**实验四**

1. **实验要求**

1)实现求有向图的强连通分量的算法。有向图的顶点数 N 的取值分别为：8、16、32、64、128、256，弧的数目为 NlogN, 随机生成 NlogN条弧，统计算法所需运行时间 ，画出时间曲线。

2)实现求所有点对最短路径的Johnson算法。生成连通的无向图，图的顶点数 N 的取值分别为： 8、16、32、64、128、256 ，边的数目为 NlogN, 随机生成 NlogN 条边，统计算法所需运行时间 ，画出时间曲线。

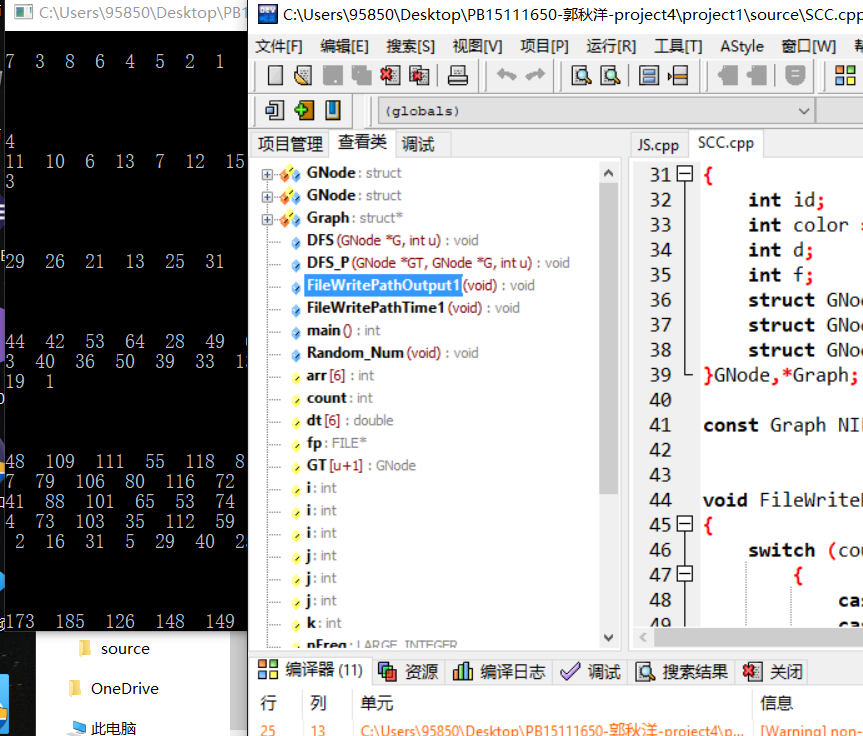
1. **实验环境**

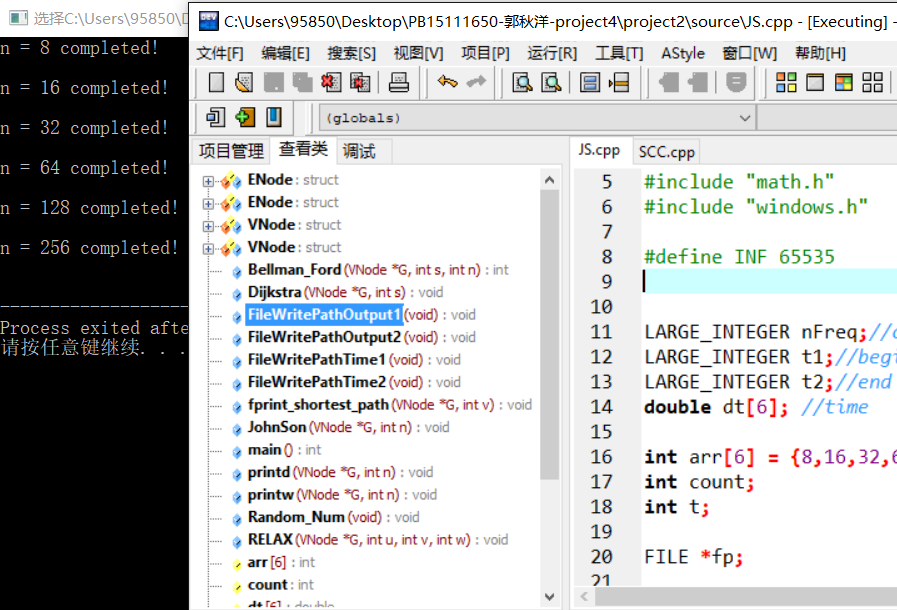
编译环境：Dev C++ 5.11

机器内存：16G

时钟主频：2.20GHz

1. **实验过程**



