最小费用最大流

最小费用最大流（Minimum Cost Maximum Flow,MCMF）,是一般网络流模型的一种变型。每条边除了容量以外，还存在**单位费用**。在一条边上的费用就是单位时间×流量，现在要求满足最大流的同时，总费用最少的一种情况。

【模板】最小费用最大流

在最大流模型基础上给定每条边的单位费用，求最大流量以及最小费用。

```
#include<bits/stdc++.h>
//#define int long long
#define inf 0x3f3f3f3f
using namespace std;
const int N=5005,M=1e5+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

struct E{int v,w,f,nxt;}e[M];
int head[N],cur[N],cnt=1;
inline void addedge(int u,int v,int w,int f){e[++cnt]=(E)
{v,w,f,head[u]};head[u]=cnt;}

int n,m,s,t,mcost,mflow;
int dis[N],inq[N],pre[N],flow[N];

inline bool spfa(){
    memset(dis,inf,sizeof(dis));
    memset(inq,0,sizeof(inq));
    queue<int>q;
    q.push(s);
    inq[s]=1;dis[s]=0;flow[s]=inf;
    while(q.size()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
```

```
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w,f=e[i].f;
            if(w&&dis[to]>dis[fro]+f){
                dis[to]=dis[fro]+f;
                flow[to]=min(flow[fro],w);
                pre[to]=i;
                if(!inq[to]){
                    q.push(to);
                    inq[to]=1;
                }
            }
        }
    }
    return dis[t]!=inf;
}

inline void update(){
    int x=t;
    while(x!=s){
        int i=pre[x];
        e[i].w-=flow[t];
        e[i^1].w+=flow[t];
        x=e[i^1].v;
    }
    mflow+=flow[t];
    mcost+=flow[t]*dis[t];
}

int EK(){
    int ans=0;
    while(spfa()){
        update();
    }
    return ans;
}

signed main(){
    n=read();m=read();s=read();t=read();
    for(int i=1,u,v,w,f;i<=m;++i){
        u=read();v=read();w=read();f=read();
        addedge(u,v,w,f);
        addedge(v,u,0,-f);
    }
    EK();
    write(mflow);putchar(' ');write(mcost);
    return 0;
}
```

动态加点 P2050 [NOI2012] 美食节

对于每个菜建立一个点，源点向其连一条流量为需求量费用为 0 的边。

然后再建一层点，分别表示第j个厨师做第倒数i道菜。向汇点连一条流量为 1 费用为 0 的边。

假设有一个点表示第j个厨师做第倒数k道菜，那么对于菜i，向其连一条流量为 1 ，费用为k×a(i,j)的边。这表示第j个厨师做的倒数第k道菜是菜i，那么就要做a(i,j)这么长的时间，有k个人要等这么长的时间。

