网络流

1.最大流

最大流最小割定理：

1. 最小割等价于最大流。
2. 最小割在最大流中一定是满流边，是增广路径中容量最小的边。
3. 一条增广路径只对应一条最小割。（如果一条增广路中两条满流且都需要割掉，那一定通过反向边分成两条增广路）

EK算法(使用 longlong+处理重边)

```
#include <bits/stdc++.h>
using namespace std;
int n,m,s,t,u,v;
long long w,ans,dis[520010];
int tot=1,vis[520010],pre[520010],head[520010],flag[2510][2510];

struct node {
        int to,net;
        long long val;
} e[520010];

void add(int u,int v,long long w) {
        e[++tot].to=v;
        e[tot].val=w;
        e[tot].net=head[u];
        head[u]=tot;
        e[++tot].to=u;
        e[tot].val=0;
        e[tot].net=head[v];
        head[v]=tot;
}

inline int bfs() {  //bfs寻找增广路
        for(int i=1;i< =n;i++) vis[i]=0;
        queue<int> q;
        q.push(s);
        vis[s]=1;
        dis[s]=2005020600;
        while(!q.empty()) {
                int x=q.front();
                q.pop();
                for(int i=head[x];i;i=e[i].net) {
```

```
                        if(e[i].val==0) continue;   //我们只关心剩余流量>0的边
                        int v=e[i].to;
                        if(vis[v]==1) continue;   //这一条增广路没有访问过
                        dis[v]=min(dis[x],e[i].val);
                        pre[v]=i;   //记录前驱，方便修改边权
                        q.push(v);
                        vis[v]=1;
                        if(v==t) return 1;   //找到了一条增广路
                }
        }
        return 0;
}

void update() {   //更新所经过边的正向边权以及反向边权
        int x=t;
        while(x! =s) {
                int v=pre[x];
                e[v].val-=dis[t];
                e[v^1].val+=dis[t];
                x=e[v^1].to;
        }
        ans+=dis[t];    //累加每一条增广路经的最小流量值
}

int main() {
        scanf("%d%d%d%d",&n,&m,&s,&t);
        for(int i=1;i< =m;i++) {
                scanf("%d%d%lld",&u,&v,&w);
                if(flag[u][v]==0) {   //处理重边的操作（加上这个模板题就可以用Ek算法过了）
                        add(u,v,w);
                        flag[u][v]=tot;
                }
                else {
                        e[flag[u][v]-1].val+=w;
                }
        }
        while(bfs()≠0) {   //直到网络中不存在增广路
                update();
        }
        printf("%lld",ans);
        return 0;
}
```

Dinic(当前弧优化+剪枝)

```
#include <bits/stdc++.h>
using namespace std;
```

```
const long long inf=2005020600;
int n,m,s,t,u,v;
long long w,ans,dis[520010];
int tot=1,now[520010],head[520010];

struct node {
        int to,net;
        long long val;
} e[520010];

void add(int u,int v,long long w) {
        e[++tot].to=v;
        e[tot].val=w;
        e[tot].net=head[u];
        head[u]=tot;

        e[++tot].to=u;
        e[tot].val=0;
        e[tot].net=head[v];
        head[v]=tot;
}

int bfs() {  //在惨量网络中构造分层图
        for(int i=1;i< =n;i++) dis[i]=inf;
        queue<int> q;
        q.push(s);
        dis[s]=0;
        now[s]=head[s];
        while(!q.empty()) {
                int x=q.front();
                q.pop();
                for(int i=head[x];i;i=e[i].net) {
                        int v=e[i].to;
                        if(e[i].val>0&&dis[v]==inf) {
                                q.push(v);
                                now[v]=head[v];
                                dis[v]=dis[x]+1;
                                if(v==t) return 1;
                        }
                }
        }
        return 0;
}

int dfs(int x,long long sum) {  //sum是整条增广路对最大流的贡献
        if(x==t) return sum;
        long long k,res=0;  //k是当前最小的剩余容量
        for(int i=now[x];i&&sum;i=e[i].net) {
                now[x]=i;  //当前弧优化
                int v=e[i].to;
```

```
                if(e[i].val>0&&(dis[v]==dis[x]+1)) {
                        k=dfs(v,min(sum,e[i].val));
                        if(k==0) dis[v]=inf;  //剪枝，去掉增广完毕的点
                        e[i].val-=k;
                        e[i^1].val+=k;
                        res+=k;  //res表示经过该点的所有流量和（相当于流出的总量）
                        sum-=k;  //sum表示经过该点的剩余流量
                }
        }
        return res;
}

int main() {
        scanf("%d%d%d%d",&n,&m,&s,&t);
        for(int i=1;i<=m;i++) {
                scanf("%d%d%lld",&u,&v,&w);
                add(u,v,w);
        }
        while(bfs()) {
                ans+=dfs(s,inf);  //流量守恒（流入=流出）
        }
        printf("%lld",ans);
        return 0;
}
```

2.最小费用最大流

```
#include <cstdio>
#include <queue>
#include <cstring>
using namespace std;
const int MAXN = 5001;
const int MAXM = 50001;
int n, m, s, t, edge_sum = 1;
int maxflow, mincost;
int dis[MAXN], head[MAXN], incf[MAXN], pre[MAXN];//dis表示最短路，incf表示当前增广路上最小流量，pre表示前驱
bool vis[MAXN];
struct Edge {
        int next, to, dis, flow;
}edge[MAXM << 1];
void addedge(int from, int to, int flow, int dis) {
        edge[++edge_sum].next = head[from];
        edge[edge_sum].to = to;
        edge[edge_sum].dis = dis;
        edge[edge_sum].flow = flow;
        head[from] = edge_sum;
}
```

```
inline bool spfa() {//关于SPFA，他诈尸了
        queue <int> q;
        memset(dis, 0x3f, sizeof(dis));
        memset(vis, 0, sizeof(vis));
        q.push(s);
        dis[s] = 0;
        vis[s] = 1;
        incf[s] = 1 << 30;
        while(!q.empty()) {
                int u = q.front();
                vis[u] = 0;
                q.pop();
                for(int i = head[u]; i; i = edge[i].next) {
                        if(!edge[i].flow) continue;//没有剩余流量
                        int v = edge[i].to;
                        if(dis[v] > dis[u] + edge[i].dis) {
                                dis[v] = dis[u] + edge[i].dis;
                                incf[v] = min(incf[u], edge[i].flow);//更新
incf
                                pre[v] = i;
                                if(!vis[v]) vis[v] = 1, q.push(v);
                        }
                }
        }
        if(dis[t] == 1061109567) return 0;
        return 1;
}
void MCMF() {
        while(spfa()) {//如果有增广路
                int x = t;
                maxflow += incf[t];
                mincost += dis[t] * incf[t];
                int i;
                while(x ! = s) {//遍历这条增广路，正向边减流反向边加流
                        i = pre[x];
                        edge[i].flow -= incf[t];
                        edge[i^1].flow += incf[t];
                        x = edge[i^1].to;
                }
        }
}
signed main() {
        scanf("%d%d%d%d", &n,&m,&s,&t);
        for(int u,v,w,x,i = 1; i < = m; ++i) {
                scanf("%d%d%d%d",&u,&v,&w,&x);
                addedge(u,v,w,x);
                addedge(v,u,0,-x);//反向边费用为-f[i]
        }
        MCMF();//最小费用最大流
        printf("%d %d\n",maxflow,mincost);
```

```
        return 0;
}
```