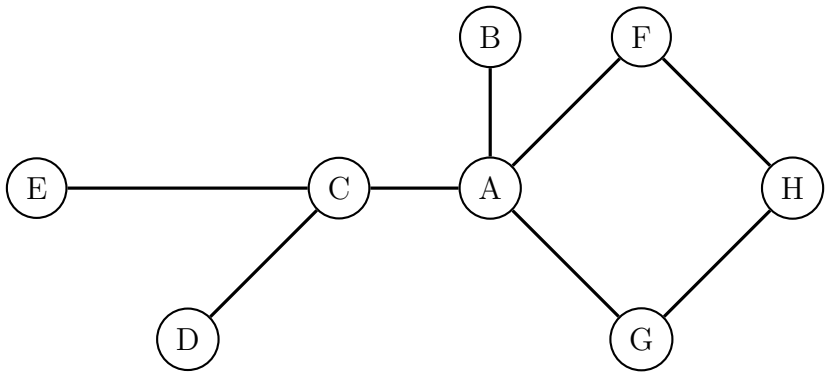
CSE 332
Summer 2018
Section 8 – Graphs!

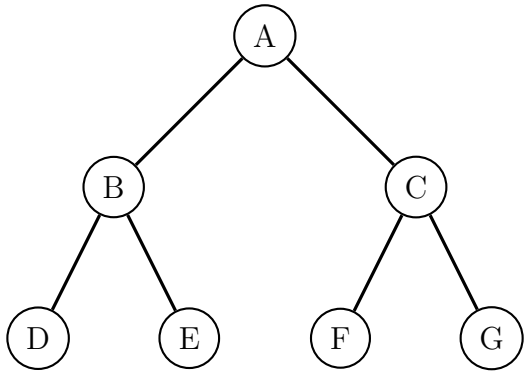
1 Graph Search

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



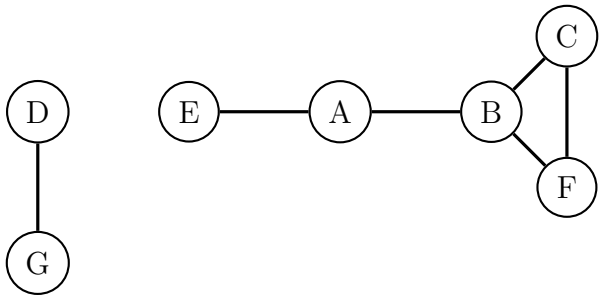
   If we traverse this using *breadth-first search*, what are *two* possible orderings of the nodes we visit? What if we use *depth-first search*?

2. Same question, but on this graph:



2 Graph properties

1. Consider the *undirected, unweighted* graph below.



Answer the following questions about this graph:

(a) Find V, E, $|V|$, and $|E|$.

(b) What is the maximum *degree* of the graph?

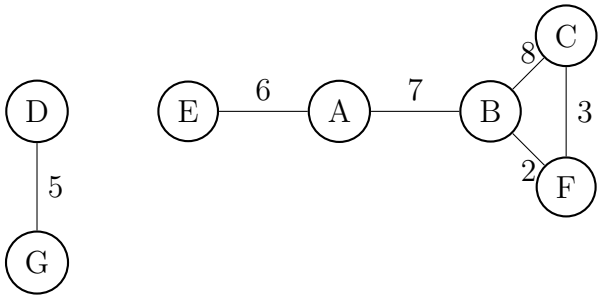
(c) Are there any cycles? If so, where?

(d) What is a maximum length path in this graph?

(e) What is one edge you could add to the graph that would increase the length of the maximum length path of the new graph to 6?

(f) What are the *connected components* of the graph?

2. Consider the *undirected, weighted* graph below.



Answer the following questions about this graph:

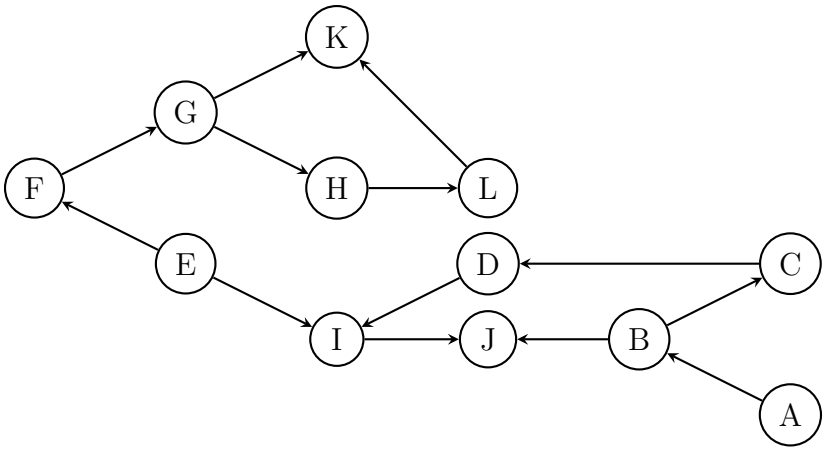
(a) What is the path involving the least number of nodes from E to C? What is its cost?

