Econ 381

Homework Assignment 4

Q 1.1: Explain what a minimum spanning tree (MST)?

A: **Minimum Spanning Tree (MST)** of a connected, undirected, and weighted graph is a subset of the edges that:

1. Connects all the vertices together (spans all vertices).
2. Contains no cycles.
3. Has the minimum possible total edge weight.

Q 1.2: Then create an MST for the above graph. You can use Kruskal's algorithm or any similar algorithm.

A: **Kruskal's Algorithm for MST:**

Steps:

1. **Sort all edges** in non-decreasing order of their weights.
2. **Pick the smallest edge**. If it doesn't form a cycle, include it in the MST.
3. **Repeat** until all vertices are connected (i.e., the MST has V−1 edges, where V is the number of vertices).

Edges with Weights:

- (C, E) = 1
- (C, B) = 3
- (D, F) = 3
- (D, E) = 4
- (A, B) = 4
- (D, B) = 6
- (D, A) = 7
- (E, F) = 9

Step 1: Sorting

- (C, E) = 1
- (C, B) = 3
- (D, F) = 3
- (D, E) = 4
- (A, B) = 4
- (D, B) = 6

- (D, A) = 7
- (E, F) = 9

Step 2: Build the MST = 5 (V – 1 = 6 - 1 = 5)

1. (C, E) = 1
2. (C, B) = 3
3. (D, F) = 3
4. (D, E) = 4
5. (A, B) = 4

Total Weight of MST: 1 + 3 + 3 + 4 + 4 = **15**

Q 2: We need to know if the MST you created is unique or not. Do some research on the conditions for uniqueness of a generated MST. Is the above graph suitable for guaranteed uniqueness?

A: What is “MST”?

A unique MST occurs when there is only one possible MST for a given graph. This uniqueness is guaranteed if all the edge weights in the graph are distinct. In other words, if no two edges have the same weight, the MST will be unique. This is because the algorithm used to find the MST will always make the same choices when selecting edges, leading to a single, unique MST.

07-mst.pdf

According to these results, the MST that is currently formed is a completely unique MST.

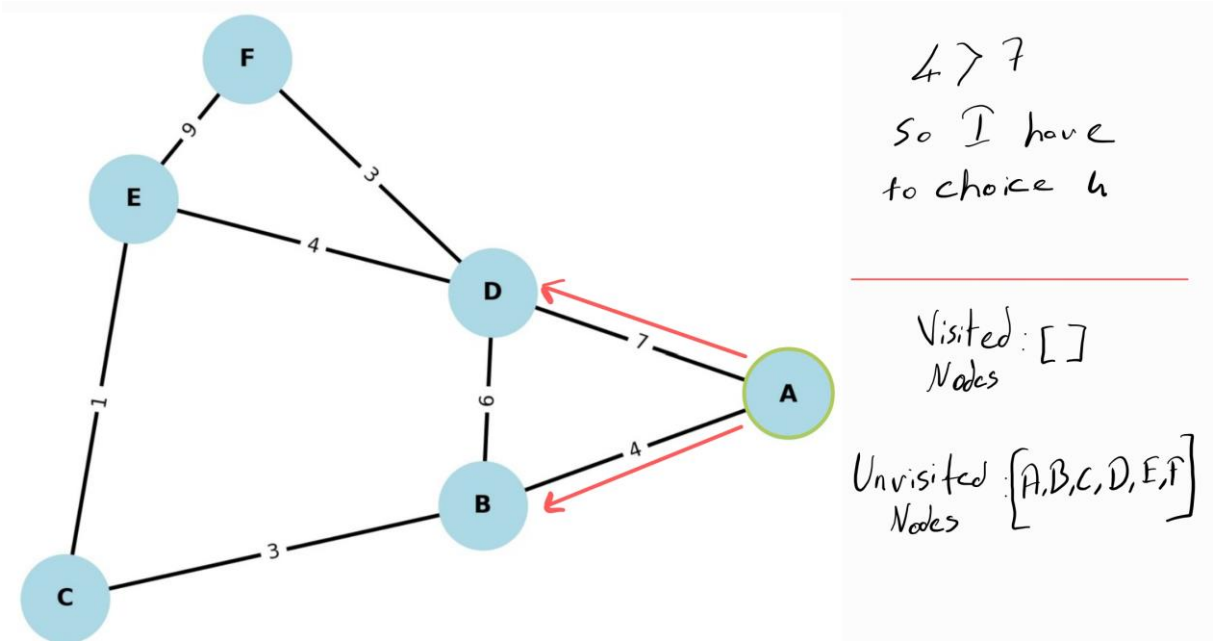
Q 3: Calculate the shortest paths from node A to all other nodes using Dijkstra's algorithm. Show all your steps. This may take some space.