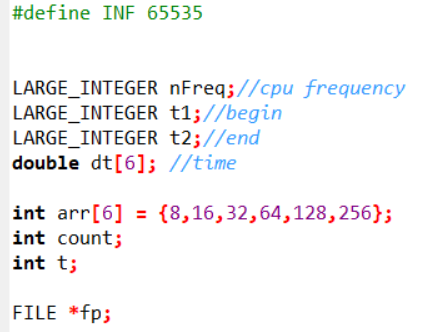


1. **实验关键代码截图（结合文字说明）**

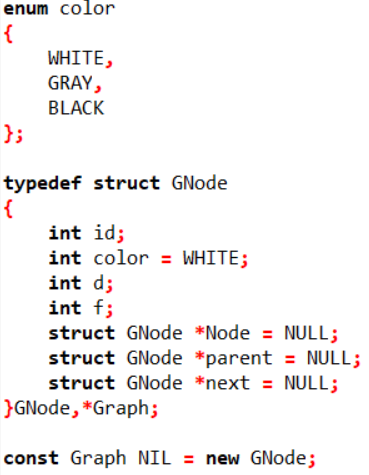
实验1：

1. 全局变量:

t记录DFS中对结点操作的时间戳。

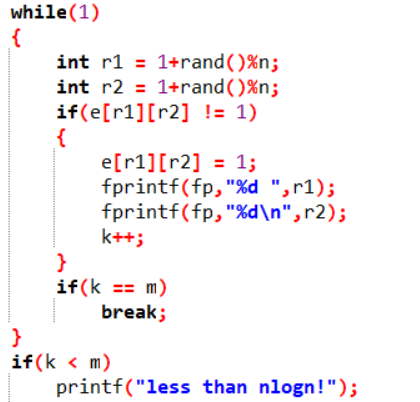


1. 图的“邻接链表”



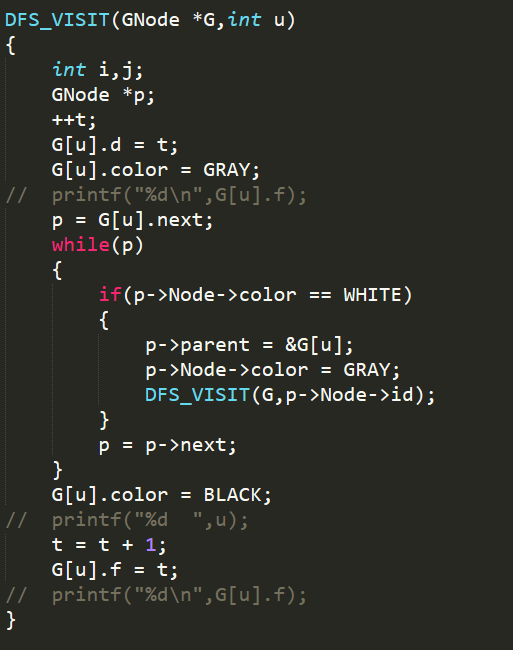
由于开始对邻接链表不熟，所以误以为链表里存放的是结点。所以没有定义边结构，而是全部用结点结构表示，并且额外定义了id标识结点，Node指向本结点，这两个属性是多余的。所以这不是标准的邻接链表，但仍然可以实现各种图的操作。d和f为时间戳，分别记录结点第一次被发现时（涂上灰色）和完成对结点邻接链表扫描时（涂上黑色）的时间t。

