- Breadth-first search (BFS)
- Queue ADT

  Ring buffer: Implementation technique for queues.

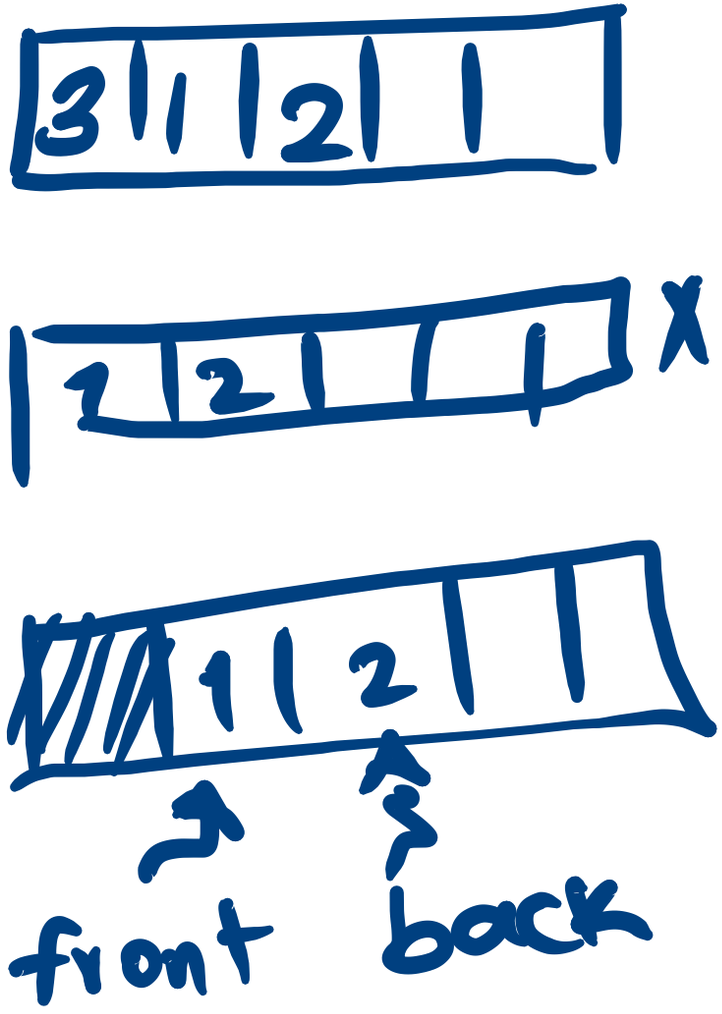
Queue<T> FIFO ~ First-in First-out

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



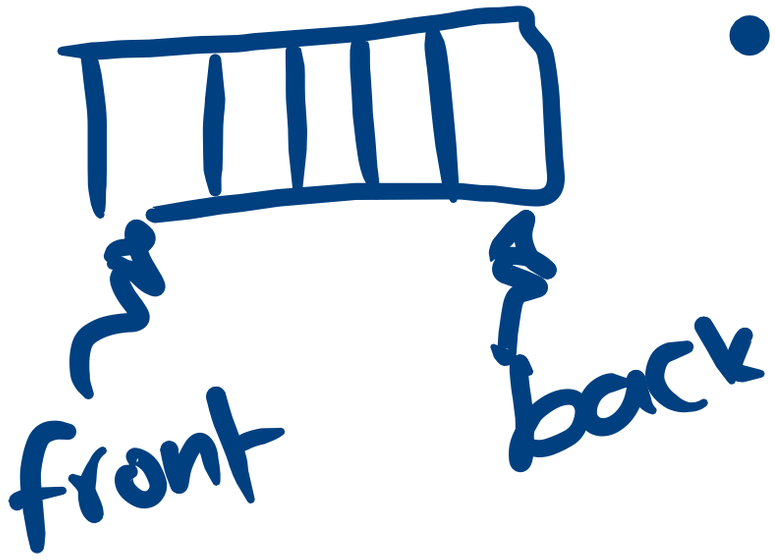
e(3)
e(1)
e(2)

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.

