学号 WA2214014 专业 人工智能 姓名 杨跃浙

实验日期 **7月1号**  教师签字 成绩

实验报告

【实验名称】 图

【实验目的】

掌握图的两种储存方式：邻接矩阵表示法借助二维数组来表示元素之间的关系；邻接表属于链式存储结构。

图的两种遍历：深度优先遍历和广度优先遍历。

最小生成树算法：普里姆算法和克鲁斯卡算法。

【实验原理】

邻接矩阵表示法借助二维数组来表示元素之间的关系；邻接表属于链式存储结构，通过链表表示顶点元素和边之间的关系。

图的深度优先遍历（DFS）

最小生成树算法（普里姆算法)

【实验内容】

使用邻接矩阵表示一个如图所示无向有权权图，增加顶点,增加边，输出图中的所有顶点和边

使用邻接链表表示一个有向有权图，增加顶点，增加有向弧，输出图中的所有顶点和弧

使用DFS遍历题目1所生成的图

使用Prim算法生成题目1的最小生成树

#include <iostream>

using namespace std;

#define OK 1

#define ERROR 0

#define OVERFLOW -2

#define MVNum 100

#define MAXINT 32767

typedef int Status;

typedef int VerTexType;

typedef int ArcType;

typedef struct

{

VerTexType vexs[MVNum];

ArcType arcs[MVNum][MVNum];

int vexnum, arcnum;

}AMGraph;

bool visited[MVNum \* MVNum] = { false };

int LocateVex\_AM(AMGraph G, int v)

{

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

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

return MAXINT;

}

Status CreateUDN(AMGraph& G)

{

cin >> G.vexnum >> G.arcnum;

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

cin >> G.vexs[i];

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

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

G.arcs[i][j] = MAXINT;

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

{

int v1, v2, w;

cin >> v1 >> v2 >> w;

int i = LocateVex\_AM(G, v1);

int j = LocateVex\_AM(G, v2);

G.arcs[i][j] = w;

G.arcs[j][i] = w;

}

return OK;

}

void Print\_AMGraph(AMGraph G)

{

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

{

cout << G.vexs[i] << endl;

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

if (G.arcs[i][j] != MAXINT) cout << G.vexs[i] << "->" << G.vexs[j] << ' ' << G.arcs[i][j] << endl;

}

}

typedef int OtherInfo;

typedef struct ArcNode

{

int adjvex;

struct ArcNode\* nextarc;

OtherInfo info;

};

typedef struct VNode

{

VerTexType data;

ArcNode\* firstarc;

}VNode, AdjList[MVNum];

typedef struct

{

AdjList vertices;

int vexnum, arcnum;

}ALGraph;

int LocateVex\_AL(ALGraph G, int v)

{

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

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

return MAXINT;

}

Status CreatDG(ALGraph& G)

{

cin >> G.vexnum >> G.arcnum;

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

{

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

G.vertices[i].firstarc = NULL;

}

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

{

int v1, v2, w, i, j;

cin >> v1 >> v2>>w;

i = LocateVex\_AL(G, v1);

j = LocateVex\_AL(G, v2);

ArcNode\* p1 = new ArcNode;

p1->adjvex = j;

p1->info = w;

p1->nextarc = G.vertices[i].firstarc;

G.vertices[i].firstarc = p1;

}

return OK;

}

void Print\_ALGraph(ALGraph G)

{

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

{

cout << G.vertices[i].data << endl;

ArcNode\* p;

p = G.vertices[i].firstarc;

while (p)

{

cout << G.vertices[i].data << "->" << p->adjvex << ' ' << p->info << endl;

p = p->nextarc;

}

}

}

void DFS\_AM(AMGraph G, int v)

{

cout << G.vexs[v]<<' ';

visited[v] = true;

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

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

}

struct

{

VerTexType adjvex;

ArcType lowcost;

}closedge[MVNum];

int Min(int num)

{

int k = -1;

for (int i = 0; i < num; ++i)

if (((closedge[i].lowcost!=0)&&(k==-1))||((closedge[i].lowcost != 0) && (closedge[i].lowcost <closedge[k].lowcost))) k = i;

return k;

}

void MiniSpanTree\_Prim(AMGraph G, VerTexType u)

{

int k = LocateVex\_AM(G, u);

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

if (j != k) closedge[j] = { u,G.arcs[k][j] };

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

{

k = Min(G.vexnum);

int u0 = closedge[k].adjvex;

int v0 = G.vexs[k];

cout << u0 <<"->" << v0 <<" " << closedge[k].lowcost << '\t';

closedge[k].lowcost = 0;

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

if (G.arcs[k][j] < closedge[j].lowcost)

closedge[j] = { G.vexs[k], G.arcs[k][j] };

}

}

int main()

{

AMGraph AMG;

ALGraph ALG;

CreateUDN(AMG);

Print\_AMGraph(AMG);

CreatDG(ALG);

Print\_ALGraph(ALG);

DFS\_AM(AMG, 0);

cout << endl;

MiniSpanTree\_Prim(AMG, 1);

cout << endl;

return 0;

}

/\*

6 10

1 2 3 4 5 6

1 3 6

1 2 9

1 4 3

2 4 5

2 5 8

3 5 9

4 6 7

5 6 4

3 6 5

4 3 2

6 8

0 1 2 3 4 5

0 5 100

0 4 30

0 2 10

1 2 5

2 3 50

4 3 20

3 5 10

4 5 60

\*/

【小结或讨论】

通过该次实验我掌握了图的基本算法，包括图的创建邻接矩阵创建无向带权图和邻接表创建有向有权图，并能够应用深度优先算法搜索图，能运用普里姆算法生成最小生成树等。