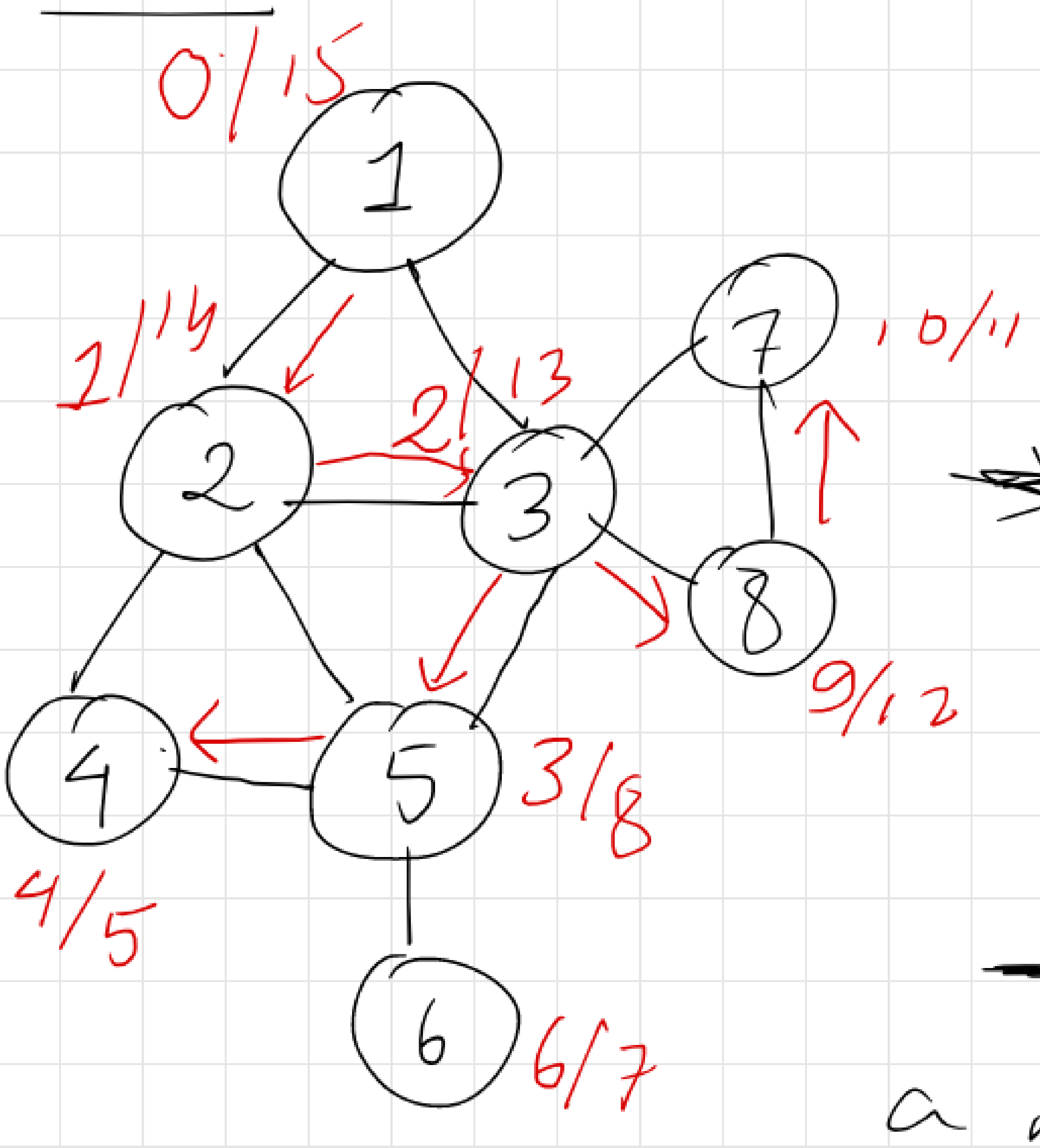
"Depth First Search"

Idea:

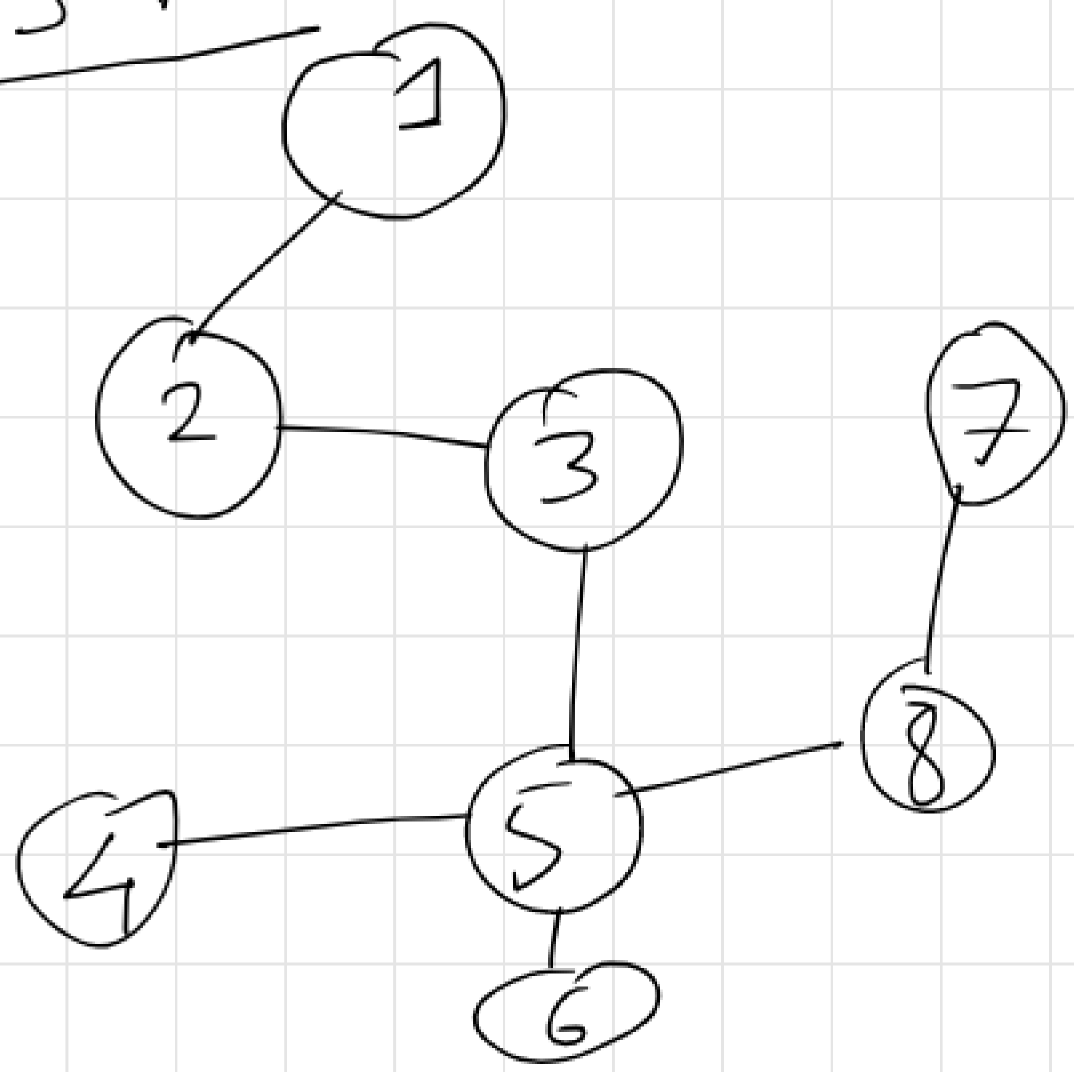


Timestamps
discovery
processing done

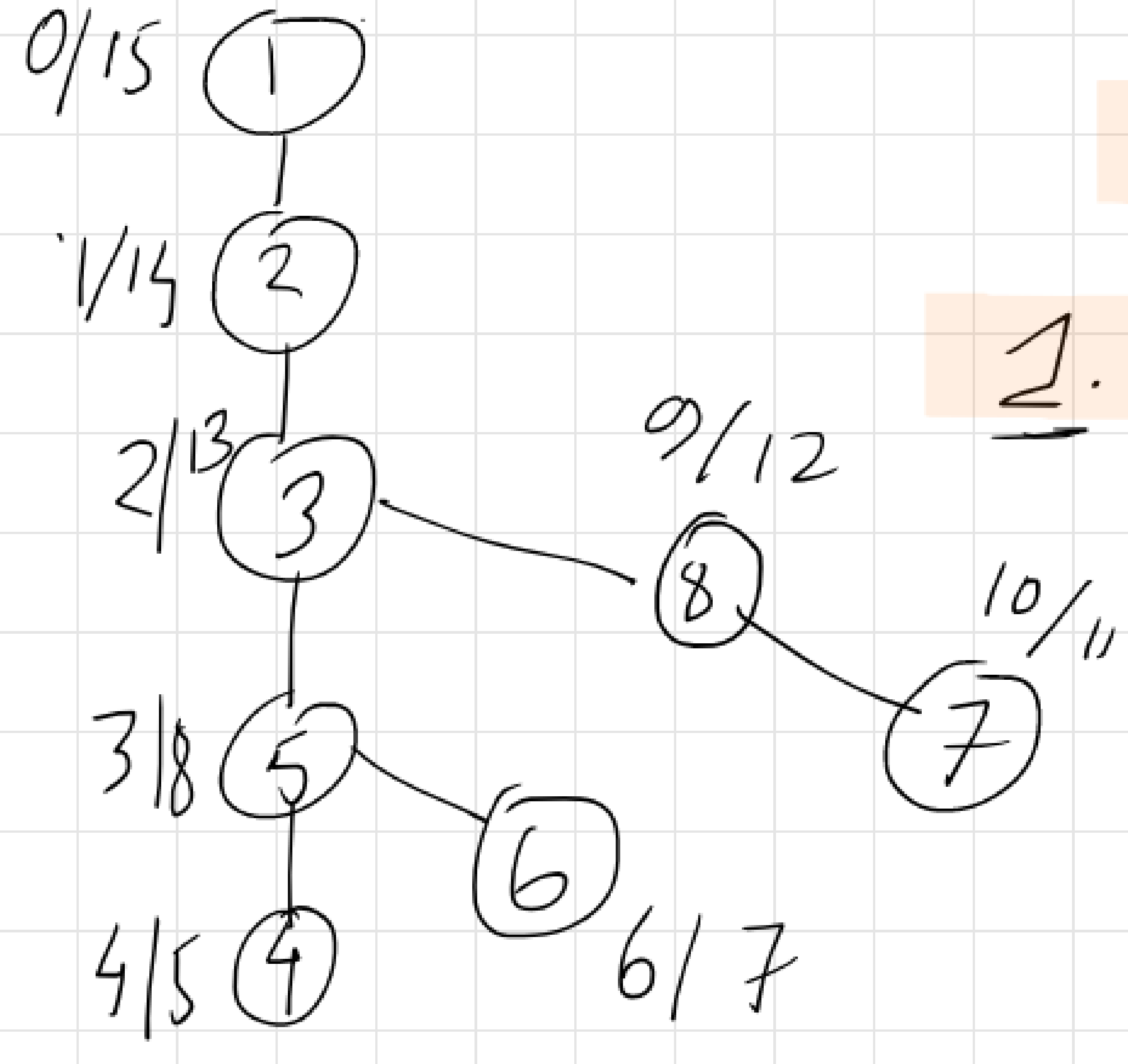
→ explore a node only if it is unexplored

