- Breadth-first search (BFS)
- Queue ADT

Ring buffer : Implementation technique for queues.

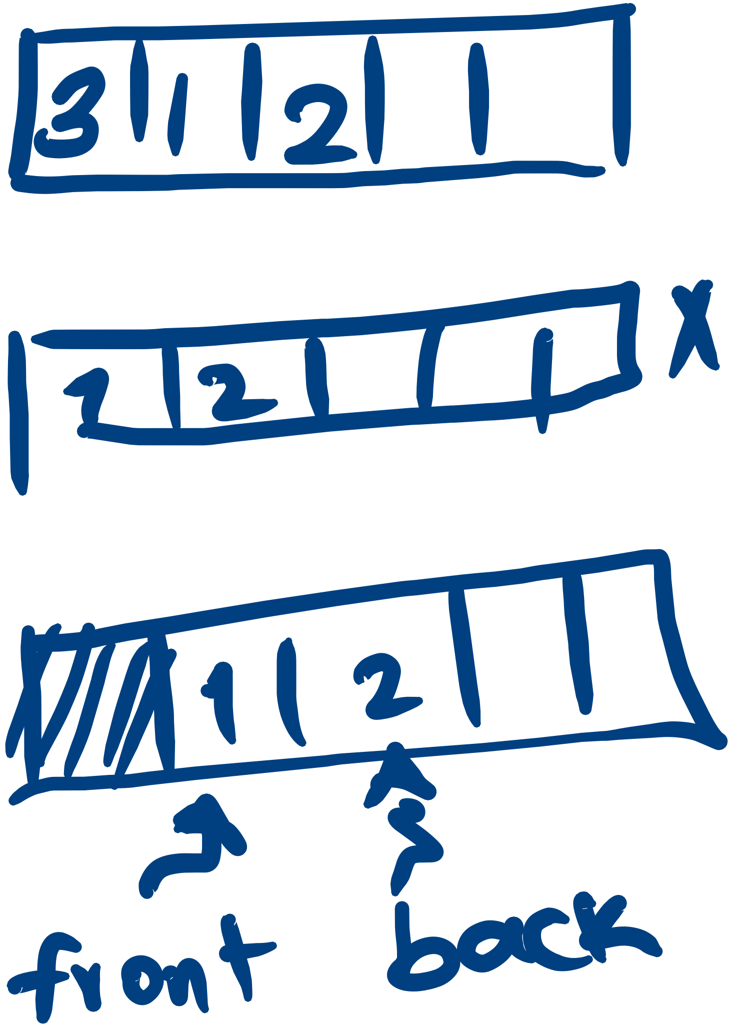
queue<T>   FIFO ~ First-in First-out

is.empty(q)

enqueue(q, e) ⇝ adds elt to queue

dequeue(q) ⇝ returns the earliest element in the queue and removes it.

Array Implementation of queues



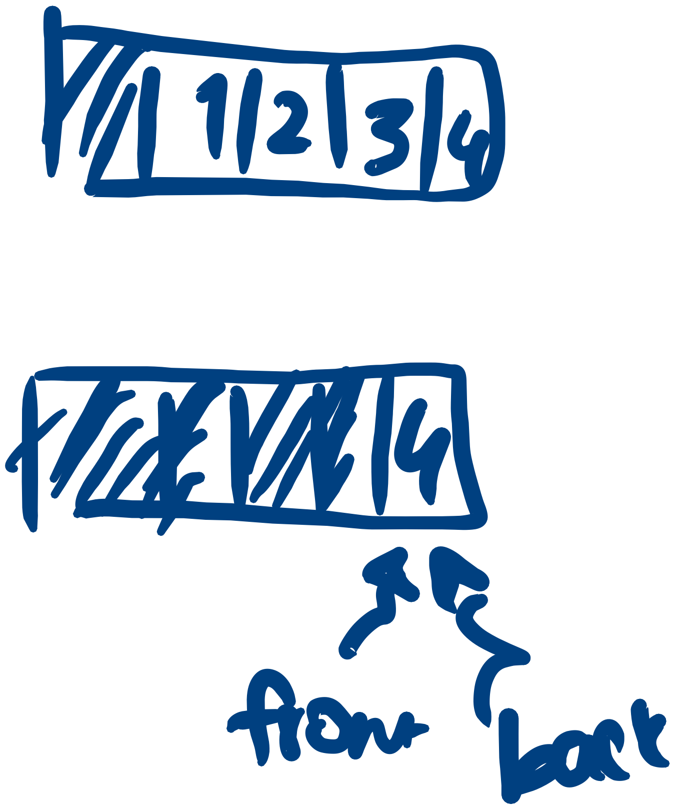
e(3)

e(1)

e(2)

d() ⤳ 3

e() & d() are now O(1) operations.