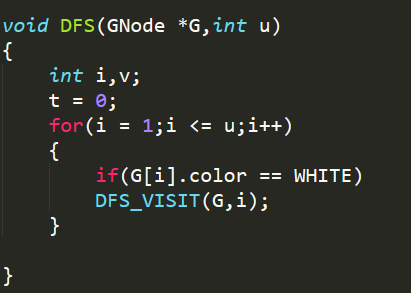
1. 图的生成函数



这里使用邻接矩阵生成图更方便。

4.DFS算法

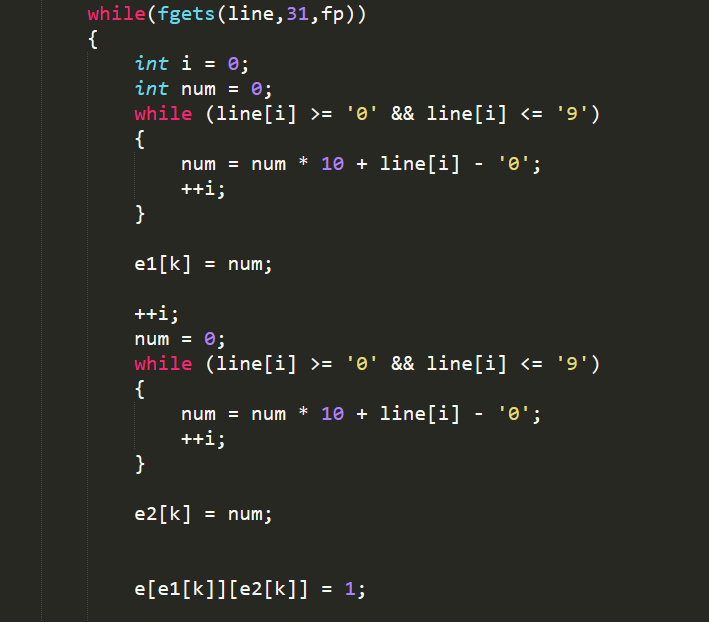




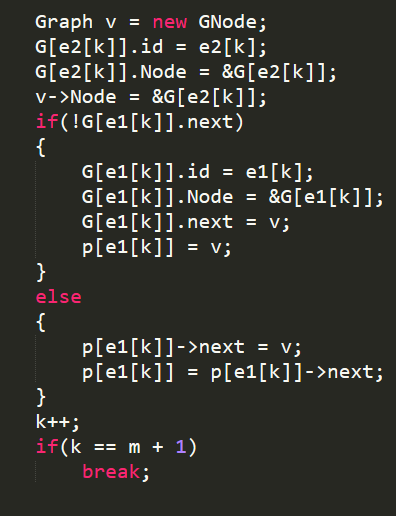
这里p->Node->id可直接用u代替，p->Node->color可直接用G[u].color代替。

