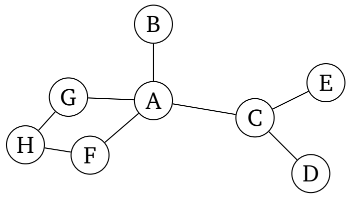
Section 07: Graphs and Graph Algorithms

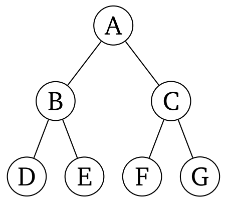
1. Graph traversal

(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:



2. 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?

3. Design Problem: Pathfinding in mazes

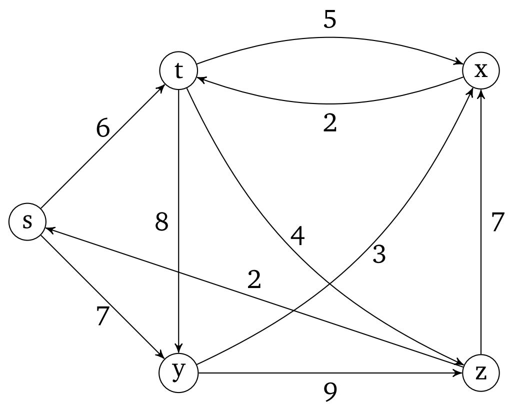
Suppose we are trying to design a maze within a 2d top-down video-game. The world is represented as a grid, where each tile is either an impassable wall, an open space a player can pass through, or a *wormhole*. On each turn, the player may move one space on the grid to any adjacent open tile. If the 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.

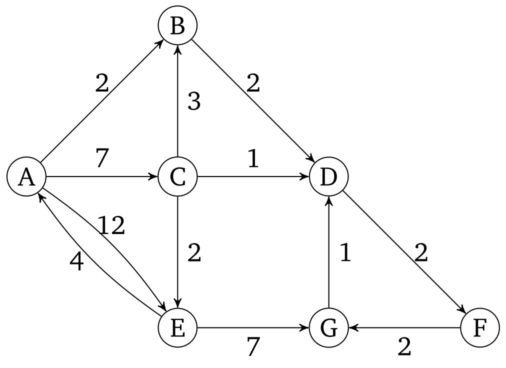
4. Simulating Dijkstra's

(a) Consider the following graph:



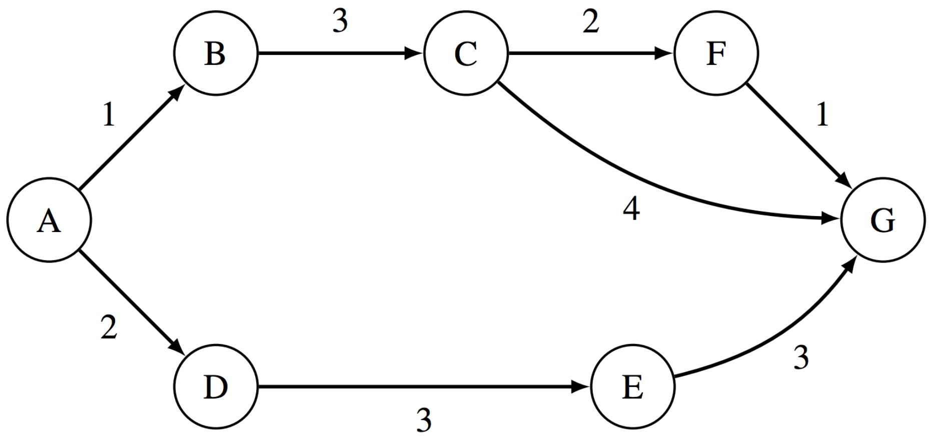
Suppose we run Dijkstra's algorithm on this graph starting with vertex s. What are the final costs of each vertex and the shortest paths from s to each vertex?

(b) Here is another graph. What are the final costs and shortest paths if we run Dijkstra's starting on node A?



5. A* Search

For the graph below, let `h(u, v)` be the value returned by the heuristic for any nodes `u` and `v`.



Heuristics
$h(A,G) = 8$
$h(B,G) = 6$
$h(C,G) = 5$
$h(F,G) = 1$
$h(D,G) = 6$
$h(E,G) = 3$

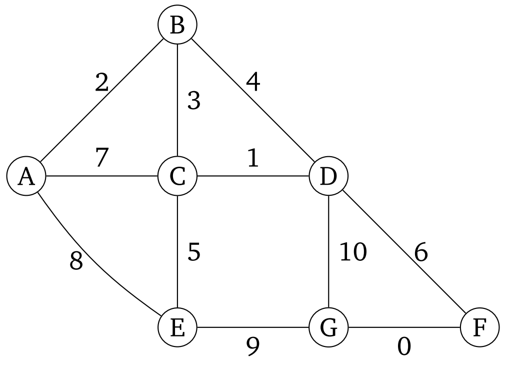
(a) Given the weights and heuristic values for the graph above, what path would A* search return, starting from A and with G as a goal?