e(3)

e(4)

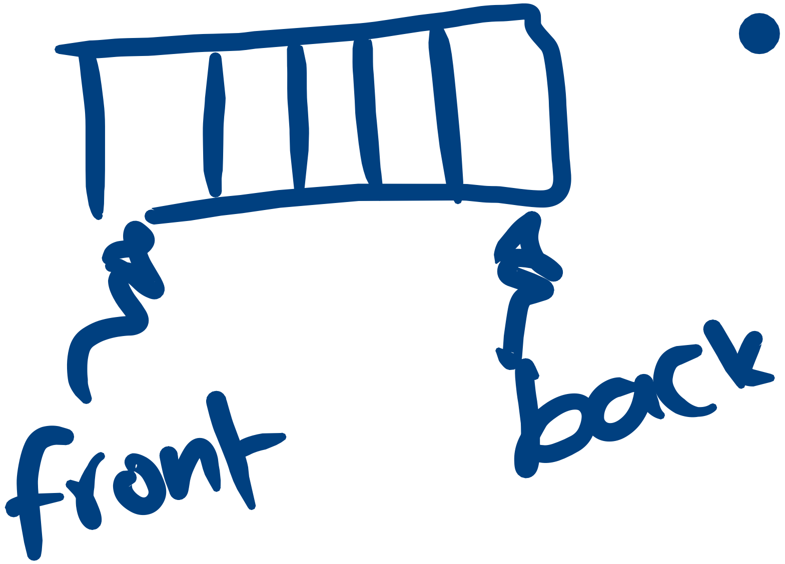
d( )

d( )

d( )

How to solve?. Imagine first elt of array follows the last.

(Ring buffer)



- len of the queue.

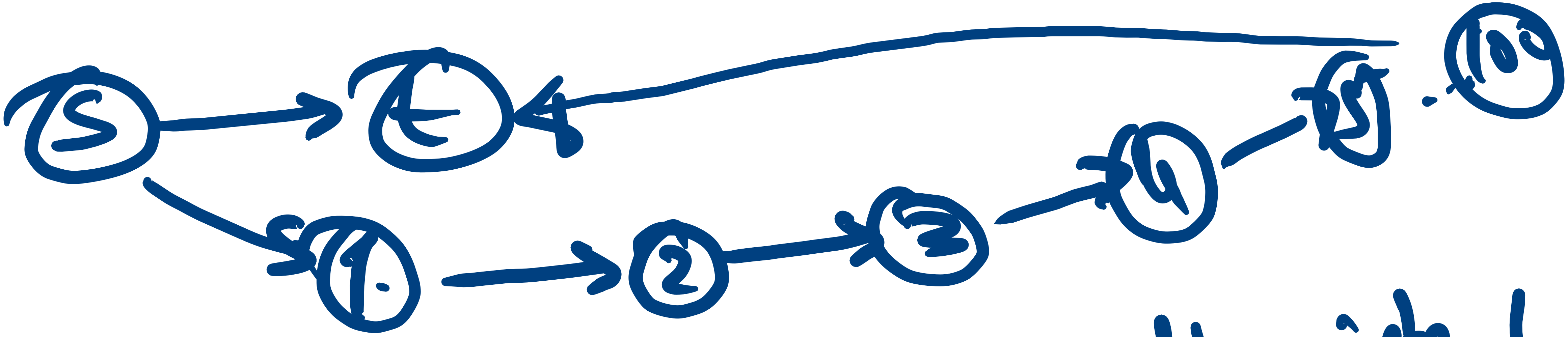
Why BFS?

