Name: Georges Hatem

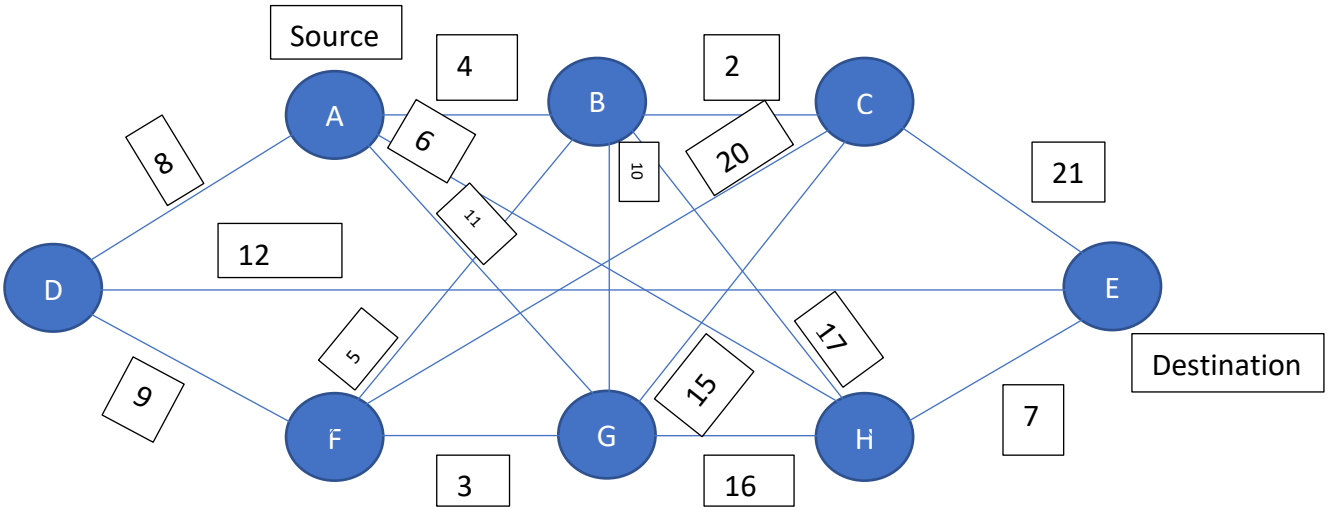
CS590 Homework Assignment 12: Shortest Paths Reinforcement Exercises

Due Date: April 17, 2022

Problem 14.7.1:

Dijkstra's Algorithm is mainly used to determine the shortest path from one node to every other within the same graph. Generally, the algorithm will run until all the vertices in the graph have been visited, which means it looks for the shortest path between the two nodes.

An example of Dijkstra's Algorithm with a simple, connected, weighted, undirected graph with 8 vertices and 16 edges, each with distance edge weight, is as follows:



Rules:

In the above graph, there will be many possible paths from source vertex to destination vertex, but the Dijkstra's Algorithm helps to find the shortest path between two nodes.

There are some rules to run the Dijsktra's Algorithm, and they are as follows:

- Start from the source node and visit the vertex with the smallest distance or cost.
- Once the smallest cost vertex is identified, check the neighboring nodes.
- Calculate the cost for neighboring nodes by summing the cost of the edges from the start vertex.
- If the distance/cost to the vertex is less than the known distance, then update the shortest distance for the vertex.

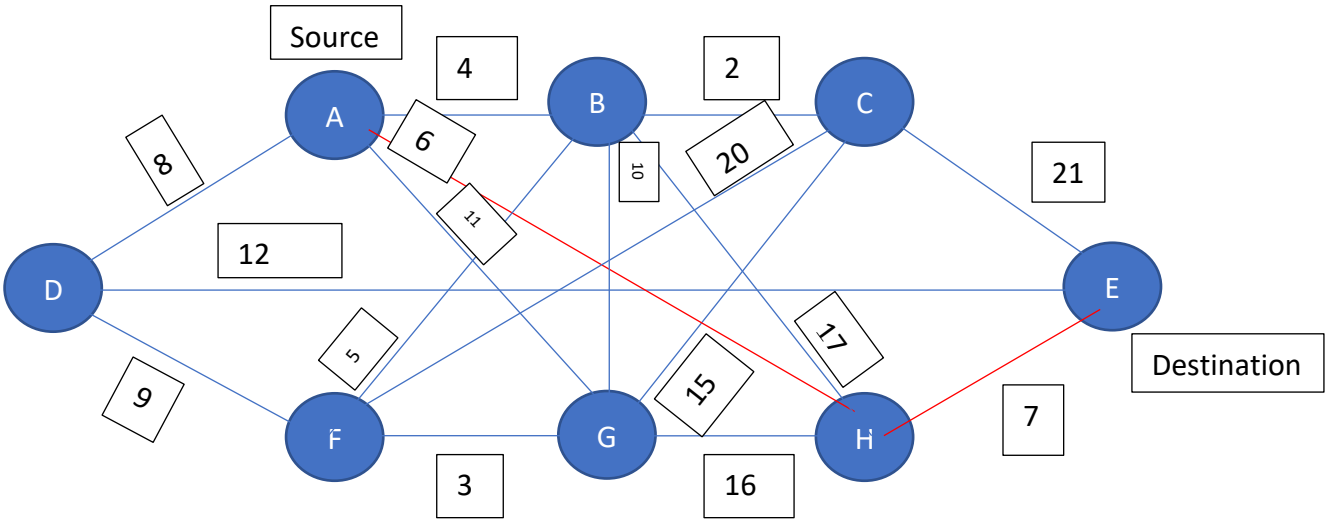
Explanation:

