

# 图论 3

## 网络流经典例题

### 方格取数问题

[洛谷p2774](#)

#### 题意

给定一个  $n \times m$  的方格图，每个方格中都有一个正整数。

现要从方格中取数，使任意两个数所在方格没有公共边，且取出的数的总和最大，请求出最大的和。

#### 题解

由于每个方格限制较少，故考虑利用限制建边。

考虑先取所有方格，再删去权值和最小的一组方格，使得剩下的方格没有公共边。

问题转化为建立一个模型，支持以下操作：

- 支持删除元素，且删除代价为元素点权
- 保证操作最优或者操作可逆
- 最终状态为剩余元素没有冲突

发现可以将方格进行黑白二染色，显然同色方格间没有限制不连边，所以可以得到二分图。

从源点向每个黑格连一条边，容量为黑格点权。如果删去该边，表示删除黑格。

从白格向汇点连一条边，容量为白格点权。如果删去该边，表示删除白格。

最后从黑格向有冲突的白格连一条边，容量待定。

问题转化为最小割问题，事实上图不连通等价于没有从源点到黑格，黑格经过冲突边到白格，白格再到汇点的路径。

这又等价于不会同时选择冲突的黑格和白格，即最终的目的。

然后需要保证最小割只选择源点到黑格的边和白格到汇点的边，所以把黑格向有冲突的白格连的边容量设为  $\infty$

最后求最小割，答案为点权总和减去最小割。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
```

```
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n' ||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXS=105,MAXN=MAXS*MAXS,MAXM=6*MAXN,Inf=0x7fffffff;
const int dir[][2]={{0,1},{0,-1},{1,0},{-1,0}};
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;
int n,m,a[MAXS][MAXS];
int Id(int r,int c){
return (r-1)*m+c;
}
bool in(int r,int c){return r>0&&r<=n&&c>0&&c<=m;}
int main()
{
n=read_int(),m=read_int();
int sum=0,s=Id(n,m)+1,t=s+1;
Clear();
_rep(i,1,n)
_rep(j,1,m){
a[i][j]=read_int(),sum+=a[i][j];
if((i+j)&1)
Insert(Id(i,j),t,a[i][j]);
else{
Insert(s,Id(i,j),a[i][j]);
_for(k,0,4){
int ti=i+dir[k][0],tj=j+dir[k][1];
if(in(ti,tj))
Insert(Id(i,j),Id(ti,tj),Inf);
}
}
}
sum-=solver.Maxflow(s,t);
enter(sum);
return 0;
}
```

## 最长 $k$ 可重区间集问题

[洛谷p3358](#)

### 题意

给定  $n$  个开区间，要求选取若干个区间，使得  $X$  轴上每个点最多被覆盖  $k$  次。

问在所有可能方案中，选取区间长度和最大值为多少。

## 题解

很自然想到每个点最多覆盖  $k$  次可以理解为  $X$  轴上流量最大为  $k$  而长度和最大可以理解为费用流。

首先考虑到费用流没有点权，考虑将一个区间拆成左端点和右端点，两端点间连一条容量为  $1$ ，费用为区间长度相反数的边。

没有相交的区间不存在约束，考虑在靠左区间的右端点向靠右区间的左端点间连一条容量无穷大且费用为  $0$  的边，表示可以同时选择。

而有相加区间的之间则不连边，使得两个区间竞争来自源点的流量，相互约束。

然而按上述方案连边，边的数量可能达到  $O(n^2)$  难以接受。

考虑将点离散化，分配到  $X$  轴，在  $X$  轴所有相邻点间连一条容量为无穷大(其实  $k$  就够了)，费用为  $0$  的边。

对每个区间，仍然按上述方案连边。最后从源点连一条容量为  $k$  的边到最左端点，从最右端点连一条容量为  $k$  的边到汇点，跑费用流即可。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
