DSC 40B Discussion 9 Worksheet

Friday, May 29, 2026

Imagine you are a tutor, and you need to explain something to someone who has only a vague understanding of what's going on. Do your best to get the point across. **ALTERNATE**.

If you're not sure, ask your peer if they know how to explain it, then REPEAT the explanation back. Remember, the point of the exercise is to **vocalize** your thoughts.

If you are both unsure, make sure to come to/tutor for help! Do not use ChatGPT or any other LLM! The point of discussion is to prepare you for the exams.

Have one person in the pair open the Discussion 1 Gradescope assignment. After both you **and** your partner finish a problem **and explain it**, answer the corresponding multiple choice/short answer questions for both versions.. You will receive instant feedback on the questions, whether you are correct or not. If your answer is incorrect, please go back to the question to understand it and feel free to ask tutors about anything you are confused about. Once you have finished the entire worksheet or discussion is ending, turn in your assignment and add your partner to the submission.

Quick Icebreaker

Introduce yourself to your partner by sharing name, year, major, etc. What are your summer plans? (Do not spend more than 2 minutes!)

Questions

Question 1:

You are starting your fitness journey and decide to start with hiking trails.

First Person:

You decide that distance hiked is the most important statistic to keep track of. Due to your extensive knowledge in graph theory, you decide to model your local hiking trails as a weighted graph.

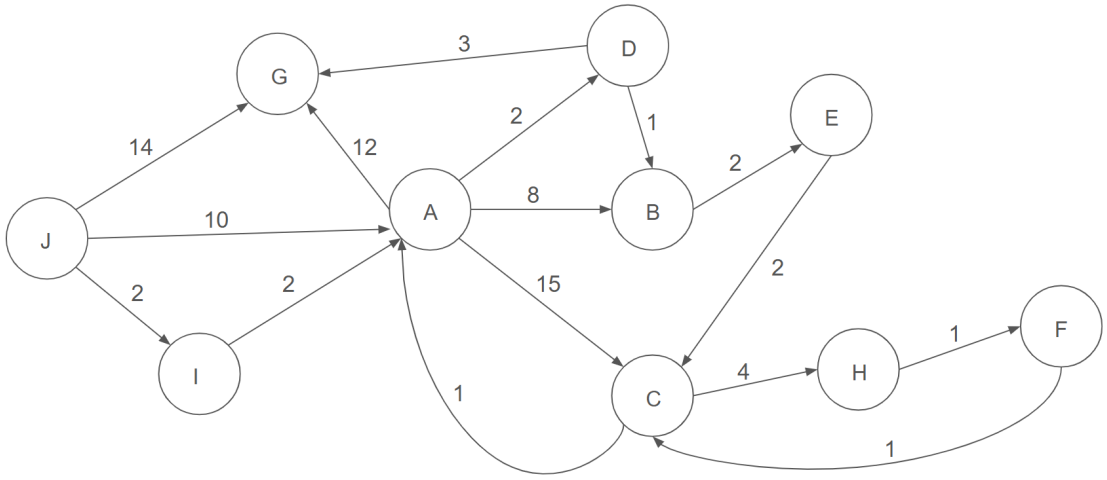
What do the nodes, edges, and edge weights in your graph represent? Do you have negative edge weights?

Second Person:

Your friend notes that distance is not the only factor, as elevation plays a role in how hard the hike really is. You decide to model a separate weighted graph to keep track of your elevation.

What do your nodes, edges, and edge weights represent? Do you have negative edge weights?

Question 2:



Second Person:

Given the graph above, run Dijkstra's from the source of **node J**. Give the resulting distance and predecessor of each node. Use the convention that `graph.neighbors()` returns successors in alphabetical order.
Show your work and explain your reasoning.

First Person:

Given the graph above, run Dijkstra's from the source of **node A**. Give the resulting distance and predecessor of each node. Use the convention that `graph.neighbors()` returns successors in alphabetical order.
Show your work and explain your reasoning.

Question 3:

First Person:

Given the graph in Question 2, run BFS from **node A**. Give the resulting distance and predecessor of each node. Use the convention that `graph.neighbors()` returns successors in alphabetical order.

How many nodes have the incorrect distance and predecessor? Show your work and explain your reasoning.

Second Person:

Given the graph in Question 2, run BFS from **node J**. Give the resulting distance and predecessor of each node. Use the convention that `graph.neighbors()` returns successors in alphabetical order.

