Educational Codeforces Round 102

比赛链接

D. Program

比赛时用线段树写的，时间复杂度 $O(n+m\log n)$。

由于只查询前后缀，根据题目进行预处理可以做到 $O(n+m)$。

E. Minimum Path

题意

给定 n 阶连通图，求点 1 到其他各点的最短路，其中路径长度的定义为所有边的权值和减去最大权值加上最小权值。

题解

不难发现所有边的权值和减去最大权值加上最小权值不大于所有边的权值和减去其中任意一条边权值再加上其中任意一条边权值。

不难发现将路径长度定义改为所有边的权值和减去其中任意一条边权值再加上其中任意一条边权值后，各点最短路值仍然不变。

于是设 $dis(i,0/1,0/1)$ 表示从点 1 到点 i 且是否减去一条边权值是否加上一条边权值的最小距离，直接跑最短路算法即可。

时间复杂度 $O(n\log m)$。

```
const int MAXN=2e5+5;
int head[MAXN],edge_cnt;
struct Edge{
    int to,next,w;
}edge[MAXN<<1];
void Insert(int u,int v,int w){
    edge[++edge_cnt]=Edge{v,head[u],w};
    head[u]=edge_cnt;
}
struct Node{
    int u;
    LL dis;
    bool s1,s2;
    Node(int u=0,LL dis=0,bool s1=0,bool
```

```
s2=0):u(u),dis(dis),s1(s1),s2(s2){}
    bool operator < (const Node &b)const{
        return dis>b.dis;
    }
};
LL dis[MAXN][2][2];
bool vis[MAXN][2][2];
int main()
{
    int n=read_int(),m=read_int();
    mem(dis,-1);
    dis[1][0][0]=0;
    while(m--){
        int u=read_int(),v=read_int(),w=read_int();
        Insert(u,v,w);
        Insert(v,u,w);
    }
    priority_queue<Node>q;
    q.push(Node(1,0,0,0));
    while(!q.empty()){
        Node t=q.top();q.pop();
        if(vis[t.u][t.s1][t.s2])continue;
        vis[t.u][t.s1][t.s2]=true;
        for(int i=head[t.u];i;i=edge[i].next){
            int v=edge[i].to;
            _for(j,0,2)_for(k,0,2){
                LL d=t.dis+edge[i].w;
                bool s1=t.s1,s2=t.s2;
                if(j==1&&!s1){
                    d-=edge[i].w;
                    s1=true;
                }
                if(k==1&&!s2){
                    d+=edge[i].w;
                    s2=true;
                }
                if(dis[v][s1][s2]==-1||dis[v][s1][s2]>d){
                    dis[v][s1][s2]=d;
                    q.push(Node(v,d,s1,s2));
                }
            }
        }
    }
    _rep(i,2,n)
    space(dis[i][1][1]);
    return 0;
}
```

F. Strange Set

题意

给定一个长度为 n 的序列 A，其中 $1\le a_i\le 100$，每个 a_i 有一个权值 b_i。

要求选择一个下标集合 S，满足对任意 $i\in S$，若 $j\lt i$ 且 $a_j\mid a_i$，则 $j\in S$。

定义 S 的权值为 $\sum_{i\in S}b_i$，问 S 的权值最大和。

题解

若 $j\lt i$ 且 $a_j\mid a_i$，且加入连边 $i\to j$，不难发现这题变为最大权闭合子图模型。

另外为控制边的数量，考虑若 $k\lt j\lt i$ 且 $a_k=a_j\mid a_i$，则只连边 $i\to j$，于是只需要维护每个值的最后出现位置即可。

由于 a_i 取值小，所以连边有限，时间复杂度为 $O(n^2)$。

```
const int MAXN=3005,MAXM=60005,Inf=0x7fffffff;
struct Edge{
    int to,cap,next;
    Edge(int to=0,int cap=0,int next=0){
        this->to=to;
        this->cap=cap;
        this->next=next;
    }
}edge[MAXM<<1];
int head[MAXN],edge_cnt;
void Clear(){mem(head,-1);edge_cnt=0;}//边从0开始编号
void Insert(int u,int v,int c){
    edge[edge_cnt]=Edge(v,c,head[u]);
    head[u]=edge_cnt++;
    edge[edge_cnt]=Edge(u,0,head[v]);
    head[v]=edge_cnt++;
}
struct Dinic{
    int s,t;
    int pos[MAXN],vis[MAXN],dis[MAXN];
    bool bfs(int k){
        queue<int>q;
        q.push(s);
        vis[s]=k,dis[s]=0,pos[s]=head[s];
        while(!q.empty()){
            int u=q.front();q.pop();
            for(int i=head[u];~i;i=edge[i].next){
                int v=edge[i].to;
                if(vis[v]!=k&&edge[i].cap){
```

```
                    vis[v]=k,dis[v]=dis[u]+1,pos[v]=head[v];
                    q.push(v);
                    if(v==t)
                    return true;
                }
            }
        }
        return false;
    }
    int dfs(int u,int max_flow){
        if(u==t||!max_flow)
        return max_flow;
        int flow=0,temp_flow;
        for(int &i=pos[u];~i;i=edge[i].next){
            int v=edge[i].to;
if(dis[u]+1==dis[v]&&(temp_flow=dfs(v,min(max_flow,edge[i].cap)))){
                edge[i].cap-=temp_flow;
                edge[i^1].cap+=temp_flow;
                flow+=temp_flow;
                max_flow-=temp_flow;
                if(!max_flow)
                break;
            }
        }
        return flow;
    }
    int Maxflow(int s,int t){
        this->s=s;this->t=t;
        int ans=0,k=0;
        mem(vis,0);
        while(bfs(++k))
        ans+=dfs(s,Inf);
        return ans;
    }
}solver;
const int MAXV=105;
int a[MAXN],b[MAXN],last[MAXV];
int main()
{
    int n=read_int(),s=0,t=n+1,ans=0;
    Clear();
    _rep(i,1,n)
    a[i]=read_int();
    _rep(i,1,n){
        b[i]=read_int();
        if(b[i]>0)
        Insert(s,i,b[i]),ans+=b[i];
        else if(b[i]<0)
        Insert(i,t,-b[i]);
    }
```

```
    _rep(i,1,n){
        _for(j,1,MAXV){
            if(a[i]%j==0&&last[j])
            Insert(i,last[j],Inf);
        }
        last[a[i]]=i;
    }
    enter(ans-solver.Maxflow(s,t));
    return 0;
}
```