8-Puzzle

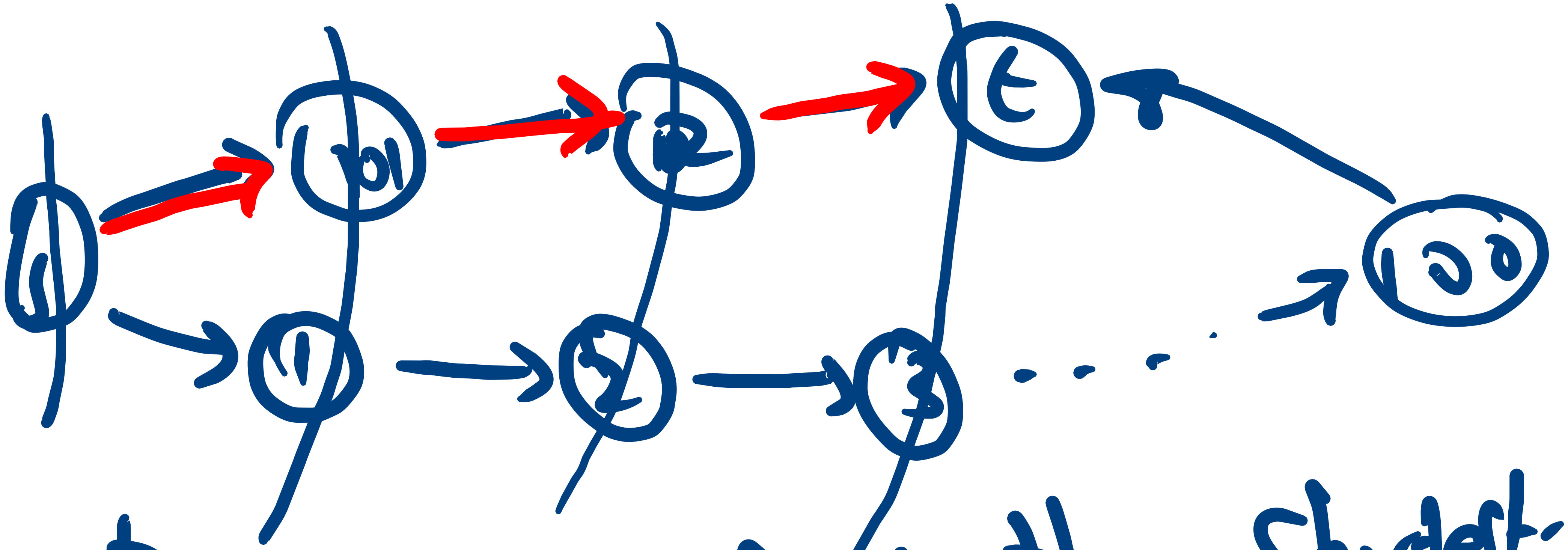
$G = ($V = Board state

E = ≤ 4 neighbors/board state$)$

Why not a DFS to find a Path to the Solution?



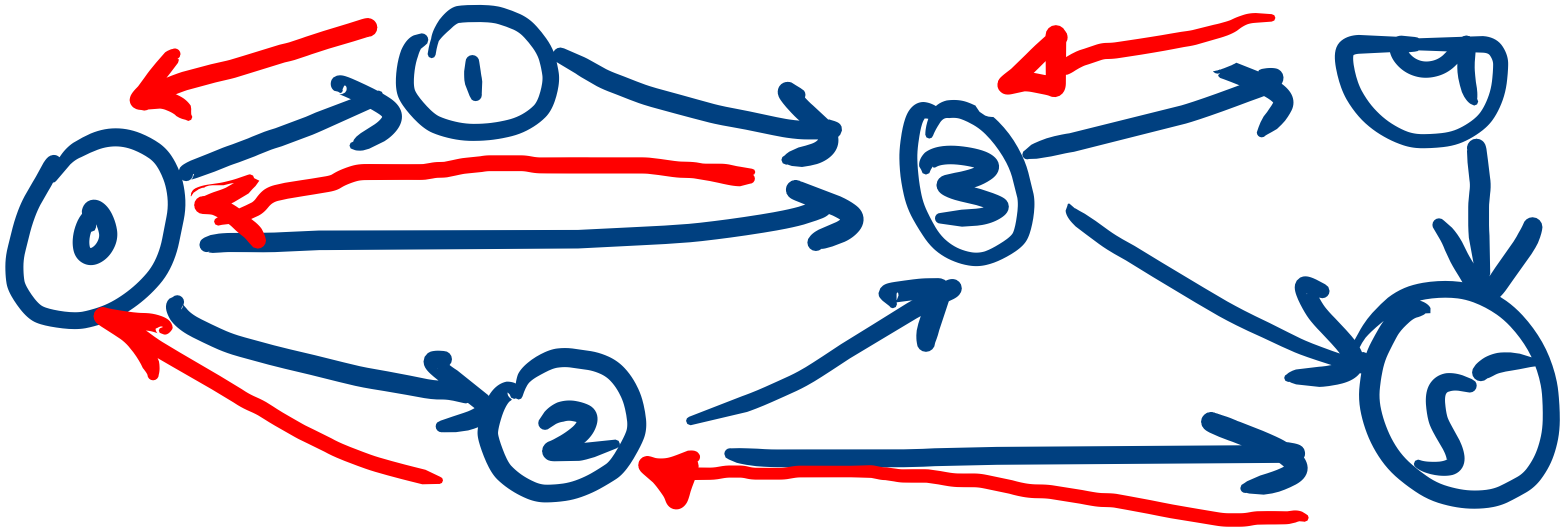
DFS may find long Paths instead of Short Paths.



