**实验六 无向连通图的建立和遍历**

应用技术学院 学院 计算机科学与技术 专业 19计科1班 班

学号 201833050025 姓名 成凯

一．实习目的

1. 掌握无向图的建立；

2. 掌握无向连通图的深度优先遍历递归算法；

3. 掌握无向连通图的广度优先遍历算法；

二．实习内容

1. 输入无向图的顶点和边，在计算机中存储无向图的顶点信息和邻接矩阵；

2. 编写图的深度优先遍历程序，完成对无向图的深度优先遍历，并输出深度优先遍历序列；

3. 编写图的广度优先遍历程序，完成对无向图的广度优先遍历，并输出广度优先遍历序列；

三、实验步骤

1. 无向图的顶点表定义；

2. 无向图的邻接矩阵定义；

3. 创建无向图并存储相应的顶点表和邻接矩阵；

4. 使用深度优先遍历算法对创建的无向图进行深度优先遍历，并输出结果；

5. 使用广度优先遍历算法对创建的无向图进行广度优先遍历，并输出结果；

四．程序运行

1.程序代码

#include <printf.h>

#include <stdlib.h>

#include "ds-queue.c"

#define OK 1

#define ERROR 2

#define INFINITY 65535 // 最大值

#define MAX\_VERTEX\_NUM 20 // 最大顶点个数

typedef char VertTexType; // 顶点数据类型

typedef int ArcType; // 边的权值类型

typedef struct {

VertTexType vexs[MAX\_VERTEX\_NUM]; // 顶点表

ArcType arcs[MAX\_VERTEX\_NUM][MAX\_VERTEX\_NUM]; // 邻接矩阵

int vexnum, arcnum; // 图的当前顶点数和边数

} MGraph;

int createUDN(MGraph\*);

void printGraph(MGraph\*);

void bfsTraverse(MGraph\*);

void dfsTraverse(MGraph\*);

int main() {

MGraph\* g = (MGraph\*) malloc(sizeof(MGraph));

createUDN(g);

printGraph(g);

bfsTraverse(g);

dfsTraverse(g);

return 0;

}

/\*\*

\* 返回up在顶点表中的位置

\*/

int locateVex(MGraph graph, VertTexType up) {

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

if (up == graph.vexs[i]) {

return i;

}

}

return -1;

}

/\*\*

\* 创建无向网 - 邻接矩阵表示法

\*/

int createUDN(MGraph\* g) {

printf("请输入顶点数和边数：\n");

scanf("%d,%d", &g->vexnum, &g->arcnum);

// 输入顶点信息，建立顶点表

printf("请输入顶点信息：\n");

for (int i = 0; i < g->vexnum; ++i) {

scanf(" %c", &g->vexs[i]);

}

printf("输入的顶点信息：");

for (int i = 0; i < g->vexnum; ++i) {

printf("%c ", g->vexs[i]);

}

printf("\n");

// 初始化邻接矩阵

for (int i = 0; i < g->vexnum; ++i) {

for (int j = 0; j < g->vexnum; ++j) {

g->arcs[i][j] = INFINITY;

}

}

// 读取边，建立邻接矩阵

int w, i, j;

VertTexType v1, v2;

for (int k = 0; k < g->arcnum; ++k) {

printf("输入一条边依附的顶点和权值w：\n");

scanf(" %c %c %d", &v1, &v2, &w);

i = locateVex(\*g, v1);

j = locateVex(\*g, v2);

// 没找到顶点

if (i == -1 || j == -1) {

printf("未找到顶点！");

exit(ERROR);

}

// 边(v1,v2)的权值置为w

// 无向图，矩阵对称 (v1,v2) (v2,v1)的权值相同

g->arcs[i][j] = g->arcs[i][j] = w;

}

return OK;

}

/\*\*

\* 广度优先遍历

\*/

void bfsTraverse(MGraph\* g) {

printf("\n");

printf("广度优先遍历：");

// 初始化访问标志

int visited[100] = {0};

// 置空的辅助队列

SqQueue queue;

initQueue(&queue);

// 对每一个顶点做循环

for (int i = 0; i < g->vexnum; ++i) {

// 未访问过就处理

if (!visited[i]) {

// 标记访问过

visited[i] = 1;

// 输出顶点

printf("%c ", g->vexs[i]);

// 将此顶点入队列

enQueue(&queue, i);

while (!isQueueEmpty(&queue)) {

// 若当前队列不为空

// 将对头元素出队列并赋值给i

deQueue(&queue, &i);

for (int j = 0; j < g->vexnum; ++j) {

// 判断其他顶点是否与当前顶点存在 且 未访问过

if (g->arcs[i][j] == 1 && !visited[j]) {

// 将找到的此顶点标记为已访问

visited[j] = 1;

