Kruskal's Algorithm

1. Rank your edges from lowest to highest cost

2. Have all vertices been connected to the minimum spanning tree?

   A. If yes, stop. You have found the minimum spanning tree.

   B. If no, go to step 3.

3. Pick one edge with the lowest cost that has not yet been added to the minimum spanning tree

   A. Does the edge create a cycle?

      - If yes, eliminate the edge from consideration and go to step 3

      - If no, add the edge to the minimum spanning tree and go to step 2

Prim's Algorithm

1. Pick a vertex in the network randomly, add that vertex to the spanning tree

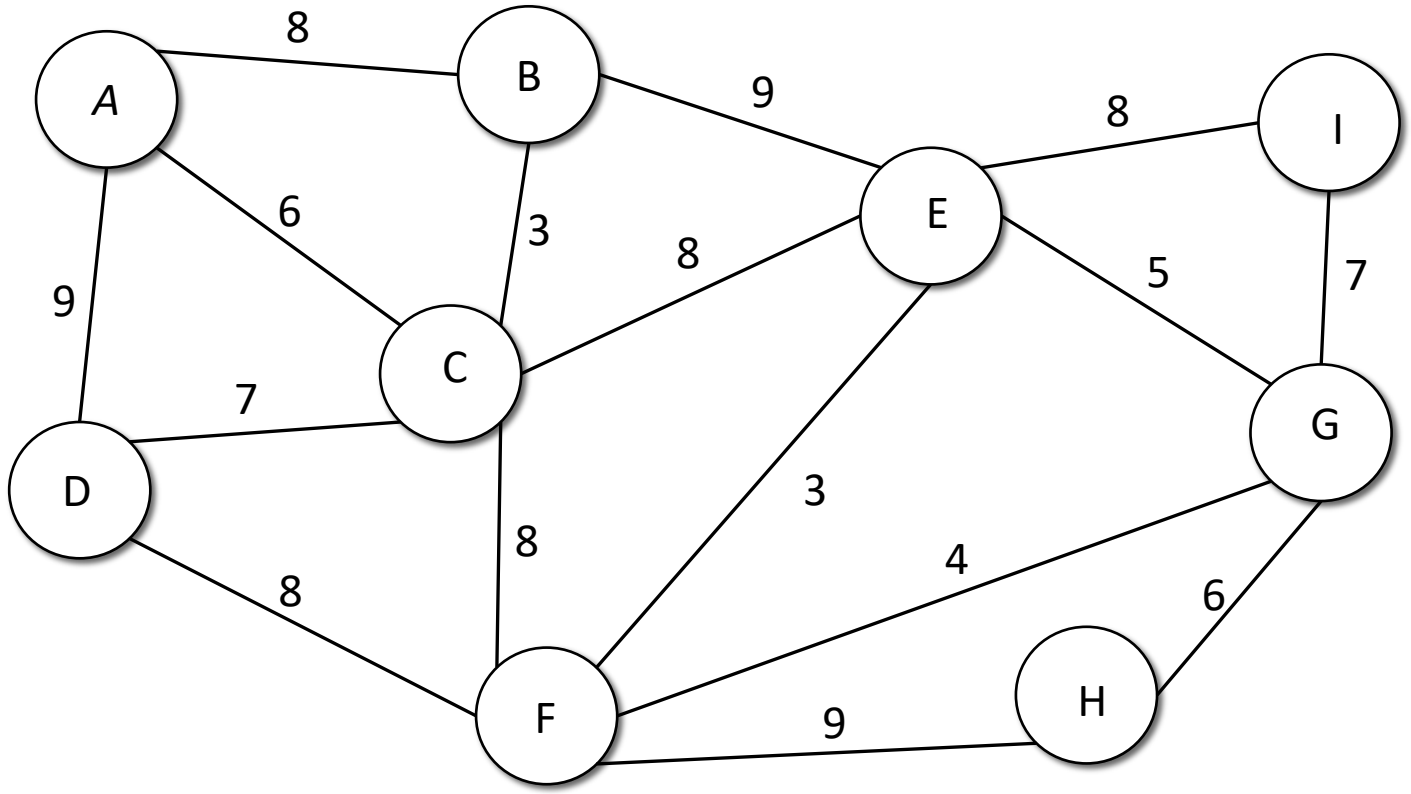
2. Examine all edges connected from a vertex in the spanning tree to a vertex NOT in the spanning tree

   - Of those edges, choose the one with the lowest (minimum) cost and add its attached vertex to the spanning tree
   - Break ties arbitrarily

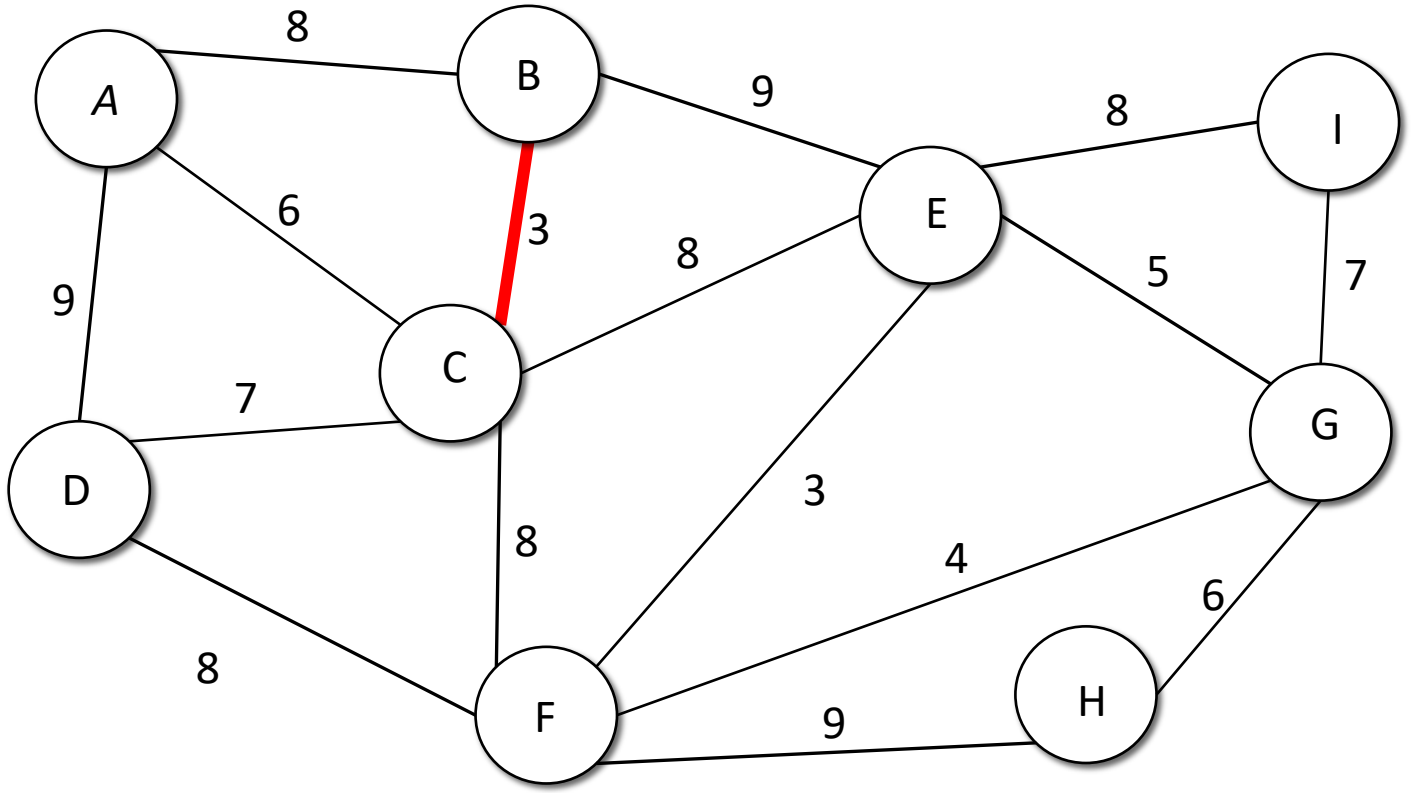
3. Are all vertices attached to the spanning tree?

   - If yes, stop
   - If no, go to step 2.

Practice Kruskal's Algorithm

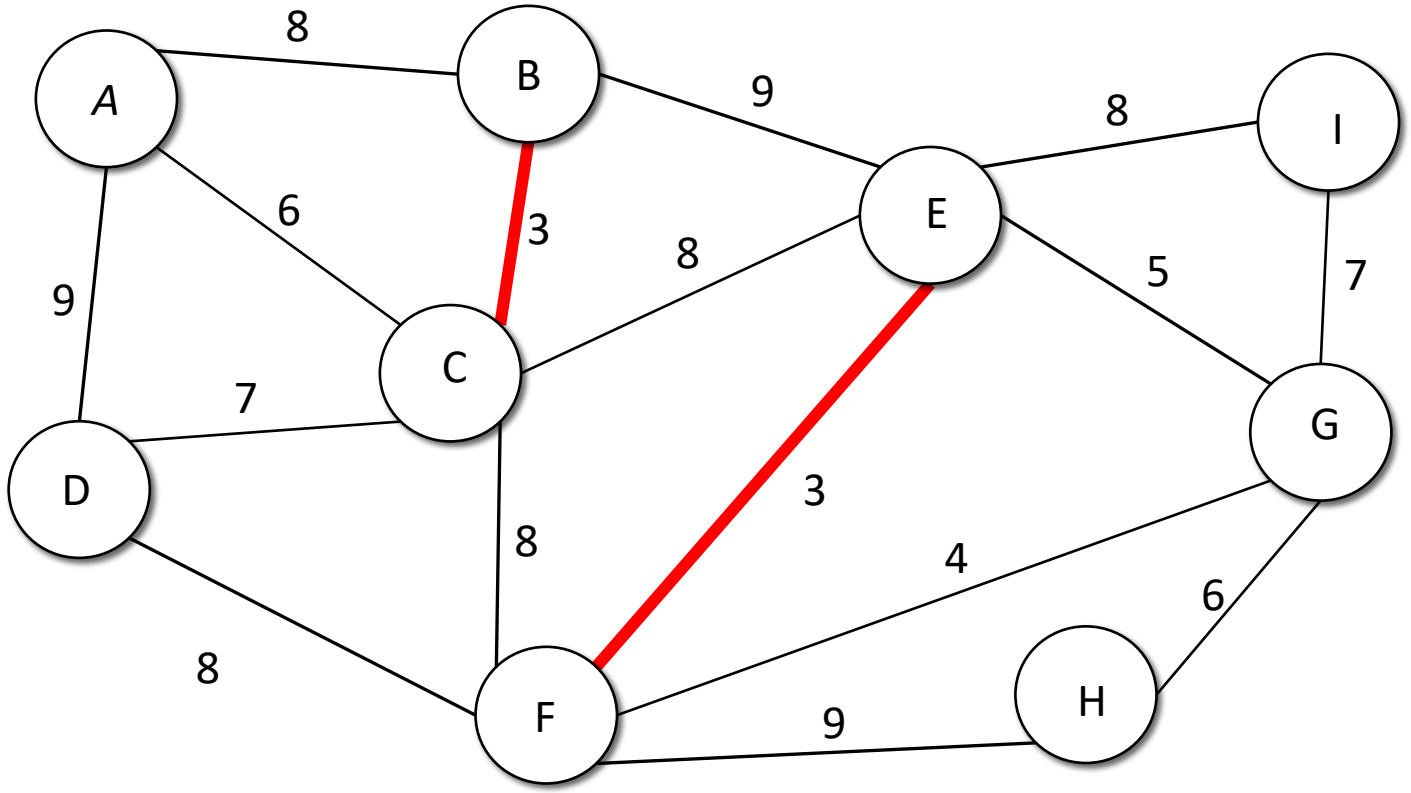


Practice Kruskal's Algorithm



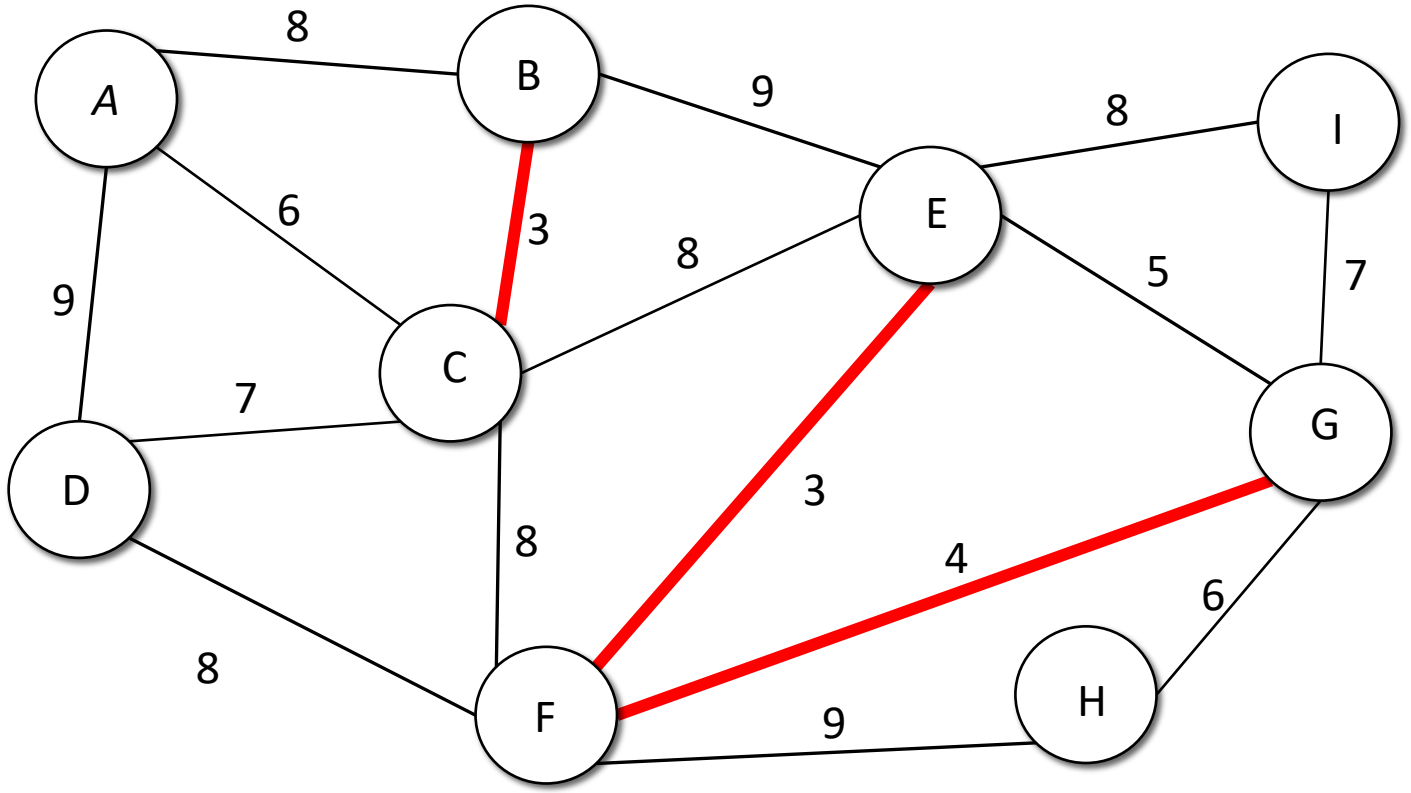
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



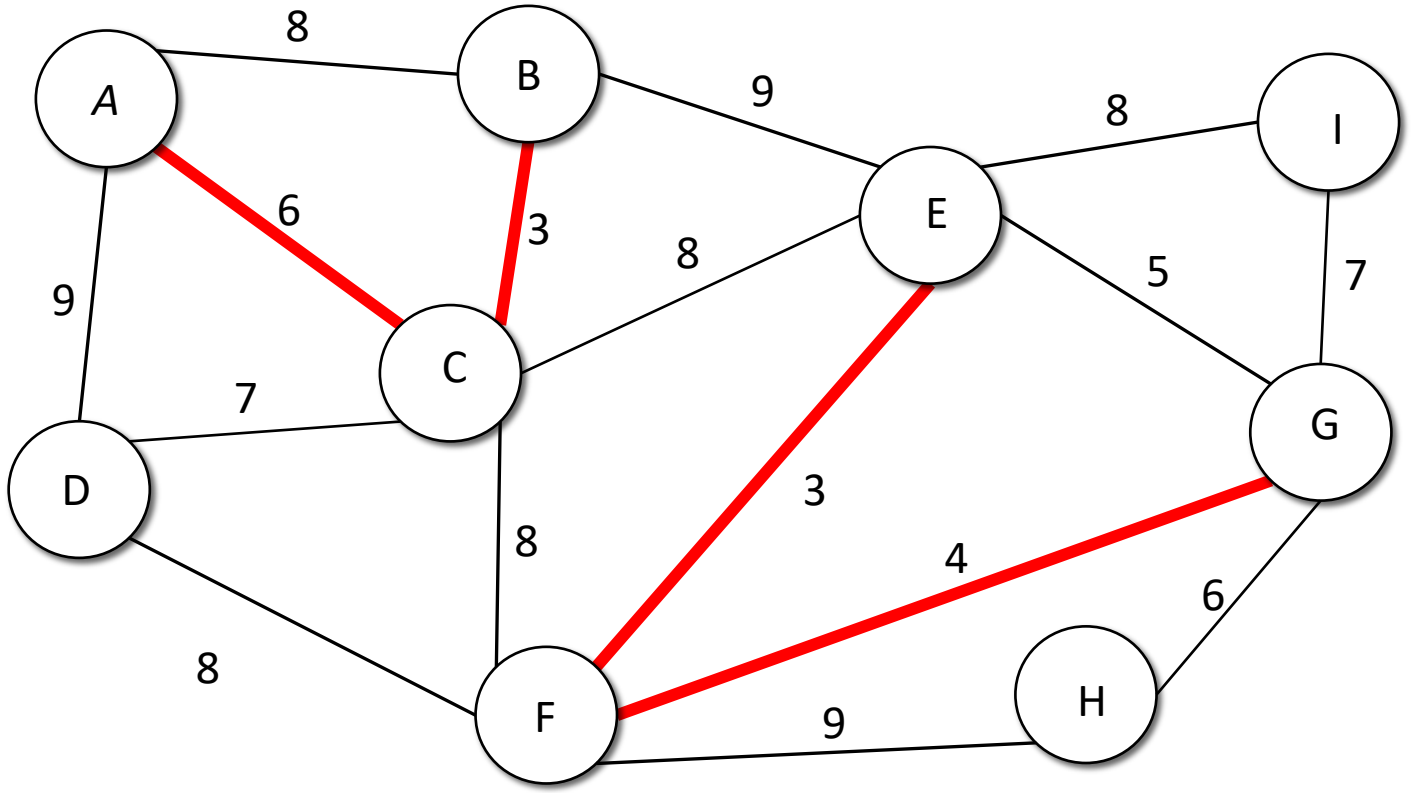
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



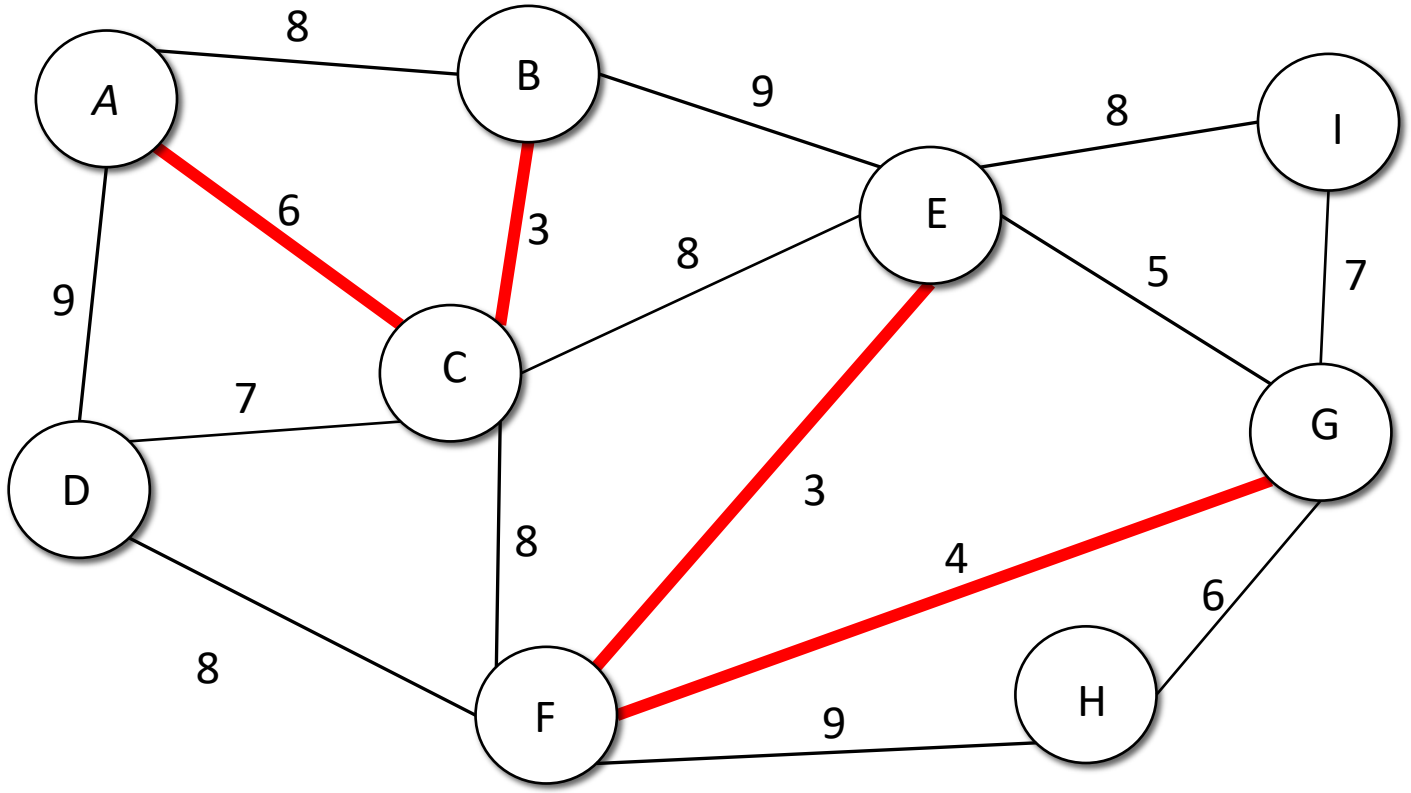
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



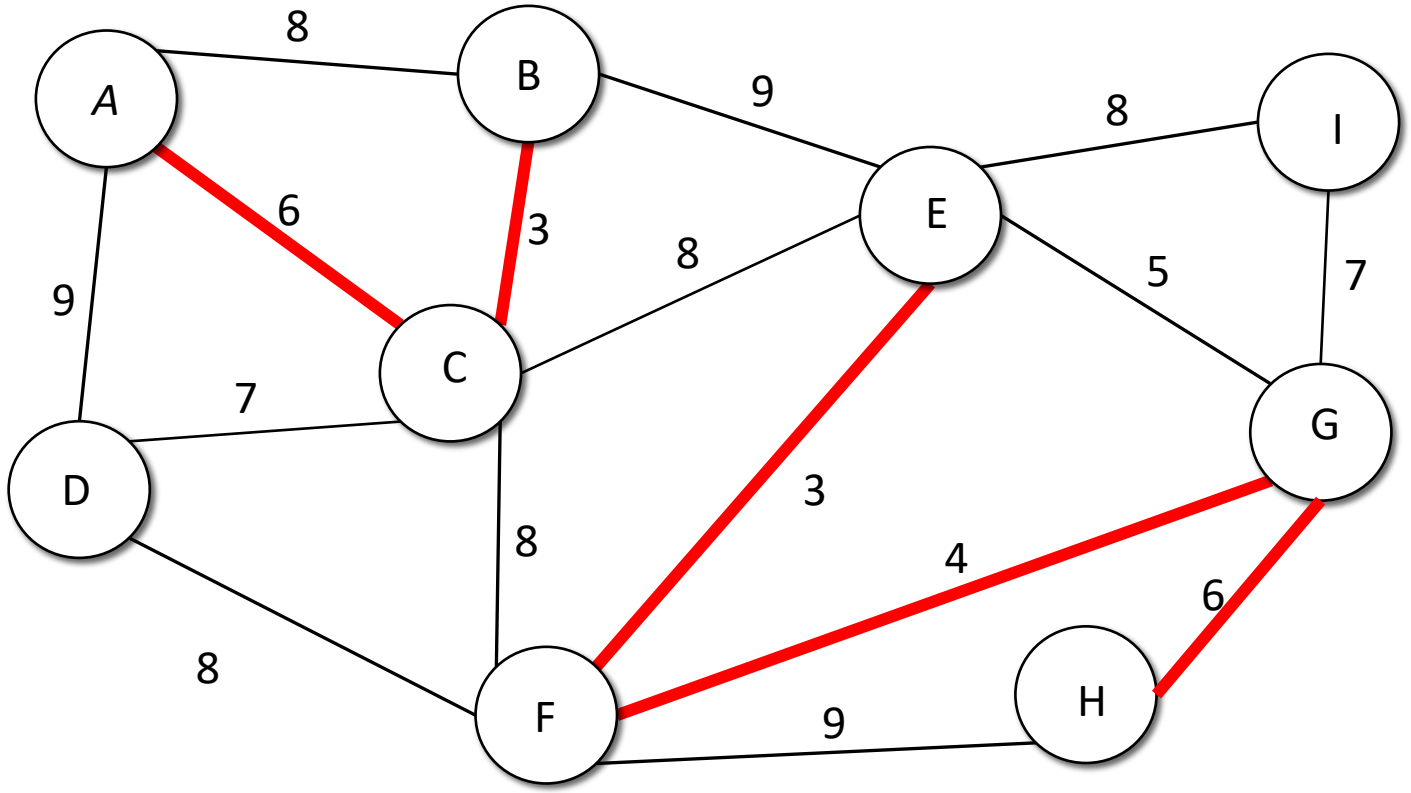
B – C = 3
E – F = 3
F – G = 4
~~E – G = 5~~
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