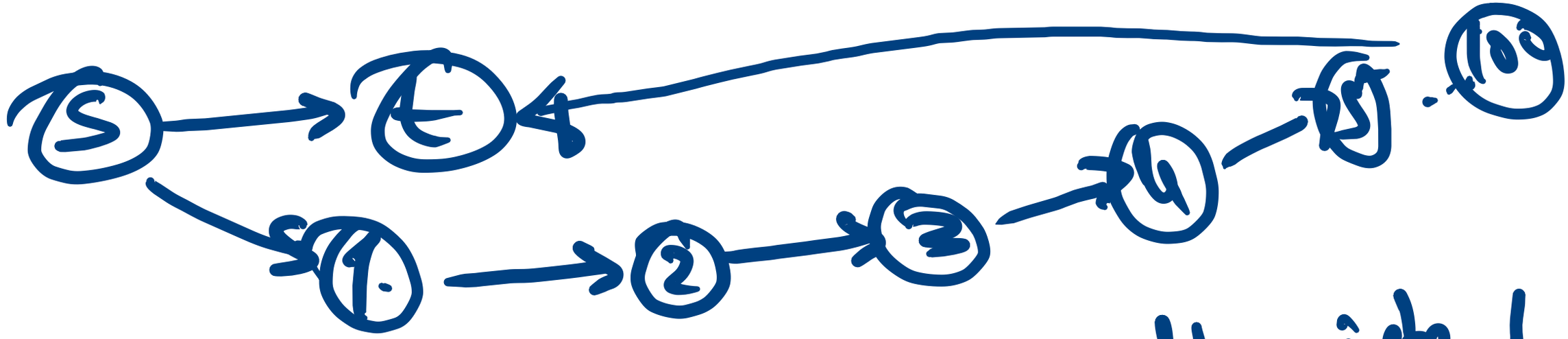
Why BFS?

8-Puzzle

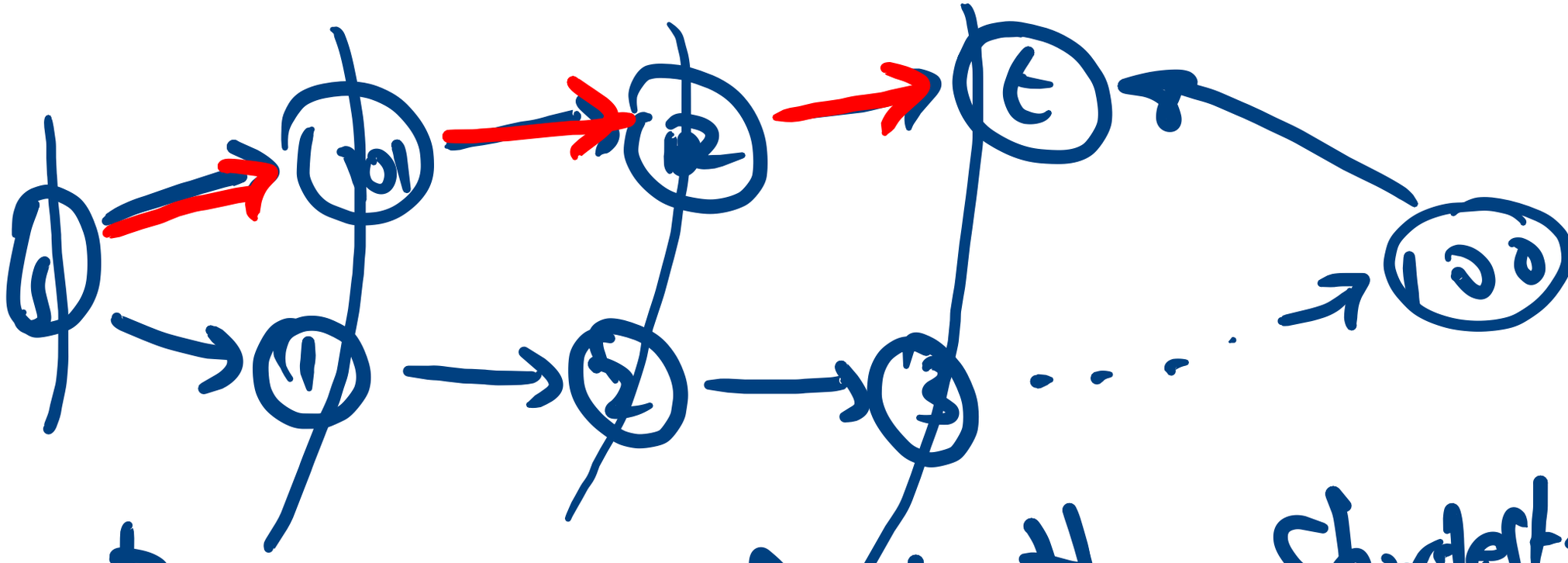
$G = (V =$ Board state

$E = \leq 4$ neighbors / boardState)

Why not a DFS to find a path to the solution?



