计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：朱龙康 学号：1824120002 专业：计算机科学与技术\_ 年级：\_2018级

课程：数据结构 主讲教师：\_袁彩虹\_ 辅导教师：\_\_袁彩虹\_\_

实验时间：\_\_2020\_年 \_5\_月 \_\_24\_日 \_上\_午\_8\_时至\_10\_时，实验地点\_\_\_\_\_\_\_

实验题目： 图的操作

实验目的： （1）掌握图的邻接矩阵和邻接表存储方式；

（2）掌握图的遍历算法；

（3）掌握图的实际应用——最短路径算法。

实验环境（硬件和软件） CodeBlocks

实验内容：

*（1）采用邻接矩阵/邻接表建立图；*

*（2）采用深度优先/广度优先搜索方式遍历图；*

*（3）编程实现Dijkstra最短路径算法。*

实验步骤：

**//---------------------------图的邻接矩阵存储结构**

**#define MAXINT 32767 //极大值**

**#define MVNUM 100 //最大顶点数**

**typedef struct**

**{**

**char vexs[MVNUM]; //顶点表**

**int arcs[MVNUM][MVNUM]; //邻接矩阵**

**int vexnum,arcnum; //图的当前点数和边数**

**}AMGraph;**

**//用邻接矩阵创建无向网**

**int Location(AMGraph G, char v){//获取v对应下标函数**

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

**if(G.vexs[i]==v)return i;**

**}**

**return -1; //未找到 返回-1**

**}**

**bool CreateUDN(AMGraph &G){**

**cout<<"输入顶点数和边数"<<endl;**

**cin>>G.vexnum>>G.arcnum; //输入总顶点数，总边数**

**cout<<"输入顶点的信息"<<endl;**

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

**cin>>G.vexs[i]; //输入点的信息**

**}**

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

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

**G.arcs[i][j]=MAXINT; //初始化邻接矩阵，所有边设为极大值**

**}**

**}**

**char v1,v2;**

**int w;**

**int i,j;**

**for(int k=0;k<G.arcnum;k++){**

**cout<<"输入一条边两个顶点和对应权值"<<endl;**

**cin>>v1>>v2>>w; //依次输入一条边对应的顶点和权值**

**i=Location(G,v1);j=Location(G,v2); //确定v1和v2的下标**

**G.arcs[i][j]=G.arcs[j][i]=w; //置<v1,v2>和<v2,v1>为权值**

**}**

**}**

**//-----------------------图的邻接表存储结构**

**#define MVNUM 100 //最大顶点数**

**typedef struct ArcNode //边节点**

**{**

**int adjvex; //该边指向的顶点位置**

**struct ArcNode \* next; //指向下一条边**

**}ArcNode;**

**typedef struct VNode**

**{**

**char data;**

**ArcNode \*firstarc;**

**}VNode,AdjList[MVNUM];**

**typedef struct {**

**AdjList vertices;**

**int vexnum,arcnum;**

**}ALGraph;**

**//用邻接表创建无向图**

**int Location2(ALGraph G,char v){//查找 v在G中的位置**

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

**if(G.vertices[i].data==v)return i;**

**}**

**return -1;//未找到**

**}**

**bool CreateUDG(ALGraph &G){**

**cin>>G.vexnum>>G.arcnum;//输入顶点数，边数**

**for(int i=0;i<G.vexnum;i++){//输入各点，构造表头节点表**

**cin>>G.vertices[i].data;**

**G.vertices[i].firstarc=NULL;**

**}**

**char v1,v2;**

**int i,j;**

**ArcNode\* p1,\*p2;**

**for (int k = 0; k <G.arcnum ; ++k) {//构造各边**

**cin>>v1>>v2;//输入一条边依附的两个节点**

**i=Location2(G,v1);j=Location2(G,v2);//确定v1 v2在G中的位置**

**p1=new ArcNode;//生成新边节点**

**p1->adjvex=j;//临接点序号为j**

**p1->next=G.vertices[i].firstarc;G.vertices[i].firstarc=p1;//将新节点插入vi的边表头**

**p2=new ArcNode;//生成新边节点**

**p2->adjvex=i;//临接点序号为i**

**p2->next=G.vertices[j].firstarc;G.vertices[j].firstarc=p2;//将新节点插入vj的边表头**

**}**

**return true;**

**}**

**//-------------DFS**

**//邻接矩阵DFS**

**void DFS\_AM(AMGraph G,int v,bool visited[])//从第v个顶点出发深度优先**

**{**

**cout<<v;visited[v]=true;//访问第v个节点，并标记**

**for(int k=0;k<G.vexnum;k++){**

**if((G.arcs[v][k]!