A: What is “Dijkstra's algorithm”?

Dijkstra's algorithm is a classic algorithm used to find the shortest paths between nodes in a weighted graph. The algorithm works by repeatedly selecting the unvisited vertex with the smallest known distance from the start vertex, then updating the distances to its neighboring vertices.

These are the steps our problem

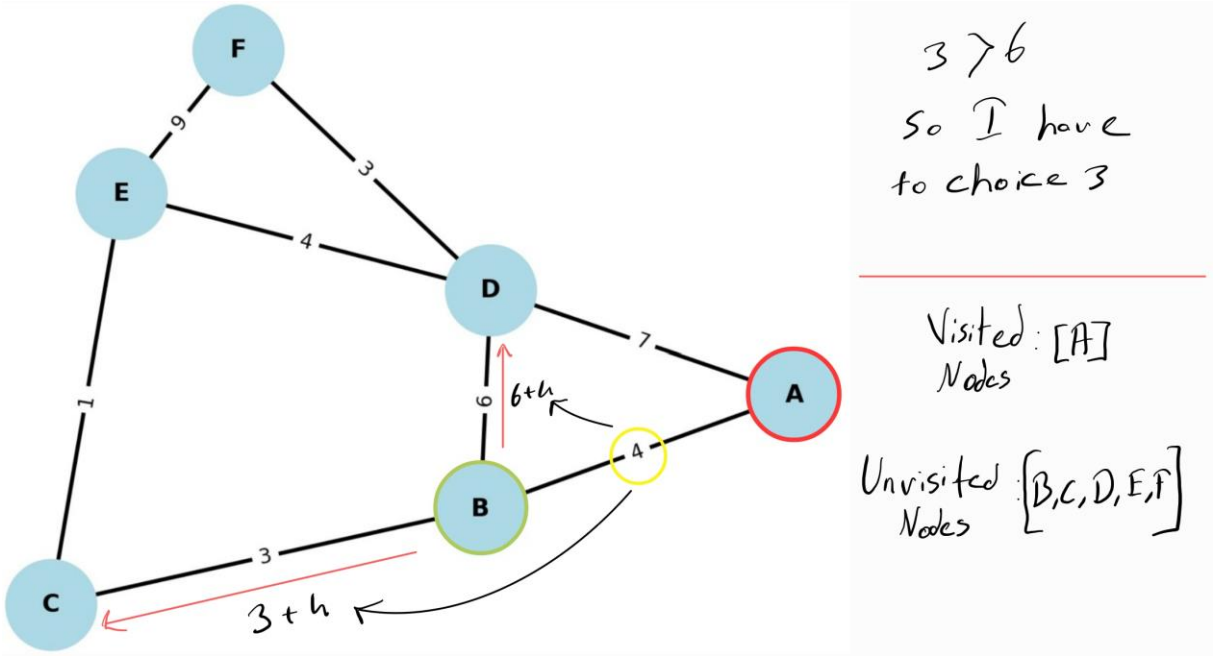
Step 1:



Node	Shortest Distance	Previous Node
A	0	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	

- We start with 0 to starting point (starting point dosen't matter) (But I choise A to F) and choise the less value.
- Since B is 4 we choise B

