**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计**

**实验名称： 桥**

**学院：计算机与软件学院 专业： 计算机科学与技术（创新班）**

**报告人： 何泽锋 学号： 2022150221 班级： 高性能特色班**

**指导教师： 杨烜**

**实验时间： 2024年5月26日-2024年5月30日**

**实验报告提交时间： 2024年6月3日**

**教务处制**

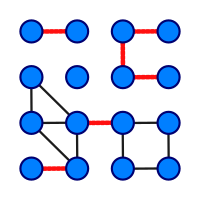
**一．实验目的**

1. 掌握图的连通性。

2. 掌握并查集的基本原理和应用

**二．实验要求**

1. 实现基准算法。
2. 设计的高效算法中必须使用并查集，并用例子验证算法正确性。



1. 使用文件 mediumG.txt和largeG.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。

**三．实验原理及数据分析**

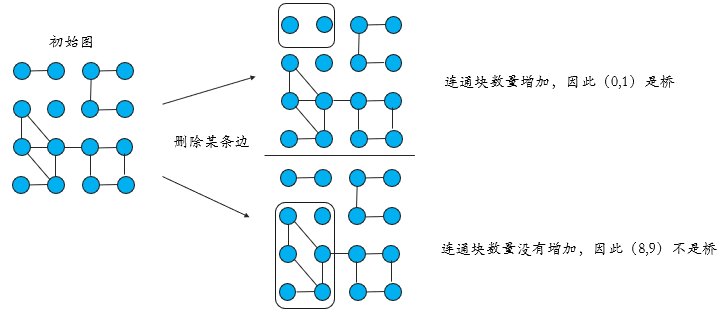
**1.基准算法**

（1）**算法原理**：首先通过DFS递归计算无向图的连通块数量，然后将无向图的每条边从图中依次移除，判断图的连通性是否改变，若发生改变则此条边为桥；接着将移除的边恢复，对下一条边进行判断，直到遍历完所有边。

（2）如图所示，对于每条边，删除后存在两种情况

①**删除后连通分量数量增加，说明该边是桥边**

②**删除后连通分量不变，说明该边是环边，不是桥**



（3）利用该图验证基准法正确性，并继续使用mediumG.txt和largeG.txt图验证效率，具体时间如下所示，可以发现前两图较快得到答案，但最后一幅图无法在有限时间内求出结果



