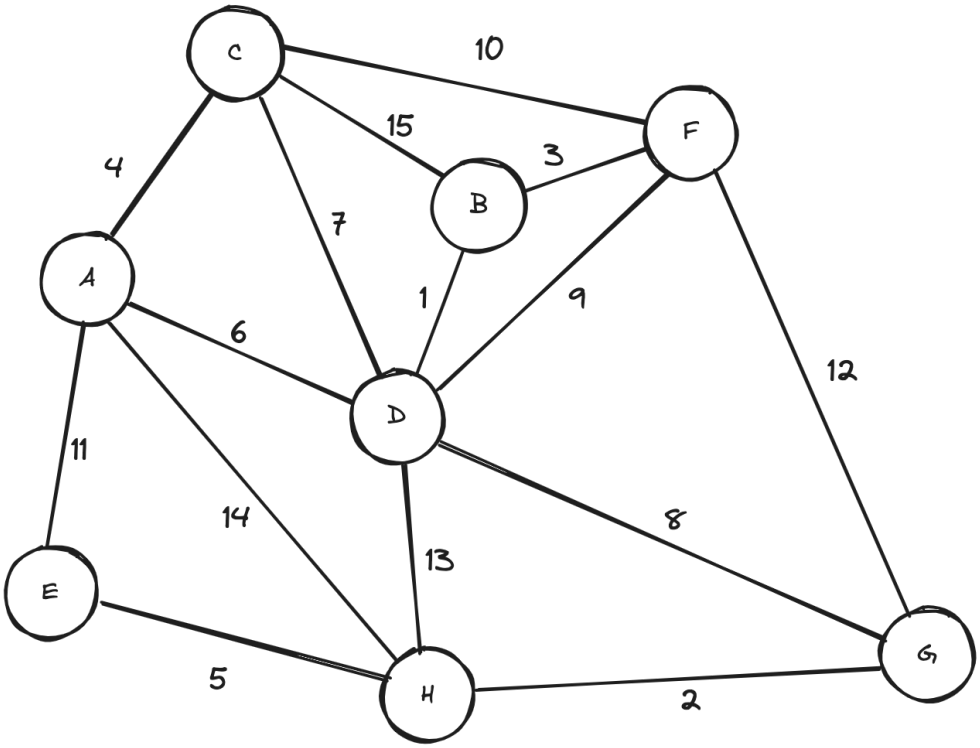
CS161 SU24: Homework 6 (Due July 20 11:59pm)

❖ Problem 1 (30 points)

For the following graph, state the order in which edges get added to the MST if we use

1. Kruskal's algorithm?
2. Prim's algorithm with source vertex B?

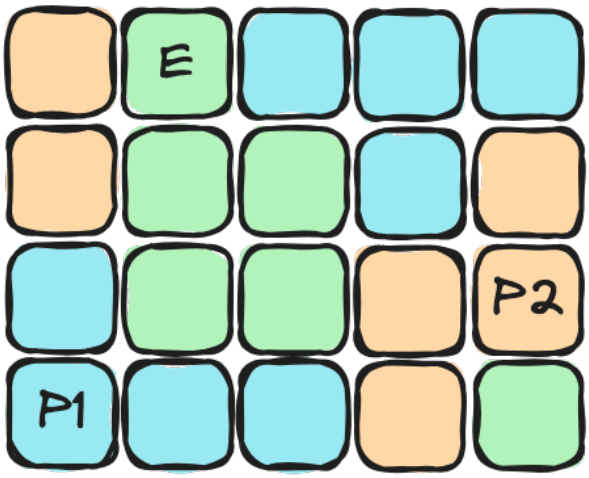


❖ Problem 2 (30 points)

You are implementing a video game that plays on a square grid, with 2 players. You want to implement a path finding algorithm for enemies to be able to efficiently find the nearest player, and start navigating towards them via the shortest path. There are three types of cells in the game: ground, water, and grass. To transition from ground or grass to water, the enemy must take some time to build a boat, but once in water they can move quickly. The following summarizes the time it takes an enemy to move from one type of cell to another.

- Ground to ground: 2 seconds
- Grass to grass: 4 seconds
- Water to water: 1 second
- Ground to grass OR grass to ground: 3 seconds
- Ground to water OR grass to water: 6 seconds
- Water to ground OR water to grass: 5 seconds

In the following figure, ground is denoted in brown, grass in green, and water in blue. The enemy is located in the grid labeled E, and the players are located in the grids labeled by P1 and P2 respectively.



1. Demonstrate a run of Dijkstra's algorithm on the above grid. You may want to find a way to express the grid as a graph. In your solution, write down the table as we

did in class, and the values it stores at the end of the algorithm. It may help to write the table using the same shape as the grid so they don't need explicit labels.

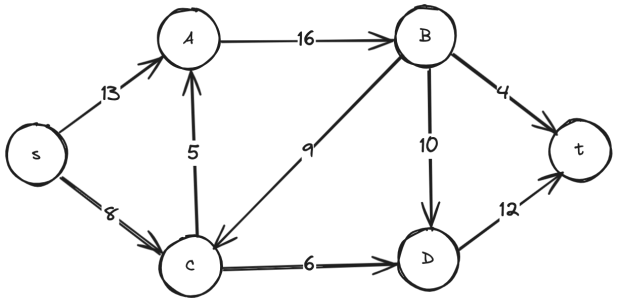
2. Based on the above, which player will the enemy chase?

❖ Problem 3 (20 points)

The implementation of Bellman-Ford we saw in class uses $O(n^2)$ storage for the DP table. Explain how we could reduce the storage space to $O(n)$ if we don't care about reconstructing the path, and only want to know the distances.

❖ Problem 4 (30 points)

1. In class, we discussed the BFS algorithm and showed that it can successfully find a shortest path in unweighted graphs. Come up with a weighted graph for which BFS fails to work if we apply the algorithm as we learned it in class directly.
2. Ford-Fulkerson uses a weighted graph, but we use BFS in our implementation. Explain why this doesn't violate what we discovered above.
3. Demonstrate a run of the Ford-Fulkerson algorithm to find the maximum flow in the following graph. For the first two iterations, draw the
   (a) flow at the start of that iteration,
   (b) the residual network induced from that flow,
   (c) and one augmenting path that can be found in that residual network.



❖ Leetcode

Here is a sample of some Leetcode problems related to graph algorithms that you should be able to start attempting now. Graph problems are quite common, and it will help a lot to be familiar with standard implementations of the data structures and algorithms we learned in class in your language of choice when you prepare for interviews.

They are also a good mix of techniques we've learned in earlier parts of the class including DP and greedy, so it's a good way to review those concepts as well.

- Network Delay Time [Medium]

  https://leetcode.com/problems/network-delay-time/description/

- Path with Maximum Probability

  https://leetcode.com/problems/path-with-maximum-probability/description/

- Longest Increasing Path in a Matrix [Hard]

  https://leetcode.com/problems/longest-increasing-path-in-a-matrix/description/

- Find Critical and Pseudo-Critical Edges in MST [Hard]

  https://leetcode.com/problems/find-critical-and-pseudo-critical-edges-in-minimum-sp
  description/