**深 圳 大 学 实 验 报 告**

**课程名称：­ 算法设计与分析**

**实验项目名称： 图论——桥的查找**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 马里佳**

**报告人： 张欣杰 学号： 2020151091**

**实验时间： 2022/05/27**

**实验报告提交时间： 2022/06/06**

**教务部制**

|  |
| --- |
| **一、实验目的：**  1. 掌握图的连通性  2. 掌握并查集的基本原理和应用 |
| **二、实验原理：**   1. 实现上述基准算法。 2. 设计的高效算法中必须使用并查集，如有需要，可以配合使用其他任何数据结构。 3. 使用文件 mediumG.txt 中的无向图测试基准算法和高效算法的性能，记录两个算法的运行时间。 4. 设计的高效算法的运行时间作为评分标准之一。 5. 提交程序源代码。 6. 实验报告中要详细描述算法设计的思想，核心步骤，使用的数据结构。 |
| **三、实验用品：**  操作系统：Windows 10  实验环境：Microsoft Visual Studio 2022 |
| **四、实验过程及内容：**   1. 基准法 2. 算法分析   算法基本思想：将图中某一边删除，计算删边后图的连通块是否增加，是的话说明该边是桥，否则不是。   1. 算法使用的数据结构   当数据量较大的时候，由于数据中的图为稀疏图，使用邻接矩阵会导致空间不足而无法运行，因此使用邻接表来储存该图。使用邻接表来存储矩阵的时候，空间比邻接矩阵节省了许多，但是也存在局限性，在本次实验中，图是无向图，因此使用邻接表存储的时候，对称边需要存储再另一个链表中，再操作的时候需要重复地寻找一条边的对称边并做相同的修改，因此可以在邻接表的基础上，再节点成员中增加一个edge结点指针，用来指向对称边节点的指针，实现O(1)时间内访问对称边，从而提高算法的效率   1. 基准法算法伪代码   广度优先搜索生成树：    计算连通分量    基准算法核心伪代码     1. 高效算法 2. 算法分析   实现高效算法前，首先考虑：任意一个图中的边，除了重复边和自环边（指向自身的边），就只有桥和环边（该边参与构成一个环），所以如果能够从图中计算环边、重复边和自环边的数量，就可直接用所有边数量减去上述几类边的数量，就得到桥的数量。在读取数据并存储的时候，若读取到重复边和自环边，则记录其数量、不将其加入存储，方便后续的算法实现。该算法使用到广度优先搜索，得到的生成树用并查集表示（存于数组tree中，tree[i]的值为点i的父节点在tree中的索引值(从0开始)；iii. 在生成树中，由于桥的存在，使得原本两个独立的连通块相连为一个连通块，所以所有的桥都必然出现在生成树中，而一部分没有进入生成树的边则全部是环边，结合下图解释这些非生成树环边如何处理：    如上图所示，很明显能看出，9-10和12-13是该图的桥，以4为根，进行广度优先搜索，得到下图，与原图对比，黑色边为非生成树环边。    将非生成树环边放入环边，可以看到生成树中多出来一个环，那假设该环包含的边为X，则有X-1条边再生成树中，剩下的那一条边是我们放进去的非生成树边，然后尝试将非生成树边8-9放入树种，4-8和4-9两条边标记并记录数量，然后再将8-9移除(如下，紫色为标记边)；放入9-13之后，也能形成环，但是4-8和4-9已经被标记，因此只需要标记8-13。以此类推，即可标记出了桥之外的所有边。     1. 算法伪代码   广搜生成树代码：    计数器核心伪代码：     1. 时间复杂度分析 2. 基准法   设图中顶点个数为n，边个数为e 。使用邻接表对边集进行遍历的时间复杂度为O(e)。一次 BFS 遍历需遍历所有点，时间复杂度为O (n+e) 。又因为一共需要进行e次 BFS 操作，因此算法的总时间复杂度为：   1. 高效算法LCA   与基准法相比，高效算法一次查找公共祖先只需要O(n)，而基准算法需要O(n+e)，因此与基准算法相比，的算法执行e次查找降低了e\*O(e)，即没有了O(e^2)项的复杂度，因此优化是较为显著的。此外，查找的时间复杂度O(n)是最差情况，对于大数据量级下的查找操作，经过并查集的路径压缩，很快需要查找的节点基本上父节点大部分都已经被设置为最近公共祖先（LCA）。 |
| **五、实验现象及数据处理：**   1. 验证算法正确性 2. 基准法   使用文件给定小图验证算法正确性，在这个小图中，一共有16个点，边的数量为15条，其中桥的数量为6条，将图的数据依次输入，然后使用基准算法运行程序，检测算法正确性，结果如下    可以看到，结果正确。   1. 高效算法   同样使用16点15边的图验证算法的正确性，但是在我的算法中，由于是使用寻找总边数、环边、不在生成树的边的数量之间的关系来寻找桥，因此并未列出桥的坐标，列出的是整个图的桥的总数量。运行结果如下：    结果正确，其中Circle为自环边的数量，Repeat为重边的数量，由于给定的数据没有重边和自环边的数量，因此数量为0，Not in tree为不在生成树的边的数量，Cur为出了不在生成树的环边数量，因此最终结果为总边数-Cur-Not in tree=15-6-3=6。因此桥的数量为6。   1. 运行mediumG.txt 2. 基准法      1. 高效法      1. 运行大数据文件 2. 基准法   由于基准法运行的时间复杂度较高，在运行大数据(点的个数为1000000，边数为7586063)的时候，算法效率实在过低，因此在运行了3个小时之后依旧没有结果，因此基准法并不适合运行大规模数据。   1. 高效法、   相比于基准法来说，高效法的效率是远远高于基准法的，即使是1000000个点的超大图，高效法的运行时间也是比较快的。     1. 随机数据测试 2. 基准法     在基准算法中，即使是在稀疏图中，求解该问题的时间复杂度依旧是十分高的，在图的点数为10个的时候，运行时间为44us，但是当点数增大的时候，平均运行时间是急剧上升的，当点数达到10000的时候，平均运行时间达到了2500Wus，而点数到了20000运行时间，更是达到了1.1亿us。   1. 高效法     使用高效算法时，速度比基准算法快了许多，特别是在大规模数据的时候，高效算法算法在时间复杂度的优化上展现了巨大的优越性，在数据量达到20000的时候，运行速度是比基准算法100个数据时候要快的，因此可以看出，高效算法的时间开销相对于基准算法是极小的。 |
| **六、实验结论：**   1. 基准法时间复杂度为，而高效算法能达到O(e)，两种在应用大数据运行的时间差别极大，基准法花费的时间超过60小时，而高效算法只需5秒多，约为基准算法的40000倍，这体现了算法效率的重要性。 2. 使用BFS会比DFS更好，DFS会形成栈溢出，而且时间也会偏长。而且储存点使用用链表而不是二维数组，更加节省空间。 3. 当数据量在较小的时候，使用基准算法所花费的时间是比较小的。但是当数据量很大的时候，再使用基准算法来进行递归就要花费很多的时间。而使用并查集的算法求桥的时间是比较少的，因为并查集可以大大提高找桥的效率。 |
| **思考题：** |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  **年 月 日** |
| **备注：** |

**注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。**

**2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内**。