（4）时间复杂度：

①在**稀疏图（e=n）**情况下，时间复杂度为

②在**稠密图（e= ）**情况下，时间复杂度为

（5）时间复杂度证明：

**I.稀疏图**

①需要检查的边数：边的数量大约是 ，即 ，其中 n 是顶点的数量。

②检查一条边是否为桥的过程：在稀疏图中，每个顶点的度数相对较小，因此搜索的时间复杂度可以认为，但由于 E ≈ V，这可以简化为

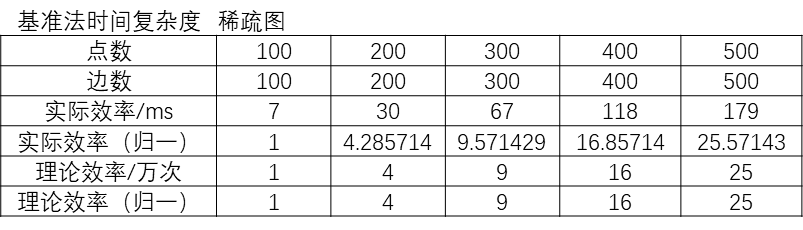
**II.稠密图**

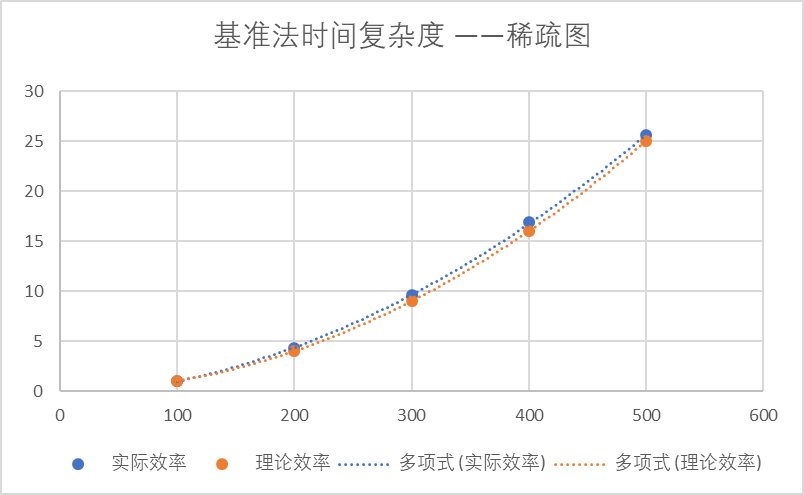
①边的数量大约为 ,每次检查都需要进行一次深搜索，其时间复杂度为。

②但是，这个时间复杂度分析没有考虑到在深度优先搜索中寻找桥的过程中，还需要对每个顶点的邻接表进行遍历，这将增加一个额外的 的时间复杂度。因此，总的时间复杂度会变为 = 。

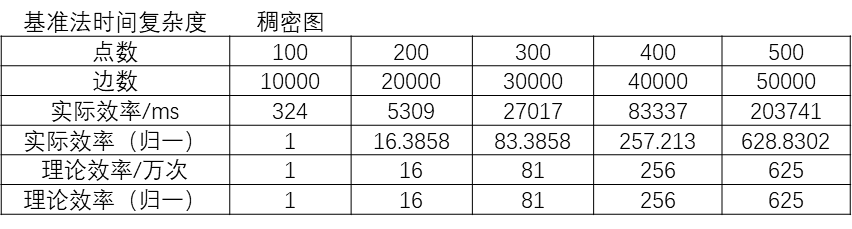
（6）数据验证：

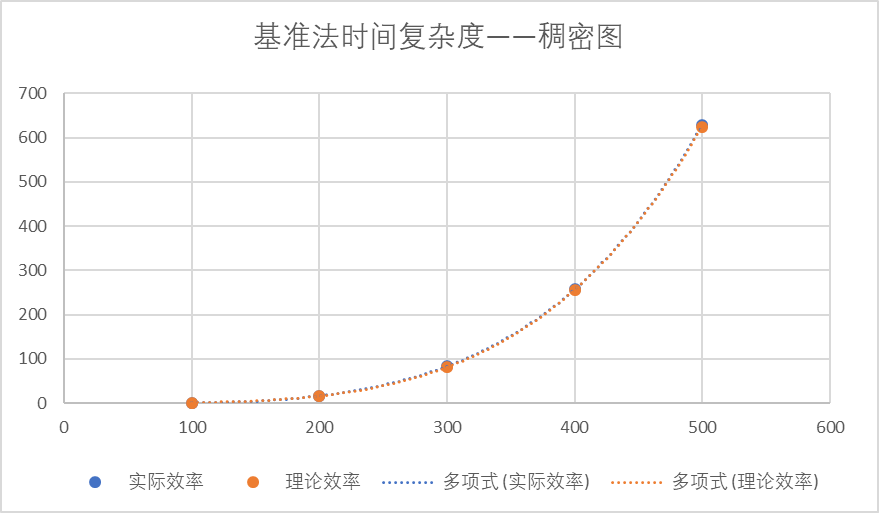
①稀疏图





②稠密图





**2.并查集法**

（1）**基本原理**：与基准法思路相同，通过删除边并计算连通块数目来查找桥，计算连通分支时使用并查集。并查集计算连通分支数目的步骤为：枚举边，对每个边上的两点v1和v2，查询v1和v2所属的集合f1,f2，如果v1和v2不在同一个集合则合并v1和v2所属的两个集合，最后统计集合的个数，即为连通分支数目。