DFS may find long paths instead of short paths.

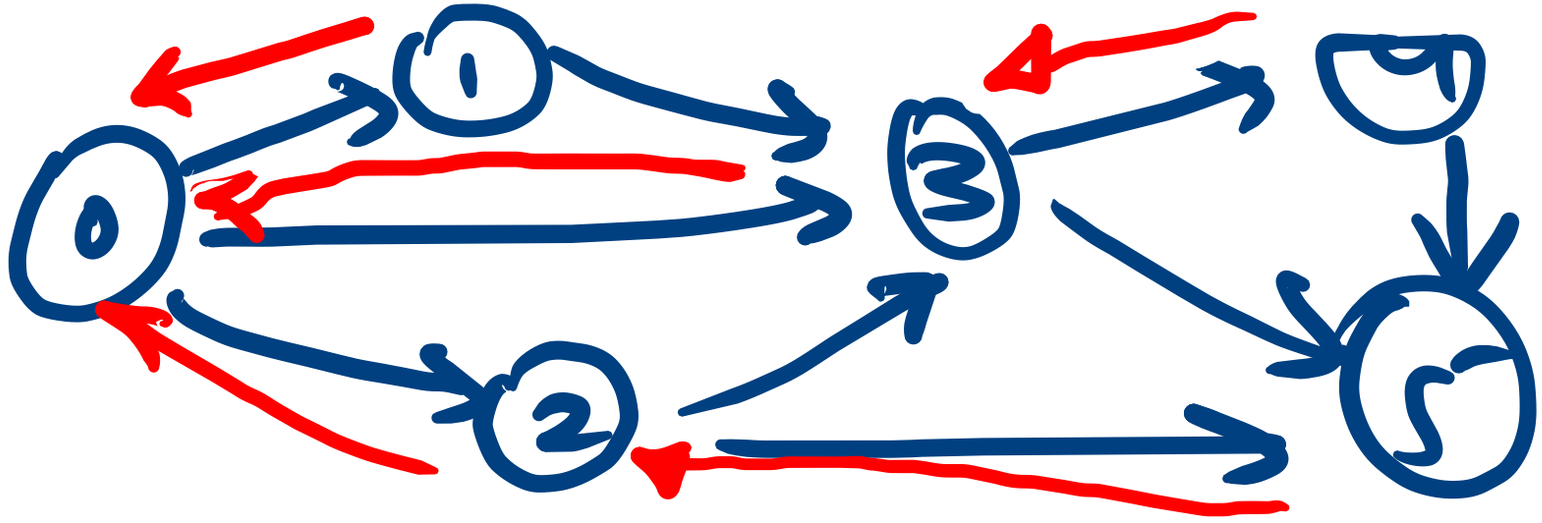


BFS would find the shortest-path from s to t.

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

```
bfs(G, source)
    queue ← {source}
    visited[v] ← false   ∀ v ∈ V(G)
    Parent[v] ← v        ∀ v ∈ V(G)
    while q is not empty
        u ← dequeue(q)
        if visited[u] is false
            visited[u] ← true
            for each u → v
```