(b) What is the minimum cost path from E to C? What is its cost?

(c) What is the minimum length path from E to C if we forget the weights of the edges? What is its length?

3 It Rhymes with Flopological Sort

Consider the following graph:

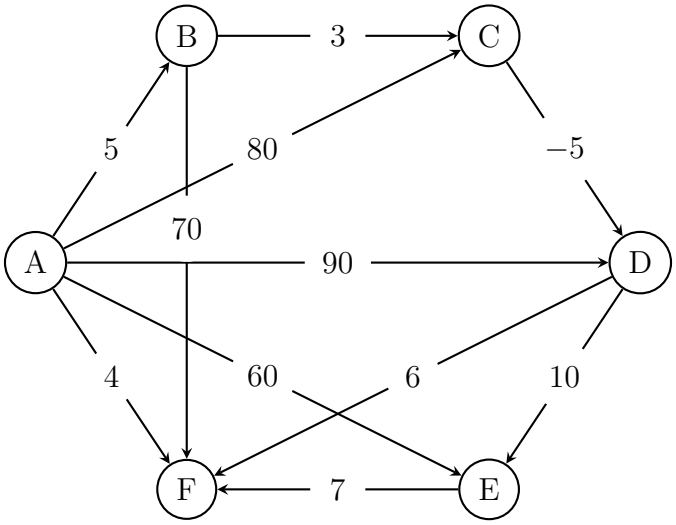


1. Does this graph have a topological sort? Explain why or why not. If you answered that it does not, remove the MINIMUM number of edges from the graph necessary for there to be a topological sort and carefully mark the edge(s) you are removing. Otherwise, just move on to the next part.

2. Find a topological sort of the graph from your answer from part (1).

4 Better Find the Shortest Path Before It Catches You!

Consider the following graph:



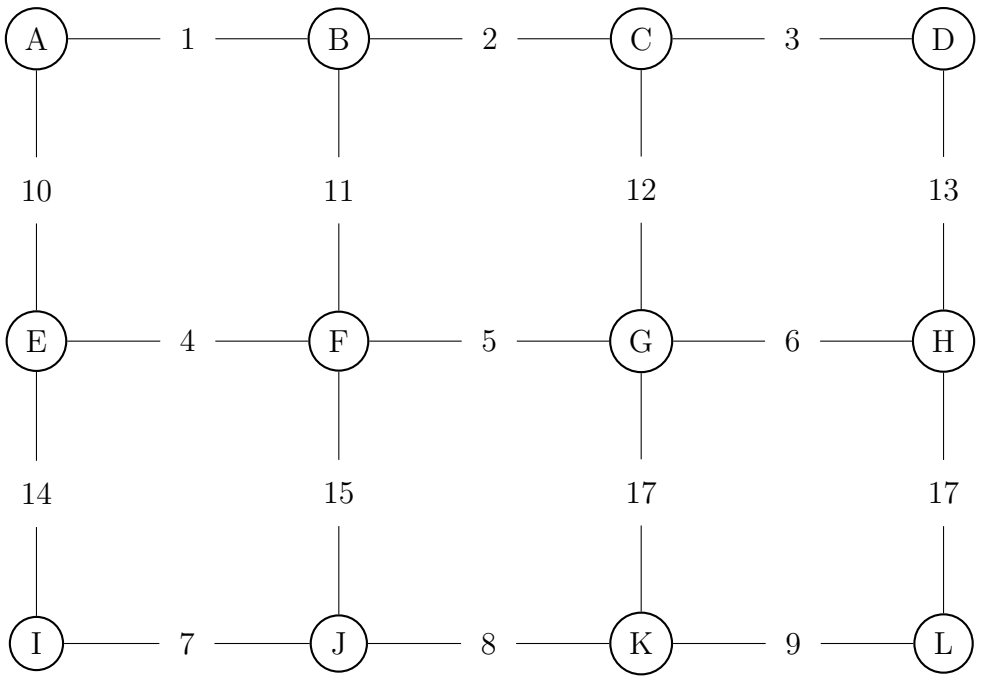
1. Use Dijkstra's Algorithm to find the lengths of the shortest paths from A to each of the other vertices. For full credit, you must show your work at every step.