（2）利用该图验证基准法正确性，并继续使用mediumG.txt和largeG.txt图验证效率，具体时间如下所示，可以发现前两图较快得到答案，但最后一幅图无法在有限时间内求出结果，结果与基准法接近



（3）时间复杂度：

①在**稀疏图（e=n）**情况下，时间复杂度为

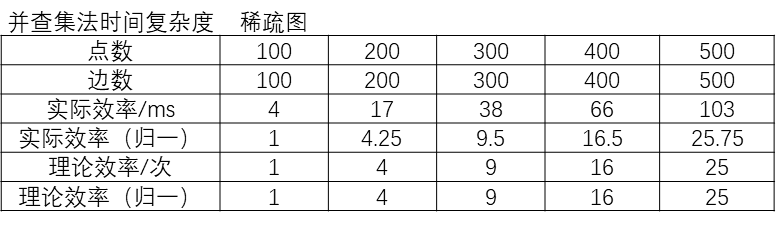
②在**稠密图（e= ）**情况下，时间复杂度为

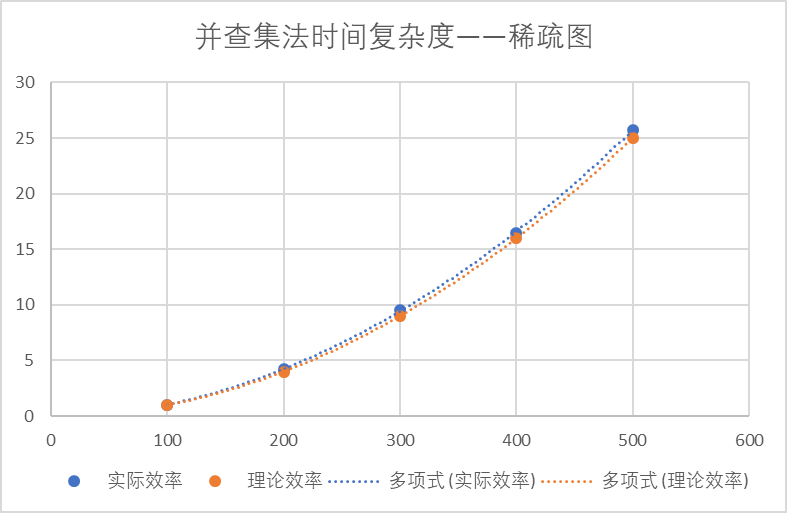
（4）时间复杂度证明:

