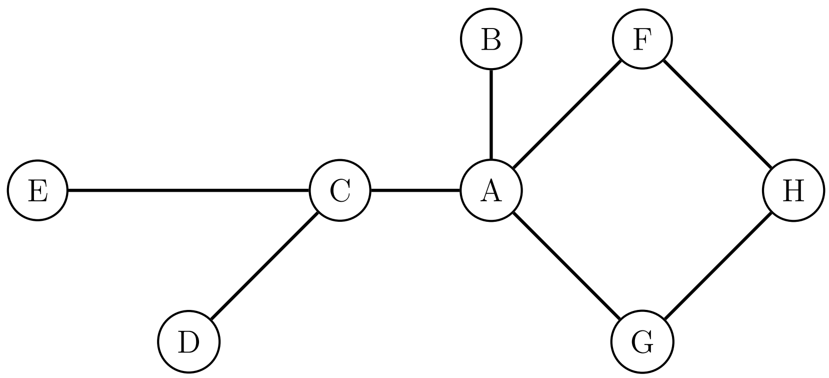
CSE 332
Summer 2018
Section 8 – Graphs!

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*?

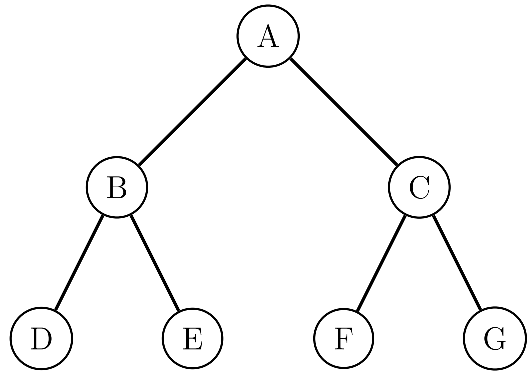
Solution: Here are two possible orderings for BFS:

- A, G, F, B, C, H, D, E
- A, C, B, F, G, D, H, E

Here are two possible orderings for DFS:

- A, G, H, F, C, D, E, B
- A, B, C, E, D, F, H, G

2. Same question, but on this graph:



Solution: Here are two possible orderings for BFS:

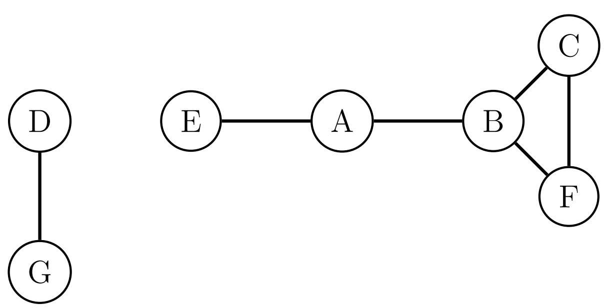
- A, B, C, D, E, F, G
- A, C, B, F, G, D, E

Here are two possible orderings for DFS:

- A, B, D, E, C, F, G
- A, C, G, F, B, E, D

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?

Solution:

(a) $V = \{A, B, C, D, E, F, G\}$ and
$E = \{(D, G), (E, A), (A, B), (B, F), (F, C), (C, B)\}$. This means that $|V| = 7$ and $|E| = 6$.

(b) The vertex with the max degree is B, which has a degree of 3.

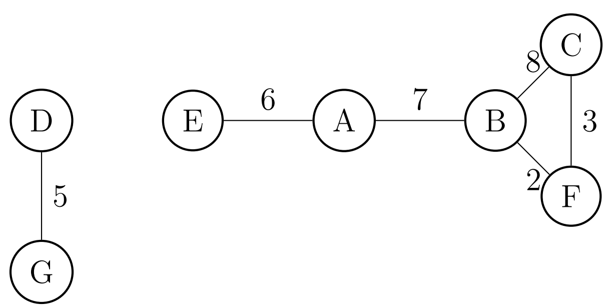
(c) There is indeed a cycle, between B, C, and F.

(d) A maximum length path is $(E, A), (A, B), (B, F), (F, C)$.

(e) We could add the edge (D, E).

(f) One connected component is $\{D, G\}$. Another one is $\{E, A, B, C, F\}$.

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?

