COMP 3804: Assignment 3

Fall 2022

Solution given by Heny Dave

School of Computer Science Carleton University

Due Date: Sunday, November 20^{th} at 11:59PM

********** Note that we extended the deadline for this assignment to the 21st **********

Your assignment should be submitted online on Brightspace as a single PDF file; the filename includes your studentID. No late assignments will be accepted. You can type your assignment or you can upload a scanned copy of it. Please use a good image capturing device. Make sure that your upload is clearly readable. If it is difficult to read, it will not be graded!

Question 1 [20 marks]

Show how Bellman-Ford's and Dijkstra's algorithms would compute the shortest paths from *A* to all vertices of the graph shown in Figure 2. Illustrate the algorithms as shown in class: Bellman-Ford via a sequence of graphs (see slide 22). Dijkstra's (see slide 36) by filling in a table with step numbers as row index, vertices as column index, and d[], [] values as entries. Circle a d[]-value when it is finalized, i.e., when it become [] .

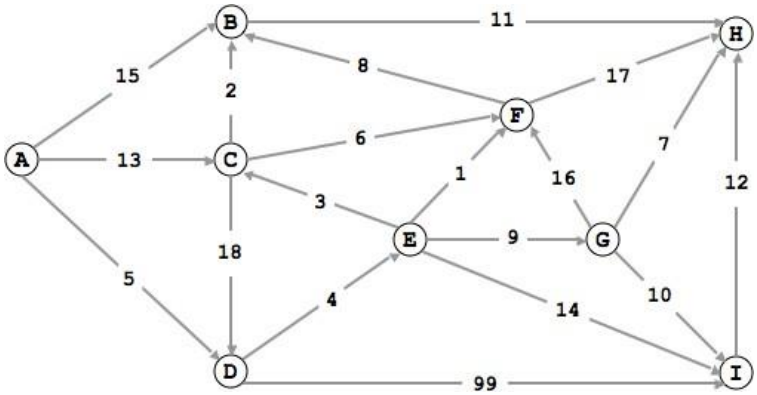


Figure 1: The input graph for Question 1.

We will order the edges lexicographically as: AB, AC, AD, BC, BF, BH, CD, CE, CF, DE, DI, EF, EG, EI, FG, FH, GH, GI, HI. Execution of the Bellman-Ford algorithm is shown in Table 1. No further changes occur after Step 3. Different edge orders would result in different tables which may have changes over more of the iterations. Execution of Dijkstra's algorithm is shown in Table 2.

	Init	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8
A	0/nil	0/nil	0/nil	0/nil	0/nil	0/nil	0/nil	0/nil	0/nil
B	∞/nil	15/A	15/A	14/C	14/C	14/C	14/C	14/C	14/C
C	∞/nil	13/A	12/E	12/E	12/E	12/E	12/E	12/E	12/E
D	∞/nil	5/A	5/A	5/A	5/A	5/A	5/A	5/A	5/A
E	∞/nil	9/D	9/D	9/D	9/D	9/D	9/D	9/D	9/D
F	∞/nil	10/E	10/E	10/E	10/E	10/E	10/E	10/E	10/E
G	∞/nil	18/E	18/E	18/E	18/E	18/E	18/E	18/E	18/E
H	∞/nil	25/G	25/G	25/G	25/G	25/G	25/G	25/G	25/G
I	∞/nil	23/E	23/E	23/E	23/E	23/E	23/E	23/E	23/E

Table 1: Execution of the Bellman-Ford algorithm

⟨step#⟩	set S	A	B	C	D	E	F	G	H	I
Init.	{}	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
1	{A}		15/A	13/A	5/A	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
2	{A,D}		15/A	13/A		9/D	∞/nil	∞/nil	∞/nil	104/D
3	{A,D,E}		15/A	12/E			10/E	18/E	∞/nil	23/E
4	{A,D,E,F}		15/A	12/E				18/E	27/F	23/E
5	{A,C,D,E,F}		14/C					18/E	27/F	23/E
6	{A,B,C,D,E,F}							18/E	25/B	23/E
7	{A,B,C,D,E,F,G}								25/B	23/E
8	{A,B,C,D,E,F,G,I}								25/B	
9	{A,B,C,D,E,F,G,H,I}									

Table 2: Execution of Dijkstra's algorithm

Question 2 [20 marks]

Assume that we are given an undirected graph G=(V,E). Consider that Dijkstra's algorithm founda shortest path in G, called SP, between two nodes A and X of V. Is it true or false that if we reverse the nodes on SP, we get a shortest path from X to A? Prove or disprove.

