|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据结构》实验报告** |
|  |
| 实验四  图形结构及其应用  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: |  | | 学 号: |  | | 专 业: | 计算机科学与技术 | | 日 期: | 2020-05-8 | |

# **一、问题分析**

（1）实验内容一中，谣言的传播时间和关系网组成了一个有向图，并且给出了谣言的出发点，要求谣言充斥整个关系网的最短时间。由于从出发点到关系网中每一个点都有一条的最短路径，所以从出发点到关系网中最远的一点的最短路径长度就是所求的最短时间。所以问题可以理解为求出从出发点到所有点的最短路径，再进行一次比较，找到其中的最长时间即可。解决问题的关键在于建立一个有向带权图和求单源最短路径。可以采用邻接矩阵或者邻接表的方式存储该有向图。因为权值非负，用Dijkstra算法可以求出单源最短路径。

（2）实验内容二中，引入了神经网络的概念，并抽象为一个有向带权无环图，并将结点分为三层：输入层，隐藏层和输出层。根据输入层的初始化权值（结点本身带的数据），经过中间隐藏层的的时候进行数据处理，最后在输出层得到最终结果。因为不同层的结点没有交集，且每一层的结点需要上一层的结点处理数据后才能进一步加工数据，根据无环图的特点，所以可以采用广度优先搜索的方法对神经网络进行遍历，一层一层的加工数据。可以采用邻接矩阵或者邻接表的方式存储该有向图，通过逐层迭代，即所谓正向传播，可以得到最终的输出结果。

# **二、详细设计**

2.1设计思想

（1）任务一和任务二中都采用了链式前向星的方式建立有向图的邻接表。 邻接表中含有向边数组，起点数组和一个记录当前边编号的值。其中起点数组用来记录以该起点为弧尾的第一条有向边（实际上是输入数据时与该起点为前驱的最后一条边），有向边数组含有该边的弧头，权值和以该边弧尾为弧尾的上一条边的编号。首先要将每一条边的起点，终点和权值读入，然后利用链式前向星的巧妙构造，将以该起点为弧尾的第一条边的编号赋给新边的*next*，方便通过起点来遍历连接它的每一条边。这种操作相当于在邻接表中的以该起点为头结点的链表中，插入一条新边，位于头结点和原第二个结点之间。这为任务一和任务二中的遍历提供了十分方便快捷的方式。

（2）任务1采用Dijkstra算法求给定起点的单源最短路。用一个*ans*数组来记录从起点到某一点的最短路，数组的下标即是该点的编号。同时在遍历时，需要用一个辅助数组来标记所有被确定为最终最短路的点（即从起点到该点的路径最小值已被求出，不会再被更新），同时用一个数组存上更新了最短路的结点。这两个数组用于在每次遍历时，找到与起点路径最短，同时还没有被确定为最终最短路的点。以该点为前驱，就可以进行下一次更新所有以它为前驱的后继点的最短路。因为输入的数据不保证从起点出发可以到达任意点，还需要在做出选择后将已到达点数增一，如果最后到达点数与有向图的总点数不一致，就输出-1,；否则遍历*ans*数组，找到最大值就是所要求的最短时间。

（3）任务2采用广度优先搜索的思想来遍历神经网络。因为该网络是有向无环图，在广度优先搜索时不会出现层与层之间交集的情况。因为一开始只有输入层有数据，可以从输入层做第一次扩展，得到隐藏层中与输入层相关联的结点，作为第二层结点。这里采用数组和一个*left*和一个*right*下标来构成一个队列，将每次扩展得到的结点编号入队，在遍历时出队，然后继续将它关联的结点入队。还要注意隐藏层结点的入度不一定为1，所以入队时要用一个*vis*辅助数组标记已入队的结点编号，这样在搜索时就不会重复入队，重复计算了。如果搜索到输出层结点，同样不再入队，因为它出度为0。根据扩展的逻辑，就可以将整个神经网络的数据更新一遍，得到答案。

