**道路和航路**

问题描述

农夫约翰正在针对一个新区域的牛奶配送合同进行研究。他打算分发牛奶到T个城镇（标号为1..T），这些城镇通过R条标号为（1..R）的道路和P条标号为（1..P）的航路相连。

每一条公路i或者航路i表示成连接城镇Ai（1<=A\_i<=T）和Bi（1<=Bi<=T）代价为Ci。每一条公路，Ci的范围为0<=Ci<=10,000；由于奇怪的运营策略，每一条航路的Ci可能为负的，也就是-10,000<=Ci<=10,000。

每一条公路都是双向的，正向和反向的花费是一样的，都是非负的。

每一条航路都根据输入的Ai和Bi进行从Ai->Bi的单向通行。实际上，如果现在有一条航路是从Ai到Bi的话，那么意味着肯定没有通行方案从Bi回到Ai。

农夫约翰想把他那优良的牛奶从配送中心送到各个城镇，当然希望代价越小越好，你可以帮助他嘛？配送中心位于城镇S中（1<=S<=T）。

输入格式

输入的第一行包含四个用空格隔开的整数T，R，P，S。

接下来R行，描述公路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

接下来P行，描述航路信息，每行包含三个整数，分别表示Ai，Bi和Ci。

输出格式

输出T行，分别表示从城镇S到每个城市的最小花费，如果到不了的话输出NO PATH。

样例输入

6 3 3 4  
1 2 5  
3 4 5  
5 6 10  
3 5 -100  
4 6 -100  
1 3 -10

样例输出

NO PATH  
NO PATH  
5  
0  
-95  
-100

数据规模与约定

对于20%的数据，T<=100，R<=500，P<=500；

对于30%的数据，R<=1000，R<=10000，P<=3000；

对于100%的数据，1<=T<=25000，1<=R<=50000，1<=P<=50000。

锦囊1

使用最短路径。

锦囊2

将城镇看成结点，将公路和航路看成边，使用带堆优化的Dijkstra来求到每个城镇的最短路。

C++代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <queue>

#include <stack>

#include <vector>

#define clr(a,b) memset(a, b, sizeof(a))

using namespace std;

const int N = 25050;

const int E = 150500;

//邻接表

int h[N], v[E], w[E], nxt[E], el;

void initEdge() {

clr(h, -1); el = 0;

}

void addEdge(int x, int y, int z) {

v[el] = y; w[el] = z; nxt[el] = h[x]; h[x] = el++;

}

//belong[i] 表示节点 i 所在的强连通分量；

//cnt 表示强连通分量的个数；

int dfn[N], sta[N], low[N], belong[N];

int top, cnt, ind, n;

bool vis[N];

void TarjanSolve(int u) {

dfn[u] = low[u] = ++ind;

vis[u] = true;

sta[++top] = u;

for(int p=h[u]; ~p; p=nxt[p]) {

int i = v[p];

if(!dfn[i]) {

TarjanSolve(i);

if(low[i] < low[u]) low[u] = low[i];

}

else

if(vis[i] && dfn[i] < low[u])

low[u] = dfn[i];

}

if(dfn[u] == low[u]) {

++cnt;

while(1) {

int i = sta[top--];

vis[i] = false;

belong[i] = cnt;

if(i == u) break;

}

}

}

void Tarjan() {//注意节点是从几开始存的

clr(dfn, 0);

clr(vis, 0);

top = cnt = ind = 0;

for(int i=1; i<=n; i++)//这里节点从1开始存，若从0开始存要改这里

if(!dfn[i]) TarjanSolve(i);

}

struct EDGE {

int u, v, w;

bool flag;

EDGE(){}

EDGE(int x, int y, int z, bool f):u(x), v(y), w(z), flag(f){}

} edge[E];

int edgel;

bool visitable[N];

void dfs(int x) {

visitable[x] = true;

for(int i=h[x]; ~i; i=nxt[i]) {

if(!visitable[v[i]]) {

dfs(v[i]);

}

}

}

int indegree[N];

//链表

int lh[N], lel, lv[E], lnxt[E];

void initLink() {

clr(lh, -1); lel = 0;

}

void addLink(int x, int y) {

lv[lel] = y; lnxt[lel] = lh[x]; lh[x] = lel++;

