Graphs III

AGENDA

- Earliest Ancestors
- Connected Components
- Building your own social graph
- *I’m hosting a Q&A on Thursday 6-7PM PST*

LAST SESSION RECAP

- How to Solve Any Graph Problem
  - Translate the problem into graph terminology
  - Build Your Graph
  - Traverse Your Graph
- Problem Walkthrough: Earliest Ancestor

EARLIEST ANCESTOR

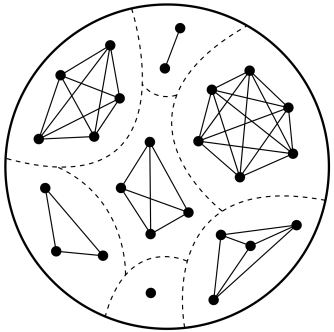
- Translate the problem into graph terminology
    - Vertices = Person/ID
    - Edge = parent-child relationship
    - Path = ancestor tree
- Build Your Graph
    - Build a directed graph from child to parent
- Traverse Your Graph
    - Traverse a person’s family tree upwards to its farthest ancestor

Connected Components

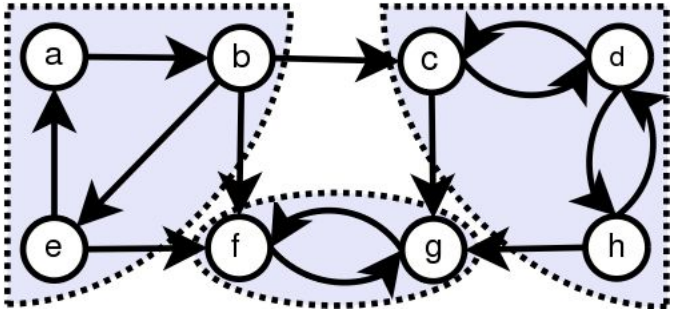
CONNECTED COMPONENTS

- A graph can be composed of subgraphs
- These subgraphs are called ***connected components***

Connected Components



Strongly Connected Graph



USES OF CONNECTED COMPONENTS

- Determining your social circle (e.g. mutual friends)
- Determining connected nodes/devices in a network
- Analyzing epidemics (spread, origin, etc.)
- Determining if there’s a possible route between two places (e.g. flights)

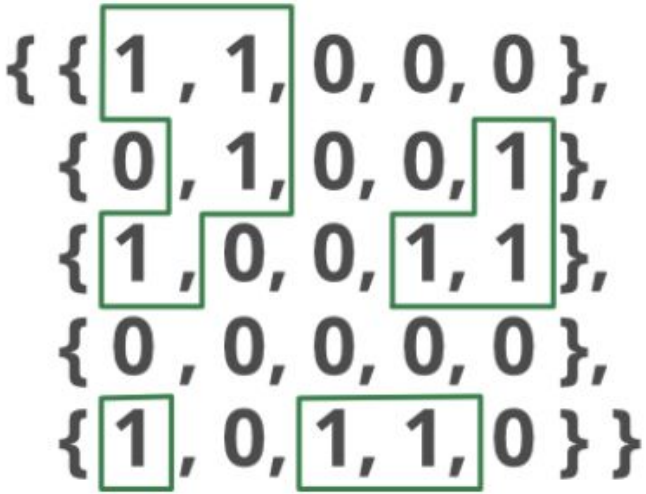
HOW DO WE FIND CONNECTED COMPONENTS?

- Traverse the graph and keep tabs on which nodes are connected to each other
- DFS or BFS works

Finding Connected Components Demo

COUNT ISLANDS

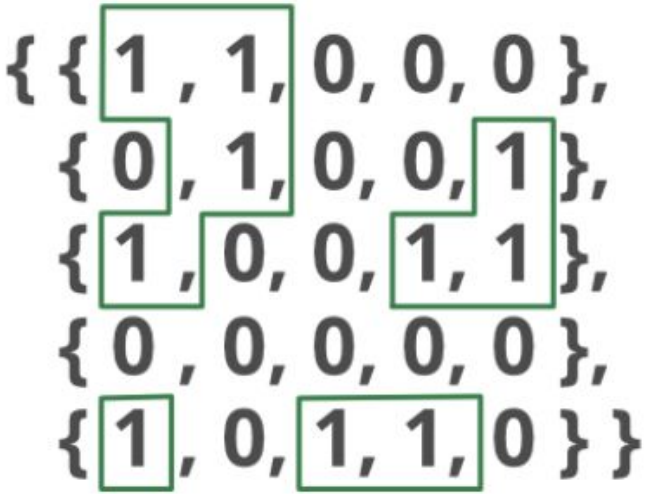
- Count the number of islands in a matrix
- [Leetcode Link]



Count Islands Demo

COUNT ISLANDS: RECAP

- We found the connected components via DFS
- You don’t need a traditional graph representation to do BFS/DFS
- You can use these traversal techniques for 2-dimensional arrays as well



Building Your Own Social Graph

SOCIAL GRAPH

- Build your own randomized social graph
- Create users
- Add friendships
- Traverse the graph to see social connections
- Link

Social Graph Demo