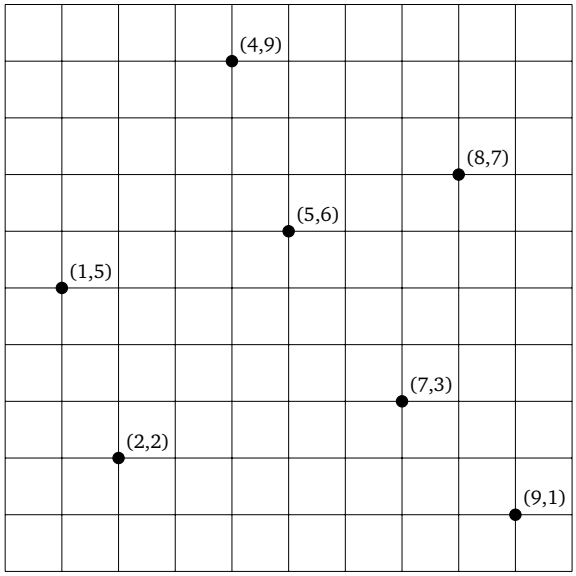
Section 06: Graphs

1. k-d Trees

(a) Given the points shown in the grid to the right, draw a perfectly balanced k-d tree in the box below. For this tree, first split on the x dimension. The resulting tree should be complete with height 2. Then, draw the corresponding splitting planes on the grid to the right.

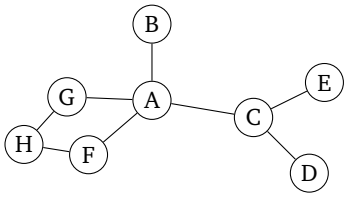


(b) Insert the point (6, 2) into the k-d tree you drew below. Then, add that point to the grid and draw the corresponding splitting plane.

(c) Find the nearest point to (3, 6) in your k-d tree. Mark each branch that is not visited (pruned in execution of `nearest`) with an `X` through the branch.

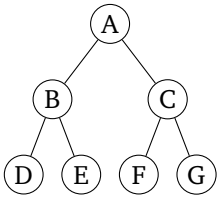
2. Graph traversal

(a) Consider the following graph. Suppose we want to traverse it, starting at node A.



If we traverse this using *breadth-first search*, what are *two* possible orderings of the nodes we visit? What if we use *depth-first search*?

(b) Same question, but on this graph:



3. Implementing graph searches

(a) Come up with pseudocode to implement *breadth-first search* on a graph, given a starting node and an adjacency list representation of the graph. Is your method recursive or not? What data structures do you use?

(b) Come up with pseudocode to implement *depth-first search* on a graph, given a starting node and an adjacency list representation of the graph. Is your method recursive or not? What data structures do you use?

4. Design Problem: Pathfinding in mazes

Suppose we are trying to design a maze within a 2d top-down video-game. The world is represented as a grid, where each tile is either an impassable wall, an open space a player can pass through, or a *wormhole*. On each turn, the player may move one space on the grid to any adjacent open tile. If the player is standing on a wormhole, they can instead use their turn to teleport themselves to the other end of the wormhole, which is located somewhere else on the map.

Now, suppose the there are several coins scattered throughout the map. Your goal is to design an algorithm that finds a path between the player and some coin in the fewest number of turns possible.

Describe how you would represent this scenario as a graph (what are the vertices and edges? Is this a weighted or unwighted graph? Directed or undirected?). Then, describe how you would implement an algorithm to complete this task.