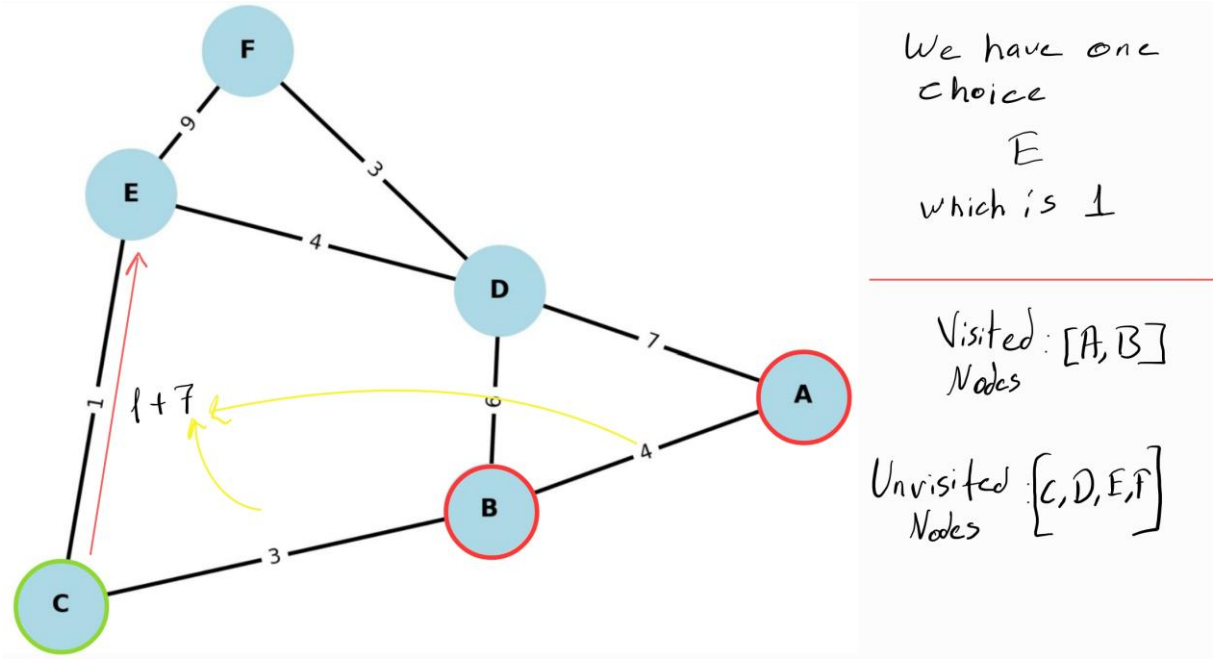
Step 2:



Node	Shortest Distance	Previous Node
A	0	
B	4	A
C	7	B
D	7	A
E	∞	
F	∞	

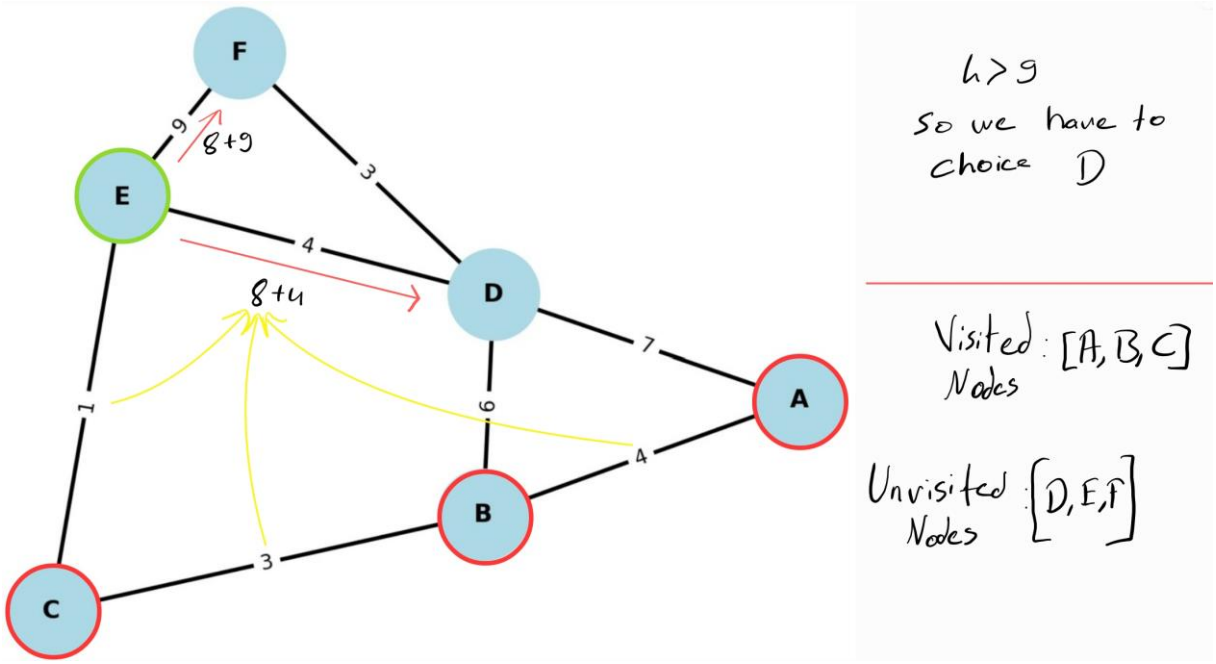
- Since 6+4=10 and also 7>10 we do not change the value of D and we write 7 insted of 10
- We choice C because 3>6