In the above graph, the following are the possible paths from starting vertex “A” to destination vertex “E”:

- A-B-C-E – 27(4+2+21)
- A-B-H-E – 28(4+17+7)
- A-H-E – 13(6+7)

Thus, from the above cost, the minimum cost for path to reach the destination vertex “E” is “13” from source vertex “A”.

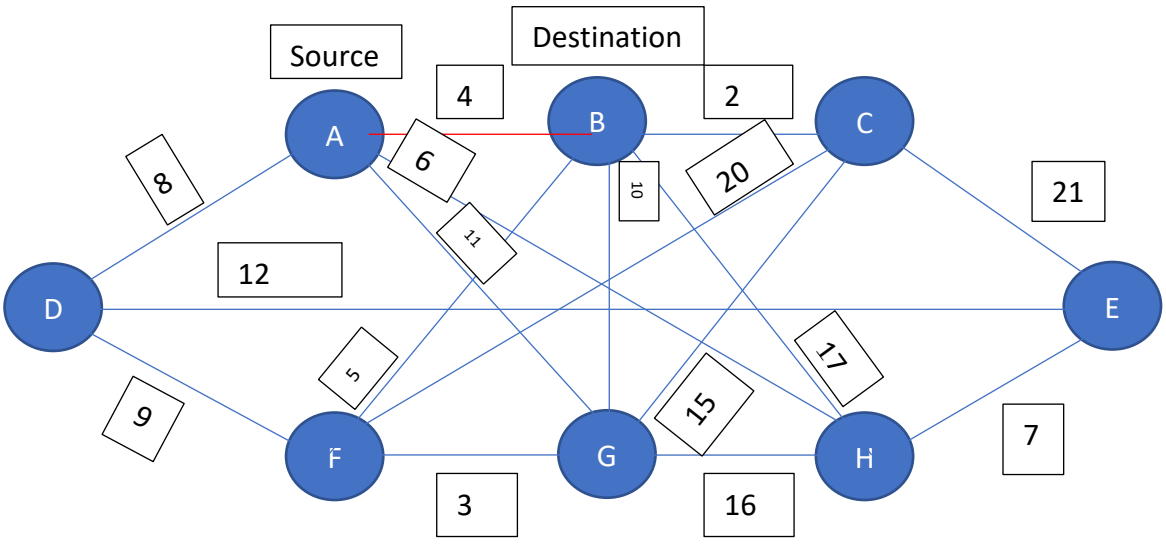
Therefore, the shortest path is A-H-E, and the following below is the diagrammatic representation (The shortest path A-H-E is shown in red below):



