()){
            int v=g[u][i];
            dp[v]=max(dp[v],dp[u]+1);
            deg[v]--;
            if(!deg[v])
            q.push(v);
        }
    }
    return ans;
}
int main()
{
    int n=read_int();
    int m=read_int();
    _for(i,0,m){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    int ans=n-1;
    _rep(i,1,n)a[i]=i;
    while((double)clock()/CLOCKS_PER_SEC<0.9){
        random_shuffle(a+1,a+n+1);
        ans=min(ans,cal(n));
    }
    enter(ans);
    return 0;
}
```

题解 2

考虑对无向图进行染色，使得同色点之间没有连边，最小化颜色种数。颜色代表数值高的点向颜色代表数值低的点连边，此时答案为染色数 -1。

考虑 $O(n2^n)$ 标记所有独立子集，然后 $O(3^n)$ 子集枚举计算染色数。

```
int G[20],dp[1<<20];
bool ok[1<<20];
int n,m;
int main()
{
    scanf("%d%d",&n,&m);
    ok[0]=true;
    while(m--)
```

```
    {
        int x,y;
        scanf("%d%d",&x,&y);
        x--;y--;
        G[x]|=1<<y;G[y]|=1<<x;
    }
    for(int i=0;i<(1<<n);i++)
    {
        for(int j=0;j<n;j++)
        {
            ok[i|(1<<j)] |= ok[i] && !(i&(1<<j)) && !(i&G[j]);
            //取出一个独立集
            //j这个点不在方案i中并且j和当前的集合没有直接相连的边
        }
    }
    for(int i=0;i<(1<<n);i++) dp[i]=1<<20;
    dp[0]=0;
    for(int i=1;i<(1<<n);i++)
    {
        if(ok[i]) dp[i]=1; //独立集只需要一种颜色
        for(int j=i&(i-1);;j=i&(j-1))
        {
            if(ok[j])
            dp[i]=min(dp[i],dp[i^j]+1);//取出一个独立集给一种新颜色
            if(j==0) break;
        }
    }
    printf("%d\n",dp[(1<<n)-1]-1);
    return 0;
}
```

G. 火山哥周游世界

题意

给定一棵边权树和 k 个关键点，问从第 $i(1\le i\le n)$ 点出发经过所有关键点的最短路程。

题解

不难发现答案为第 i 个点与所有关键点构成的生成树的边权和的两倍 $-$ 第 i 个点到最远关键点的距离。

换根 dp 维护每个结点的生成树边权和，每个结点子树方向的最远关键结点、次远关键结点以及根节点方向的最远关键结点即可。

时间复杂度 $O(n)$

```
const int MAXN=5e5+5;
LL inf=1e15;
struct Edge{
    int to,w,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt,k;
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,w,head[u]};
    head[u]=edge_cnt;
}
LL f[MAXN],dp[MAXN][3],ans[MAXN];
int sz[MAXN],hson[MAXN];
void dfs1(int u,int fa){
    hson[u]=0;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs1(v,u);
        sz[u]+=sz[v];
        if(sz[v])f[u]+=f[v]+edge[i].w*2;
        if(dp[v][0]+edge[i].w>dp[u][0]){
            hson[u]=v;
            dp[u][1]=dp[u][0];
            dp[u][0]=dp[v][0]+edge[i].w;
        }
        else if(dp[v][0]+edge[i].w>dp[u][1])
        dp[u][1]=dp[v][0]+edge[i].w;
    }
}
void dfs2(int u,int fa){
    ans[u]=f[u]-max(dp[u][0],dp[u][2]);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        if(sz[v]==0)
        f[v]=f[u]+edge[i].w*2;
        else if(sz[v]!=k)
        f[v]=f[u];
        if(v==hson[u])
        dp[v][2]=edge[i].w+max(dp[u][1],dp[u][2]);
        else
        dp[v][2]=edge[i].w+max(dp[u][0],dp[u][2]);
        dfs2(v,u);
    }
}
int main()
{
    int n=read_int();
    k=read_int();
    _for(i,1,n){
```

