**数据结构实验报告**

**（三）**

**学院 自动化学院**

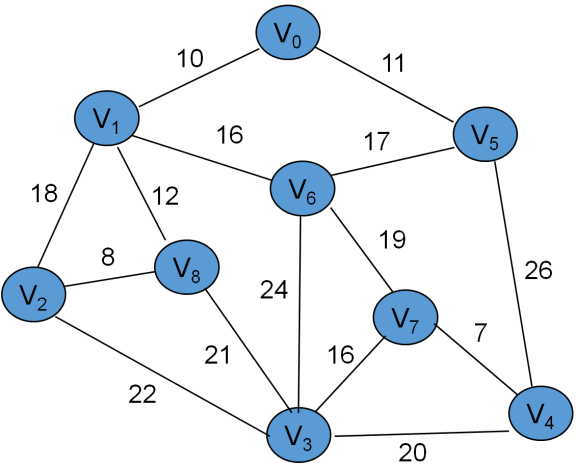
**学号 08022311**

**姓名 陈鲲龙**

**日期 2023-12-20**

**实验内容**

股票经纪人要在一群人（ n 个人的编号为0~ n -1）中散布一个传言，传言只在认识的人中间传递。题目中给出人与人的认识关系以及传言在某两个认识的人中传递所需要的时间。编写程序求出以哪个人为起点可以在耗时最短的情况下让所有人收到消息。下图是人与人认识关系的示例图。



**实验要求**

1. 要求编码实现的功能：

* 请选择一种合适的数据结构表示人与人的认识关系，请用2种存储结构来实现。
* 对各人员信息进行遍历和打印。
* 查询与某人相关联的其他人的信息。
* 使用2种算法，求出以哪个人为起点可以在耗时最短的情况下让所有人收到消息。
* 假设信息传递的花费与用时成正比，求出如何进行信息传递能够让所有人都收到传言且花费最少。

2) 调研该数据结构在实际生活中的其他应用，具体阐述并给出实现思路（可不用代码实现）。

1. **需求分析**

**设计任务：**本实验的需求是用图结构实现一个传言策略的程序，具体要求包含图的存储、遍历、最小生成树、最短路径等问题。

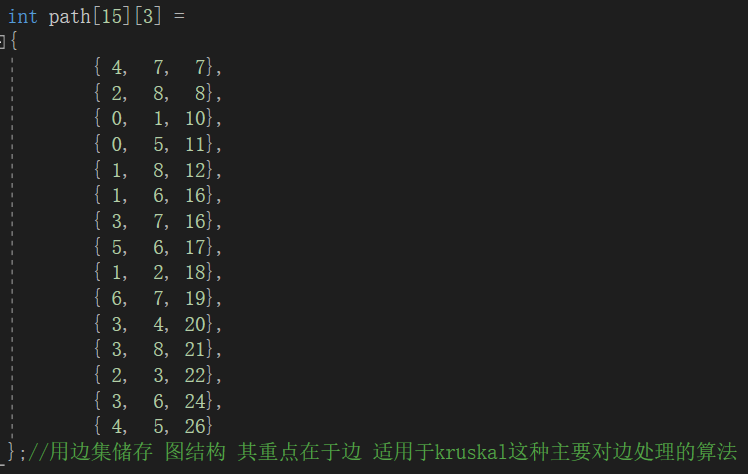
**输出形式：**基于cout打印程序对应内容

**测试：**先根据程序窗口的提示输入对应数字选项即可。

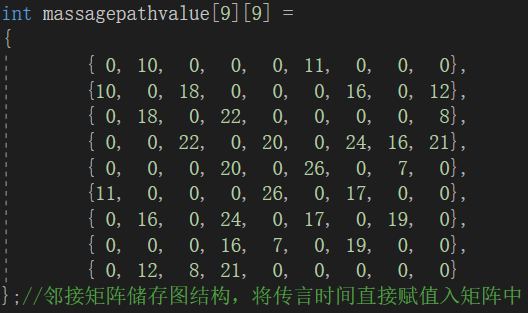
1. **概要设计**

1.图结构的存储用边集和邻接表矩阵，比较适用于kruskal这种主要对边处理的算法。

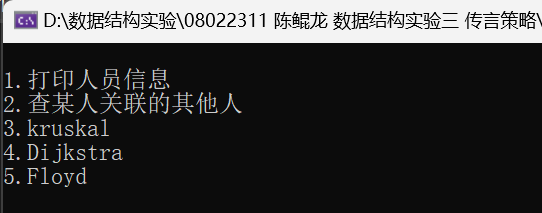
边集：



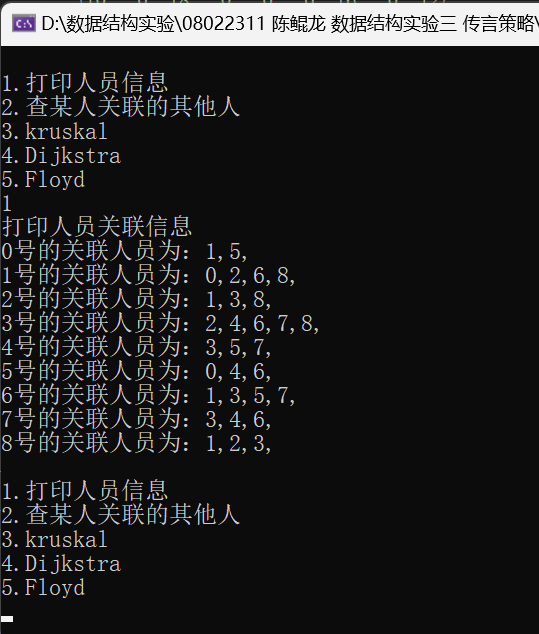
邻接表：



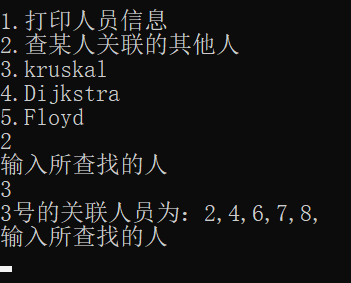
1. 关于“使用2种算法，求出以哪个人为起点可以在耗时最短的情况下让所有人收到消息。”，分析此问题，传言是可以由一个人同时向所有认识的人传言的，所以要求出总时间最短，我们所求的其实是图的最短路径问题而非最小生成树，可以使用Dijkstra或者Floyd算法，具体函数实现详见下文详细设计。
2. 关于“假设信息传递的花费与用时成正比，求出如何进行信息传递能够让所有人都收到传言且花费最少。”，分析此问题，要想花费最少，是需要路径总和最小，并且传到每个人，所以是个最小生成树问题，可以用kruskal算法解决。
3. 结果复现：运行程序看到5个选项：