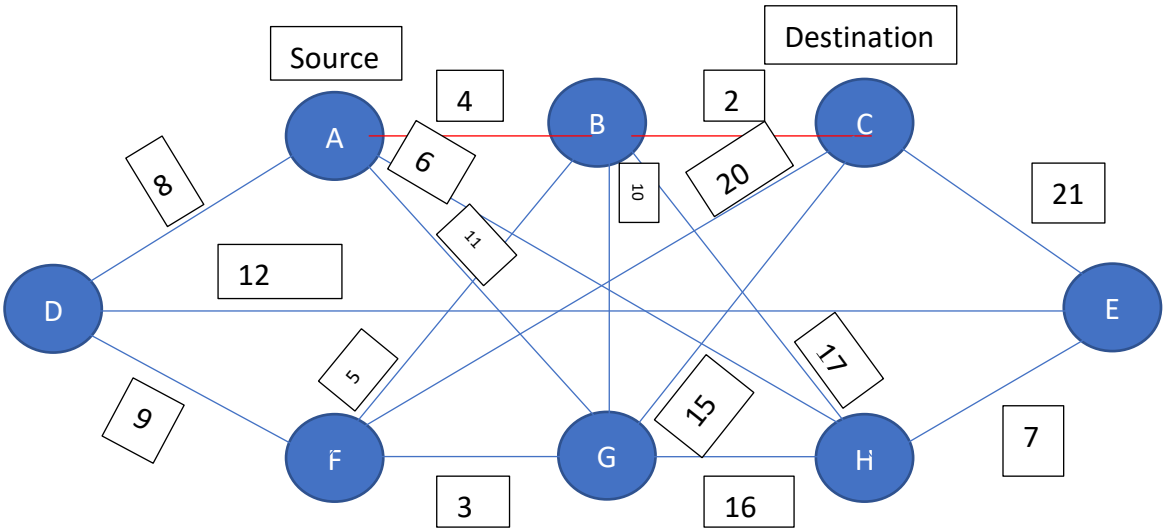
It is very important to note that Dijktra’s Algorithm runs for all vertices other than the

source. This is just an example using Destination E. The same concept applies for all other vertices.

Suppose vertex B is the destination, then the shortest way to go from A to B is A-B. There are multiple paths that can be taken to get to B. But, A-B is the shortest path to get from A to B that will be returned by Dijkstra's Algorithm if B is the destination. A-B – (4). I highlighted the shortest path from A to B below in red.

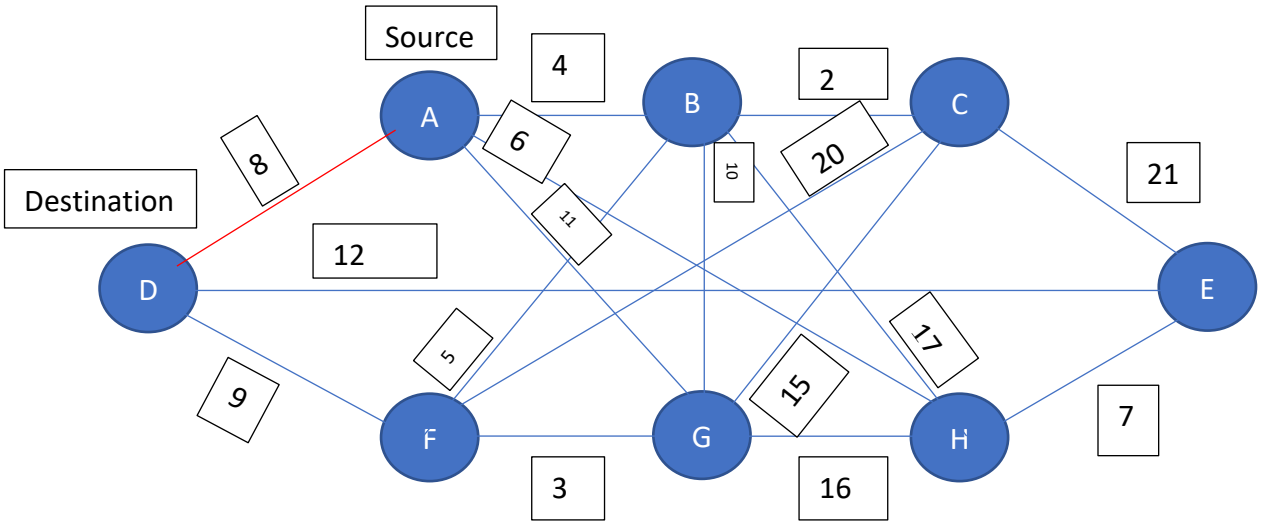


Suppose vertex C is the destination, then shortest path to get from A to C is A-B-C. As indicated above, there are multiple paths that can be taken to get to C. But, A-B-C is the shortest path to get from A to C that will be returned by Dijkstra's Algorithm if C is the destination. A-B-C – 6(4+2). I highlighted the shortest path from A to C below in red.



