DFS numbering

→ maintain a global counter

→ when entering a node, record what counter is and increment counter ("pre")

→ similarly when leaving a node once and for all, record what counter is ("post")

and increment counter

Code

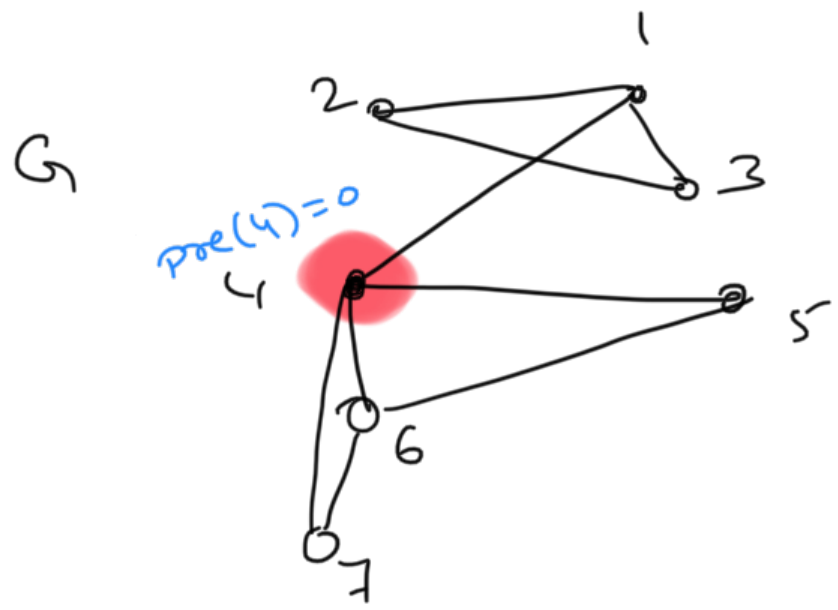
```
global counter = 0

def DFS (start):
    visited [start] = True
    pre [start] = counter
    counter = counter + 1
    for neighbour w of start :
        if visited [w] = False
            parent [w] = start
            DFS (w)
    post [start] = counter
    counter = counter + 1
```

