```
#include <stdio.h>
#include <queue>
#include <vector>
#define MAX 100001
#define INF 1000000
using namespace std;

//int z[MAX];
//int A[MAX][MAX];
//int c[MAX][MAX];// c[u][i] la trong so cung (u,A[u][i])
vector<int> A[MAX];
vector<int> c[MAX];
int n,m;

int d[MAX];// d[v] la can tren cua do dai duong di ngan nhat tu s den cac dinh
int node[MAX];// node[i] la dinh thu i trong HEAP
int idx[MAX];// chi so cua dinh trong HEAP
int sH;// size of heap
bool fixed[MAX];
int s,t;

void swap(int i, int j){

   int tmp = node[i]; node[i] = node[j]; node[j] = tmp;
   idx[node[i]] = i; idx[node[j]] = j;
}
```

```
void upHeap(int i){
    if(i == 0) return;

    while(i > 0){
        int pi = (i-1)/2;
        if(d[node[i]] < d[node[pi]]){
            swap(i,pi);
        }else{
            break;
        }
        i = pi;
    }
}
void downHeap(int i){
    int L = 2*i+1;
    int R = 2*i+2;
    int maxIdx = i;
    if(L < sH && d[node[L]] < d[node[maxIdx]]) maxIdx = L;
    if(R < sH && d[node[R]] < d[node[maxIdx]]) maxIdx = R;
    if(maxIdx != i){
        swap(i,maxIdx); downHeap(maxIdx);
    }
}
void insert(int k, int v){
    // add element key = k, value = v into HEAP
    d[v] = k;
```

```
    node[sH] = v;
    idx[node[sH]] = sH;
    upHeap(sH);
    sH++;
}
int deleteMin(){
    int sel_node = node[0];
    swap(0,sH-1);
    sH--;
    downHeap(0);
    return sel_node;
}
void input(){
    scanf("%d%d",&n,&m);
    for(int k = 1; k <= m; k++){
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        A[u].push_back(v);
        c[u].push_back(w);

    }
    scanf("%d%d",&s,&t);
}
void input(char* filename){
    FILE* f = fopen(filename,"r");
    fscanf(f,"%d%d",&n,&m);
```

```
    for(int k = 1; k <= m; k++){
        int u,v,w;
        fscanf(f,"%d%d%d",&u,&v,&w);
        A[u].push_back(v);
        c[u].push_back(w);



    }
    fscanf(f,"%d%d",&s,&t);
    fclose(f);
}
void printHeap(){
    for(int i = 0; i < sH; i++){
        printf("%d: d[%d] = %d\n",i,node[i],d[node[i]]);
    }
    for(int v = 1; v <= n; v++) if(idx[v] >= 0) printf("idx[%d] = %d ",v,idx[v]);
    printf("\n-------------------------\n");
}
void solve(int s){
    sH = 0;
    for(int v = 1; v <= n; v++){
        fixed[v] = false;
        idx[v] = -1;
    }
    d[s] = 0;
```

```
    fixed[s] = true;
    for(int i = 0; i < A[s].size(); i++){
        int v = A[s][i];
        insert(c[s][i],v);
    }
    //printHeap();
    while(sH > 0){
        int u = deleteMin();
        fixed[u] = true;
        //printf("FIX d[%d] = %d\n",u,d[u]);
        //printHeap();

        for(int i = 0; i < A[u].size(); i++){
            int v = A[u][i];
            if(fixed[v]) continue;
            if(idx[v] == -1){
                int w = d[u] + c[u][i];
                insert(w,v);
                //printf("insert node %d with d[idx[u]] = %d, c[u][i] = %d, d[idx[v]]
= %d\n",v,d[idx[u]],c[u][i],d[idx[v]]);
            }else{
               if(d[v] > d[u] + c[u][i]){
                  d[v] = d[u] + c[u][i];
                  upHeap(idx[v]);
                  //printf("update d[%d] = %d -> upHeap(%d)\n",v,d[idx[v]],idx[v]);
               }
```

```
            }
        }
        //printHeap();
    }
}
void solve(char* fi, char* fo){
    input(fi);
    solve(s);
    FILE* f = fopen(fo,"w");
    int rs = d[t];
    if(idx[t] < 0) rs = -1;
    fprintf(f,"%d",rs);
    printf("%d",rs);
    fclose(f);
}
void solve(){
    input();
    solve(s);
    int rs = d[t];
    if(!fixed[t]) rs = -1;
    printf("%d",rs);

}

int main(){
    solve();
```

```
    //solve("Test09/MINPATH.INP","Test09/MINPATH.OUT");
}
```