最小代价流

常见的图模型

一、最短路径

单源单汇问题

$$min = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_{ij}$$

$$s.t \begin{cases} \sum_{j=1}^{N} x_{ij} - \sum_{k=1}^{N} x_{ij} = \begin{cases} 1, i = 1 \\ -1, i = N \\ 0, i \neq 1, N \end{cases} \\ x_{ij} = 0 \ or \ 1 \end{cases}$$

选择的路径只有每个节点其流入边和流出边一定有且只有0或1个，对于起点和终点则允许有一条流出边或一条流入边。

二、最大流问题

最大流问题就是选择从起点到终点的最大流量分配，与最短路的最大区别在于最短路问题中每个节点只能选择一条流出边和一条流入边，而最大流问题则只要满足边容量限制，则可任意选择流入流出边数量。

$$max = v(f) = \sum_{j} f_{sj} - \sum_{i} f_{is}$$

$$s.t. \sum_{j} fij = \sum_{k} f_{ki} r \sum_{j} f_{sj} = \sum_{k} f_{kt} = v(f) r \ 0 \leq f_{ij} \leq w_{ij}$$

中间节点无论会不会通过，其入边流量之和 = 流出边流量之和，从起点流出的总流量 = 流入终点的总流量，每条边的流量有上限。

学习算法：**Edmond-Karp算法**、**Dinic算法**.....

三、最小费用问题

最小费用的约束条件和最大流的一样，目标是选择费用最短的流。但与最大流问题不同，需要设置起始点的流出流量，如果在最大流限制下求解最小费用流，就是最小费用最大流问题（**是毕设中所涉及到的问题**）

$$min = \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}x_{ij}$$

$$s.t \begin{cases} \sum_{j=1}^{N} x_{ij} - \sum_{k=1}^{N} x_{ij} = \begin{cases} > 0, i = 1 \\ < 0, i = N \\ = 0, i \neq 1, N \end{cases} \\ 0 \leq x_{ij} \leq w_{ij} \end{cases}$$

联系到多目标跟踪任务，其数据关联任务从短期来看就是一个二分图匹配问题，从长期来看就是一个图网络模型

- 如果使用最短路模型描述数据关联问题，节点就是跟踪对象id，边代表跟踪轨迹和检测之间的代价，不足以描述该问题，因为跟踪轨迹和检测数量是大于 1 的，所以从形式上是多元多汇最短路径。但是最短路径没有限制中间节点的可重复性，所以应该用路由问题中 K — 最短不相交路线描述该问题。
- 最大流模型描述该问题，不同于最短路，边容量代表跟踪轨迹和检测之间的链接可能性，所以是 0 和 1。最终要求是最大流量，由于边容量的限制，不可能重复，也就是最多可能轨迹。
- 最小费用问题来描述该问题，从多源多汇问题变成给定出事流量的情形，距离变成了费用。和最短路径的区别在于需要合理设定出事流量（代表最终有多少条轨迹），还要设定边容量，防止所有流都流向同一条边。
- 最大流模型只需要设定跟踪轨迹和检测的连接可能性，但是缺乏了相对性。而最小费用流则只需要设定代价值，但是需要设定初始流量。这里的初始流量代表了轨迹数量，所以先用最大流模型求出最大流，即可作为初始的轨迹数量，然后再求最小费用流即可，也就是**最小费用最大流**。不过两个任务都有着相同的任务，那就是寻找目标轨迹，所以这样来说时间效率会较低。一般来说我们会直接使用最大流模型/最小割模型，或者直接使用最小费用流+搜索算法。

最小费用最大流算法过程

1. 求出从发点到收点的最小费用通路$\mu(s,t)$
2. 对该通路$\mu(s,t)$分配最大可能的流量：$\overline{f}=min_{(i,j)\in\mu(s,t)}\{c(i,j)\}$并让通路上所有边的容量相应减少了$\overline{f}$。这时，对于通路上的饱和边，其单位流费用相应改为$\infty$
3. 作该通路$\mu(s,t)$上所有边 (i,j) 的反向边 (j,i) ,令 $c(j,i)=\overline{f},d(j,i)=-d(i,j)$ 。
4. 在这样构成的新网络中，重复上述步骤 $1,2,3$ 直到从发点到收点的全部流量等于 f_v 为止（或者再也找不到从 s 到 t 的最小费用通路）

算法练习

题目描述： 有 n 个景点，一个人要从 1 号景点走到 n 号景点，再从 n 号景点走到 1 号（回来的路不能重复，不一定走完所有景点，只要求从 1 到 n 即可），给你一些景点之间的路的长度（双向），问你最短需要走多少路才能回来?

解法：

最小费用就是路径长度的总和，最大流就是来回的 两条路。
由于去和回来可以看成：2条从1到n的不同的路。 所以转化成求从1到n的两条不同的路。 假设a b之间有长度为c的路。按照最小费用流建图：

- ab 之间费用为 c，容量是 1
- ba 之间费用为 c，容量是 1
- 建立一个源点，连接1号景点，无费用，容量为 2（表示可以有两条路）
- 同理，建立一个汇点，连接n号景点，无费用，容量为 2

这样，如果求的的最大流是 2 ，就表示了有两条从 1 到 n 的不同的路。（因为中间的点边容量只是 1，只能用一次），所求的最小费用最大流的最小费用就是最短路径长度。

代码实现：

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

```cpp
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
using namespace std;
const int inf=1e10;
const int N=1e6;
struct node{int y,v,w,n;}e[N];
int lin[N],d[N],v[N],incf[N],pre[N],len=1,S,T,n,m,x,y,w,ans=0;
void read(int x,int y,int v,int w)
{e[++len].y=y,e[len].v=v,e[len].w=w,e[len].n=lin[x],lin[x]=len;}
void add(int x,int y,int v,int w)
{read(x,y,v,w),read(y,x,0,-w);}
bool SPFA(int S){
    memset(incf,0,sizeof(incf));
    memset(d,0x3f,sizeof(d));
    memset(v,0,sizeof(v));
    queue<int> q; q.push(S); d[S]=0; incf[S]=inf; v[S]=1;
    while(q.size()){
        int x=q.front();q.pop();v[x]=0;
        for(int i=lin[x];i;i=e[i].n){
            int y=e[i].y;
            if(!e[i].v)continue;
            if(d[y]>d[x]+e[i].w){
                d[y]=d[x]+e[i].w;
                incf[y]=min(incf[x],e[i].v);
                pre[y]=i;
                if(!v[y])q.push(y),v[y]=1;
            }
        }
    }
    return incf[n]>0;
}
void upd(int S){
    int x=n;
    while(x!=S){
        int i=pre[x];
        e[i].v-=incf[n];
        e[i^1].v+=incf[n];
        x=e[i^1].y;
    }
    ans+=incf[n]*d[n];
}
int main()
{
    scanf("%d%d",&n,&m); S=n+1,T=S+1;
    rep(i,1,m){scanf("%d%d%d",&x,&y,&w);add(x,y,1,w),add(y,x,1,w);}
    add(S,1,2,0);
    while(SPFA(S))upd(S);
    printf("%d",ans);
    return 0;
}
```