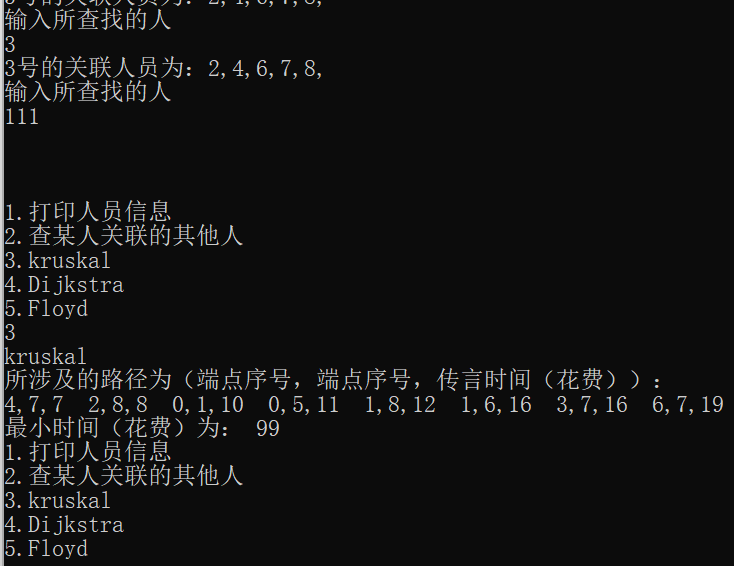
输入1可看到全图信息：



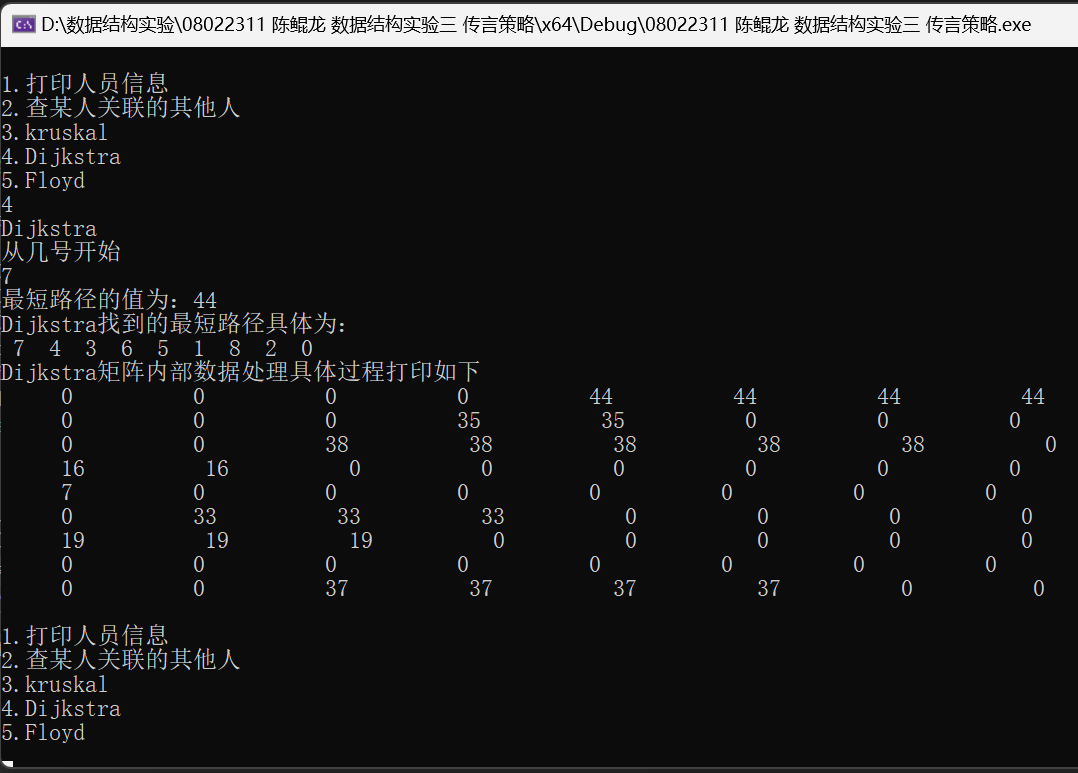
输入2可查看某个人（结点）的关联信息：

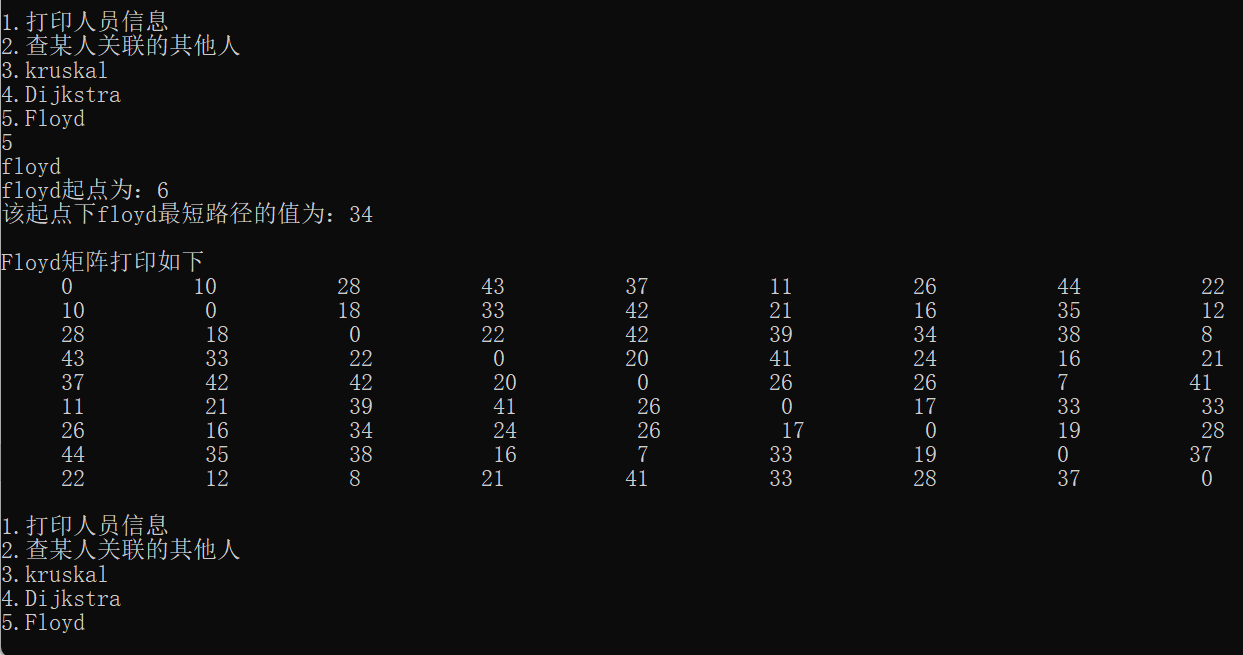


输入3可看到kruskal算法最小生成树的结果：（输入超出0-8的数字即可跳出选项2）



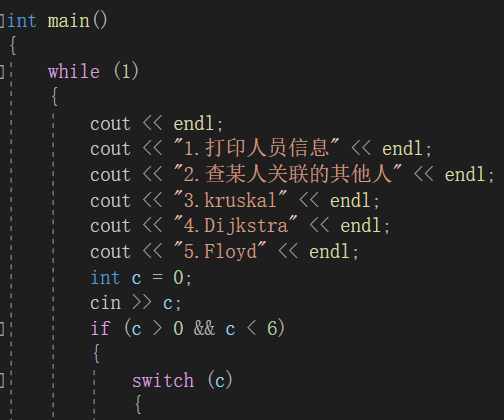
输入4、5可看到最短路径Dijkstra或者Floyd算法的结果，注意4里要输入起点是谁：



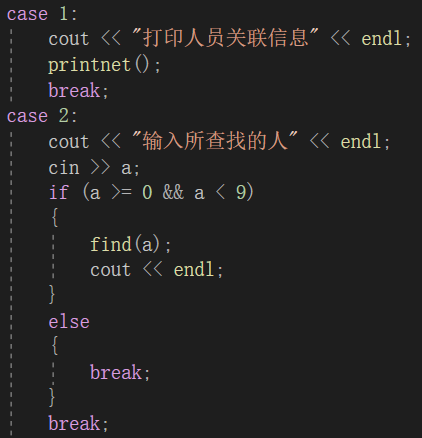


1. **详细设计**

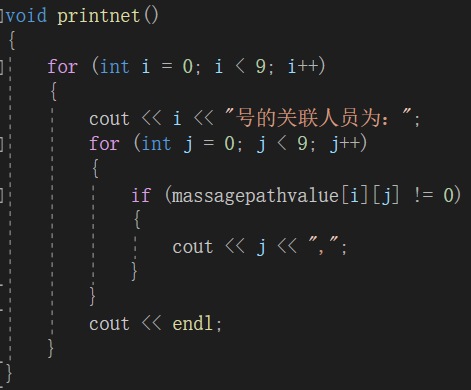
1.主函数：使用死循环，根据输入的数字选项进行switch case，并执行各case里的语句。