2.2 存储结构及操作

2.2.1存储结构

任务一和任务二均采用链式前向星的方式构建有向图的邻接表，其结构体中含有：

自定义*Edge*类型的边数组*edges*；

*int*类型的起点数组*head*；

*int*类型的当前边数组的编号*tot*。

其中*Edge*类型为自定义的结构体，含有：

*int*类型的边弧头顶点的编号*to*；

*int*类型的边权*weight*；

*int*类型的记录上一条边的编号*next*；

2.2.2 涉及的操作

（1）两个任务中的相关函数，如表1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数 | 功能 | 返回值 |
| init\_graph | 无 | 构造一个新的空有向图 | 指向新有向图的指针 |
| add\_edge | 一条边的弧尾，弧头，  权值和该边所在的有向图 | 在该有向图的邻接表中加入一条新边 | 无 |
| get\_min\_time | 求单源最短路的有向图，  其顶点数和起点 | 任务一：  用Dijkstra算法求出从该起点出发到所有点的最短路，并比较出其中的最大值 | 如果从该起点出发可以到达任意点，则返回该最大值，否则返回-1 |
| forward\_propagation | 代表神经网络的有向无环图，该图的输入层、输出层和总结点数，输入层的初始权值 | 任务二：  使用广度优先搜索，通过正向传播得到输出层各结点的数据 | 无 |

