附件1：实验报告中的诚信设计

数据结构与算法导论

实验报告

实验题目： 邻接矩阵的实现

姓 名： 梁睿鸣

学 号： 2020210139

日 期： 2021/6/17

自我评分： A+ 【 X 】 95

自我评分说明：A+，A，B+，B，B-，C，D，分别对应分数95、90、85、80、75、70、60

诚信声明

本人郑重承诺：本实验程序和实验报告均是本人独立学习和工作所获得的成果。尽我所知，实验报告中除特别标注的地方外，不包含其他同学已经发表或撰写过的成果；实验程序中对代码工作的任何帮助者所作的贡献均做了明确的说明，并表达了谢意。

如有抄袭，本人原因承担因此而造成的任何后果。

特此声明。

签名：梁睿鸣

日期：2021/6/17

程序引用说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 引用项 | 来源 | 代码引用行数 |
| 1 | 构造函数 | 课本 | 14 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 小计 1 | | | 14 |

总代码行数 105 ； 引用占比 13.33%

1、实验简介

根据图的抽象数据类型的定义，使用邻接矩阵或邻接表实现一个图。

图的基本功能：

1、图的建立

2、图的销毁

3、深度优先遍历图

4、广度优先遍历图

5、其他：比如连通性判断等自定义操作

编写测试main()函数测试图的正确性

2、程序框架

【实验程序共包含哪些函数，一一列举函数名和函数功能】

MGraph(int ver, T v[], int arc, int a[][2]); //构造函数

void DFS(int v); //深度优先搜索

void BFS(int v); //广度优先搜索

bool IsConnected(int v1,int v2); //判断连通性

3、关键代码实现

【哪些函数是你认为最能体现自己工作成果的函数，说明函数实现基本思想（可用文字或图表示），以及具体的实验步骤（用伪代码或带注释代码）】

3.1 DFS(int v)

基本思想：

先遍历第一个节点并入栈

每次从栈顶获取一个节点并尝试前进

每前进一步都遍历所在节点

直到不能再前进，于是回退，再尝试前进

直到不能再回退，遍历完成

void MGraph<T>::DFS(int v)

{

    bool visted[MAXSIZE] = {0};

    stack<int> vistedNodes;

    int now;

    cout<<vertex[v]<<endl; //输出第一个节点

    vistedNodes.push(v); //第一个节点入栈

    visted[v] = true; //标记已访问第一个节点

    while (!vistedNodes.empty())

    {

        now = vistedNodes.top(); //从栈顶开始搜索

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

            if (!visted[i] && arc[now][i]) //有可前进的节点，记录并输出

            {

                cout<<vertex[i]<<endl;

                vistedNodes.push(i);

                visted[i] = true;

                break;

            }

        if (now == vistedNodes.top()) //如果没有前进则回退

        {

            vistedNodes.pop();

        }

    }

}

3.2 BFS(int v)

基本思想：

起点元素先入队

每次出队一个元素并将相连接的元素全部入队

直到没有剩余元素在队列中

template<class T>

void MGraph<T>::BFS(int v)

{

    bool visted[MAXSIZE] = {0};

    queue<int> vistedNodes;

    int now;

    vistedNodes.push(v);

    visted[v] = true; //起点节点入队并标记

    while(!vistedNodes.empty()) //队列为空代表遍历完成

    {

        now = vistedNodes.front(); //队列首元素出队

        vistedNodes.pop();

        cout<<vertex[now]<<endl; //遍历队首元素

        for(int i=0;i<vNum;i++)

            if(!visted[i] && arc[now][i]) //将所有该节点未标记的联通节点都入队

            {

                vistedNodes.push(i);

                visted[i] = true;

            }

    }

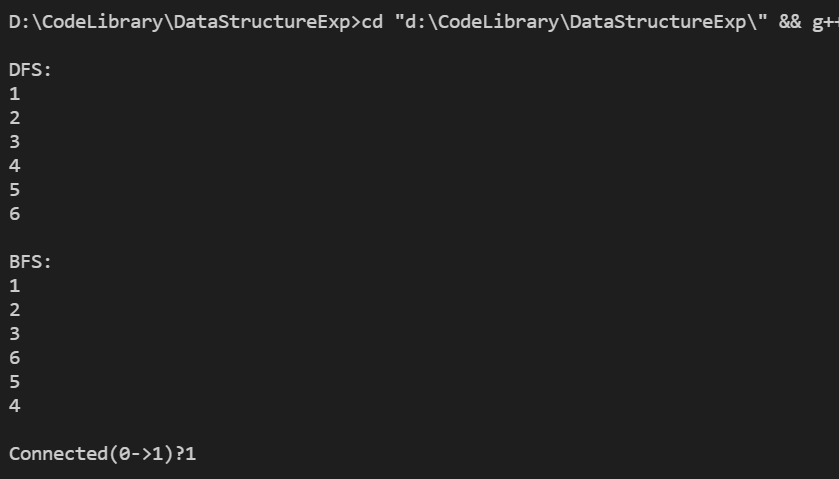
}

4、不足

【实验程序哪些函数功能还有缺陷或不足，或者程序架构有不足，或者性能还有待提高、或者代码不和规范等等，所有你自己对程序不满意的地方】

无法遍历完全非连通图，只能遍历无向连通图，而且不能定义边的权重

5、运行效果图



6、心得体会

【所有你在实验中的感受和想和老师说的话都可以放在这里，篇幅不限】

图的实现用到多种数据结构，很有锻炼价值