Tarjan 离线 LCA 算法及其线性改进

问题求解 Open Topic IV

黄文睿　学号: 221180115

南京大学　2022 级计算机拔尖班

2023 年 6 月 2 日

LCA 问题的常见算法

算法	在/离线	时间复杂度	空间复杂度
倍增法	在线	$O(n\log n)\sim O(\log n)$	$O(n\log n)$
树链剖分法	在线	$O(n)\sim O(\log n)$	$O(n)$
Link Cut Tree 法	在线	$O(n\log n)\sim O(\log n)$	$O(n)$
RMQ + ST 表	在线	$O(n\log n)\sim O(1)$	$O(n\log n)$
Tarjan LCA	离线	$O(n+q\alpha(q+n,n))$[1]	$O(n+q)$
Tarjan LCA(改进)	离线	$O(n+q)$	$O(n+q)$
标准 RMQ	在线	$O(n)\sim O(1)$	$O(n)$

其中 n 为结点个数，离线算法中 q 为询问数，在线算法时间复杂度 $O(T_1)\sim O(T_2)$ 表示预处理 $O(T_1)$，单次询问均摊 $O(T_2)$.

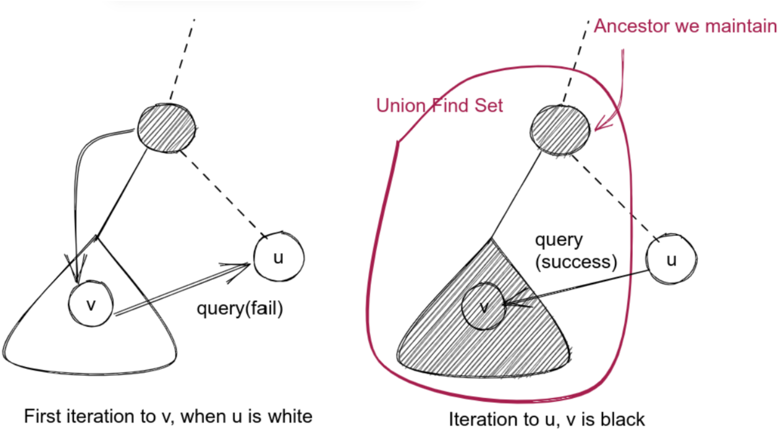
以上复杂度分析均在 RAM 模型下进行.

[1] $\alpha(m,n)=\min\{i\geq 1 : A_i(\lfloor m/n\rfloor)\geq \log n\}$，后续会用到以分析更精确的界.

Tarjan 离线 LCA 算法

```
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} ∈ P
      if v.color == BLACK
          print "The least common ancestor of"
              u "and" v "is" FIND-SET(v).ancestor
```

算法解释


u
v
query(fail)
First iteration to v, when u is white
Ancestor we maintain
Union Find Set
query
(success)
u
v
Iteration to u, v is black

简要证明

注意到以下事实:

- 如果 u 为白色而 v 为黑色，那么所有以 u 的祖先到 LCA(u, v)(不含 LCA) 为根的子树都已经合并成了一个连通块 (因为它们所在的递归已经完整退出).
- LCA(u, v) 还在执行中 (因为 v 是 LCA(u, v) 的子孙)，故它尚未与它的父亲合并．由于算法的执行过程保证该连通块的代表元 (或者说代码中的 ancestor) 是深度最小的节点，也就是 LCA.

算法复杂度分析

- 空间复杂度显然为 $\Theta(n+q)$.
- 时间复杂度分析如下:
  1. 上界分析: 在每次递归, 除去处理询问的部分, 其余部分均花费常数时间, 合计时间 $O(n)$; 每次处理询问需要调用一次 FIND-SET, 单次均摊为 $O(\alpha(q+n,n))$. 故上界为 $O(n+q\alpha(q+n,n))$.
  2. 下界分析: Tarjan 的一篇论文[2]给出了并查集在最坏情况下的下界为 $\Omega(n+q\alpha(q+n,n))$. 该下界证明容易拓展到 Tarjan 离线 LCA 算法.

故时间 $O(n+q\alpha(q+n,n))$ 已达理论最优.

[2]Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. Journal of Computer and System Sciences, 18(2):110–127, 1979

一种改进到线性的做法

Gabow 和 Tarjan 提出了一种特殊情况下的并查集方法[3]: 当并查集的 Union 过程已知 (不要求顺序已知, 只要求操作已知) 时, 在 RAM 上可以将 n 次 Make-Set、q 次 Find-Set 和 Union 操作的并查集改进成线性 $O(n+q)$ 复杂度.

[3] Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. Journal of Computer and System Sciences, 30(2):209–221, 1985

算法描述

由于 UNION 过程已知, 可以将需要执行 UNION(u, v) 的元素之间建边, 则可以形成一棵 "合并树"[4]. 任选一点为根, 只需完成以下两种操作:

- UNION-TREE(u): 把 u 和 $u.p$ 合并到同一个连通块中, 并把原来 $u.p$ 所在连通块的代表元设置为新连通块的代表元.

  (对每个结点至多执行一次 UNION-TREE 操作, 且不对根执行).
- FIND-TREE(u): 查找 u 所在连通块的代表元.