Proof by Contradiction:

Lets assume that there exists a shortest path from X to A which is SP2 and the one from A to X be SP1. This means that SP1 is not equal to SP2.

Now, as Dijkstra computes shortest path SP1 from A to X and moreover, by reversing the nodes between AX or XA the shortest distance should remain the same because of the property of undirected graph which states that the weight of the edge remains the same in any direction (as it is undirected graph). This means that as all edges between AX and XA will cost the same thus SP1 = SP2

Hence, the reversing the nodes of the shortest path SP1 obtained from A to X by Dijksta's is the same shortest path for X to A.

Question 3 [10 marks]

1. List three different topological orderings for the enclosed graph.

    1. 2,5,4,1,3,6,7,9,8
    2. 5,4,1,3,2,6,7,9,8
    3. 5,2,1,4,3,6,7,9,8

2. Draw a (connected, directed) graph for which no topological sorting order exists.
   Any connected and directed graph having a cycle can not have a topological sorting order as shown in Figure 1

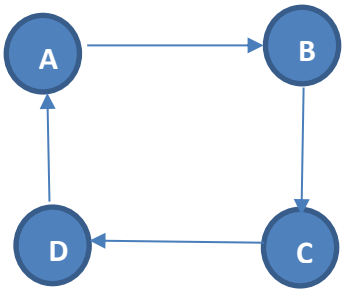


Figure 1: Cyclic graph having no topological sorting order

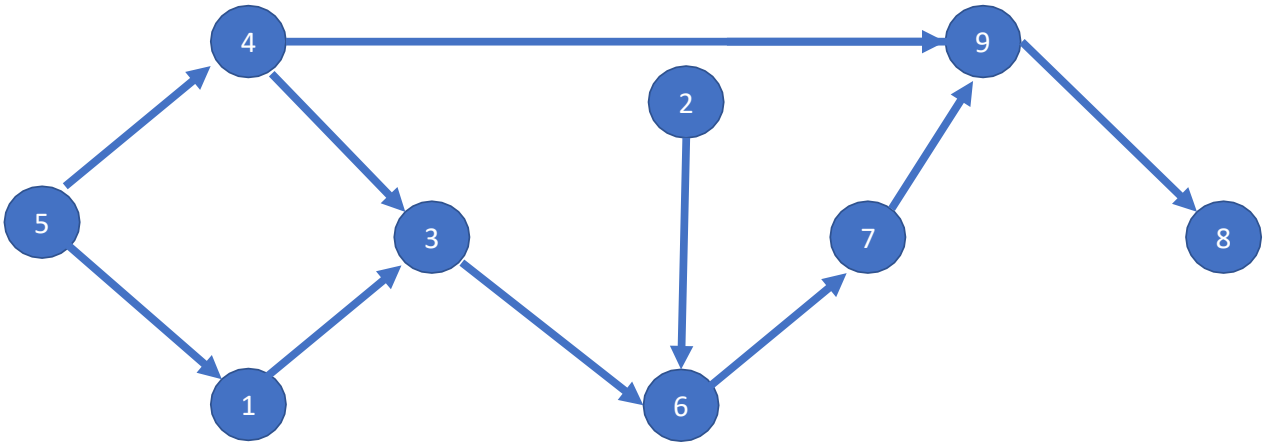


Figure 2: The input graph for Question 3.

Question 4 [10 marks]

Can Dijkstra's algorithm be used to determine if a graph is connected? How does that compareto algorithms for graph connectivity that you may know from previous classes or that you read up (and cite)? Just list the algorithm used and compare the time complexity.

Dijkstra's Algorithm can be used to determine if a graph is connected or not. As Dijkstra's algorithm is used to calculate the shortest path, using the similar algorithm we can explore over all possible nodes and check if their initial values are updated or not. The graph is disconnected if a particular node cannot be reached and thus, leaving its weight to be infinite. Therefore, Dijkstra's algorithm can be used to determine whether a graph is connected.

Another algorithm used to fund connectivity is depth first search. The time complexity of depth first search is O(|V| + |E|). The running time of Dijkstra's Algorithm is O((|V|+ |E|)log |V|) and O(|V| log |V|+ |E|) using Fibonacci heaps. Therefore, the Depth first search has a better time complexity and is faster than Dijkstra's

Question 5 [10 marks]

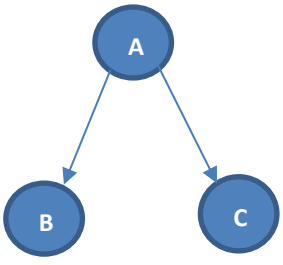
We said that the input to PRIMs algorithm is a connected, undirected graph. Suppose by accident, we give an input graph that is not connected. What would happen?

