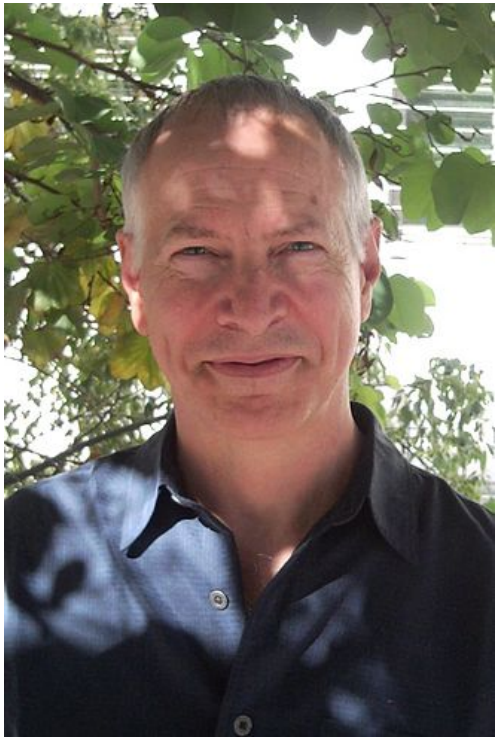
Tarjan's Off-line LCA Algorithm

杜星亮 stardustdl@163.com (mailto:stardustdl@163.com)

南京大学

Tarjan



Tarjan

Robert Endre Tarjan，美国计算机科学家，数学家。他参与的一些工作：

- Tarjan's off-line lowest common ancestors algorithm
- Tarjan's strongly connected components algorithm
- The Hopcroft-Tarjan planarity testing algorithm
- the median of medians linear time selection algorithm (co-authors)
- Fibonacci heaps (co-inventor)
- Splay trees (co-inventor)

Off-line

对于带查询的问题，分为离线和在线两种。

- 离线：在得到查询结果前 **能** 预先已知所有查询
- 在线：在得到查询结果前 **不能** 预先已知所有查询

Least Common Ancestor

有根树 T 中，两个结点 u,v 的 LCA 是指这样一个结点 w：

- w 是 u,v 的祖先
- w 是满足上述条件的结点中深度最大的

问题描述

给定

- 一棵有根树 T
- 一个由 T 中结点的无序对构成的集合 P={{u,v}}

确定 P 中每个对的最近公共祖先。

可能解法？

- 离线 LCA

可能解法？

- 离线 LCA
  - O(n^2)-O(1)

可能解法？

- 离线 LCA
  - O(n^2)-O(1)
  - More?

算法描述

```
for each vertex v in T
    v.color = WHITE
LCA(T.root)

LCA(u)
    MAKE-SET(u)
    FIND-SET(u).ancestor = u
    for each child v of u in T
        LCA(v)
        UNION(u, v)
        FIND-SET(u).ancestor = u
    u.color = BLACK
    for each node v such that {u, v} in P
        if v.color == BLACK
            print "The least common ancestor of" u "and" v
                "is" FIND-SET(v).ancestor
```

TC 21-3

a) 证明最后一行对每个对恰好执行一次

TC 21-3

a) 证明最后一行对每个对恰好执行一次

考虑结点被染成黑色的顺序。

TC 21-3

a) 证明最后一行对每个对恰好执行一次

考虑结点被染成黑色的顺序。

b) 证明调用 LCA(u) 时，当前集合数等于 u 的深度

TC 21-3

a) 证明最后一行对每个对恰好执行一次

考虑结点被染成黑色的顺序。

b) 证明调用 LCA(u) 时，当前集合数等于 u 的深度

进入函数时调用 MAKE-SET，LCA(x) 执行后，整棵 x 子树被合并到一个集合中。使用数学归纳法。

TC 21-3

a) 证明最后一行对每个对恰好执行一次

考虑结点被染成黑色的顺序。

b) 证明调用 LCA(u) 时，当前集合数等于 u 的深度

进入函数时调用 MAKE-SET，LCA(x) 执行后，整棵 x 子树被合并到一个集合中。使用数学归纳法。

c) 证明算法正确性

d) 分析运行时间

算法正确性

- 对任意一对 u,v
- 设 LCA(u,v)=w
- 则在 LCA(w) 过程中，会发生调用 LCA(u) 和 LCA(v)
- 考虑其调用先后顺序。

时间复杂度分析

```
LCA(u)
    MAKE-SET(u)                                 # O(1)
    FIND-SET(u).ancestor = u                    # O(1)
    for each child v of u in T
        LCA(v)
        UNION(u, v)                             # O(1)
        FIND-SET(u).ancestor = u                # Sum O(|T|alpha(|T|))
    u.color = BLACK                             # O(1)
    for each node v such that {u, v} in P
        if v.color == BLACK                     # Sum O(|P|)
            print "The least common ancestor of" u "and" v
                "is" FIND-SET(v).ancestor
```

- O(|T|alpha(|T|)+|P|)

Open Topic

关于 LCA 还有什么算法呢?

- 在线 LCA

Open Topic

关于 LCA 还有什么算法呢?

- 在线 LCA
  - O(nlog n)-O(log n)

Open Topic

关于 LCA 还有什么算法呢?

- 在线 LCA
  - O(nlog n)-O(log n)
  - O(nlog n)-O(1)

Open Topic

关于 LCA 还有什么算法呢?

- 在线 LCA
  - O(nlog n)-O(log n)
  - O(nlog n)-O(1)
  - More efficient?

Open Topic

关于 LCA 还有什么算法呢?

- 在线 LCA
  - O(nlog n)-O(log n)
  - O(nlog n)-O(1)
  - More efficient?
  - **O(n)-O(1)**

Open Topic

关于 LCA 还有什么算法呢?

- 相关问题与算法

Open Topic

关于 LCA 还有什么算法呢?

- 相关问题与算法
  - RMQ
    - Range Minimum/Maximum Query

Open Topic

关于 LCA 还有什么算法呢?

- 相关问题与算法
  - RMQ
    - Range Minimum/Maximum Query
  - DFS
    - Depth First Search

Open Topic

关于 LCA 还有什么算法呢?

- 相关问题与算法
  - RMQ
    - Range Minimum/Maximum Query
  - DFS
    - Depth First Search
  - Doubling Method
    - 倍增法

Open Topic

关于 LCA 还有什么算法呢?

- 相关问题与算法
  - RMQ
    - Range Minimum/Maximum Query
  - DFS
    - Depth First Search
  - Doubling Method
    - 倍增法
  - Divide into Block
    - 分块法

Open Topic

关于 LCA 还有什么算法呢?

- 动态树上LCA?

Open Topic

关于 LCA 还有什么算法呢?

- 动态树上LCA?
  - Link-Cut-Tree
    - Daniel Dominic Sleator & Robert Endre Tarjan
    - (1983) “A Data Structure for Dynamic Trees”
  - Splay
    - Daniel Dominic Sleator & Robert Endre Tarjan
    - (1985) “Self-Adjusting Binary Search Trees”
    - PS: 

Thank you