Graph (II)

Luo Tsz Fung {pepper1208}

2025-02-26

What is graph theory?

- A branch in Mathematics
- Study relationships between nodes (vertices).

Graph (I) review

- Normally, we use ***vector*** to store a graph.
- A graph can be either undirected / directed, and weighted / unweighted.
- Tree is a special type of a graph.

Graph (II)

1. Depth-First Search (DFS)
2. Breadth-First Search (BFS)

Graph (II)

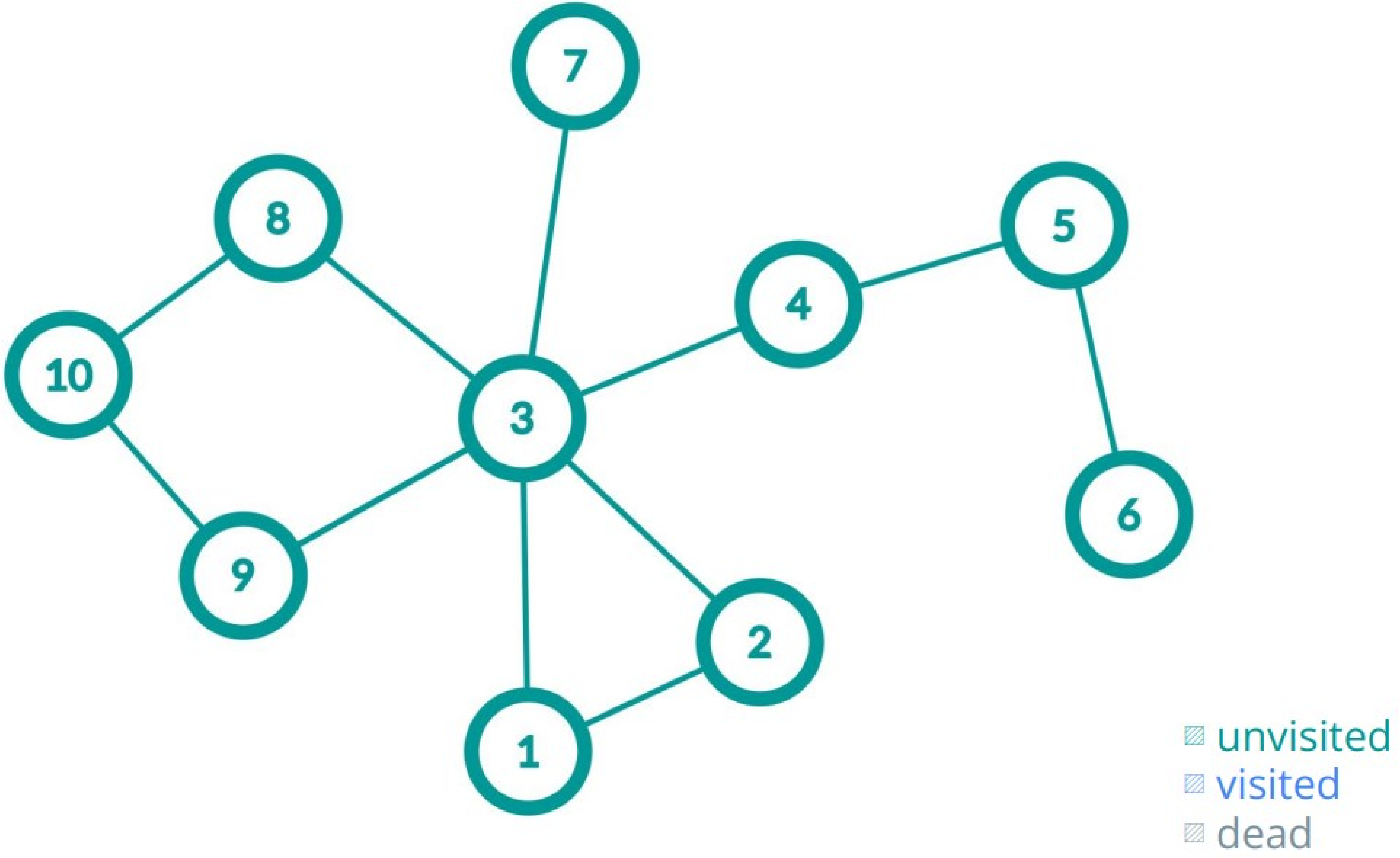
1. **Depth-First Search (DFS)**
2. Breadth-First Search (BFS)

What is DFS?