此处不做过多说明，其方法与基准法基本完全一致，区别在于连通块的记录方式，因此复杂度证明方式与基准法一致

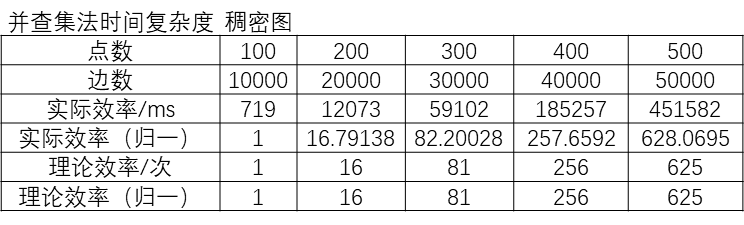
（5）数据验证：

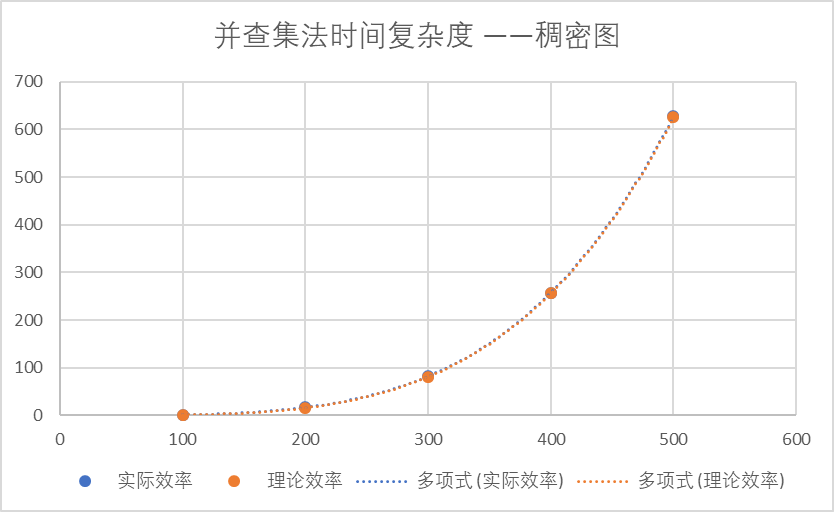
①稀疏图：





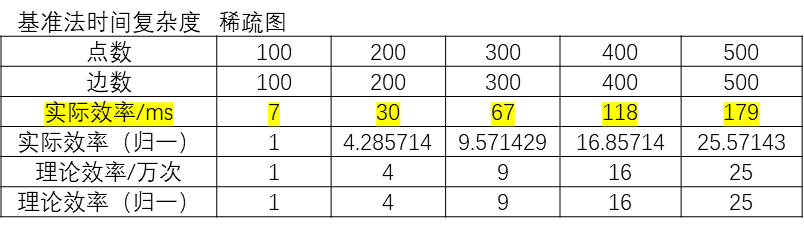
②稠密图：

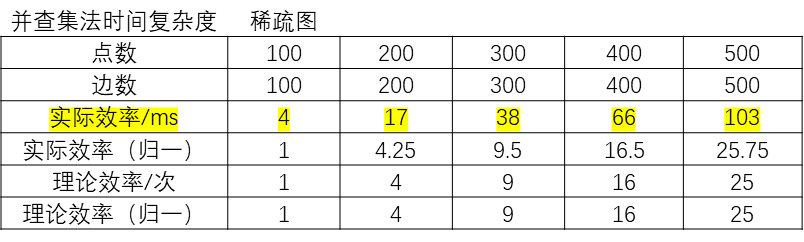




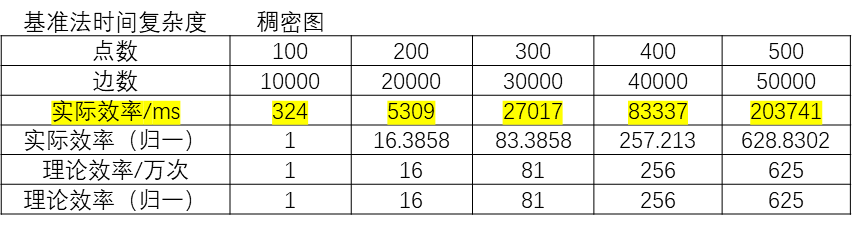
（6）简单对比，虽然基准法与并查集法在时间复杂度上基本一致，但二者在稀疏图和稠密图中的实际耗时有所不同

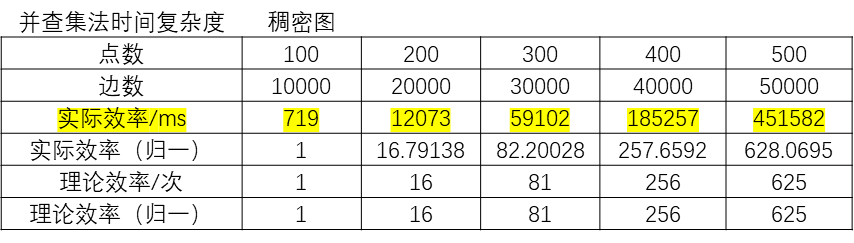
①稀疏图情况下，可以看到并查集的时间比基准法少





②稠密图情况下，可以看到基准法比并查集时间少





③分析，在稀疏图情况下使用并查集可以较快进行集合合并，而基准法需要多层递归才能得到连通分量。而在稠密图情况下，并查集没有使用路径压缩等优化方法，需要逐层往上递归寻找父节点，因此实际耗时更长