5读取文件中的图

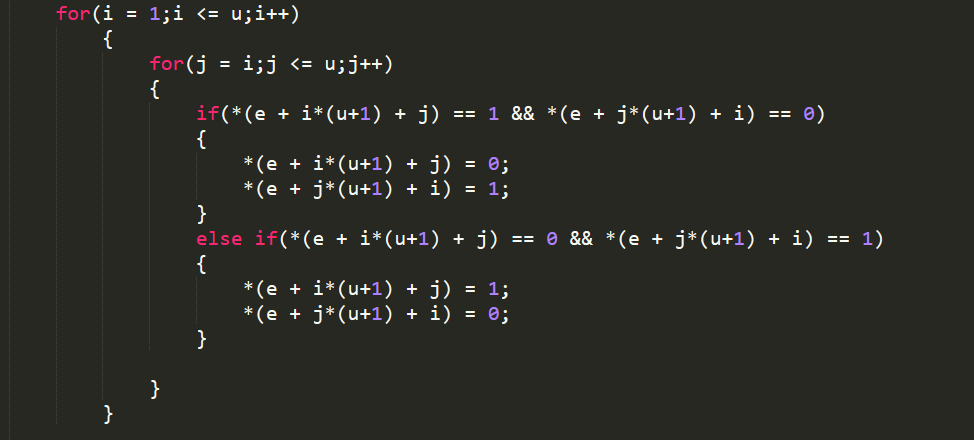
这里用了一个邻接矩阵存储读出来的图主要是为了测试时方便打印图矩阵以及方便图转置，也可直接生成邻接链表。