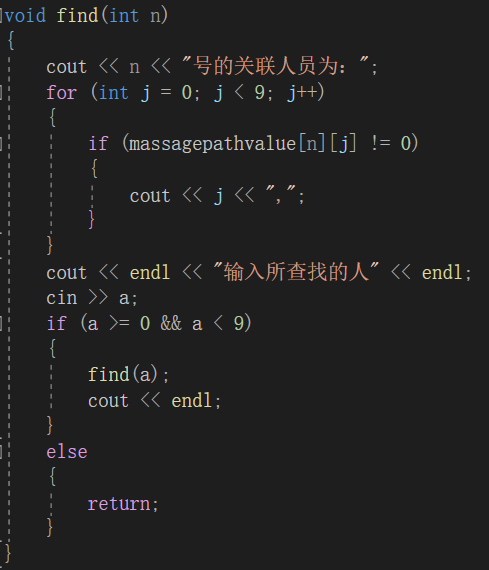


2.case1、2主要是查找、遍历图的信息：



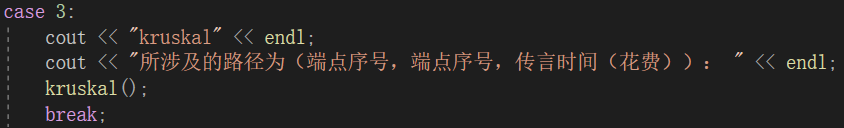
具体相关函数实现：





思想都是遍历某结点（行数）的每一列，如果邻接表矩阵中不是零，说明两结点之间有联系。

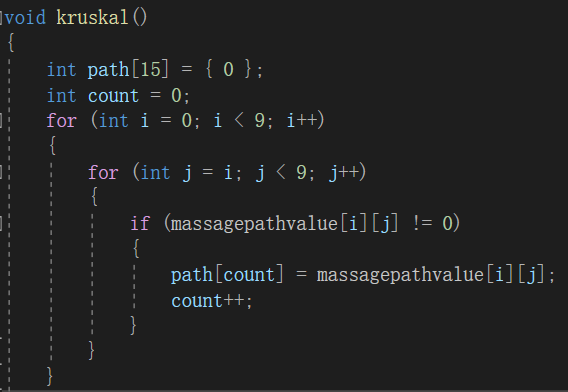
3.case3:kruskal算法最小生成树求解



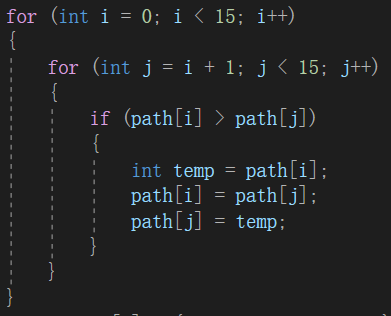
具体函数实现：

kruskal算法主要逻辑就是根据路权从小到大依次选边加入最小生成树中，唯一的规则就是不能有环路。

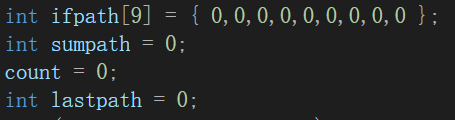
我们先将所有的路权存放在path数组中，共15条边，即15个边权值：



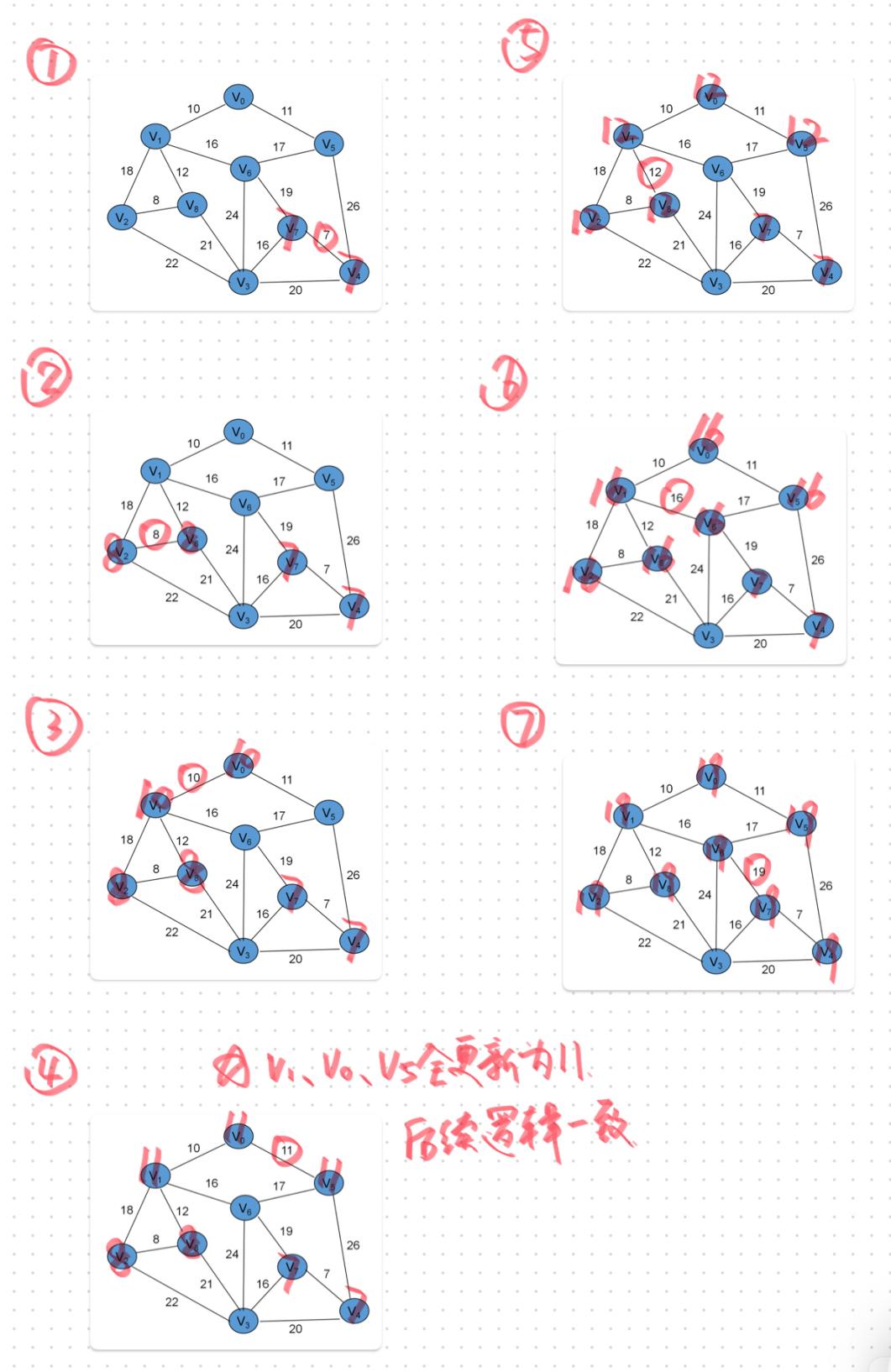
接着从小到大进行排序，数据量小，简单冒泡排序一下，供kruskal算法使用：



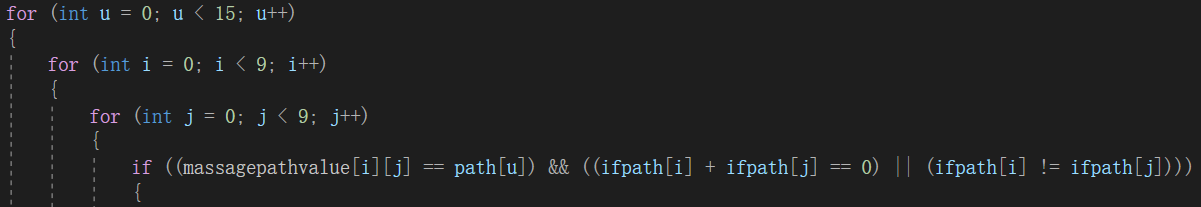
以下是最重要的判断是否环路并求出最小生成树部分：



下图体现了我是如何判断环路的，每当我选出一条边加入物品的最小生成树时，我就会将ifpath数组中这两个节点的值改为它们之间的路权，如果说之后某次选到与之有联系的边，那么所有有联系的结点在ifpath数组中的值都更新为新的路权，这样一来，在ifpath中值一样的结点就代表它们是互相连接的，值不一样就代表不是一路上的结点，这样一来，在树中加入新的边时我先判定是否值不一样，就能保证不出现环路，最终ifpath中的值全会相等。