**3.优化并查集**

（1）**基本原理**：优化并查集法是基于并查集的原理

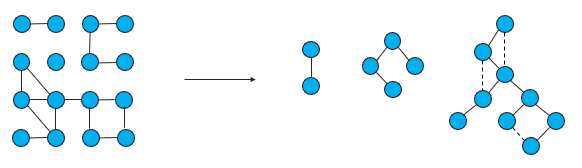
①首先通过DFS生成树，此时需要记录的是每个点的深度、父节点。

②再次遍历边，此时判断边连接的两点的父子关系，若不是父子关系则说明为非树边，因此记录为环边。

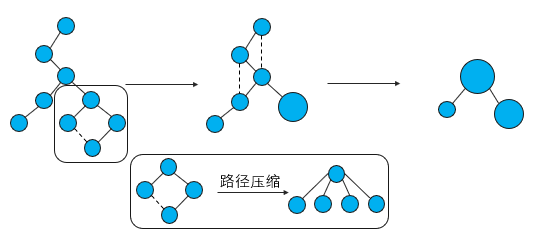
③然后通过添加环边，利用LCA将环压缩为点，LCA是指将环上的各点都用最近的公共祖先点来替代，此时遍历到环上的点都会被视为同一个点，因此实现了环的压缩。此时树上剩余的边即为桥

（2）图示过程：

①首先将图中的点和边生成树，如图中实现即为树边，虚线是非树边,是环，不会在树中记录，此图生成了三棵树，实际上是将图生成为森林



②添加环边，利用LCA压缩环。添加环边后，通过寻找最近公共祖先从而将环内的点压缩为公共祖先点，而压缩采用了并查集的路径压缩，可以加快压缩过程，不需要一层层寻找父节点。



③压缩完后，树中剩余的边即为桥

（3）利用该图验证基准法正确性，并继续使用mediumG.txt和largeG.txt图验证效率，具体时间如下所示，可以看到在ms时间内小图和中图都能很快得到结果，而大图也能在1s内计算得到结果