if visited[v] = false
enqueue(q, v)
Parent[v] ← u



$q \leftarrow \{0\}$

$P[v] \leftarrow v \quad v \neq 0..5$

4: $u \leftarrow 3$

$V[4] \leftarrow$ true

$P[4] \leftarrow 3$

$e(4)$

1: $V[0] \leftarrow$ true

$e(1)$

$e(2)$

$e(3)$

$q \leftarrow 1\ 2\ 3$

$V[1..3] \leftarrow$ true

$P[1..3] \leftarrow 0$

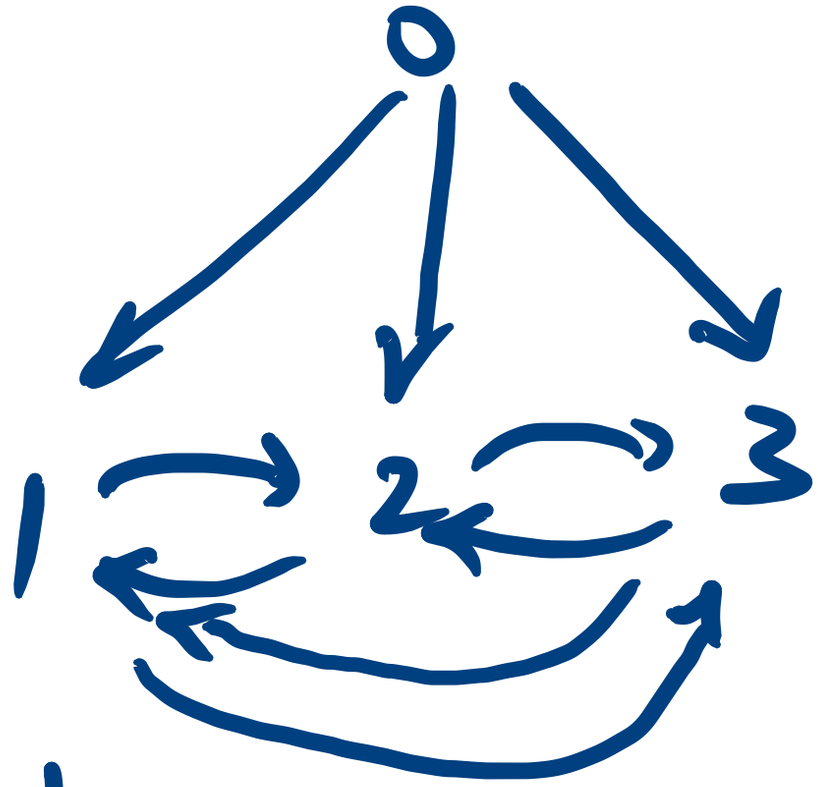
2: $u \leftarrow 1$

3: $u \leftarrow 2$

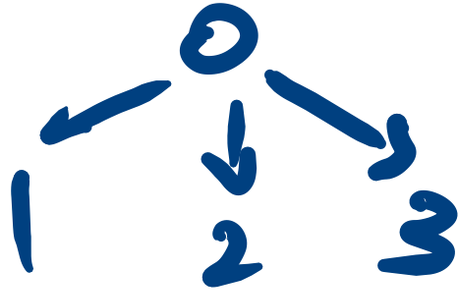
$e(5)$

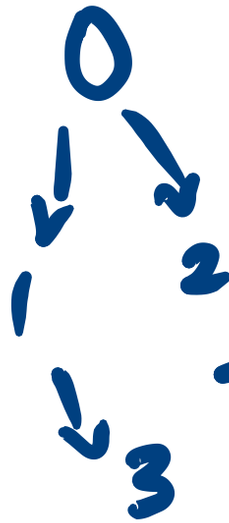
$V[5] \leftarrow$ true

$P[5] \leftarrow 2$

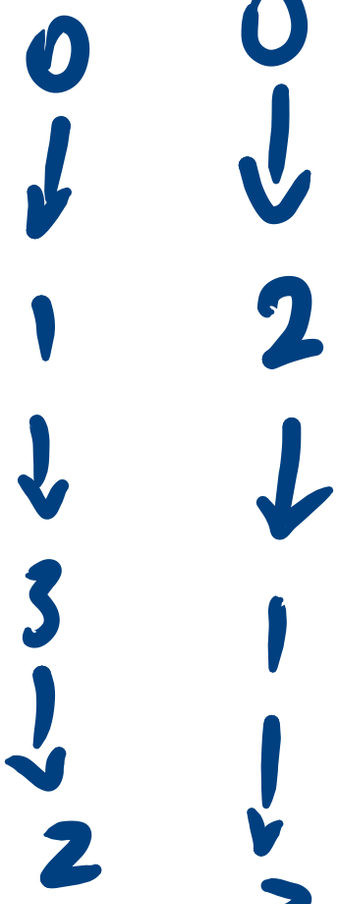


BFS tree





DFS trees



Not DFS tree, Not BFS tree.

DFS . Explore from the latest visited vertex ⇒ stack.

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