Step 3:



Node	Shortest Distance	Previous Node
A	0	
B	4	A
C	7	B
D	7	A
E	8	C
F	∞	

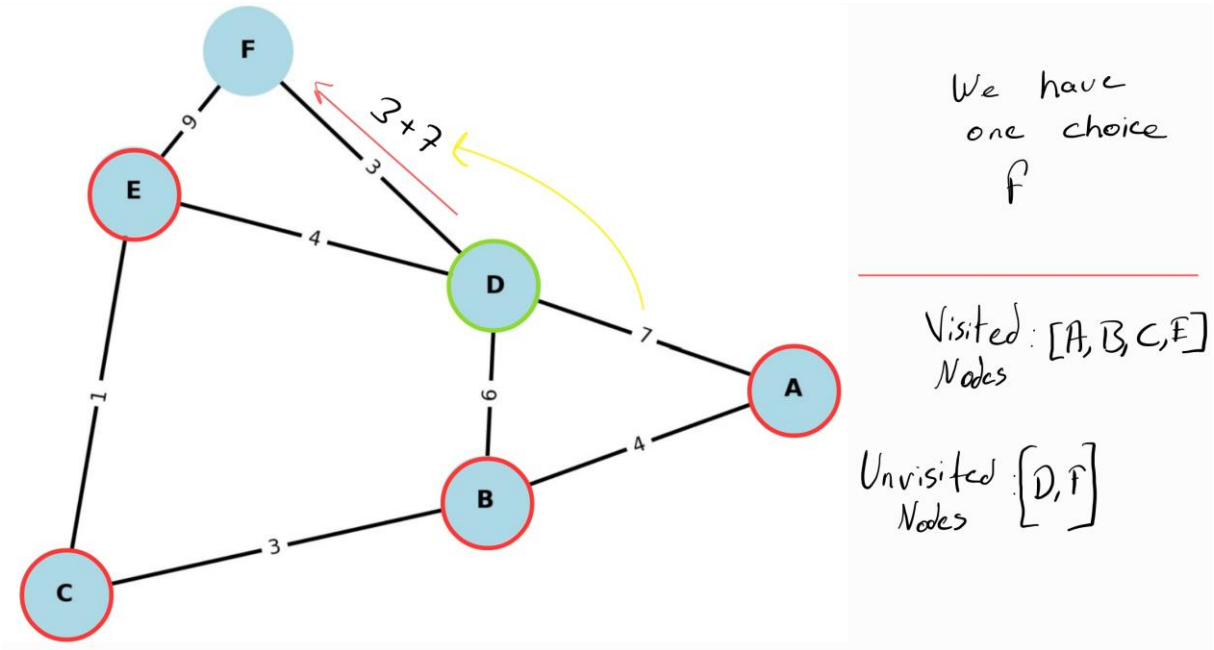
Step 4:



Node	Shortest Distance	Previous Node
A	0	
B	4	A
C	7	B
D	7	A
E	8	C
F	17	E

- Since 4>9 we have to choice D
- However D value does not change. that means insted A-B-C-E-D=12 we can go like A-D=7.
- 7<12