Pseudocode

```
PQ = new PriorityQueue()
PQ.add(A, h(A))
PQ.add(v, infinity) # (all nodes except A).

distTo = {} # map
distTo[A] = 0
distTo[v] = infinity # (all nodes except A).

while (not PQ.isEmpty()):
    poppedNode, poppedPriority = PQ.pop()
    if (poppedNode == goal): terminate

    for child in poppedNode.children:
        if PQ.contains(child):
            potentialDist = distTo[poppedNode] + edgeWeight(poppedNode, child)

            if potentialDist < distTo[child]:
               distTo.put(child, potentialDist)
               PQ.changePriority(child, potentialDist + h(child))
```

6. Minimum spanning trees

Consider the following graph:



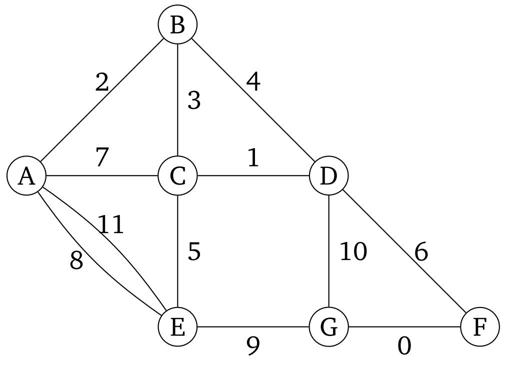
(a) What happens if we run Prim's algorithm starting on node A? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(b) What happens if we run Prim's algorithm starting on node E? What are the final cost and edges selected? Give the set of edges in the resulting MST.

(c) What happens if we run Prim's algorithm starting on *any* node? What are the final costs and edges selected? Give the set of edges in the resulting MST.

(d) What happens if we run Kruskal's algorithm? Give the set of edges in the resulting MST.

(e) Suppose we modify the graph above and add a heavier parallel edge between A and E, which would result in the graph shown below. Would your answers for above subparts (a, b, c, and d) be the same for this following graph as well?



7. More MSTs

Answer each of these true/false questions about minimum spanning trees.

(a) A MST contains a cycle.

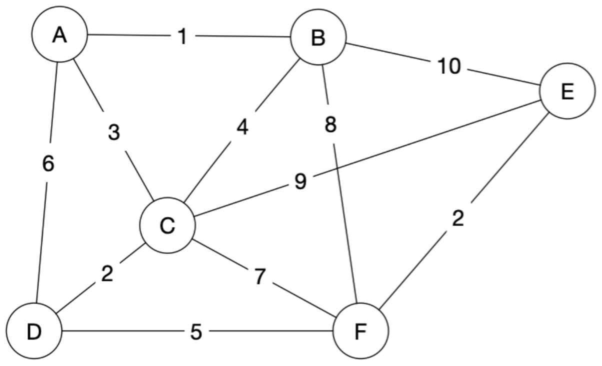
(b) If we remove an edge from a MST, the resulting subgraph is still a MST.

(c) If we add an edge to a MST, the resulting subgraph is still a MST.

(d) If there are V vertices in a given graph, a MST of that graph contains $|V| - 1$ edges.

Answer these questions about Kruskal's algorithm.

(a) Execute Kruskal's algorithm on the following graph. Fill the table.



Step	Components	Edge	Include?
1	{A} {B} {C} {D} {E} {F}	A, B	Yes
2	{A, B} {C} {D} {E} {F}	D, C	
3			
4			
5			
6			
7			
8			
9			
10			
11			

(b) In this graph there are 6 vertices and 11 edges, and the for loop in the code for Kruskal's runs 11 times, a few more times after the MST is found. How would you optimize the pseudocode so the for loop terminates early, as soon as a valid MST is found.

8. Graph Modeling 1: DJ Kistra

You've just landed your first big disk jockeying job as "DJ Kistra."

During your show you're playing "Shake It Off," and decide you want to slow things down with "Wildest Dreams." But you know that if you play two songs whose tempos differ by more than 10 beats per minute or if you play only a portion of a song, that the crowd will be very disappointed. Instead you'll need to find a list of songs to play to gradually get you to "Wildest Dreams." Your goal is to transition to "Wildest Dreams" as quickly as possible (in terms of seconds).

You have a list of all the songs you can play, their speeds in beats per minute, and the length of the songs in seconds.

(a) Describe a graph you could construct to help you solve the problem. At the very least you'll want to mention what the vertices and edges are, and whether the edges are weighted or unweighted and directed or undirected.

(b) Write pseudocode to construct your graph from the previous part. You may assume your songs are stored in whatever data structure makes this part easiest. Assume you have access to a method `makeEdge(v1, v2, w)` which creates an edge from `v1` to `v2` of weight `w`.

