Atcoder Rugular Contest 107

比赛链接

D - Number of Multisets

题意

给定 N,K要求将 K 分解为 N 个数，每个数均为 $1,\frac 12,\frac 14\cdots \frac 1{2^i}\cdots$询问分解方案数。

题解

设 $\text{dp}(i,j)$ 表示需要将当前值分解为 i 个数，且如果使用当前可以用的最大的数分解需要 j 个的方案数。

于是假如使用一个当前可以用的最大的数，则有 $\text{dp}(i,j)\gets \text{dp}(i-1,j-1)$ 表示使用一个当前可以用的最大的数。

$\text{dp}(i,j)\gets \text{dp}(i,j<<1)$ 表示禁止使用当前可以用的最大的数。于是可以 $O(nk)$ 完成转移。

```
const int MAXN=3005,Mod=998244353;
int dp[MAXN][MAXN];
int dfs(int n,int k){
    if(k>n)return 0;
    if(n==0||k==0)return n==0&&k==0;
    if(~dp[n][k])return dp[n][k];
    return dp[n][k]=(dfs(n-1,k-1)+dfs(n,k<<1))%Mod;
}
int main()
{
    int n=read_int(),k=read_int();
    mem(dp,-1);
    enter(dfs(n,k));
    return 0;
}
```

F - Sum of Abs

题意

给定 n 个点 m 条边的无向图。图中每个点有两个权值 a_i,b_i

要求选择若干点并进行删除(可以不选)，费用为所有选择的点的 a_i 之和。然后收益为剩下图中每个连通块的 b_i 之和的绝对值之和。

要求输出最大的收益 $-$ 费用。

题解

对于剩下图的一个连通块的收益，要么为 $\sum b_i$，要么均为 $\sum -b_i$。

于是对于一个点，它的收益为 $-a_i,-b_i,b_i$ 三者中的一种。

同时对于原图中一条边对应的两个点 u,v，要满足约束不出现者两个点贡献为 $-b_u,b_v$ 或 $b_u,-b_v$ 的情况。

然后开始建图，需要最大化收益，则不妨将所有收益取相反数，然后跑最小割模型。

将一个点 i 拆为 x_i 和 y_i，然后源点 $s\to x_i$ 连一条容量为 $-b_i$ 的边，$x_i\to y_i$ 连一条容量为 a_i 的边，$y_i\to t$ 连一条容量为 b_i 的边。

于是这样就可以表示一个点需要从三个收益中选择一个，接下来是对约束的控制，不妨设原图中 u,v 之间有一条连边。

于是 $y_v\to x_u$ 和 $y_u\to x_v$ 之间连一条权值为 ∞ 的边，这样就可以禁止只选择 $s\to x_u,y_v\to t$ 和 $s\to x_v,y_u\to t$ 的情况。

最后由于最小割模型不能处理负容量的边，于是考虑为原图中每个点对应的三条边加上 $|b_i|$ 的偏移量，总时间复杂度 $O(n^2(n+m))$。

```
const int MAXN=305*2,MAXM=305*5,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;}
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;
int a[MAXN],b[MAXN];
int main()
{
    Clear();
    int n=read_int(),m=read_int(),s=2*n+1,t=2*n+2,ans=0;
    _rep(i,1,n)a[i]=read_int();
    _rep(i,1,n)b[i]=read_int(),ans+=abs(b[i]);
    _rep(i,1,n){
```

```
        Insert(s,i,-b[i]+abs(b[i]));
        Insert(i,i+n,a[i]+abs(b[i]));
        Insert(i+n,t,b[i]+abs(b[i]));
    }
    while(m--){
        int u=read_int(),v=read_int();
        Insert(u+n,v,Inf);
        Insert(v+n,u,Inf);
    }
    enter(ans-solver.Maxflow(s,t));
    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:arc_107

Last update: **2021/02/17 15:08**