/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//作者：司玉龙

//时间：2018年11月12日

//程序：求关键路径

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

#define MAX 10000000

#define MAX\_VERTEX\_NUM 20

int ve[MAX\_VERTEX\_NUM];

/\*顺序栈的定义\*/

#define Stack\_Size 100

typedef struct sqStack

{

int \*elem;

int top;

int stackSize;//栈数组长度

}sqStack;

/\*顺序栈的初始化\*/

void initStack\_Sq(sqStack &S)

{

S.elem=new int[Stack\_Size];

S.top=-1;

S.stackSize=Stack\_Size;

}

/\*入栈\*/

void push(sqStack &S,int x)

{

if(S.top==Stack\_Size-1)

cout<<"Stack Overflow!";

S.elem[++S.top]=x;

}

/\*出栈\*/

int pop(sqStack &S)

{

int x;

if(S.top==-1)

cout<<"Stack Empty!";

x=S.elem[S.top--];

return x;

}

typedef struct EdgeNode

{//边表结点的定义

int adjvex;//存放邻接点在顶点表中的位置

struct EdgeNode \* nextedge;//指向下一个边表结点

int weight;

}EdgeNode;

typedef struct VexNode

{//顶点表结点的定义

char vex;//存放顶点信息

EdgeNode \* firstedge;//指向第一个边表结点

int indegree;

}VexNode;

typedef struct

{//顶点表的定义

VexNode vexs[MAX\_VERTEX\_NUM];

int vexnum,edgenum;

}LGraph;

/\*构造有向图的邻接表\*/

void CreateDG\_AL(LGraph &G,int n,int e)

{

int i,j,k,w;

G.vexnum=n;

G.edgenum=e;

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

{

cin>>G.vexs[i].vex;

G.vexs[i].firstedge=NULL;//初始化为空

}

for(k=0;k<e;k++)

{

EdgeNode \*p;

cin>>i>>j>>w;

p=new EdgeNode;

p->adjvex=j;

p->weight=w;

p->nextedge=G.vexs[i].firstedge;

G.vexs[i].firstedge=p;//采用头插法

}

}

//拓扑排序并求各顶点事件的最早发生时间及拓扑逆序列

void TopoSort(LGraph &G,sqStack &T)

{

sqStack S;

initStack\_Sq(S);

EdgeNode \*p;

int count=0;

int i;

for(i=0;i<G.vexnum;i++)

G.vexs[i].indegree=0;//初始化为0

for(i=0;i<G.vexnum;i++)

{//计算各个顶点的入度

p=G.vexs[i].firstedge;

while(p)

{

G.vexs[p->adjvex].indegree++;

p=p->nextedge;

}

}

for(i=0;i<G.vexnum;i++)

if(G.vexs[i].indegree==0)

push(S,i);//将度为0的顶点入栈,这里进栈的是入度为0的顶点在数组中的位置

for(i=0;i<G.vexnum;i++)

ve[i]=0;//初始化顶点事件的最早发生时间为0

while(S.top!=-1)

{

i=pop(S);

cout<<G.vexs[i].vex<<" ";//将栈顶的元素出栈且输出，即将入度为0的顶点输出

push(T,i);//为了求得拓扑序列的逆序列，将元素依次进栈就得到了逆序列

count++;//计数器加1

p=G.vexs[i].firstedge;//让p指向入度为0的顶点的第一个边表结点

while(p)

{

int k;

int dut;

dut=p->weight;

k=p->adjvex;

G.vexs[k].indegree--;//将入度为0的顶点的邻接点的入度减1

if(G.vexs[k].indegree==0)

push(S,k);//度减1后的顶点如果其入度为0，则将其入栈

if(ve[i]+dut>ve[k])

ve[k]=ve[i]+dut;//经过while循环，将顶点事件的所有邻接点的最早发生时间算出来，

//并且经过外层的while循环，不断地更新为较大的ve[k]值

p=p->nextedge;

}

}

cout<<endl;

if(count<G.vexnum)

cout<<"Network G has citcuits!"<<endl;

}

//求关键路径和关键活动

void CriticalPath(LGraph &G)

{

int i,j,k,dut;

int ee,el;

int vl[MAX\_VERTEX\_NUM];

EdgeNode \*p;

sqStack T;

initStack\_Sq(T);

TopoSort(G,T);

for(i=0;i<G.vexnum;i++)

vl[i]=ve[G.vexnum-1];//初始化顶点事件的最迟发生时间为汇点的最早发生时间

//因为求最迟发生时间是从汇点向源点开始计算的

while(T.top!=-1)

{//经过while循环，按堆栈顺序求出每个顶点的最迟发生时间

for(j=pop(T),p=G.vexs[j].firstedge; p ;p=p->nextedge)

{//这里应该注意for循环的机制：每一次循环都要判断一次条件，包括第一次

k=p->adjvex;

dut=p->weight;

if(vl[k]-dut<vl[j])

vl[j]=vl[k]-dut;//按堆栈T中事件的顺序，将该顶点事件的最迟发生时间经过for循环算出来，

//注意：经过for循环算出的是一个顶点的最迟发生时间

}

}

for(i=0;i<G.vexnum;i++)

{//依次遍历每一个活动

for(p=G.vexs[i].firstedge;p;p=p->nextedge)

{

k=p->adjvex;

dut=p->weight;

ee=ve[i];//求活动的最早开始时间

el=vl[k]-dut;//求活动的最迟开始时间

if(ee==el)

{//若两者相等，说明这这个活动为关键活动

cout<<"("<<G.vexs[i].vex<<","<<G.vexs[k].vex<<")"<<dut<<" ";

cout<<"ee="<<ee<<","<<"el="<<el<<endl;

}

}

}

}

void main()

{

freopen("in.txt","r",stdin);

LGraph G;

CreateDG\_AL(G,9,11);

CriticalPath(G);

}