

KEYS AND ROOMS

There are n rooms labeled from 0 to $n - 1$ and all the rooms are locked except for room 0. Your goal is to visit all the rooms. However, you cannot enter a locked room without having its key.

When you visit a room, you may find a set of distinct keys in it. Each key has a number on it, denoting which room it unlocks, and you can take all of them with you to unlock the other rooms.

Given an array `rooms` where `rooms[i]` is the set of keys that you can obtain if you visited room i , return `true` *if you can visit all the rooms*, or `false` *otherwise*.

Input Format

In the function an integer vector of vectors is passed.

Output Format

Return `true` or `false`.

Constraints:

- $n == \text{rooms.length}$
- $2 \leq n \leq 1000$
- $0 \leq \text{rooms}[i].\text{length} \leq 1000$
- $1 \leq \text{sum}(\text{rooms}[i].\text{length}) \leq 3000$
- $0 \leq \text{rooms}[i][j] < n$
- All the values of `rooms[i]` are unique.

Sample Input

```
[[1], [2], [3], []]
```

Sample Output

```
true
```

Explanation

We visit room 0 and pick up key 1. We then visit room 1 and pick up key 2. We then visit room 2 and pick up key 3. We then visit room 3. Since we were able to visit every room, we return `true`.

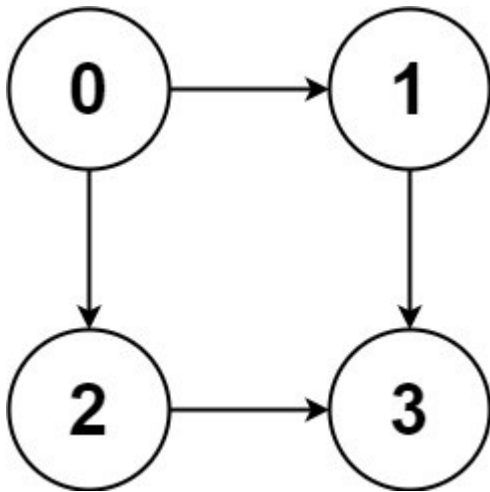
Solution: `keysAndRooms.cpp`

All Paths From Source to Target

Given a directed acyclic graph (DAG) of n nodes labeled from 0 to $n - 1$, find all possible paths from node 0 to node $n - 1$ and return them in any order.

The graph is given as follows: $\text{graph}[i]$ is a list of all nodes you can visit from node i (i.e., there is a directed edge from node i to node $\text{graph}[i][j]$).

Example :

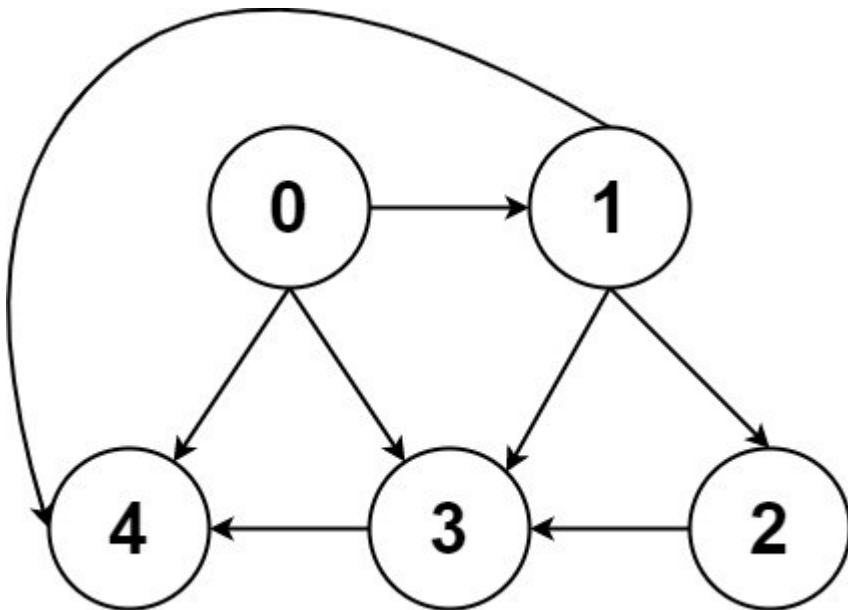


Input: $\text{graph} = [[1,2],[3],[3],[]]$

Output: $[[0,1,3],[0,2,3]]$

Explanation: There are two paths: $0 \rightarrow 1 \rightarrow 3$ and $0 \rightarrow 2 \rightarrow 3$.

Example 2:



Input: $\text{graph} = [[4,3,1],[3,2,4],[3],[4],[]]$

Output: $[[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]$

Constraints:

- $n == \text{graph.length}$
- $2 \leq n \leq 15$
- $0 \leq \text{graph}[i][j] < n$
- $\text{graph}[i][j] \neq i$ (i.e., there will be no self-loops).
- All the elements of $\text{graph}[i]$ are unique.
- The input graph is guaranteed to be a DAG.

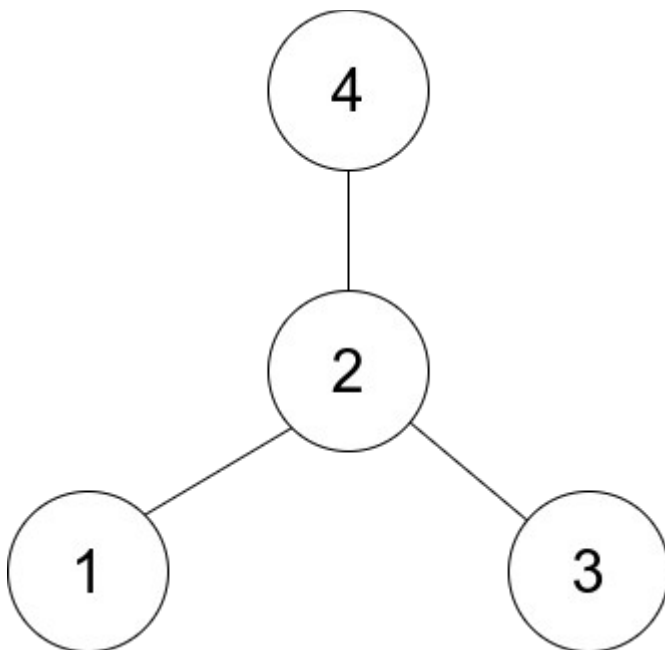
Solution: allPaths.cpp

Find Star in the graph

There is an undirected star graph consisting of n nodes labeled from 1 to n . A star graph is a graph where there is one center node and exactly $n - 1$ edges that connect the center node with every other node.

You are given a 2D integer array `edges` where each `edges[i] = [ui, vi]` indicates that there is an edge between the nodes `ui` and `vi`. Return the center of the given star graph.

Example :



Input: `edges = [[1,2],[2,3],[4,2]]`

Output: 2

Explanation: As shown in the figure above, node 2 is connected to every other node, so 2 is the center.

Constraints:

- $3 \leq n \leq 105$
- `edges.length == n - 1`
- `edges[i].length == 2`
- $1 \leq u_i, v_i \leq n$
- $u_i \neq v_i$

The given edges represent a valid star graph.

Solution: findStar.cpp