→ backtrack from a node along the same edge that has taken to reach the node.

DFS Tree



n-1 edges
Connected
acyclic
} Tree



Observaton

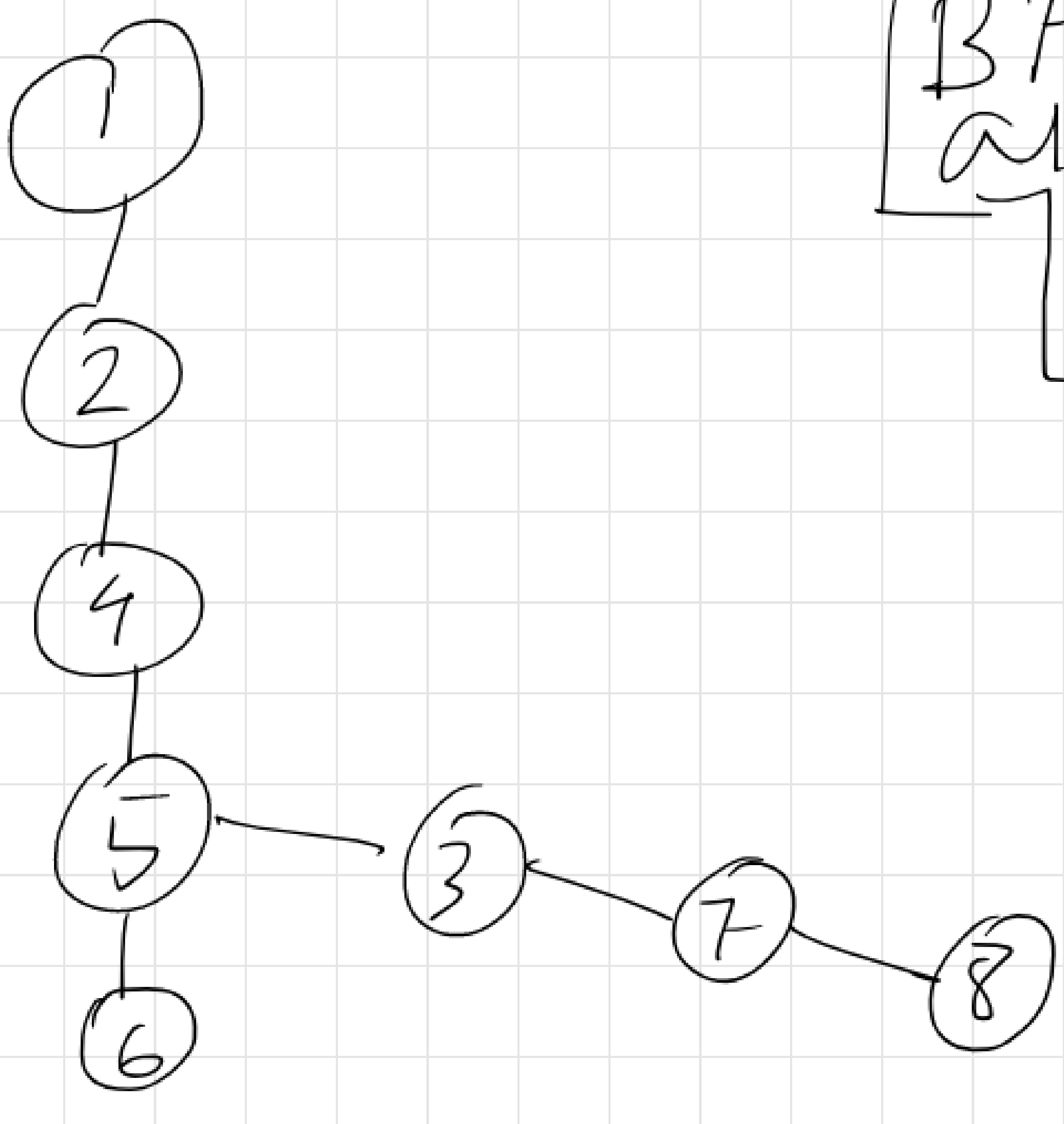
1. discovery (ancestor) < discovery (descendant) < finish (descendant) < finish (ancestor)