- DFS is an algorithm to traverse the graph.
- “Depth-First” means to traverse to the deeper node first for each time.
- Do not mix it with the DFS technique on branch and bound.

- Time complexity for adjacency list: $O(|V| + |E|)$
- Time complexity for adjacency matrix: $O(|V|^2)$

Function call



Procedure (using recursion)

```
procedure DFS (vertex x)
        mark x as visited
        do something
        for all vertices that are neighbors of x and are unvisited (v)
                call DFS(v)
```

To start DFS, simply call DFS(source vertex).

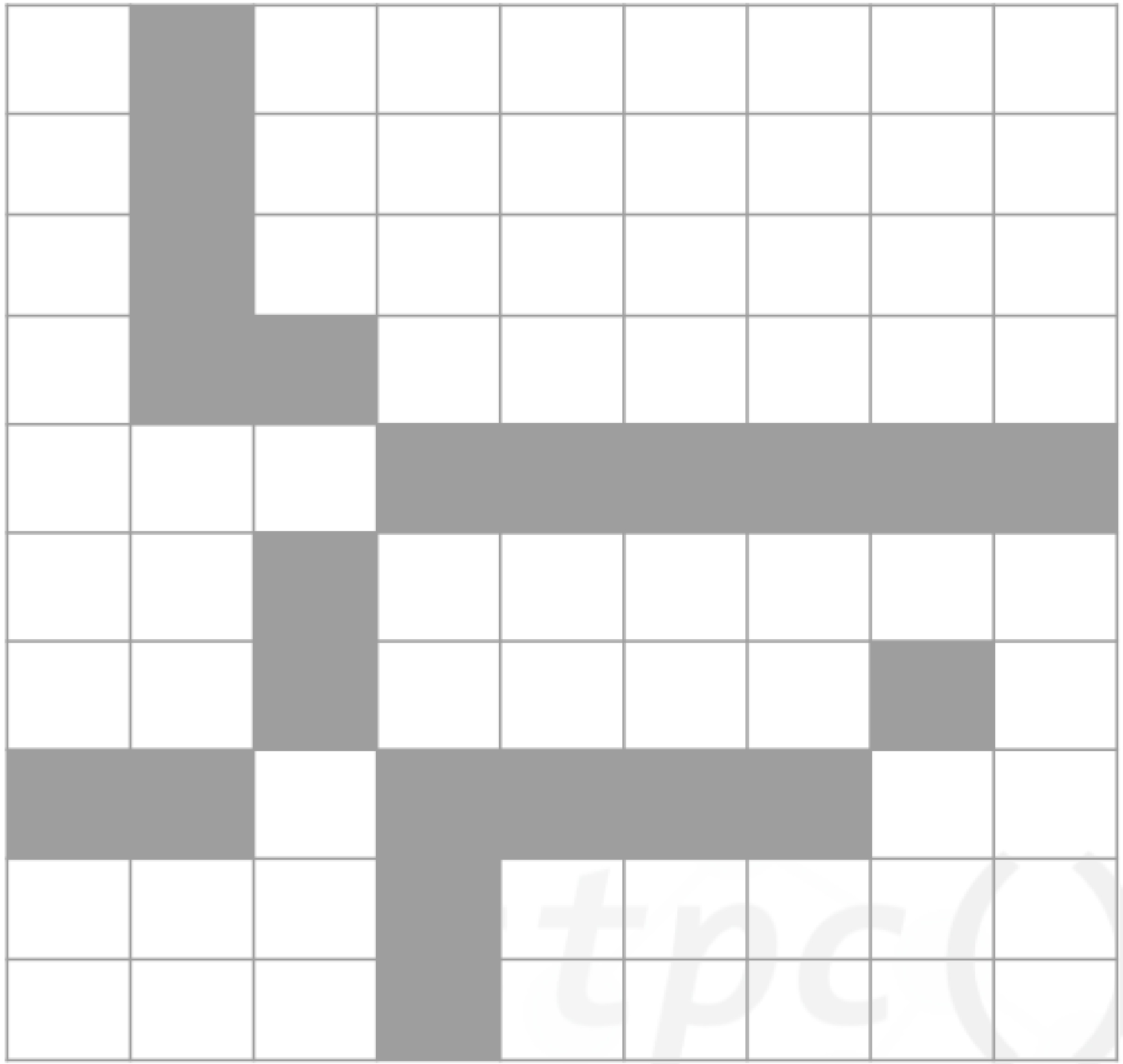
Procedure (using stack)

```
push source vertex (u) into stack S
while (S is not empty)
      pop the top element (x) in S
      if x is not visited
            do something
            mark x as visited
            push all unvisited vertices that are neighbors of x into S
```