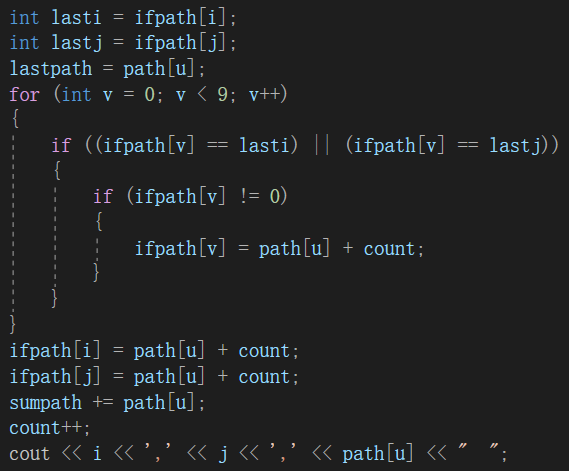


具体代码实现：



Path[u]循环，path是排过序的，即从最小的边开始逐一将符合要求的边加入最小生成树，

进入行列大循环后，先找到邻接表矩阵中所找的边的位置，然后判断是否能加入最小生成树，就是说这条边的头尾必须在当先的树中属于不同的路，不能是环路，==0那个或条件是由于刚开始时ifpath全初始化为0，此时其中的相等并不代表环路。



如果判断为真，那么if语句中执行的操作如上图，先用last系列变量寄存一下当前要加入的边的头尾当前在ifpath中的值，然后遍历ifpath，将所有等于头或者尾的值的地方全更新为新的路权，当然也包含头尾它们自己，这就是将原来两条路更新标记为同一条路的过程。Sumpath的累加就是最小生成树的路权之和。

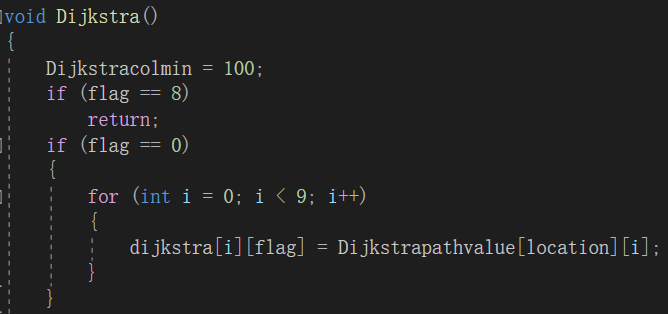
4.Dijkstra算法求解最短路径问题：选到case4后要输入起点，不同起点不同结果，这也就是为什么我们要找出最短路径最小的那个起点：



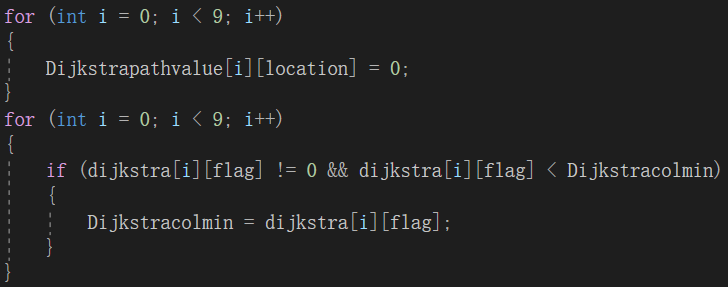
以起点为6时为例，演示我是如何实现Dijkstra算法的：

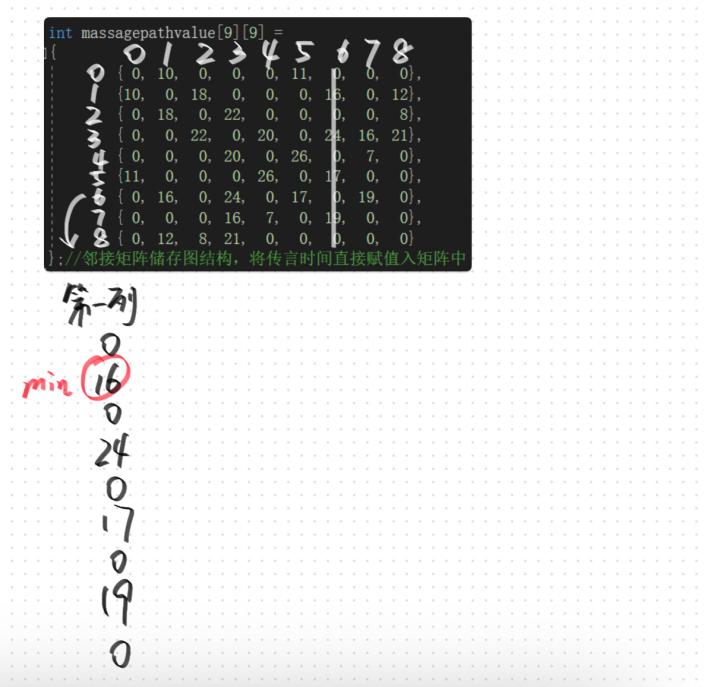
Flag代表算法执行到哪一步了，一共九个点，起点选定了，所以flag=0 ++ -> 7一共8步