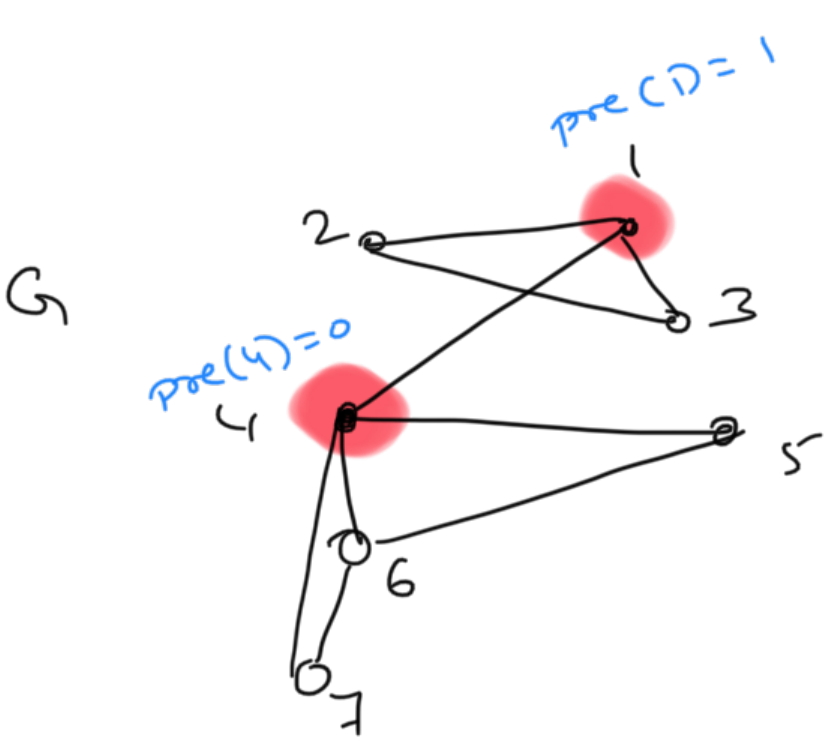
Dry run

start = 4
counter = 0

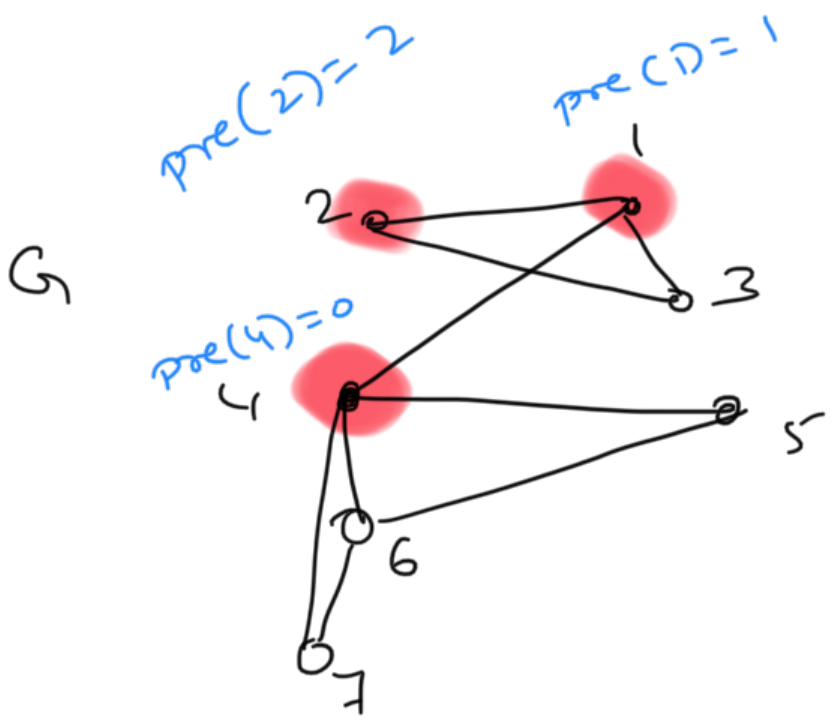
↓



pre(4) = 0
counter = 1



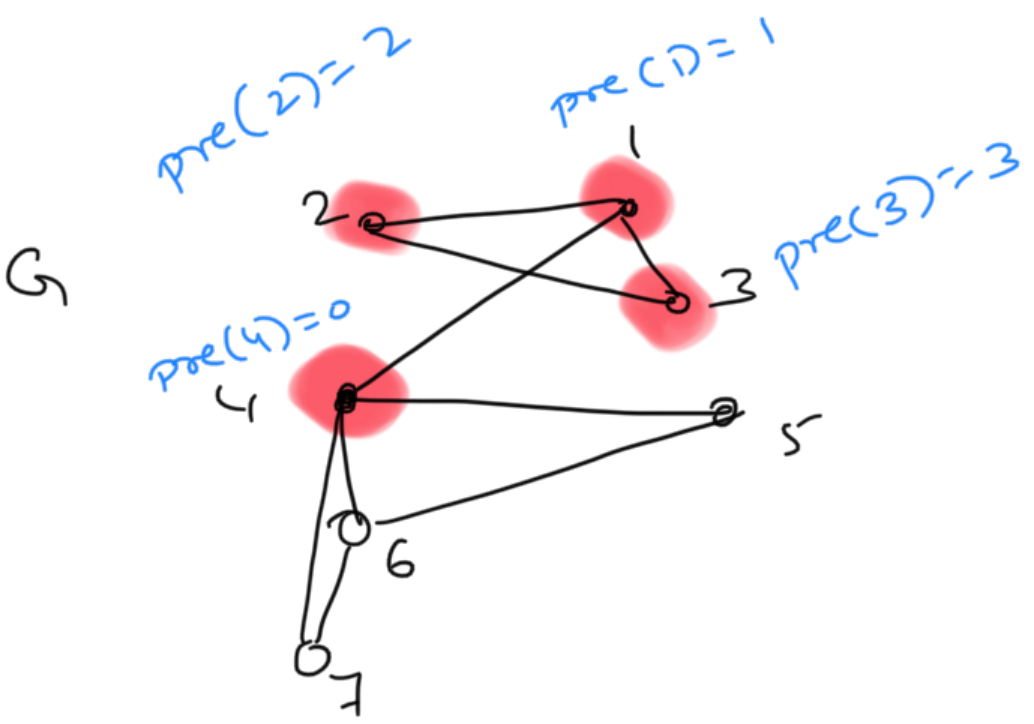