Suppose vertex D is the destination, then the shortest path to get from A to D is A-D. As indicated above, there are multiple paths that

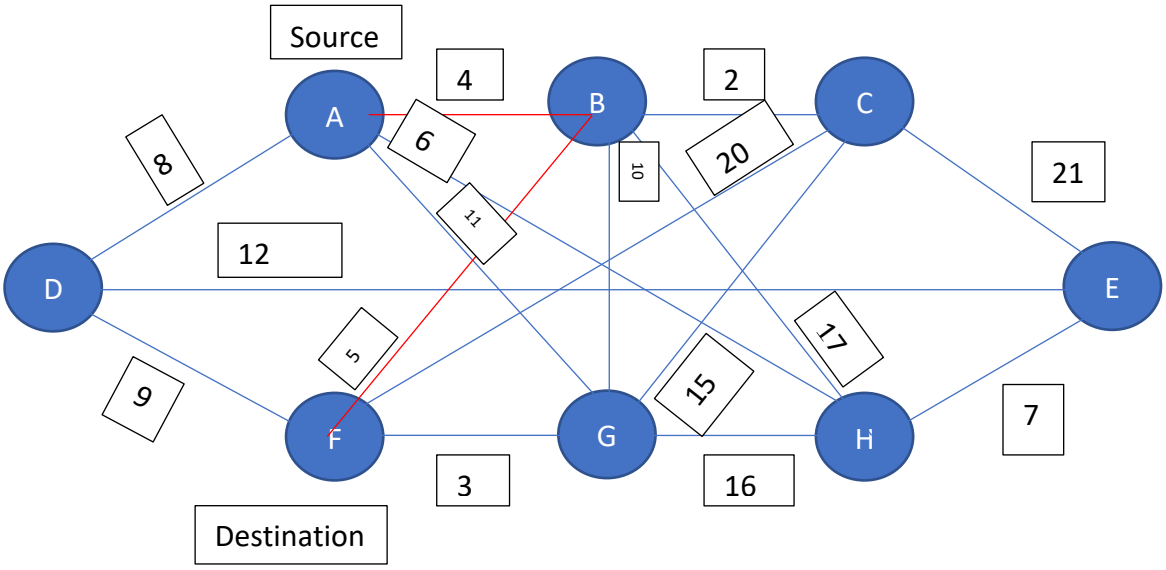
can be taken to get to D. But, A-D is the shortest path to get from A to D that will be returned by Dijkstra's Algorithm if D is the destination. A-D – (8). I highlighted the shortest path from A to D below in red.



Suppose vertex F is the destination, then let's analyze a bit:

A-G-F – 14(11+3)
A-D-F – 17(8+9)
A-B-F – 9(4+5)

There are multiple paths other than these paths indicated above. However, the shortest path between all paths from A to F would be A-B-F. A-B-F is the shortest path to get from A to F that will be returned by Dijkstra's Algorithm if F is the destination. A-B-F – 9(4+5). I highlighted the shortest path from A to F below in red.

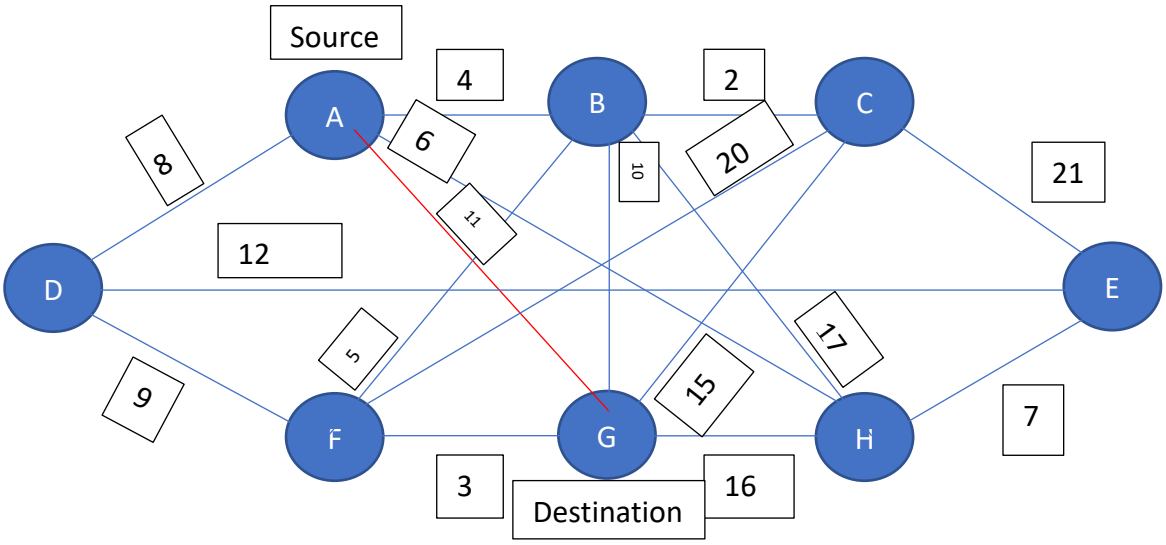


Suppose vertex G is the destination, then let's analyze as we did above for F:

