Problem 1. Assume you use breadth-first search to actually find your way out of a maze (consisting of rooms and hallways, which represent the vertices and edges, respectively). One room is the start room, and one room is the exit room. We want to count how many hallways you need to walk down. If you walk down the same hallway twice that counts as two. Each direction counts once, so back-and-forth across a hallway counts as two. The first time you visit a room must be in the order of breadth-first search. Other than that you can be clever.

(a) Assume the maze is one long path of \(n\) rooms where you start on one end and the exit is at the other end. How many hallways do you walk down? Justify.

(b) Assume the maze consists of a start room with \(n-1\) other rooms directly connected to it. (In other words it is a star graph.) Assume the last room you visit is the exit room. How many hallways do you walk down? Justify.

(c) Assume the maze is a complete binary tree with \(n = 2^h - 1\) rooms. Assume that the root is the start room. To keep things simple, assume the root is also the exit room but only after you have visited all of the other rooms. How many hallways do you walk down? Justify.

Problem 2. Design an efficient algorithm to find a universal sink, i.e., a vertex with in-degree \(n - 1\) and out-degree 0 in a directed graph. The graph is given to you in an adjacency matrix representation and the weight of each edge is 1. How many exact cells would you need to check in the adjacency matrix?