pre(1) = 1
counter = 2



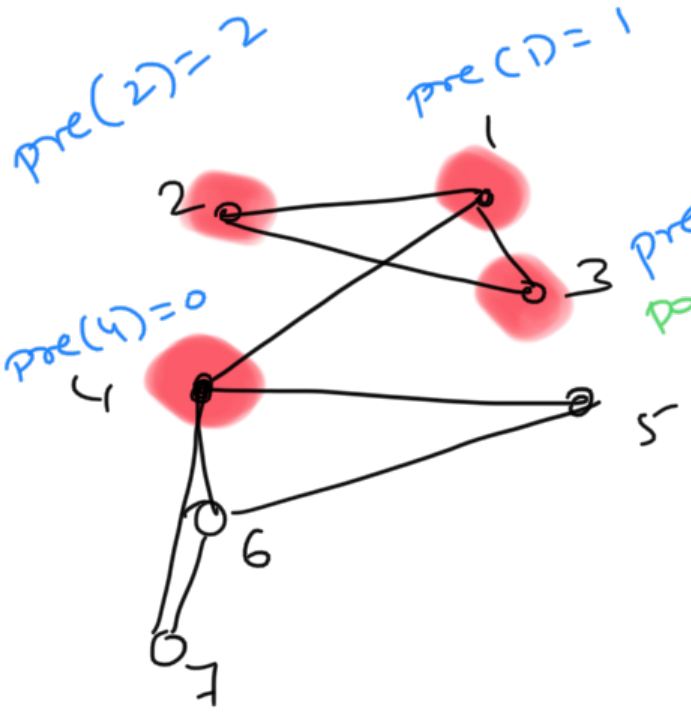
pre(2) = 2
counter = 3

pre(3) = 3
counter = 4

3 dead end



G



3 dead end

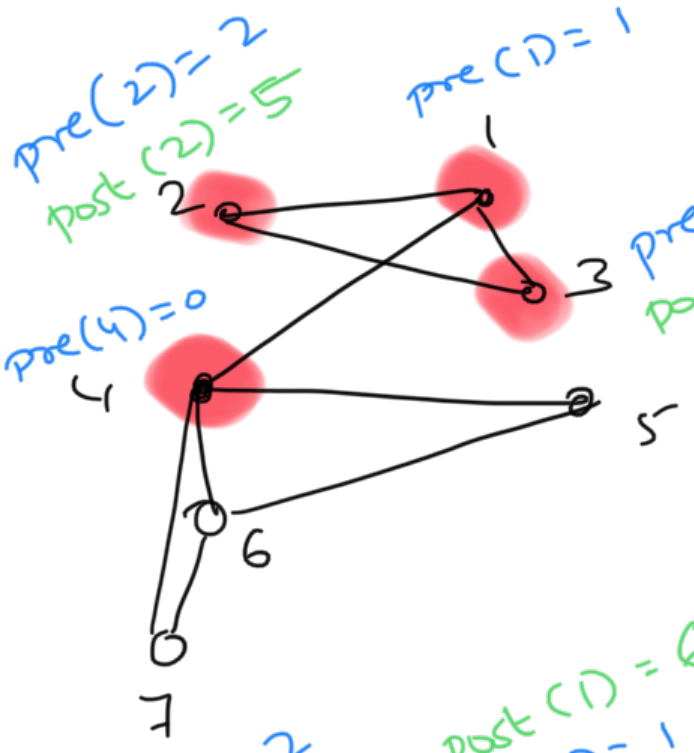
leave 3