}

int dis[N];

bool tag[N];

int main() {

int r, p, s;

while(~scanf("%d%d%d%d", &n, &r, &p, &s)) {

clr(visitable, 0);

initEdge();

edgel = 0;

int x, y, z;

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

scanf("%d%d%d", &x, &y, &z);

addEdge(x, y, z);

addEdge(y, x, z);

edge[edgel++] = EDGE(x, y, z, false);

}

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

scanf("%d%d%d", &x, &y, &z);

addEdge(x, y, z);

edge[edgel++] = EDGE(x, y, z, true);

}

Tarjan();

dfs(s);

initEdge();

initLink();

clr(indegree, 0);

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

if(visitable[edge[i].u] && visitable[edge[i].v]) {

addEdge(edge[i].u, edge[i].v, edge[i].w);

if(edge[i].flag) {

++ indegree[belong[edge[i].v]];

addLink(belong[edge[i].v], edge[i].v);

} else {

addEdge(edge[i].v, edge[i].u, edge[i].w);

}

}

}

stack<int> zeroDegree;

priority\_queue<pair<int,int> > que;

clr(vis, false);

clr(tag, false);

clr(dis, 0x3f);

dis[s] = 0;

que.push(make\_pair(0, s));

while(!que.empty() || !zeroDegree.empty()) {

if(que.empty()) {

int x = zeroDegree.top(); zeroDegree.pop();

for(int i=lh[x]; ~i; i=lnxt[i]) {

int y = lv[i];

if(!vis[y]) {

vis[y] = true;

que.push(make\_pair(-dis[y], y));

}

}

} else {

int x = que.top().second; que.pop();

if(tag[x]) continue;

tag[x] = true;

for(int i=h[x]; ~i; i=nxt[i]) {

int y = v[i];

if(!tag[y] && dis[y] > dis[x] + w[i]) {

dis[y] = dis[x] + w[i];

if(belong[x] == belong[y]) {

que.push(make\_pair(-dis[y], y));

}

}

if(belong[x] != belong[y]) {

-- indegree[belong[y]];

if(indegree[belong[y]] == 0) {

zeroDegree.push(belong[y]);

}

}

}

}

}

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

if(visitable[i]) {

printf("%d\n", dis[i]);

} else {

puts("NO PATH");

}

}

}

return 0;

}

C代码：

#include<stdio.h>

int T , R , P , S , tol;

int l,r;

const int MAXE = 200007;

const int MAXP = 50007;

const int INF = 2147483000;

int lx[200007] , ne[200007] , vx[200007] , se[50007] , v[50007],q[50007];

int Lx[2000007];

void init()

{

int i;

for(i=1;i<=T;i++) v[i] = INF;

v[S] = 0;

tol = 0;

}

void add\_edge(int x,int y,int z){

tol ++;

lx[tol] = y;

ne[tol] = se[x];

vx[tol] = z;

se[x] = tol;

}

void spfa(int x)

{

int t = se[x];

int V = v[x];

int p;

while(t!=0){

p = lx[t];

if(v[p] > V + vx[t]){

v[p] = V + vx[t];

if(l!=r && v[p] < v[Lx[l + 1]]){

q[p] = 1;

Lx[l] = p;

l--;

}

if(!q[p]){

q[p] = 1;

r++;

Lx[r] = p;

}

}

t = ne[t];

}

q[x] = 0;

}

int main()

{

int x,y,z;

while(scanf("%d%d%d%d",&T,&R,&P,&S)!=EOF){

init();

int i;

for(i=1;i<=R;i++){

scanf("%d%d%d",&x,&y,&z);

add\_edge(x,y,z);

add\_edge(y,x,z);

}

for(i=1;i<=P;i++){

scanf("%d%d%d",&x,&y,&z);

add\_edge(x,y,z);

}

l = 500000; r = 500001; Lx[500001] = S; q[S] = 1;

while(l<r){

l++;

while(l<=r &&q[Lx[l]]==0) l++;

if(l>r) break;

spfa(Lx[l]);

}

for(i=1;i<=T;i++) if(v[i] !=INF) printf("%d\n",v[i]);

else printf("NO PATH\n");

}

return 0;

}