Documentation of the "547. Number of Provinces"

1. **Problem Statement**

   You are given an n x n adjacency matrix isConnected representing n cities.

   - isConnected[i][j] = 1 means city i is directly connected to city j.
   - isConnected[i][j] = 0 means there is no direct connection.

   A province is defined as a group of cities that are directly or indirectly connected and no other cities outside this group.

   Goal: Return the total number of provinces.

2. **Intuition**

   This is essentially a graph traversal problem, where:

   - Cities are nodes
   - Direct connections are edges

   We need to count how many disconnected components (connected subgraphs) are present. Each component equals one province.

3. **Key Observations**

   - The matrix is symmetric: if city i is connected to j, then j is connected to i.
   - A city is always connected to itself (isConnected[i][i] = 1).
   - This is an undirected graph.
   - The problem boils down to counting the number of connected components in the graph.

4. **Approach (DFS-Based)**

We use Depth-First Search (DFS):

- Keep a visited list to track visited cities.
- Loop through each city:
  - If it's unvisited, we perform DFS from that city.
  - Each DFS call explores an entire province (connected component).
  - Increment the province count.
- Return the total count.

5. **Edge Cases**

- All cities are connected: Output should be 1.
- No cities are connected: Output should be n.
- Single city (n=1): Output is 1.
- Matrix with only diagonals as 1s: Each city is isolated → n provinces.

6. **Complexity Analysis**

 Time Complexity

- $O(n^2)$ — We potentially examine each cell in the matrix.

 Space Complexity

- $O(n)$ — For the visited list and recursion stack (DFS).

7. **Alternative Approaches**

- Union-Find (Disjoint Set Union)
  - Uses a parent array to track connected components.
  - Efficient for sparse connections.

- Union cities if they are directly connected.
- Final count of unique roots gives the number of provinces.

- Breadth-First Search (BFS)
    - Similar to DFS, but uses a queue to traverse connected cities level-by-level.

8. Test Cases

isConnected Input	Expected Output	Explanation
[[1,1,0],[1,1,0],[0,0,1]]	2	Cities 0-1 are connected; 2 is isolated
[[1,0,0],[0,1,0],[0,0,1]]	3	All cities are isolated
[[1,1,1],[1,1,1],[1,1,1]]	1	All cities are connected to each other
[[1]]	1	Single city, one province
[[1,0,1],[0,1,0],[1,0,1]]	1	Indirect connections form a single province

9. Final Thoughts

- This is a classic connected components problem in graph theory.
- DFS is intuitive and efficient for medium-size graphs (up to n = 200).
- For optimization or dealing with large datasets, Union-Find can be more performant.
- Ensure you don't revisit already visited cities to avoid infinite recursion.