```

```
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n' ||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=1005,MAXM=2005,Inf=0x7fffffff;
struct Edge{
    int to,cap,w,next;
    Edge(int to=0,int cap=0,int w=0,int next=0){
        this->to=to;
        this->cap=cap;
        this->w=w;
        this->next=next;
    }
}edge[MAXN<<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 w,int c){
    edge[edge_cnt]=Edge(v,c,w,head[u]);
    head[u]=edge_cnt++;
    edge[edge_cnt]=Edge(u,0,-w,head[v]);
    head[v]=edge_cnt++;
}
struct Dinic{
    int s,t;
    int pos[MAXN],dis[MAXN];
    bool inque[MAXN];
    bool spfa(){
        mem(dis,127/3);
        queue<int>q;
        q.push(s);
        inque[s]=true,dis[s]=0,pos[s]=head[s];
        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[u]+edge[i].w<dis[v]&&edge[i].cap){
                    dis[v]=dis[u]+edge[i].w,pos[v]=head[v];
                    if(!inque[v]){
```

```

        q.push(v);
        inque[v]=true;
    }
}
}
return dis[t]!=dis[0];
}
int dfs(int u,int max_flow){
    if(u==t||!max_flow)
        return max_flow;
    int flow=0,temp_flow;
    inque[u]=true;
    for(int &i=pos[u];~i;i=edge[i].next){
        int v=edge[i].to;
        if(!inque[v]&&dis[u]+edge[i].w==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;
        }
    }
    inque[u]=false;
    return flow;
}
void MCMF(int s,int t,int &flow,LL &cost){
    this->s=s;this->t=t;
    cost=flow=0;
    int temp_flow;
    mem(inque,0);
    while(spfa()){
        temp_flow=dfs(s,Inf);
        flow+=temp_flow;
        cost+=1LL*temp_flow*dis[t];
    }
}
}solver;
int x[MAXN],lef[MAXN],rig[MAXN],len[MAXN];
int main()
{
    Clear();
    int n=read_int(),k=read_int(),m;
    _for(i,0,n){
        lef[i]=read_int(),rig[i]=read_int(),len[i]=rig[i]-lef[i];
        x[m++]=lef[i],x[m++]=rig[i];
    }
    sort(x,x+m);
    m=unique(x,x+m)-x;
}