优化

由于此题数据量很大，把所有边连完后再跑费用流是一定会GG的（60分）

由于我们跑一次spfa只能找出一次增广路，所以我们可以暂时不连不需要的边。一开始，我们把所有厨师做倒数第 1 道菜与所有菜连好，然后找一条增广路，这条增广路上一定经过了一个点，表示第j个厨师做倒数第 1 道菜，于是我们添加点（第j个厨师做倒数第 2 道菜），与汇点和所有菜连边，以此类推。

```
#include<bits/stdc++.h>
//#define int long long
#define inf 0x3f3f3f3f
using namespace std;
const int N=1e5+7,M=2e6+7;

inline int read(){
    int x=0;
    char ch=getchar();
    while(ch<'0'||ch>'9') ch=getchar();
    while(ch>='0'&&ch<='9') x=x*10+ch-'0',ch=getchar();
    return x;
}

inline void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

struct E{
    int v,w,f,nxt;
}e[M];
int head[N],cnt=1;
inline void addedge(int u,int v,int w,int f){
    e[++cnt]={v,w,f,head[u]};head[u]=cnt;
    e[++cnt]={u,0,-f,head[v]};head[v]=cnt;
}

int n,m,s,t,sum=0,p[N],mp[50][105],dish[N],cook[N];
int mcost,mflow;
int dis[N],inq[N],pre[N],flow[N];

inline bool spfa(){
    for(int i=s;i<=t;++i) dis[i]=inf,inq[i]=0;
    queue<int>q;
    q.emplace(s);
    inq[s]=1;dis[s]=0;flow[s]=inf;
    while(q.size()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w,f=e[i].f;
            if(w&&dis[to]>dis[fro]+f){
                dis[to]=dis[fro]+f;
                flow[to]=min(flow[fro],w);
                pre[to]=i;
                if(!inq[to]){
                    q.emplace(to);
                    inq[to]=1;
                }
            }
        }
```

```
    }
    return dis[t]!=inf;
}

inline void update(){
    int x=t;
    while(x!=s){
        int i=pre[x];
        e[i].w-=flow[t];
        e[i^1].w+=flow[t];
        x=e[i^1].v;
    }
    mflow+=flow[t];
    mcost+=flow[t]*dis[t];
}

void EK(){
    while(spfa()){
        update();
        int tmp=e[pre[t]^1].v;
        addedge(tmp+1,t,1,0);
        for(int i=1;i<=n;++i){
            addedge(i+m*sum,tmp+1,1,mp[i][cook[tmp]]*(dish[tmp]+1));
        }
    }
}

signed main(){
    n=read();m=read();
    for(int i=1;i<=n;++i) p[i]=read(),sum+=p[i];
    s=0;t=n+sum*m+1;
    for(int i=1;i<=n;++i) addedge(s,i+sum*m,p[i],0);
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            mp[i][j]=read();
            addedge(i+sum*m,(j-1)*sum+1,1,mp[i][j]);
        }
    }
    for(int i=1;i<=m;++i) addedge((i-1)*sum+1,t,1,0); //预先加上所有厨师处理倒数第一道
菜
    for(int i=1;i<=m;++i){
        for(int j=1;j<=sum;++j){
            int tmp=(i-1)*sum+j;
            dish[tmp]=j;cook[tmp]=i;
        }
    }
    EK();
    //write(mflow);putchar(' ');write(mcost);
    write(mcost);
}
```

[SDOI2010]星际竞速

给定瞬移到星球i的价值a_i，和m条带权边，问所有星球都走过一遍的最小代价。

- $s \to 1_x \sim n_x$,流量1，费用0
- $s \to 1_y \sim n_y$,流量1，费用a_i
- $1_y \sim n_y \to t$,流量1，费用0
- $u_x \sim v_y$,流量1，费用w

```
#include<bits/stdc++.h>
//#define int long long
#define inf 0x3f3f3f3f
using namespace std;
const int N=5005,M=1e5+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(int x){if(x>9) write(x/10);putchar(x%10+'0');}

struct E{int v,w,f,nxt;}e[M];
int head[N],cur[N],cnt=1;
inline void addedge(int u,int v,int w,int f){
    e[++cnt]=(E){v,w,f,head[u]};head[u]=cnt;
    e[++cnt]=(E){u,0,-f,head[v]};head[v]=cnt;
}

int n,m,s,t,mcost,mflow,a[N];
int dis[N],inq[N],pre[N],flow[N];

inline bool spfa(){
    memset(dis,inf,sizeof(dis));
    memset(inq,0,sizeof(inq));
    queue<int>q;
    q.push(s);
    inq[s]=1;dis[s]=0;flow[s]=inf;
    while(q.size()){
        int fro=q.front();
        q.pop();
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w,f=e[i].f;
            if(w&&dis[to]>dis[fro]+f){
                dis[to]=dis[fro]+f;
                flow[to]=min(flow[fro],w);
                pre[to]=i;
                if(!inq[to]){
                    q.push(to);
                    inq[to]=1;
                }
            }
        }
    }
    return dis[t]!=inf;
}

inline void update(){
```

```
    int x=t;
    while(x!=s){
        int i=pre[x];
        e[i].w-=flow[t];
        e[i^1].w+=flow[t];
        x=e[i^1].v;
    }
    mflow+=flow[t];
    mcost+=flow[t]*dis[t];
}

int EK(){
    int ans=0;
    while(spfa()){
        update();
    }
    return ans;
}

signed main(){
    n=read();m=read();
    s=0;t=2*n+2;
    for(int i=1;i<=n;++i) a[i]=read();
    for(int i=1;i<=n;++i)
addedge(s,i,1,0),addedge(s,i+n,1,a[i]),addedge(n+i,t,1,0);
    for(int i=1,u,v,w,f;i<=m;++i){
        u=read();v=read();f=read();
        if(u>v) swap(u,v);
        addedge(u,n+v,1,f);
    }
    EK();
    //write(mflow);putchar(' ');
    write(mcost);
    return 0;
}
```