第一步时先将起点与其他点的关系即邻接表的行 复制到dijkstra的第一列：

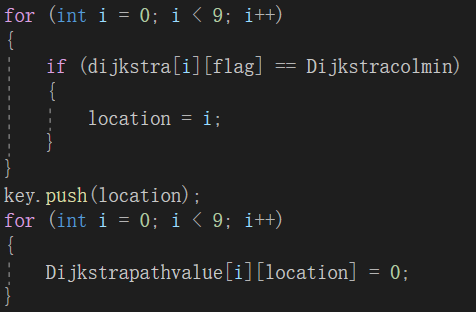


随后先将起点与其他点的关系全部切断置0，防止重复，然后找出dijkstra第一列最小值



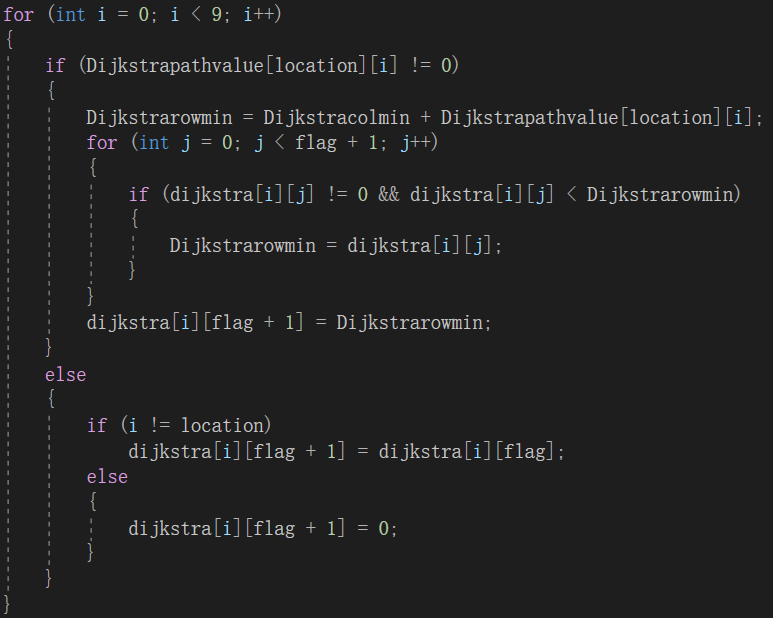


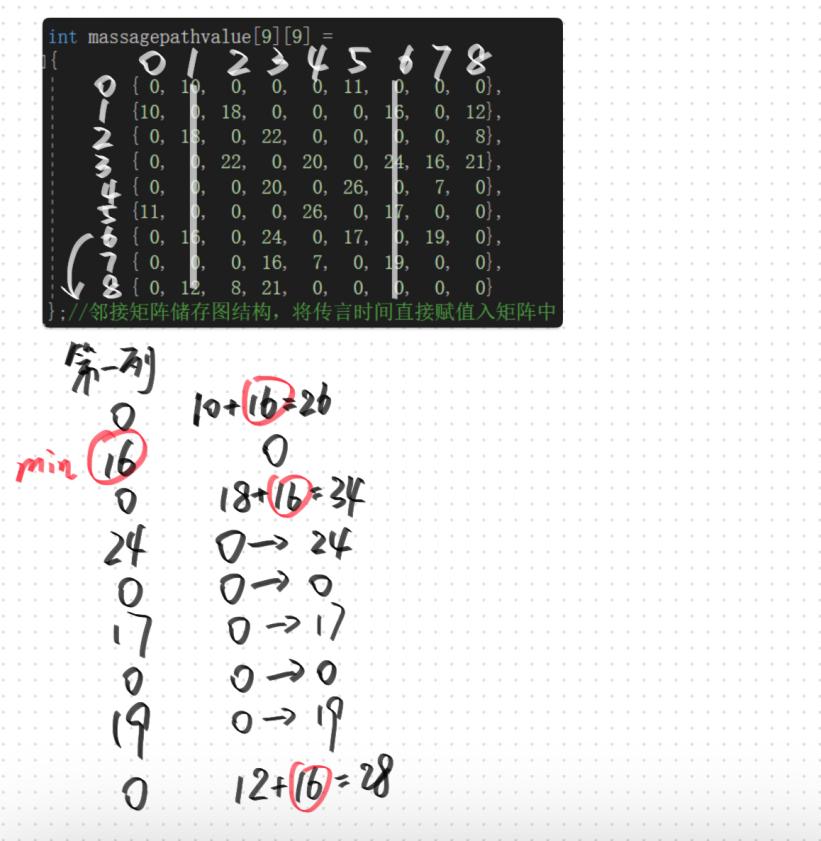
这个最小值对应的点就是新的起点，存放在location变量中，将零阶矩阵中其相关点路径全置0防止重复，并将这个点推进队列key中。



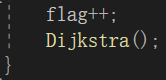
随后是dijkstra最核心的操作：

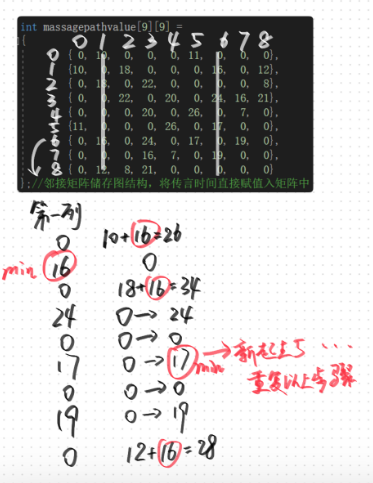
遍历新起点在邻接表矩阵中的行，不为0的话就加上前一列找到的列最小值，即新起点和旧起点的路权，而且这个加和还要和dijkstra矩阵此行的前一列比较大小，取行最小值，如果为0，那就保持前一列的不变，起点这行后面全置0。





最后通过递归，重复这一过程



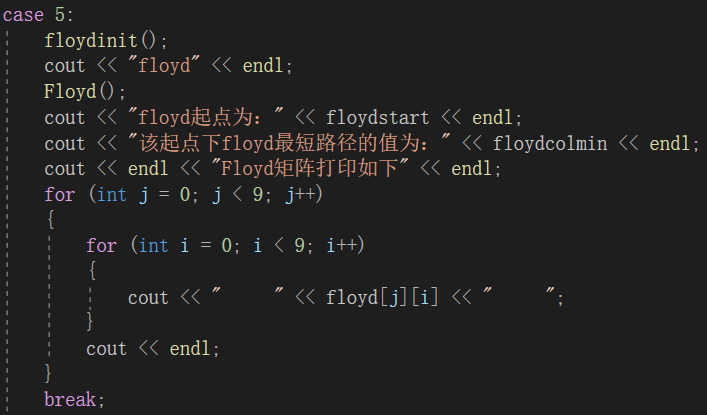


输出结果时我会把这个矩阵打印出来，pop队列key即可获得最短路径具体经过结点的顺序。



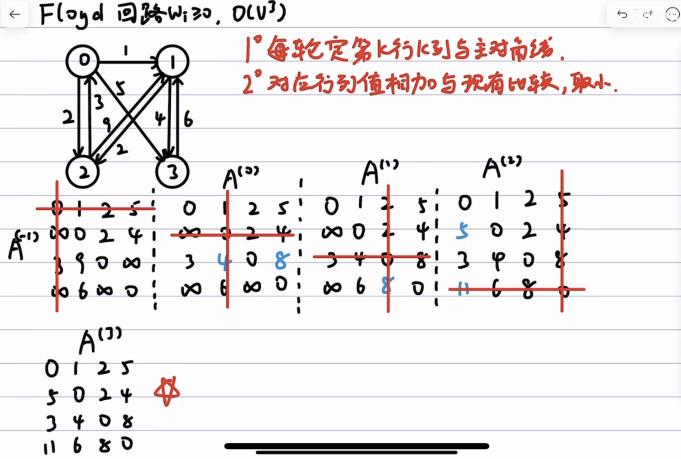
0-8所有结点都运行一遍dijkstra算法，可发现大多数为44、43、42、41左右的值，只有当6为起点时，最短路径最小为34，而事实上，从题目的图中，我们直观的判断也可以推测出从6开始传言最快，因为6位于图的中心，与最多数量的结点有联系，而且其路权也较大。

