Nets hw 2

juliaguo

February 2026

1 Part 1

Question 1

This is true:

Consider when the DFS algorithm discovers node A:

Case 1: B already discovered before A

Then

$$start(B) < start(A) < finish(A),$$

which means that

$$start(B) < finish(A)$$

Case 2: B is still undiscovered when A is discovered

Since there exists a directed path from A to B, node B is reachable from A. DFS explores all reachable undiscovered vertices before going back to A. Therefore,

$$start(A) < start(B) < finish(A),$$

which again means that

$$start(B) < finish(A)$$

Since in both cases, $start(B) < finish(A)$ holds, this is true.

Question 2

At each step, choose the smallest node with in-degree 0.

Firstly, the two nodes with in-degree 0 are 0 and 1. Since $0 < 1$, we remove 0 and update the in-degree of its neighbors:

$$in - deg(4) : 2 \to 1, \qquad in - deg(2) : 2 \to 1$$

1 still has 0 in-degs so remove that and update the in-deg of its neighbors.

$$in - deg(4) : 1 \to 0, \quad in - deg(5) : 1 \to 0$$

$4 < 5$ so remove 4 and update its neighbor's in-degs.

$$in - deg(7) : 1 \to 0$$

$5 < 7$ so remove 5 and update in-degs.

$$in - deg(6) : 2 \to 1$$

Remove 7 and update in-deg.

$$in - deg(6) : 1 \to 0$$

Remove 6 and update in-deg.

$$in - deg(2) : 1 \to 0$$

Remove 2 and update in-deg.

$$in - deg(3) : 1 \to 0$$

Remove 3

Therefore, the final topological sort ordering considering the modification listed in the instructions is:

$$0,\ 1,\ 4,\ 5,\ 7,\ 6,\ 2,\ 3$$

Question 3

Initialize:

$$d(A) = 0, \quad d(B) = d(C) = d(D) = d(E) = d(F) = \infty$$

Relax edges from A:

$$d(E) = 2, \quad d(F) = 3, \quad d(C) = 8$$

Process E (2):

$$d(C) = \min(8,\, 2 + 3) = 5$$

Process F (3):

$$d(D) = 3 + 1 = 4$$

Process D (4):

$$d(B) = 4 + 6 = 10$$

Further relaxations do not improve distances.

Final shortest path costs from A:

$$
\begin{aligned}
d(A) &= 0 \\
d(E) &= 2 \\
d(F) &= 3 \\
d(D) &= 4 \\
d(C) &= 5 \\
d(B) &= 10
\end{aligned}
$$

Paths:

$$
\begin{aligned}
&E : A \to E \\
&F : A \to F \\
&D : A \to F \to D \\
&C : A \to E \to C \\
&B : A \to F \to D \to B
\end{aligned}
$$

Question 4

For the next two examples, I'll refer to the directed graph shown in Question 3.

Part a: Incorrect Answer

Suppose the edge $C \to D$ is changed from 4 to -4.

Originally, Dijkstra finalizes D via the path

$$A \to F \to D$$

with cost

$$3 + 1 = 4$$

However, after processing node C, we would obtain a new path

$$A \to E \to C \to D$$

with cost

$$2 + 3 + (-4) = 1$$

This is strictly smaller than 4. However, Dijkstra's algorithm would already have finalized D before processing C, since $d(D) = 4$ and $d(C) = 5$ at that stage. Because Dijkstra does not revisit finalized nodes, it would incorrectly keep $d(D) = 4$ instead of updating it to 1.

Part b: Correct Answer

If we change the edge weight from $A \to E = 2$ to $A \to E = -2$, then the path

$$A \to E \to C$$

has total cost

$$-2 + 3 = 1$$

The direct edge $A \to C$ has weight 8, so the shortest path from A to C remains

$$A \to E \to C$$

In this case, Dijkstra's algorithm still produces the correct answer. The negative edge comes from the source node A, and no path discovered after lowers the distance of a node that has already been finalized. Therefore, the final path remains the correct answer.

Part c: Explanation

Essentially, Dijkstra's algorithm will assume that the shortest distance to a vertex is finalized after the node is picked. If all weights are positive, there is no possibility for a shorter path to be found. However, negative weights can affect the shortest path by decreasing it. This goes against the logic of Dijkstra's algorithm. However, in the case that there are negative weights but they don't change the shortest distance, that's when we see results similar to that of part b because it wasn't small enough to change the overall result. Thus, sometimes negative weights can still give us the correct result for Dijkstra's algorithm.

Question 5: In the social network depicted in Figure 3.23 with each edge labeled as either a strong or weak tie, which two nodes violate the Strong Triadic Closure Property? Provide an explanation for your answer

Under the Strong Triadic Closure Property we know that if a node has strong ties to two other nodes, then those two nodes that it's strongly connected to must be connected by either a weak or strong connection.

The first violation occurs with Node C because it has strong connections with both Nodes B and E. However, there is no connection/edge between Nodes B and E. Thus, this is a violation of the Strong Triadic Closure Property.

The second violation occurs with Node E because it has strong connections with both Nodes D and C. However, there is no connection/edge between Nodes D and C. Thus, this is another violation of the Strong Triadic Closure Property.

Question 6

The clustering coefficient is 0.5 because it's 3/6.

The neighborhood overlap of the A-B edge is 2/3.