**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： 桥-问题**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 陆玉武**

**报告人： 刘俊楠 学号： 2017303010 班级： 01**

**实验时间： 2020/05/31**

**实验报告提交时间： 2020/05/31**

**教务处制**

### 一个、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、内容：

**1. 桥的定义**

在图论中，一个条边被称为“桥”代表这条边一个旦被删除，这张图的连通块数量会增加。等价地说，一个条边是一个座桥当且仅当这条边不在任何环上。一个张图可以有零或多座桥。

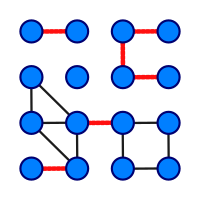
 

图 1 没有桥的无向连通图 图 2 这是有16个顶点和6个桥的图

（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.

**应用并查集设计一个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个比基准算法更高效的算法。**

### 三、实验要求

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. 用图2的例子验证算法正确性。
4. 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。
5. 设计的高效算法的运行时间作为评分标准之一个。
6. 提交程序源代码。
7. 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。

**四、实验过程**

1. **算法原理描述**
2. 基准方法
3. 基本思想

根据桥的定义，可以遍历图的所有边，依次删除并求得连通分量个数（DFS），若连通分量在删除边后增加，则该边是桥。

1. 实现思路

遍历所有边，并对每条边进行以下操作：

-删除该边

-利用DFS求图的连通分量，如果产生的新连通分量大于原来的连通分量数目，则该边判断是桥

-将边添加回来

1. 并查集
2. 并查集的基本操作

-查找：判断两元素是否在同一个集合

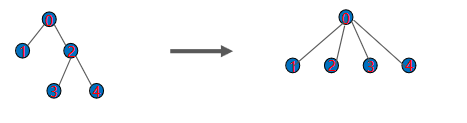
-合并：若两元素不在同一个集合，将它们合并为同一个集合

1. 实现思路

通常采用父指针表示法来表示树，用数组的索引来表示结点，数组中的元素表示当前结点的父节点。

1. 并查集+路径压缩

将原先与根间接相连的结点，直接与根相连，即找到该结点根时顺便把该结点连上去，可减少查找根的时间。



1. **算法实现的核心伪代码**
2. 基准方法

get\_bridge(graph \*G)

num\_bridge =0

conps = DFS\_T(G) // 求当前连通分量

for i=0 to G->num\_vex

\*p = G->vertexlist[i].edge

while(p)

delete edge // 删除边

new\_conps= DFS\_T(G) // 求新连通分量

if(new\_conps>conps) num\_bridge++

add edge // 恢复边

p=p->next

return num\_bridge

1. 并查集

-查找操作

Find(int x)

r = x, j;

    while(pre[x] != x)

         x = pre[x]

     while(r!=pre[r])    // 路径压缩（非递归方法）

         j = r

        r = pre[r]

        pre[j] = x

     return x

-合并操作

Union(int a,int b)

    int preA = Find(a)

    int preB = Find(b)

    if(preA != preB)

        pre[preA] = preB

1. **算法测试结果及效率分析**

1. MediumDG

|  |  |  |  |
| --- | --- | --- | --- |
| mediumDG | 基准算法 | 并查集+tarjin |  |
| 耗时(ms) | 470ms | 1ms |

1. **总结**
2. 通过小规模数据可知并查集的引入对稀疏图的算法性能有极大提高，同时再对并查集进行路径压缩优化，可以避免产生线性树，从而进一个步减少遍历次数，提高效率。
3. 在大规模数据中，由于基准算法使用的是DFS计算连通分量，而DFS使用了递归，递归需要压栈，因为数据规模大导致爆栈使得程序异常，所有没有跑出结果。
4. 利用tarjin算法与路径压缩，我们可以避开二维数组巨大的空间复杂度，实现计算百万级别的点集，并在较快的时间跑出结果。

**四．实验心得**

1. 通过本次实验，学习了并查集的基本原理和应用

2. 领会到了算法的魅力以及数据结构在计算机中的重要性，提示我们要用有限的资源去创造尽可能多的价值。

3．了解了tarjan算法的原理以及使用它的优越性，能够实现一维数组对图结构里桥的检索。

|  |
| --- |
| 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。