5.项目对于最短路径问题要求两种算法，我选择的是Floyd算法，它的结果也是起点为6时最短路径为34，正好印证了上文dijkstra算法的结果的正确性。



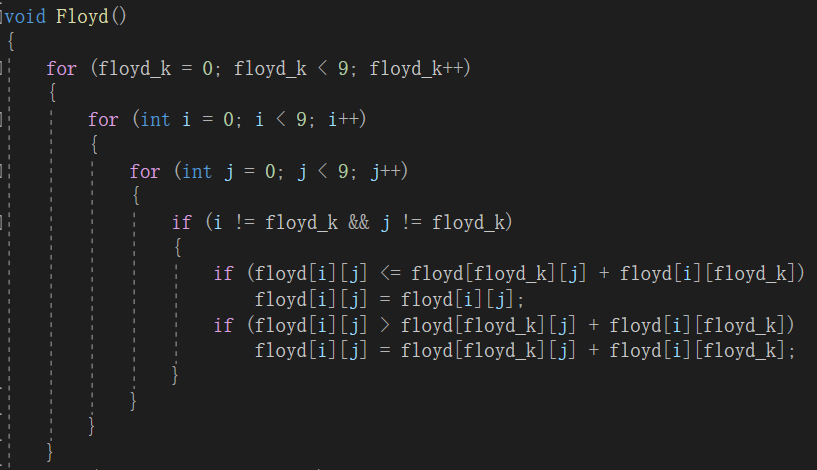
Floyd也是在邻接表矩阵上操作，

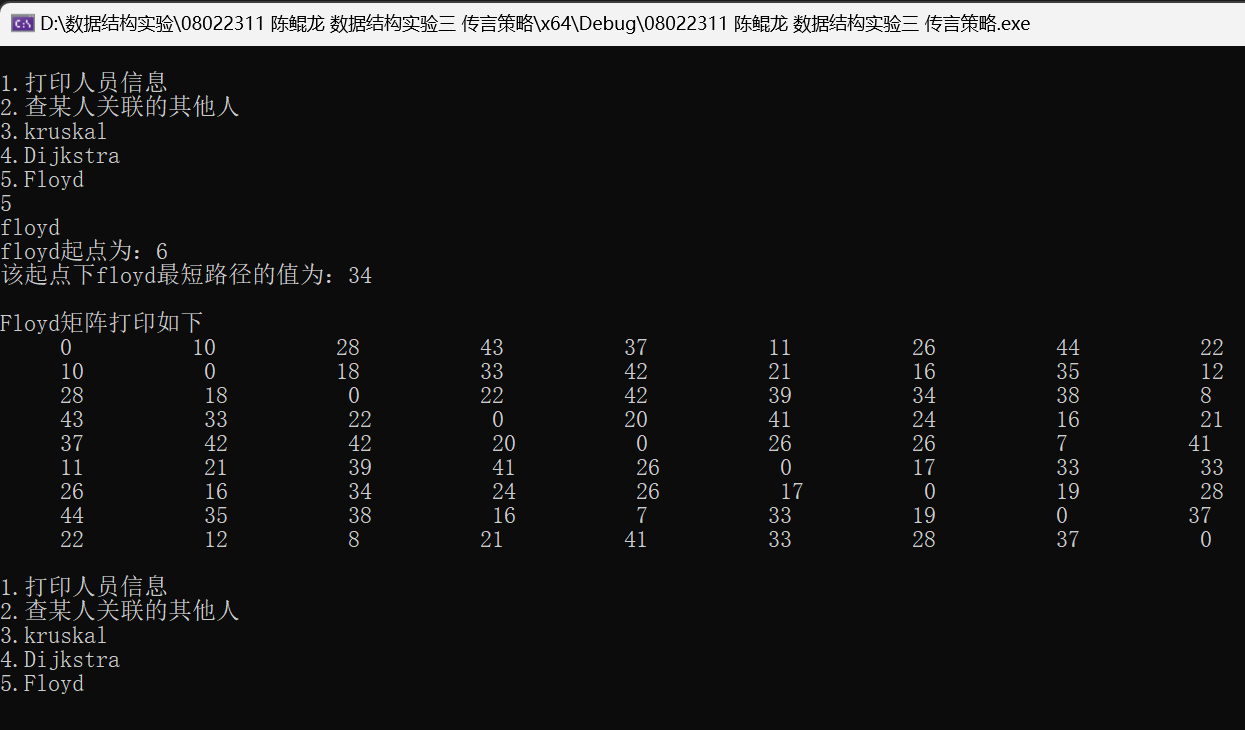
以下图这个小例子阐述我是怎么操作实现该算法的



首先邻接表矩阵除了对角线的0不变，其他0都要变成无限大（任意比较大的数）

从0行0列作为主行列开始，对其余行列的所有元素进行判断，计算主行上该元素的列的位置的元素值加上主列上该元素的行的位置的元素值，这个加和与元素原来的值比较取小，主行列++，重复以上操作。





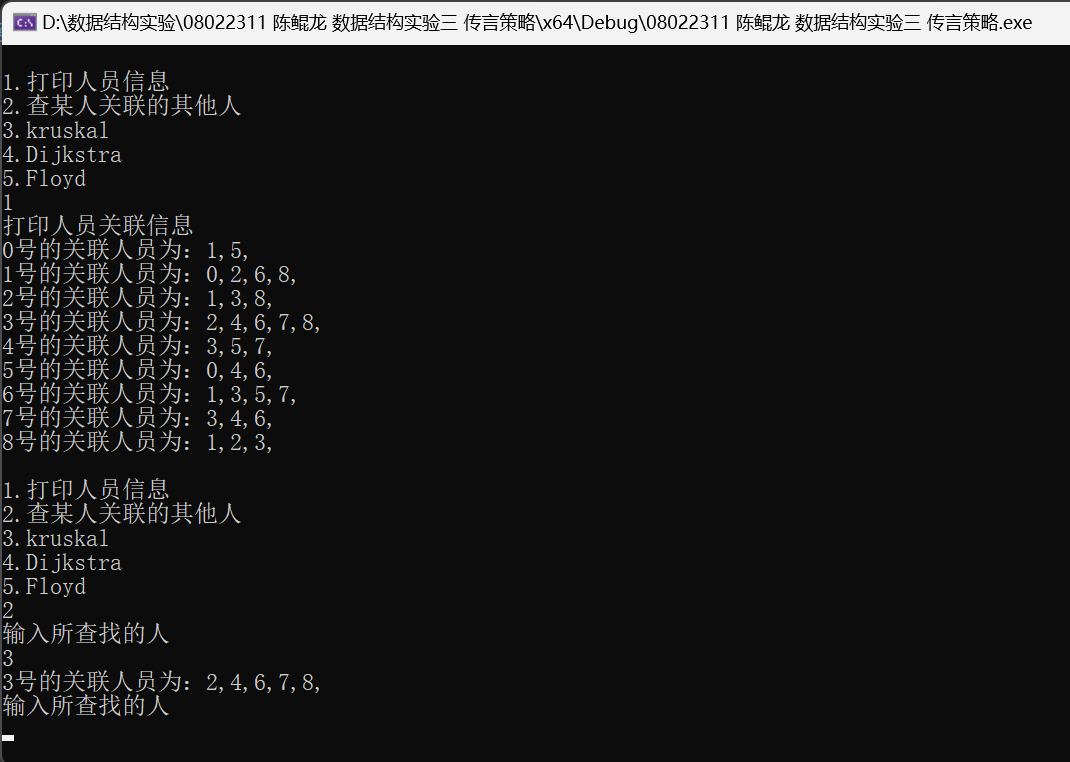
最终矩阵中每行最大代表了不同起点对应的最短路径的值，与dijkstra算法计算结果一致，其中最小的是起点为6的34最短路径。