Actually, recursion made use of stack!

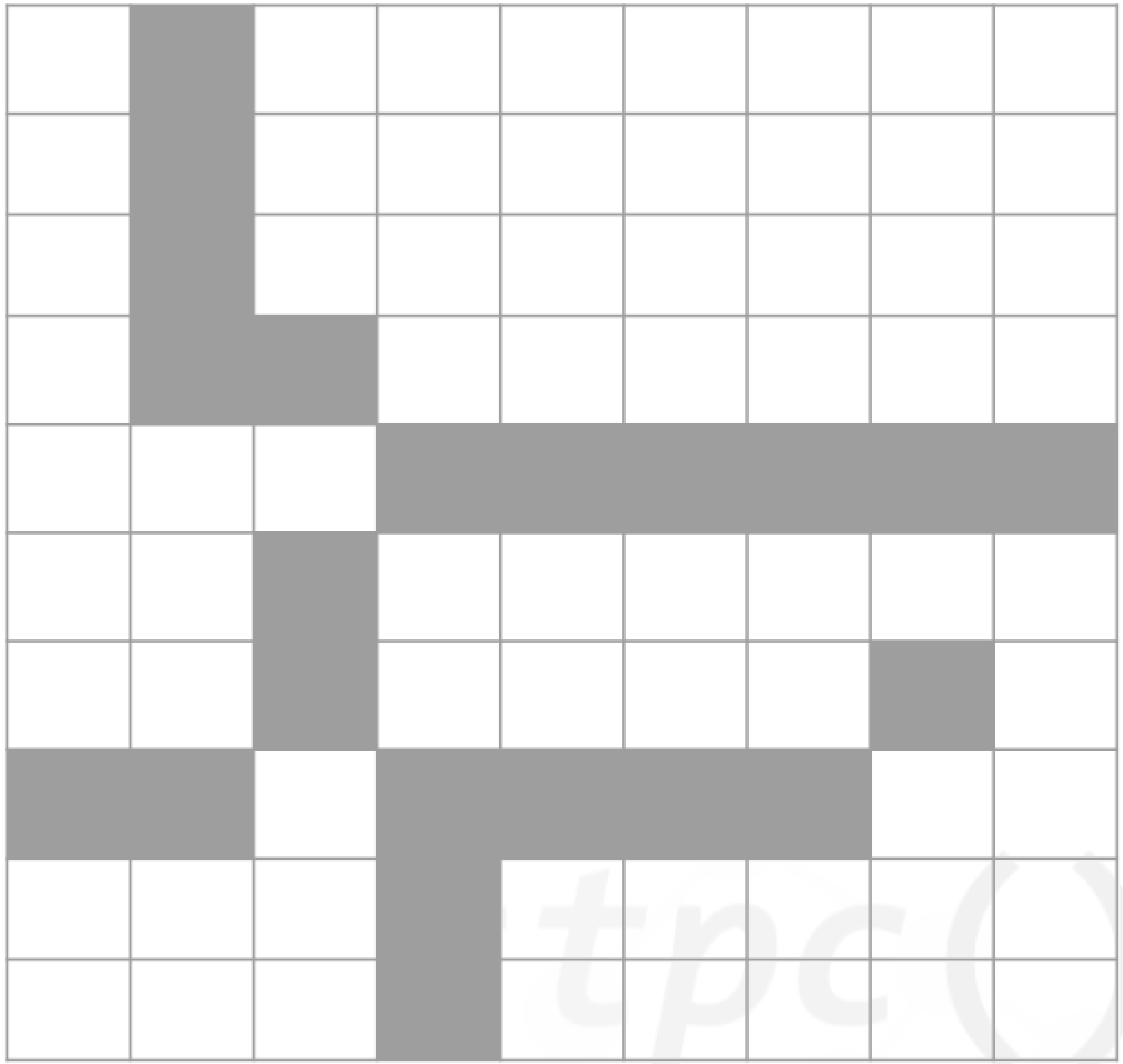
Flood fill

- A grid can be modelled as an undirected graph.
- How many regions are there?
- How large is each region?



Flood fill

- When we detect an unvisited empty cell, we run DFS on it and count the number of cells in this region.
- After visiting a region, we perform DFS on the remaining unvisited empty cells until all empty cells are visited.



Cycle detection

- How to detect a cycle in an undirected graph / a directed graph?
- During our DFS process, if there exists a visited neighbor of current node except its “parent”, the graph contains a cycle.

- For directed graph, you can use Disjoint Union-Find to find cycles.