How many nodes have the incorrect distance and predecessor? Show your work and explain your reasoning.-

Question 4:

Second Person:

True or False: Dijkstra's algorithm can still return incorrect shortest paths even if there is only one negative edge and no negative cycle.

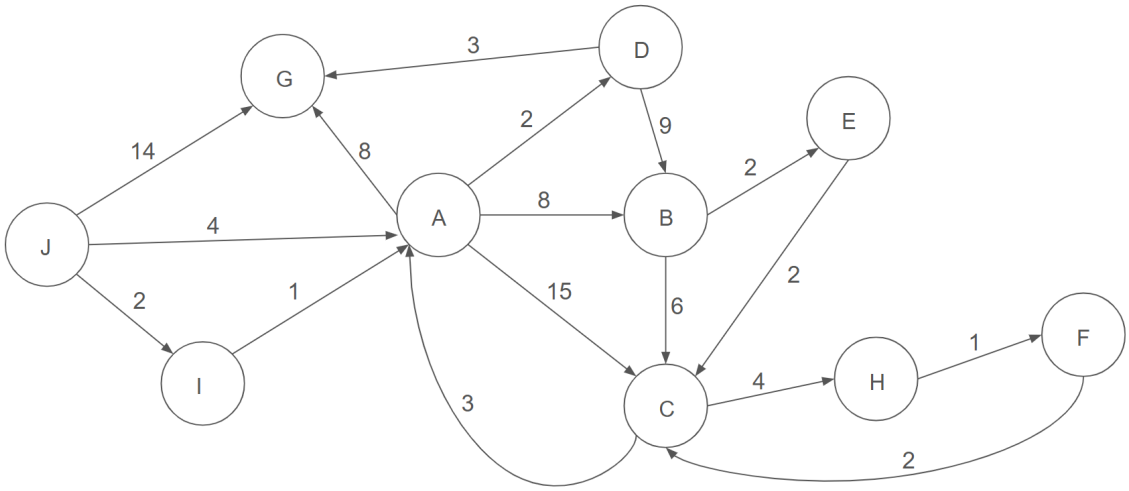
- ☐ True
- ☐ False

First Person:

True or False: Dijkstra's algorithm can still return all the correct shortest paths even if there are more than one negative edge..

- ☐ True
- ☐ False

Question 5:



First Person:

Consider the updated graph above. Running Dijsktra's with a source of **node A**, what is the most amount of times a node's distance is updated? Which node is this? Use the convention that `graph.neighbors()` returns successors in alphabetical order. Show you work and explain your reasoning.

Second Person:

Consider the updated graph above. Running Dijsktra's with a source of **node J**, what is the most amount of times a node's distance is updated? Which node is this? Use the convention that `graph.neighbors()` returns successors in alphabetical order. Show you work and explain your reasoning.

Question 6:

Second Person:

Given a sparsely connected graph, where each node has at most a degree of 2, what is the Big O runtime of Dijkstra's Algorithm in terms of $|V|$?

First Person:

Given a fully connected graph, where each node is connected to every other node, what is the Big O runtime of Dijkstra's Algorithm in terms of $|V|$?

Question 7:

First Person:

```
def dijkstra(graph, weights, source):
    est = {node: float('inf') for node in graph.nodes}
    est[source] = 0

    pred = {node: None for node in graph.nodes}

    priority_queue = PriorityQueue(est)

    while priority_queue:
        u = priority_queue.extract_min()

        for v in graph.neighbors(u):
            changed = update(u, v, weights, est, pred)

          # Added loop
          for x in graph.nodes:
              print("I love DSC40B!")

          if changed:
              priority_queue.change_priority(v, est[v])

    return est, pred
```

