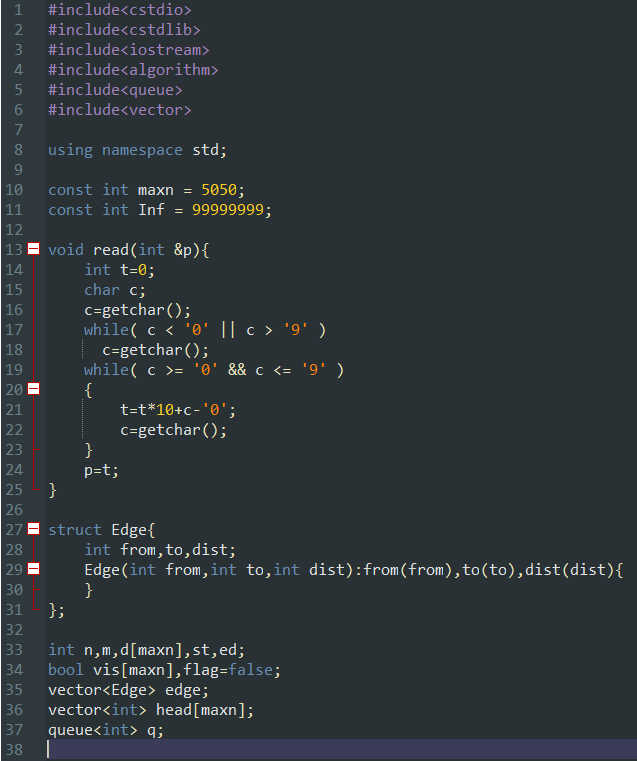
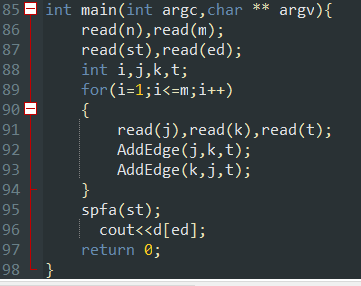
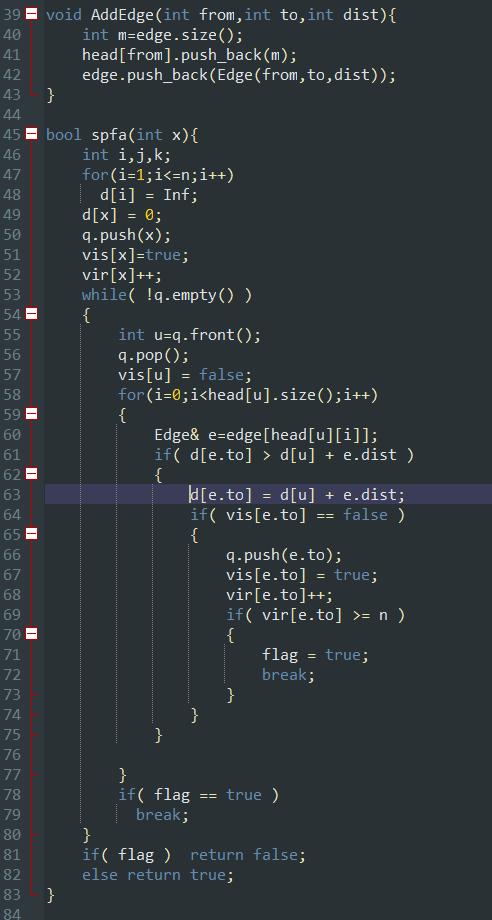
标准SPFA





#include<cstdio>

#include<cstdlib>

#include<iostream>

#include<algorithm>

#include<queue>

#include<vector>

using namespace std;

const int maxn = 5050;

const int Inf = 99999999;

void read(int &p){

int t=0;

char c;

c=getchar();

while( c < '0' || c > '9' )

c=getchar();

while( c >= '0' && c <= '9' )

{

t=t\*10+c-'0';

c=getchar();

}

p=t;

}

struct Edge{

int from,to,dist;