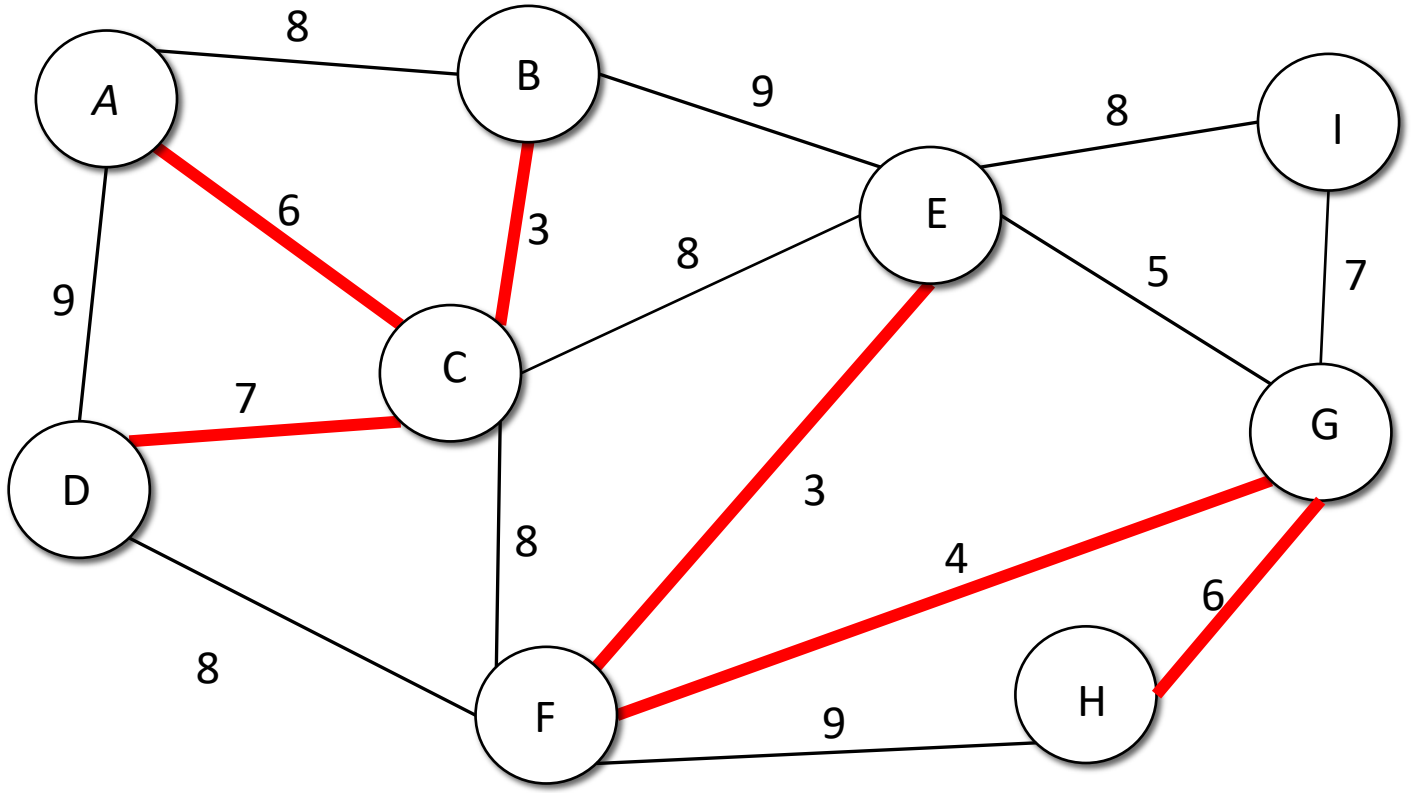
B – C = 3
E – F = 3
F – G = 4
~~E – G = 5~~
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



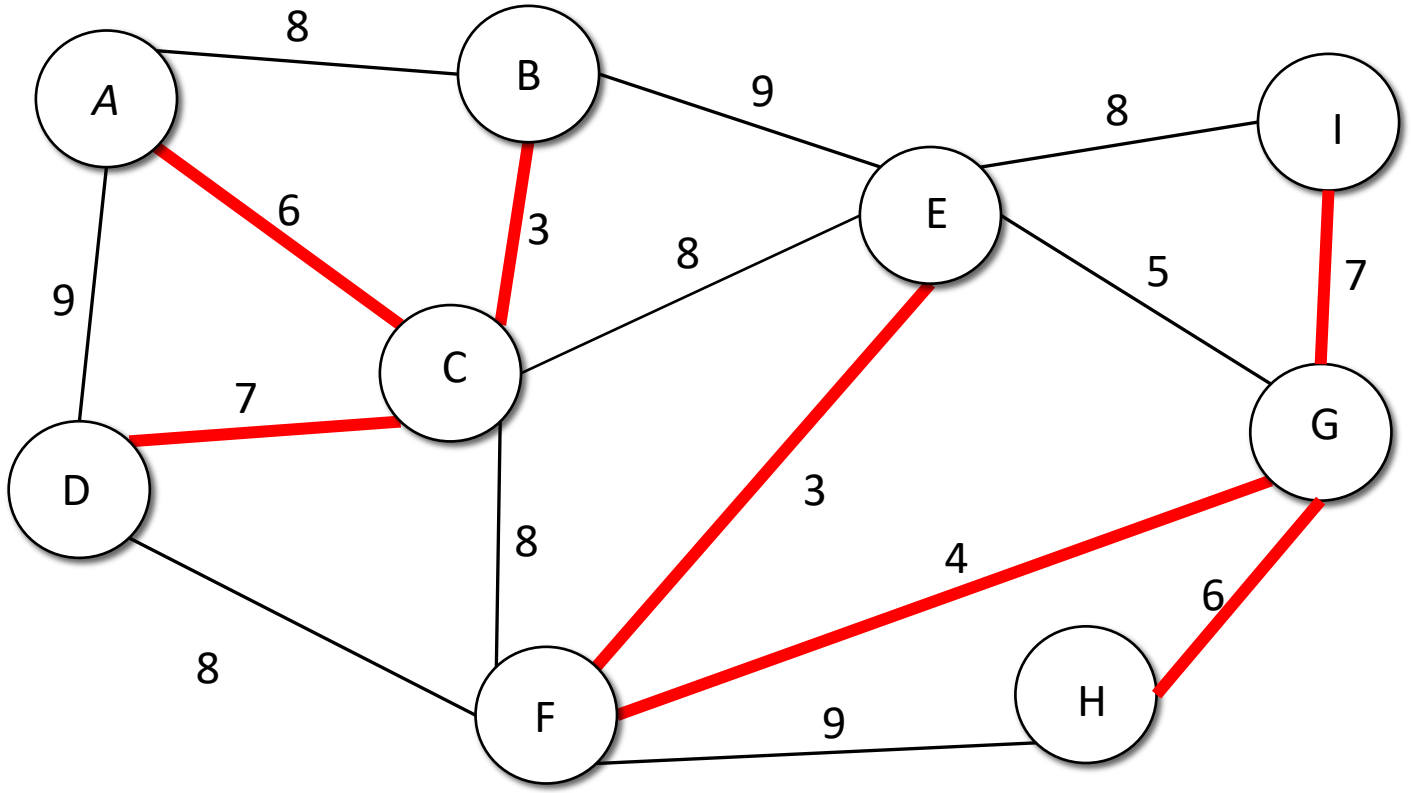
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



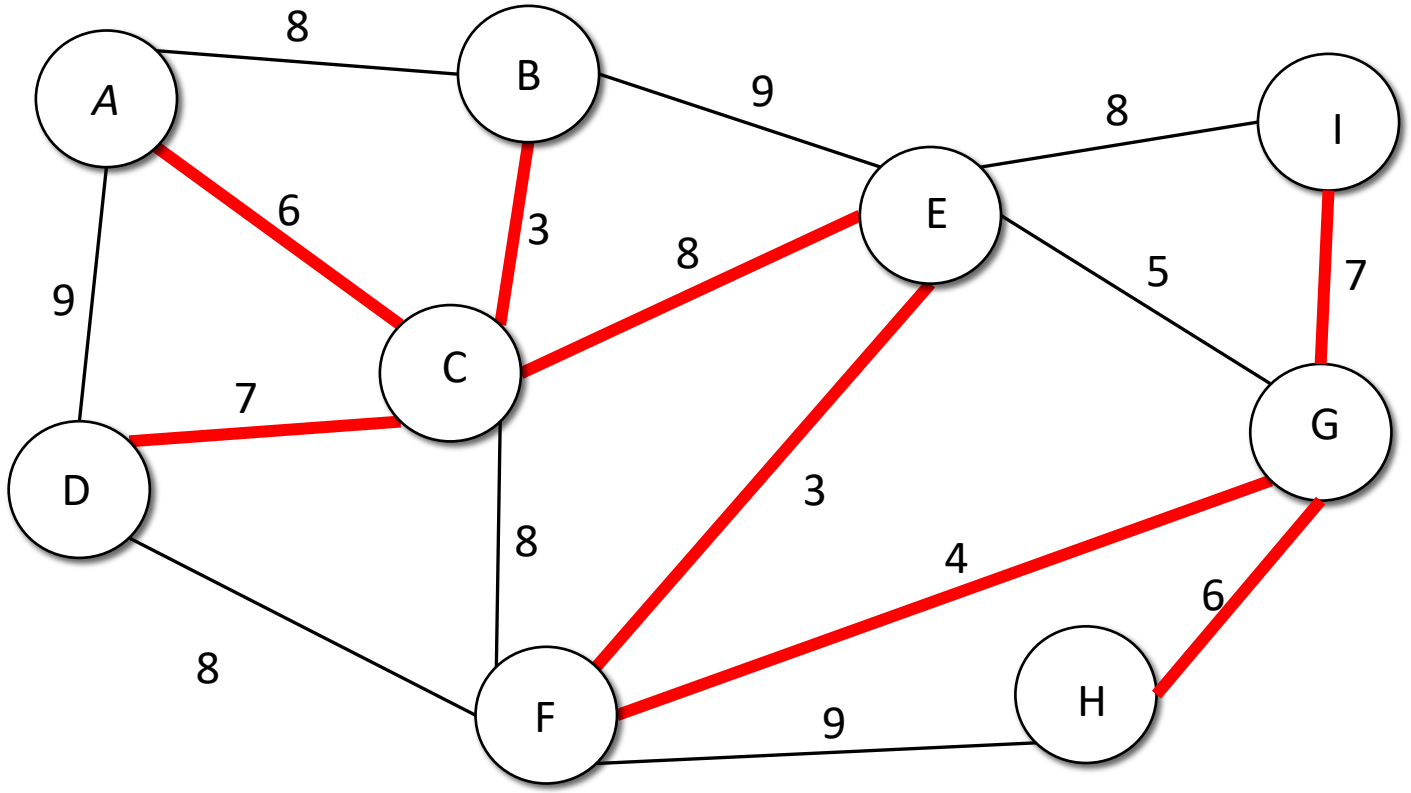
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



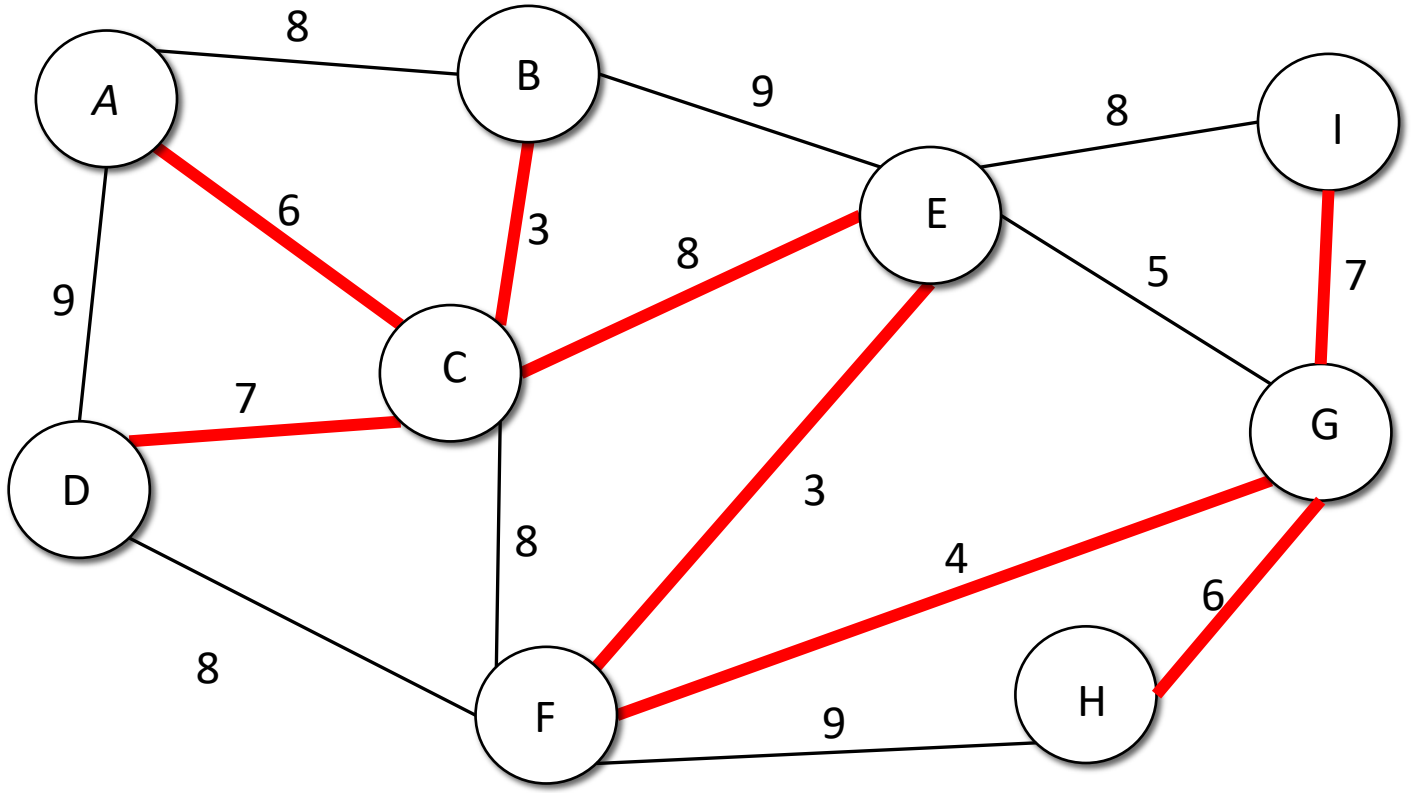
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
A – B = 8
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm



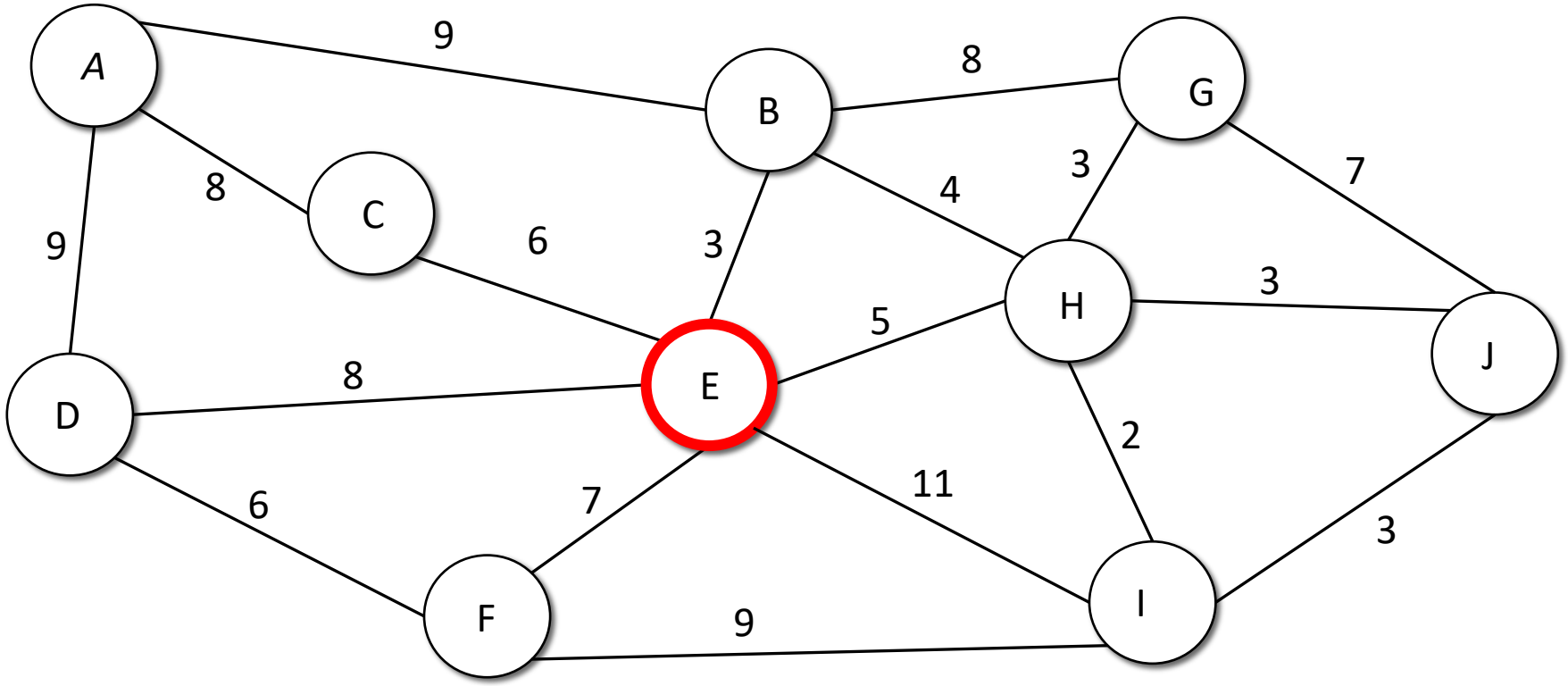
B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
~~A – B = 8~~
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

Practice Kruskal's Algorithm

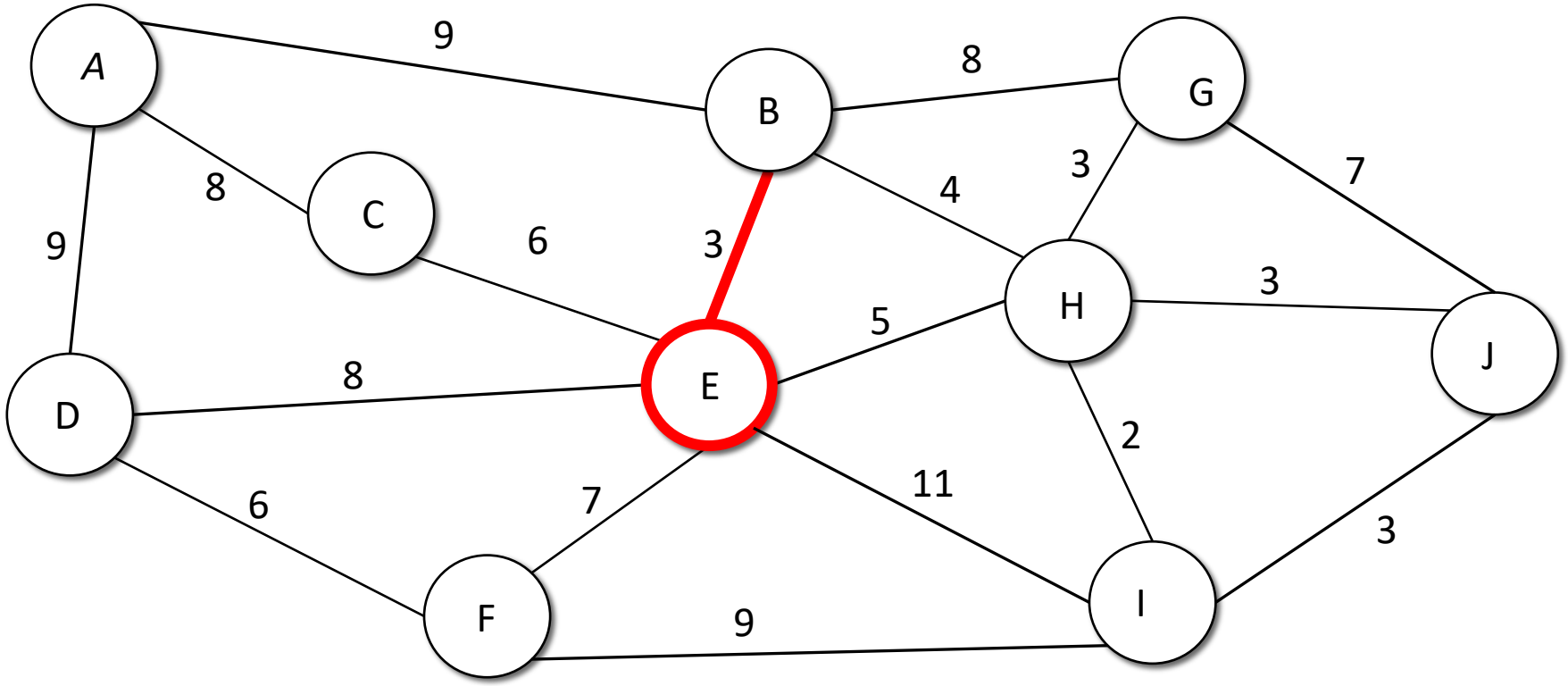


B – C = 3
E – F = 3
F – G = 4
E – G = 5
A – C = 6
G – H = 6
C – D = 7
I – G = 7
~~A – B = 8~~
C – E = 8
C – F = 8
D – F = 8
E – I = 8
A – D = 9
B – E = 9
F – H = 9

