Day 10 : Graphs :

(1) Representations:

Task:

There are n cities. Some of them are connected, while some are not. If city a is connected directly with city b, and city b is connected directly with city c, then city a is connected indirectly with city c.

A province is a group of directly or indirectly connected cities and no other cities outside of the group.

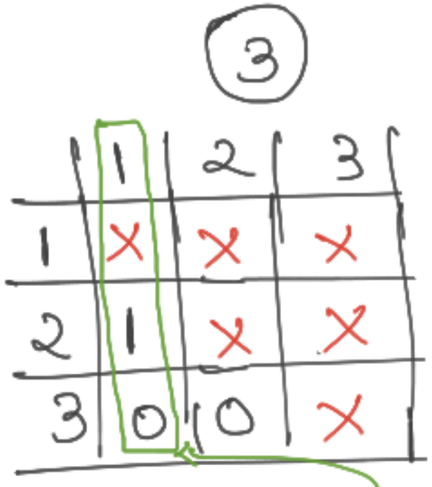
Return the total number of provinces.

Takeaway:

Given some connected components,

1. create an adjacency matrix
2. perform DFS / BFS
3. loop through remaining nodes

Adjacency Matrix



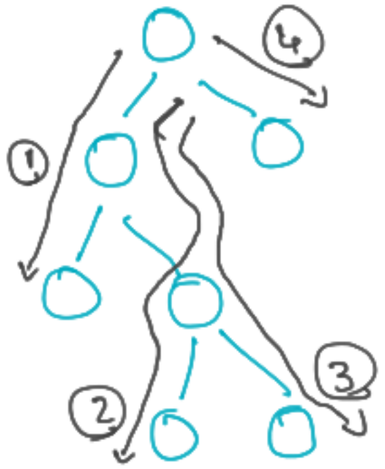
province_1 = [1] — DFS or BFS

province_1 = [1,2] → no other adjacent node

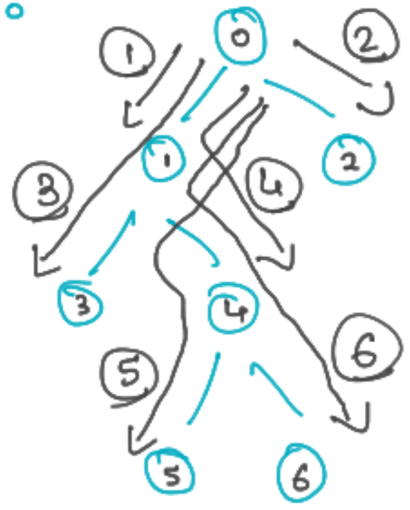
province_2 = [3]

(2) Graph Search:

DFS:



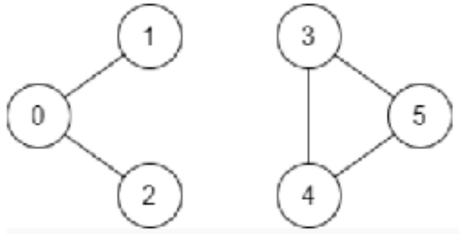
BFS:



when to use what? (just heuristics)

1. If you know a solution isn't far from the root: BFS
2. If tree is very deep and solutions are rare: BFS
3. If tree is wide: BFS would take too much memory
4. If solution are frequent, and located deep within the tree: DFS
5. If tree is very deep, restrict the depth of DFS
   → Iterative Deepening

Task: Find if there is a valid path that exists from vertex start (given in the input) to vertex end (given in the output).




```
Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]],
start = 0, end = 5
Output: false
Explanation: There is no path from vertex 0 to vertex 5.
```

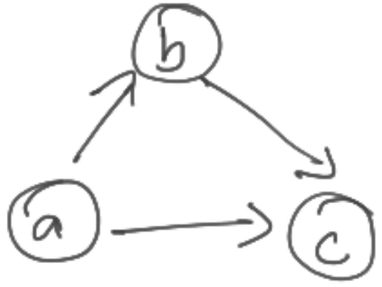
③ Graph Cycles:

Same concept as DFS / BFS

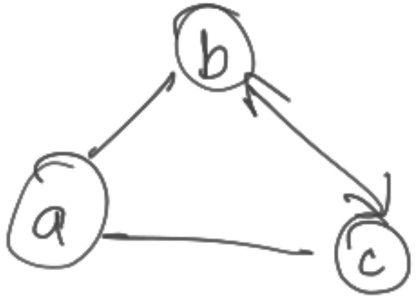
Cycle exists if we encounter already visited node.

Things to note:

1. Directed Graphs:



2. Undirected Graphs: /Bidirected



Good to have High level Idea of:

1. Kruskal's Min. Spanning Tree algorithm
2. Prim's Min. Spanning Tree algo
3. Dijkstra's shortest path algo.
4. Topological sort for Directed Acyclic Graphs