Solution:

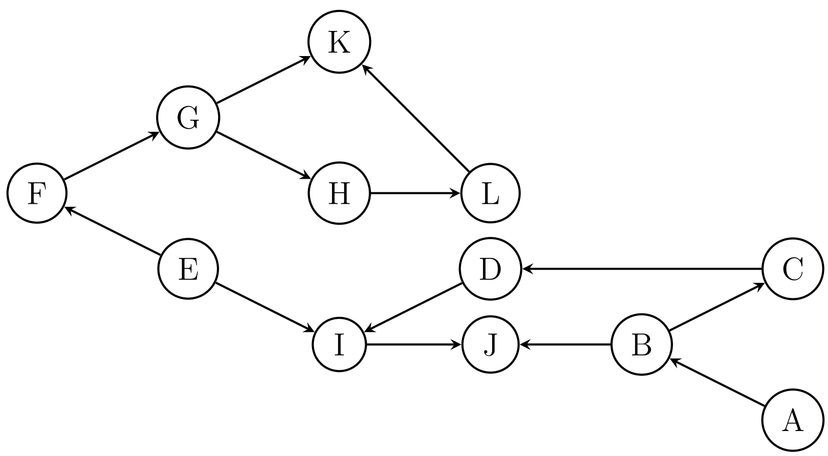
(a) The path with the least number of nodes is $(E, A), (A, B), (B, C)$. The cost is 21.

(b) The minimum cost path is actually $(E, A), (A, B), (B, F), (F, C)$. The cost is 18.

(c) The path with the shortest length is $(E, A), (A, B), (B, C)$. The length is 3.

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.

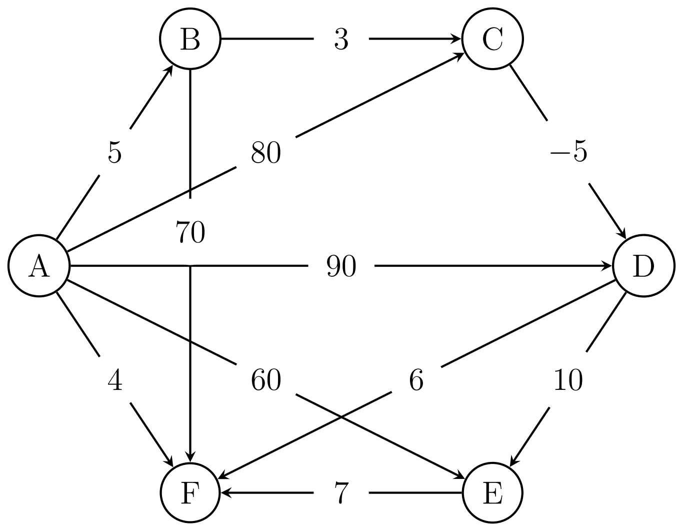
Solution: Yes, it does. This is a DAG (i.e., it has no cycles).

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

Solution: There are many. One example is e, f, g, h, l, k, a, b, c, d, i, j.

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.

Solution:

Vertex	Distance	Predecessor	Processed
A	0	–	Yes
B	5	A	Yes
C	~~80~~ 8	B	Yes
D	~~90~~ 3	C	Yes
E	~~60~~ 13	D	Yes
F	4	A	Yes

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

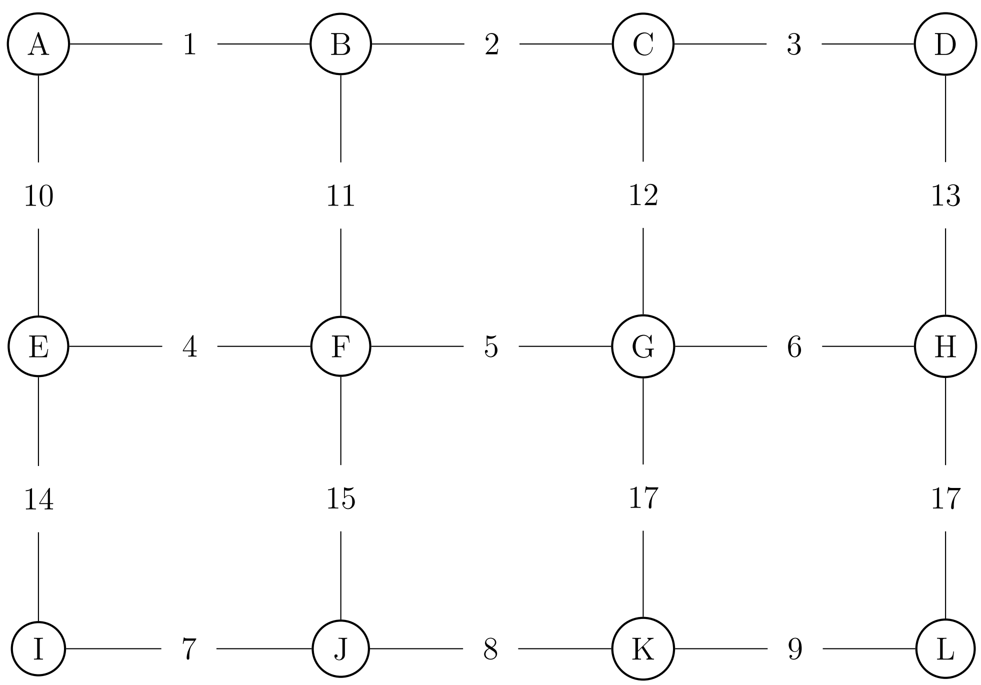
Solution: In this case, no. In general, Dijkstra's Algorithm does not necessarily work correctly with negative edge weights, but here, it actually returns the right result.

