**河北师范大学普通本科学生学业考核**

装

订

线

线

**论文（报告、作业） 标准封面**

题目：图的基本操作及应用

|  |  |
| --- | --- |
| 课  程  信  息 | 开课学期 2022--2023学年第1学期  课程名称 数据结构与算法  开课学院 计算机与网络空间安全学院  任课教师 王秀青 |
| 学  生  信  息 | 姓 名 贾天乐  学 号 2021011206  学 院 计算机与网络空间安全学院  专 业 数据科学与大数据技术  年级班级 2021级 数据科学与大数据技术一 班 |
| 评议  结果 | 成绩  评阅人 评阅时间 2022年 月 |

**【问题描述】**

图的基本操作及应用

**【概要设计】**

分别以邻接矩阵和邻接表为存储结构构造 无向图和有向图，

**【详细设计】**

1．依次输入顶点数目，边的数目，各条边的信息建立无向图的邻接矩阵和邻接表；

2．依次输入顶点数目，弧的数目，各条弧的信息以及弧上的权的信息建立有向网的邻接矩阵和邻接表；

3．输出邻接矩阵存储的无向图的深度优先遍历和广度优先遍历结果；

4．输出邻接表存储的有向图的深度优先遍历和广度优先遍历结果；

5．以邻接矩阵为存储结构，实现Prim算法，输出最小生成树的各条边；

6．以邻接表为存储结构，实现拓扑排序算法，输出拓扑排序序列；

7．以邻接矩阵为存储结构，输出一个AOE网的关键路径；

**【算法实现】**

void ToplogicalSort(ALGraph G)

{

int a[G.vexnum];

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

{

a[i]=0;

}

ArcNode \*p;

for(int j=0;j<G.vexnum;j++)

{

for(p=G.vertices[j].firstarc;p;p=p->nextarc)

a[p->adjvex]++;

}

Stack S;

InitStack(S);

int count=0;

for(int u=0;u<G.vexnum;u++)

{

if(a[u]==0)

Push(S,u);

}

int n,k;

while(S.top!=0)

{

Pop(S,n);

printf("%c ",G.vertices[n].data);

count++;

ArcNode \*p;

for(p=G.vertices[n].firstarc;p;p=p->nextarc)

{

k=p->adjvex;

a[k]--;

if(a[k]==0)

Push(S,k);

}

}

printf("\n");

if(count<G.vexnum)

printf("排序出错\n");

}

bool ToplogicalOrder(ALGraph G,Stack &T,int \*a,int \*Ve)

{

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

{

a[i]=0;

Ve[i]=0;

}

ArcNode \*p;

for(int j=0;j<G.vexnum;j++)

{

for(p=G.vertices[j].firstarc;p;p=p->nextarc)

a[p->adjvex]++;

}

Stack S;

InitStack(S);

InitStack(T);

int count=0;

for(int u=0;u<G.vexnum;u++)

{

if(a[u]==0)

Push(S,u);

}

int n,k;

while(S.top!=0)

{

Pop(S,n);

Push(T,n);

count++;

ArcNode \*p;

for(p=G.vertices[n].firstarc;p;p=p->nextarc)

{

k=p->adjvex;

a[k]--;

if(Ve[k]<Ve[n]+p->lenth)

Ve[k]=Ve[n]+p->lenth;

if(a[k]==0)

Push(S,k);

}

}

printf("\n");

if(count<G.vexnum)

{

printf("排序出错\n");

return false;

}

return true;

}

int Criticalpath(ALGraph G)

{

int a[G.vexnum],Ve[G.vexnum];

Stack S;

if(!ToplogicalOrder(G,S,a,Ve)) return 0;

int Vl[G.vexnum];

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

{

Vl[i]=Ve[i];

}

int j,k,l;

char tag;

ArcNode \*p;

while(S.top!=0)

{

Pop(S,j);

for(p=G.vertices[j].firstarc;p;p=p->nextarc)

{

if(Vl[p->adjvex]-p->lenth<Vl[j])

Vl[j]=Vl[p->adjvex]-p->lenth;

}

}

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

{

for(p=G.vertices[j].firstarc;p;p=p->nextarc)

{

int e=Ve[j];

l=Vl[p->adjvex]-p->lenth;

tag=(e==l)?'\*':' ';

printf("%c%c%c ",G.vertices[j].data,tag,G.vertices[p->adjvex].data);

}

}

printf("\n");

return 1;

}

void Prim(MGraph G, int u)

{

closedge C;

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

{

if(i!=u)

{

C[i].adjvex=u;

C[i].lowcost=G.arcs[u][i];

}

}

C[u].lowcost=0;

int k,l;

for(int j=1;j<G.vexnum;j++)

{

k=INFINITY,l=-1;

for(int q=0;q<G.vexnum;q++)