6生成邻接链表

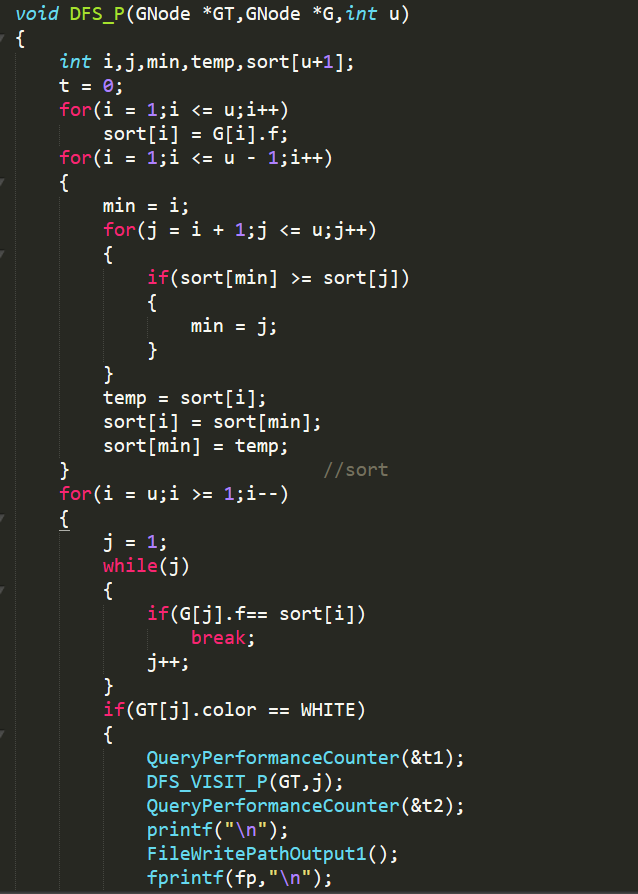


7图转置

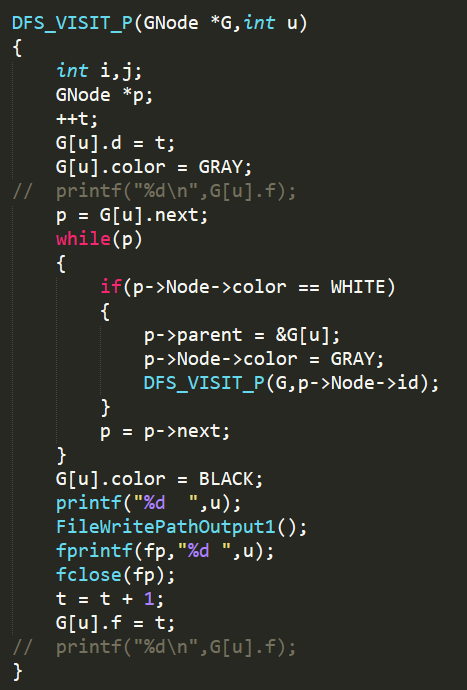


邻接矩阵进行图转置很方便，然后再按6的方法生成邻接链表。

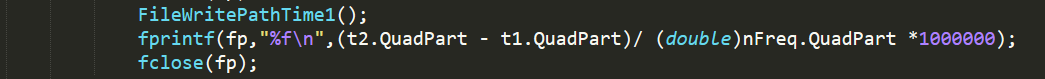
8.按第一次DFS生成的u.f从大到小对GT运行DFS\_VIDSIT\_P



9在DFS\_VISIT\_P中打印连通分量

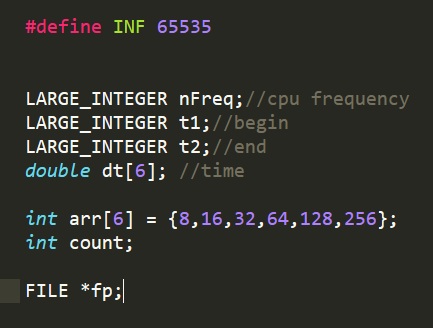


10在DFS\_P中打印求解每个连通分量的运行时间

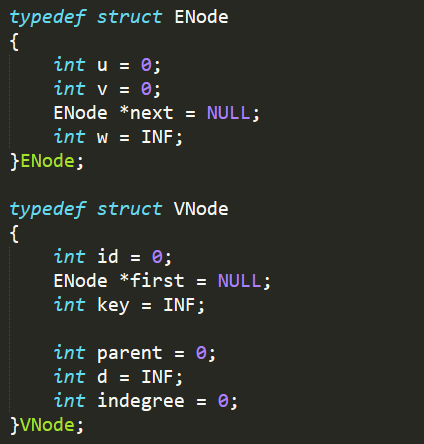


实验2：

1全局变量:

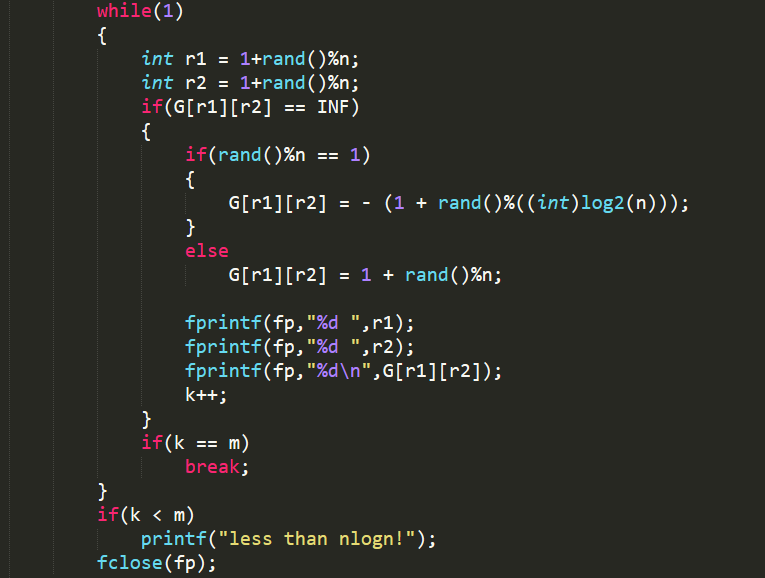


2邻接链表

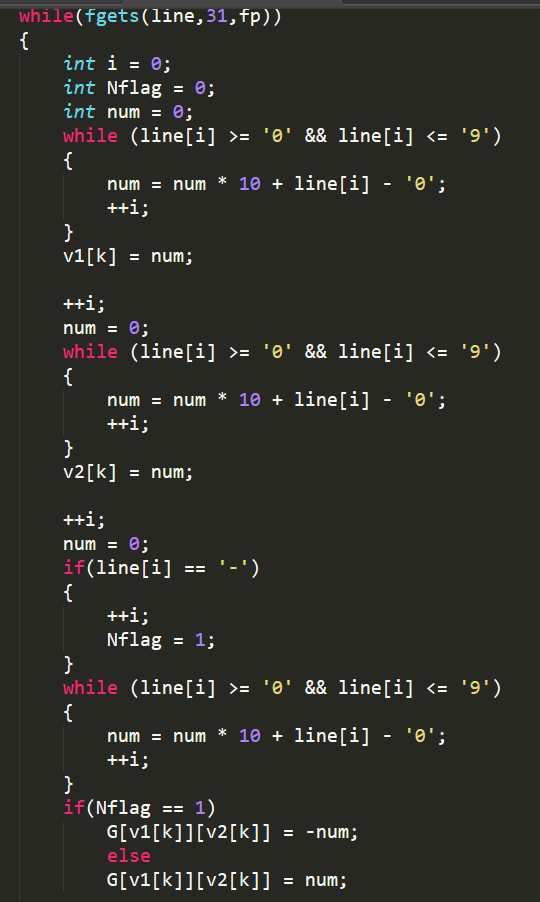


分别定义结点结构VNode和边结构ENode。ENode中，属性u，v为边的两端，方向u->v，w为权值；VNode中id标识结点，key为结点携带的值，d为最短路径上界，indegree为入度。本实验中id，key，indegree没有用到。

3图的生成函数

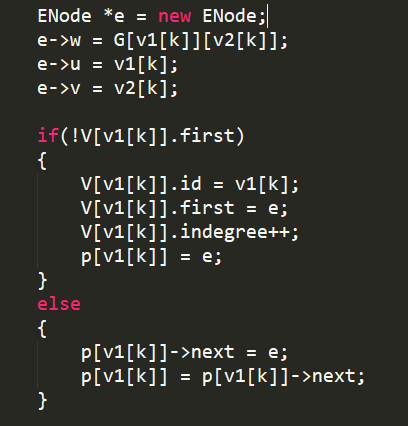
生成的图满足：1/n的边的权值为负的，且绝对值不大于log n。其他边权值为正，且不大于n。同样用邻接矩阵存储。