H. Hamilton - The Musical

给定一个排列以及每两个数之间的路程，偶数位固定，重排奇数位使得从1到n的路程和最小。卡了很久的常数。

```
#pragma GCC optimize("Ofast", "inline", "-ffast-math")
#pragma GCC target("avx,sse2,sse3,sse4,mmx")
#include<bits/stdc++.h>
#define inf 0x7f7f7f7f
using namespace std;
const int N=805,M=6e5+7;
typedef long long ll;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}
void write(ll x){if(x>9) write(x/10);putchar(x%10+'0');}

//struct E{int v,w,f,nxt;}e[M];
```

```
int v[M],nxt[M];
bool flow[N],w[M];
ll f[M];

int head[N],cnt=1;
inline void addedge(int x,int y,bool z,ll k){
    ++cnt;v[cnt]=y;w[cnt]=z;f[cnt]=k;nxt[cnt]=head[x];head[x]=cnt;
    ++cnt;v[cnt]=x;w[cnt]=0;f[cnt]=-k;nxt[cnt]=head[y];head[y]=cnt;
}

int n,m,s,t;
ll dis[N],mcost,mflow;
int inq[N],pre[N];

int mp[505][505],tot=0;
int q[M*10];

inline bool spfa(){
    for(int i=s;i<=t;++i) dis[i]=inf,inq[i]=0;
    int L=1,R=1;q[L]=s;
    inq[s]=1;dis[s]=0;flow[s]=inf;
    while(L<=R){
        int fro=q[L];
        ++L;
        inq[fro]=0;
        for(int i=head[fro];i;i=nxt[i]){
            if(w[i]&&dis[v[i]]>dis[fro]+f[i]){
                dis[v[i]]=dis[fro]+f[i];
                flow[v[i]]=(flow[fro]&w[i]);
                pre[v[i]]=i;
                if(!inq[v[i]]){
                    q[++R]=v[i];
                    inq[v[i]]=1;
                }
            }
        }
    }
    return dis[t]!=inf;
}

inline void update(){
    int x=t;
    while(x!=s){
        int i=pre[x];
        w[i]-=flow[t];
        w[i^1]+=flow[t];
        x=v[i^1];
    }
    mflow+=flow[t];
    mcost+=flow[t]*dis[t];
}

void EK(){
    while(spfa()){
        update();
    }
}
signed main(){
```

```
    n=read();
    s=0;
    for(int i=1;i<=n;++i){
        for(int j=1;j<=n;++j){
            mp[i][j]=read();
        }
    }
    for(int i=2;i+2<=n;i+=2){
        addedge(s,i+2,1,0);
        for(int j=1;j<=n;j+=2) addedge(i+2,j,1,mp[i][j]+mp[j][i+2]);
    }
    addedge(s,2,1,0);
    for(int j=1;j<=n;j+=2) addedge(2,j,1,mp[2][j]);
    if((n&1)&&n>4){
        addedge(s,n+1,1,0);
        for(int j=1;j<=n;j+=2) addedge(n+1,j,1,mp[n-1][j]);
    }
    t=n+3;
    for(int i=1;i<=n;i+=2) addedge(i,t,1,0);
    EK();
    printf("%lld\n",mcost);
    return 0;
}
```