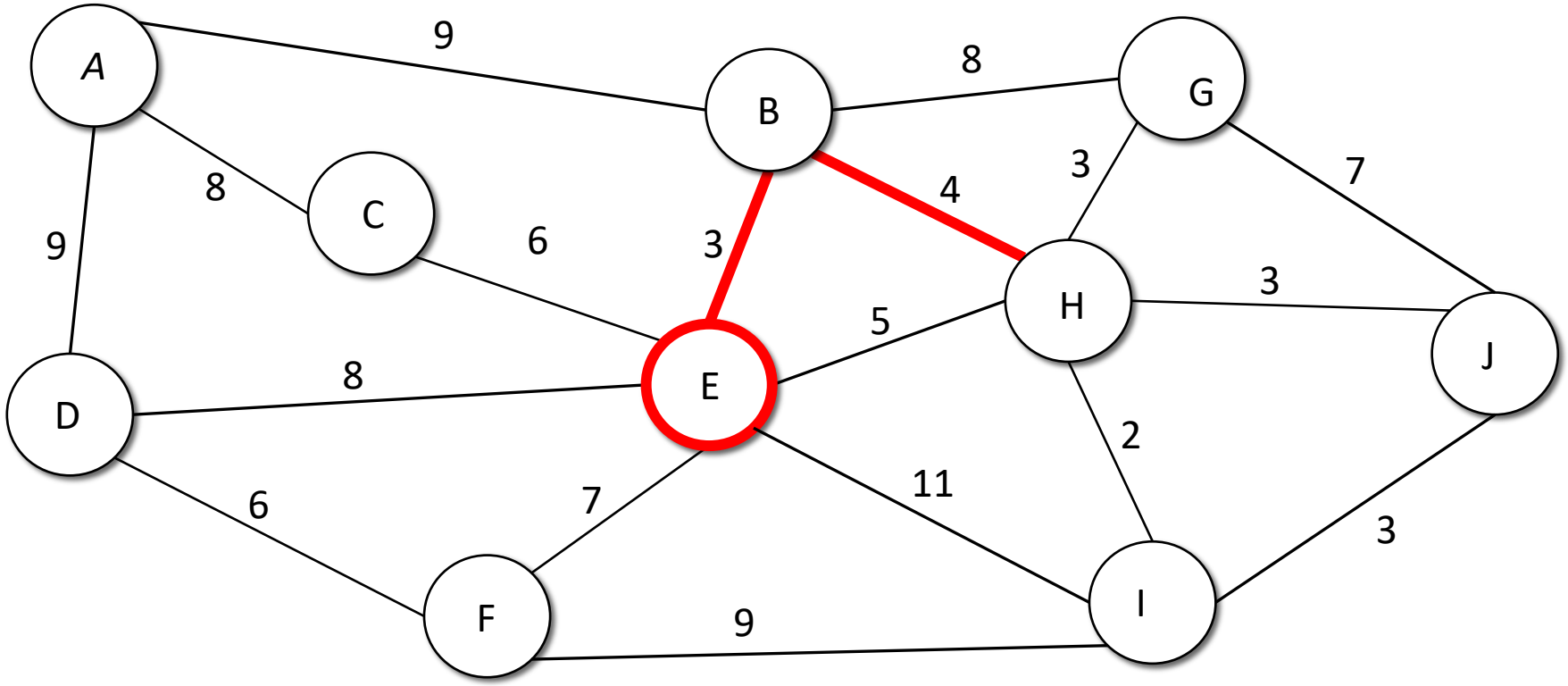
Practice Prim's Algorithm



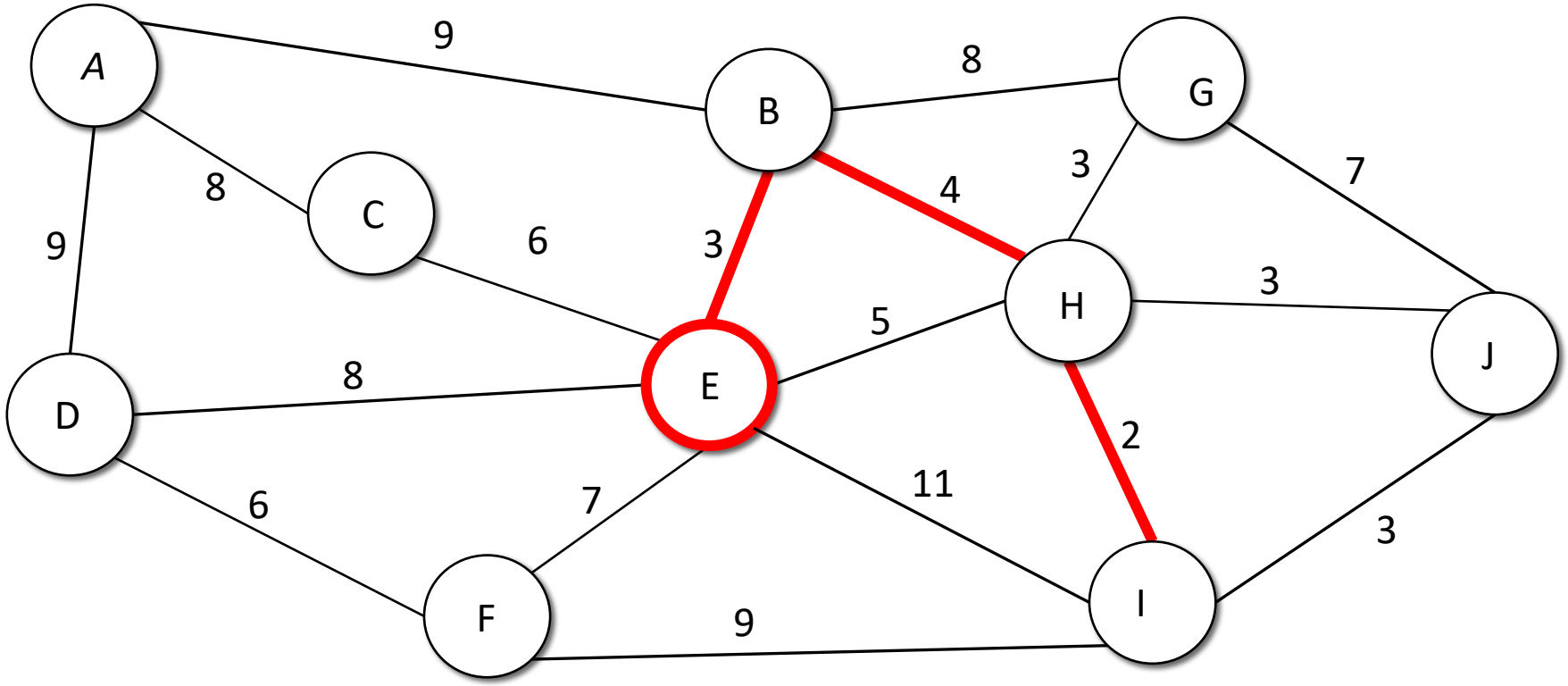
Practice Prim's Algorithm



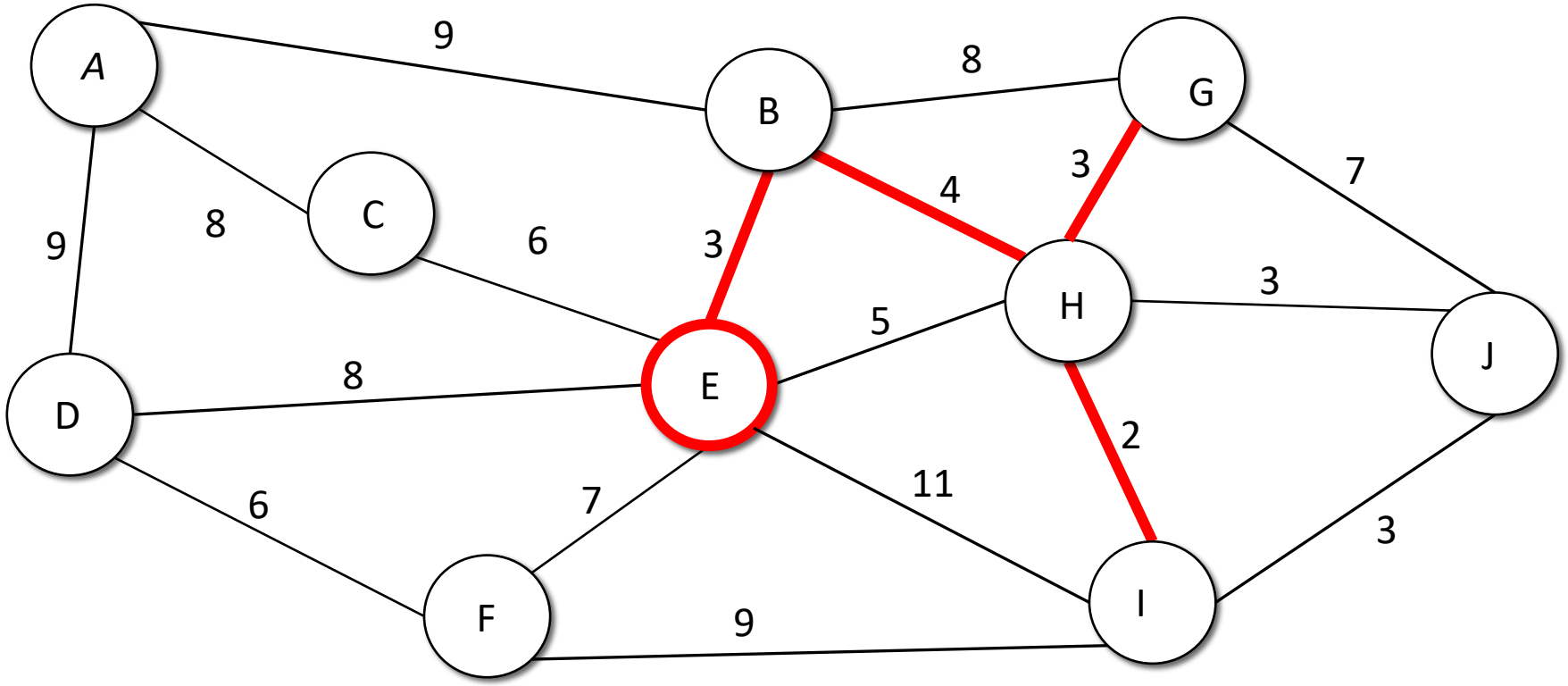
Practice Prim's Algorithm



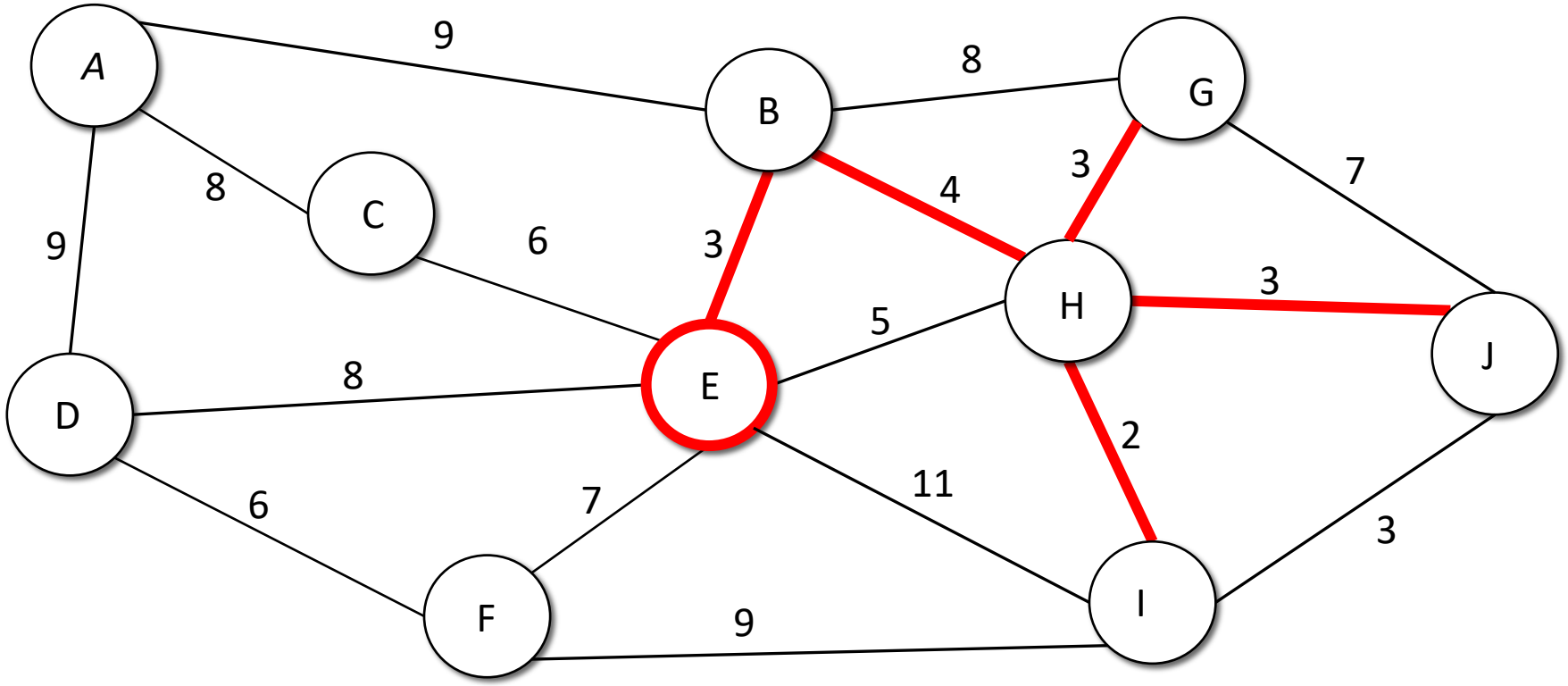
Practice Prim's Algorithm



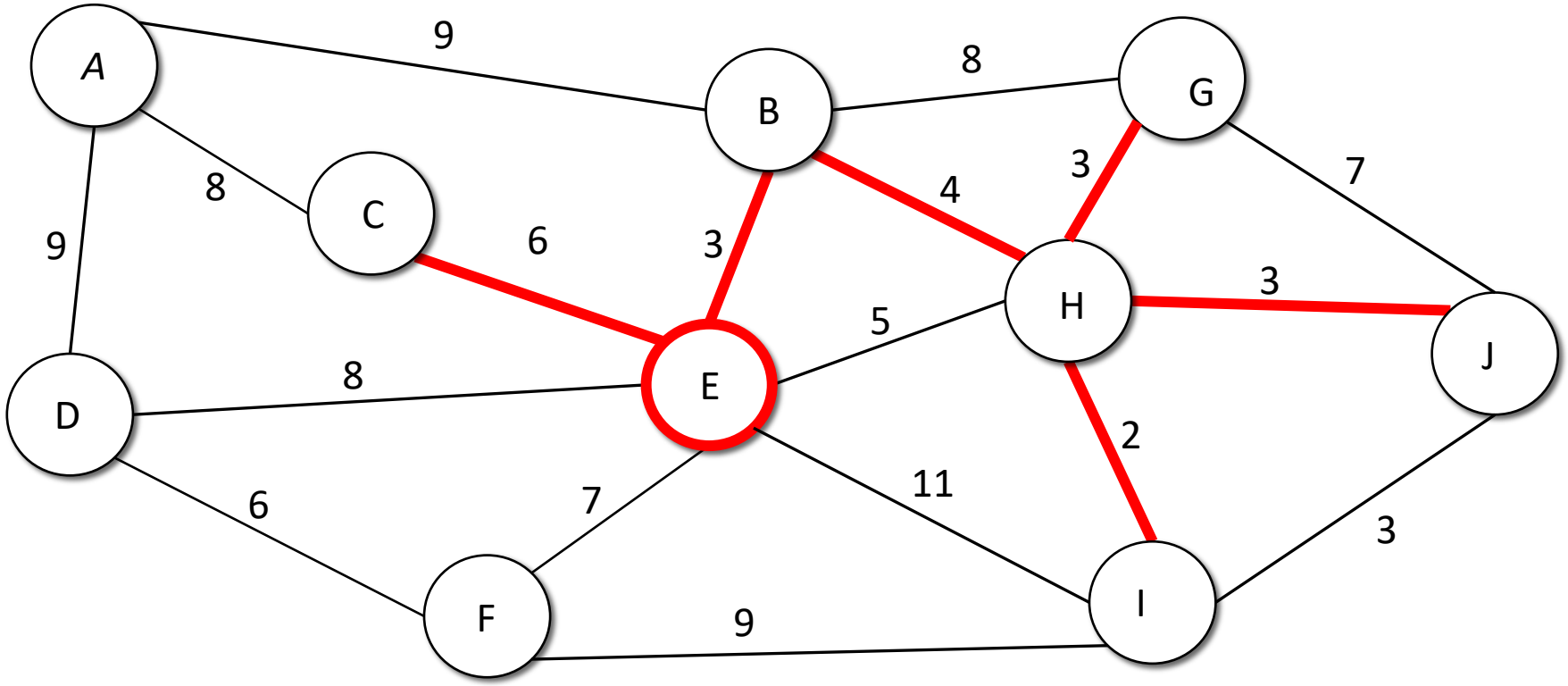
Practice Prim's Algorithm



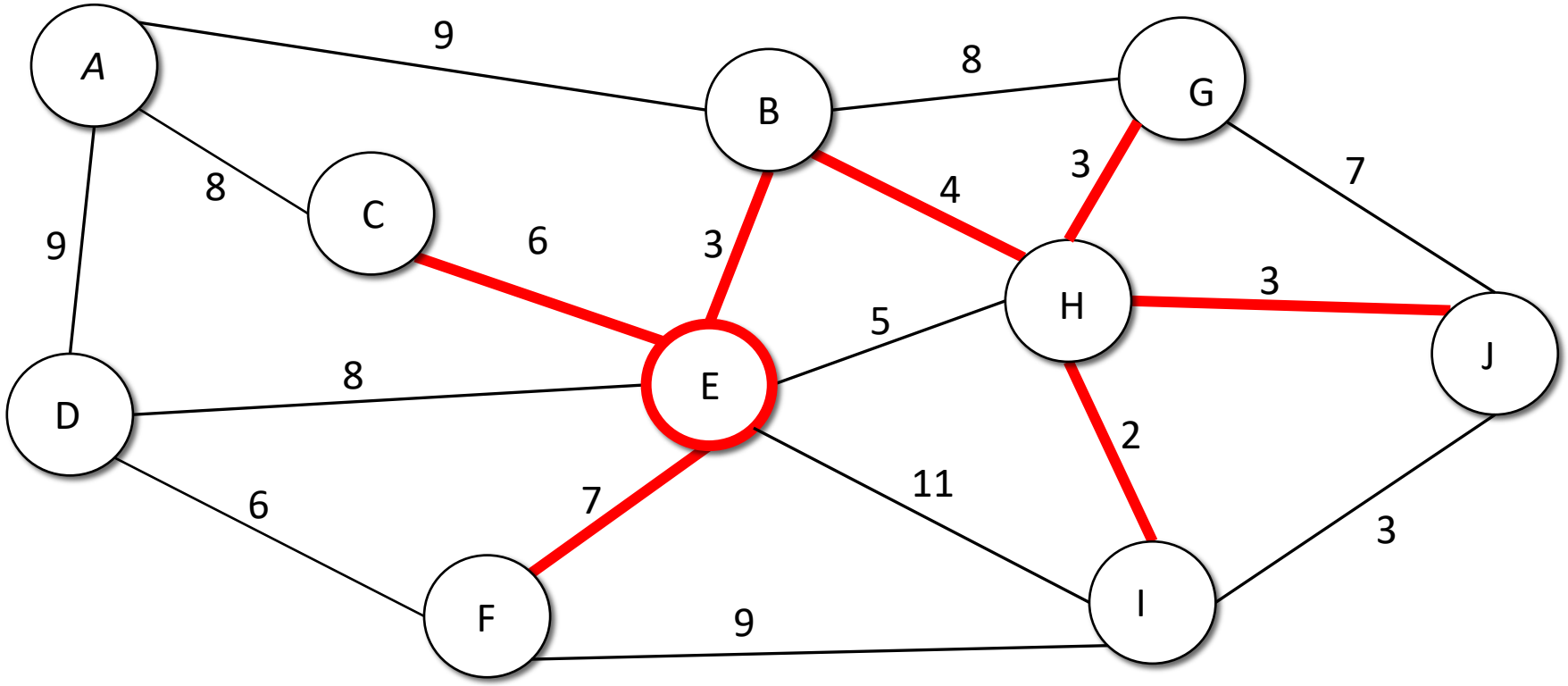
Practice Prim's Algorithm



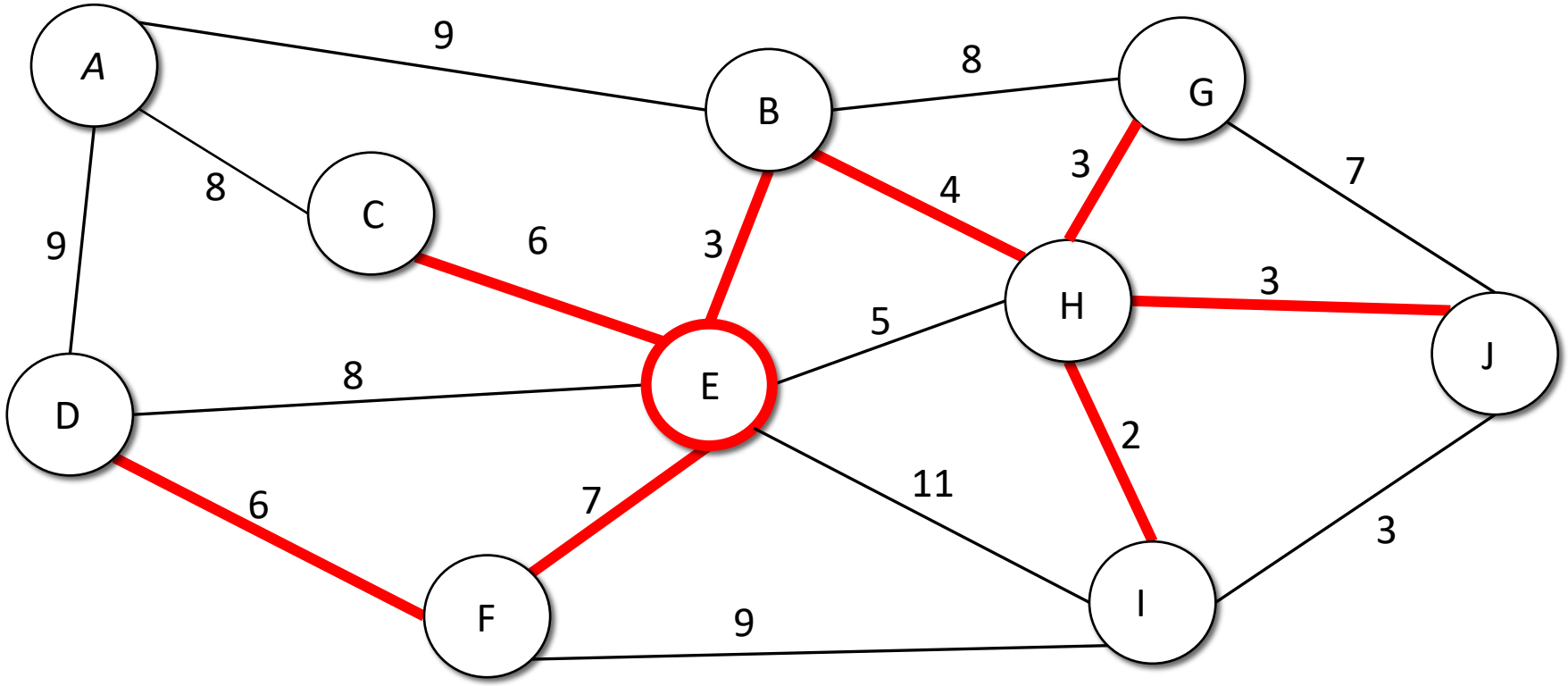
Practice Prim's Algorithm



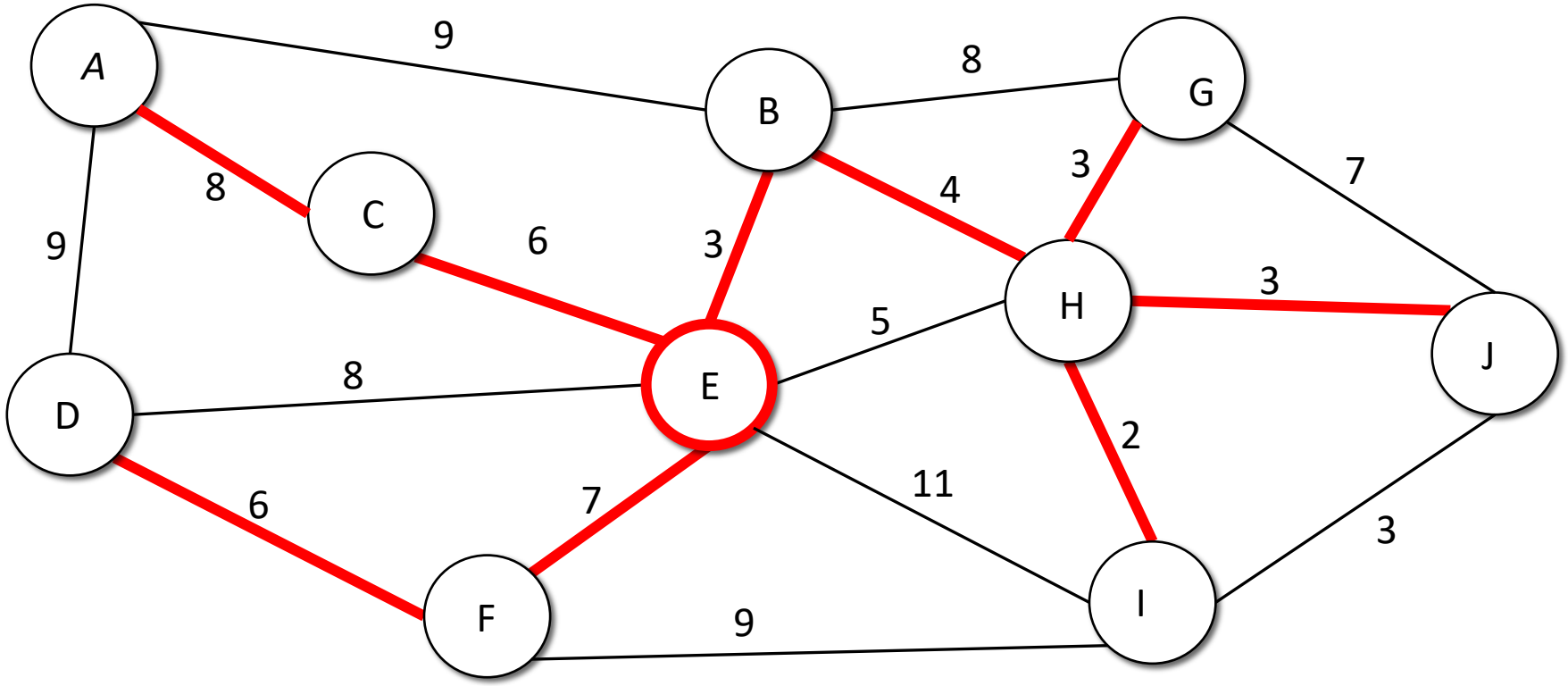
Practice Prim's Algorithm



Practice Prim's Algorithm



Practice Prim's Algorithm



Dijkstra Shortest Path Algorithm

Guaranteed to give the optimal solution

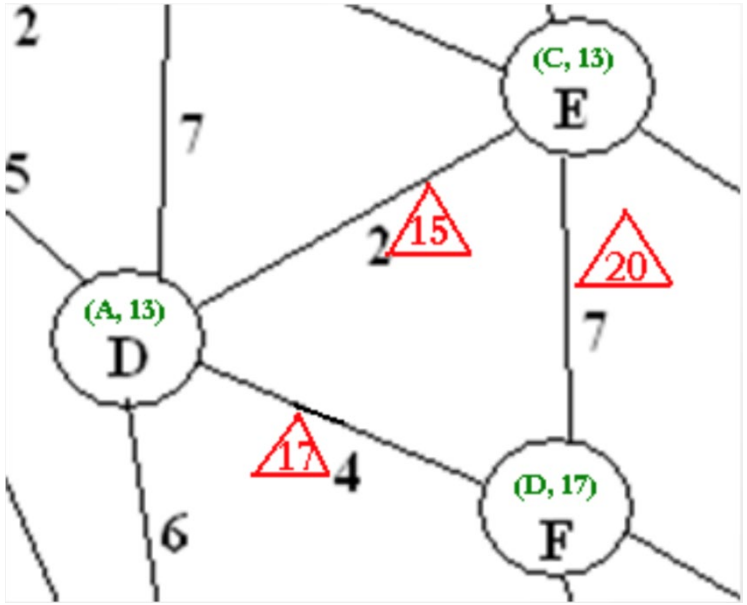
Label setting algorithm

Two types of labels

- **Temporary labels on Edges:**
  - **A representation of *total cost* to cross that edge *from the origin***
  - **Designated by a number surrounded by a triangle**

