**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： 实验5 图论（桥）**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 杨烜**

**报告人： 谢弘烨 学号： 2020151036**

**实验时间： 2022年5月30日**

**实验报告提交时间： 2022年5月30日**

**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：** | | |
| **三、实验用品：**   1. Visual Studio 2022 2. MicroSoft Office 2019 | | |
| **四、实验过程及内容：**  **1. 桥的定义**  在图论中，一条边被称为“桥”代表这条边一旦被删除，这张图的连通块数量会增加。等价地说，一条边是一座桥当且仅当这条边不在任何环上。一张图可以有零或多座桥。  https://upload.wikimedia.org/wikipedia/commons/thumb/b/bf/Undirected.svg/125px-Undirected.svg.png  图 1 没有桥的无向连通图  https://upload.wikimedia.org/wikipedia/commons/thumb/d/df/Graph_cut_edges.svg/200px-Graph_cut_edges.svg.png  图 2 这是有16个顶点和6个桥的图（桥以红色线段标示）  **2. 求解问题**  找出一个无向图中所有的桥。  **3. 算法**  （1）基准算法  For every edge (u, v), do following  a) Remove (u, v) from graph  b) See if the graph remains connected (We can either use BFS or DFS)  c) Add (u, v) back to the graph.  **（2）应用并查集设计一个比基准算法更高效的算法。不要使用Tarjan算法，如果使用Tarjan算法，仍然需要利用并查集设计一个比基准算法更高效的算法。** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **五、实验现象及数据处理：**   1. **问题描述**    1. **桥的定义**   一条边被称为“桥”代表这条边一旦被删除，这张图的连通块数量会增加。等价地说，一条边是一座桥当且仅当这条边不在任何环上。一张图可以有零或多座桥。   * 1. **求解问题**   求出给定无向图中所有的桥边   1. **基准算法**    1. **基本思路**   由桥的定义可知：在无向图中删除桥边会导致图中联通分支数增加。那么，对于图中的每一条边都进行删除后统计连通分支数的操作，若分支数增加，则被删除的边就是桥边。   * 1. **伪代码**   For every edge (u, v), do following  a) Remove (u, v) from graph  b) See if the graph remains connected (We can either use BFS or DFS)  c) Add (u, v) back to the graph.   * 1. **效率分析**   不妨设图中顶点个数为n，边个数为e，则：  以邻接表保存无向图的DFS算法时间复杂度为：  遍历所有边进行DFS求连通分支操作，即基准算法时间复杂度为：  稀疏图时为：  稠密图时为：  依照上述思路编写代码实现，以材料中给出的16点数据验证：    可见算法正确。  对于随机生成的地图数据（）：   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 | | 实测 | 1.12 | 4.26 | 8.48 | 14.22 | 20.59 | 31.32 | 40.93 | 53.31 | 69.89 | 82.25 | | 理论 | 0.82 | 3.29 | 7.40 | 13.16 | 20.56 | 29.61 | 40.30 | 52.64 | 66.62 | 82.25 |   可见实测效率贴合较好   1. **基准算法优化**    1. **基本思路**   基准算法操作过程中对于每一条边删除之后都需要进行全图遍历，如果此图中存在多个连通分支，那么对于这条边之外的连通分支的遍历就是无用的操作。  同时，由于桥边两端分别为两个较为独立的连通分支，删除桥边后左侧分支的点与右侧分支里的点必定不再连通。  因此，对边进行删除操作后，仅需要由该边的一个端点开始，如果能够到达另一个端点，说明这两点间除删除的边之外还有别的路径，则这条边必定不是桥边。   * 1. **效率分析**   该优化过程没有涉及基准算法核心过程的改变，因此时间复杂度与前相同。  对于给出的mediumG.txt，测得基准算法优化前后效率为：   |  |  | | --- | --- | | **基准** | **基准优化** | | **1.0714** | **0.7575** |   可见优化有效但不甚明显，仍需进一步寻求效率更高的高效算法。   