### Min Cost Max Flow, O(n\*m\*log(n)) [not confirmed]

### For each undirected graph, 4 edge needs to be added. For directed, 2 needed.

### Tested on: Uva 10594

### struct Edge

### {

### int u, v, next;

### long long cost, cap;

### } e[5002\*4]; //maximum edge = 5000

### int source, sink, edgenum, first[102], par[102]; //maximum node = 100

### long long tData, dist[102], C;

### void addEdge( int u, int v, long long c, long long f)

### {

### e[edgenum].u = u, e[edgenum].v = v, e[edgenum].cost = c;

### e[edgenum].cap = f, e[edgenum].next = first [u], first[u] = edgenum;

### edgenum++;

### }

### void SPFA(int node)

### {

### queue < int > q;

### int vis[node+2];

### for ( int i = 0 ; i <= node; i++) dist[i] = LLONG\_MAX;

### CLR(vis);

### SET(par);

### dist[source] = 0, vis[source] = 1 ;

### q.push (source);

### while (!q.empty ())

### {

### int u = q.front();

### vis[u] = 0 ;

### q.pop ();

### for ( int k = first[u]; k != - 1 ; k = e[k].next)

### {

### int v = e[k].v;

### if (e[k].cap && (dist[u] + e[k].cost) < dist[v])

### {

### par[v] = k, dist[v] = dist[u] + e[k].cost;

### if (!vis[v])

### {

### vis[v] = 1 ;

### q.push (v);

### }

### }

### }

### }

### return;

### }

### int mincost\_maxflow (int node)

### {

### long long F=0, mn;

### C=0; //cost

### while ( 1 )

### {

### SPFA (node);

### if (dist[sink] == LLONG\_MAX) break ;

### mn= LLONG\_MAX;

### for ( int k = par[sink]; k != - 1 ; k = par[e[k].u])

### mn=min(mn, e[k].cap);

### for ( int k = par[sink]; k != - 1 ; k = par[e[k].u])

### {

### e[k].cap -= mn;

### e[k^1].cap += mn;

### }

### F += mn; //nessary change if F given

### C += mn \* dist [sink];

### }

### return F; //total flow

### }

### int main ()

### {

### int node, edge, u, v;

### long long cap;

### while(scanf("%d%d", &node, &edge)==2)

### {

### int u[edge+2], v[edge+2];

### long long cost[edge+2];

### for ( int i = 0 ; i < edge; i ++) scanf ( "%d%d%lld" , &u[i], &v[i], &cost[i]);

### scanf ( "%lld%lld" , &tData, &cap);

### edgenum = 0 ;

### SET(first);

### for ( int i = 0 ; i < edge; i ++)

### {

### addEdge (u[i], v[i], cost[i], cap);

### addEdge (v[i], u[i], -cost[i], 0 );

### addEdge (v[i], u[i], cost[i], cap);

### addEdge (u[i], v[i], -cost[i], 0 );

### }

### addEdge(0, 1, 0, tData);

### addEdge(1, 0, 0, 0);

### source = 0, sink = node;

### long long res = mincost\_maxflow (node);

### if (res==tData) printf ( "%lld\n" , C);

### else printf ( "Impossible.\n" );

### }

### return 0 ;

### }