# aStar

//2449

#include<cstring>

#include<algorithm>

#include<queue>

#include<cstdio>

using namespace std;

#define INF 0xffffff

#define MAXN 100010

struct node{

int to;

int val;

int next;

};

struct node2{

int to;

int g,f;

bool operator<(const node2 &r ) const{

if(r.f==f)

return r.g<g;

return r.f<f;

}

};

node edge[MAXN],edge2[MAXN];

int n,m,s,t,k,cnt,cnt2,ans;

int dis[1010],visit[1010],head[1010],head2[1010];

void init(){

memset(head,-1,sizeof(head));

memset(head2,-1,sizeof(head2));

cnt=cnt2=1;

}

void addedge(int from,int to,int val){

edge[cnt].to=to;

edge[cnt].val=val;

edge[cnt].next=head[from];

head[from]=cnt++;

}

void addedge2(int from,int to,int val){

edge2[cnt2].to=to;

edge2[cnt2].val=val;

edge2[cnt2].next=head2[from];

head2[from]=cnt2++;

}

bool spfa(int s,int n,int head[],node edge[],int dist[]) {

queue<int>Q1;

int inq[1010];

for(int i=0;i<=n;i++){

dis[i]=INF;

inq[i]=0;

}

dis[s]=0;

Q1.push(s);

inq[s]++;

while(!Q1.empty()){

int q=Q1.front();

Q1.pop();

inq[q]--;

if(inq[q]>n)

return false;

int k=head[q];

while(k>=0){

if(dist[edge[k].to]>dist[q]+edge[k].val){

dist[edge[k].to]=edge[k].val+dist[q];

if(!inq[edge[k].to]){

inq[edge[k].to]++;

Q1.push(edge[k].to);

}

}

k=edge[k].next;

}

}

return true;

}

int A\_star(int s,int t,int n,int k,int head[],node edge[],int dist[]){

node2 e,ne;

int cnt=0;

priority\_queue<node2>Q;

if(s==t)

k++;

if(dis[s]==INF)

return -1;

e.to=s;

e.g=0;

e.f=e.g+dis[e.to];

Q.push(e);

while(!Q.empty()){

e=Q.top();

Q.pop();

if(e.to==t)//找到一条最短路径

cnt++;

if(cnt==k)//找到k短路

return e.g;

for(int i=head[e.to]; i!=-1; i=edge[i].next) {

ne.to=edge[i].to;

ne.g=e.g+edge[i].val;

ne.f=ne.g+dis[ne.to];

Q.push(ne);

}

}

return -1;

}

int main(){

while(~scanf("%d%d",&n,&m)){

init();

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

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

addedge(a,b,c);

addedge2(b,a,c);

}

scanf("%d%d%d",&s,&t,&k);

spfa(t,n,head2,edge2,dis);

ans=A\_star(s,t,n,k,head,edge,dis);

printf("%d\n",ans);

}

return 0;

}