Eg:

$$\begin{cases} d(2) < d(6) < f(6) < f(2) \\ d(3) < d(5) < f(5) < f(3) \end{cases}$$

Is DFS Tree unique ?? → NO

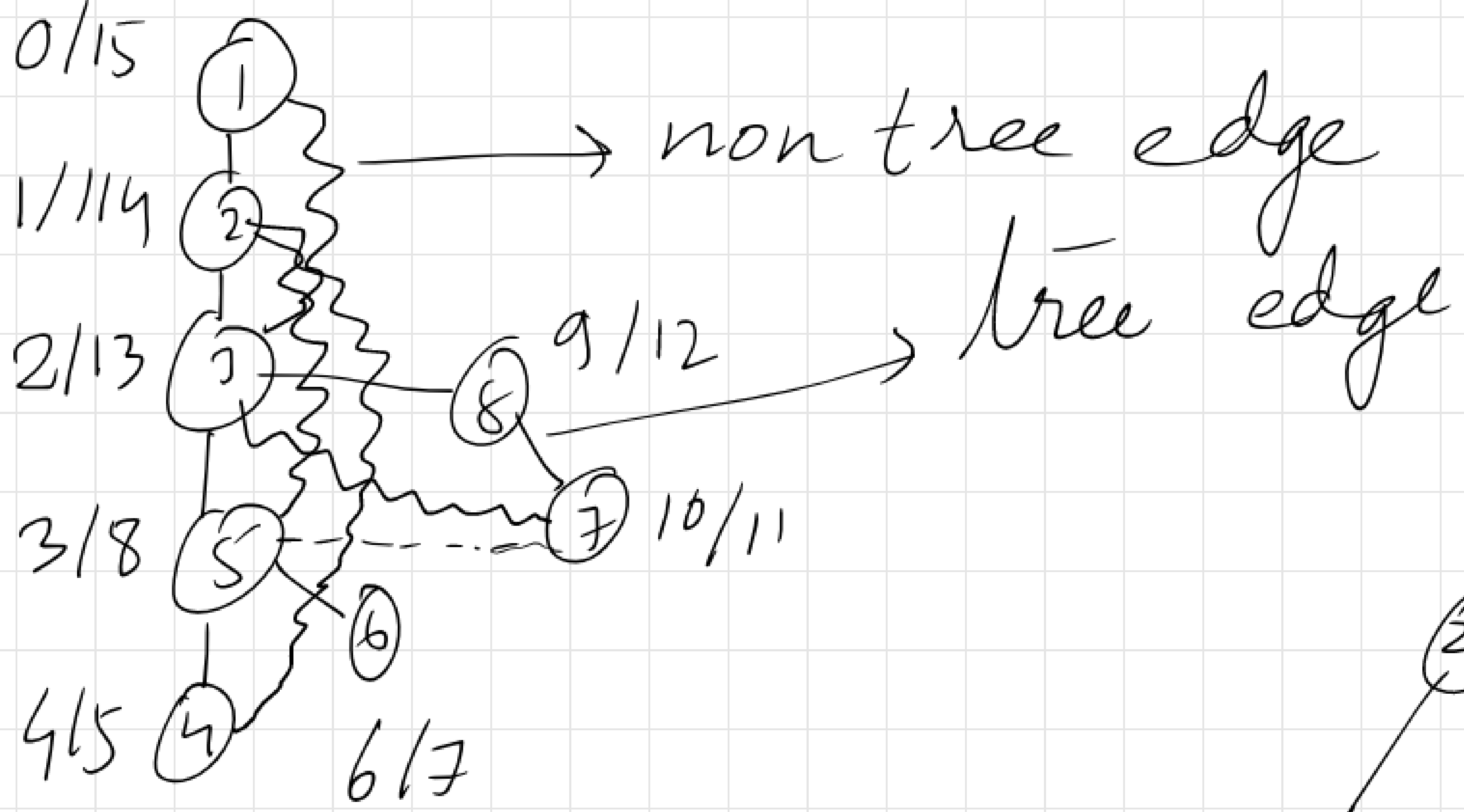
BFS tree is also not unique

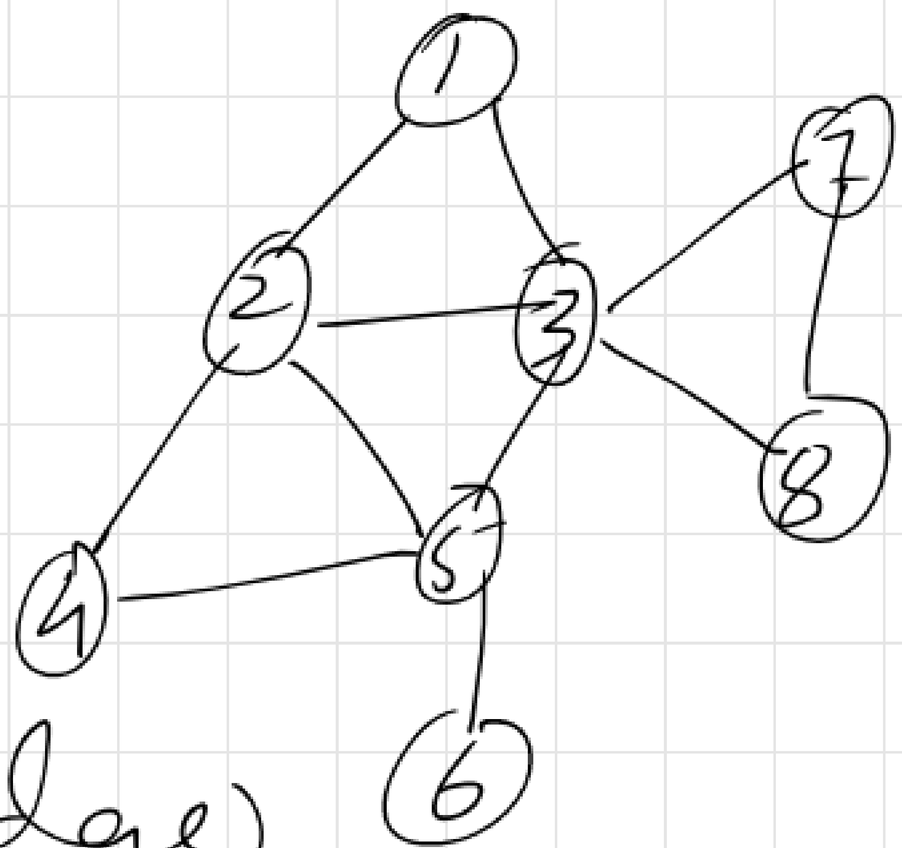


DFS trees (narrow and deep)

BFS tree (short and broad)

↳ $(x, y) \in E \quad x \in L_i \quad y \in L_j \quad |i-j| \leq 1$





〰 (non tree edge) part of G

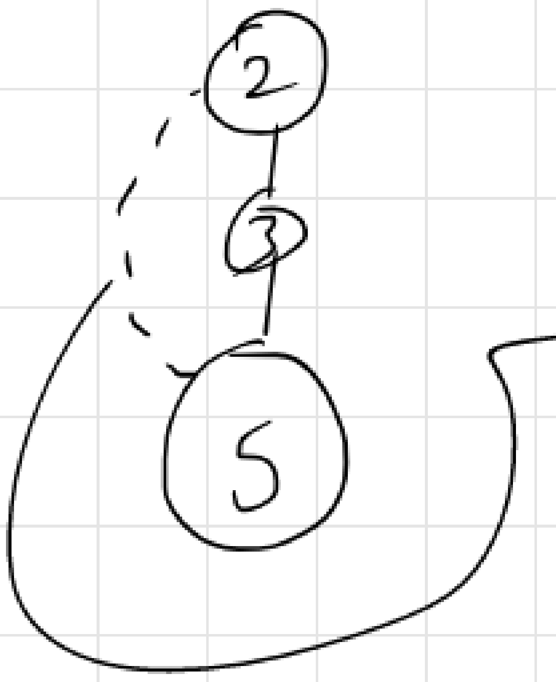