{

if((C[q].lowcost!=0)&&(C[q].lowcost<k))

{

k=C[q].lowcost;

l=q;

}

}

printf("第%d条边为%c到%c,长度为%d\n",j,G.vex[C[l].adjvex],G.vex[l],C[l].lowcost);

C[l].lowcost=0;

for(int o=0;o<G.vexnum;o++)

{

if(G.arcs[l][o]!=0&&G.arcs[l][o]<C[o].lowcost)

{

C[o].adjvex=l;

C[o].lowcost=G.arcs[l][o];

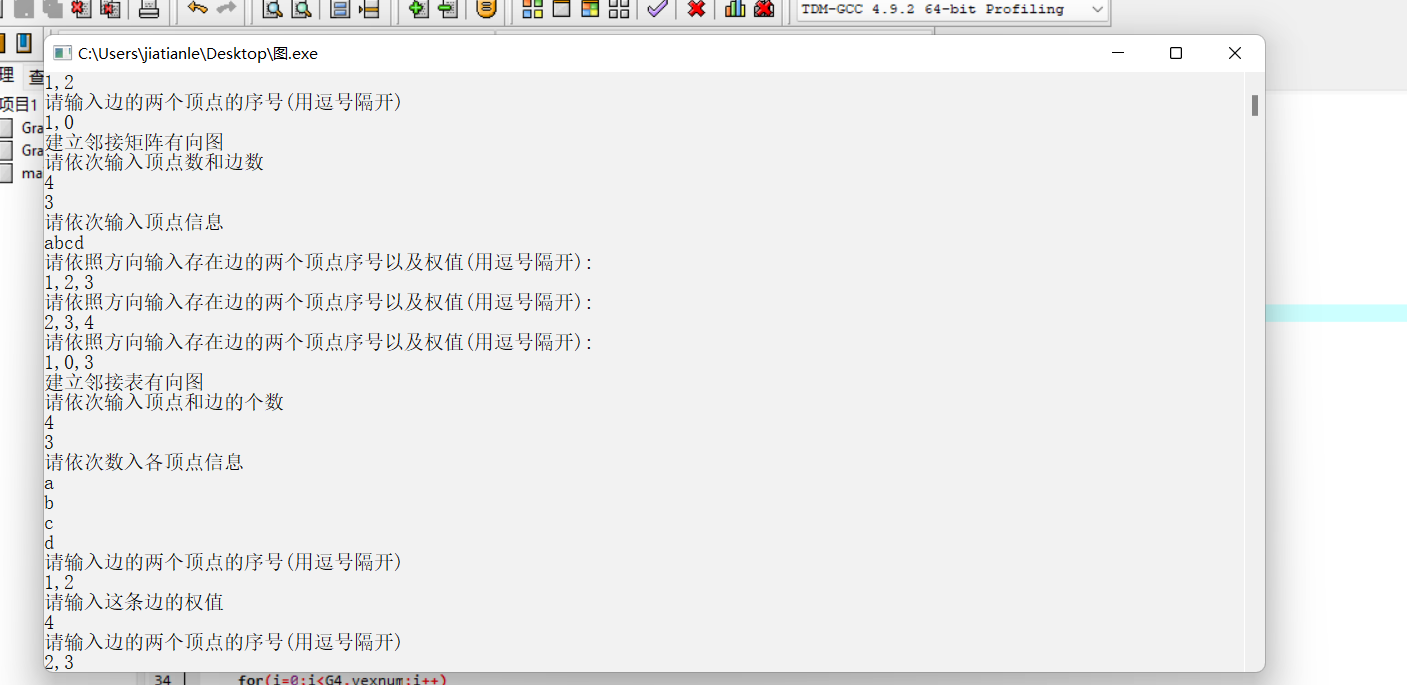
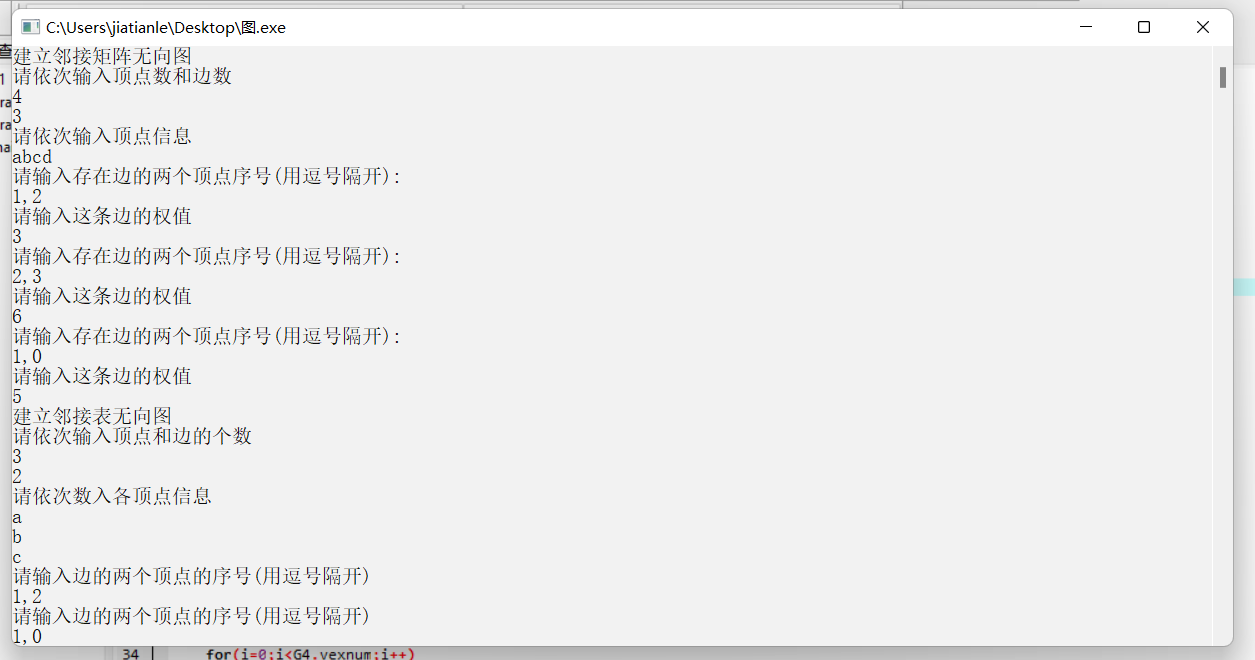
}