Permanent labels on Nodes:

- **Are two-part labels represented by an ordered pair in parentheses**
  - **The first part is the node from which you traveled to get to the node you are labeling**
  - **The second part is the total cost you have incurred to reach the node you are labeling from the origin**



Dijkstra's Algorithm

1) Start at the origin vertex (S) and give it a permanent label

A) This permanent label will be(-,0) to represent coming from nowhere at zero cost

B) All other edges and vertices are unlabeled

2) For each permanently labeled vertex

A) Give each edge connected from it to an unlabeled vertex a temporary label denoting the total least cost to travel across that edge *from the origin*

B) Choose the edge with the smallest total cost (the smallest temporary label) and permanently label its connected vertex with the name of the vertex you traveled from and the cost to get there *from the origin*

C) Identify that edge as possibly being part of the final path solution (highlight the edge)

D) Eliminate temporary labels on edges between permanently labeled vertices

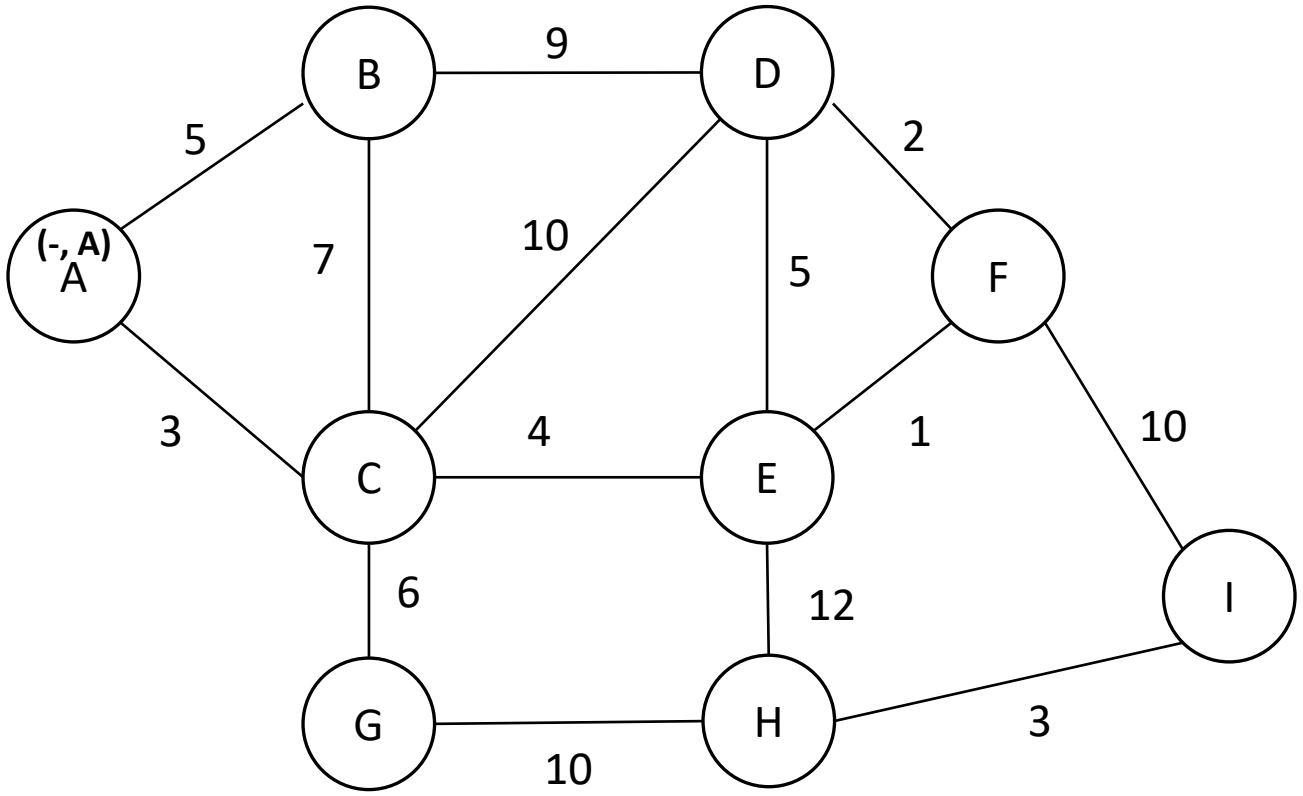
3) If vertex T has been permanently labeled stop since a shortest path from S to T has been found. If vertex T has not been labeled go to step 2.

4) When T is reached use the permanent labels on the vertices to work backwards to the origin defining the shortest path

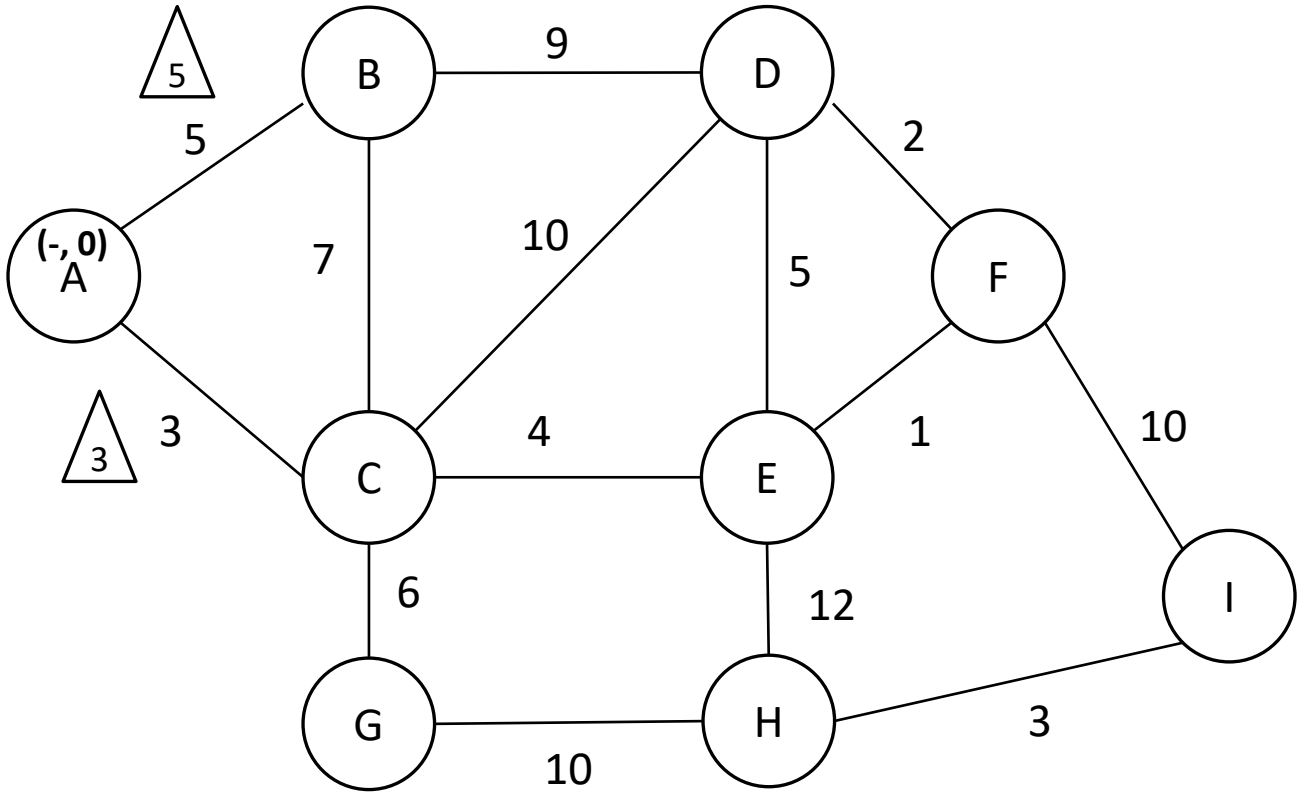
BREAK TIES ARBITRARILY. IT DOESN'T MATTER WHICH VERTEX YOU LABEL.

You might not get the same path (alternate optima) but you will get the same cost

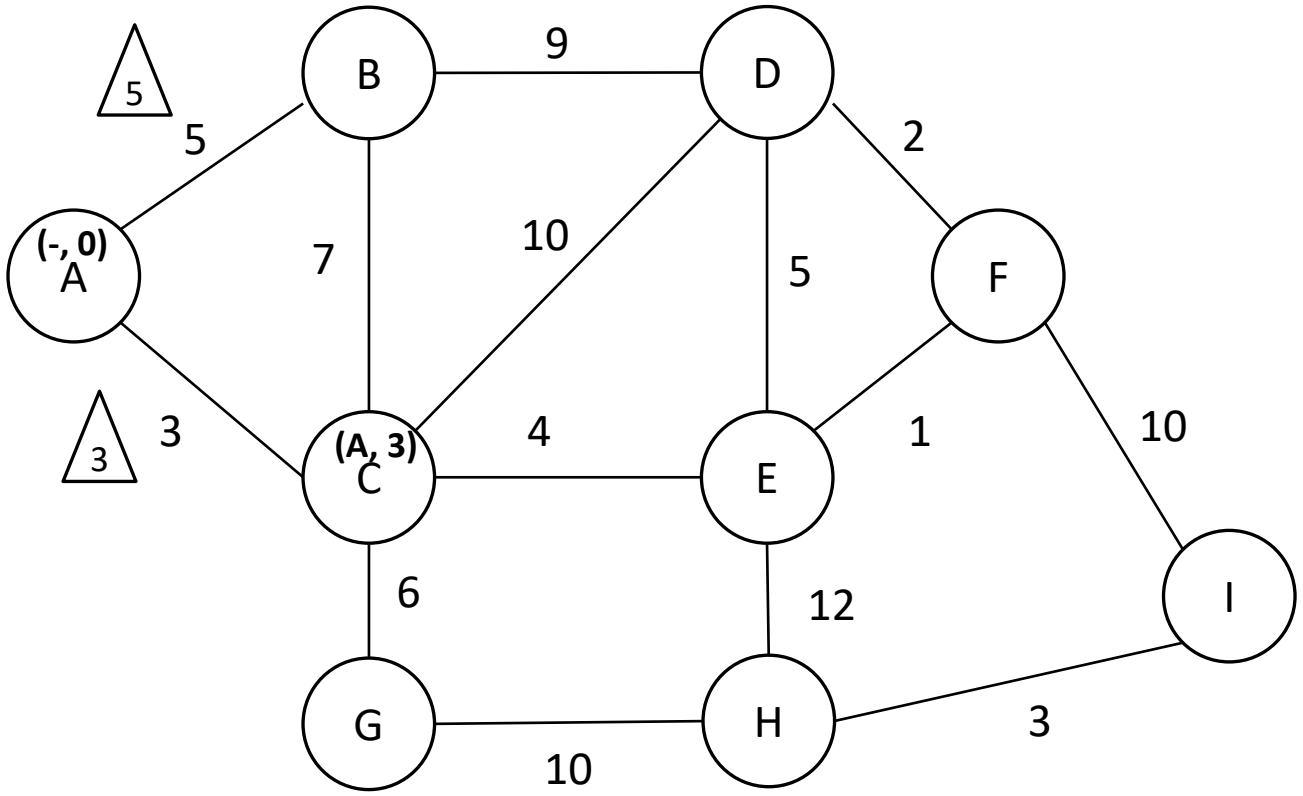
Dijkstra Shortest Path Algorithm: A to H



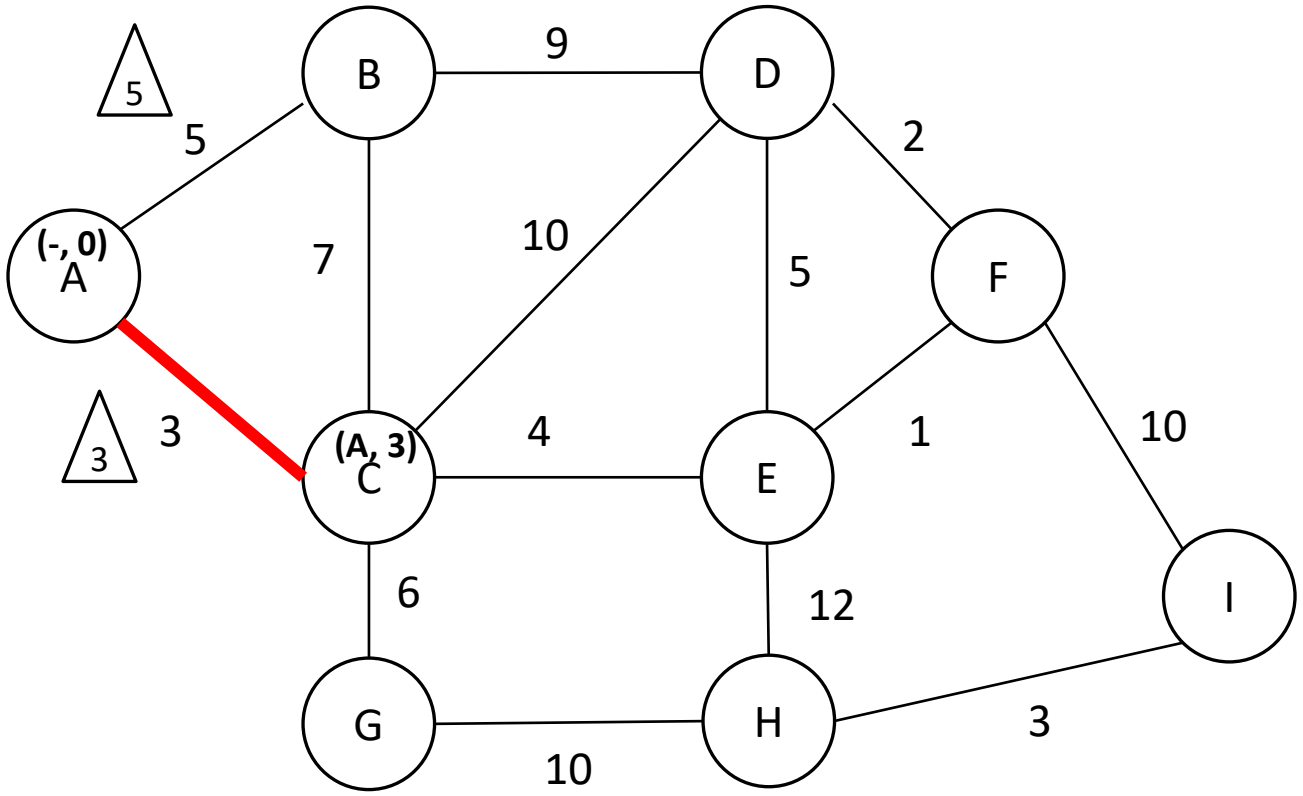
Dijkstra Shortest Path Algorithm: A to H



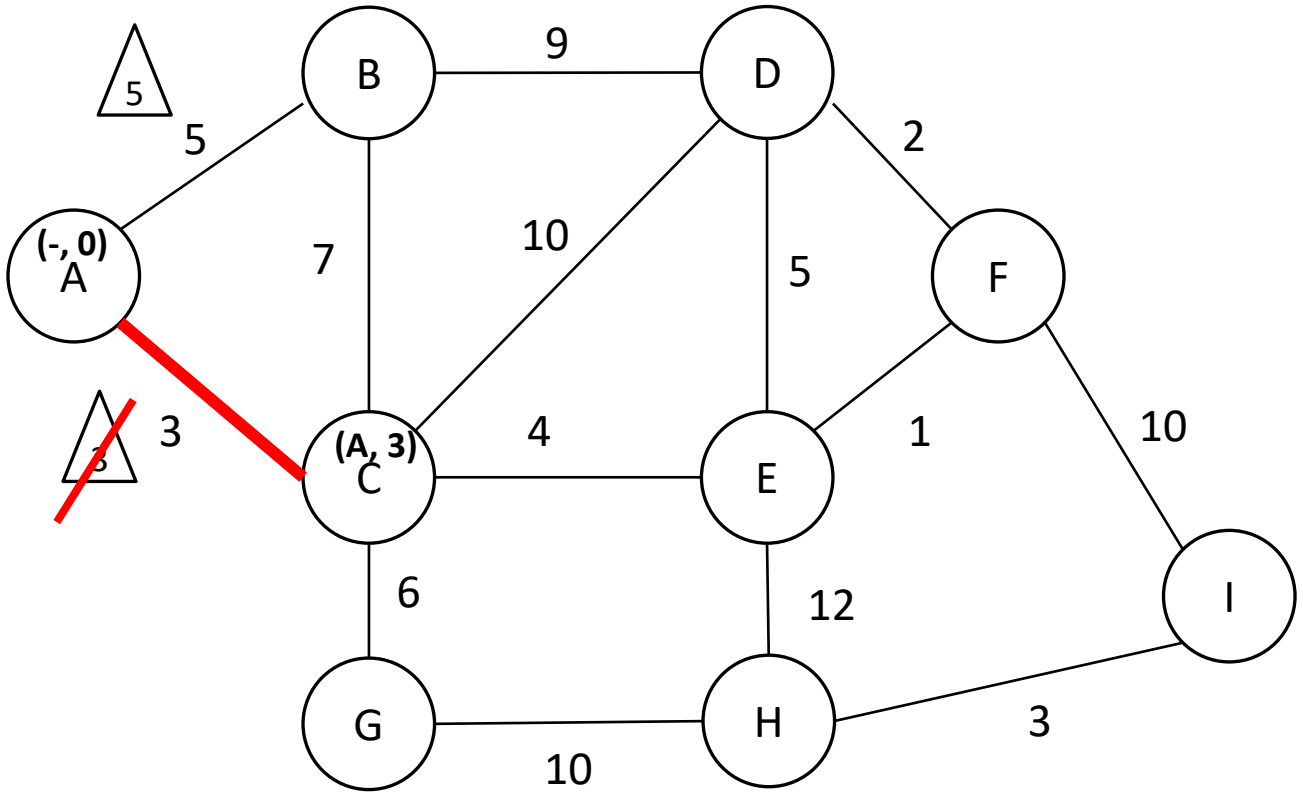
Dijkstra Shortest Path Algorithm: A to H



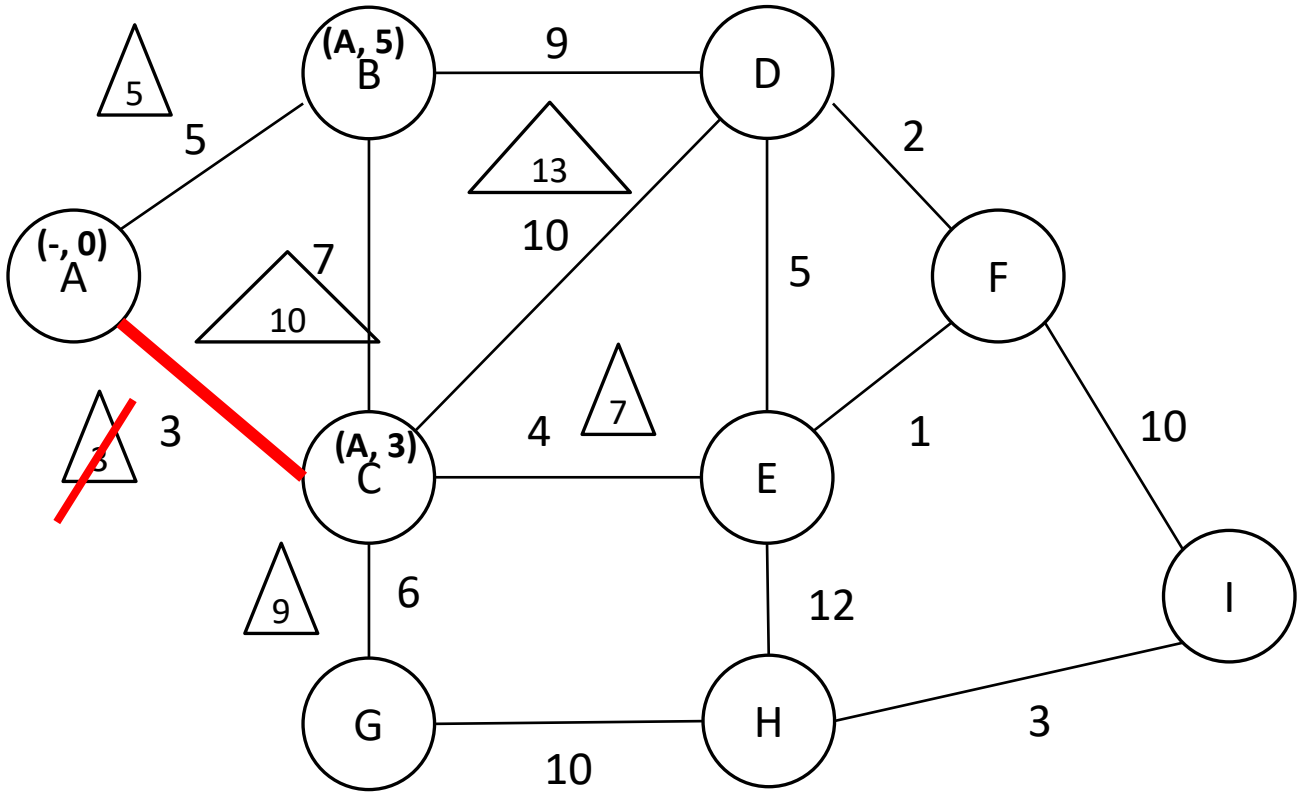
Dijkstra Shortest Path Algorithm: A to H



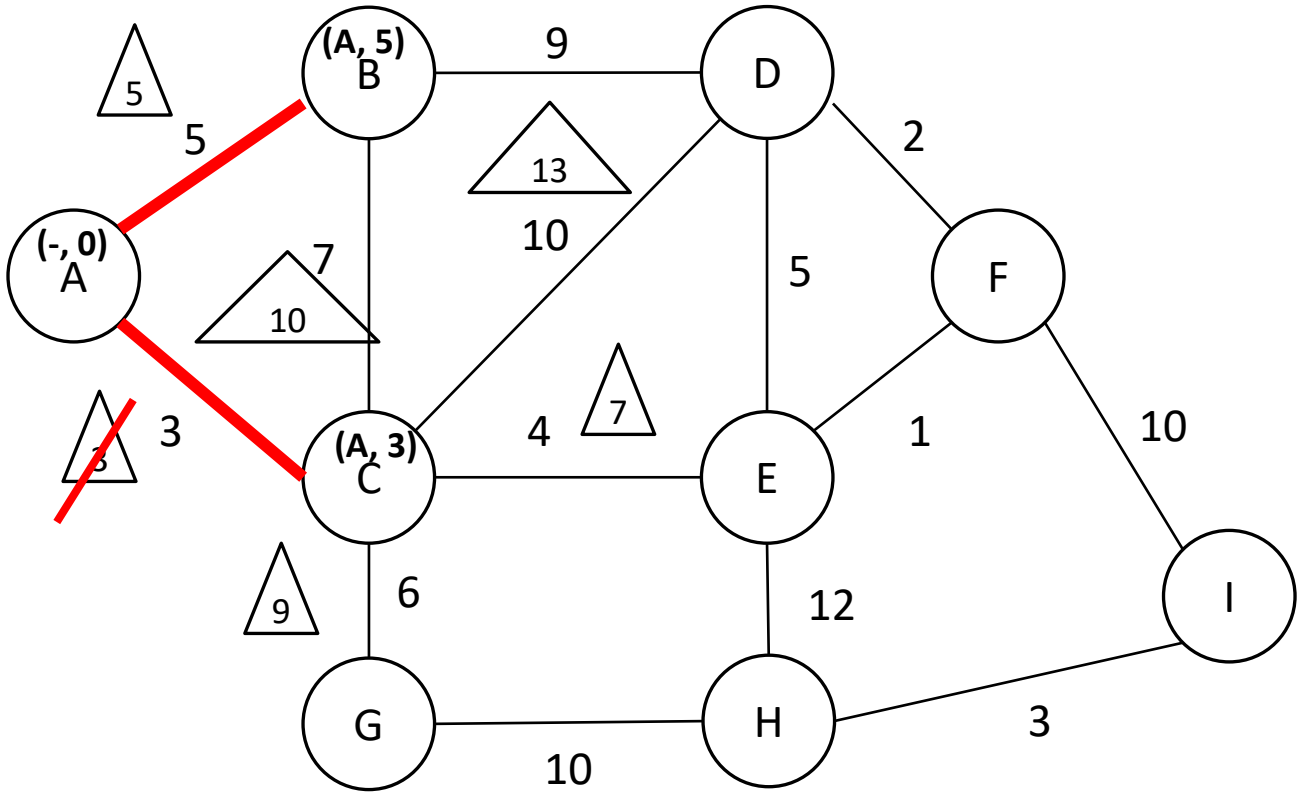
Dijkstra Shortest Path Algorithm: A to H