[Can we have $(5,7) \in E$ in G? Is it possible]

$f(5) = 8$

$s(7) = 10$

$(6,7) \in E$ not possible

Non-tree edges connect only ancestors to descendant non tree edges are called **back edges**



→ you realised this edge when you were exploring node 5 not (2)

[thus the name back edge]

Claim:

Let T be a DFS tree. Let x and y be nodes in T. Let $(x,y) \in E$.
Then one of x and y is an ancestor of the other.

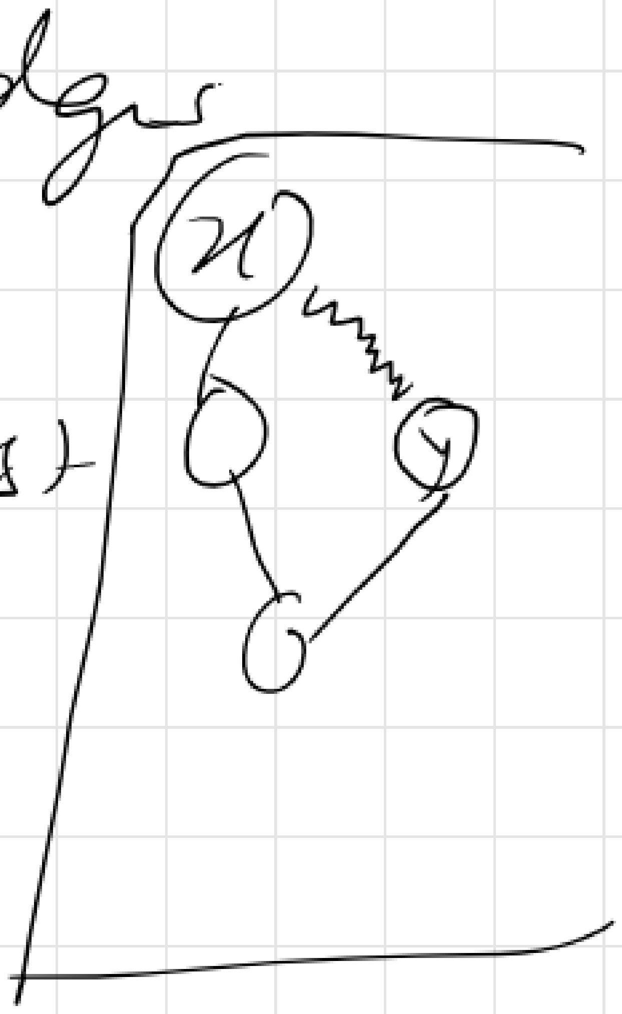
Pf:

$(x, y) \longrightarrow$ tree edge
(Claim) ✓

let $(x,y) \longrightarrow$ non tree edge
$(x, y) \notin T$

W.L.O.G algo explored x first

$d(x) < d(y) < f(x)$



Implementing DFS

Recursive

given G as adjacency list

DFS(u)

set explored[u] = true

add u to R.

For each edge (u, v) // Check in Adj List(u)

if explored[v] = false

DFS(v)

endif

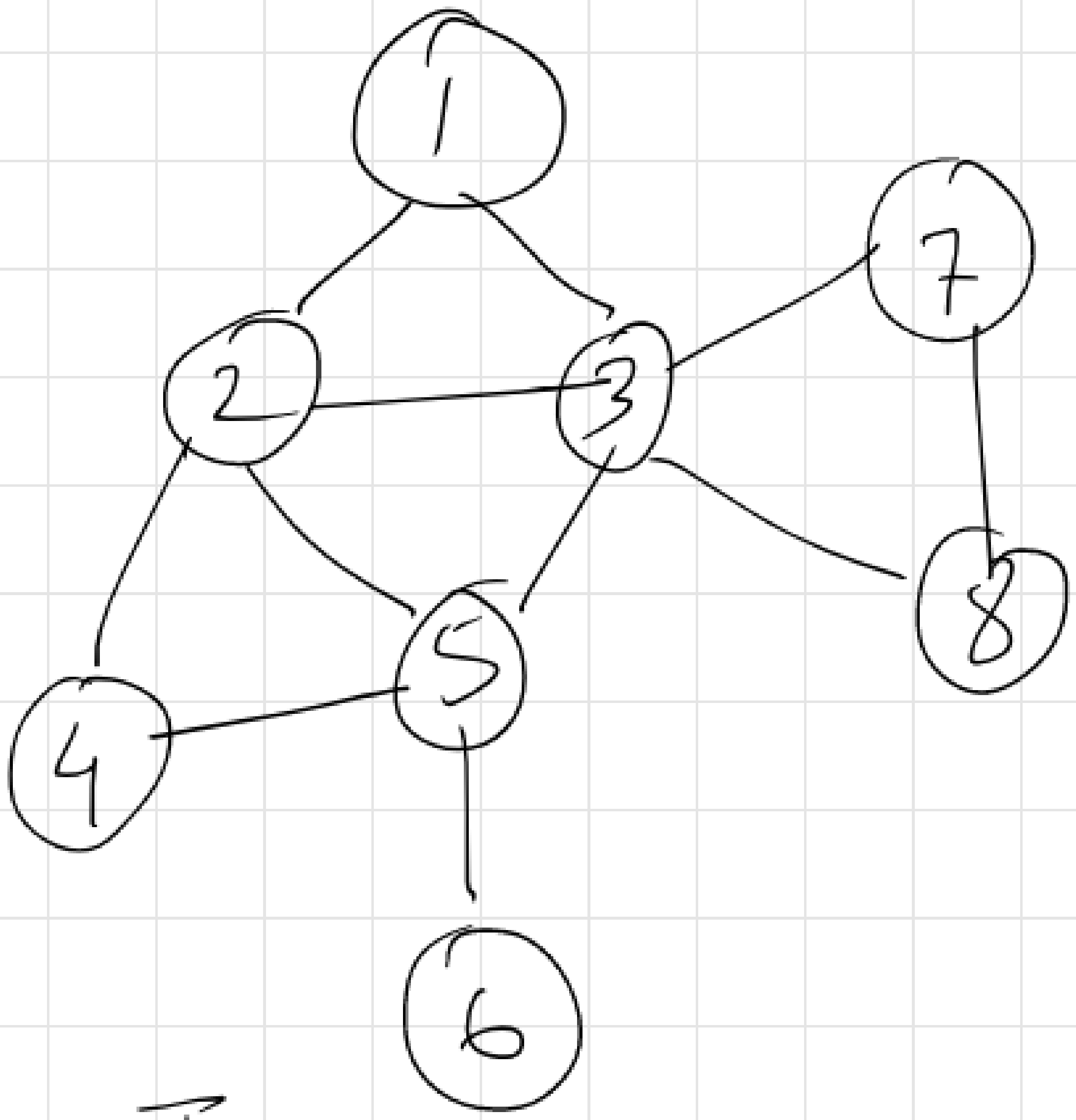
end for

DFS(1)