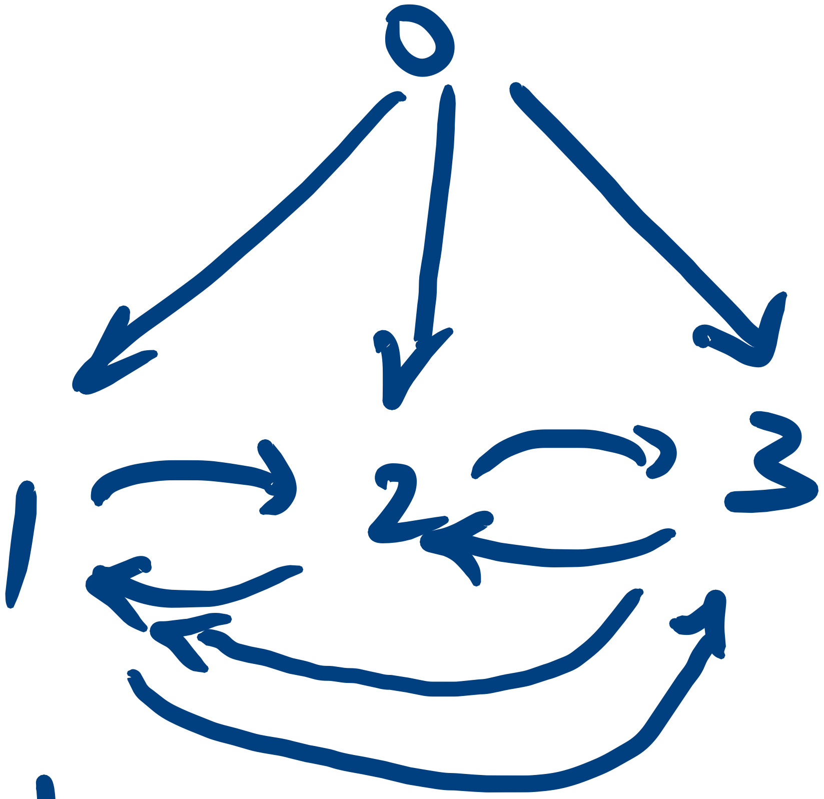
BFS would find the shortest path from s to t.

'BFS is obtained by switching stack in DFS with a queue.