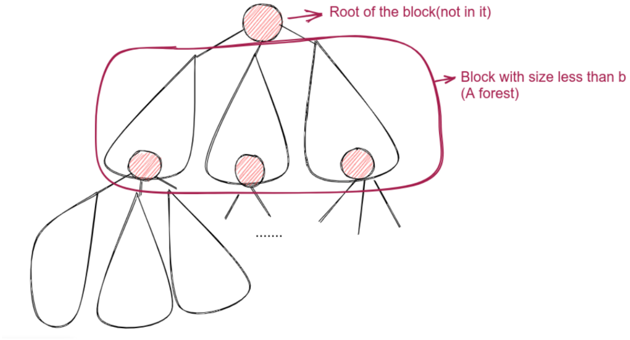
[4]如果 UNION(u, v) 操作少于 $n-1$, 得到的是一个森林, 但不影响讨论.

主要思想

对合并树进行分块, 在**块**内使用查表快速 FIND-TREE, **在块的根**之间使用传统的 α 并查集进行 FIND-TREE.

利用这种方法, 可以将时间复杂度优化到 $O(n+q)$, 空间复杂度仍为 $O(n+q)$.

分块方法



每一个**块**的大小 $< b$，是一个森林，森林中的每一棵树都是原树中的一个连通块，并且它们的深度最低结点有着公共的父亲，称作这个块的**根**．注意根不在块中．

进行查询

- 如果一个结点 u 已经执行过 Union-Tree(u) 了, 称它是**被标记**的. 被标记的结点不是代表元, 那么 Find-Tree 中, 要寻找的是离 u 最近的**未被标记**的祖先.
- 这个祖先可能在块中, 也可能不在块中.
- 于是 Find-Tree(u) 分为两步, 顺序执行:
  1. **先块内查询** Find-In-Block.
  2. 若未找到, **再块间查询** Find-Across-Block.

块内查询

- $\textsc{Find-In-Block}(u)$: 在块内查询 u 未被标记的祖先. 若未找到, 返回**块的根**.

- 主要思想: 查表. 块中最多 $b-1$ 个元素.
  1. 枚举父子关系 $p(i,j)=0/1$ 表示 i 是否是 j 的父亲;
  2. 枚举每个结点的标记状态 $m(i)=0/1$;
  3. 枚举当前查询的 u.

  如果各用一个二进制数来表示 p,m,u, 其中 p 需要用 $2^{(b-1)\lceil\log b\rceil}$ 位表示. 若 RAM 字长 $w=O(\log n)$, 那么取 $b=O(\log n/\log\log n)$ 即可.
  并且答案表 table $=(p,m,u)$ 的空间为 $O(n)$, 查表时间为 $O(1)$, 从而 $\textsc{Find-In-Block}$ 时间为 $O(1)$.

- Gabow 和 Tarjan 在原文给出了 $O(n)$ 构造 table 的方法.

块间查询

- 若 $\textsc{Find-In-Block}(u)$ 返回 u 的块的根，需要进行块间查询 $\textsc{Find-Across-Block}(x)$，其中 x 是 u 的块的根.
- 在每个**块的根**之间建立 α 并查集，有 $O(n/b)$ 个结点.

  $\textsc{Find-Across-Block}(x)$ 流程如下:

  1. 执行 $x \leftarrow \alpha\textsc{Find}(x)$.
  2. 执行 $y \leftarrow \textsc{Find-In-Block}(x)$. 如果 y 与 x 同块，则返回 y; 否则执行 $\alpha\textsc{Union}(y, x)$，并 $x \leftarrow \alpha\textsc{Find}(x)$.

  其中 α 并查集用来快速排除某段路径：在已经合并了的块的根之间不会有未被标记的结点.
- 时间复杂度为 $O((q + O(n/b))\alpha(q + O(n/b), O(n/b)) + O(n/b))$，当取 $b = \Omega(\log\log n)$ 时为 $O(n)$.

合并操作与分块构建

- $\textsc{Union-Tree}(u)$ 是 $O(1)$ 的, 只需要更新块内标记即可.
- 预处理分块容易在 $O(n)$ 内通过一次遍历完成.

结论

- 在分块中取 $b = \Omega(\log\log n)$ 并 $b = O(\log n / \log\log n)$, 可以在 $O(n+q)$ 时间, $O(n)$ 空间内解决 RAM 模型上的已知合并结构的并查集查询合并问题.
- 在 Tarjan 离线 LCA 问题中使用上述算法, 可以优化到 $O(n+q)$ 时间, $O(n+q)$ 空间.
- 尽管可能并不实用, 但具有理论价值.

总结

主要介绍了 Tarjan 离线 LCA 算法及其线性改进算法.

- Tarjan 离线 LCA 算法是一种时间复杂度 $O(n + q\alpha(q + n, n))$, 空间复杂度 $O(n + q)$ 的离线算法, 在实践中可以认为是线性的.

- Gabow 和 Tarjan 提出了一种利用 RAM 模型的特性, 并利用合并过程事先已知的并查集的特性的算法, 时间为 $O(n + q)$, 空间为 $O(n)$ 的算法, 应用到 Tarjan 离线 LCA 上, 可得到时间、空间均为 $O(n + q)$ 的算法, 具有一定理论价值.

参考文献

[1] Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. Journal of Computer and System Sciences, 18(2):110–127, 1979

[2] Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. Journal of Computer and System Sciences, 30(2):209–221, 1985

谢谢大家!