```

```
int s=m+1,t=m+2;
_for(i,1,m)
Insert(i,i+1,0,k);
Insert(s,1,0,k);Insert(m,t,0,k);
_for(i,0,n){
    lef[i]=lower_bound(x,x+m,lef[i])-x+1;
    rig[i]=lower_bound(x,x+m,rig[i])-x+1;
    Insert(lef[i],rig[i],-len[i],1);
}
int flow;LL cost;
solver.MCMF(s,t,flow,cost);
enter(-cost);
return 0;
}
```

## 拓展

如果题目把开区间改成闭区间，拆点的时候当作  $[\text{lef}, \text{rig}+1)$  处理就行。

## 最小路径覆盖问题

[洛谷p2764](#)

### 题意

给定一个  $\text{DAG}$  求最小的不相交路径集，使得每个点恰好属于其中一条路径。（允许路径长度为  $0$ ，此时路径仅覆盖一个点）

### 题解

先用  $n$  条长度为  $0$  的路径覆盖  $n$  个点，然后考虑合并路径。

将每个点拆成入点和出点，易知每个入点/出点最多被合并一次，否则有两条路径相交。

从源点连一条容量为  $1$  的边到每个入点，从每个出点连一条容量为  $1$  的边到汇点，则可以满足上述的合并次数限制。

最后，对每条边，从入点连一条容量为  $1$  的边到出点，表示以入点结束的路径可与以出点开始的路径合并一次。

最后需要合并次数最多，跑最大流即可。最小路径覆盖为节点数减去最大流。

关于路径打印，只要在求完最大流后跑残量网络即可。

```
#include <iostream>
#include <cstdio>
```

```

#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=305,MAXM=18005,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;
int Next[MAXN],Pre[MAXN];
int main()
{
    int n=read_int(),m=read_int(),u,v;
    int s=2*n+1,t=2*n+2;
    Clear();
    _rep(i,1,n){
        Insert(s,i,1);
        Insert(i+n,t,1);
    }
    while(m--){
        u=read_int(),v=read_int();
        Insert(u,v+n,1);
    }
    int flow=solver.Maxflow(s,t);
    _rep(u,1,n){
        for(int i=head[u];~i;i=edge[i].next){
            int v=edge[i].to;
            if(v!=s&&edge[i].cap==0){
                Next[u]=v-n;
                Pre[v-n]=u;
            }
        }
    }
    _rep(i,1,n){
        if(!Pre[i]){
            int pos=i;
            while(pos){
                space(pos);
                pos=Next[pos];
            }
            puts("");
        }
    }
    enter(n-flow);
    return 0;
}

```

## 最长不下降子序列问题

[洛谷p2766](#)

## 题意

给定一个正整数序列  $x_1, x_2, \dots, x_n$

1. 计算其最长不下降子序列的长度  $s$
2. 如果每个元素只允许使用一次，计算从给定的序列中最多可取出多少个长度为  $s$  的不下降子序列
3. 如果允许在取出的序列中多次使用  $x_1, x_n$  则从给定序列中最多可取出多少个不同的长度为  $s$  的不下降子序列

## 题解

第一问考虑  $dp$  求出以第  $i$  个数结尾的最长序列长度。

第二问先考虑只允许使用一次这个限制条件，只需要把每个点拆成入点和出点，然后连一条容量为  $1$  的边即可。

接下若满足  $a_j \leq a_i, dp_{j+1} = dp_i$  则连一条  $j \rightarrow i$  的容量为  $1$  的边。

最后考虑从源点向  $dp_i = 1$  的点连一条容量为  $1$  的边，从  $dp_i = s$  的点向汇点连一条容量为  $1$  的边。

最后跑最大流即可求出第二问答案。

对第三问，考虑把第  $1, n$  个点的入点和出点的连边容量改为  $\infty$  同时把从源点到第  $1$  个的连边的容量改为  $\infty$

如果  $dp_n = s$  还需要把从第  $n$  个点向汇点的连边的容量改为  $\infty$  然后重新跑一遍最大流。

但实际编程中不需要修改边，额外添加边即可。同时也不需要重新跑最大流，在残量网络上跑最大流然后累加上第二问答案即可。

需要注意  $s=1$  的特例，题目要求不同的不下降子序列就是为了防止此时答案为  $\infty$  的情况。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
```

```

#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n' ||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=1005,MAXM=5e5+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;}//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;
int a[MAXN],dp[MAXN],f[MAXN],pos;
int main()
{
    int n=read_int();
    _rep(i,1,n)
        a[i]=read_int();
    f[pos=0]=-Inf;
}
```

```

_rep(i,1,n){
    if(a[i]>=f[pos]){
        f[++pos]=a[i];
        dp[i]=pos;
    }
    else{
        int t=upper_bound(f,f+pos+1,a[i])-f;
        f[t]=a[i];
        dp[i]=t;
    }
}
enter(pos);
if(pos==1){
    sort(a+1,a+n+1);
    int ans=unique(a+1,a+n+1)-a-1;
    enter(ans);
    enter(ans);
    return 0;
}
int s=2*n+1,t=2*n+2;
Clear();
_rep(i,1,n){
    Insert(i,i+n,1);
    if(dp[i]==1)
        Insert(s,i,1);
    if(dp[i]==pos)
        Insert(i+n,t,1);
    _for(j,1,i){
        if(a[j]<=a[i]&&dp[j]+1==dp[i])
            Insert(j+n,i,1);
    }
}
int ans=solver.Maxflow(s,t);
enter(ans);
Insert(s,1,Inf);Insert(1,1+n,Inf);
if(dp[n]==pos){
    Insert(n,2*n,Inf);
    Insert(2*n,t,Inf);
}
ans+=solver.Maxflow(s,t);
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:%E5%9B%BE%E8%AE%BA\\_3&rev=1595303135](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E5%9B%BE%E8%AE%BA_3&rev=1595303135)

Last update: 2020/07/21 11:45