post(3) = 4

counter = 5

2 dead end

G



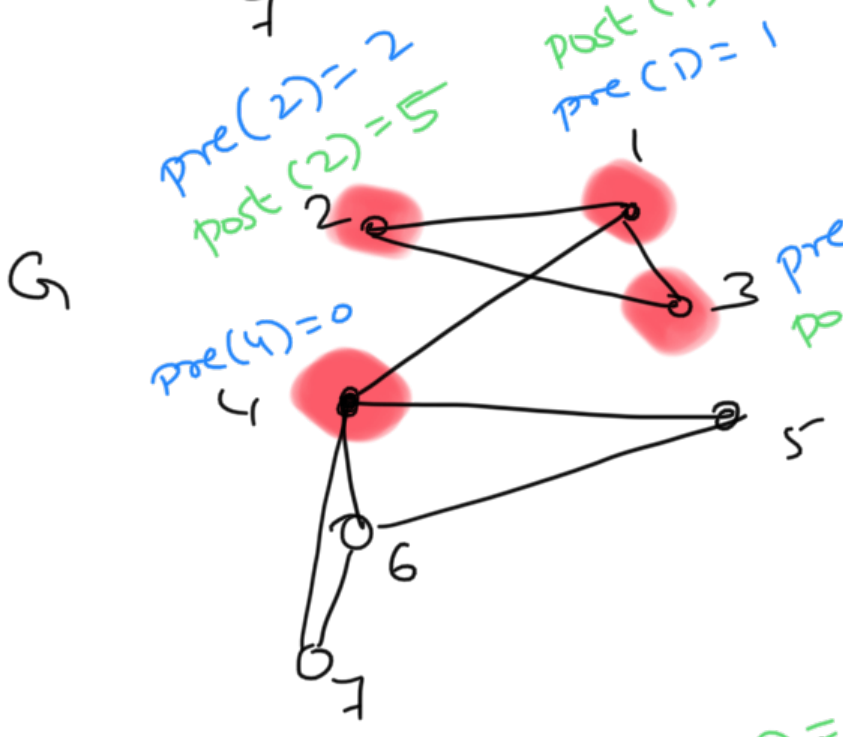
leave (2)

post(2) = 5

counter = 6

1 dead end

G



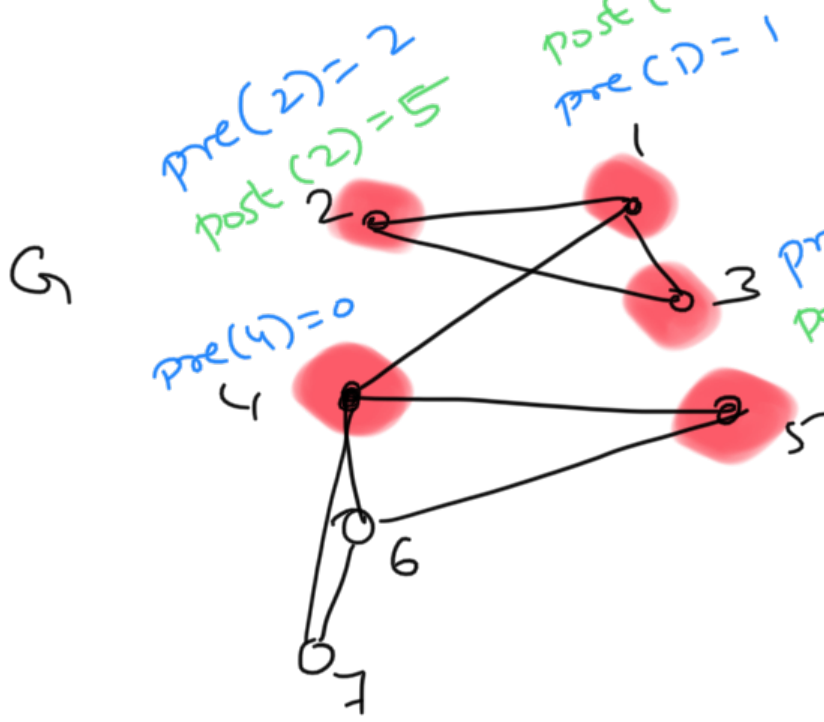
leave (1)