DFS(2)

DFS(3)

⋮



T

F	~~F~~	F	F	F	F	F	F

Non Recursive

DFS(s)

Initialized explored[v] = false $\forall v \in V$ $\rightarrow O(n)$

Initialized S to be a stack

push(s, S)

while (S ≠ 0)

  u ← pop(S) $\rightarrow O(1)$

  if explored[u] = false

    explored[u] = true

    for each edge (u, v) incident to u $\rightarrow$ in Adj list

      push(S, v) $\rightarrow O(1)$

    end for

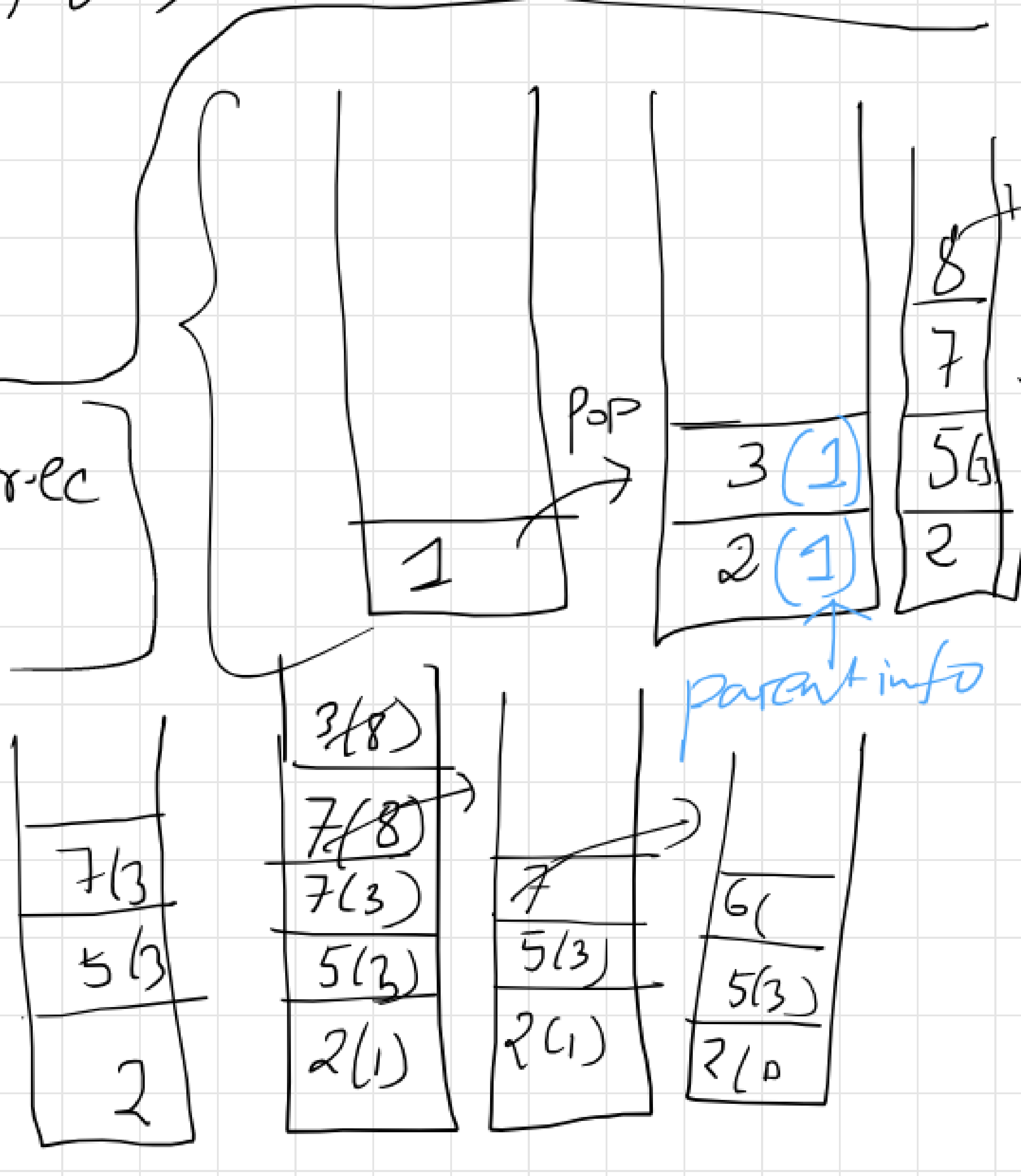
