**算法训练 最短路**

时间限制：1.0s   内存限制：256.0MB

问题描述

给定一个n个顶点，m条边的有向图（其中某些边权可能为负，但保证没有负环）。请你计算从1号点到其他点的最短路（顶点从1到n编号）。

输入格式

第一行两个整数n, m。

接下来的m行，每行有三个整数u, v, l，表示u到v有一条长度为l的边。

输出格式

共n-1行，第i行表示1号点到i+1号点的最短路。

样例输入

3 3  
1 2 -1  
2 3 -1  
3 1 2

样例输出

-1  
-2

数据规模与约定

对于10%的数据，n = 2，m = 2。

对于30%的数据，n <= 5，m <= 10。

对于100%的数据，1 <= n <= 20000，1 <= m <= 200000，-10000 <= l <= 10000，保证从任意顶点都能到达其他所有顶点。

锦囊1

使用最短路算法。

锦囊2

使用Dijkstra算法，此图的边数比点数的平方要少很多，因此应该使用带堆优化的Dijkstra。

本题的C++参考代码如下：

#include<iostream>

#include<cstdio>

#include<cmath>

#include<algorithm>

#include<queue>

#include<cstring>

#define ENT putchar('\n')

using namespace std;

const int maxn=100000+10,INF=-1u>>1;

struct Tedge{int x,y,w,next;}adj[maxn\*2];int fch[maxn],ms=0;

void AddEdge(int u,int v,int w){

adj[++ms]=(Tedge){u,v,w,fch[u]};fch[u]=ms;return;

}

bool vis[maxn];int dist[maxn],n,m;

void SPFA(){

queue<int>Q;memset(vis,false,sizeof(vis));

for(int i=1;i<=n+1;i++)dist[i]=INF;dist[1]=0;

Q.push(1);vis[1]=true;

while(!Q.empty()){

int u=Q.front();Q.pop();vis[u]=false;

for(int i=fch[u];i;i=adj[i].next){

int v=adj[i].y;

if(dist[v]>dist[u]+adj[i].w){

dist[v]=dist[u]+adj[i].w;

if(!vis[v]) vis[v]=true,Q.push(v);

}

}

} return ;

}

inline int read(){

int x=0,sig=1;char ch=getchar();

while(!isdigit(ch)){if(ch=='-') sig=-1;ch=getchar();}

while(isdigit(ch)) x=10\*x+ch-'0',ch=getchar();

return x\*=sig;

}

inline void write(int x){

if(x==0){putchar('0');return;}if(x<0) putchar('-'),x=-x;

int len=0,buf[15];while(x) buf[len++]=x%10,x/=10;

for(int i=len-1;i>=0;i--) putchar(buf[i]+'0');return;

}

void init(){

n=read();m=read();int a,b,c;

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

a=read();b=read();c=read();

AddEdge(a,b,c);

} return;

}

void work(){

SPFA();

return;

}

void print(){

for(int i=2;i<=n;i++) write(dist[i]),ENT;

return;

}

int main(){

init();work();print();return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#define inf 100000

struct In{

int e;

int w;

int next;

}map[200010];

int dis[20010],Q[20010];

int vis[20010],head[20010];

void SPFA(int n){

int i,j,front,rear,temp;

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

dis[i]=inf;

}

dis[1]=0;vis[1]=1;

front=0;rear=1;

Q[front]=1;

while(front<rear){

temp=Q[front++];

vis[temp]=0;

j=head[temp];

while(j>0){

if(dis[map[j].e]>map[j].w+dis[temp]){

dis[map[j].e]=map[j].w+dis[temp];

if(!vis[map[j].e]){

Q[rear++]=map[j].e;

vis[map[j].e]=1;

}

}

j=map[j].next;

}

}

}

int main(){

int n,m,i,j,a,b,val;

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

memset(Q,0,sizeof(Q));

memset(head,0,sizeof(head));

memset(vis,0,sizeof(vis));

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

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

map[i].e=b;

map[i].w=val;

map[i].next=head[a];

head[a]=i;

}

SPFA(n);

for(i=2;i<=n;i++){

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

}

}

return 0;

}

本题的Java参考代码如下：

import java.io.\*;

import java.util.\*;

class Main

{

static int n,m;

static int[] u;

static int[] v;

static int[] w;

static int[] d;

static int[] first;

static int[] next;

static Queue<Integer> q=new LinkedList<Integer>();

static boolean[] inq;

public static void main(String[] args) throws IOException

{

int i;

BufferedReader bfr=new BufferedReader(new InputStreamReader(System.in));

String str = bfr.readLine();

String[] s = str.split("\\s");

n=Integer.parseInt(s[0]);

m=Integer.parseInt(s[1]);

n++;

m++;

u=new int[m];

v=new int[m];

w=new int[m];

first=new int[n];

next=new int[m];

d=new int[n];

inq=new boolean[n];

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

first[i]=-1;

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

{

str = bfr.readLine();

s = str.split(" ");

u[i]=Integer.parseInt(s[0]);

v[i]=Integer.parseInt(s[1]);

w[i]=Integer.parseInt(s[2]);

next[i]=first[u[i]];

first[u[i]]=i;

}

spfa(1);

for(i=2;i<n;i++)

System.out.println(d[i]);

}

public static void spfa(int s)

{

int i,x;

for(i=2;i<n;i++)

d[i]=Integer.MAX\_VALUE;

q.offer(s);

while(!q.isEmpty())

{

x=q.poll();

inq[x]=false;

for(i=first[x];i!=-1;i=next[i])

if(d[v[i]]>d[x]+w[i])

{

d[v[i]]=d[x]+w[i];

if(!inq[v[i]])

{

inq[v[i]]=true;

q.offer(v[i]);

}

}

}

}

}