算法设计与分析作业四

作者：吴润泽　　**学号：**181860109

Email：181860109@smail.nju.edu.cn

2020 年 4 月 13 日

目录

Chapter 4

problem 4.2

(1)

⇒ 如果 w 是 v 在 DFS 树中的后继结点，那么 $actice(w) \subseteq active(v)$: 当 $w \neq v$ 时，因为 w 是 v 的后继节点，所以 $v.discover < w.discover$, 并且 $v.finish > w.finish$。所以 $actice(w) \subset active(v)$。当 $w = v$ 时显然成立。⇐ 如果 $actice(w) \subseteq active(v)$，那么 w 是 v 在 DFS 树中的后继结点: 当 $w \neq v$ 时，因为 $active(w) \subseteq active(v)$，即 $v.discover < w.discover$，并且 $v.finish > w.finish$，即在遍历 v 的过程中将 w 遍历，即 w 是 v 的后继结点。

(2)

由 (1) 可知，w 不是 v 的后继结点 ⇔ $actice(w) \not\subset active(v)$。v 不是 w 的后继结点 ⇔ $actice(v) \not\subset active(w)$ 得证。

(3)

① ⇒ 如果 vw 是 CE，那么 v 和 w 没有祖先和后继关系，由 (2) 可知 active(w) 和 active(v) 互不包含。同时 CE 说明在 v 指向 w 时，w 已经是黑色节点，w 已经遍历结束，所以 active(w) 在 active(v) 之前。

⇐ active(w) 在 active(v) 之前，w 先完成整个遍历过程，后才遍历到 v。且二者没有祖先后继关系，那么边 vw 即为 CE。

② ⇒vw 是 DE，即 v 指向 w 时 w 为黑色，并且 $active(w) \subset active(v)$，若不存在第三个节点 x，满足 x 是 v 的后继，w 是 x 的后继，则 v 遍历到 w 时 w 一定为白色，边为 TE。即一定有 $active(w) \subset active(x) \subset active(v)$。

⇐ 如果存在结点 x，满足 $active(w) \subset active(x) \subset active(v)$，由 (1) 可知，x 是 v 的后继，w 是 x 的后继，且在遍历时 v 先走到 x，然后 x 走到 w，即 v 是 w 的祖先结点，因此 vw 是 DE。

③ ⇒vw 是 TE，即 v 指向 w 时 w 为白色，w 是 v 的后继，由 (1) 可知，$active(w) \subset active(v)$。若存在 x，满足 $active(w) \subset active(x) \subset active(v)$，则在遍历时 v 先走到 x，然后 x 走到 w，那么 v 是 w 的祖先而非父结点，与 vw 是 TE 矛盾。

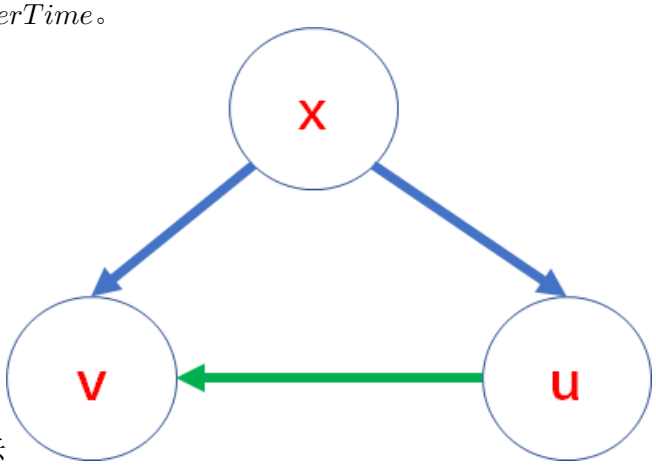
⇐ 同理可得 w 是 v 的后继，且 v 直接指向 w，则 w 是白色，即 vw 是 TE。

④ vw 是 BE⇔v 是 w 的后继 ⇔ $active(v) \subset active(w)$ 得证。

problem 4.5

1. 不可能是 TE。如果是 TE，则有 $active(v) \subset actice(u)$，即 $v.finishTime > u.discoverTime$，故不成立。
2. 不可能是 BE。如果是 BE，则有 $active(u) \subset actice(v)$，即 $v.finishTime > u.discoverTime$，故不成立。
3. 不可能是 DE。如果是 DE，同样的 v 是 u 的后继结点，满足 $active(v) \subset actice(u)$，同 1. 不成立。
4. 可能是 CE。x 结点先遍历 v，然后从 v 返回 x，x 遍历 u，易知 $v.finishTime < u.discoverTime$。

如图所示



problem 4.7

在第一次 DFS 中，将结点压栈，同一强连通片的源头结点是最后一个压栈的。**(引理 4.4)** 若 l 是某个强连通片首节点，x 是另一个强连通片中的节点，并且存在 l 通向 x 的路径，则 x 比 l 先结束遍历，即 x 先进栈 l 后进栈。满足这些性质，才能保证第二次 DFS 中，按正确的顺序取出每个 SCC 的首节点。

无论是 DFS 还是 BFS 在一次遍历中都可以访问一个或者多个强连通片的所有节点。

如果第一次 DFS 换为 BFS，由于 BFS 中按层序遍历，同一连通片中出度不为 0 的点可能先入栈。在第二次 DFS 的时候，不能有正确的访问顺序。

如果第二次 DFS 换为 BFS，由于出栈的访问首节点顺序正确，BFS 同样可以正确地划分强连通片。

因此第一次必须为 DFS，第二次 DFS 和 BFS 都可以。

problem 4.8

充要条件: 对于无向连通图的 DFS 生成树的根节点 v，v 是割点，当且仅当 v 有两个及两个以上的子树。

证明: $\Rightarrow$ 如果 v 是割点，假设 v 只有一个子树，易知子树是连通的，将 v 删除，剩下的部分为 v 的子树仍然连通，这与 v 是割点相矛盾。因此 v 有两个及两个以上的子树。

$\Leftarrow$ 如果 v 有两个及两个以上的子树，因为图本身连通，则子树之间相连必然通过 v，即 v 是割点。

problem 4.9

正确

证明: 当从 TE vw 回退时，如果以 w 为根的子树存在 BE 指向 v 的祖先，则 v 的祖先的 discoverTime 会被赋值给以 w 为根的子树中某个节点的 back 值，并最终会传递到 w.back，即必有 $w.back < v.discoverTime$；

否则，没有存在 BE 指向 v 的祖先，如果子树不存在 BE，则子树所有点 back 值均为初始值，即 $w.back > v.discoverTime$，如果子树中存在 BE，则子树中的所有结点的 back 值也将大于等于 $v.discoverTime$，因为 BE 只能指向 v 或者子树内部结点，故 back 值最小也不会低于 $v.discoverTime$，仍然满足 $w.back > v.discoverTime$，即仍能正确判断是否为割点。

Chapter 5