post(1) = 6

counter = 7

continue exploring 4

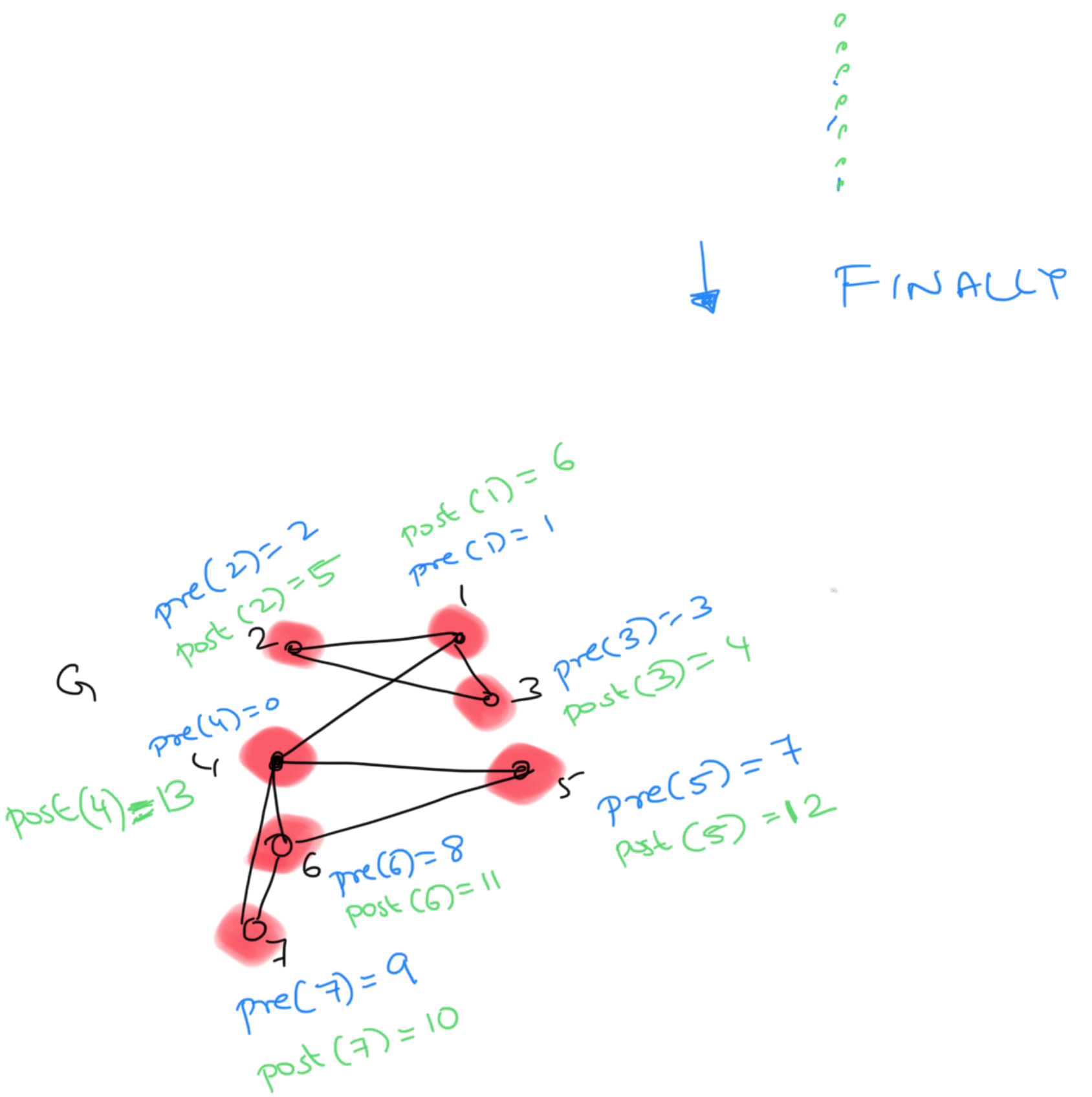
G



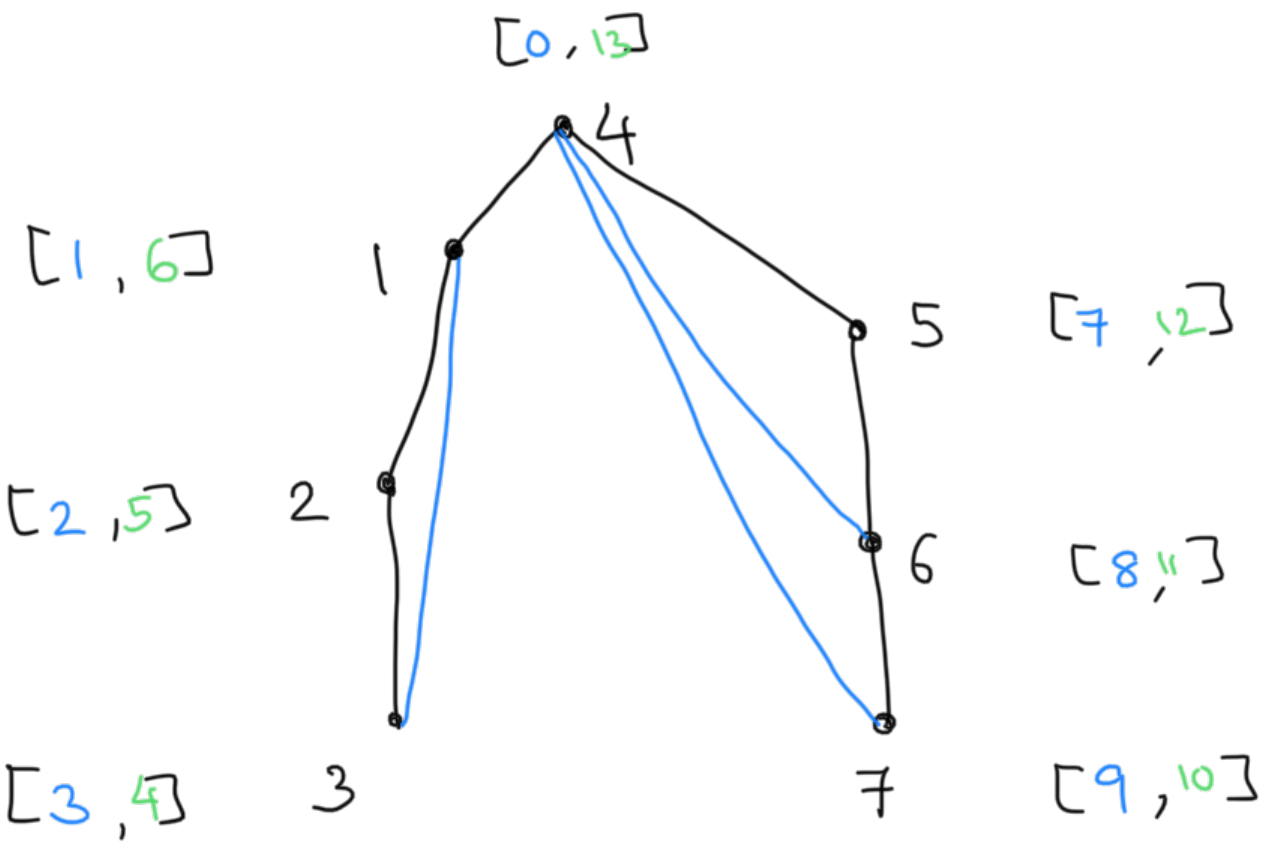
pre(5) = 7

pre(5) = 7

counter = 8



Redrawing graph



↝ exploring vertex i bet [pre[i], post[i]]
steps

↝ Black edges " tree edges"