Edge(int from,int to,int dist):from(from),to(to),dist(dist){

}

};

int n,m,vir[maxn],d[maxn],st,ed;

bool vis[maxn],flag=false;

vector<Edge> edge;

vector<int> head[maxn];

queue<int> q;

void AddEdge(int from,int to,int dist){

int m=edge.size();

head[from].push\_back(m);

edge.push\_back(Edge(from,to,dist));

}

bool spfa(int x){

int i,j,k;

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

d[i] = Inf;

d[x] = 0;

q.push(x);

vis[x]=true;

vir[x]++;

while( !q.empty() )

{

int u=q.front();

q.pop();

vis[u] = false;

for(i=0;i<head[u].size();i++)

{

Edge& e=edge[head[u][i]];

if( d[e.to] > d[u] + e.dist )

{

d[e.to] = d[u] + e.dist;

if( vis[e.to] == false )

{

q.push(e.to);

vis[e.to] = true;

vir[e.to]++;

if( vir[e.to] >= n )

{

flag = true;

break;

}

}

}

}

if( flag == true )

break;

}

if( flag ) return false;

else return true;

}

int main(int argc,char \*\* argv){

read(n),read(m);

read(st),read(ed);

int i,j,k,t;

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

{

read(j),read(k),read(t);

AddEdge(j,k,t);

AddEdge(k,j,t);

}

spfa(st);

cout<<d[ed];