- If you have learnt about topological sort, you can use Kahn’s algorithm to find the existence of a cycle in the graph.

Practice Problem

- Flooded City
- Given a $N*M$ grid, find the number of cells which are surrounded by walls.
- Flood fill → It counts the number of cells in each regions
- How to ensure the region is not surrounded by walls?
- Method 1: Check if some cell inside the region is near the boundary of the grid.
- Method 2: Construct an outermost cells surrounding the grid, and all the regions near the boundary of the grid is connected.

Graph (II)

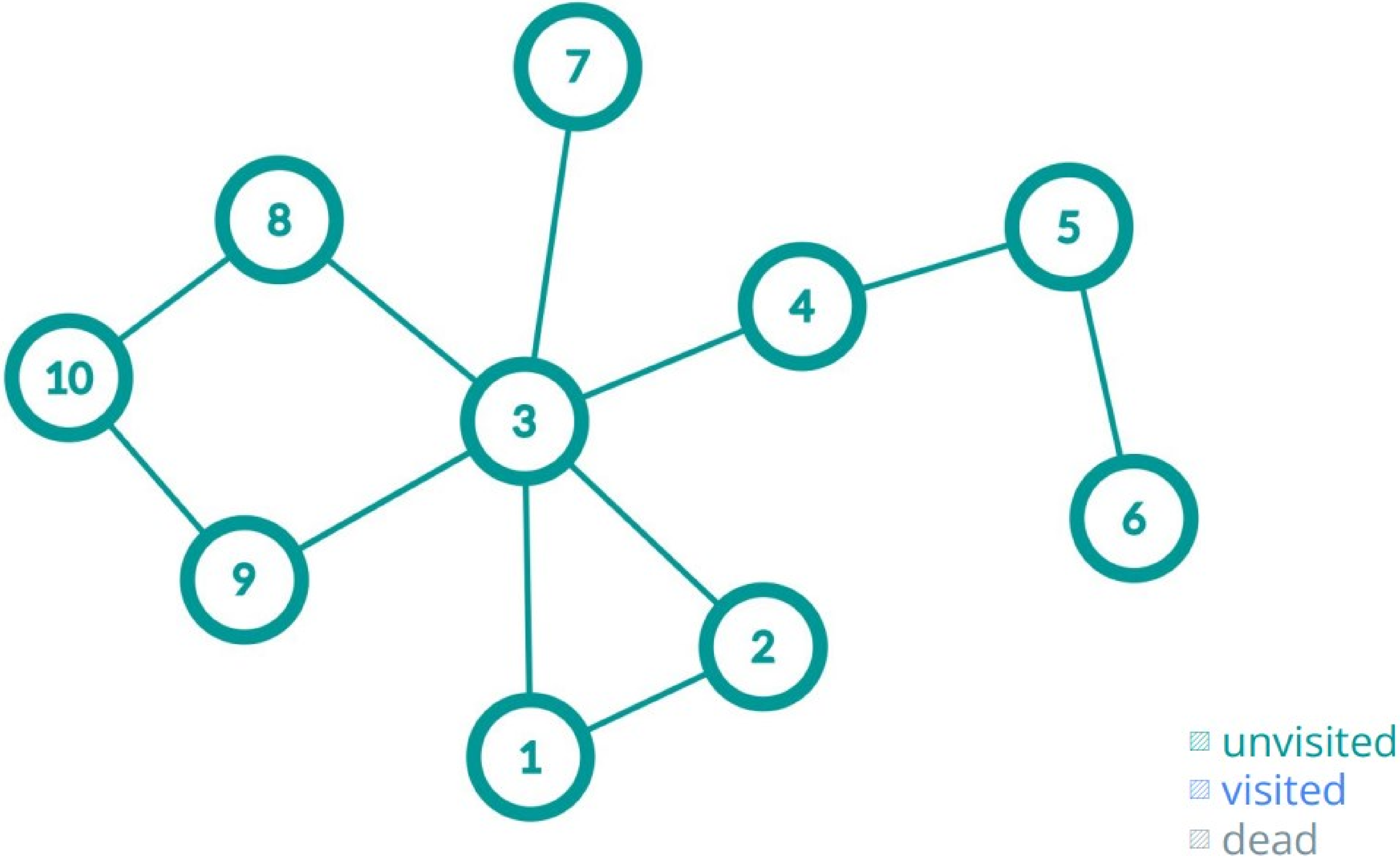
1. Depth-First Search (DFS)
2. **Breadth-First Search (BFS)**