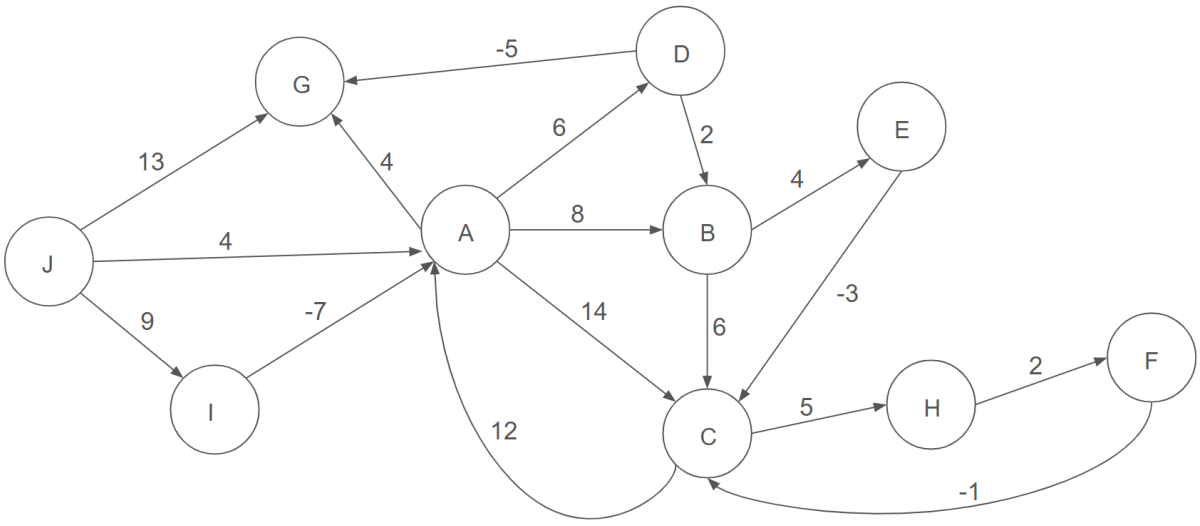
Given the modified version of Dijsktra's above, what is the new Big O runtime?

Second Person:

```
def dijkstra(graph, weights, source):
    est = {node: float('inf') for node in graph.nodes}
    est[source] = 0

    pred = {node: None for node in graph.nodes}

    priority_queue = PriorityQueue(est)

    while priority_queue:
        u = priority_queue.extract_min()

        # Added loop
        for x in graph.nodes:
            print("I love DSC40B!")

        for v in graph.neighbors(u):
            changed = update(u, v, weights, est, pred)
            if changed:
                priority_queue.change_priority(v, est[v])

    return est, pred
```

Given the modified version of Dijsktra's above, what is the new Big O runtime?

Question 8:



Second Person:

Given the modified graph above with negative weights, run Dijkstra's with a source of **node J**. How many nodes have an incorrect distance assigned? Show your work and explain your reasoning.

FirstPerson:

Given the modified graph above with negative weights, run Dijkstra's with a source of **node A**. How many nodes have an incorrect distance assigned? Show your work and explain your reasoning.

Question 9:

First Person:

Suppose we run Bellman Ford on a directed graph where every vertex has degree 4. What is the time complexity of the algorithm? (Here V is the number of vertices in the graph).

- ☐ A. O(V)
- ☐ B. O(V log V)
- ☐ C. O(V^2)
- ☐ D. O(V^2 log V)
- ☐ E. O(V^3)

Second Person:

Suppose we run Bellman Ford on a complete directed graph, i.e. every pair of distinct vertices is connected by a pair of unique edges. What is the time complexity of the algorithm? (Here V is the number of vertices in the graph).

- ☐ A. O(V)
- ☐ B. O(V log V)
- ☐ C. O(V^2)
- ☐ D. O(V^2 log V)
- ☐ E. O(V^3)

Question 10:

Second Person:

Given the graph in Question 8, run Bellman Ford with a source of **node J**. How many total times is a node's distance and predecessor updated? Do not count setting the source as a distance of 0 and all other nodes at ∞. Use the convention that `graph.edges` returns the edges in alphabetical order (A → B is first, then A → C, with J → I being last). Show your work and explain your reasoning.

First Person:

Given the graph in Question 8, run Bellman Ford with a source of **node A**. How many total times is a node's distance and predecessor updated? Do not count setting the source as a distance of 0 and all other nodes at ∞. Use the convention that `graph.edges` returns the edges in alphabetical order (A → B is first, then A → C, with J → I being last). Show your work and explain your reasoning.

Question 11:

First Person:

Given a fully connected graph with $|V|$ vertices, you run Bellman Ford with early stopping and negative cycle detection. What is the minimum and maximum number of iterations through `graph.edges` your algorithm can do, considering the fact that you do not know the order the edges are returned? Show your work and explain your reasoning.

Second Person:

Given a sparsely connected graph with $|V|$ vertices, where each vertex has at most a degree of two, you run Bellman Ford with early stopping and negative cycle detection. What is the minimum and maximum number of iterations through `graph.edges` your algorithm can do, considering the fact that you do not know the order the edges are returned? Show your work and explain your reasoning.