```
        int u=read_int(),v=read_int(),w=read_int();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    _rep(i,1,n)
    dp[i][0]=dp[i][1]=dp[i][2]=-inf;
    _for(i,0,k){
        int u=read_int();
        dp[u][0]=dp[u][1]=dp[u][2]=0;
        sz[u]=1;
    }
    dfs1(1,0);
    dfs2(1,0);
    _rep(i,1,n)
    enter(ans[i]);
    return 0;
}
```

H. 火山哥的序列

题意

给定长度为 n 的序列 a_i，给定以下函数

$$ g(l,r)=\max_{i,j\in [1,l)\cup (r,n],i\le j}\text{gcd}(i,j) $$

若满足上述条件的 (i,j) 不存在，则 $g(l,r)=0$，求 $\sum_{i=1}^n\sum_{j=i}^ng(i,j)$。

题解

$$
\sum_{i=1}^n\sum_{j=i}^ng(i,j)=\sum_{k=1}^\text{V}k\sum_{i=1}^n\sum_{j=i}^n[g(i,j)==k]=\sum_{k=1}^\text{V}k\sum_{i=1}^n\sum_{j=i}^n([g(i,j)\ge k+1]-[g(i,j)\ge k]) $$

问题转化为如何计算 $\sum_{i=1}^n\sum_{j=i}^n[g(i,j)\ge k](k=1\sim V)$。

设 $f(l,k)=\max\{r|g(l,r)\ge k\}$，于是有

$$ \sum_{i=1}^n\sum_{j=i}^n[g(i,j)\ge k](k=1\sim V)=\sum_{i=1}^n (f(i,k)-i+1)=\sum_{i=1}^n f(i,k)-\frac {n(n-1)}2 $$

问题转化为维护 $f(1\sim n,k)$，考虑 $f(i,k+1)\to f(i,k)$ 的状态转移。

枚举 k 的倍数，假定 $a_i\mid k$ 的所有位置从小到大依次为 $p_1,p_2\cdots p_{m-1},p_m$。

若 $m\lt 2$，则有 $f(i,k)=f(i,k+1)$，否则有如下转移：

\begin{equation}\begin{split} &1\le i\le p_1\to f(i,k)=\max\left(f(i,k+1),p_{m-1}-1\right)\\ &p_1+1\le i\le p_2\to f(i,k)=\max\left(f(i,k+1),p_m-1\right)\\ &p_2+1\le i\le n\to f(i,k)=\max\left(f(i,k+1),n\right)

\end{split}\end{equation}

吉司机线段树维护区间最值操作和区间和查询即可$f(i,k)$ 初始值为 $i-1$时间复杂度 $V\log V$

```
const int MAXN=2e5+5,inf=1e9,MAXV=2e5;
int
lef[MAXN<<2],rig[MAXN<<2],minv[MAXN<<2],minc[MAXN<<2],secv[MAXN<<2],lazy[MA
XN<<2];
LL s[MAXN<<2];
void push_up(int k){
    s[k]=s[k<<1]+s[k<<1|1];
    if(minv[k<<1]==minv[k<<1|1]){
        minv[k]=minv[k<<1];
        minc[k]=minc[k<<1]+minc[k<<1|1];
        secv[k]=min(secv[k<<1],secv[k<<1|1]);
    }
    else if(minv[k<<1]<minv[k<<1|1]){
        minv[k]=minv[k<<1];
        minc[k]=minc[k<<1];
        secv[k]=min(secv[k<<1],minv[k<<1|1]);
    }
    else{
        minv[k]=minv[k<<1|1];
        minc[k]=minc[k<<1|1];
        secv[k]=min(minv[k<<1],secv[k<<1|1]);
    }
}
void push_tag(int k,int v){
    if(minv[k]>=v)return;
    s[k]+=1LL*(v-minv[k])*minc[k];
    minv[k]=lazy[k]=v;
}
void push_down(int k){
    if(lazy[k]){
        push_tag(k<<1,lazy[k]);
        push_tag(k<<1|1,lazy[k]);
        lazy[k]=0;
    }
}
void build(int k,int L,int R){
    lef[k]=L,rig[k]=R,lazy[k]=0;
    int M=L+R>>1;
    if(L==R){
        s[k]=minv[k]=M-1;
        minc[k]=1;
        secv[k]=inf;
        return;
    }
    build(k<<1,L,M);
```

```
    build(k<<1|1,M+1,R);
    push_up(k);
}
void update(int k,int L,int R,int v){
    if(minv[k]>=v)return;
    if(L<=lef[k]&&rig[k]<=R&&secv[k]>v)
    return push_tag(k,v);
    int mid=lef[k]+rig[k]>>1;
    push_down(k);
    if(mid>=L)
    update(k<<1,L,R,v);
    if(mid<R)
    update(k<<1|1,L,R,v);
    push_up(k);
}
int a[MAXN],p[MAXN];
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int();
        build(1,1,n);
        mem(p,0);
        _rep(i,1,n){
            a[i]=read_int();
            p[a[i]]=i;
        }
        LL last=0,ans=0;
        for(int i=MAXV;i;i--){
            int max1=0,max2=0,min1=inf,min2=inf,cnt=0;
            for(int j=1;i*j<=MAXV;j++){
                if(p[i*j]){
                    if(p[i*j]>=max1){
                        max2=max1;
                        max1=p[i*j];
                    }
                    else
                    max2=max(max2,p[i*j]);
                    if(p[i*j]<=min1){
                        min2=min1;
                        min1=p[i*j];
                    }
                    else
                    min2=min(min2,p[i*j]);
                    cnt++;
                }
            }
            if(cnt<2)continue;
            update(1,1,min1,max2-1);
            update(1,min1+1,min2,max1-1);
            update(1,min2+1,n,n);
```

```
            LL cur=s[1]-1LL*n*(n-1)/2;
            ans+=1LL*(cur-last)*i;
            last=cur;
        }
        enter(ans);
    }
    return 0;
}
```

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:ccpc_wannafly_winter_camp_day3

Last update: **2021/05/02 15:52**