What is BFS?

- BFS is another algorithm to traverse the graph.
- “Breadth-First” means to traverse to the node with the same height first for each time.

- Time complexity for adjacency list: $O(|V| + |E|)$
- Time complexity for adjacency matrix: $O(|V|^2)$

Queue



Procedure (using queue)

```
push source vertex (u) into queue Q
while (Q is not empty)
      deq the top element (x) in Q
      if x is not visited
            do something
            mark x as visited
            enq all unvisited vertices that are neighbors of x into S
```

Water Jug Problem

- There are two water jugs with capacities N and M liters respectively.
- Initially, both of them are empty.
- You can perform the following operations for infinitely many times (one operation a time)
  1. Empty a jug
  2. Fully fill a jug
  3. Pour water from one jug to another until either one jug is empty / full
- How to get a specific volume K in one of the jugs?

Water Jug Problem

- Run BFS from (0, 0)

N = 3, M = 4, K = 2

States:

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)
(1, 0)	(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 0)	(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 0)	(3, 1)	(3, 2)	(3, 3)	(3, 4)

initial state
target states

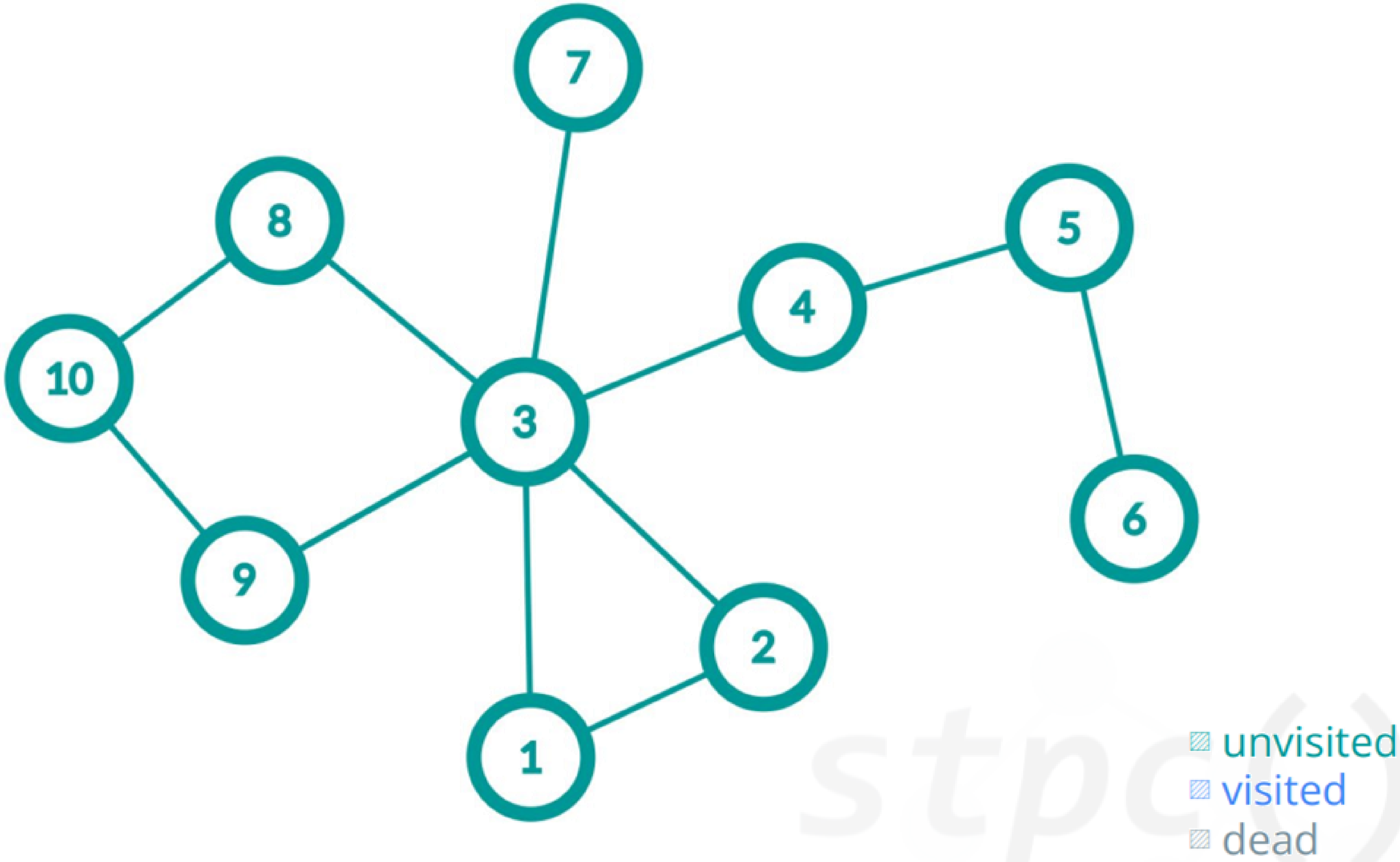
Water Jug Problem

- There are four possible transition we could do
  - Empty a jug (x, 0) and (0, y)
  - Fully fill a jug (x, M) and (N, y)
  - Pour water from the first jug to the second jug
  - Pour water from the second jug to the first jug

- This is a common technique to list all possible transitioning first before implementing BFS.