(c) Describe an algorithm you could run on the graph you just constructed to find the list of songs you can play to get to "Wildest Dreams" the fastest without disappointing the crowd.

(d) What is the running time of your plan to find the list of songs? You should include the time it would take to construct your graph and to find the list of songs. Give a simplified big-O running time in terms of whatever variables you need.

9. Graph Modeling 2: Snow Day

After 4 snow days last year, UW has decided to improve its snow response plan. Instead of doing "late start" days, they want an "extended passing period" plan. The goal is to clear enough sidewalks that everyone can get from every classroom to every other **eventually** but not necessarily very quickly.

Unfortunately, UW has access to only one snowplow. Your goal is to determine which sidewalks to plow and whether it can be done in time for the first 8:30 AM lectures.

You have a map of campus, with each sidewalk labeled with the time it will take to plow to clear it.

(a) Describe a graph that would help you solve this problem. You will probably want to mention at least what the vertices and edges are, whether the edges are weighted or unweighted, and directed or undirected.

(b) What algorithm would you run on the graph to figure out which sidewalks to plow? Explain why the output of your algorithm will be able to produce a "extended passing period" plowing plan.

(c) How can you tell whether the plow can actually clear all the sidewalks in time?

10. Design Problem: Negative edge weights

If you enjoy reading Pokémon jokes, you can read the flavor text to understand where the graph problem comes from. Otherwise, you can skip those parts and just read the formal statements.

Flavor Text You and your trusty Pikachu have made it halfway through Viridian Forest, but things have taken a turn for the worst. 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]

Formal Statement In a video game you are playing, each step you take costs a character (Pikachu) one unit of health. You have a map of the level (Viridian Forest) – your goal is to reach the end of the level (marked on your map) while losing as little health as possible.

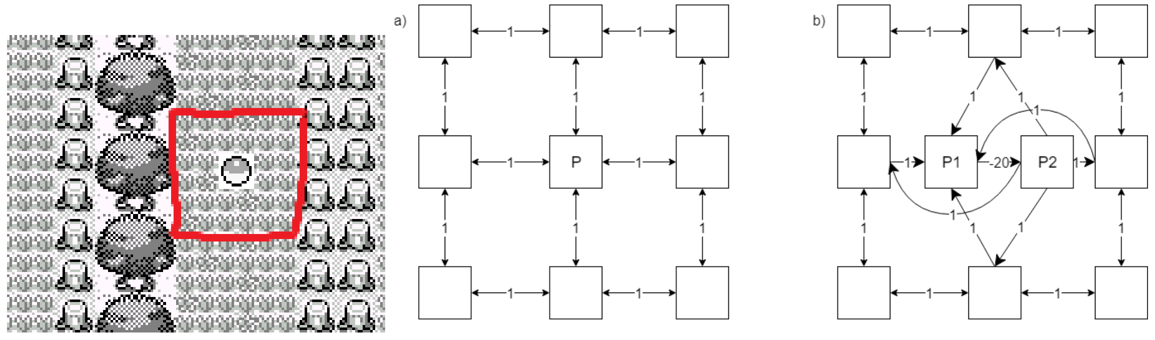
(a) Describe a graph and an algorithm run on that graph to find the path through the forest to save as many of Pikachu's hit points as possible (i.e. the path with the fewest number of steps).

(b) **Flavor Text** 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.

Formal Statement You realize your character doesn't have enough health to make it to the edge of the forest. But you know there is a healing item (a "potion") somewhere in the forest, that will give you back 20 units of health.

A friend (Brock) suggests the following update: add a dummy vertex to the graph where the healing item is and connect up the new vertex with a (directed) edge of length -20, to represent undoing the loss of 20 hit points.

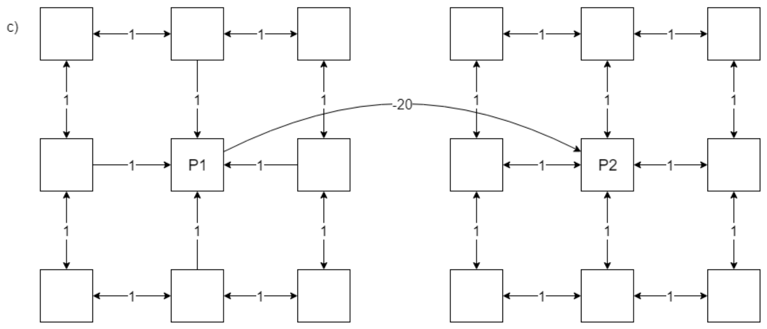


9 spots in Viridian Forest, the corresponding vertices before Brock's transformation and the same vertices after the transformation.

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?).

[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.

(c) 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). His new graph looks something like this.



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

(d) **Challenge Problem** Misty says she knows about a second potion somewhere else in the forest. Describe how to modify the graph to handle both of the potions.