PRIMs algorithm on a disconnected graph will result in a MST forest which is basically a MST for each graph component. As the graph is disconnected only the cost of the first node of the very first component will be 0 and all other source nodes for subsequent components will have code infinity. Thus, making the overall total weight of the MST to be infinity.

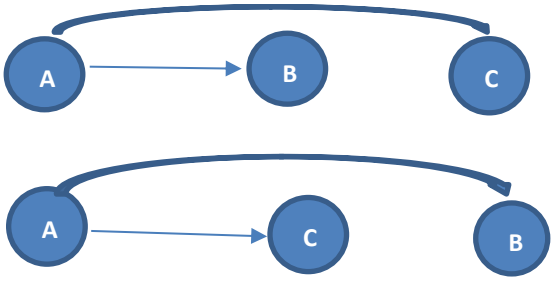
Question 6 [12 marks]

Suppose that we are considering a directed, acyclic graph G. A *Hamiltonian path* is a path in G that visits each vertex of G exactly once. Considering the below four statements, when does topological sorting of G yield a unique solution? Draw a counter-example and argue for each of the four statements in case it is incorrect below is incorrect and prove its correctness, otherwise.

1. When a single vertex exists in G with indegree 0.



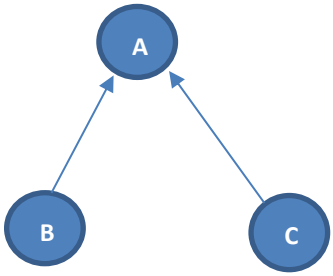
Graph



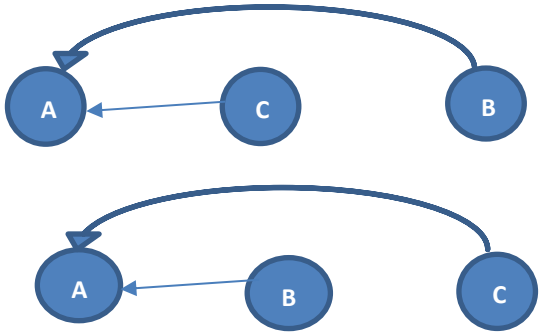
Topological sorting

Topological soritng is not unique

2. When several vertices have indegree 0.



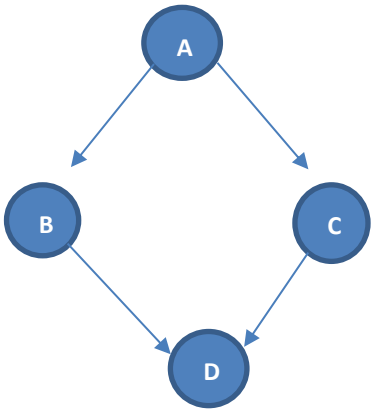
Graph



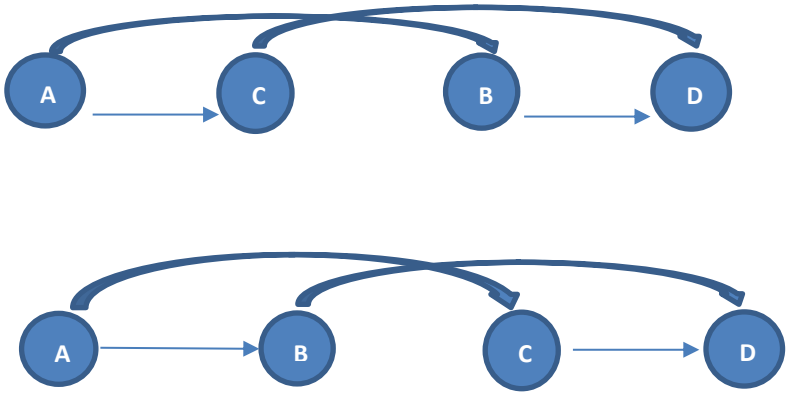
Topological sorting

Topological sorting is not unique

3. When a single vertex exists in G with outdegree 0.



Graph



Topological Sorting

Topological Sorting not unique

4. When there exists a Hamiltonian path in G.

The existence of Hamiltonian path in a directed acyclic graph will result in a unique topological sorting. As in case of a DAG without a Hamiltonian path there might be two consecutive vertices without being connected by an edge, and thus, these two vertices can generate multiple topological orders as the two vertices can be used interchangeably.