1. **高效算法**    1. **基本思路**   由桥的定义易得，桥边必定出现在无向图的生成树上，同时桥边必定不出现在环中，则无向图中除环边以外的所有边均为桥边。  因此，在读入边集时，可以利用并查集将边集提前分成树边与环边两类。再遍历环边，向生成树中添加该环边，求出生成树中的环边。  最后剩下的所有树边即为桥边。   * 1. **LCA求环边**   LCA（Least Common Ancestors），最近公共祖先。对于树结构，其中的每一个节点都有其祖先节点，而任意两个节点的祖先节点至少有一个公共节点，即最近公共祖先。  对于无向图的生成树，一条环边的两个端点的到其最近公共祖先节点的所有路径均为环边。    如该生成树，若添加环边11-15，则从11、15到其最近公共祖先10经历过的所有的边均为环边。   * 1. **路径压缩**   在上述LCA求环边的过程中，容易出现一种重复情况：后添加的一条边在求其两个端点的最近公共祖先的过程中，大部分路径以及在之前某条边求最近公共祖先过程中经过。此时不光会产生重复计算环边，还会浪费大量资源。  在这里插入图片描述  如图，先后添加重边1、2时，可以发现添加重边2进行LCA的过程中会重复经过添加重边1所经过的路径。  解决办法：  在这里插入图片描述  直接将3、7的父节点设置为其最近公共祖先可以大大压缩4、8寻找最近公共祖先的路径。   * 1. **伪代码**   PERFORMANCE(n)  For edge(u, v) in loopEdges:  Tmp\_v=v  Tmp\_u=u  While tmp\_u.height < tmp\_v.height:  Tmp\_v = tmp\_v.parent  While tmp\_u.height > tmp\_v.height:  Tmp\_u = tmp\_u.parent  While tmp\_u.parent != tmp\_v.parent:  Tmp\_v=tmp\_v.parent  Tmp\_u=tmp\_u.parnet  v.parent = tmp\_v.parent  u.parent = tmp\_u.parent   * 1. **效率分析**   要求出生成树上所有的环边，就需要对以及确定的环边集进行遍历，该过程时间复杂度为：  对于每条边都进行LCA求其两个端点的最近公共祖先，该过程时间复杂度为：  总体时间复杂度应为𝑂(𝑛𝑒)，相较于基准算法优化了  然而由于路径优化的存在，在实际执行算法过程中单次LCA复杂度几乎。总体时间复杂度可近似为  稀疏图时为  稠密图时为  依照上述思想编写代码实现，以16点图验证：    可见算法正确  对于mediumG.txt有：   |  |  |  |  | | --- | --- | --- | --- | |  | **基准** | **基准优化** | **高效** | | **小规模** | **1.0714** | **0.7575** | **0.0050** | | **大规模** | **//** | **//** | **1139.667** |   可见高效算法优化效果明显  对于随机生成的地图数据（）：   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 | | 实测 | **0.0051** | **0.0110** | **0.0158** | **0.0209** | **0.0268** | **0.0326** | **0.0380** | **0.0429** | **0.0475** | **0.0526** | | 理论 | **0.0053** | **0.0105** | **0.0158** | **0.0210** | **0.0263** | **0.0316** | **0.0368** | **0.0421** | **0.0473** | **0.0526** |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **50** | **100** | **150** | **200** | **250** | **300** | **350** | **400** | **450** | **500** | | **基准** | **1.12** | **4.26** | **8.48** | **14.22** | **20.59** | **31.32** | **40.93** | **53.31** | **69.89** | **82.25** | | **高效** | **0.0051** | **0.0110** | **0.0158** | **0.0209** | **0.0268** | **0.0326** | **0.0380** | **0.0429** | **0.0475** | **0.0526** | |
| **六、实验结论：**   1. 常规基准算法的效率常常是比较低的。我们可以转换思路，利用排除法从而提高算法的效率。 2. 在对于一些图的算法中，可以通过借助树面对算法进行优化 3. 算法正确性验证是必要的，更是算法效率优化的基础 4. 选择合适的数据结构可以使得算法有更低的时间复杂度 5. 在对于图的算法中，需要注意在稀疏图和稠密图下不同的时间复杂度 |

|  |
| --- |
| **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**。