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

> **Solution:** Have each vertex remember its predecessor (A predecessor is the node we took an edge from to get to the node.) Or if you can't store data in nodes, keep an extra dictionary which maps nodes to their predecessors. Then, walk from the target vertex back toward the source vertex using the predecessor map.

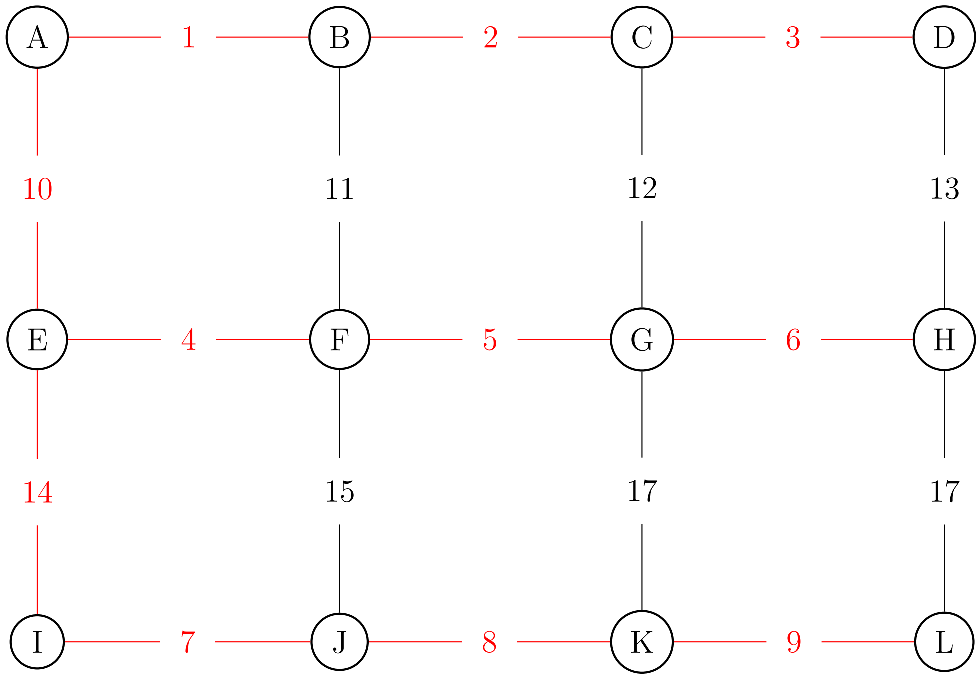
5 LMNST!

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.

Solution:



- Using Prim's algorithm:

Vertex	Distance	Best Edge	Processed
A	–	–	Yes
B	1	(A, B)	Yes
C	2	(B, C)	Yes
D	3	(C, D)	Yes
E	10	(A, B)	Yes
F	~~11~~ 4	~~(B, F)~~ (E, F)	Yes
G	~~12~~ 5	~~(C, G)~~ (F, G)	Yes
H	~~13~~ 6	~~(D, H)~~ (G, H)	Yes
I	14	(E, I)	Yes
J	~~15~~ 7	~~(F, J)~~ (I, J)	Yes
K	~~17~~ 8	~~(G, K)~~ (J, K)	Yes
L	~~17~~ 9	~~(H, L)~~ (K, L)	Yes

Solution:

- Using Kruskal's algorithm:

Edge	Include?	Reason
(A, B)	Yes	–
(B, C)	Yes	–
(C, D)	Yes	–
(E, F)	Yes	–
(F, G)	Yes	–
(G, H)	Yes	–
(I, J)	Yes	–
(J, K)	Yes	–
(K, L)	Yes	–
(A, E)	Yes	–
(B, F)	No	Cycle (A, B, F, E, A)
(C, G)	No	Cycle (A, B, C, G, F, E, A)
(D, H)	No	Cycle $(A, B, C, D, H, G, F, E, A)$
(E, I)	Yes	–
(F, J)	No	Cycle (E, F, J, I, E)
(G, K)	No	Cycle (E, F, G, K, J, I, E)
(H, L)	No	Cycle $(E, F, G, H, L, K, J, I, E)$

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?