表1

2.3 程序整体流程

（1）链式前向星的工作流程如图1所示。

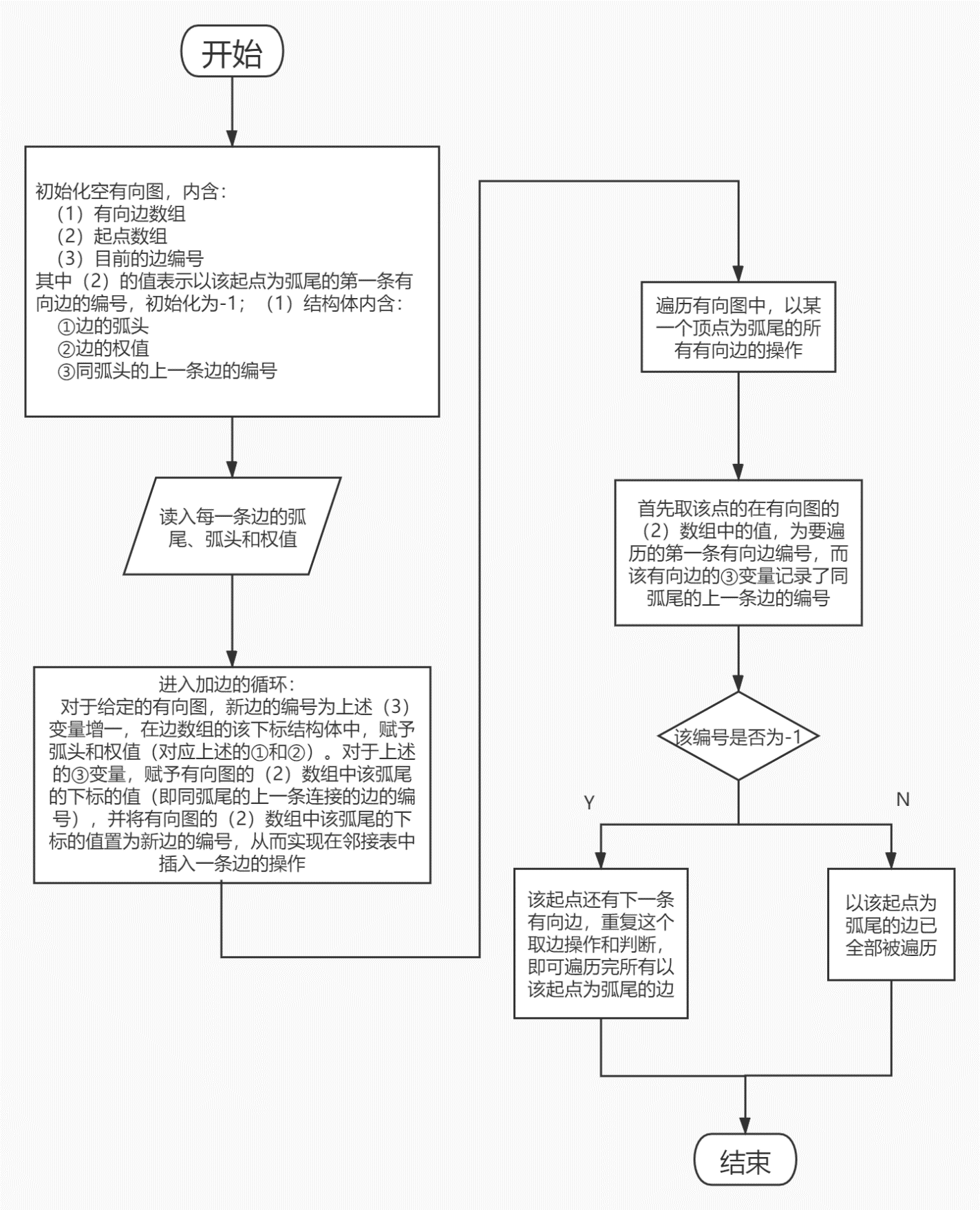


图 1

（2）实验内容一的整体流程与具体算法，如图2所示。

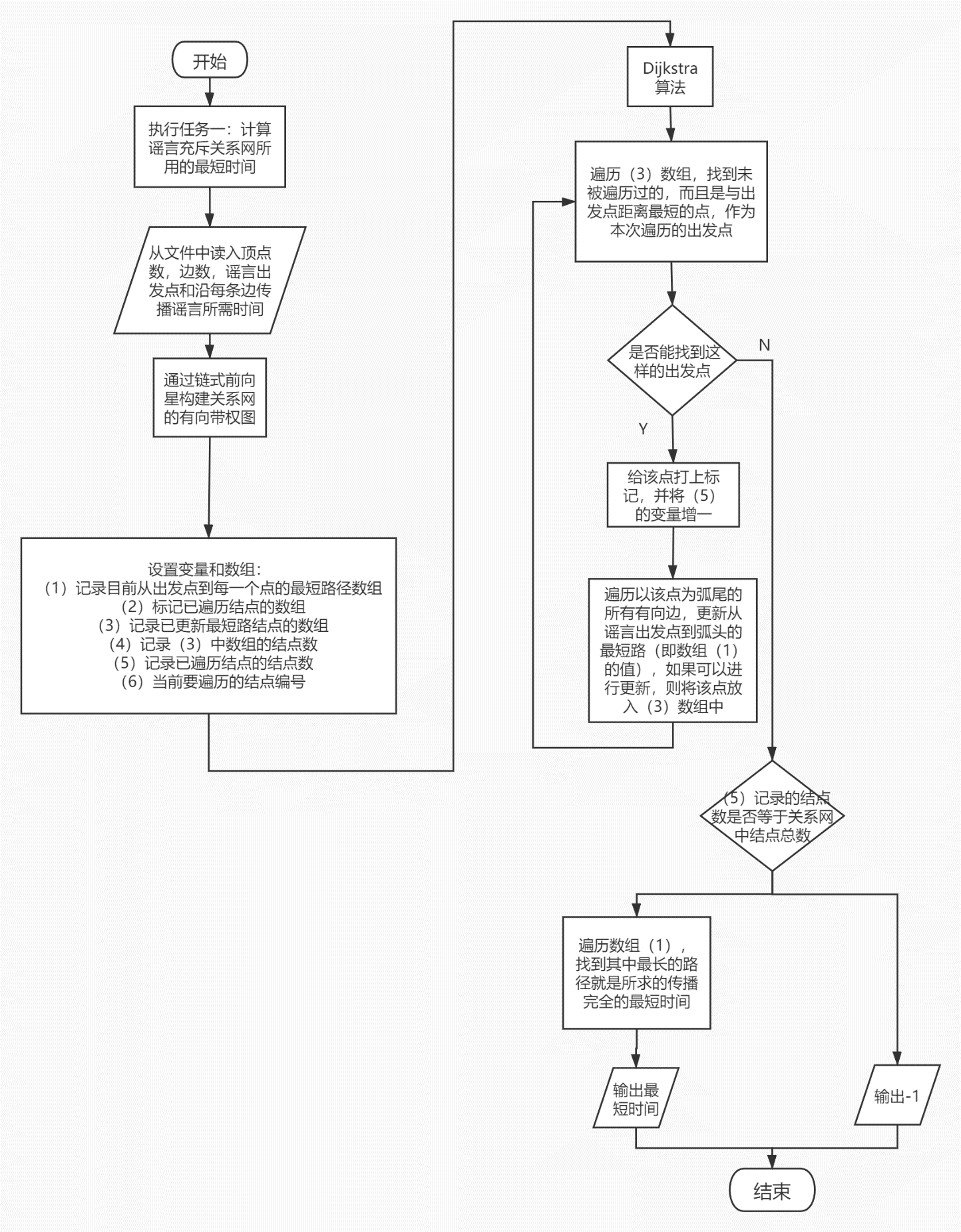


图 2

（3）实验内容2的整体流程和具体算法，如图3所示。

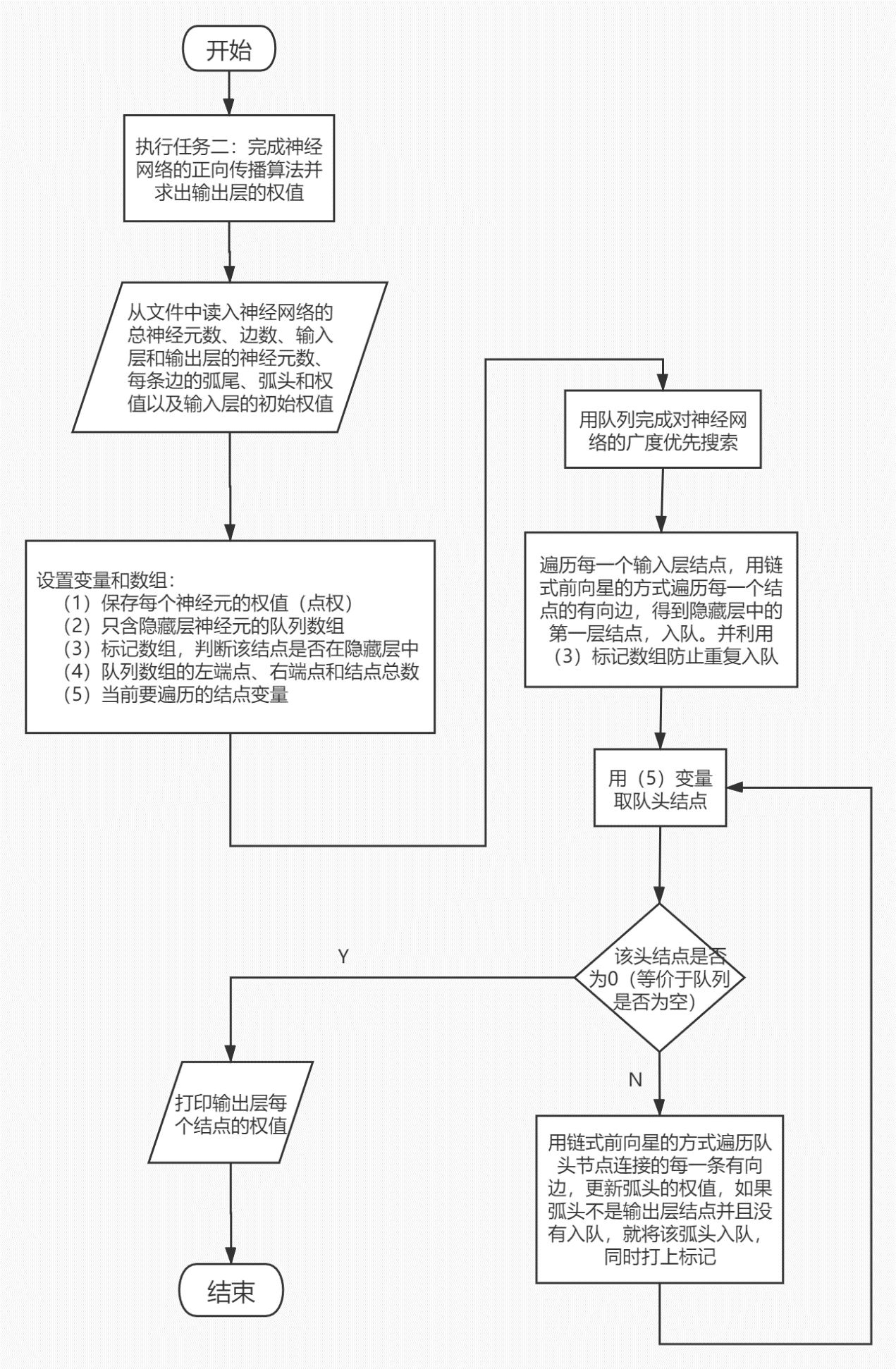


图 3

# 三、用户手册

程序分为实验内容一和实验内容二，均采用从文件中读取数据的方式读入，并含有多组数据的读入。在程序运行后无需其他操作就可得到结果。

（1）

在实验内容一中，将会读取文件名为“4\_1\_input.in”的文件中数据，此文件要放置在.c文件的同一目录下。每组数据的第一行包含三个整数，分别表示点的个数、有向边的个数、出发点的编号。

接下来有*M*行，每一行有三个整数，分别表示第条有向边的出发点、目标点和传播时间，传播时间非负。

对于每一组输入，程序都会输出一个整数，代表谣言从出发点*S*到达该有向图所有顶点的最短时间。如果从*S*出发无法到达某个顶点，则输出-1。每组数据输出各占一行。

（2）

在实验内容二中，将会读取文件名为“4\_2\_input.in”的文件中数据，此文件要放置在*.*c文件的同一目录下。每组数据的第一行包含四个正整数，表示总神经元个数，总的边个数，输入层神经元个数以及输出层神经元个数。