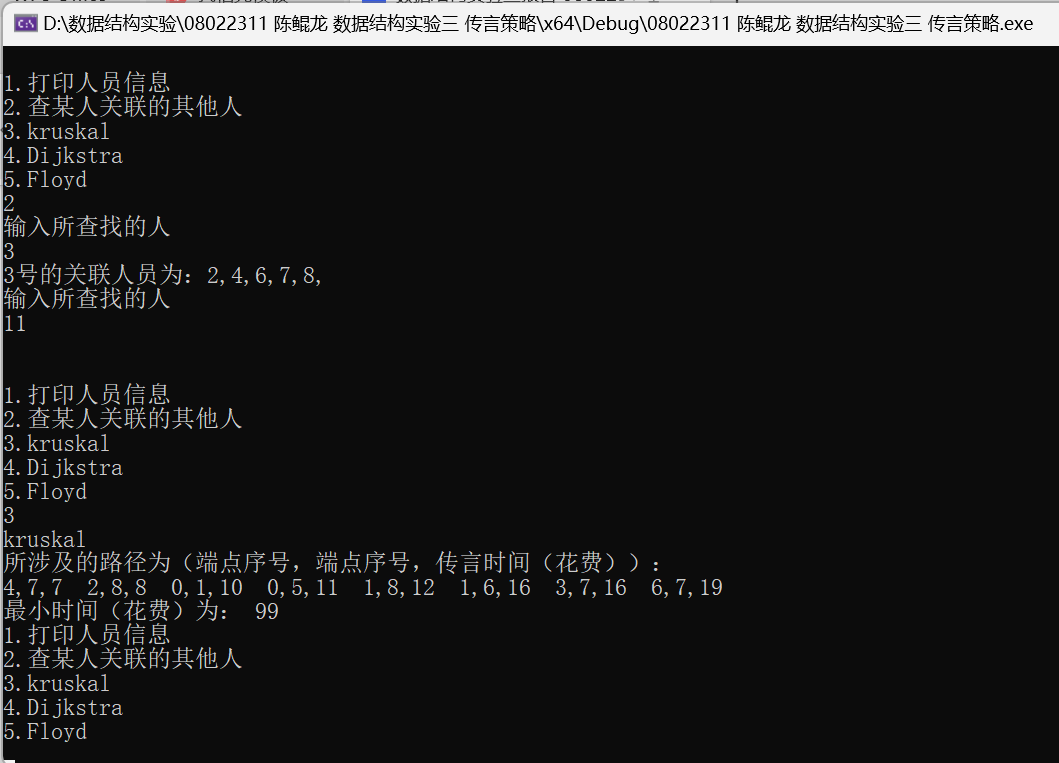
1. **调试分析**

&

1. **测试结果**

调试与测试：程序运行结果与上文叙述的算法思路一致，实现了算法功能，解决了题目需求。



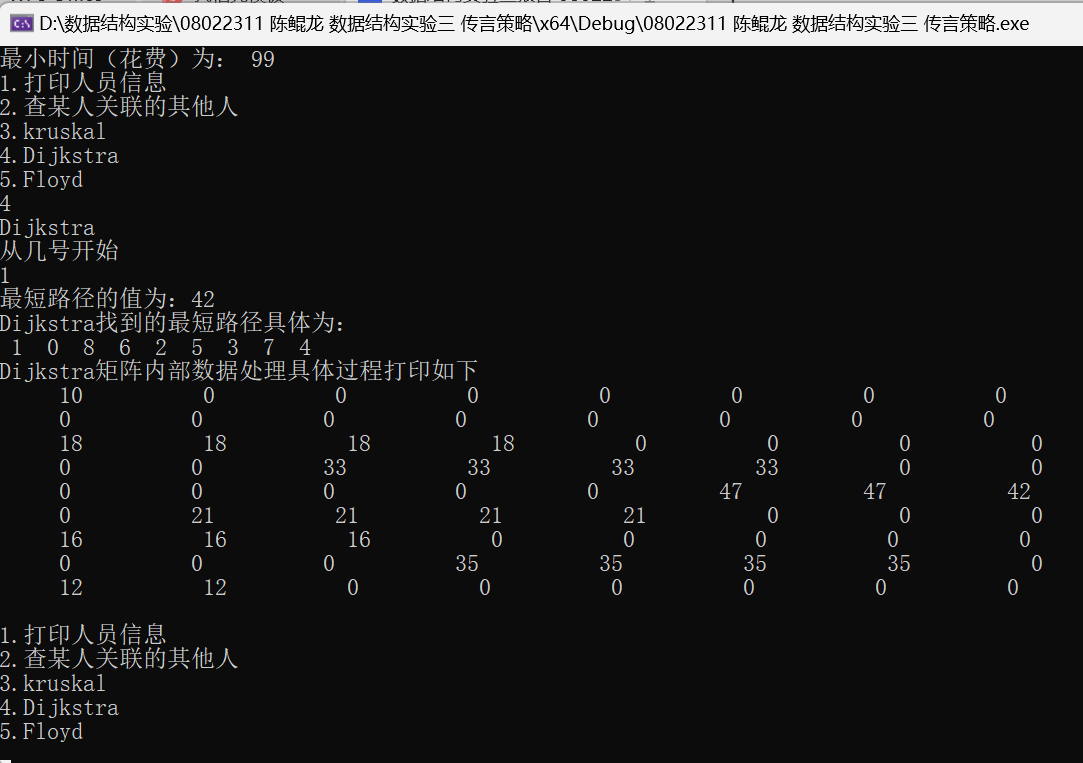


Dijkstra算法：

起点为0时：



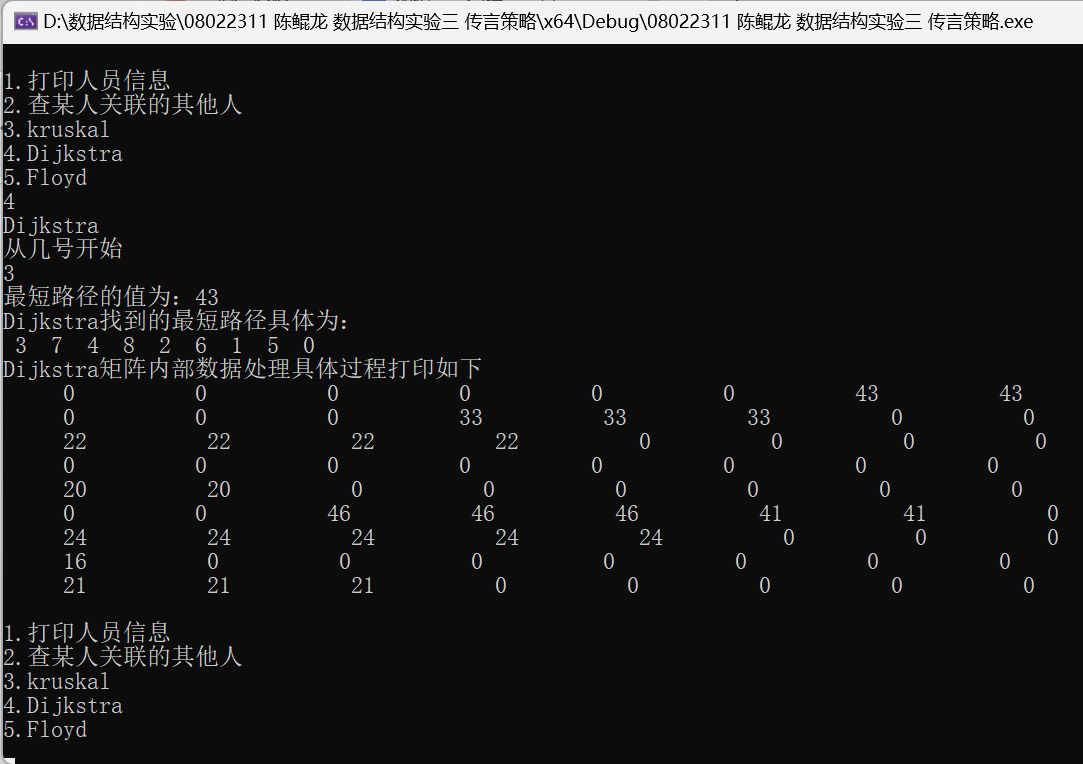
起点为1时：



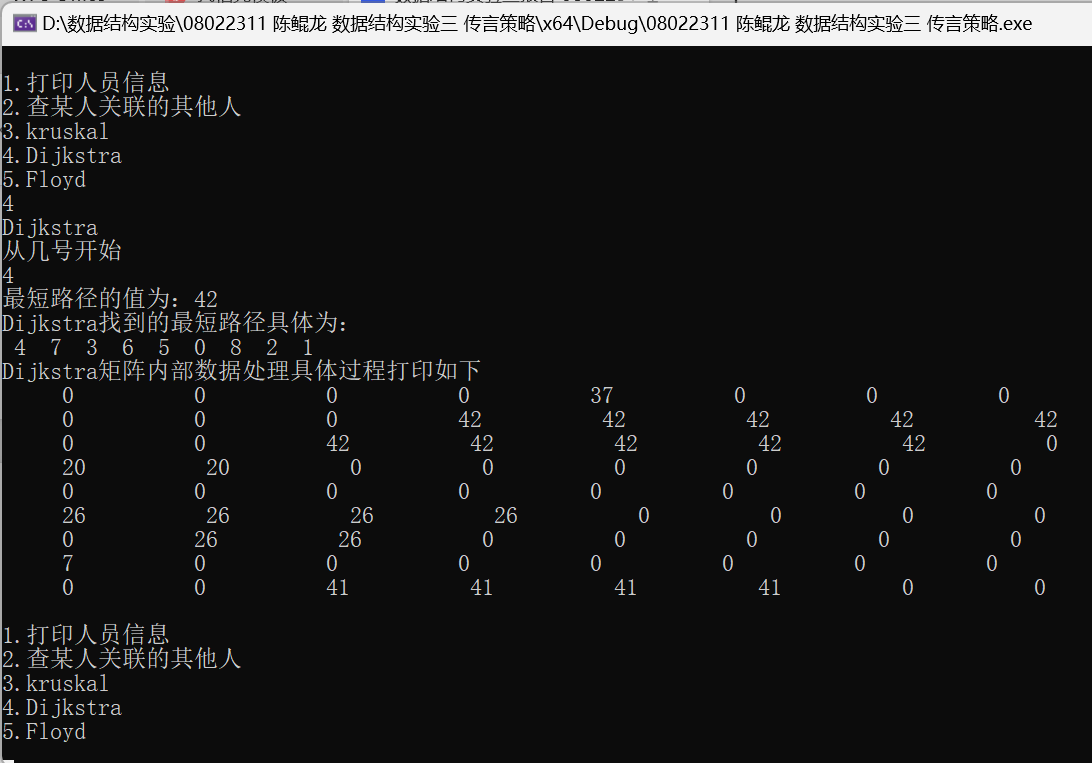
起点为2时：



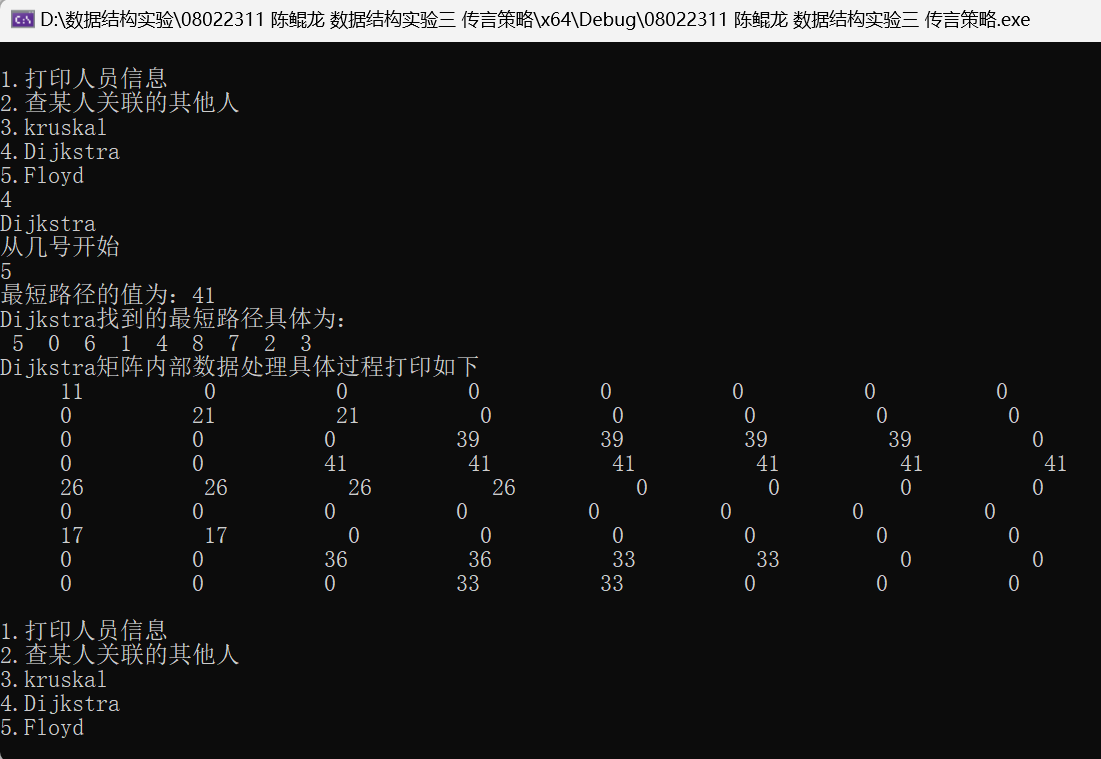
起点为3时：



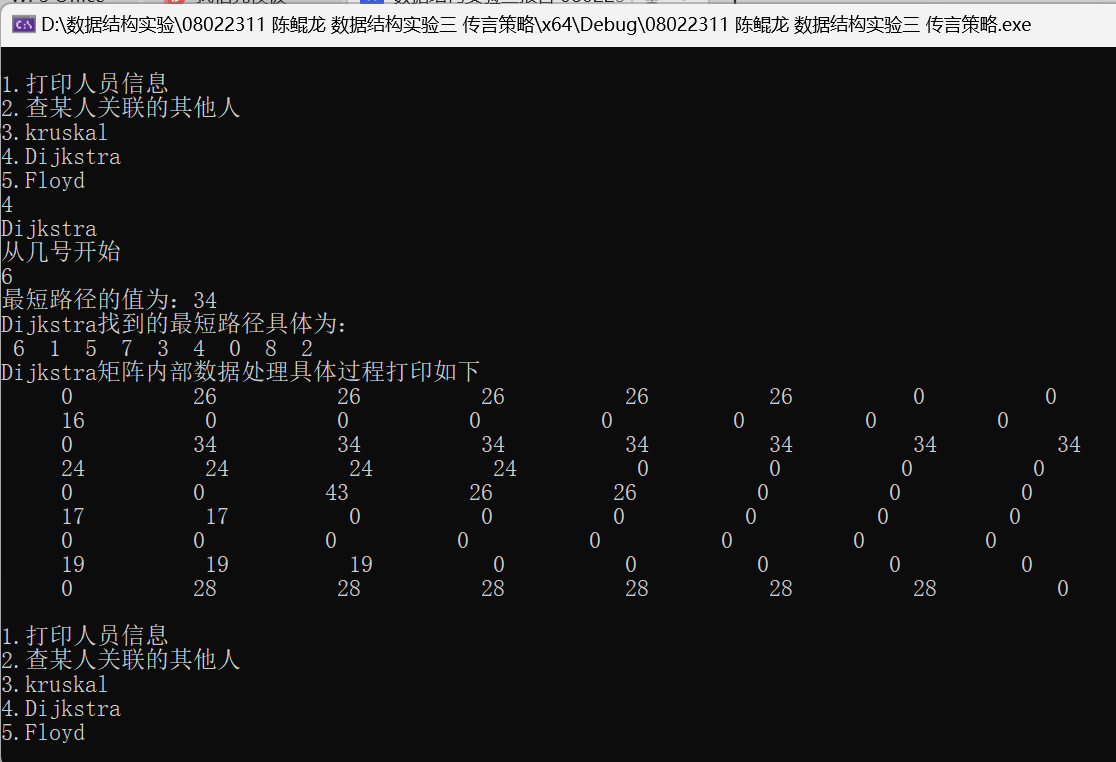
起点为4时：



起点为5时：



起点为6时：



起点为7时：



起点为8时：



1. **实验总结**

可能我能力差点儿，我来不及做，但我宁愿迟交，也不允许出自自己之手的作品是草草了事抄出来的，本次实验的编写过程中，我对图结构、kruskal、floyd、dijkstra算法的理解无疑更深刻了，我撰写代码的熟练度也在提升，交出了这个达到我自己对自己要求的作品。