  endif

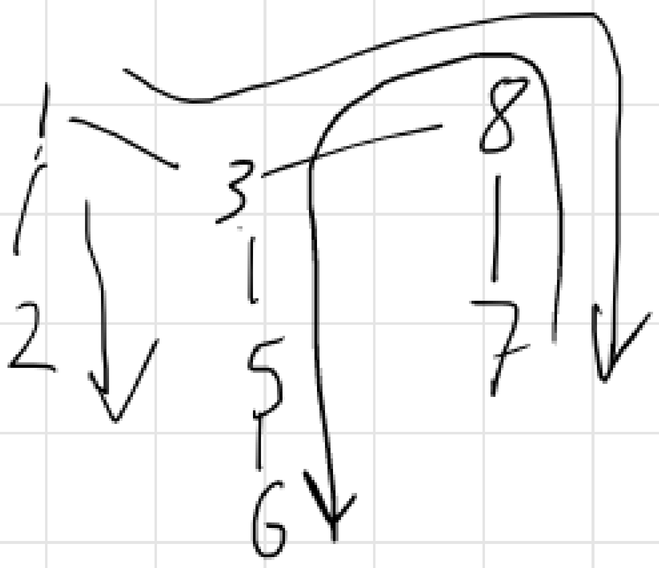
endwhile

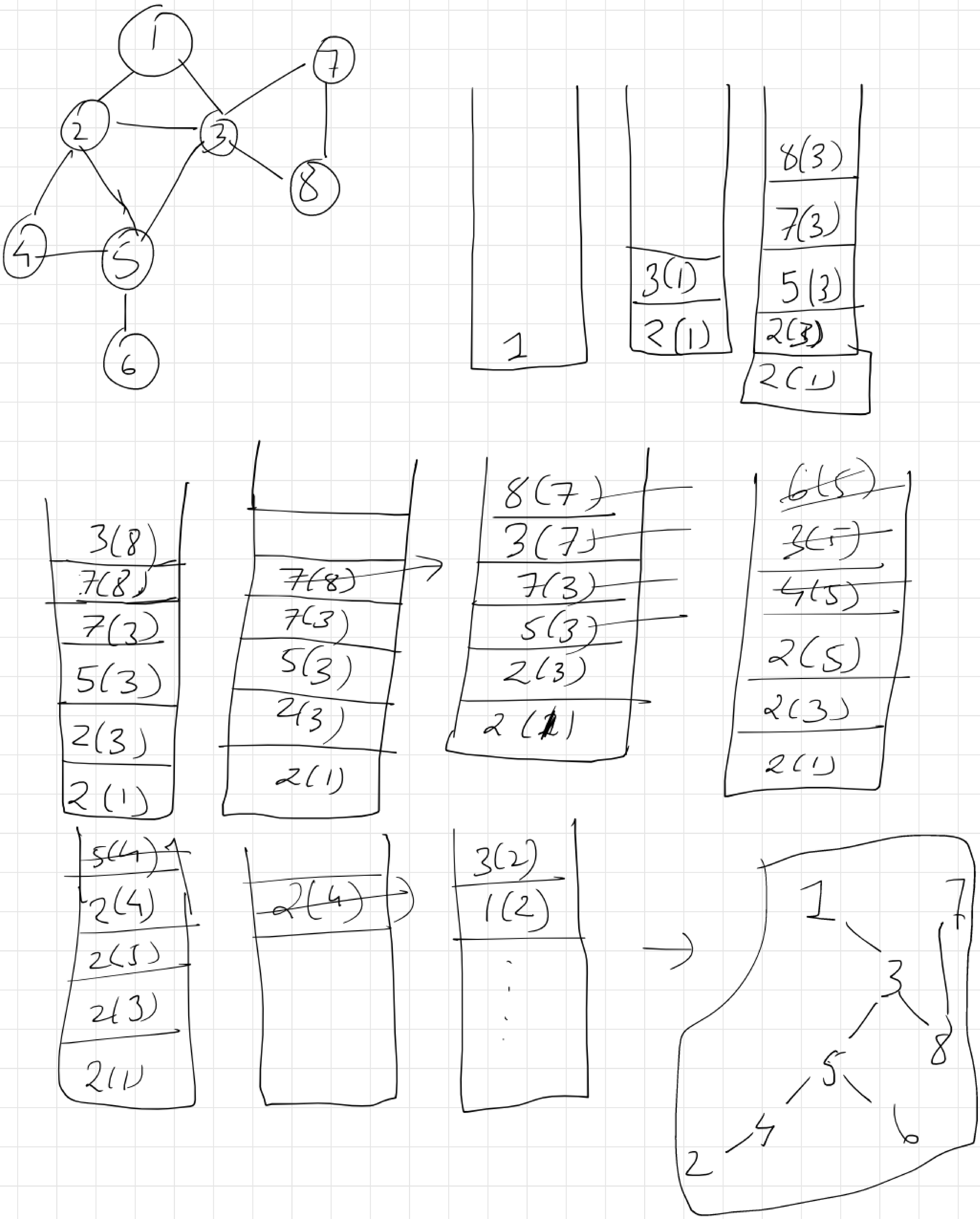
Time complexity

push $= \sum \deg(u) = O(m)$

want to create a DFS tree
push parent information in stack







1 2 3 4 5 6 7 8
1
3(1)
2(1)
8(3)
7(3)
5(3)
2(3)
2(1)
3(8)
7(8)
7(3)
5(3)
2(3)
2(1)
7(8)
7(3)
5(3)
2(3)
2(1)
8(7)
3(7)
7(3)
5(3)
2(3)
2(1)
6(5)
3(5)
4(5)
2(5)
2(3)
2(1)
5(4)
2(4)
2(5)
2(3)
2(1)
2(4)
3(2)
1(2)
1 7 3 8 5 4 6 2