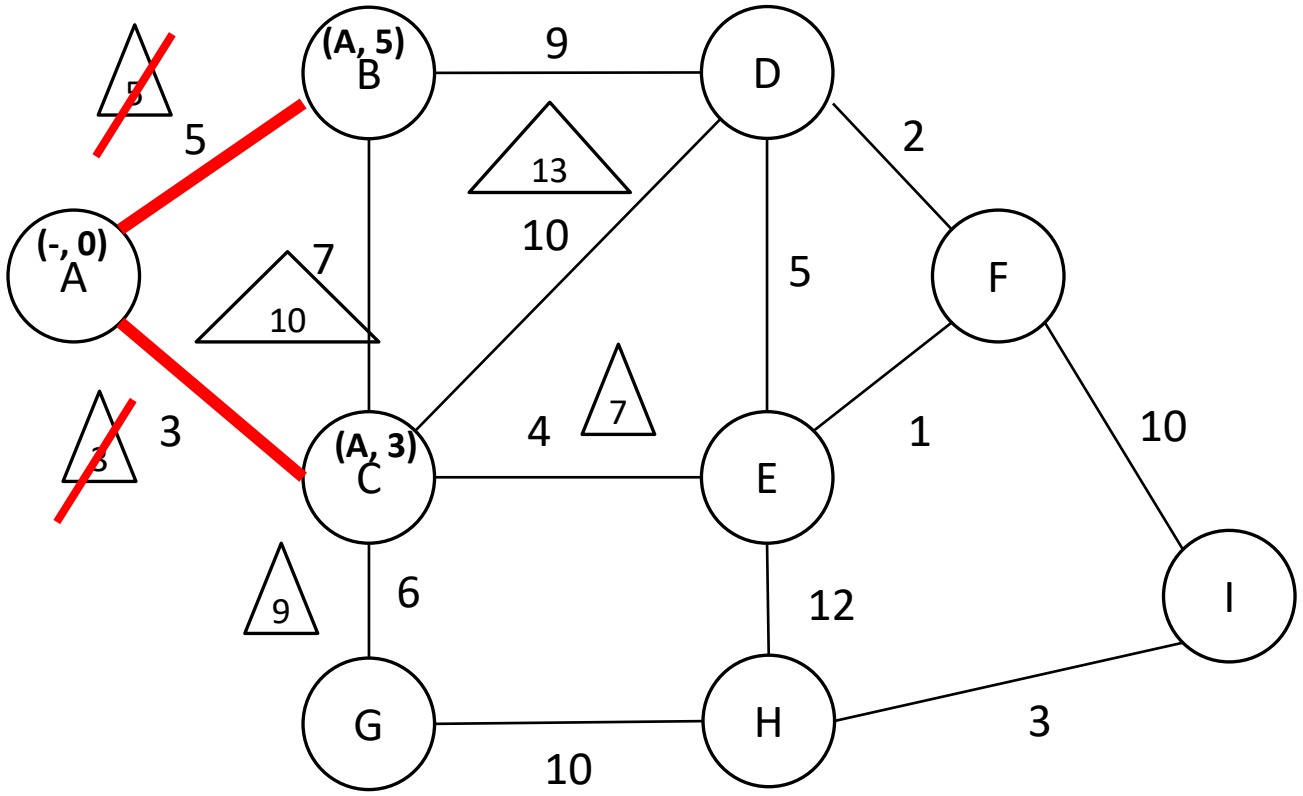
Dijkstra Shortest Path Algorithm: A to H



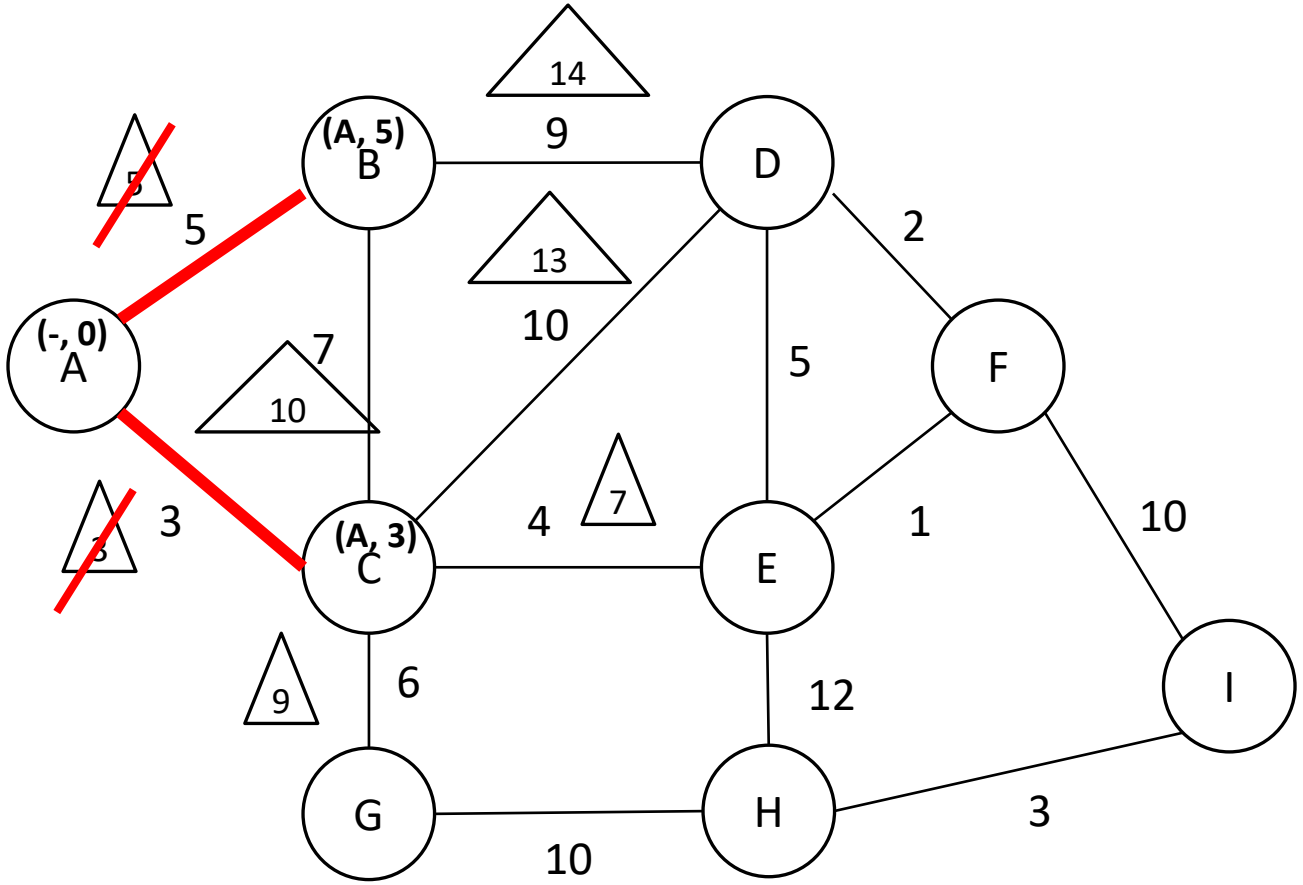
Dijkstra Shortest Path Algorithm: A to H



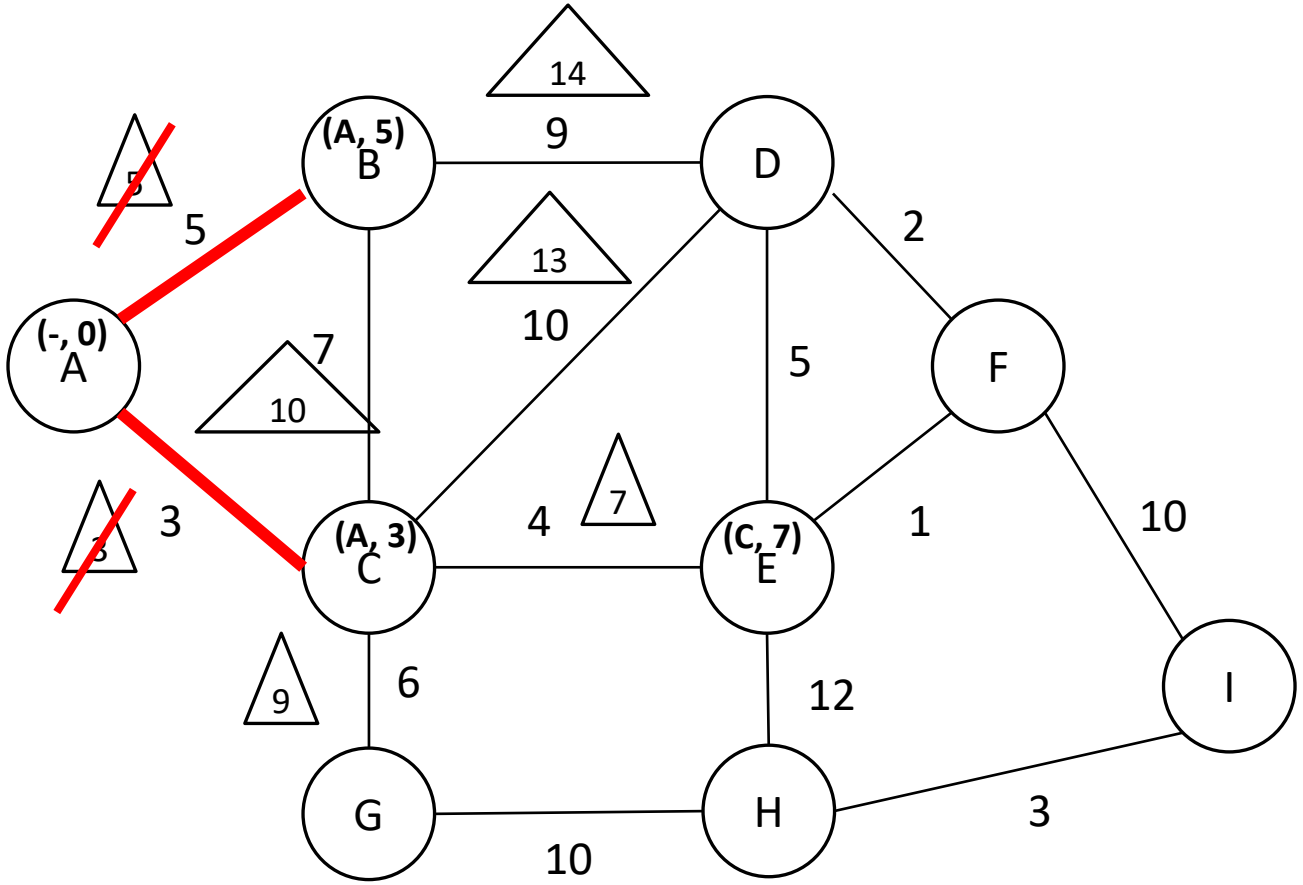
Dijkstra Shortest Path Algorithm: A to H



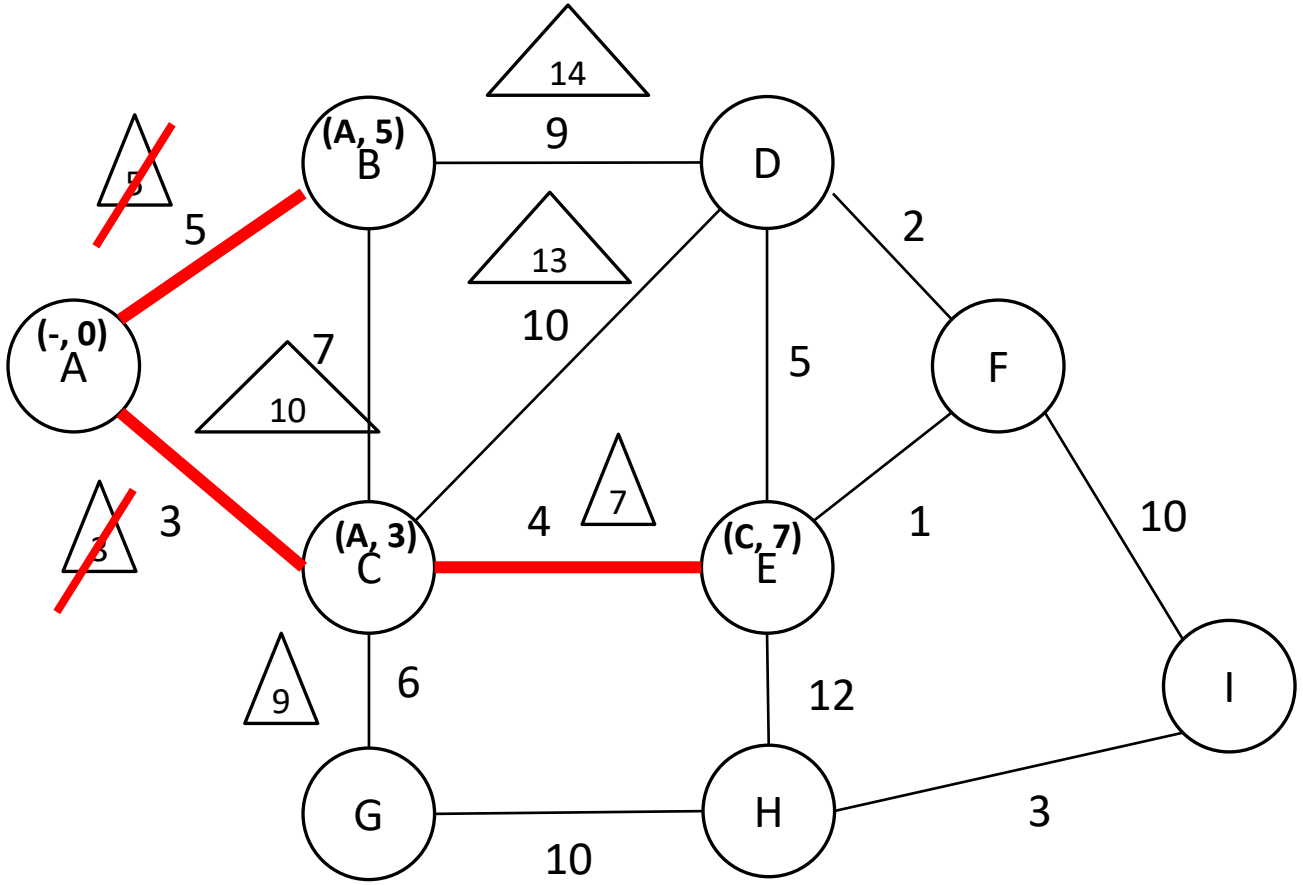
Dijkstra Shortest Path Algorithm: A to H



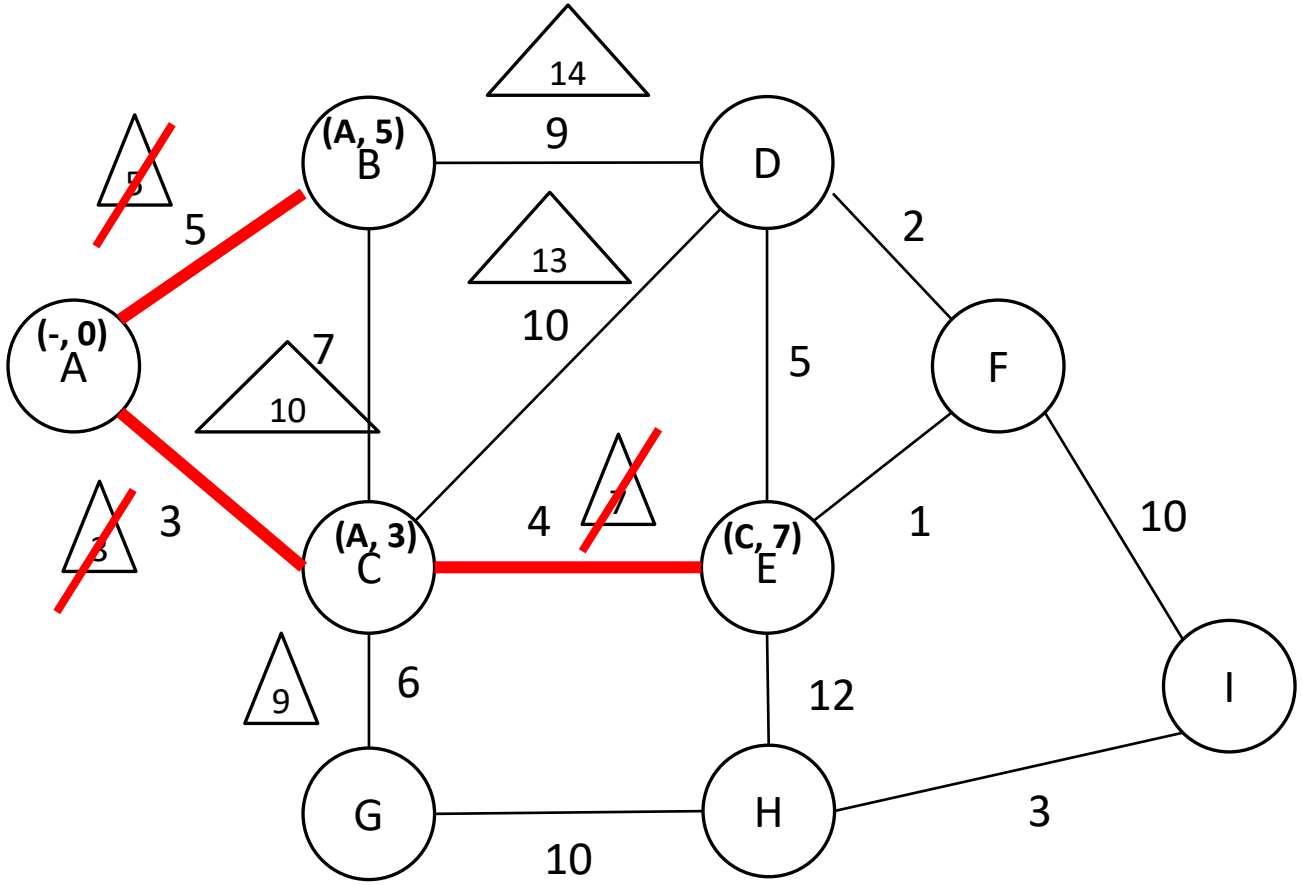
Dijkstra Shortest Path Algorithm: A to H



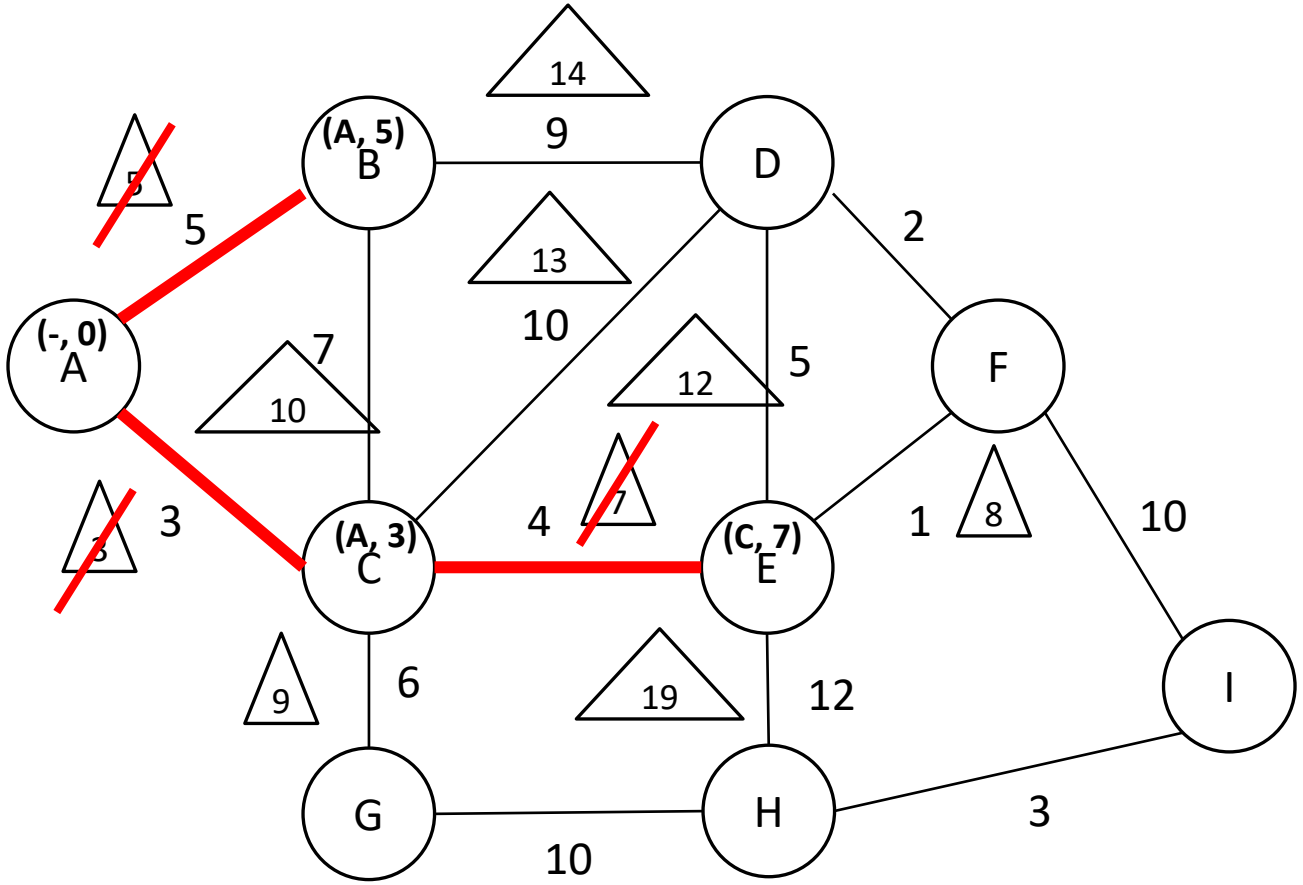
Dijkstra Shortest Path Algorithm: A to H



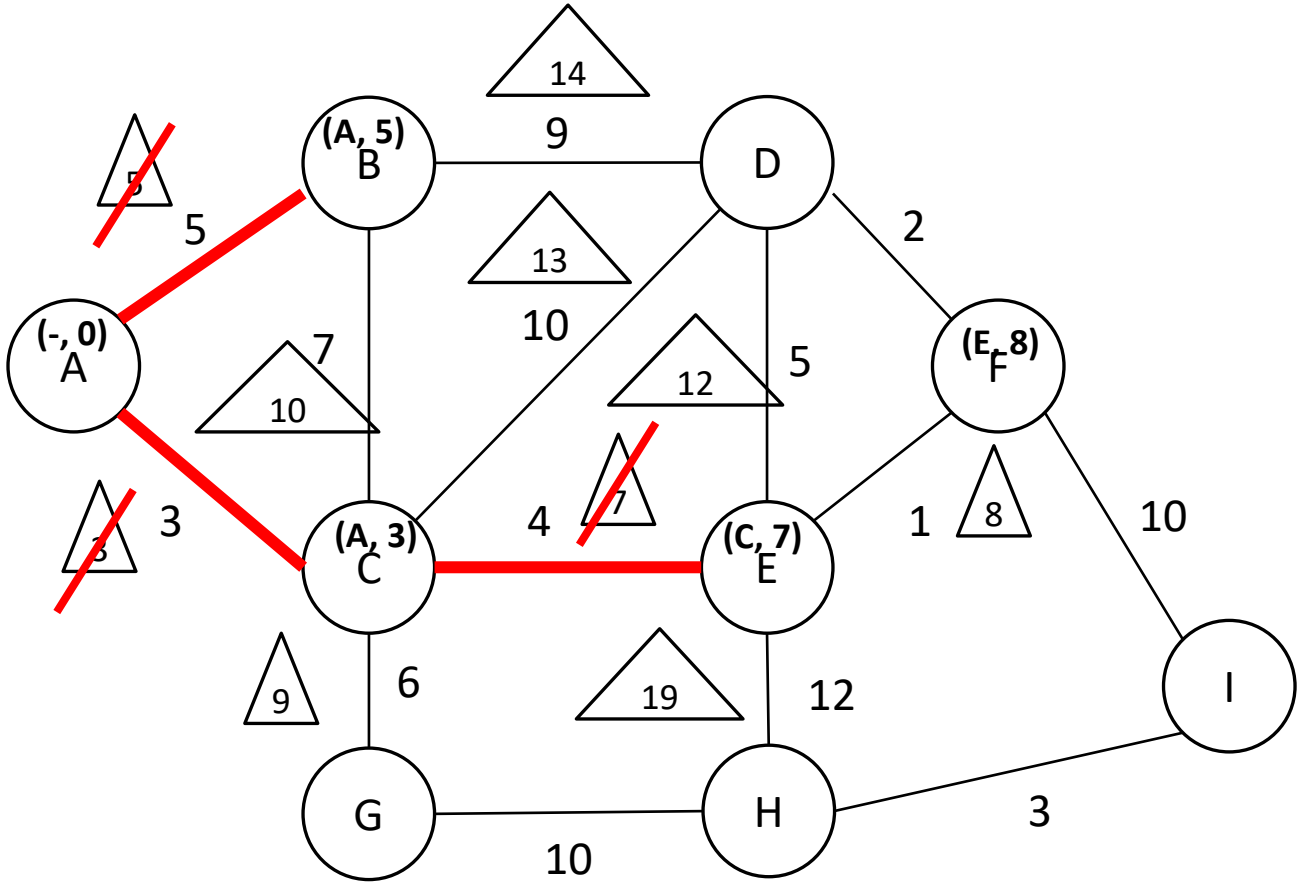
Dijkstra Shortest Path Algorithm: A to H



