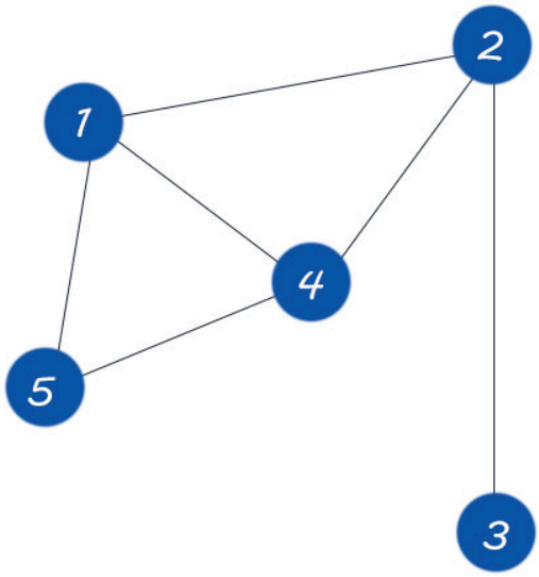
Exercise 4: bfs & graph

TA: Ren, Peng

Date: 11/29/2023

Question 1:

1. How many edges are here in the following graph? How many vertices are there?
2. Given the graph, perform a Breadth-First Search (BFS) starting from node 1. Write down the order in which the nodes are visited.



3. Please write down the adjacent array of this graph. It should be a 2d array.
4. Please write down the adjacent matrix of this graph, it should be a 2d matrix.

Question 2:

The following quesiton is a medium question in leetcode:

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**.

Let's assume the queue has been implemented:

```c
#define MAX_ROOMS 1000

typedef struct {
    int items[MAX_ROOMS];
    int front;
    int rear;
} Queue;

void initQueue(Queue *q) {
    q->front = -1;
    q->rear = -1;
}

int isEmpty(Queue *q) {
    return q->front == -1;
}
```

```c
void enqueue(Queue *q, int item) {
    if (q->front == -1) {
        q->front = 0;
    }
    q->rear = (q->rear + 1) % MAX_ROOMS;
    q->items[q->rear] = item;
}

int dequeue(Queue *q) {
    int item = q->items[q->front];
    if (q->front == q->rear) {
        // Queue is now empty
        q->front = -1;
        q->rear = -1;
    } else {
        q->front = (q->front + 1) % MAX_ROOMS;
    }
    return item;
}
```

Please complete the following bfs code according to the pseudocode given:

```c
int canVisitAllRooms(int **rooms, int roomsSize, int* roomsColSize){
    int visited[MAX_ROOMS] = {0};
    Queue q1; // Initialize a queue 'q1'
    initQueue(&q1);
    enqueue(&q1, 0); // Enqueue the starting room (0) into 'q1'
    visited[0] = 1; // Mark room 0 as visited
    int visitedCount = 1; // Initialize visitedCount to 1

    // Please write code according to the pseudocode below
    /*
    While 'q1' is not empty:
        Initialize a second queue 'q2'

        While 'q1' is not empty:
            Dequeue an element from 'q1' and store it in integer named 'room'
            Iterate over each key in rooms[room]:
                If the key's corresponding room has not been visited:
                    Mark this room as visited
                    Increment visitedCount
                    Enqueue the key into 'q2'

        Transfer all elements from 'q2' to 'q1'
```

```
    */

    // Determine the return value:
    // How would you determine whether to return true or false?
    return // Complete this statement.
}
```