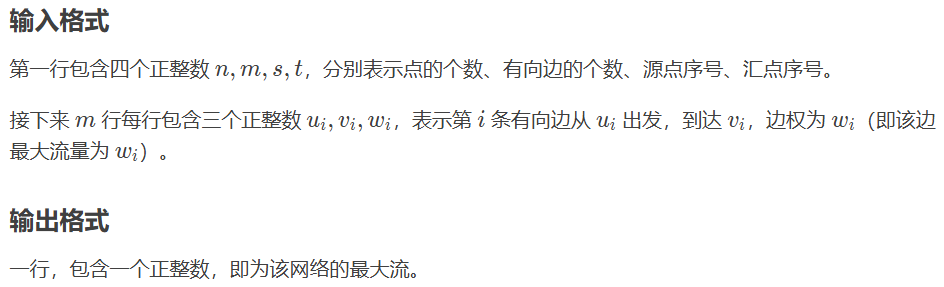
# P3376 【模板】网络最大流 修正P257 ISAP

****

#include <bits/stdc++.h>

typedef long long LL;

const int MAXN = 1200 + 10, MAXM = 120000+ 10;

const LL INF = 0x7f7f7f7f7f7f7f7f;

int n, m, s, t, dep[MAXN], gap[MAXN], cnt\_Edge = 1, Head[MAXM << 1], cur[MAXN];

struct node {int to; LL val; int Next;} Edge[MAXM << 1];

int q[MAXN], l, r;

LL ans;

int read()

{

    int sum = 0, fh = 1; char ch= getchar();

    for (; ch < '0' || ch > '9'; ch = getchar()) fh -= (ch == '-') << 1;

    for (; ch >= '0' && ch <= '9'; ch = getchar()) sum = (sum << 3) + (sum << 1) + (ch ^ 48);

    return (fh == 1) ? sum : -sum;

}

LL Min(LL fir, LL sec) {return (fir < sec) ? fir : sec;}

void add\_Edge(int x, int y, int z) {Edge[++cnt\_Edge] = (node){y, (LL)z, Head[x]}; Head[x] = cnt\_Edge;}

void bfs()

{

    q[l = r = 1] = t;

    memset(dep, -1, sizeof(dep)); dep[t] = 0; ++gap[0];

    //注意这地方 dep 初始化为 0 会出现一些奇奇怪怪的问题

    while (l <= r)

    {

        int x = q[l++];

        for (int i = Head[x]; i; i = Edge[i].Next)

        {

            int u = Edge[i].to;

            if (dep[u] != -1) continue ;

            dep[u] = dep[x] + 1; q[++r] = u; ++gap[dep[u]];

        }

    }

}

LL dfs(int now, LL Flow)

{

    if (now == t) return Flow;

    LL used = 0;

    for (int i = cur[now]; i; i = Edge[i].Next)

    {

        cur[now] = i; int u = Edge[i].to;

        if (Edge[i].val && dep[now] == dep[u] + 1)//注意控制层数

        {

            LL Minn = dfs(u, Min(Edge[i].val, Flow - used));

            if (Minn)

            {

                Edge[i].val -= Minn; Edge[i ^ 1].val += Minn; used += Minn;

                if (used == Flow) return used;

            }

        }

    }

    --gap[dep[now]];//提高层数

    if (gap[dep[now]] == 0) dep[s] = n + 1;//出现断层

    ++dep[now]; ++gap[dep[now]];

    return used;

}

int main()

{

    n = read(), m = read(), s = read(), t = read();

    for (int i = 1; i <= m; ++i)

    {

        int x = read(), y = read(), z = read();

        add\_Edge(x, y, z); add\_Edge(y, x, 0);

    }

    bfs();

while (dep[s] <= n) {for (int i = 1; i <= n; ++i) cur[i] = Head[i]; ans += dfs(s, INF);}

//出现断层就结束算法

    printf("%lld\n", ans);

    return 0;