A-G – 11(11)
A-B-G – 14(4+10)

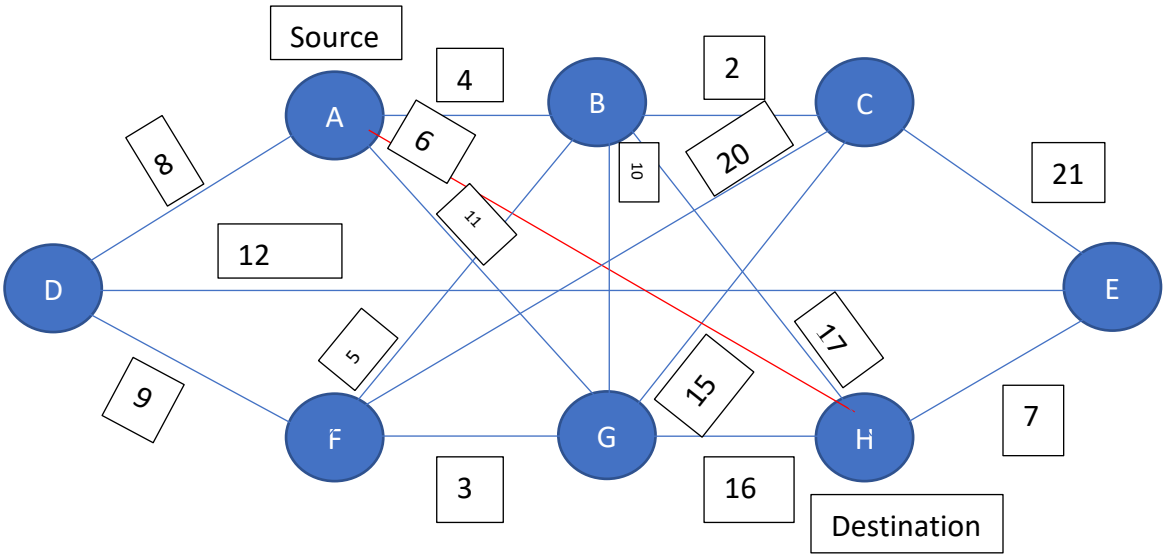
A-B-F-G – 12(4+5+3)
A-H-G – 22(6+16)

There are multiple paths that can be taken other than the illustrated above. However, the shortest path from A to G would be A-G. A-G is the shortest path to get from A to G that will be returned by Dijkstra's Algorithm if G is the destination. A-G – 11(11). I highlighted the shortest path from A to G below in red.



Suppose vertex H is the destination, then if we analyzed like we did above for F and G, we will

determine that the shortest path from A to H is A-H. A-H is the shortest path to get from A to H that will be returned by Dijkstra's Algorithm if H is the destination. A-H – 6(6). I highlighted the shortest path from A to H below in red.



Problem 14.7.3:

Solution:

For each vertex u, use a variable adjcentvertx[u], which stores its adjacent vertex in cloud. To construct a tree, make 2 changes in Dijkstra's Algorithm:

1) When a vertex u (u#v) with minimum distance value D(u) removed from priority queue Q, add edge (adjcentvertx[u], u) to tree T.
2) If(D[u] + w((u,z)) < D[z]), set adjcentvertx[z] to u.

The Modified Dijistra's Algorithm is given below with its added steps described above. The Dijkstra's Algorithm in Figure 14.3.1 is used to add steps to it:

Algorithm DijikstraShortestPath(G,v):
Input: A simple undirected weighted graph G with nonnegative edge weights, and a distinguished vertex v of G.
Output: A label, D[u], for each vertex u of G, such that D[u] is the distance from v to u in G, and the corresponding tree
D[v] ← 0
for each vertex u#v of G do
    D[u] ← + ∞
Let a priority queue, Q, contain all the vertices of G using the D labels as keys.
while Q is not empty do
    // pull a new vertex u into the cloud
    u ← Q.removeMin()
    add edge(adjcentvertx[u], u) to T
    for each vertex z adjacent to u such that z is

```
    in Q do
        // perform the relaxation procedure on
        // edge (u, z)
        if D[u] + w((u, z)) < D[z] then
            D[z] ← D[u] + w((u,z))
            Change the key for vertex z in Q to
            D[z]
            adjcentvertx[z] ← u
return T and the label D[u] of each vertex u
```