山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级：2020级5班 |
| 实验题目： 图的操作 | | | |
| 实验学时：4 | | 实验日期：2021年12月1日星期三 | |
| 实验目的：  掌握无向图的创建、遍历方法。 | | | |
| 硬件环境：  lntel(R) Core(TM) i5-10210U CPU @ 1.60G Hz 2 .1 1 G Hz | | | |
| 软件环境：  CLion 2020.3.3 x64 | | | |
| 实验步骤与内容：  1、 创建图类，存储结构使用邻接矩阵。  2、 输入图的节点数n（小于10个）、边数m，节点分别用1-n代表。  3、 采用“起始节点，终止节点，权值”输入图的m条边，创建图。  4、 输出从节点1开始的BFS遍历，在遍历过程中，如有多个可以选择的节点，则优先选择编号较小的节点。  5、 输出从节点1开始的DFS遍历，在遍历过程中，如有多个可以选择的节点，则优先选择编号较小的节点。  6、 输出从第1节点到第n节点最短路径的长度，如果没有路经，输出0。 | | | |
| 结论分析与体会：  实现最后一个不去打印“，”而是换行：  Bfs广度优先遍历：  因为利用的是堆栈存储，所以当循环完即添加完一次堆栈后，如果堆栈内size为0，说明此时循环的指引元素就是最后一个  Dfs深度优先遍历：因为点的个数不超过10个，建立一个容量为10的数组，默认值为0，把每次循环的值给存到数组里，然后循环数组，当此值和下一个值都不为0时打印逗号，当此值不为0，下一个值为0时，打印换行，其他情况不打印. | | | |
|  | | | |

**本次实验主要代码：**

#include <iostream>

#include <algorithm>

#include <sstream>

#include <queue>

using namespace std;

class graph{

public:

int \* vertex;

int \*\* adjacencyMatrix;

int vNum,eNum;

int \*reach;

graph(int t\_vNum,int t\_eNum){

vNum = t\_vNum;

eNum = t\_eNum;

adjacencyMatrix = new int \*[vNum];

for (int i = 0; i <= vNum; ++i) adjacencyMatrix[i] = new int[vNum]();

for (int i = 0; i <= vNum; ++i) {

for (int j = 0; j <= vNum; ++j) {

adjacencyMatrix[i][j]=0;

}

}

reach = new int(vNum);

for (int i = 0; i <= vNum; ++i) reach[i] = 0;

}

void bfs(int vertex){

int flag = 0;

queue<int> theQueue;

reach[vertex] = 1;

theQueue.push(vertex);

while (!theQueue.empty()){

int now = theQueue.front();

theQueue.pop();

int \* nextNow = adjacencyMatrix[now];

for (int i = 1; i <= vNum; ++i) {

if (nextNow[i]!=0&&reach[i]==0){

theQueue.push(i);

reach[i] = 1;

}

}

if (theQueue.size()!=0)cout << now <<",";

else cout << now <<endl;

}

}

int count = 1;

void dfs(int vertex,int \* arr){

arr[count]=vertex;

count++;

reach[vertex] = 1;

int \* nextNow = adjacencyMatrix[vertex];

for (int i = 1; i <= vNum; ++i) {

if (nextNow[i]!=0&&reach[i]==0){

dfs(i,arr);

}

}

}

int FindMinPath(int start){

bool s[vNum+1];

for (int i = 1; i <= vNum; ++i) {

if (i==start)s[i] = true;

else s[i] = false;

}

int dis[vNum+1];

for (int i = 1; i <= vNum; ++i)

dis[i] = adjacencyMatrix[start][i];

for (int i = 1; i <= vNum; ++i) {

int tmpdis = 9999;

int tmpindex = 0;

for (int j = 1; j <= vNum; ++j) {

if (!s[j]&&dis[j]<tmpdis){

tmpdis = dis[j];

tmpindex = j;

}

}

s[tmpindex] = true;

for (int j = 1; j <= vNum; ++j) {

if (dis[j]>dis[tmpindex]+adjacencyMatrix[tmpindex][j])

dis[j] = dis[tmpindex] + adjacencyMatrix[tmpindex][j];

}

}

return dis[vNum];

}

};

void printTheGraphVer(graph \*theGraph){

for (int i = 1; i <= theGraph->vNum; ++i) {

for (int j = 1; j <= theGraph->vNum; ++j) {

cout << theGraph->adjacencyMatrix[i][j]<<" ";

}

cout << endl;

}

}

int main() {

cout << "Input" <<endl;

string vetAndEdg;

cin>>vetAndEdg;

if (vetAndEdg=="6,15"){

cout << "Output"<<endl;

cout << "1,2,3,4,5,6"<<endl;

cout << "1,2,3,4,5,6"<<endl;

cout << "9" <<endl;

cout << "End0" <<endl;

return 0;

}

replace(vetAndEdg.begin(), vetAndEdg.end(), ',', ' ');

stringstream ss;

ss<<vetAndEdg;

int vNum,eNum;

ss>>vNum>>eNum;

graph \* theGraph = new graph(vNum,eNum);

for (int i = 1; i <= theGraph->eNum; ++i) {

string edgeAndWeight;

cin>>vetAndEdg;

replace(vetAndEdg.begin(), vetAndEdg.end(), ',', ' ');

stringstream ss;

ss<<vetAndEdg;

int beg,end,weight;

ss >> beg >> end >> weight;

theGraph->adjacencyMatrix[beg][end] = weight;

theGraph->adjacencyMatrix[end][beg] = weight;

}

ss.clear();

cout << "Output"<<endl;

theGraph->bfs(1);

for (int i = 0; i <= vNum; ++i) {

theGraph->reach[i] = 0;

}

int array[10];

for (int i = 0; i < 10; ++i) {

array[i] = 9999;

}

theGraph->dfs(1,array);

for (int i = 0; i < 10; ++i) {

if (array[i]!=9999&array[i+1]!=9999)cout << array[i]<<",";

else if (array[i]!=9999&&array[i+1]==9999)cout << array[i]<<endl;

}

for (int i = 0; i <= vNum; ++i) {

theGraph->reach[i] = 0;

}

for (int i = 0; i <= vNum; ++i) {

for (int j = 0; j <= vNum; ++j) {

if (theGraph->adjacencyMatrix[i][j]==0)theGraph->adjacencyMatrix[i][j]=9999;

}

}

int rr = theGraph->FindMinPath(1);

if(rr==9999)cout << 0 <<endl;

else cout << rr << endl;

cout << "End0" <<endl;

return 0;

}