}

# 超级优化HLPP

#include <bits/stdc++.h>

const int INF (INT\_MAX);

struct edge {

int to,flow,next;

edge(int to,int flow,int next):to(to),flow(flow),next(next){}

};

std::vector<edge>a[1203];

std::vector<int>list[1203],h,cnt,que,e;

typedef std::list<int> List;

std::vector<List::iterator>iter;

List dlist[1203];

typedef std::vector<edge>::iterator Iterator;

int hst,nowh;

inline void addEdge(const int u,const int v,const int f){

a[u].push\_back(edge(v,f,a[v].size()));

a[v].push\_back(edge(u,0,a[u].size()-1));

}

**inline** void relabel(int n,int t){

h.assign(n,n);h[t]=0;

cnt.assign(n,0);

que.clear();

que.resize(n+1);

int qh=0,qt=0;

for(que[qt++]=t;qh<qt;)

{

int u=que[qh++],het=h[u]+1;

for(Iterator p=a[u].begin();p!=a[u].end();++p)

{

if(h[p->to]==n&&a[p->to][p->next].flow>0)

{

cnt[h[p->to]=het]++;

que[qt++]=p->to;

}

}

}

for(register int i=0;i<=n;++i){list[i].clear();dlist[i].clear();}

for(register int u=0;u<n;++u)

{

if(h[u]<n)

{

iter[u]=dlist[h[u]].insert(dlist[h[u]].begin(),u);

if(e[u]>0)list[h[u]].push\_back(u);

}

}

hst=(nowh=h[que[qt-1]]);}

**inline** void push(int u,edge &ed){

int v=ed.to;

int df=std::min(e[u],ed.flow);ed.flow-=df;

a[v][ed.next].flow+=df;

e[u]-=df;e[v]+=df;

if(0<e[v]&&e[v]<=df)list[h[v]].push\_back(v);}

**inline** void push(int n,int u){

int nh=n;

for(Iterator p=a[u].begin();p!=a[u].end();++p){

if(p->flow>0){

if(h[u]==h[p->to]+1){push(u,\*p);if(e[u]==0)return;}

else nh=std::min(nh,h[p->to]+1);

}

}

int het=h[u];

if(cnt[het]==1){

for(register int i=het;i<=hst;++i){

for(List::iterator it=dlist[i].begin();it!=dlist[i].end();++it){cnt[h[\*it]]--;h[\*it]=n;}

dlist[i].clear();

}

hst=het-1;

}

else{

cnt[het]--;

iter[u]=dlist[het].erase(iter[u]);

h[u]=nh;

if(nh==n)return;

cnt[nh]++;

iter[u]=dlist[nh].insert(dlist[nh].begin(),u);

hst=std::max(hst,nowh=nh);

list[nh].push\_back(u);

}}

**inline** int hlpp(int n,int s,int t){

if(s==t)return 0;

nowh=0;hst=0;

h.assign(n,0);h[s]=n;

iter.resize(n);

for(register int i=0;i<n;++i)if(i!=s)iter[i]=dlist[h[i]].insert(dlist[h[i]].begin(),i);

cnt.assign(n,0);cnt[0]=n-1;

e.assign(n,0);e[s]=INF;e[t]=-INF;

for(register int i=0;i<(int)a[s].size();++i)push(s,a[s][i]);

relabel(n,t);

for(int u;nowh>=0;){

if(list[nowh].empty()){nowh--;continue;}

u=list[nowh].back();

list[nowh].pop\_back();

push(n,u);

}

return e[t]+INF;

}

**inline** int read(){

int f=0,fu=1;

char c=getchar();

while(c<'0'||c>'9'){if(c=='-')fu=-1;c=getchar();}

while(c>='0'&&c<='9'){f=(f<<3)+(f<<1)+c-48;c=getchar();}

return f\*fu;}

int n,m,s,t,u,v,f;

**signed** main(){

n=read(),m=read(),s=read(),t=read();