Solution: A graph can only have multiple MSTs if it has multiple edges of the same weight. This graph has two 17's, but neither of them are used in the MST. So, there's only one MST here.

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

Solution: Prim's Algorithm takes $\mathcal{O}(|V| \lg |V| + |E| \lg |V|)$, and Kruskal's Algorithm takes $\mathcal{O}(|E| \lg |E|)$.

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.

Solution: We can represent this as an undirected, unweighted graph where each tile is a vertex. Edges connect tiles we can travel between. When we have a wormhole, we add an extra edge connecting that wormhole tile to the corresponding end of the wormhole.
Because it takes only one turn to travel to each adjacent tile, there is actually no need to store edge weights: it costs an equal amount to move to the next vertex.
All paths are bidirectional, so we can also use an undirected graph. (If there are paths or wormholes that are one-way, we can switch to using a directed graph).
To find the shortest path, we can run BFS starting with the player and stop the moment we hit a coin.
(We can use other algorithms like DFS or Dijkstra's algorithm if we're careful, but those would be less efficient.)

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. 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).

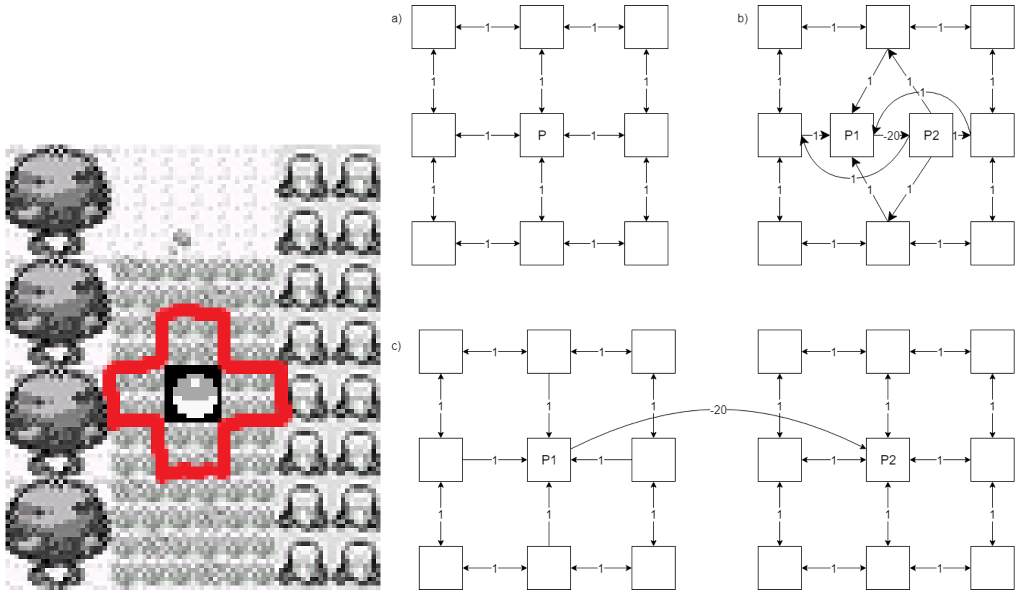
   Solution: Have a vertex for each possible location in Viridian Forest, and an edge between every two vertices we can move between in one step. Since Pikachu is losing one hit point for each step, we can just leave the graph unweighted.
   Since the graph is unweighted, we can just run BFS, starting from our current location, with a target of the edge of Viridian Forest.
   You could use either a directed graph or an undirected graph for this part.

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.

[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.



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?).

Solution: The potion edge is part of a cycle!
What happens if you go around the cycle repeatedly? Each time the distance you've gone gets "shorter!" So no matter how many times you've gone around the cycle you should go around once more, and you'll be able to find an even shorter walk from your current location to the edge of the forest. With Brock's representation, the shortest route to the edge of Viridian Forest isn't even defined!

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?

Solution: No, Dijkstra's isn't guaranteed to work when there are negative edges. Poor Brock. He knows so much about rock Pokemon but so little about algorithms. Luckily your Pokédex gets good data service in Viridian Forest, and you look up the Bellman-Ford algorithm for finding shortest paths with negative edge weights and find the new best path.

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.

Solution: We now want 4 copies of the graph. One for each of (no potions used, only potion 1 used, only potion 2 used, both potions used). Make edges of weight -20 to connect these in the same way as you did in the last part.
To make it easier to choose a final destination, add a dummy destination vertex. Then add a weight 0 edge from each copy of the edge of the forest to the dummy destination.