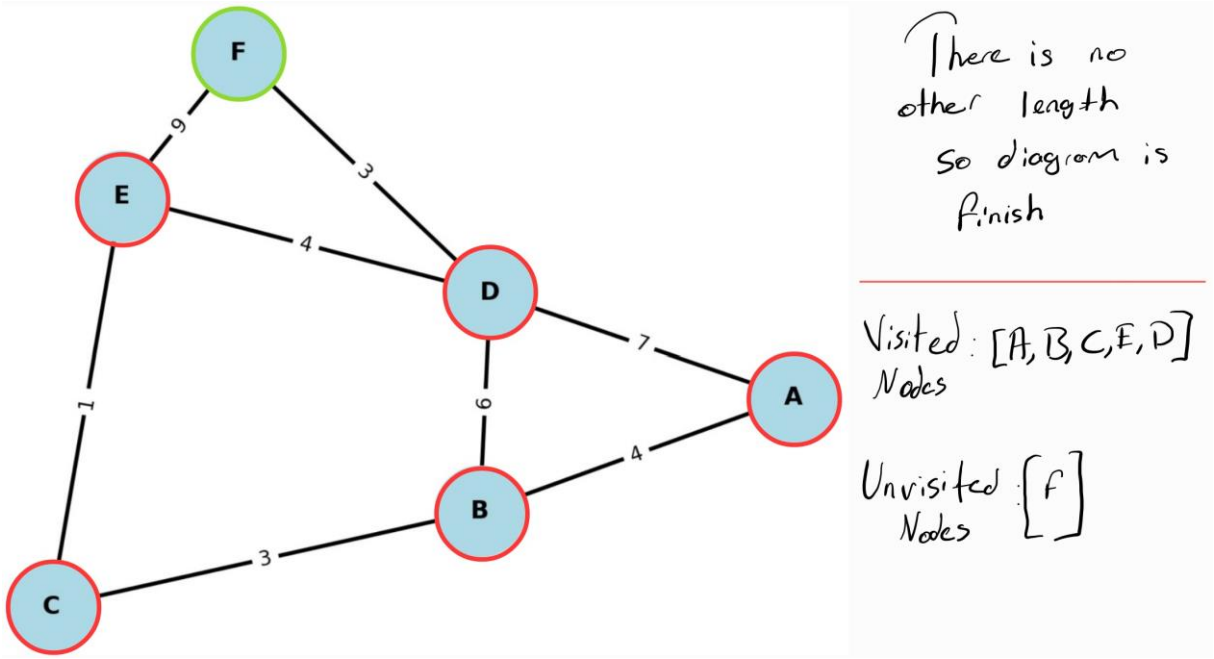
Step 5:



Node	Shortest Distance	Previous Node
A	0	
B	4	A
C	7	B
D	7	A
E	8	C
F	10	D

- F value is change because insted we go A-B-C-E-F=17 we can go A-D-F=10

Step 6:



Node	Shortest Distance	Previous Node
A	0	
B	4	A
C	7	B
D	7	A
E	8	C
F	10	D

Q 4.1: Explain what a critical edge in a graph is.

A: A **critical edge** in a graph is an edge whose removal increases the number of connected components in the graph. This means that the graph becomes disconnected if that edge is removed. Identifying critical edges is important in applications like network reliability and traffic flow optimization.

Q 4.2: Then try to find critical edges in this graph.

A:

- (C, E) = 1
- (C, B) = 3
- (D, F) = 3
- (D, E) = 4

- (A, B) = 4

These points are critical

Q 4.2: Show detailed steps for a single edge removal.

A:

For (C, E) case if we have (C, E) A-B-C-E=8 but if not A-B-D-F-E=22

For(C, B) case if we have (C, B) A-B-C=7 but if not A-B-D-F-E-C=22

For(D, F) case if we have (D, F) A-B-D-F=13 but if not A-B-D-E-F=23

For(D, E) case if we have (D, E) A-B-C-E-D=12 but if not A-B-C-E-F-D=20

For(A, B) case if we have (A, B) A-B=4 but if not A-D-B=13

Q 5.1: Explain what an articulation point in a graph is.

A: An **articulation point** (or cut vertex) in a graph is a vertex that, when removed along with its associated edges, increases the number of connected components in the graph. In other words, removing an articulation point disconnects the graph.

Q 5.2: Then try to find articulation points in this graph.

A: If the C and D disconnect then the graph It is divided into two parts.

Q 5.3: Show detailed steps for a single vertex removal

A: If we have to remove one vertex so there is no articulation points. In this graph there is two vartex that hold in one diagram this is C and D and if we remove both of them;

Graph 1.

(A, B) = 4

Graph 2.

(F, E) = 9

Q 6: Suppose you want to go from A to E, the path you are given is A-B-C-E (based on Dijkstra’s algorithm). You are at B and learn that C is now unavailable. Given that

you know there are no critical edges or articulation points beforehand, can you be sure that there is now another path towards E without any calculations?

A: Yes, there is way to go E. If the C is close we can go from D way so destination is go like A-B-D-E= 14

Q 7: Do some research on the concept of graph robustness, and explain it using critical edges and articulation points

A: Graph robustness refers to the **ability of a graph (network)** to maintain its **connectivity, functionality, or performance** despite the removal of certain nodes or edges. In simpler terms, it measures how "resilient" a graph is to failures, attacks, or disruptions.

In graph:

- Articulation points like **C** and **D** are critical to robustness.
- If **C** is removed, the graph will become disconnected, indicating it has **low robustness** in that area.
- If multiple random nodes are removed, the presence of redundant connections (e.g., between D, B, and E) helps maintain some connectivity, improving **random failure robustness**.