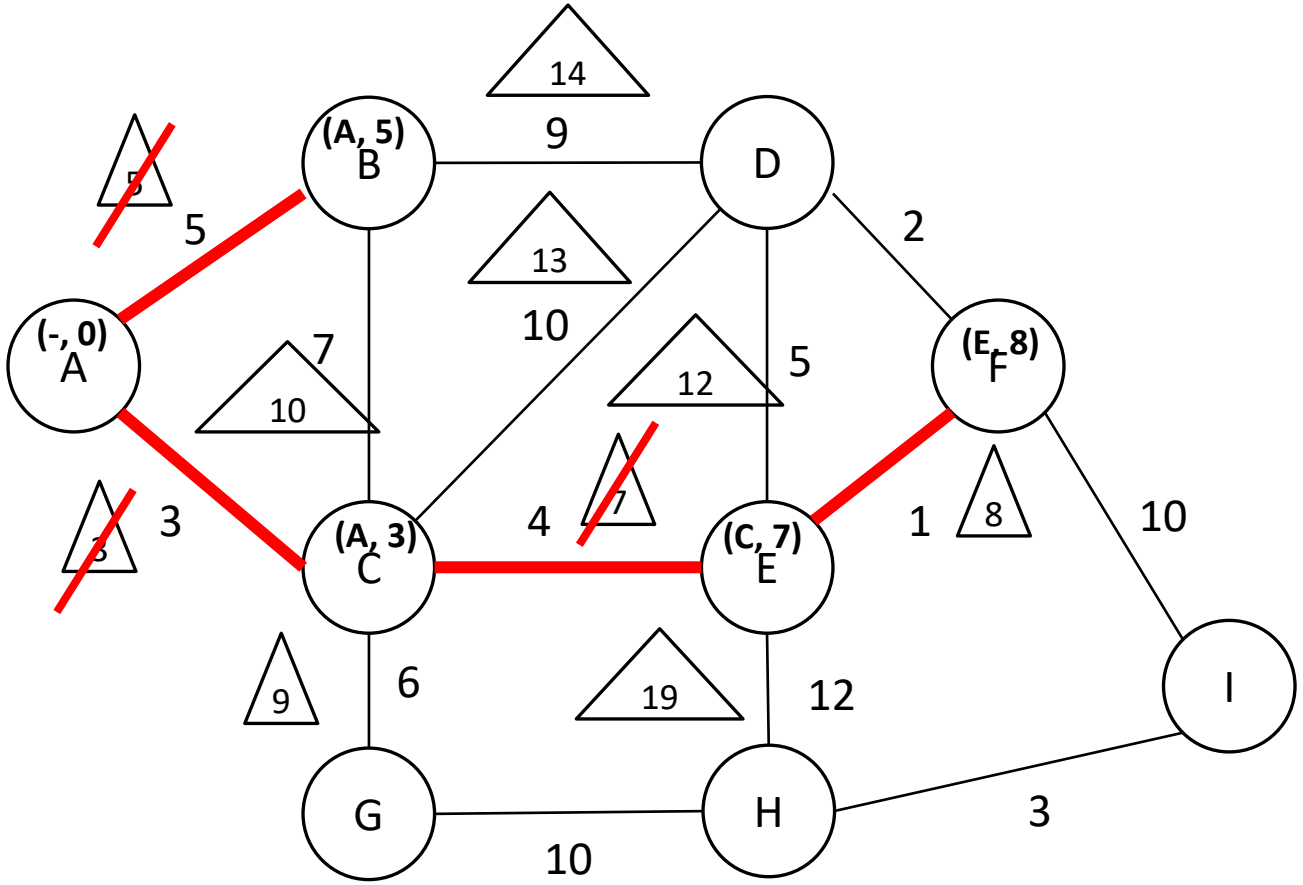
Dijkstra Shortest Path Algorithm: A to H



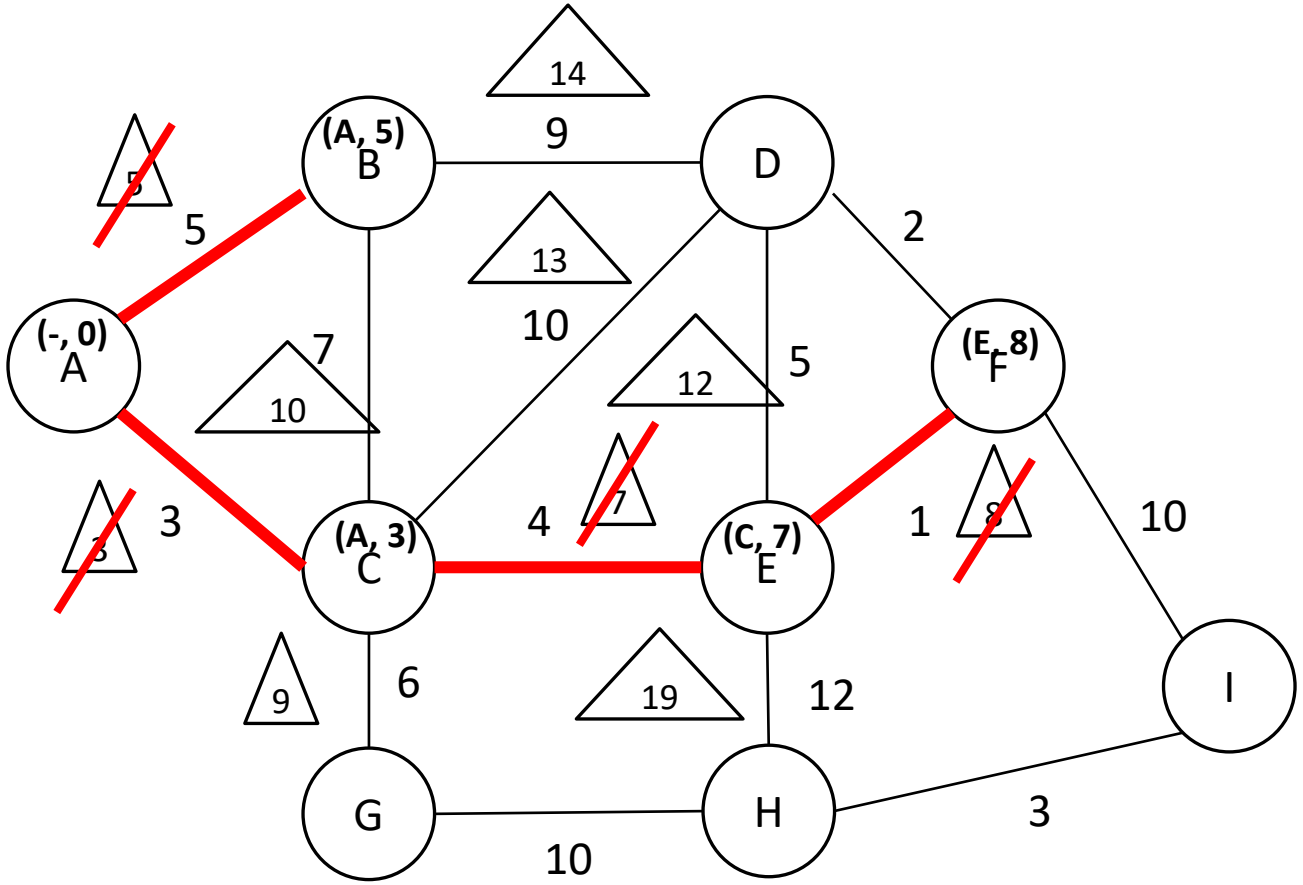
Dijkstra Shortest Path Algorithm: A to H



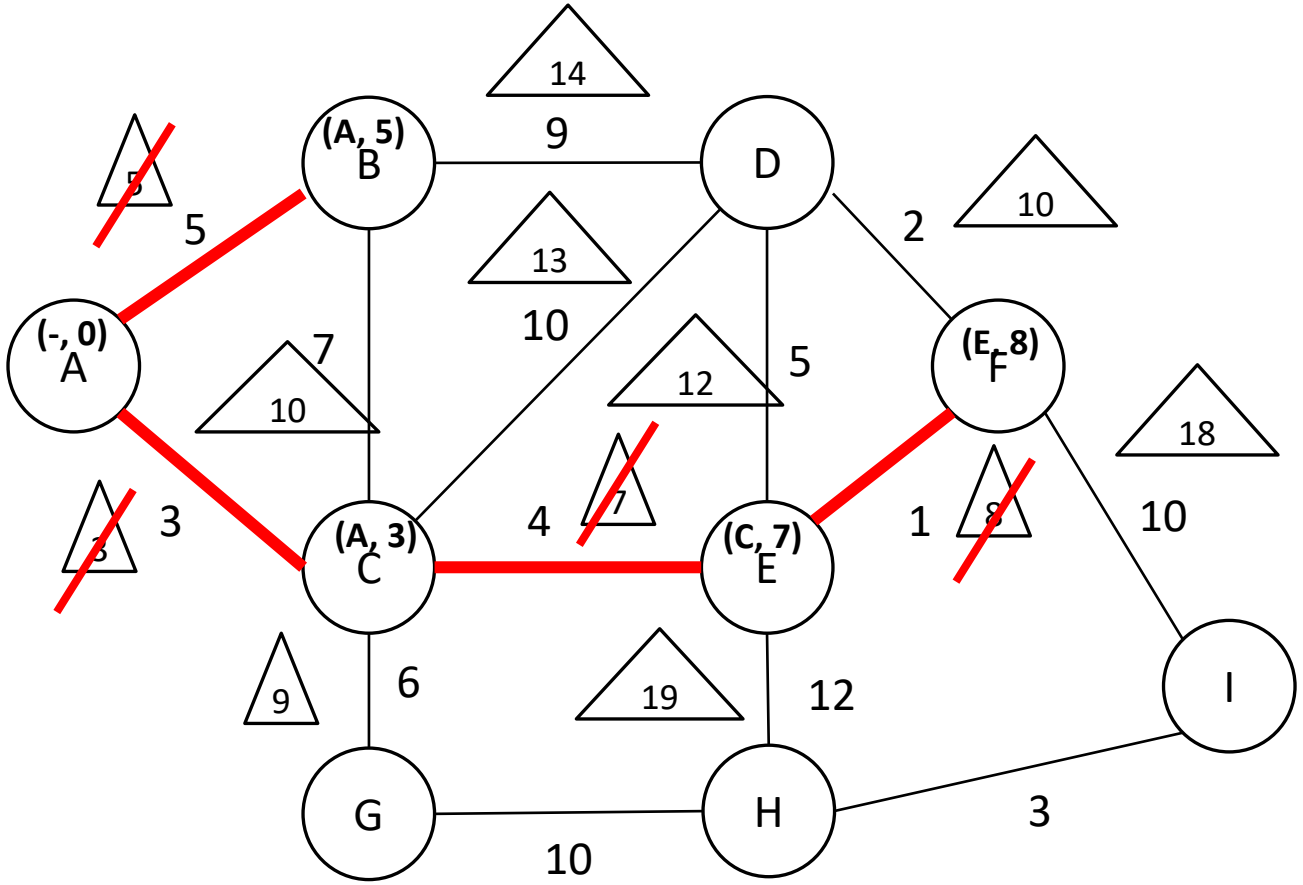
Dijkstra Shortest Path Algorithm: A to H



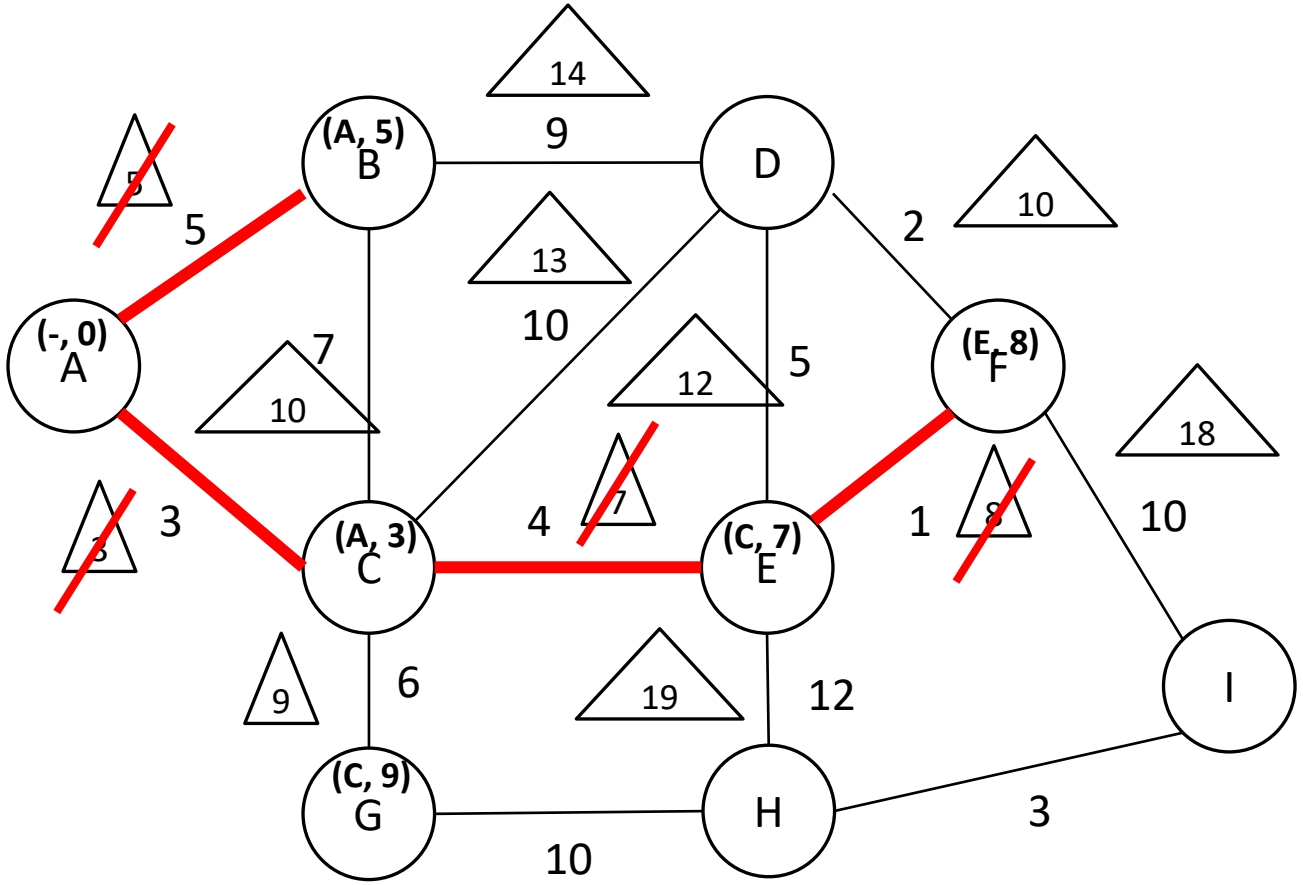
Dijkstra Shortest Path Algorithm: A to H



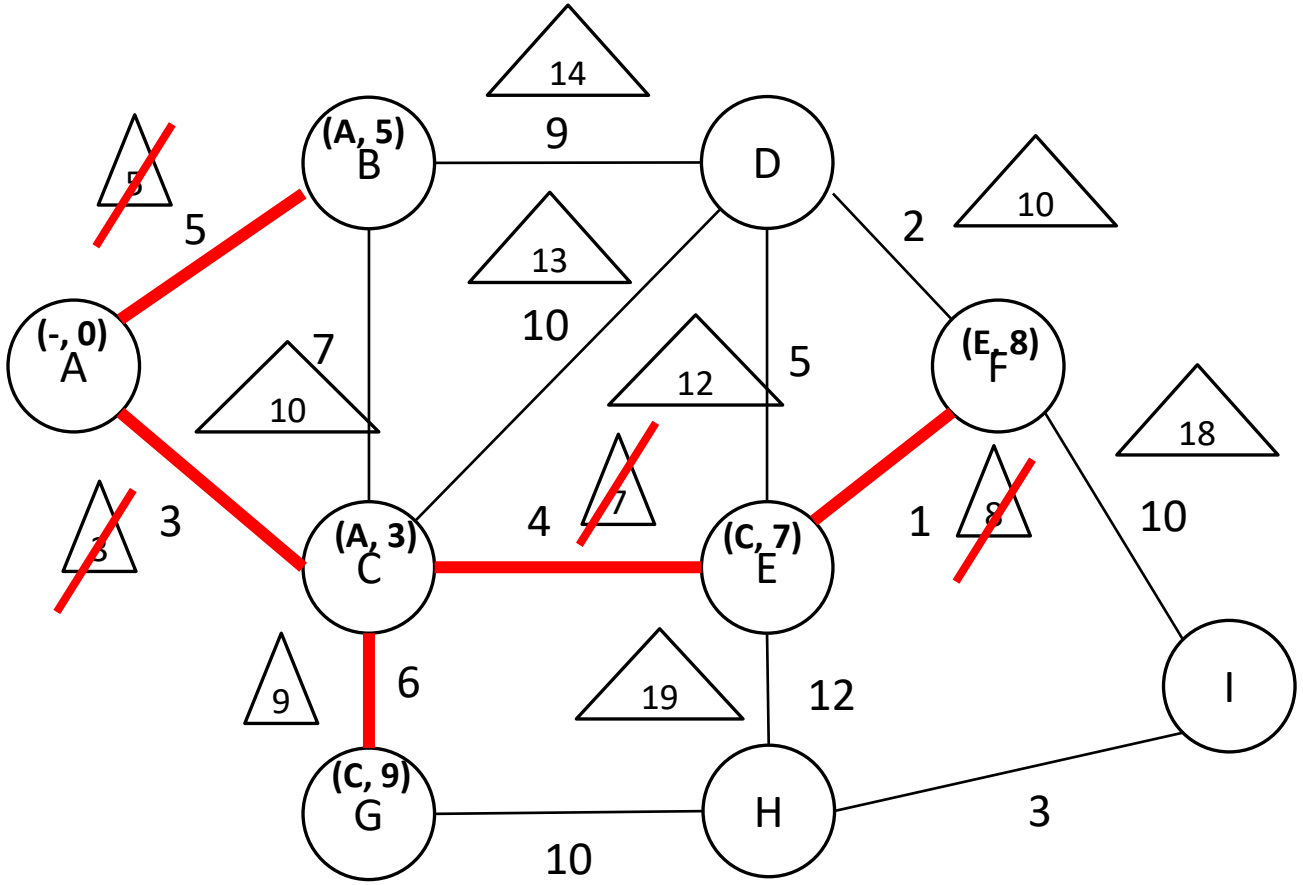
Dijkstra Shortest Path Algorithm: A to H



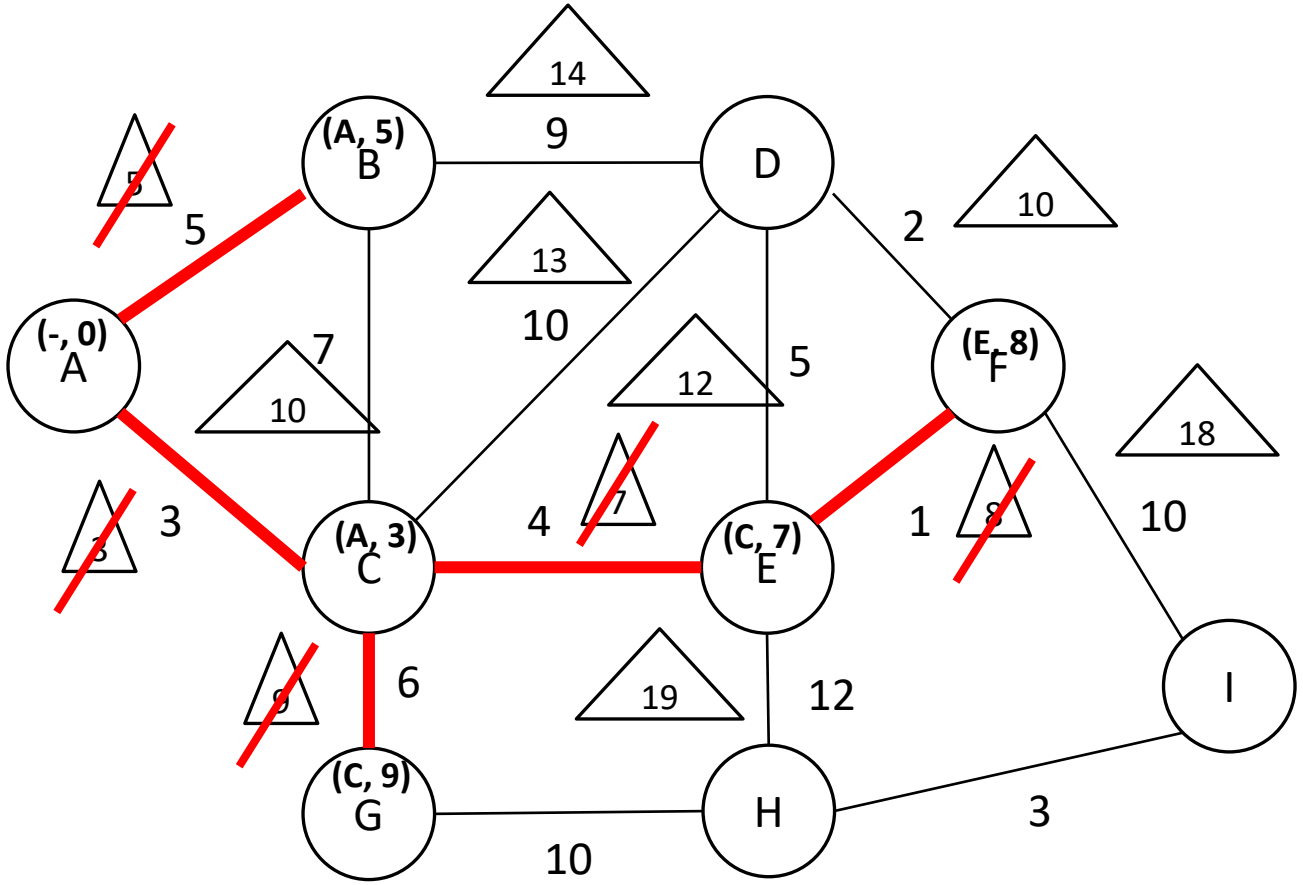
Dijkstra Shortest Path Algorithm: A to H



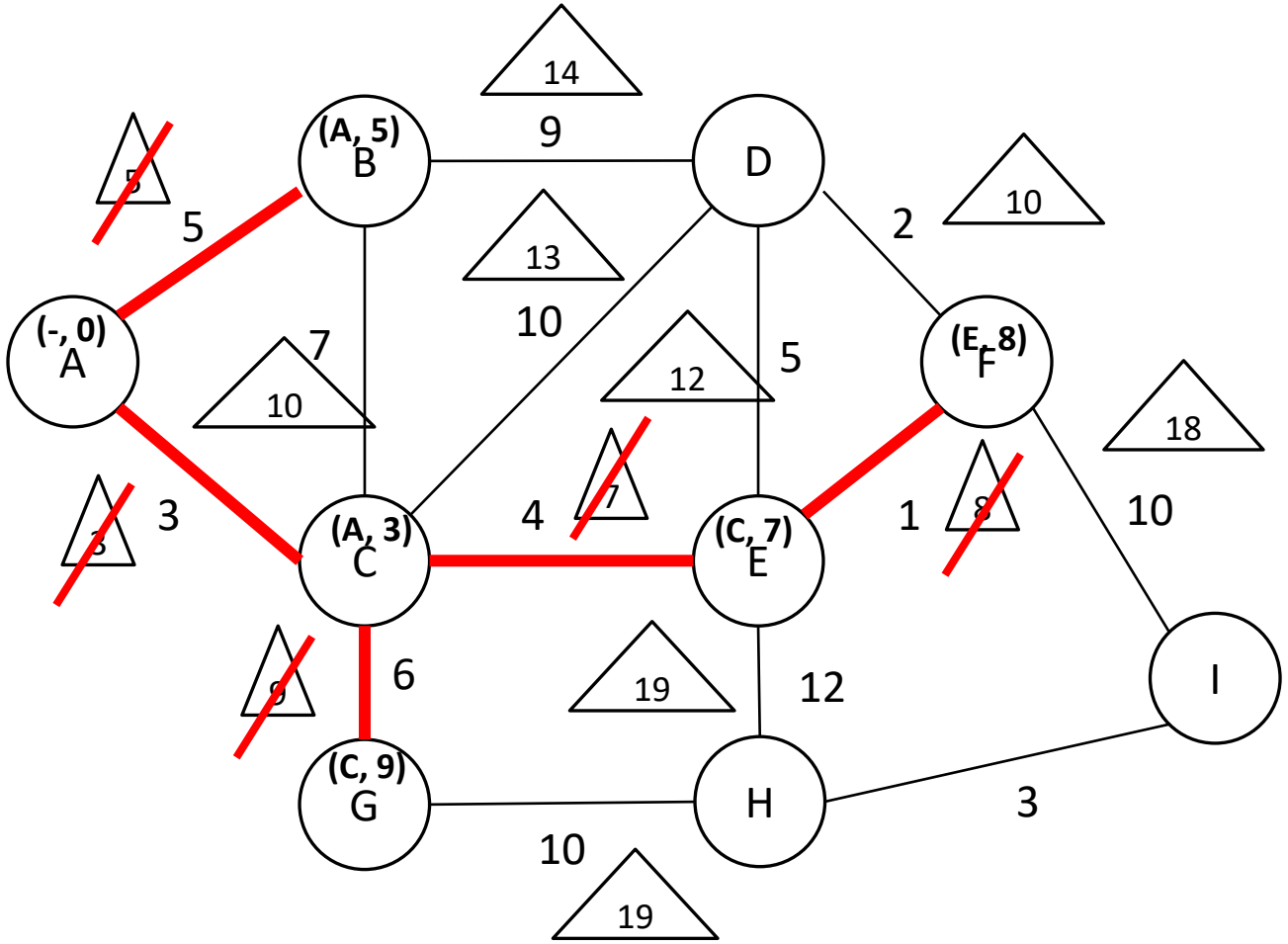
Dijkstra Shortest Path Algorithm: A to H