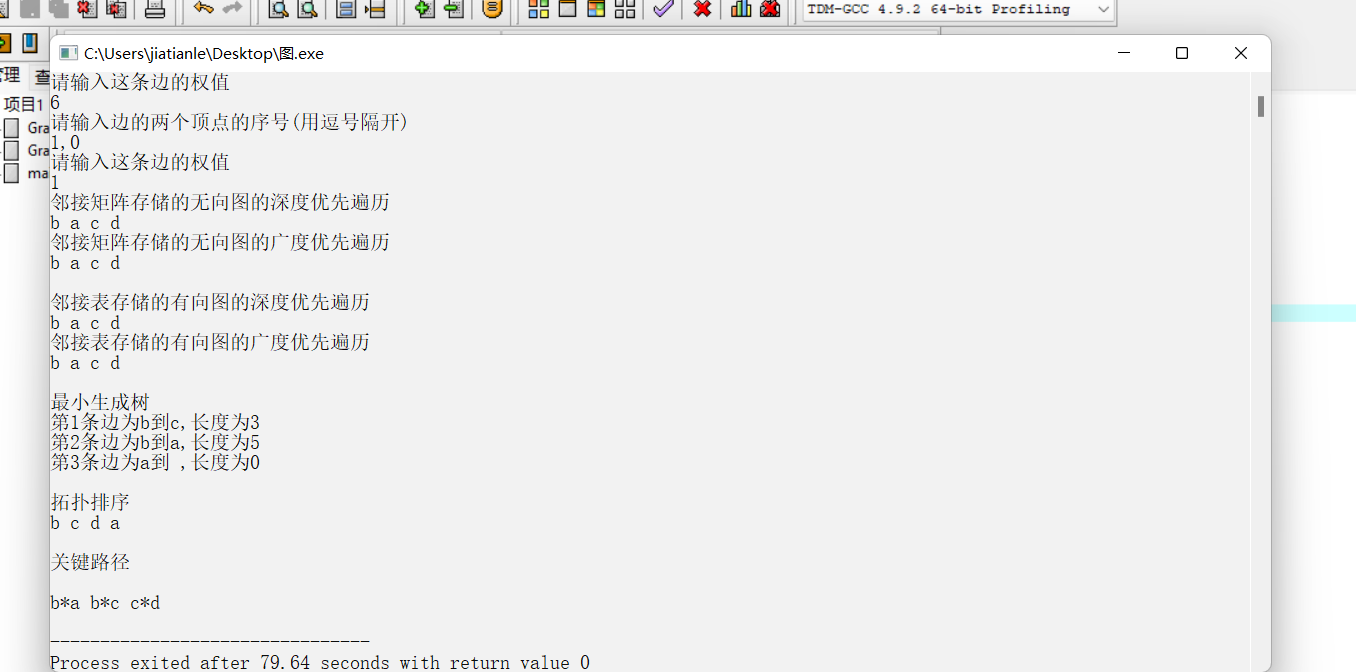
}

}

}

**【运行结果】**





**【总结与分析】**

**本次实验很复杂，要注意地址的申请，防止地址混乱**