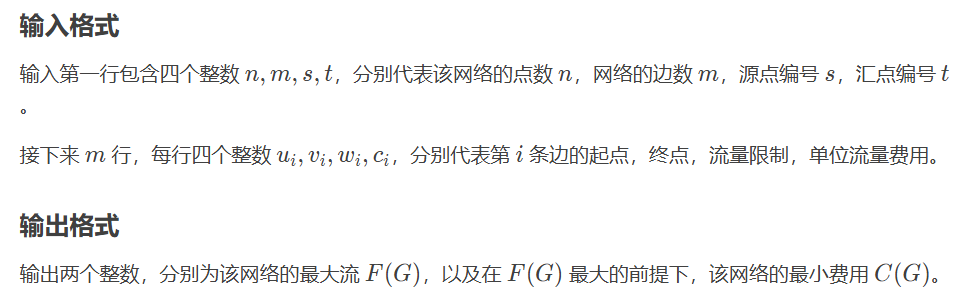
for(register int i=m;i>0;--i){u=read(),v=read(),f=read();addEdge(u,v,f);}

printf("%d",hlpp(n+1,s,t));

return 0;

}

# Primal-Dual 原始对偶算法 替换P269 ZKW费用流



#include <algorithm>

#include <cstdio>

#include <cstring>

#include <queue>

constexpr int INF = 0x3f3f3f3f;

using namespace std;

struct edge {

int v, f, c, next;

} e[100005];

struct node {

int v, e;

} p[10005];

struct mypair {

int dis, id;

bool operator<(const mypair& a) const { return dis > a.dis; }

mypair(int d, int x) { dis = d, id = x; }

};

int head[5005], dis[5005], vis[5005], h[5005];

int n, m, s, t, cnt = 1, maxf, minc;

void addedge(int u, int v, int f, int c) {

e[++cnt].v = v;

e[cnt].f = f;

e[cnt].c = c;

e[cnt].next = head[u];

head[u] = cnt;

}

bool dijkstra() {

priority\_queue<mypair> q;

for (int i = 1; i <= n; i++) dis[i] = INF;

memset(vis, 0, sizeof(vis));

dis[s] = 0;

q.push(mypair(0, s));

while (!q.empty()) {

int u = q.top().id;

q.pop();

if (vis[u]) continue;vis[u] = 1;

for (int i = head[u]; i; i = e[i].next) {

int v = e[i].v, nc = e[i].c + h[u] - h[v];

if (e[i].f && dis[v] > dis[u] + nc) {

dis[v] = dis[u] + nc;

p[v].v = u;

p[v].e = i;

if (!vis[v]) q.push(mypair(dis[v], v));

}

}

}

return dis[t] != INF;

}

void spfa() {

queue<int> q;

memset(h, 63, sizeof(h));

h[s] = 0, vis[s] = 1;

q.push(s);

while (!q.empty()) {

int u = q.front();

q.pop();

vis[u] = 0;

for (int i = head[u]; i; i = e[i].next) {

int v = e[i].v;

if (e[i].f && h[v] > h[u] + e[i].c) {

h[v] = h[u] + e[i].c;

if (!vis[v]) {

vis[v] = 1;

q.push(v);

}}}}}

int main() {

scanf("%d%d%d%d", &n, &m, &s, &t);

for (int i = 1; i <= m; i++) {

int u, v, f, c;

scanf("%d%d%d%d", &u, &v, &f, &c);

addedge(u, v, f, c);

addedge(v, u, 0, -c);

}

spfa(); // 先求出初始势能

while (dijkstra()) {

int minf = INF;

for (int i = 1; i <= n; i++) h[i] += dis[i];

for (int i = t; i != s; i = p[i].v) minf = min(minf, e[p[i].e].f);

for (int i = t; i != s; i = p[i].v) {

e[p[i].e].f -= minf;

e[p[i].e ^ 1].f += minf;

}

maxf += minf;

minc += minf \* h[t];

}

printf("%d %d\n", maxf, minc);

return 0;

}