=MAXINT)&&!visited[k])DFS\_AM(G,k,visited);**

**}**

**}**

**//------------BFS**

**//邻接矩阵BFS**

**//先定义队列**

**#define MAXSIZE 50//队列最大长度**

**typedef struct**

**{**

**int \*base;//存储空间的基地址**

**int front;//头指针**

**int rear;//尾指针**

**}SqQueue;**

**//初始化**

**bool InitQueue(SqQueue &q){**

**q.base=new int[MAXSIZE];//申请空间**

**if(!q.base)return false;//申请失败**

**q.front=q.rear=0;//队列空 头尾指针置零**

**return true;**

**}**

**//入队**

**bool InQueue(SqQueue &q,int e){**

**if((q.rear+1)%MAXSIZE==q.front)return false;//队满**

**q.base[q.rear]=e;//将e入队**

**q.rear=(q.rear+1)%MAXSIZE;//尾指针移动**

**return true;**

**}**

**//出队**

**bool OutQueue(SqQueue &q,int &m){**

**if(q.rear==q.front)return false;//队空**

**m=q.base[q.front];//将m传给参数带出**

**q.front=(q.front+1)%MAXSIZE;//头指针改变**

**return true;**

**}**

**int FirstAdjVex(AMGraph G,int u)//查找和u相连的第一个邻接点序号**

**{**

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

**if(G.arcs[u][i]!=MAXINT)return i;**

**}**

**return -1;//未找到**

**}**

**int NextAdjVex(AMGraph G,int u,int w)//查找和u相连的 在u后的顶点序号**

**{**

**for (int i = w+1; i <G.vexnum ; ++i) {**

**if(G.arcs[u][i]!=MAXINT)return i;**

**}**

**return -1;//未找到**

**}**

**void BFS\_AM(AMGraph G,int v,bool visited[],SqQueue Q)**

**{**

**cout<<v;visited[v]=true;//访问第v个顶点，并标记**

**InitQueue(Q);//初始化队列**

**InQueue(Q,v);//v入队**

**while (!(Q.rear==Q.front))//队不空**

**{**

**int u;**

**OutQueue(Q,u); //队头元素出队，赋给u**

**for(int w=FirstAdjVex(G,u);w>=0;w=NextAdjVex(G,u,w))**

**if(!visited[w])**

**{**

**cout<<w;visited[w]=true;**

**InQueue(Q,w);**

**}**

**}**

**}**

**//--------------------------------迪杰斯特拉最短路径**

**void ShortesPath\_DIJ(AMGraph G,int v0)//求v0到其余各点的最短路径**

**{**

**int n=G.vexnum; //n为G中顶点个数**

**bool S[n];**

**int D[n];**

**int Path[n];**

**for (int i = 0; i <n ; ++i) {**

**S[i]=false;**

**D[i]=G.arcs[v0][i];**

**if(D[v0]<MAXINT)Path[i]=v0;**

**else Path[i]=-1;**

**}**

**S[v0]=true;**

**D[v0]=0;**

**//-------------------**

**int v;**

**for (int i = 0; i <n ; ++i) {**

**int min=MAXINT;**

**for (int w = 0; w <n ; ++w) {**

**if(!S[w]&&D[w]<min){v=w;min=D[w];}**

**}**

**S[v]=true;**

**for (int w = 0; w <n ; ++w) {**

**if(!S[w]&&(D[v]+G.arcs[v][w])<D[w])**

**{**

**D[w]=D[v]+G.arcs[v][w];**

**Path[w]=v;**

**}**

**}**

**}**

**for (int j = 0; j < n; ++j) {**

**cout<<"到"<<G.vexs[j]<<"最短"<<D[j]<<" ";**

**}**

**}**

**int main() {**

**AMGraph G;**

**CreateUDN(G);**

**bool visited[G.vexnum];**

**for(int i=0;i<G.vexnum;i++)visited[i]=false;**

**SqQueue Q;**

**cout<<"DFS"<<endl;**

**DFS\_AM(G,0,visited);**

**cout<<endl;**

**for(int i=0;i<G.vexnum;i++)visited[i]=false;**

**cout<<"BFS"<<endl;**

**BFS\_AM(G,0,visited,Q);**

**cout<<endl;**

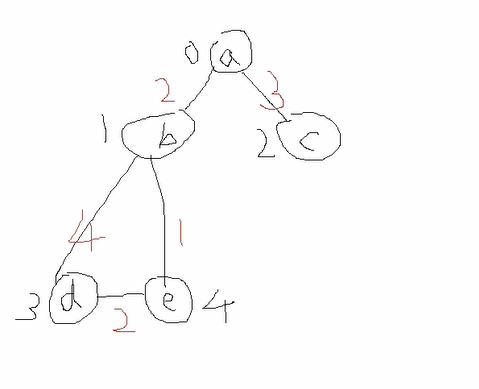
**cout<<"迪杰斯特拉"<<endl;**

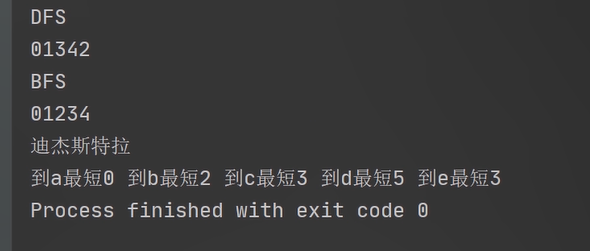
**ShortesPath\_DIJ(G,0);**

**return 0;**

**}**

实验数据记录：





问题讨论：

还有没有别的方法实现最短路径？