G. Tiles

快速傅里叶变化好题

```
const int MAXN=5005,MAXV=2e5+5,Mod=998244353;
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%Mod;
        a=1LL*a*a%Mod;
        k>>=1;
    }
    return ans;
}
namespace Poly{
    const int G=3;
    int rev[MAXN<<2],Pool[MAXN<<3],*Wn[30];
    void init(){
        int lg2=0,*pos=Pool,n,w;
        while((1<<lg2)<MAXN*2)lg2++;
        n=1<<lg2,w=quick_pow(G,(Mod-1)/(1<<lg2));
        while(~lg2){
            Wn[lg2]=pos,pos+=n;
            Wn[lg2][0]=1;
            _for(i,1,n)Wn[lg2][i]=1LL*Wn[lg2][i-1]*w%Mod;
            w=1LL*w*w%Mod;
            lg2--;n>>=1;
        }
    }
    int build(int k){
        int n,pos=0;
        while((1<<pos)<=k)pos++;
        n=1<<pos;
        _for(i,0,n)rev[i]=(rev[i>>1]>>1)|((i&1)<<(pos-1));
        return n;
    }
    void NTT(int *f,int n,bool type){
```

```
        _for(i,0,n)if(i<rev[i])
        swap(f[i],f[rev[i]]);
        int t1,t2;
        for(int i=1,lg2=1;i<n;i<<=1,lg2++){
            for(int j=0;j<n;j+=(i<<1)){
                _for(k,j,j+i){
                    t1=f[k],t2=1LL*Wn[lg2][k-j]*f[k+i]%Mod;
                    f[k]=(t1+t2)%Mod,f[k+i]=(t1-t2)%Mod;
                }
            }
        }
        if(!type){
            reverse(f+1,f+n);
            int div=quick_pow(n,Mod-2);
            _for(i,0,n)f[i]=(1LL*f[i]*div%Mod+Mod)%Mod;
        }
    }
    void Mul(int *f,int _n,int *g,int _m,int xmod=0){
        int n=build(_n+_m-2);
        _for(i,_n,n)f[i]=0;_for(i,_m,n)g[i]=0;
        NTT(f,n,true);NTT(g,n,true);
        _for(i,0,n)f[i]=1LL*f[i]*g[i]%Mod;
        NTT(f,n,false);
        if(xmod)_for(i,xmod,n)f[i]=0;
    }
}
int frac[MAXV],invfrac[MAXV],a[MAXN],b[MAXN];
int C(int n,int m){
    if(m<0||m>n)return 0;
    return 1LL*frac[n]*invfrac[m]%Mod*invfrac[n-m]%Mod;
}
int dp[MAXN<<2],temp[MAXN<<2],c[MAXN<<2];
int main()
{
    Poly::init();
    int n=read_int();
    _for(i,0,n)a[i]=read_int(),b[i]=read_int();
    frac[0]=1;
    _for(i,1,MAXV)
    frac[i]=1LL*frac[i-1]*i%Mod;
    invfrac[MAXV-1]=quick_pow(frac[MAXV-1],Mod-2);
    for(int i=MAXV-1;i;i--)
    invfrac[i-1]=1LL*invfrac[i]*i%Mod;
    dp[0]=1;
    int len=1;
    _for(i,0,n){
        _for(j,0,2*len+a[i]-b[i]-1)
        c[j]=C(a[i]+b[i],j+b[i]-len+1);
        Poly::Mul(dp,len,c,2*len+a[i]-b[i]-1);
        _for(j,0,len+a[i]-b[i])
```

```
        dp[j]=dp[j+len-1];
        len+=a[i]-b[i];
    }
    int ans=0;
    _for(i,0,len)
    ans=(ans+dp[i])%Mod;
    enter(ans);
    return 0;
}
```