接下来有行，每三个正整数，表示有一条从号神经元指向号神经元的有向边，边权为。

最后一行有个正整数，表示每个输入层神经元的输入值，即初始权值。

对于每一组输入，程序都会输出个整数，整数与整数间用空格隔开，表示每个输出层神经元的输出值。每组数据输出各占一行。

# 四、结果

截图分为两个实验内容的输入文件和程序输出截图。

（1）实验内容一中的输入如图 4和图 5所示：

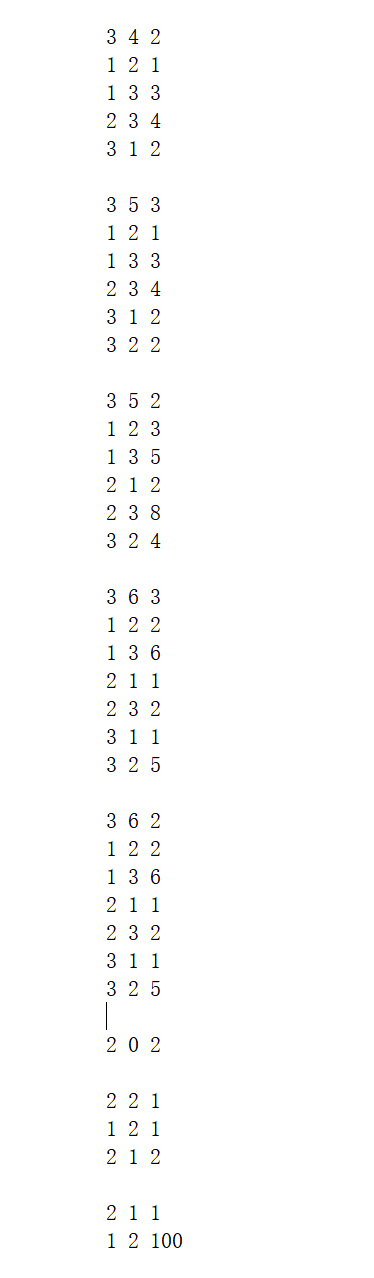
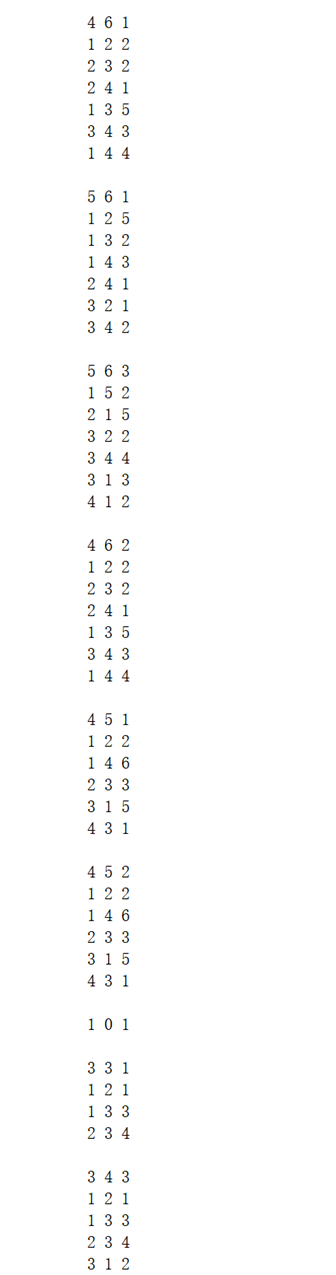


图 4

图 5

（2）实验内容二中的输入如图 6和图 7所示

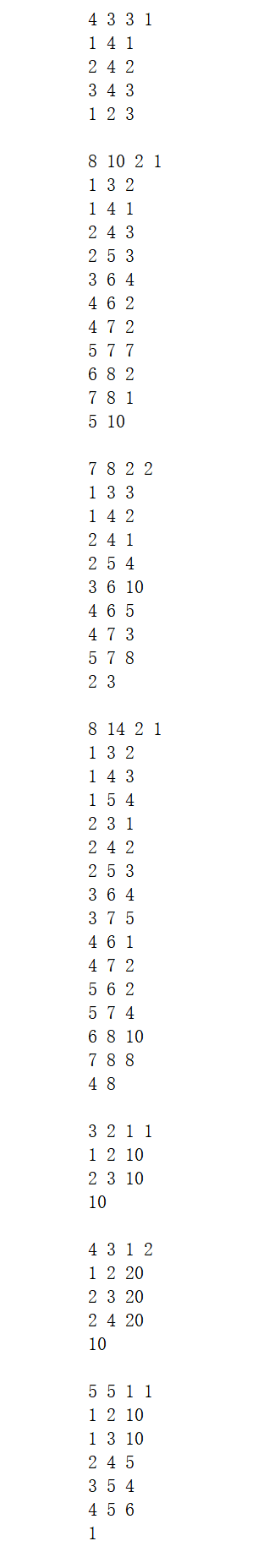
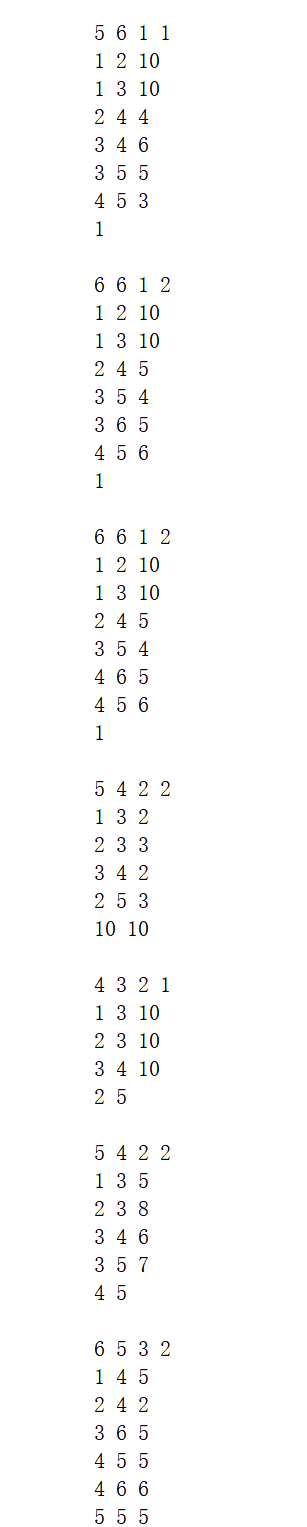
 