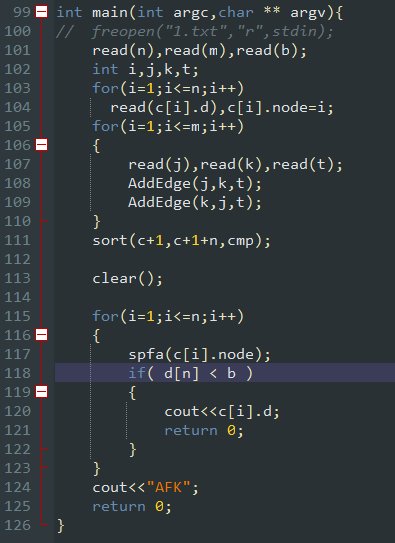
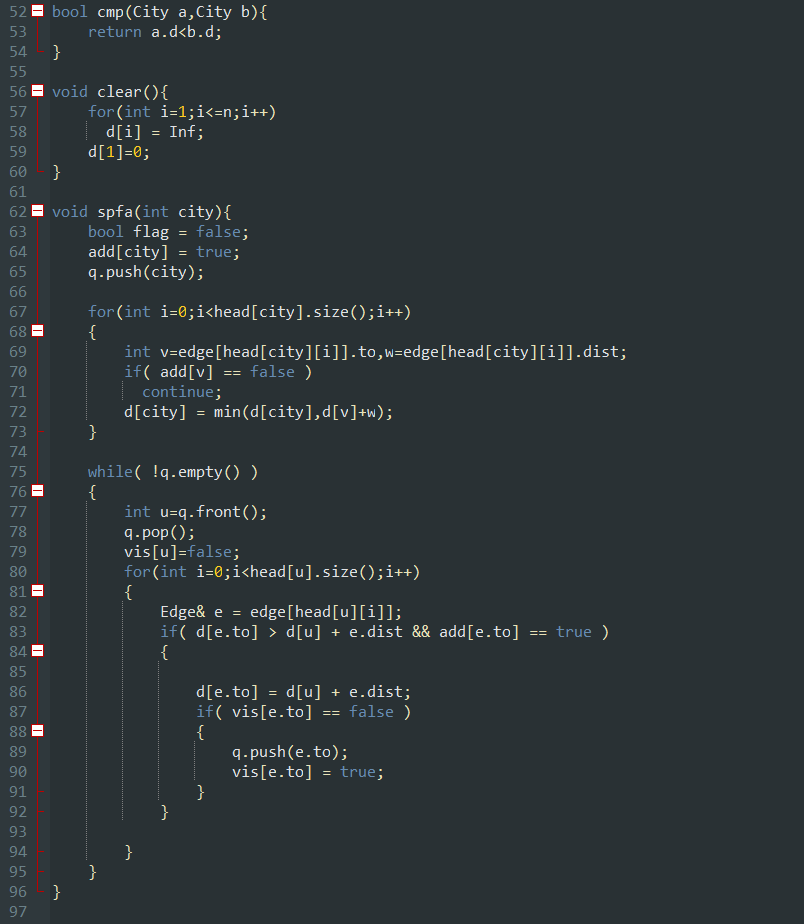
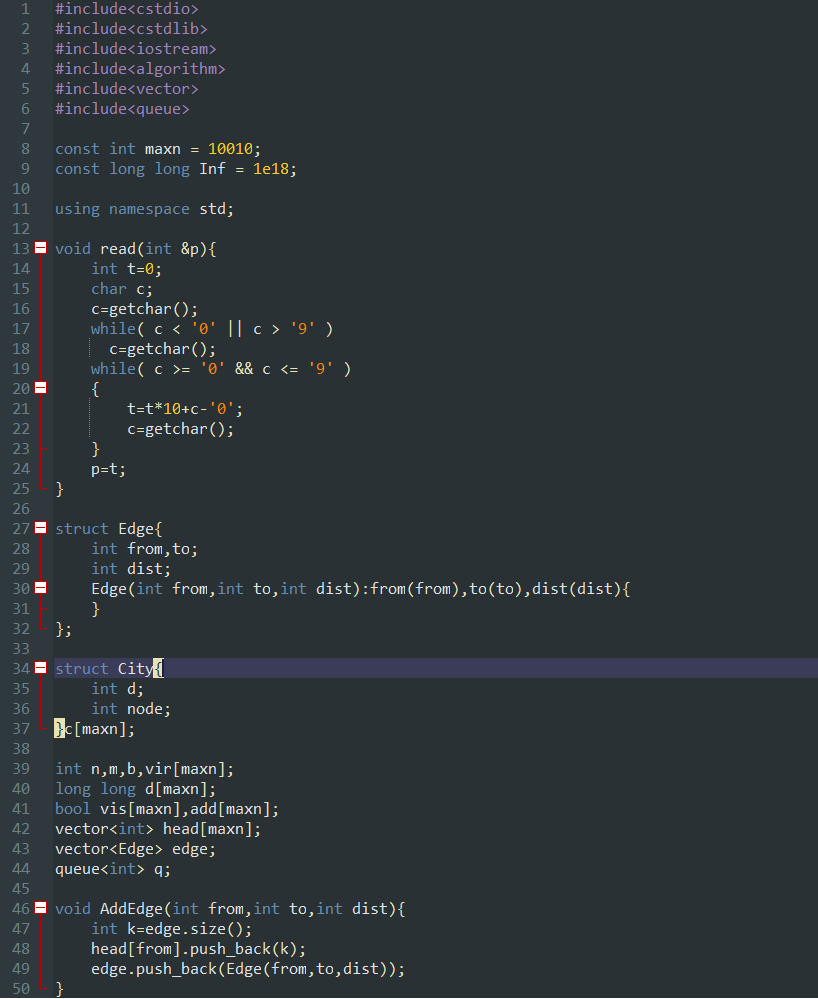
return 0;

}

SPFA动态加点

洛谷P1462





#include<cstdio>

#include<cstdlib>

#include<iostream>

#include<algorithm>

#include<vector>

#include<queue>

const int maxn = 10010;

const long long Inf = 1e18;

using namespace std;

void read(int &p){

int t=0;

char c;

c=getchar();

while( c < '0' || c > '9' )

c=getchar();

while( c >= '0' && c <= '9' )

{

t=t\*10+c-'0';

c=getchar();

}

p=t;

}

struct Edge{

int from,to;

int dist;

Edge(int from,int to,int dist):from(from),to(to),dist(dist){

}

};

struct City{

int d;

int node;