2. Are any of the lengths you computed using Dijkstra's Algorithm in part (a) incorrect? Why or why not?

3. Explain how you would use Dijkstra's Algorithm to recover the actual paths (rather than just the lengths).

5 LMNST!

Consider the following graph:



1. Find an MST of this graph using both of the two algorithms we've discussed in lecture. Make sure you say which algorithm you're using and show your work.

2. Using just the graph, how can you determine if it's possible that there are multiple MSTs of the graph? Does this graph have multiple MSTs?

3. What is the asymptotic runtime of the algorithms that you used to compute the MSTs?

6 Designing algorithms: 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.

7 Design Problem: Negative Edge Weights

You and your trusty Pikachu have made it halfway through Viridian Forest, but things have taken a turn for the worse. That last Weedle poisoned your Pikachu, and you're all out of antidotes.

In the Pokémon world, the poison doesn't do any damage as long as you stay *perfectly still*. But every time you take a step, the poison does a little bit of damage to your poor friend Pikachu.

Thanks to Bulbapedia[1], you know the exact map of Viridian Forest. Knowing that each step will cost your Pikachu exactly one of its precious hit points, you will need to find an efficient path through the forest.[2]

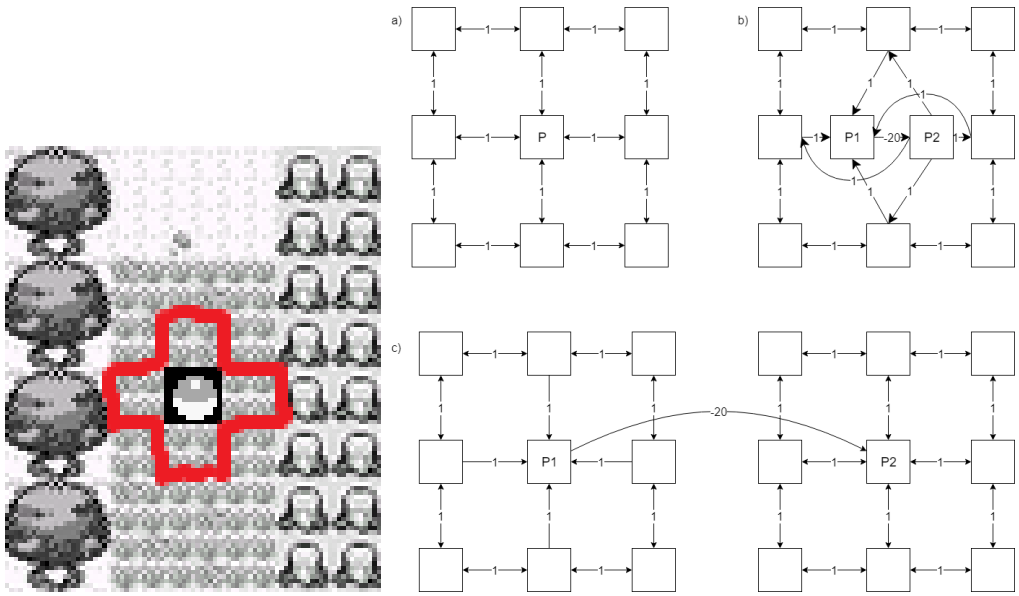
[1] Like Wikipedia, but for Pokémon!

[2] Don't worry about running into wild Pokémon. For some reason you have a huge number of repels. Next time, maybe invest in full heals or potions instead.

1. Describe a graph and an algorithm run on that graph to find the path through the forest to save as many of Pikachu's hit points as possible (i.e. the path with the fewest number of steps).

2. You run your algorithm and come to a devastating realization – the edge of Viridian Forest is at least 25 steps away, and Pikachu has only 20 hit points left. If you just walk to the end of the forest, Pikachu will faint before reaching the next Pokémon Center. So you come up with a backup plan. Returning to Bulbapedia, you see there is a potion just a little bit out of the way of the fastest path.

   Brock tells you he knows how to update your graph to find the best path now. He says he'll add a dummy vertex to the graph where the potion is and connect up the new vertex with a (directed) edge of length -20, to represent undoing the loss of 20 hit points.



5 spots in Viridian Forest, the corresponding vertices before Brock's transformation and the same vertices after the transformation.

   Tell Brock why his representation isn't quite going to work (hint: you can only use the potion once. What happens if the potion edge is part of a cycle?).

3. You convince Brock to change the graph representation. You'll now have two copies of the original Viridian Forest graph, in copy 1 the potion is still unused. In copy 2, the potion is no longer there. You add an edge of weight -20 from copy 1 to copy 2 at the location of the potion (crossing that edge represents using that potion).

   Brock says he'll start running Dijkstra's. Should you trust the output?

4. **Challenge Problem**: Misty says she knows about another potion over there somewhere. Describe how to modify the graph to handle both of the potions.