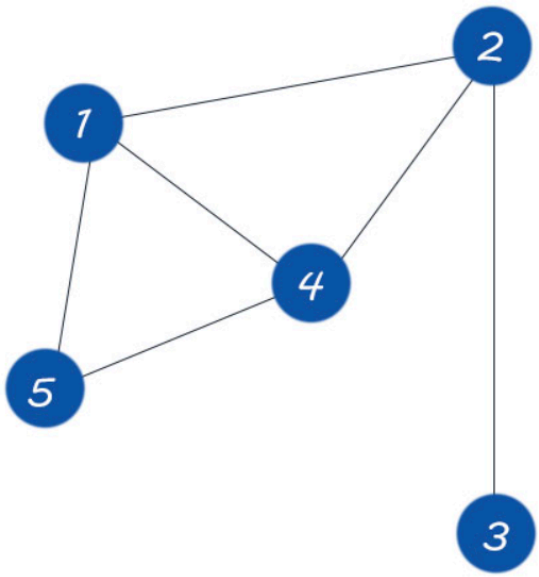
Exercise 5: dfs

TA: Ren, Peng

Date: 11/29/2023

Question 1:

1. Given the same graph below, perform a Depth-First Search (DFS) starting from node 1. Write down the order in which the nodes are visited.



Question 2:

The following quesiton is a medium question in leetcode (Same question in Exercise 4). Now we need to use dfs to solve this problem.

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*.

Example 1:

```
Input: rooms = [[1],[2],[3],[]]
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.
```

Example 2:

```
Input: rooms = [[1,3],[3,0,1],[2],[0]]
Output: false
Explanation: We can not enter room number 2 since the only key that unlocks it is in that
room.
```

Constraints:

- `n == rooms.length`
- `2 <= n <= 1000`
- `0 <= rooms[i].length <= 1000`
- `1 <= sum(rooms[i].length) <= 3000`
- `0 <= rooms[i][j] < n`
- All the values of `rooms[i]` are **unique**.

Please complete the following code:

```c
#include <stdbool.h>
#include <stdlib.h>

/**
 * bool dfs(int cur_room_id, int** rooms, int roomsSize, int* roomsColSize, bool* visited)
 *
 * @param cur_room_id: Current room number being explored.
 *
 * @param rooms: Array of arrays; each sub-array contains keys to unlock other rooms.
 *
 * @param roomsSize: Total number of rooms.
 *
 * @param roomsColSize: Array indicating the number of keys in each room.
 *
 * @param visited: Array to track whether a room has been visited.
 */
```

```c
bool dfs(int cur_room_id, int** rooms, int roomsSize, int* roomsColSize, bool* visited) {
    if (cur_room_id >= roomsSize) {
        return false;
    }

    /*
    Mark currentRoom as visited

    If all rooms are visited:
        Return true

    For each key in currentRoom:
        If the room unlocked by the key is not visited:
            Recursively call dfs on that room
            If the result is true:
                Return true

    Return false
    */
}

bool canVisitAllRooms(int** rooms, int roomsSize, int* roomsColSize) {
    bool *visited = (bool *)malloc(sizeof(bool) * roomsSize);
    for (int i = 0; i < roomsSize; i++) {
        visited[i] = false;
    }

    bool result = dfs(0, rooms, roomsSize, roomsColSize, visited);
    free(visited);
    return result;
}
```