In DFS algo, while exploring edge (v, w)
↑ visited ↑ unvisited

then v — w is a tree edge

(c) parent(w) — w is a tree edge

Non tree edges

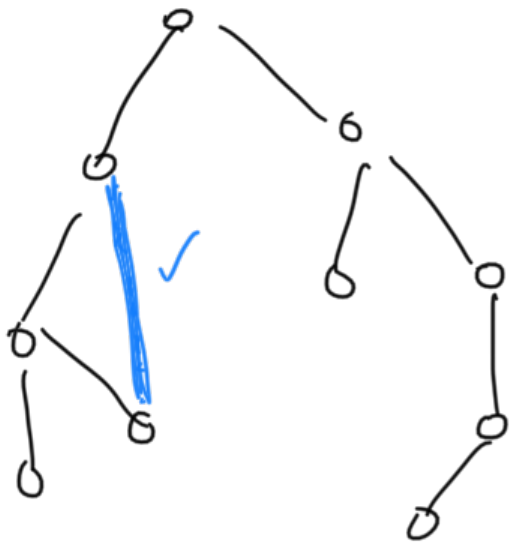
↝ edges not explored....

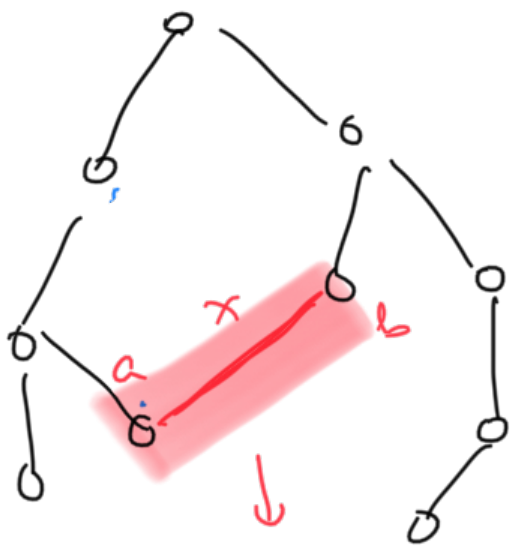
(v , w) v — w
↑ visited ↑ visited

↝ ∃ cycles in graph iff non tree edges

in DFS / BFS

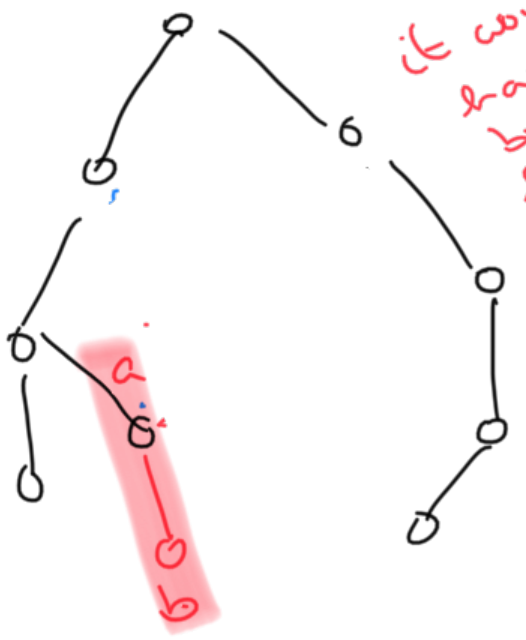
↝ non tree edges can only go back up a path.





Can't cross branches.

left to right, earlier vertices to later ones ...



it could have been like this

(because if a → b edge, you would have explored b from a, and