Shortest path for unweighted graph

- Let’s revise the BFS simulation.
- In this time, let’s focus on the distance of each node from the source vertex.


7
8
5
4
10
3
6
9
2
1
unvisited
visited
dead

Shortest path for unweighted graph

- It is obvious for us to observe a critical property of BFS.
- We can maintain an array dist[] to find the shortest path from a fixed vertex.
- This method can be both applicable to unweighted graph and weighted graph.

- For the shortest path for a weighted graph, you should learn more advanced algorithm such as Dijkstra’s algorithm or Floyd-Warshall algorithm.

Shortest path for unweighted graph

- Some properties can be further extended by the BFS idea.
- Find vertices contributes to the shortest path between two vertices.
  - Maintain two array d_A and d_B, storing the shortest distance from vertex A and B respectively.
  - For a vertex v, if $d_A[v] + d_B[v] = d_A[B]$, then v contributes to the shortest path between vertex A and B.
- The idea above can be extended to find edges contributes to the shortest path between two vertices.

Shortest path for unweighted path

- BFS is the foundation of shortest path algorithm. The idea can be further extended.
- BFS + Double-ended queue = 0-1 BFS
- BFS + Priority queue = Dijkstra's Algorithm
- … much more!

Practice Problem

- Weird Knights
- Output the minimum steps for the “weird knight” to go to (1, 1).

- The cost to perform each move is the same, therefore BFS to find the shortest path is feasible.
- Enumerate the valid position for each move before BFS.
- After that, simply BFS and AC.

Q&A