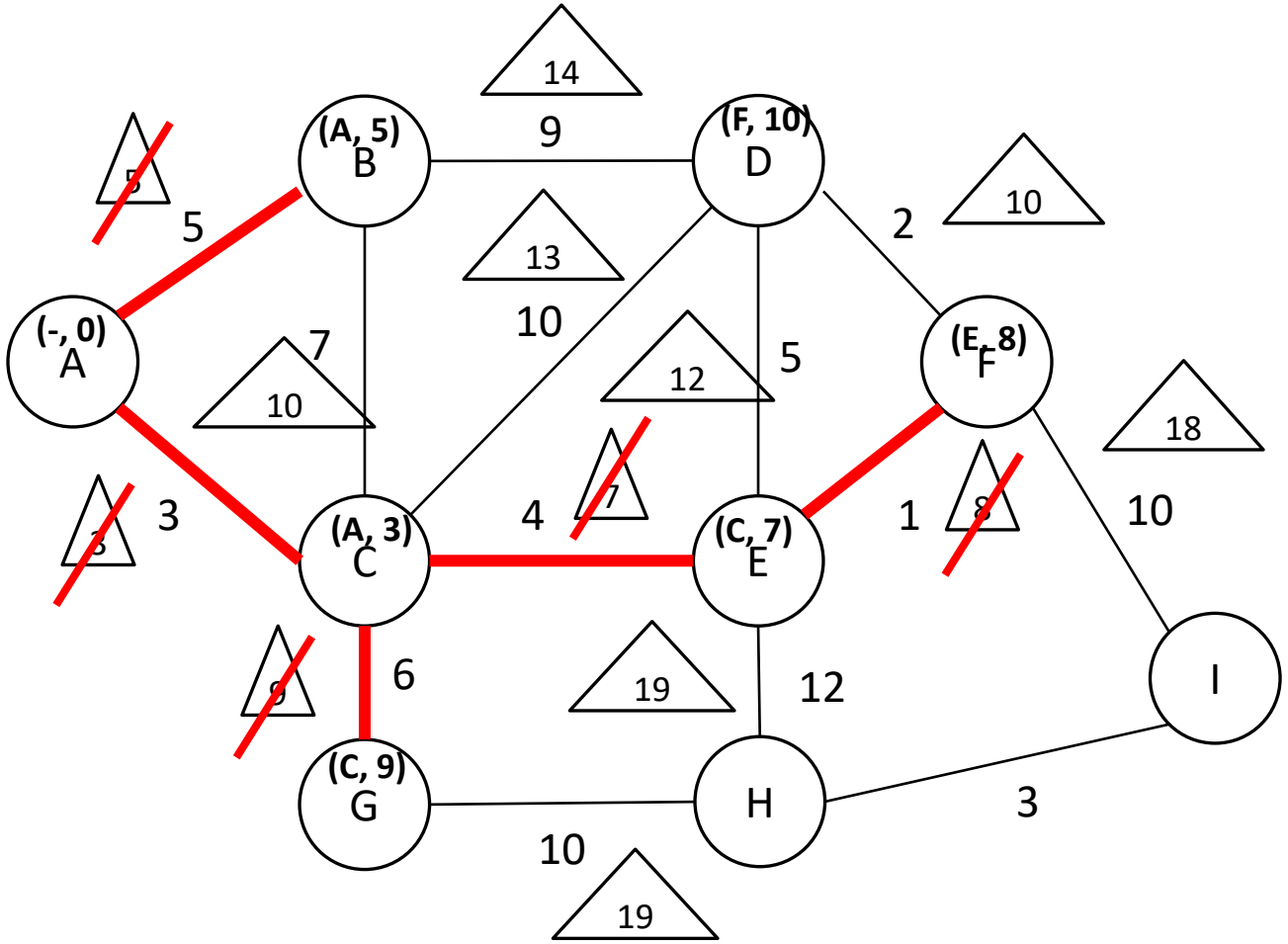
Dijkstra Shortest Path Algorithm: A to H



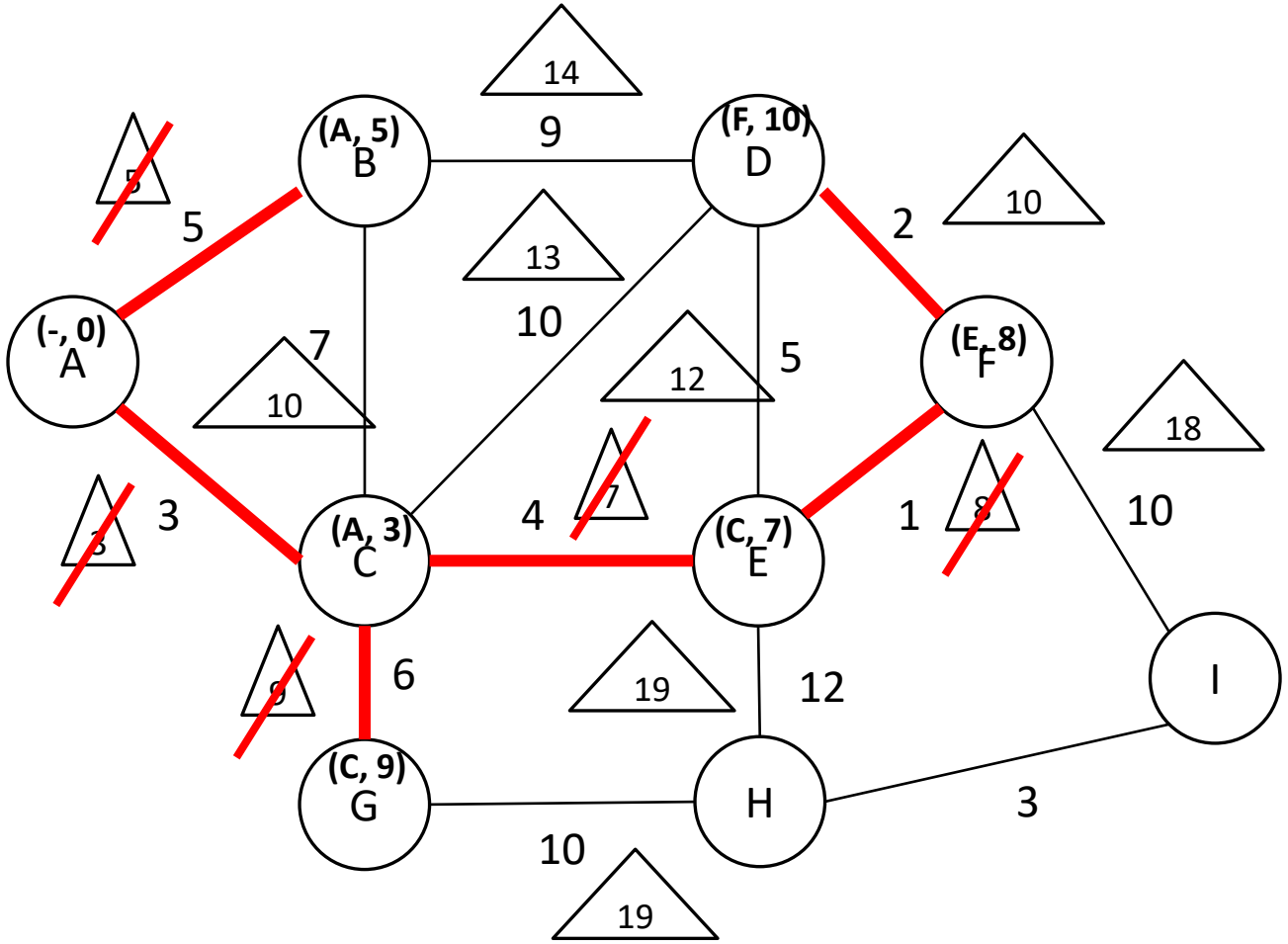
Dijkstra Shortest Path Algorithm: A to H



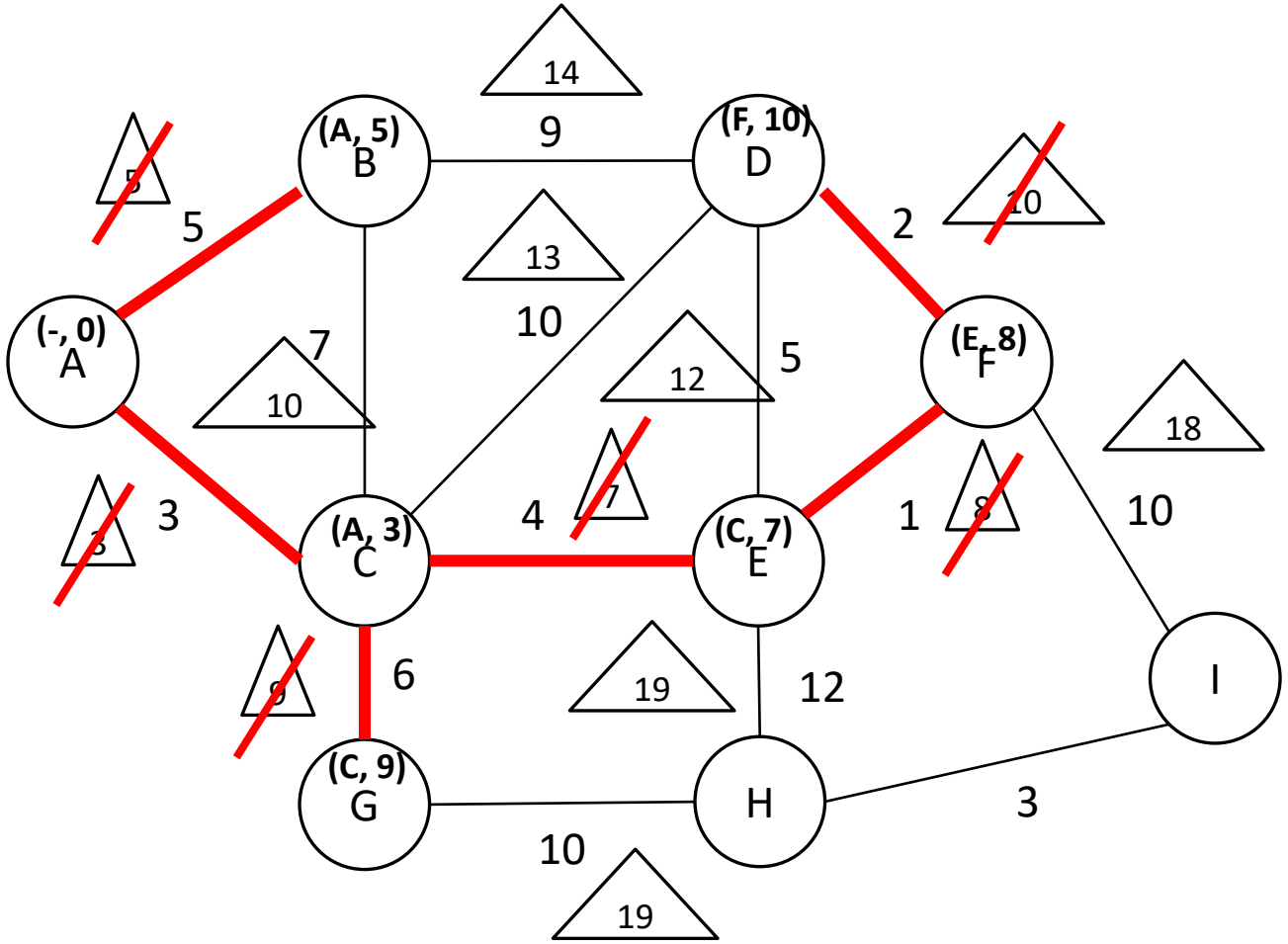
Dijkstra Shortest Path Algorithm: A to H



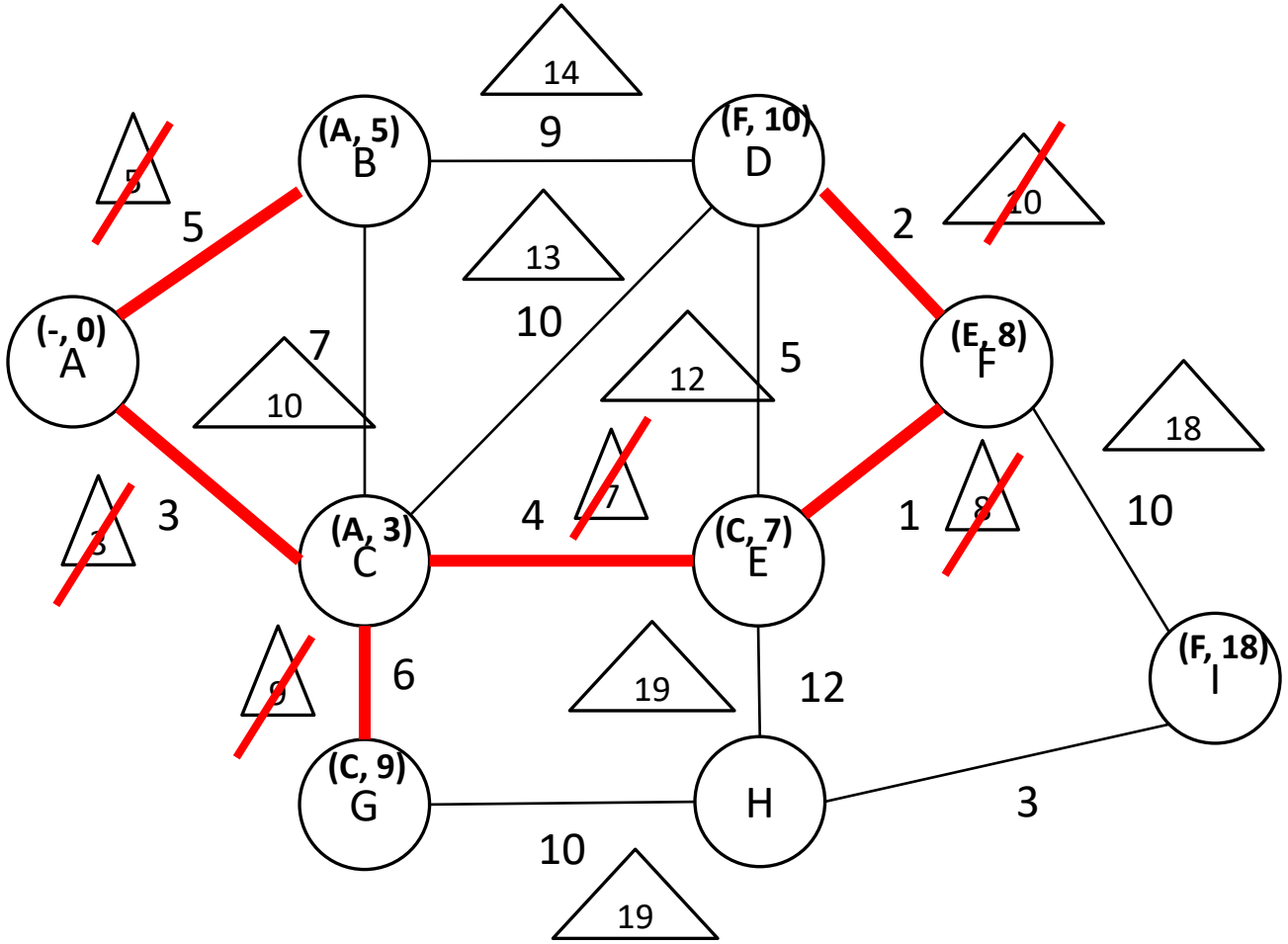
Dijkstra Shortest Path Algorithm: A to H



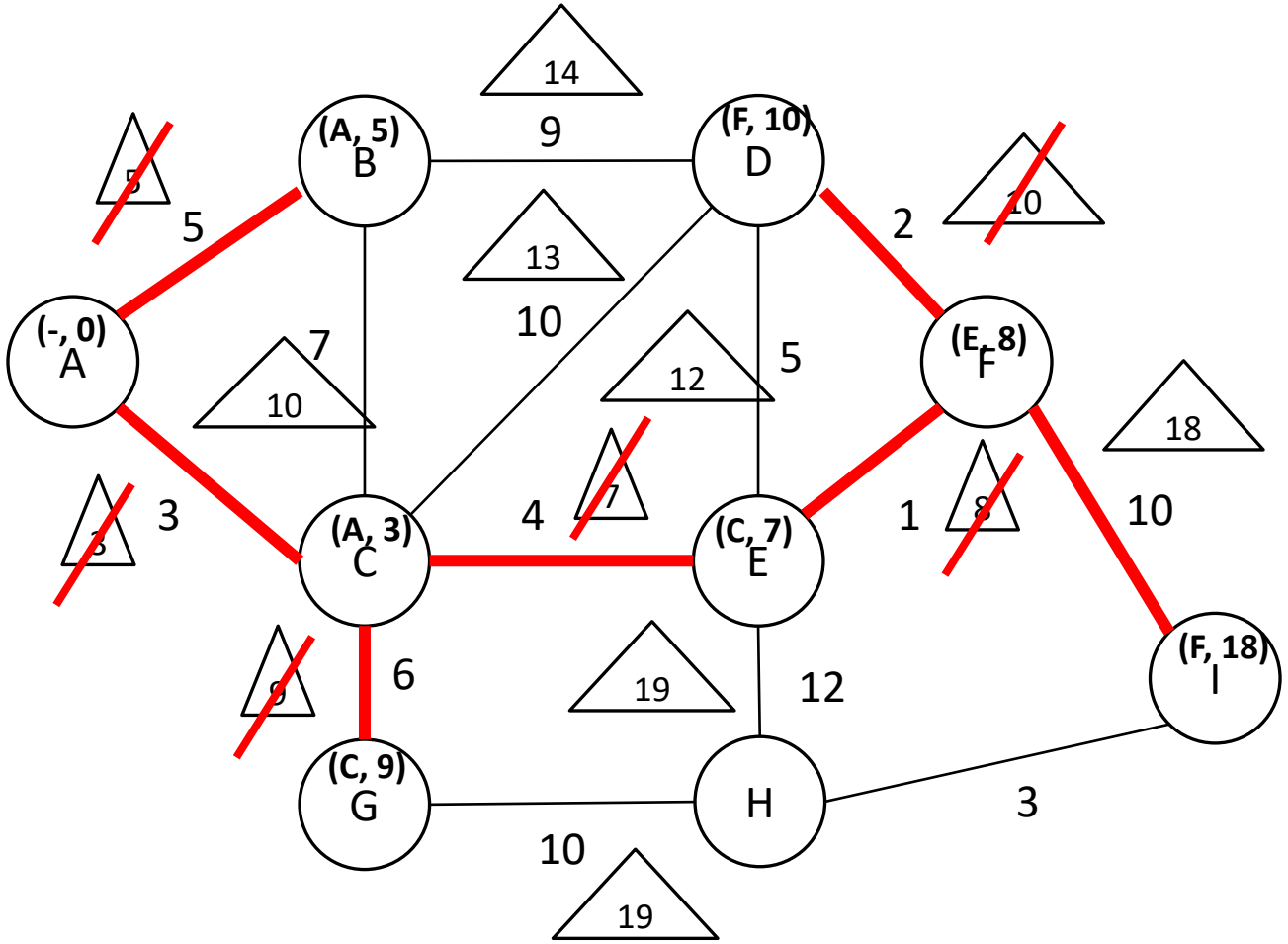
Dijkstra Shortest Path Algorithm: A to H



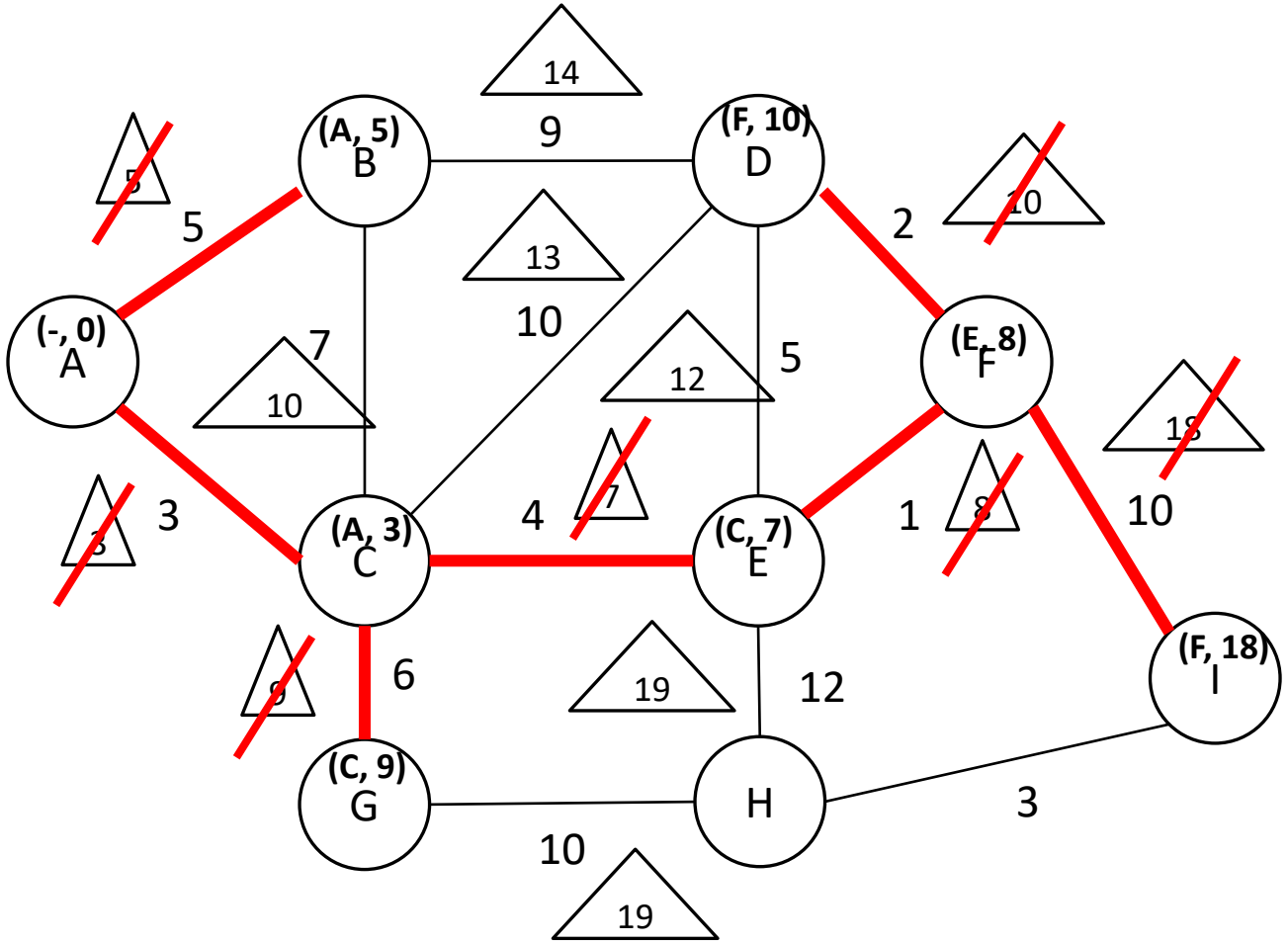
Dijkstra Shortest Path Algorithm: A to H



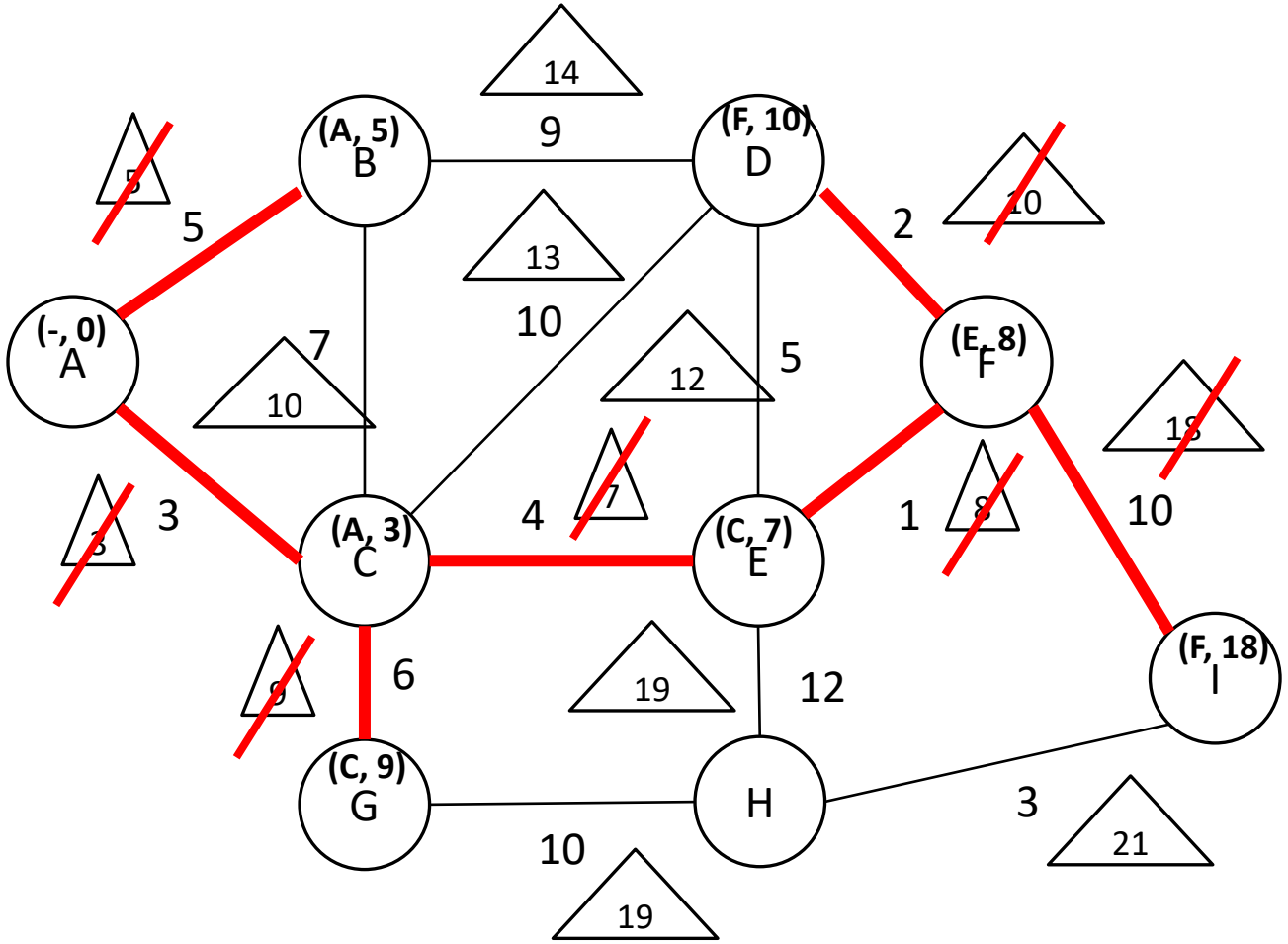
Dijkstra Shortest Path Algorithm: A to H