1. **附录**

本实验代码仅main.cpp

1. **应用调研**

1 社交网络分析

社交网络是图数据结构的典型应用。人与人之间的关系可以用图来表示，节点代表用户，边代表互相关注或其他连接。图算法可以用来识别社交网络中的社区、预测用户兴趣和行为等。例如，Facebook、Twitter等社交媒体平台利用图算法来推荐朋友、内容和广告。

2 推荐系统

图在推荐系统中也有广泛应用。通过分析用户和物品之间的关系，图可以帮助推荐系统理解用户的喜好和行为。这种方法能够提供更加个性化的推荐，提高用户体验。Netflix、Amazon等公司都在推荐系统中使用图算法。

3 知识图谱

知识图谱是一种结构化的知识表示方法，用于捕捉实体之间的关系。它在自然语言处理、搜索引擎、问题回答等领域有广泛应用。谷歌的知识图谱就是一个典型的例子，它连接了数百亿个实体，使得搜索引擎能够更好地理解用户的查询意图。

4 链接预测和社区发现

图算法可以用来预测节点之间可能的连接关系，即链接预测。这在金融风险评估、药物发现等领域有重要应用。另外，社区发现是指将图分成不同的社区或群组，有助于分析节点之间的相似性。这在社交网络分析、生物信息学等领域有广泛应用。