图 6

图 7

（3）实验内容一输出截图如图 8所示



图 8

（4）实验内容二的输出截图如图 9所示



图 9

# 五、总结

本次实验的两个内容均采用链式前向星的方式构建邻接表，从而实现了有向图的存储和遍历。最关键的就是理解链式前向星的构造方式和遍历手段。它的优点在于可以存储顶点数较大或者带重边的有向图，如果以邻接矩阵来存储会占用大量空间。并且在遍历的时候，链式前向星的遍历和邻接表的遍历时一致的，比以邻接矩阵遍历时的时间复杂度的常数要小得多。在实验内容一中，运用了Dijkstra算法来求单源最短路径，我在实验中仅仅采用数组来存储已遍历的结点，时间复杂度为O（*n²*）。在实验内容二中，运用了之前学过的广度优先搜索的思想来遍历有向无环图，在迭代中求得最后一层结点的权值。

在实验中，我充分理解了链式前向星的构造方式，学会了一种新的构建邻接表的方式，通过数组来模拟出链表的存储方式。并且在Dijkstra算法中，将链式前向星的遍历方式带入，可以简化原来Dijkstra算法的结构，更容易理解。在实验内容二中，我也是利用数组模拟出来的队列，进行广度优先搜索。用数组来操作可以简化代码量，但是增加了编程的难度，需要一些辅助数组来帮助逻辑的实现，在debug时也可以直观地知道数组里面存放的数据。

在实验中还是存在不少问题。比如在实验一中，每次循环都要找到还没有遍历的，到出发点最短的结点。这里有两个限制条件，一个是还没有遍历，一个是最短。在实现的时候我没有处理好逻辑的关系，导致直接编程的时候debug了很久。这也启发我需要要先想好逻辑层面的细节，而不是匆匆忙忙地赶代码。在求距离最短的结点时，我采用的是朴素的搜索一遍数组的方法，时间复杂度是O（*n*）。

但是实际上可以用小根堆来优化数组表示的集合。在每次将数据存入小根堆时，都会将该数据放在一个合适的位置。而小根堆的根节点就是整个堆中数据的最小值，所以取最小值时时间复杂度为O（1）。在取出最小值后要删除根节点，维护新堆的最小值。删除和插入数据的时间复杂度是O（*logn*），这显然比数组实现要快得多。这也是我在网上搜索相关最短路算法时，普遍采用的一种堆优化。所以，算法可以在原思想的基础上用其他数据结构来优化，这也是我在实验中得到一种对将来学习其他算法有促进作用的思想。