算法 homework5

窦嘉伟 518021911160

Q1:

不存在；

对于有向无环图，我们维护一个入读为 0 的节点集合，该集合中的节点为 DFS 算法开始的节点，因此不论拓扑排序从该集合哪个节点开始，都不会影响 DFS 的可行性。

对于该集合外的节点，由拓扑排序的定义：若有边（u，v）则 u 在 v 之前，而 DFS 的执行结果用栈来实现先访问结束的点在后访问结束的点之后，所以对（u，v）必然有 u 在 v 之前，因此任意拓扑排序（有向无环图）都能被 DFS 实现。

Q2:

基本算法原理：假设每条边（u，v）为事件 u 的持续时间，从起点开始，每条边都有其最早开始时间，反过来，从终点开始计算，每条边都有其最晚开始事件才能保证终点事件如期进行，我们规定起点和终点的最晚开始时间等于其最早开始时间，利用原理：

关键路径上的每个点 x, s 到 x 的最大距离和 x 到 t 的最大距离之和始终相等。

可得，最早开始时间等于最晚开始时间的点集即为关键路径点集。

代码实现如下：

构造初始化函数：

```
graph::graph(){
fstream in("./data.in",ios::in);
in>>n;
in>>E;
int v,u,value;
table = new int*[n];
time = new int*[n];
for(int i=0;i<n;++i) {
table[i]=new int[n];
time[i] =new int[2];
}
for(int i=0;i<n;++i){
for(int m=0;m<n;++m){
table[i][m]=0;
}
}
for(int i=0;i<E;++i) {
in>>v;
in>>u;
```

```
in>>value;
table[v][u]=value;
}
in.close();
//创建邻接表，初始化开始时间表

U=new int[n];//子集
numU=0;
}
```

主要计算函数：

```
void graph::findway(int start,int end){
this->start=start;
this->end=end;
time[start][0]=0;//开始节点的最早开始时间为 0
ebway(start); //计算各店最早开始时间.
time[end][1]=time[end][0];//设置终点的最晚开始时间等于最早开始
时间
numU=0;
delete U;
U = new int[n];//清空 U
lbway(end);//计算各点最晚开始时间
output();//输出路径
}
```

```
void graph::lbway(int v){
U[numU]=v;
numU++;
for(int i=0;i<n;++i){
if(inU(i)) continue;//如果点在 U 里面，寻找下一个点

//当 i 不在 U 里面时，判断其子节点是否全在 U，若全在，则计算其最晚开始
时间并加入 U，否则过滤；
bool flag=true;//若子节点全在 U，记为 true；
for(int m=0;m<n;++m){
if(table[i][m]==0) continue;//两点间无边
if(!inU(m)) {
```

```
flag=false;//不全在U，直接退出，找下个点
break;
}
}
if(flag==false) continue;
for(int m=0;m<n;++m){
if(table[i][m]==0) continue;//两点间无边
if(time[i][1]==0) time[i][1]=time[m][1]-table[i][m];
else if((time[m][1]-table[i][m])<time[i][1])
time[i][1]=time[m][1]-table[i][m];//更新该点最晚开始时间
}
lbway(i);//将该点插入U，结束函数
return;
}
}
```

```
void graph::ebway(int v){//设置各点最早开始时间
U[numU]=v;
numU++;
//将该点放入子集U中，设置起点最早开始时间等于最晚开始时间等于0
for(int i=0;i<n;++i){
if(inU(i)) continue;//如果点在U里面，寻找下一个点
//当i不在U里面时，判断其父节点是否全在U，若全在，则计算其最大路径
并加入U，否则过滤；
bool flag=true;//若父节点全在U，记为true；
for(int m=0;m<n;++m){
if(table[m][i]==0) continue;//两点间无边
if(!inU(m)) {
flag=false;//不全在U，直接退出，找下个点
break;
}
}
if(flag==false) continue;
for(int m=0;m<n;++m){
if(table[m][i]==0) continue;//两点间无边
if((time[m][0]+table[m][i])>time[i][0])
time[i][0]=time[m][0]+table[m][i];//更新该点最早开始时间
}
```

```
ebway(i);//将该点插入 U，结束函数
return;
}
}
```

输出结果:

```
1 arguments:
argv[0] = '/Users/user/Desktop/Methods/hw5/keyroad/hw5/a'
最短路径:
1       9       2       4       6       7
Process exited with status 0
```