ZKW费用流

大部分情况下都会比EK要快。

```
//P3381 【模板】最小费用最大流
#include <bits/stdc++.h>
using namespace std;
const int V=1e6;
const int M=3e6;
const int inf=0x3f3f3f3f;
int n,m,s,t,p,fans,cans;
class Graph{
public:
    int top,to[M<<1],fw[M<<1],ct[M<<1];
    vector<int> g[V];
    Graph(){top=1;}
    void add(int x,int y,int f,int c){
        g[x].push_back(++top);
        to[top]=y,fw[top]=f,ct[top]=c;
    }
    void Add(int x,int y,int f,int c){
        add(x,y,f,c),add(y,x,0,-c);
    }
};
class zkwMCMF:public Graph{
public:
    int dep[V]; bool vis[V];
    deque<int> q;
    bool spfa(){
```

```
        for(int i=1;i<=p;i++) vis[i]=0,dep[i]=inf;
        q.push_back(t),vis[t]=1,dep[t]=0;
        while(q.size()){
            int x=q.front();q.pop_front(),vis[x]=0;
            for(auto i:g[x])if(fw[i^1]&&dep[to[i]]>dep[x]-ct[i]){
                dep[to[i]]=dep[x]-ct[i];
                if(!vis[to[i]]){
                    vis[to[i]]=1;
                    if(q.size()&&dep[to[i]]<dep[q.front()])
                        q.push_front(to[i]);
                    else q.push_back(to[i]);
                }
            }
        }
        return dep[s]<inf;
    }
    int dfs(int x,int F){
        vis[x]=1;
        if(x==t||!F) return F;
        int f,flow=0;
        for(auto i:g[x])if(!vis[to[i]]&&fw[i]&&dep[x]-ct[i]
        ==dep[to[i]]&&(f=dfs(to[i],min(F,fw[i])))>0){
            cans+=f*ct[i],fw[i]-=f,fw[i^1]+=f,flow+=f,F-=f;
            if(!F) break;
        }
        return flow;
    }
    void mcmf(){
        while(spfa()){
            vis[t]=1;
            while(vis[t]){
                for(int i=1;i<=p;i++) vis[i]=0;
                fans+=dfs(s,inf);
            }
        }
    }
}network;
int main(){
    scanf("%d%d%d%d",&n,&m,&s,&t),p=n;
    for(int i=1,x,y,f,c;i<=m;i++){
        scanf("%d%d%d%d",&x,&y,&f,&c);
        network.Add(x,y,f,c);
    }
    network.mcmf();
    printf("%d %d\n",fans,cans);
    return 0;
}
```

餐巾计划问题

餐馆给定n天内每天需要的餐巾数，并给出购买餐巾的费用p,慢洗的天数$t1$和费用$f1$以及快洗的天数$t2$和费用$f2$，求最小花费。

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
const int V=1e6;
const int M=3e6;
const int inf=0x3f3f3f3f;
typedef long long ll;
int n,m,s,t,need[V];
ll fans,cans;
class Graph{
public:
    int top,to[M<<1],fw[M<<1],ct[M<<1];
    vector<int> g[V];
    Graph(){top=1;}
    void add(int x,int y,int f,int c){
        g[x].push_back(++top);
        to[top]=y,fw[top]=f,ct[top]=c;
    }
    void Add(int x,int y,int f,int c){
        add(x,y,f,c),add(y,x,0,-c);
    }
};
class zkwMCMF:public Graph{
public:
    int dep[V]; bool vis[V];
    deque<int> q;
    bool spfa(){
        for(int i=s;i<=t;i++) vis[i]=0,dep[i]=inf;
        q.push_back(t),vis[t]=1,dep[t]=0;
        while(q.size()){
            int x=q.front();q.pop_front(),vis[x]=0;
            for(auto i:g[x])if(fw[i^1]&&dep[to[i]]>dep[x]-ct[i]){
                dep[to[i]]=dep[x]-ct[i];
                if(!vis[to[i]]){
                    vis[to[i]]=1;
                    if(q.size()&&dep[to[i]]<dep[q.front()])
                        q.push_front(to[i]);
                    else q.push_back(to[i]);
                }
            }
        }
        return dep[s]<inf;
    }
    ll dfs(int x,int F){
        vis[x]=1;
        if(x==t||!F) return F;
        ll f,flow=0;
        for(auto i:g[x])if(!vis[to[i]]&&fw[i]&&dep[x]-ct[i]
        ==dep[to[i]]&&(f=dfs(to[i],min(F,fw[i])))>0){
            cans+=f*ct[i],fw[i]-=f,fw[i^1]+=f,flow+=f,F-=f;
            if(!F) break;
        }
        return flow;
    }
    void mcmf(){
        while(spfa()){
            vis[t]=1;
            while(vis[t]){
                for(int i=s;i<=t;i++) vis[i]=0;
                fans+=dfs(s,inf);
```