（4）时间复杂度：

①在**稀疏图（e=n）**情况下，时间复杂度为

②在**稠密图（e= ）**情况下，时间复杂度为

（5）**时间复杂度证明**：

①通过DFS构建最小生成树，复杂度为

②LCA查找公共祖先，最差情况下遍历n次，即

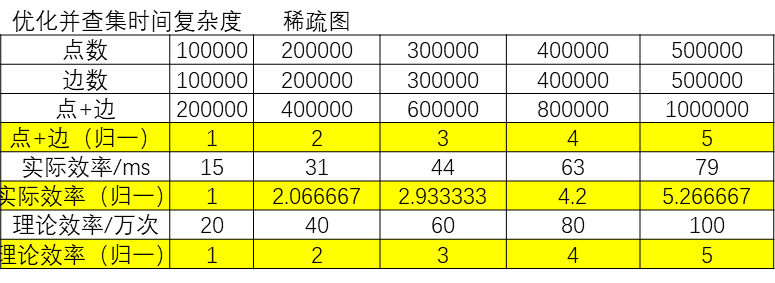
③压缩路径复杂度

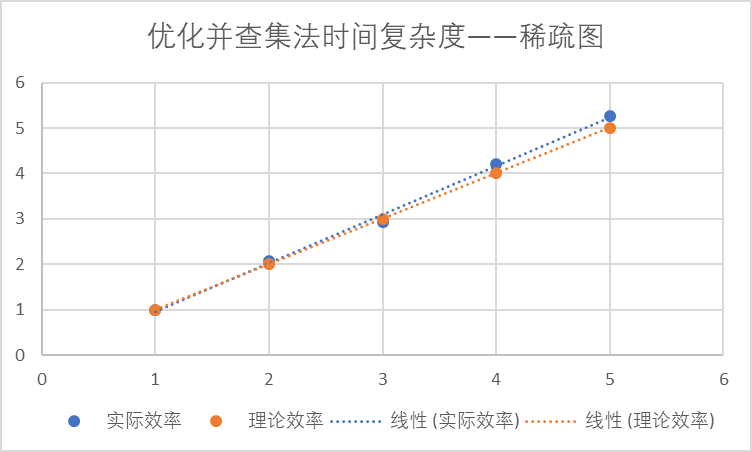
④添加环边的次数最差e次，

因此最终复杂度为

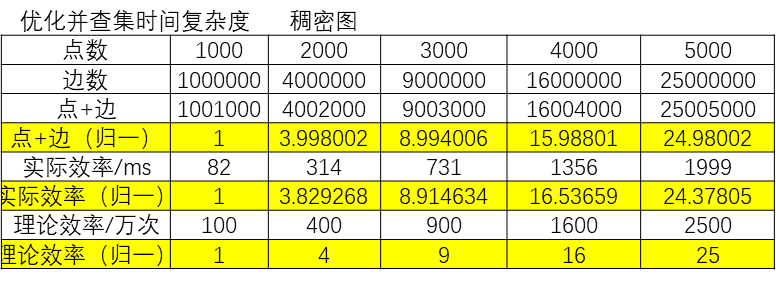
（6）数据验证：

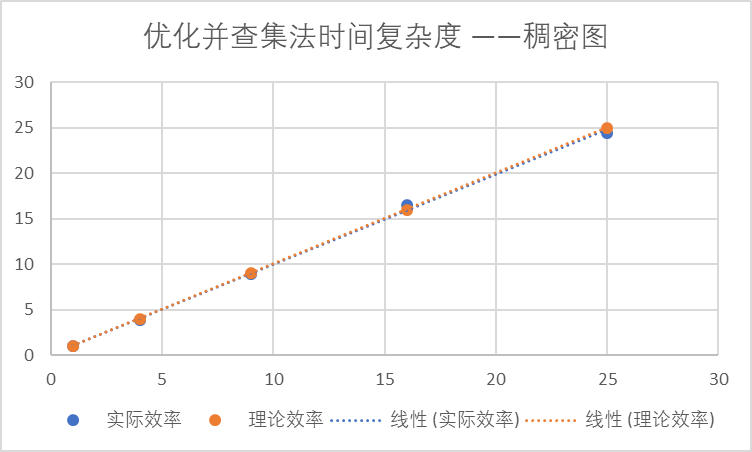
①稀疏图





②稠密图





**4.其他优化**

（1）**图存储优化**

实验刚开始使用了邻接矩阵存储，当数据量大的时候无法开辟太大的空间，导致存储失败，因此修改为邻接表存储图，大大节省空间。

（2）**数据存储优化**

其余的数据如visit判断点是否遍历，father记录父节点等，刚开始使用vector存储，但发现建树时耗费大量时间，因此改为直接用数组存，建树时间从5s降为200ms

（3）**编译优化**

代码编译时采用了O2优化，即速度优先，虽然会增加代码的编译时间，但优化了数据结构，加快了数据查询等，最终使得寻找桥的时间减少

**四. 实验结论**

通过本次实验，对图中的关系了解更加深刻，通过DFS将图中的点生成为树，并且利用并查集求得了连通分量，LCA用于将环压缩为点，而这个环的代表点为环在树中的最近公共祖先。对于同一个算法，在处理稀疏图和稠密图时其时间复杂度可能不相同。因为图的规模问题，不能过度使用vector等数据结构，这会造成遍历过慢，适当使用数组会加快程序运行时间

|  |
| --- |
| 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。