}c[maxn];

int n,m,b,vir[maxn];

long long d[maxn];

bool vis[maxn],add[maxn];

vector<int> head[maxn];

vector<Edge> edge;

queue<int> q;

void AddEdge(int from,int to,int dist){

int k=edge.size();

head[from].push\_back(k);

edge.push\_back(Edge(from,to,dist));

}

bool cmp(City a,City b){

return a.d<b.d;

}

void clear(){

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

d[i] = Inf;

d[1]=0;

}

void spfa(int city){

bool flag = false;

add[city] = true;

q.push(city);

for(int i=0;i<head[city].size();i++)

{

int v=edge[head[city][i]].to,w=edge[head[city][i]].dist;

if( add[v] == false )

continue;

d[city] = min(d[city],d[v]+w);

}

while( !q.empty() )

{

int u=q.front();

q.pop();

vis[u]=false;

for(int i=0;i<head[u].size();i++)

{

Edge& e = edge[head[u][i]];

if( d[e.to] > d[u] + e.dist && add[e.to] == true )

{

d[e.to] = d[u] + e.dist;

if( vis[e.to] == false )

{

q.push(e.to);

vis[e.to] = true;

}

}

}

}

}

int main(int argc,char \*\* argv){

// freopen("1.txt","r",stdin);

read(n),read(m),read(b);

int i,j,k,t;

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

read(c[i].d),c[i].node=i;

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

{

read(j),read(k),read(t);

AddEdge(j,k,t);

AddEdge(k,j,t);

}

sort(c+1,c+1+n,cmp);

clear();

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

{

spfa(c[i].node);

if( d[n] < b )

{

cout<<c[i].d;

return 0;

}

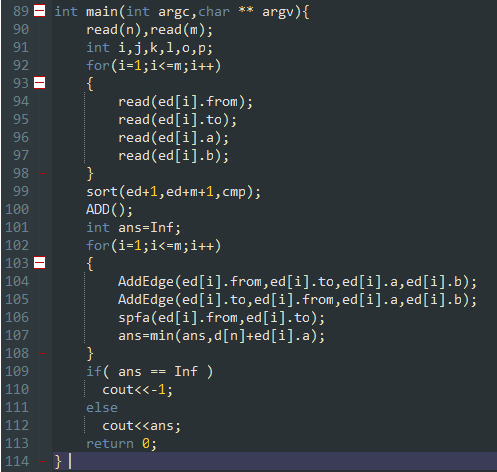
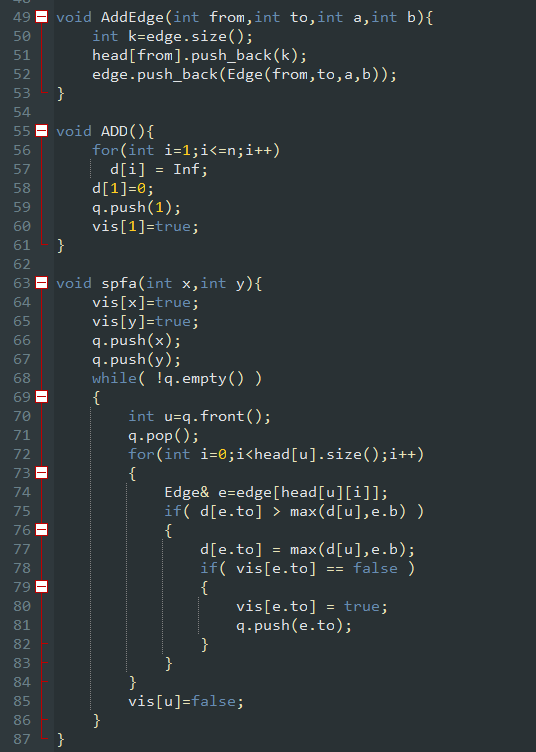
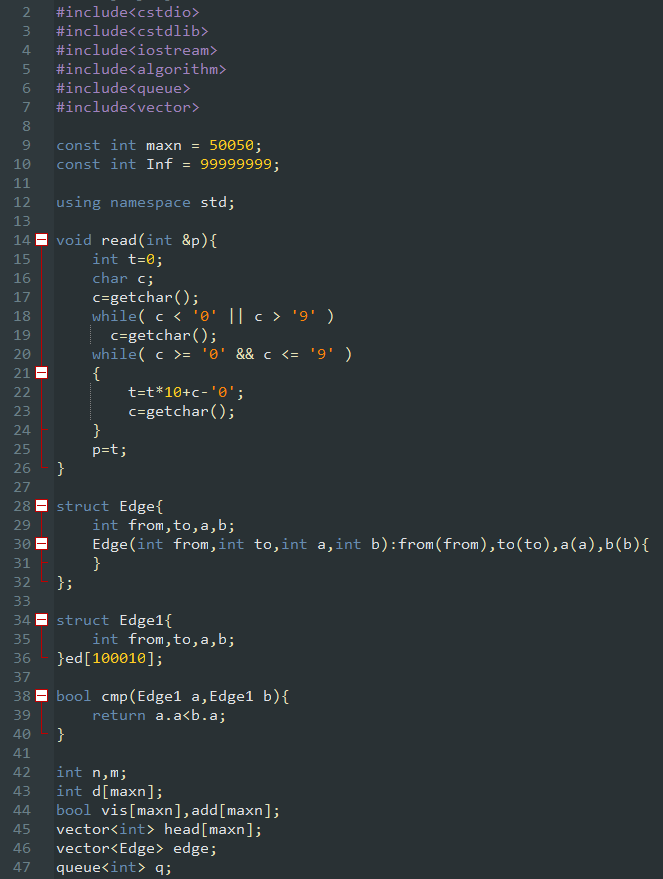
}

