实验一、图的代数表示

2018011365 张鹤潇

一、实验内容及要求

编写用于表示有向图的数据结构，以及不同表示方法之间相互转换的程序。

从文件读入一个有向图（带权，n 个结点，m 条边）的权矩阵表示，输出这个图的关联矩阵、边列表、正向表、邻接表表示。无自环，无重边。

二、设计思路

将原始数据用边列表读入，再将边列表转化为其它三种类型。最后依次输出。

数据结构

二维数组 InMatrix 表示关联矩阵，每一列的第零行记录相应边的权值。

二维数组 edgeList 表示边列表，edgelist[0],edgelist[1],edgelist[2]分别对应边列表中的 A,B,Z.

结构体 Positive_Table 表示正向表，由一维数组 index 和二维数组 sub 组成，index，sub[0]，sub[1]分别对应正向表中的 A,B,Z.

链表向量 adTable 表示邻接表。

算法

根据图的各种表示方法的定义直接进行计算。

三、编译环境说明

操作系统 Windows 家庭中文版，开发调试使用 VS2017 Community 15.9.7。清橙 OJ g++ 2.7.4 下测试。

四、关键代码分析

文件读写

使用 c++特性的流对象进行文件输入输出，注意 ifstream 和 ofstream 在作为函数的参数时必须使用引用。

```
#include <fstream>

ifstream fin("input.txt");
ofstream fout("output.txt");

fin>>...;
fout<<...;

fin.close();
fout.close();

void Ini_Digraph(vector< list<int> >&, ifstream& fin);  //初始化
void printAll(vector< list<int> >, ofstream& fout);    //屏幕输出
```

向量链表的使用

邻接表可以使用数组+指针的方法实现，但是太过繁琐，这里学习使用 STL 标准类模板嵌套，利用封装好的功能方便的实现邻接表。在使用 vector<list<int>>对象作为函数的参数时需要注意加引用。

```
#include <vector>
#include <list>

vector< list<int> > adTable(dots + 1);

adTable[j].push_back(edgeList[2][i]);
adTable[j].push_back(edgeList[1][i]);

for (vector< list<int> >::iterator iter_0 = adTable.begin() + 1;
iter_0 != adTable.end(); iter_0++)
        for  (list<int>::iterator  iter_1  =  (*iter_0).begin();
iter_1 != (*iter_0).end(); iter_1++)
                fout << *iter_1<<" ";
```

边列表到其它图的转换

将边列表转换为邻接矩阵的方法是简单的。由于边列表中的边的顺序是按照起点升序排列的，用一个整形变量 j 进行标记记录每条边的起点，方便正向表或邻接表使用。如果:

```
edgeList[0][i] ! = j;
```

则说明从结点v_j出发的所有边都已经被记录，j 自增 1。相关代码如下:

```
int j=1;
for (int i = 1; i <= edges; i++) { //邻接矩阵
      InMatrix[edgeList[0][i]][i] = 1;
      InMatrix[edgeList[1][i]][i] = -1;
      InMatrix[0][i] = edgeList[2][i];//记录权值
       //正向表和邻接表
       PoTable.index[j] = i;
      while (edgeList[0][i] == j && i <= edges) {
          PoTable.sub[0][i] = edgeList[1][i];
          PoTable.sub[1][i] = edgeList[2][i];
          i++;
            adTable[j].push_back(edgeList[2][i]);
          adTable[j].push_back(edgeList[1][i]);
      }
      j++;
   }
```

五、实验结果与分析

设计两组数据，验证程序的正确性和鲁棒性，输入输出分别如下:

数据一（极端情况）:

Input.txt:

1

0

Output.txt:

\n

\n

\n
\n
1 1
\n
\n
\n

数据二（一般情形）：
Input.txt:
3
0 3 4
1 0 1
0 2 0
Output.txt:
1 1 -1 0 0
-1 0 1 1 -1
0 -1 0 -1 1
1 1 2 2 3
2 3 1 3 2
3 4 1 1 2
1 3 5 6
2 3 1 3 2
3 4 1 1 2
3 2 4 3
1 1 1 3
2 2

结果正确合理。

六、实验小结

本实验的算法难度本身不大，但是在实现的过程中，我还是遇到了一些问题。首先是 vector 和输入输出流对象作为参数传递时应注意使用引用，其次是在输入数据为极端情形下，保证输出的合理性。为编程方便，除了邻接表之外的其它几种图的表示形式直接在全局开了大数组，这样做在一定程度上降低了代码的复用性。设想用一个类封装四种数据，或许能得到更好的效果。

刚开始编程时忽略了文档中对输入输出的要求，导致了消耗了许多无谓时间，这是需要牢记的教训——先明确需求再编程。初次接触 STL 标准库，对它的使

用虽然浅显，但是也遇到不少问题，日后要多做练习。