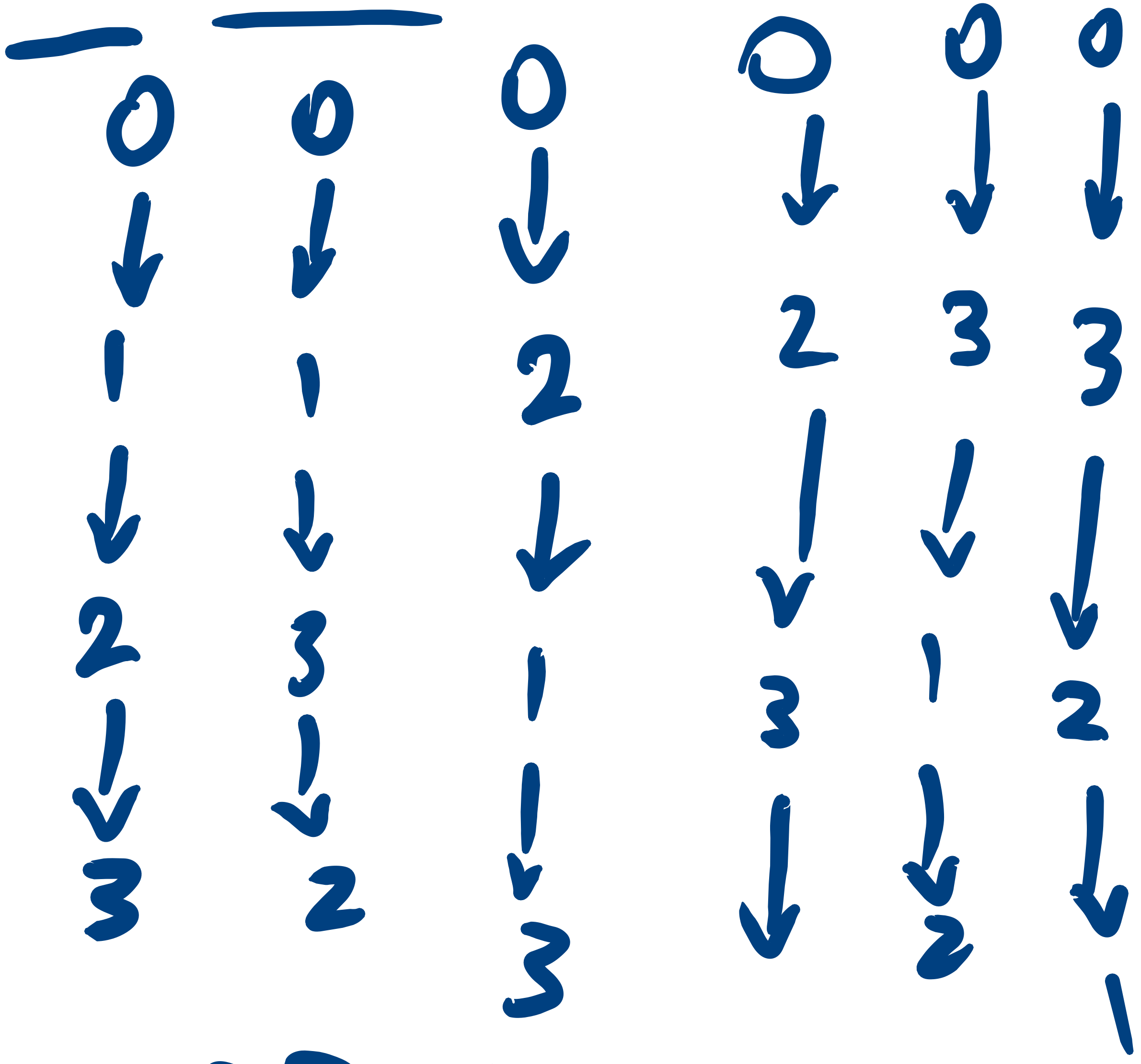
See notes on my webpage for BFS



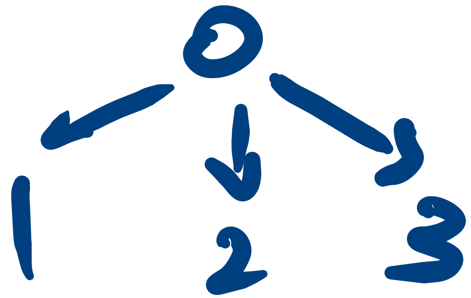
```
q ← {0}
P[v] ← v   v ∈ 0..5

1: v[0] ← true
     e(1)
     e(2)
     e(3)
   q ← 1 2 3
   v[1..3] ← true
   P[1...3] ← 0

2: u ← 1

3: u ← 2
     e(5)
   v[5] ← true
   P[5] ← 2

4: u ← 3
   v[4] ← true
   P[4] ← 3
   e(4)
```

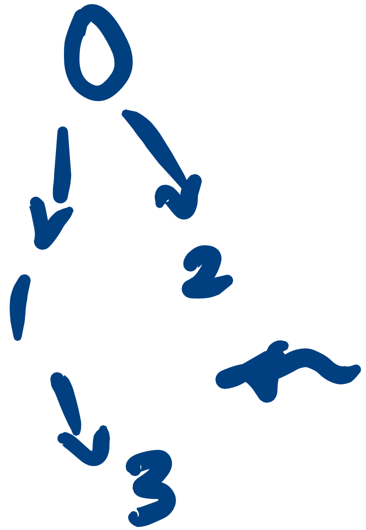


DFS trees



BFS tree





Not DFS tree,
Not BFS tree.

DFS . Explore from the latest visited vertex ⇒ stack.

BFS. Explore from the earliest (closest) visited vertex ⇒ queue.