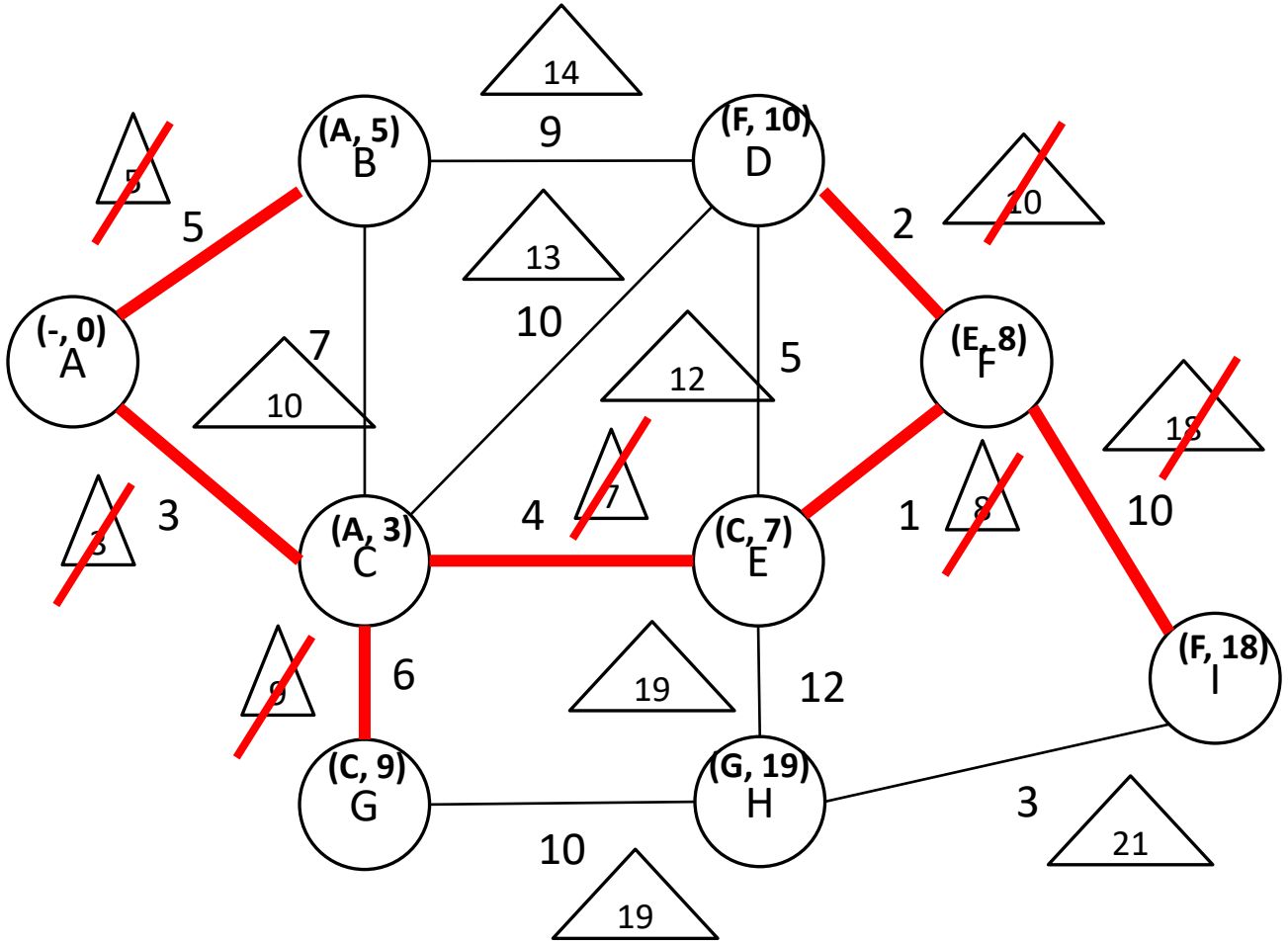
Dijkstra Shortest Path Algorithm: A to H



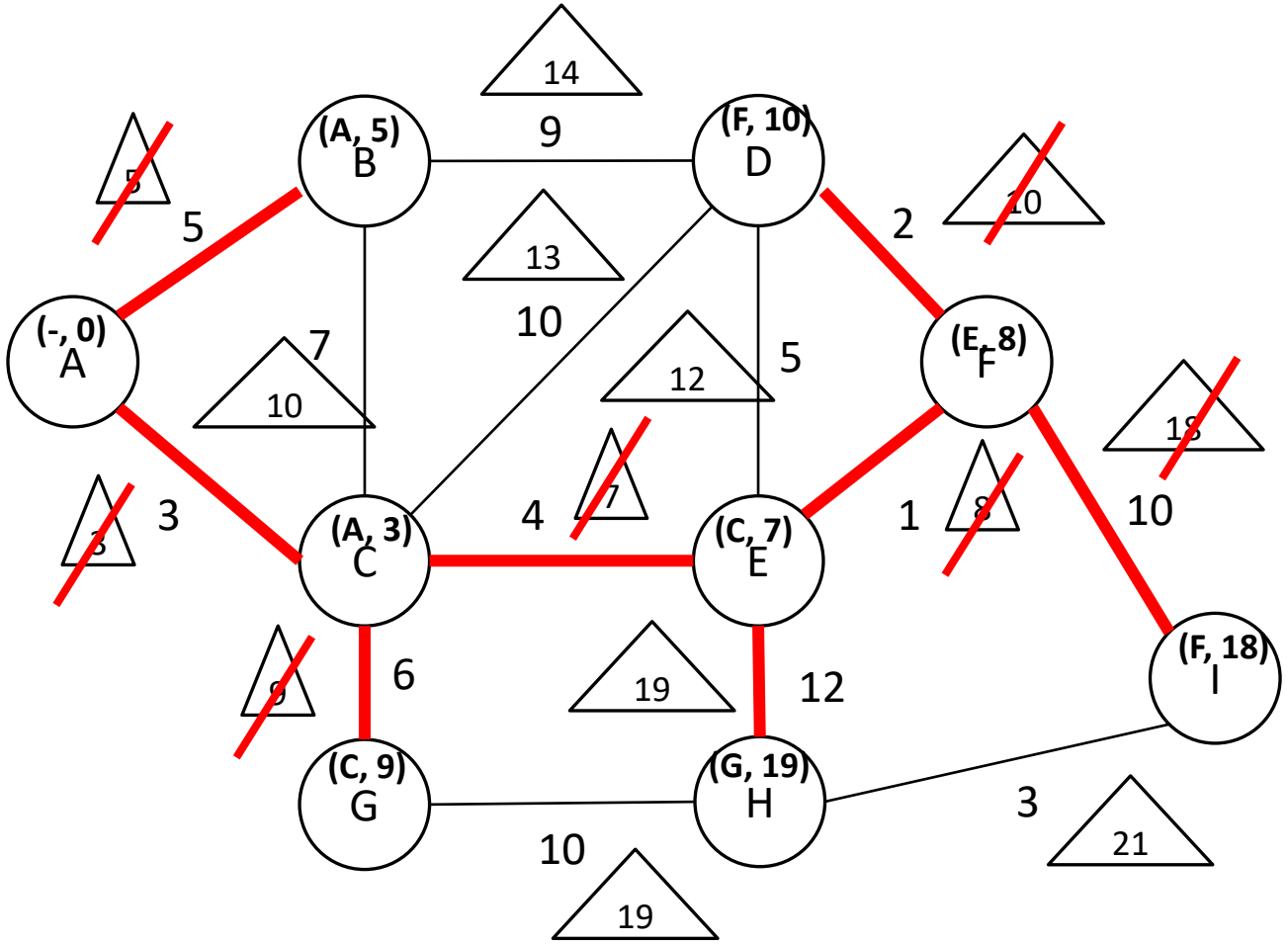
Dijkstra Shortest Path Algorithm: A to H



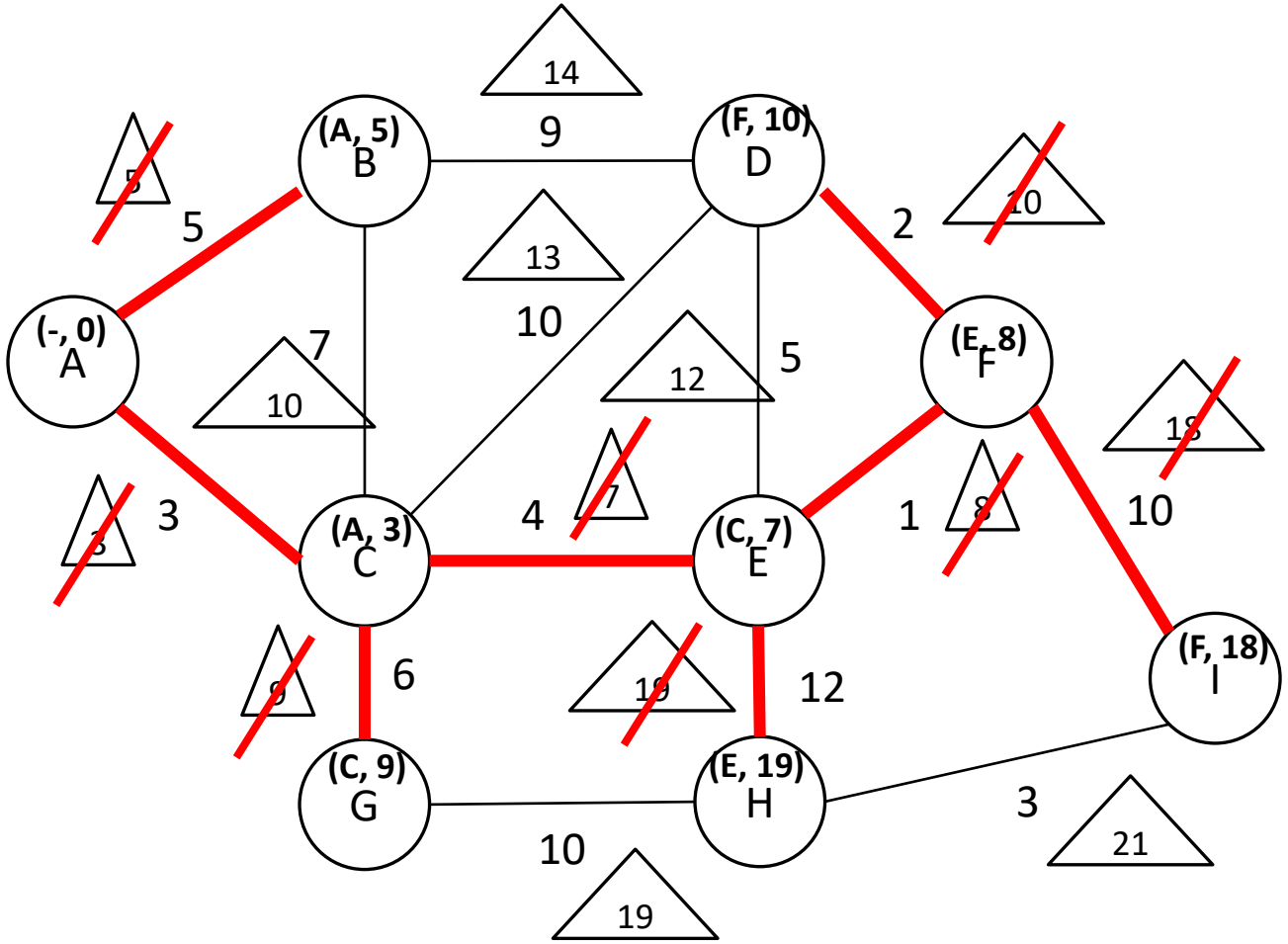
Dijkstra Shortest Path Algorithm: A to H



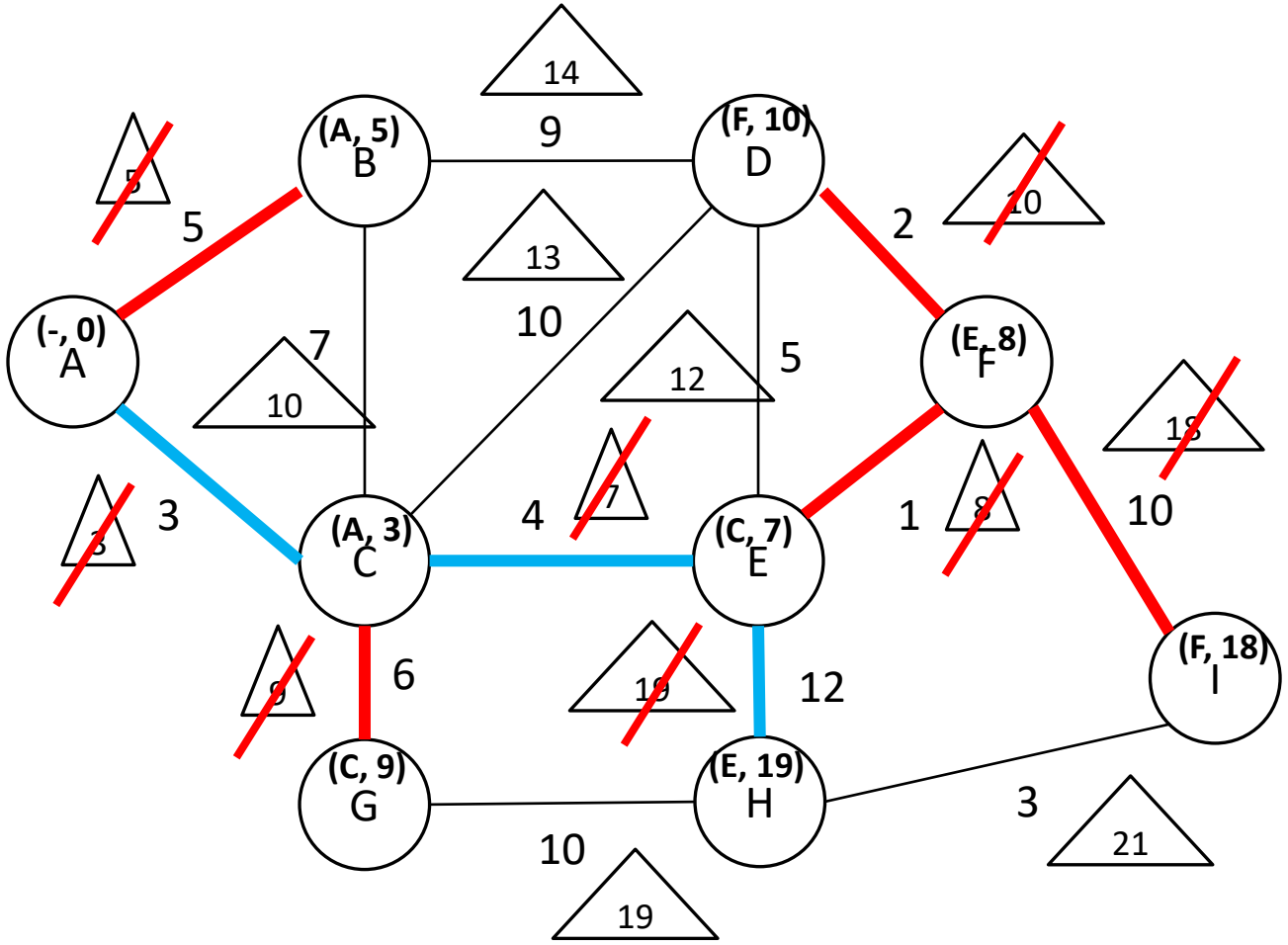
Dijkstra Shortest Path Algorithm: A to H



Dijkstra Shortest Path Algorithm: A to H



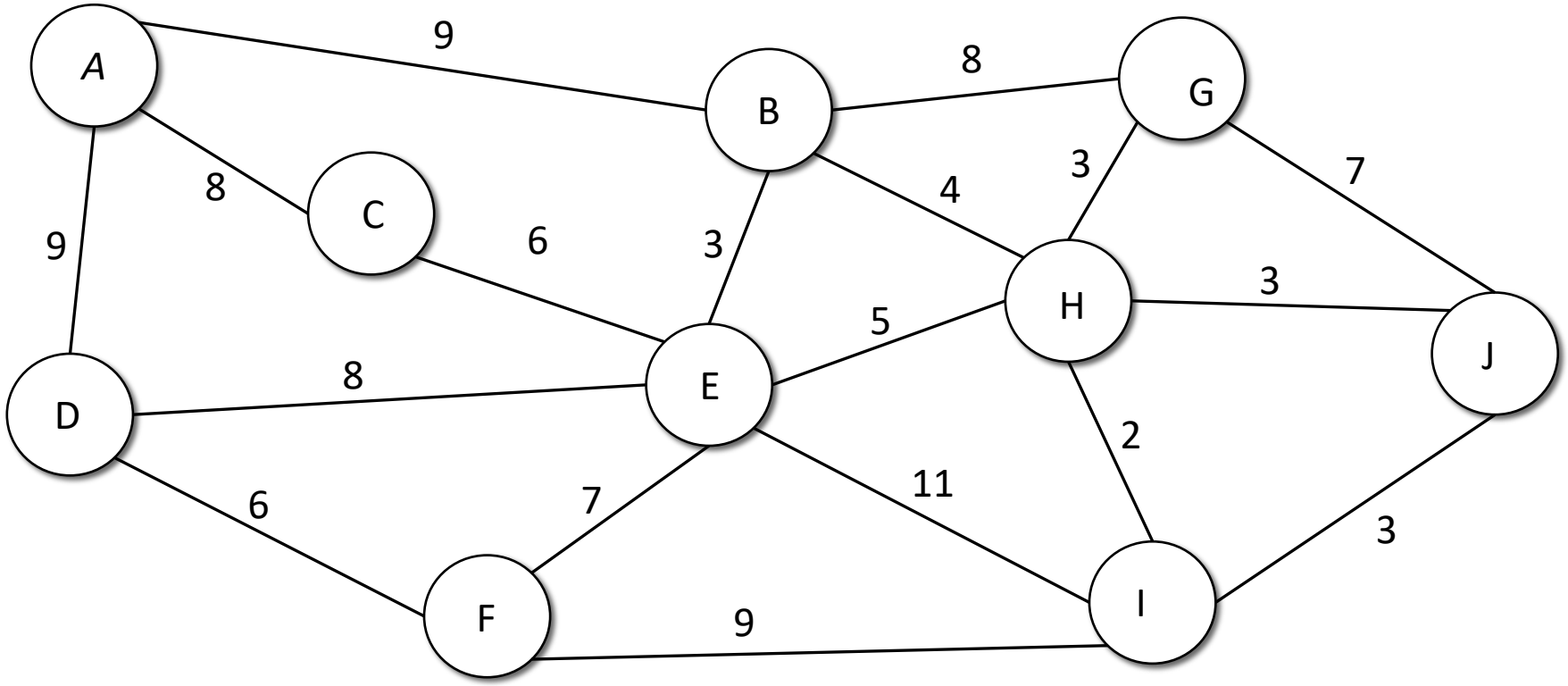
Dijkstra Shortest Path Algorithm: A to H



Path:

A -> C -> E -> H

You Try Dijkstra's Algorithm A - J



Dijkstra's Algorithm

1) Start at the origin vertex (S) and give it a permanent label

- **A) This permanent label will be(-,0) to represent coming from nowhere at zero cost**
- **B) All other edges and vertices are unlabeled**

2) For each permanently labeled vertex

- **A) Give each edge connected from it to an unlabeled vertex a temporary label denoting the total least cost to travel across that edge *from the origin***
- **B) Choose the edge with the smallest total cost (the smallest temporary label) and permanently label its connected vertex with the name of the vertex you traveled from and the cost to get there *from the origin***
- **C) Identify that edge as possibly being part of the final path solution (highlight the edge)**
- **D) Eliminate temporary labels on edges between permanently labeled vertices**

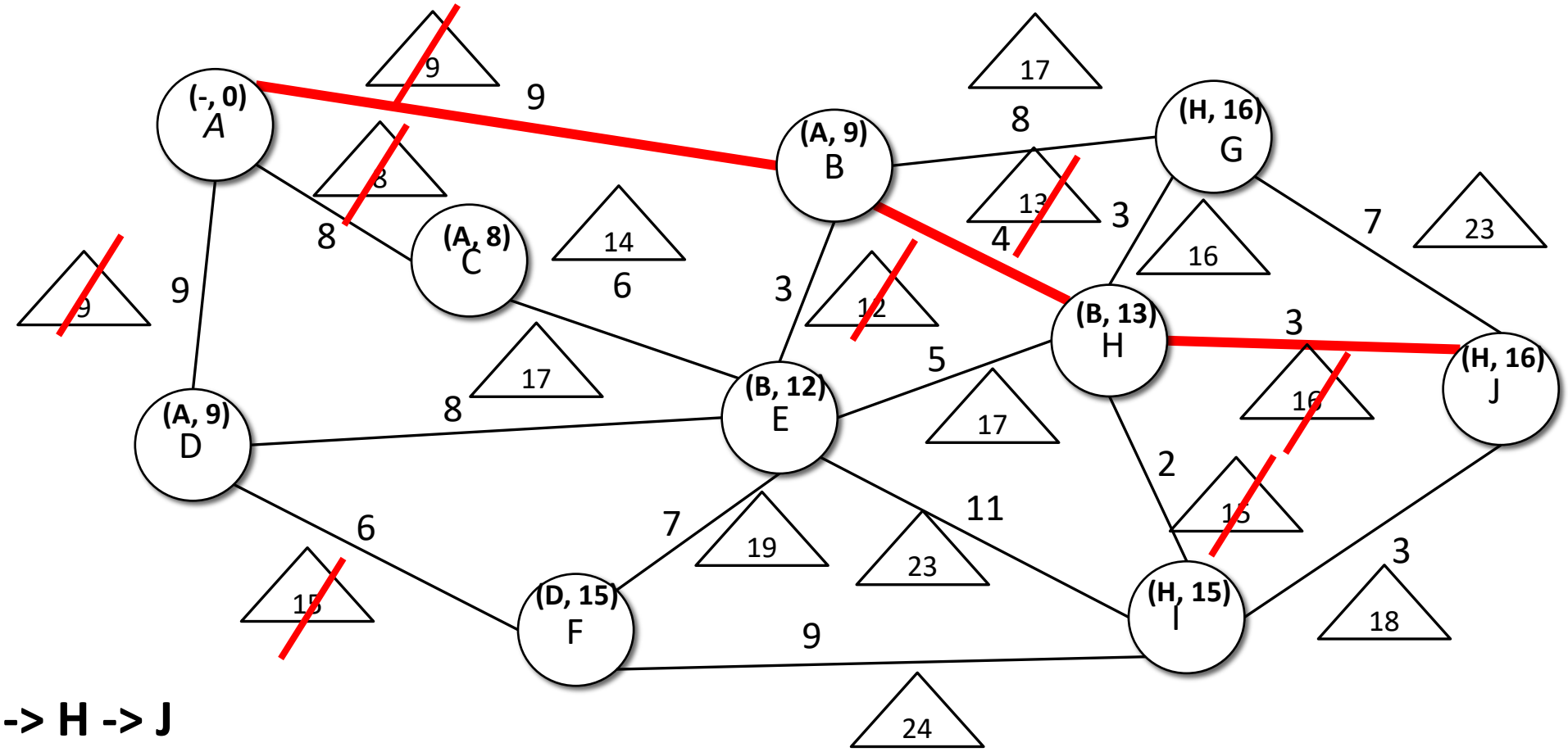
3) If vertex T has been permanently labeled stop since a shortest path from S to T has been found. If vertex T has not been labeled go to step 2.

4) When T is reached use the permanent labels on the vertices to work backwards to the origin defining the shortest path

BREAK TIES ARBITRARILY. IT DOESN'T MATTER WHICH VERTEX YOU LABEL.

- **You might not get the same path (alternate optima) but you will get the same cost.**

You Try Dijkstra's Algorithm: A to J



Path:

A -> B -> H -> J