cout<<"AFK";

return 0;

}

SPFA动态加边



// luogu-judger-enable-o2

#include<cstdio>

#include<cstdlib>

#include<iostream>

#include<algorithm>

#include<queue>

#include<vector>

const int maxn = 50050;

const int Inf = 99999999;

using namespace std;

void read(int &p){

int t=0;

char c;

c=getchar();

while( c < '0' || c > '9' )

c=getchar();

while( c >= '0' && c <= '9' )

{

t=t\*10+c-'0';

c=getchar();

}

p=t;

}

struct Edge{

int from,to,a,b;

Edge(int from,int to,int a,int b):from(from),to(to),a(a),b(b){

}

};

struct Edge1{

int from,to,a,b;

}ed[100010];

bool cmp(Edge1 a,Edge1 b){

return a.a<b.a;

}

int n,m;

int d[maxn];

bool vis[maxn],add[maxn];

vector<int> head[maxn];

vector<Edge> edge;

queue<int> q;

void AddEdge(int from,int to,int a,int b){

int k=edge.size();

head[from].push\_back(k);

edge.push\_back(Edge(from,to,a,b));

}

void ADD(){

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

d[i] = Inf;

d[1]=0;

q.push(1);

vis[1]=true;

}

void spfa(int x,int y){

vis[x]=true;

vis[y]=true;

q.push(x);

q.push(y);

while( !q.empty() )

{

int u=q.front();

q.pop();

for(int i=0;i<head[u].size();i++)

{

Edge& e=edge[head[u][i]];

if( d[e.to] > max(d[u],e.b) )

{

d[e.to] = max(d[u],e.b);

if( vis[e.to] == false )

{

vis[e.to] = true;

q.push(e.to);

}

}

}

vis[u]=false;

}

}

int main(int argc,char \*\* argv){

read(n),read(m);

int i,j,k,l,o,p;

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

{

read(ed[i].from);

read(ed[i].to);

read(ed[i].a);

read(ed[i].b);

}

sort(ed+1,ed+m+1,cmp);

ADD();

int ans=Inf;

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

{

AddEdge(ed[i].from,ed[i].to,ed[i].a,ed[i].b);

AddEdge(ed[i].to,ed[i].from,ed[i].a,ed[i].b);

spfa(ed[i].from,ed[i].to);

ans=min(ans,d[n]+ed[i].a);

}

if( ans == Inf )

cout<<-1;

else

cout<<ans;

return 0;

}