// 输出顶点

printf("%c ", g->vexs[j]);

enQueue(&queue, j);

}

}

}

}

}

printf("\n");

}

// 初始化访问标志

int visitedDfs[100] = {0};

void dfs(MGraph\* graph, int i) {

int j;

visitedDfs[i] = 1;

// 打印顶点

printf("%c ", graph->vexs[i]);

for (j = 0; j < graph->vexnum; j++) {

if (graph->arcs[i][j] == 1 && !visitedDfs[j]) {

// 对为访问的邻接顶点递归调用

dfs(graph, j);

}

}

}

/\*\*

\* 深度优先遍历

\*/

void dfsTraverse(MGraph\* graph) {

printf("\n");

printf("广度优先遍历：");

int i;

for (i = 0; i < graph->vexnum; i++) {

// 初始所有顶点状态都是未访问过状态

visitedDfs[i] = 0;

}

for (i = 0; i < graph->vexnum; i++) {

if (!visitedDfs[i]) {

// 对未访问过的顶点调用DFS，若是连通图,只会执行一次

dfs(graph, i);

}

}

}

/\*\*

\* 打印图(邻接矩阵)

\*/

void printGraph(MGraph\* g) {

printf("\n");

printf("输出：\n");

for (int i = 0; i < g->vexnum; i++) {

for (int j = 0; j < g->vexnum; j++) {

printf("%d ", g->arcs[i][j]);

}

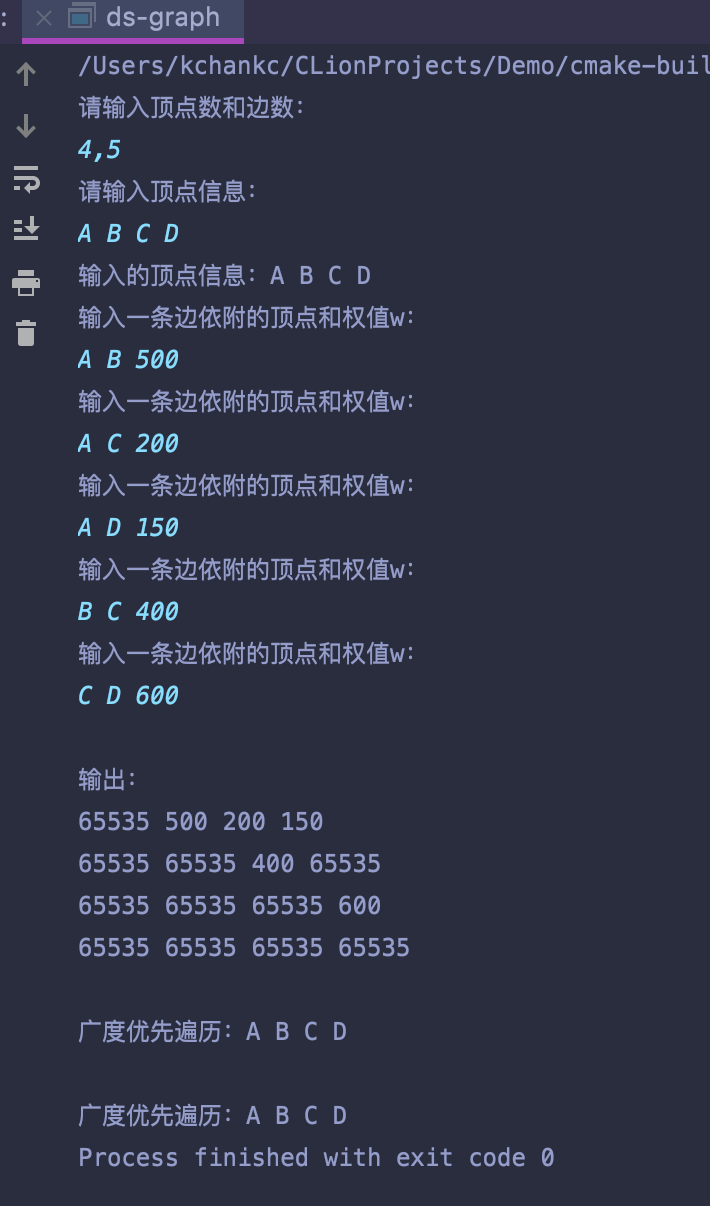
printf("\n");

}

}

2.运行程序

3.运行结果（截图）



五．实验小结

通过本次实验，我了解了无向图的顶点表以及邻接矩阵的定义与创建；学习了使用深度优先遍历算法以及广度优先算法对无向图进行深度优先遍历，但仍需要加强理解。同时回顾了队列的使用，加深了理解。