4读取文件中的图

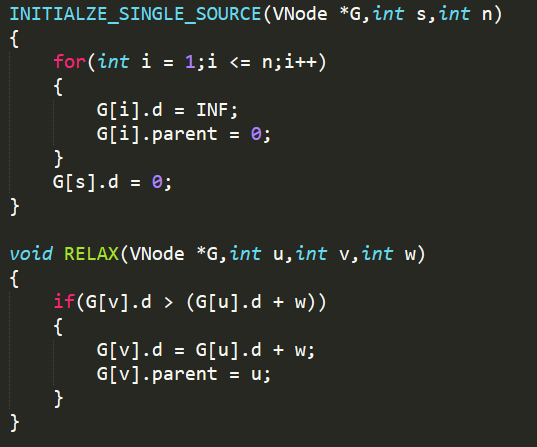


同样是先读到邻接矩阵中方便打印。在实验二由于不需要转置所以可以直接存到邻接链表中。

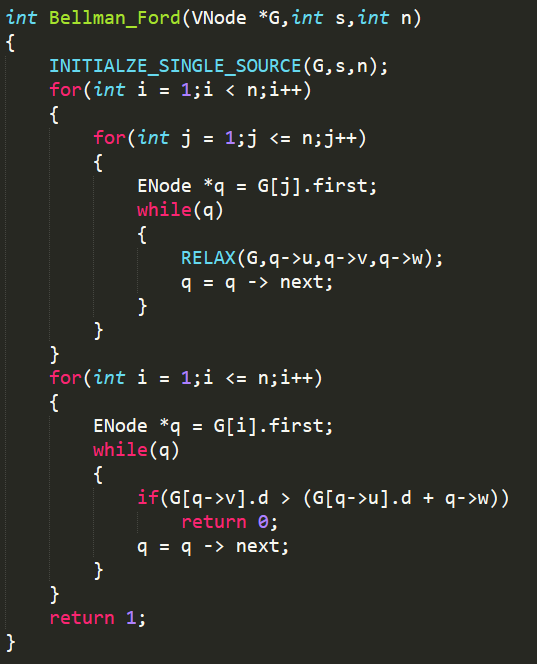
5存储到邻接链表



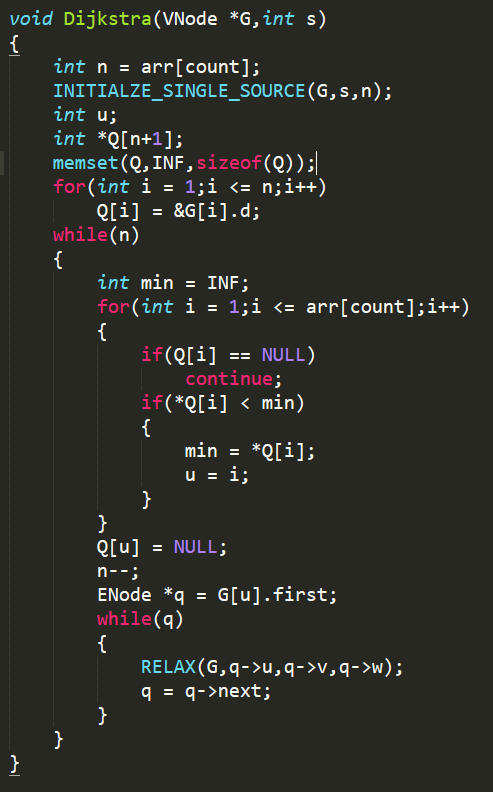
6初始化和RELAX函数



7 Bellman\_Ford算法



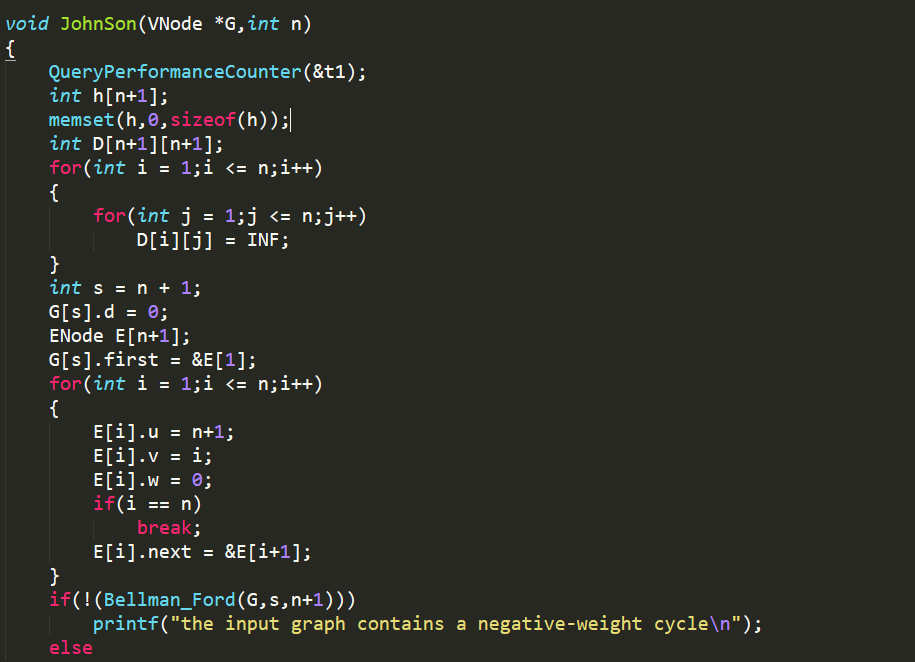
8 Dijkstra算法



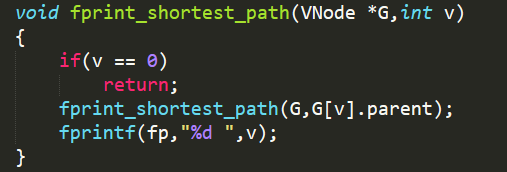
由于数据规模较小，所以这里使用简单的线性数组作为优先队列。

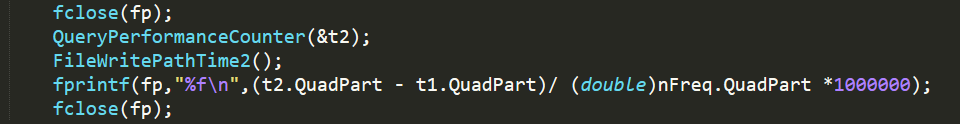
9 JohnSon算法

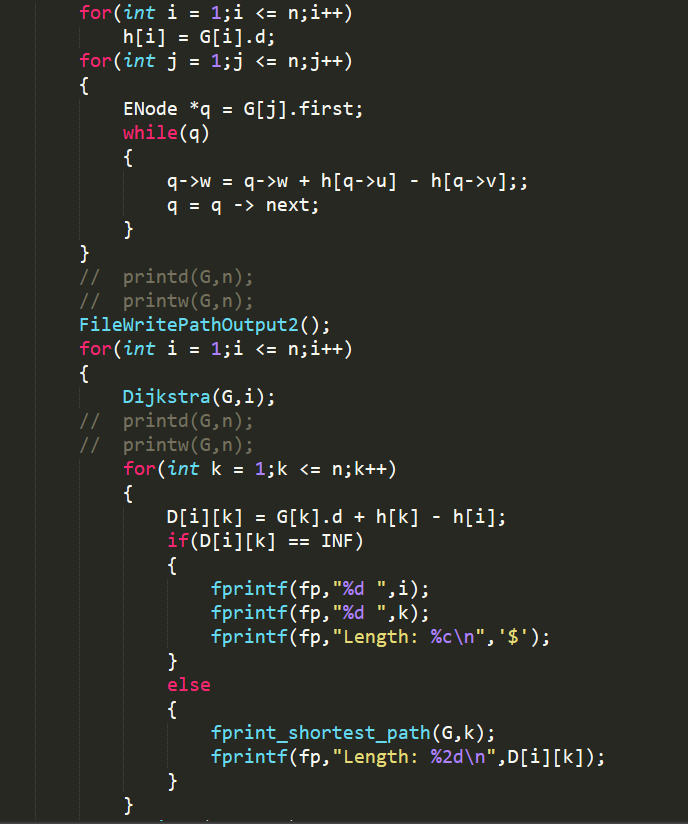
在结点数组最后增加s结点，增加n条指向n个结点的边，权值均为0，对s调用Bellman\_Ford算法。



若无负权值圈则计算每个h[i],然后重新计算边权值。最后对除s的每个结点调用Dijkstra算法，每次调用均根据parent指针递归打印最短路径经过的结点，再打印最短路径值。找到所有点对最短路径后打印运行时间。







**实验结果、分析（结合相关数据图表分析）**

**实验一中找到所有连通分量的时间（V+E = n+nlogn）**

可以看出基本满足算法导论关于强连通分量算法Θ（V+E）的论断。

**实验二中所有点对最短路径所消耗的时间(V=n)**

可以看出在n较大时基本满足算法导论关于使用线性数组作为Dijkstra算法的优先队列时，JohnSon算法的运行时间为Ο(V^3+VE)=Ο(V^3)的论断。

1. **实验心得**

通过对强连通算法的实现，复习了表示图的邻接链表和邻接矩阵，复习了DFS算法，并对强连通算法有了较深的理解。通过对JohnSon算法的实现，复习了Dijkstra算法和优先队列，理解了Bellman\_Ford算法。进一步学习了怎样测量程序运行时间以及图表的制作。