Tarjan

進階教學組

August 3, 2022

目錄

1 DFS tree

1.1 概念

這並不是真正的一個 tree，DFS tree 是一個在有向或無向圖中，把有成功進行到 DFS 的邊當成樹邊，然後變成一棵樹 + 其他邊的形式，所以依據選的根結點的不同、DFS 順序不同，DFS tree 也會跟著改變。

在實作上，我們也不會特別的把他求出來，只是剛好在 DFS 的過程中他會出現而已，而且他的特性會比較方便之後的講解

```
func DFS_tree(node current) {
    if ( current.isvisited )
        return;

    //a -> b (edge)
    //from a to b
    //edge.from = a, edge.to = b
    for edge in edge_from_current
        if( not edge.to.isvisited )
            edge.to.state <- DFS_tree_node
            DFS_tree( edge.to )
        else
            edge.to.state <- other_node
}
```

上文的虛擬碼是找出 DFS tree 的邊，因為實作不複雜而且概念也十分基本就不放完整程式碼。

可以發現，圖上其實會有些邊沒有被選到，於是為了方便分類，人們幫這些沒有被選到的邊分了類並取了名字

1. 樹邊 (Tree Edge): 在 DFS tree 上的邊

2. 回邊 (Back Edge): 從**子節點**連到**父節點**的邊

3. 前向邊 (Forward Edge): 從**父節點**連到**子結點的子樹結點**的邊

4. 交錯邊(Cross Edge) : 連到兩個非祖孫關係的邊

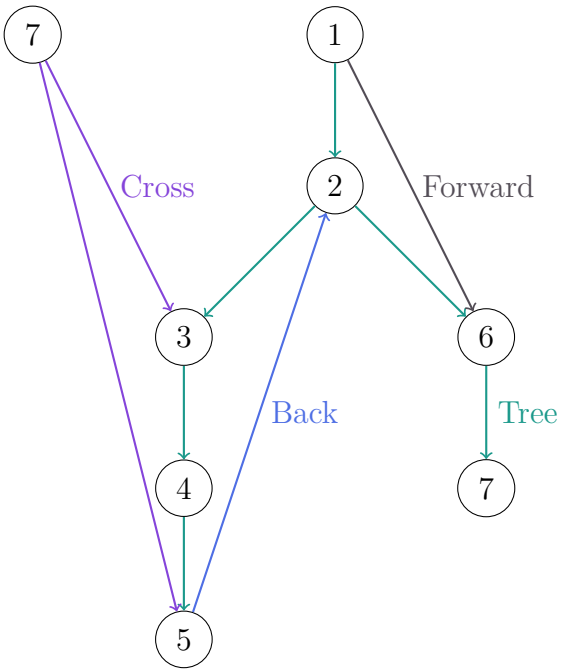
換個說法，你可以把 DFS tree 想像成一個族譜圖，然後把邊想像成認識的關係圖 (有向邊就是你認識他，但他不認識你，可能過年的時候很常出現:poop:)，如果以自身來比喻的話有點像這樣

1. 樹邊: 你認識你的孩子

2. 回邊: 你認識輩分比你大的直系血親

3. 前向邊: 和回邊互補，識輩分比你大的直系血親認識你

4. 交錯邊: 你認識你的旁系血親



(圖片原形 by 建中)

2 連通

定義 ：把邊全部轉成無向邊之後，若結點集合任意點 A 到任意點 B 都存在一條路徑，則我們即稱圖為連通的

如果以上面的比喻來說的話，就是如果你們家族都是社交大師，不會出現只有一邊認識的情況，這時候對於任一個親戚可以獲得的情報網就是連通分量。

有向圖其實還有分強連通跟弱連通，強連通是指如果 A 可以到 B，則 B 也可以到 A，兩個人都要彼此到達對方才算連通。弱連通則是即使 A 可以到 B，B 也不一定到 A，只要有一個人可以到達對方就算連通。

3 連通分量

3.1 邊雙連通分量/橋

如果移除一個邊可以使連通的塊增加，我們稱這個邊為橋，在把所有的橋都移去之後，我們就可以得到邊雙連通分量，為什麼他要叫邊雙連通呢? 因為如果想要再增加連通塊的話至少需要移除兩個邊。

3.2 點雙連通分量/割點

大致上和邊雙連通分量差不多，只是由刪除邊變成刪除點而已，而那個點稱為割點，如果把所有的割點都去掉之後我們就可以地到點雙連通分量。

4 Tarjan

4.1 無向圖

Tarjan 的無向圖比較簡單理解一些，因為無向圖有如下幾種性質

1. 沒有 Cross Edge
2. Forward Edge 和 Back Edge 被和到了一起，他們會同時出現

如果知道了以上的性質，我們要如何找出所有的橋呢? 首先，我們可以發現，Forward Edge 對答案並不會有影響，因為我們還是可以從 Tree Edge 走過去，所以現在我們只需要去考慮 Tree Edge 跟 Back Edge。先不要一次想太多，那我們不妨從只有 Tree Edge 開始，之後再一個個的把 Back Edge 加回去。

對於 Tree Edge 而言，可以發現他每一個節點都是橋，因為只要刪掉任何一個邊都會多出一個連通塊，此時，你會發現其實對於無向圖來說，最終有大小大於 2 的連通塊 = 有環。對於一個 Tree 來說，只要增加 Back Edge 就會出現**環**，來觀察一下**環**的性質，你可以發現，**環**會是 Back Edge 中間的那一條練。

那要如何去維護呢，我們可以對每個點 u 紀錄一個 low 值，即他「從 u 或 u 的子孫最多走一次回邊 (可以不走)，最高可以到達的點」，接下來講結論: 「對於一條 Tree Edge $e = (a, b)$ 而言，如果 $low(a) = a$ 則代表 e 是橋」。因為把 a 移除掉之後，其子孫無法連到上面的樹，於是就多了一個連通塊。

實作時主要是用 DFS+ **進入戳記**。在實作時需要知道幾個觀念

1. DFS 會先把所有子孫處理完才會繼續處理當前節點
2. DFS 中，在沒有遇到分岔的情況下，遍歷的順序總是一條條練。
3. 只有互為直系親屬 才可以使用進入戳記來判斷層樹高低，不然如果先遞迴到的子樹其進入戳記會比後遞迴到的子樹小。

虛擬碼

```
Time_count = 0;
Stack node_save;
func Tarjan(node current) {
```

```
    current.time_in, current.low <- Time_count
    Time_count <- Time_count+1
    node_save.push( current)

    for edge in edge_from_current
        if( not edge.to.isvisited )
            Tarjan( edge.to )
        else
            current.low = min( current.low, edge.low)



    if( current.time_in == current.low )
        node temp;
        do {
            temp = node_save.top()
            bcc[ temp.id ] = current.id
            node_save.pop()
        }
        while(temp != current);
}
```

4.2 有向圖

有向圖大部分都和無向圖差不多，不過有向邊多了 Cross Edge，所以3的部分可能會出現錯誤，那我們要如何去解決呢?

十分的簡單，在從結點 u 走到 low 函數更低的節點 v 時，我們可以先判斷 v 在不在 Stack 內。因為如果 v 是祖先，他一定還沒被踢出 Stack，而且根據2，如果 v 在別的子樹且還未被踢出 Stack 外的話，代表 $[v, LCA(u,v))$ 以下的節點都已經被拜訪完了，也就是 u,v 會有共同的連通塊