Breadth First search (BFS) (5)

→ Fundamental algorithm used to explore nodes and edges of a graph. Runs with O(V+E).

→ BFS is particularly useful for one job: finding shortest path on unweighted graphs.

→ A BFS starts at some arbitrary node on the graph and explores the neighbour nodes first, before moving to the next level neighbours.

→ implementing using queue.

→ enqueue → add to queu

→ dequeu → remove from queue.

```
# Global variable
n = number of nodes in the graph.
g = adjacency list representing unweighted graph.

# s = start node, e = end node, and 0 ≤ e, s < n

function bfs(s, e):
    # Do a bfs starting at node s.
    prev = solve(s)
    # Return reconstructed path from s → e
    return reconstructPath(s, e, prev).
```

```
function solve(s):
    q = queue data structure.
    q.enqueue(s)

    visited = [false, false .... false]   # size n
    visited[s] = true

    prev = [null, null .... null]         # size n.

    while (!q.isEmpty()):
        node = q.dequeue()
        neighbours = g.getNode(node)

        for (next : neighbours):
            if (!visited[next]):
                q.enqueue(next)
                visited[next] = true
                prev[next] = node

    return prev.
```

```
function reconstructPath(s, e, prev):

    # Reconstruct path going backwards from e.
    path = []
    for (at = e; at != null; at = prev[at])
        path.add(at)

    path.reverse()

    # If s and e are connected return the path.
    if path[0] == s:
        return path

    return [].
```