**Dinic Max Flow, <=O(V\*V\*E) (optimizable to O(V\*E\*log(V)) using link-cut tree, undirected**

**Tested on: Spoj Fast Maximum Flow**

struct flow\_graph

{

int MAX\_V, E, s, t, head, tail;

int \*cap, \*to, \*next, \*last, \*dist, \*q, \*now;

flow\_graph() {}

flow\_graph(int V, int MAX\_E)

{

MAX\_V = V, E = 0;

cap = new int[2\*MAX\_E], to = new int[2\*MAX\_E], next = new int[2\*MAX\_E];

last = new int[MAX\_V], q = new int[MAX\_V], dist = new int[MAX\_V], now = new int[MAX\_V];

fill(last,last+MAX\_V,-1);

}

void clear()

{

fill(last,last+MAX\_V,-1);

E = 0;

}

//add edge such a way that one of its (2\*n)th and (2\*n+1)th edge for residual graph

//the sequence almost random and fixed so that all e^1 forms residual graph

void add\_edge(int u, int v, int w)

{

to[E] = v, cap[E] = w, next[E] = last[u];

last[u] = E++; //edge counter (double)

to[E] = u, cap[E] = w, next[E] = last[v]; // cap[E]=0 for directional, cap[E]=w for undirectional

last[v] = E++;

}

//level graph construction using bfs in reverse direction in O(E)

bool bfs()

{

fill(dist,dist+MAX\_V,-1);

//head, tail, & q[] used replacing Queue data structure

head = tail = 0;

q[tail] = t;

++tail;

dist[t] = 0;

while(head<tail)

{

int v = q[head];

++head;

for(int e = last[v]; e!=-1; e = next[e])

{

if(cap[e^1]>0 && dist[to[e]]==-1) //cap[e^1]=0 means residual capacity is full

{

q[tail] = to[e];

++tail;

dist[to[e]] = dist[v]+1;

}

}

}

return dist[s]!=-1;

}

// blocking flow in each level graph found by dfs in O(VE) time

int dfs(int v, int f)

{

if(v==t) return f;

for(int &e = now[v]; e!=-1; e = next[e])

{

if(cap[e]>0 && dist[to[e]]==dist[v]-1)

{

int ret = dfs(to[e],min(f,cap[e]));

if(ret>0)

{

cap[e] -= ret;

cap[e^1] += ret; //as next edge is for residual graph

return ret; //return argument flow (min)

}

}

}

return 0; //no argument flow found

}

long long max\_flow(int source, int sink)

{

s = source, t = sink;

long long f = 0;

int x;

while(bfs())

{

for(int i = 0; i<MAX\_V; ++i) now[i] = last[i]; //as now[v] changing with \*e in dfs

while(true)

{

x = dfs(s,INT\_MAX);

if(x==0) break;

f += x;

}

}

return f;

}

} G;

int main()

{

int V,E,u,v,c; //V=vertex, E=edge

scanf("%d %d",&V,&E);

G = flow\_graph(V,E);

for(int i = 0; i<E; ++i)

{

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

G.add\_edge(u-1,v-1,c); //input 1 indexed, calculation 0 indexed, push swapping u-v if reverse edge

}

printf("%lld\n",G.max\_flow(0,V-1)); //input 1 indexed, calculation 0 indexed

return 0;

}

**To solve Maximum Bipertite Match with dinic change the followings**

**(let n is the number of elements in set A and m in set B)**

G = flow\_graph(n+m+2, E+n+m);

G.add\_edge(u, n+v, 1);

for(int i=1;i<=n;i++) G.add\_edge(0, i, 1); //source 0 to all node of A

for(int i=1;i<=m;i++) G.add\_edge(n+i, n+m+1, 1); //all node of B to sink

printf("%d\n",G.max\_flow(0,cow+bull+1)); //source to sink