```
            }
        }
    }
}network;

signed main(){
    scanf("%d",&n);
    s=0;t=2*n+1;
    int p,t1,t2,f1,f2;
    for(int i=1;i<=n;++i) scanf("%d",&need[i]);
    scanf("%d%d%d%d%d",&p,&t1,&f1,&t2,&f2);
    for(int i=1;i<=n;i++){
        network.Add(s,i,need[i],0);  //源点向每一天送洗的餐巾连边
        network.Add(i+n,t,need[i],0); //每天使用的餐巾向汇点连边
        network.Add(s,i+n,inf,p); //购买餐巾连边
        if(i+1<=n) network.Add(i,i+1,inf,0);  //延迟购买连边
        if(i+t1<=n) network.Add(i,i+n+t1,inf,f1);  //慢洗连边
        if(i+t2<=n) network.Add(i,i+n+t2,inf,f2);  //快洗连边
    }
    network.mcmf();
    printf("%lld\n",cans);
    return 0;
}
```

Codeforces #170 Div1 E Binary Tree on Plane

给你平面上 n 个点 (2≤n≤400)，要求用这些点组成一个二叉树(每个节点的儿子节点不超过两个)，定义每条边的权值为两个点之间的欧几里得距离。求一个权值和最小的二叉树，并输出这个权值。其中，点 i 可以成为点 j 的的父亲的条件是：点 i 的 y 坐标比 j 的 y 坐标大。

```
#include<bits/stdc++.h>
#define int long long
#define inf 1e12
using namespace std;
const int N=1005,M=4e5+7;

struct E{int v,w,nxt;double f;}e[M];
int head[N],cur[N],cnt=1;
inline void addedge(int u,int v,int w,double f){
    e[++cnt]=(E){v,w,head[u],f};head[u]=cnt;
    e[++cnt]=(E){u,0,head[v],-f};head[v]=cnt;
}

int n,m,s,t;
double mcost,mflow,dis[N];
int inq[N],pre[N],flow[N];

inline bool spfa(){
    for(int i=s;i<=t;++i) dis[i]=inf,inq[i]=0;
    queue<int>q;
    q.push(s);
    inq[s]=1;dis[s]=0;flow[s]=inf;
    while(q.size()){
        int fro=q.front();
        q.pop();
```

```
        inq[fro]=0;
        for(int i=head[fro];i;i=e[i].nxt){
            int to=e[i].v,w=e[i].w;double f=e[i].f;
            if(w&&dis[to]>dis[fro]+f){
                dis[to]=dis[fro]+f;
                flow[to]=min(flow[fro],w);
                pre[to]=i;
                if(!inq[to]){
                    q.push(to);
                    inq[to]=1;
                }
            }
        }
    }
    return dis[t]!=inf;
}

inline void update(){
    int x=t;
    while(x!=s){
        int i=pre[x];
        e[i].w-=flow[t];
        e[i^1].w+=flow[t];
        x=e[i^1].v;
    }
    mflow+=flow[t];
    mcost+=flow[t]*dis[t];
}

double EK(){
    double ans=0;
    while(spfa()){
        update();
    }
    return ans;
}

struct Nod{
    double x,y;
    inline operator <(const Nod& B)const{
        return y==B.y?x<B.x:y>B.y;
    }
}pt[N];
double distance(Nod A,Nod B){
    return sqrt(1.0*(A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));
}

signed main(){
    scanf("%lld",&n);
    s=0;t=2*n+1;
    for(int i=1;i<=n;++i) scanf("%lf%lf",&pt[i].x,&pt[i].y);
    stable_sort(pt+1,pt+1+n); //按y排序
    if(pt[1].y==pt[2].y){ //没有唯一的根
        printf("-1\n");
        return 0;
    }
    for(int i=1;i<=n;++i) addedge(s,i+n,2,0); //最多拥有2个子节点
    for(int i=2;i<=n;++i) addedge(i,t,1,0); //做根的点向汇点连边
```

```
    for(int i=1;i<=n;++i){
        for(int j=i+1;j<=n;++j){  //做根的点向下连边
            if(pt[i].y>pt[j].y) addedge(i+n,j,1,distance(pt[i],pt[j]));
        }
    }
    EK();
    if(mflow==n-1) printf("%.10lf",mcost);
    else puts("-1"); //跑不满流
    return 0;
}
```

参考资料

https://zhuanlan.zhihu.com/p/127046673