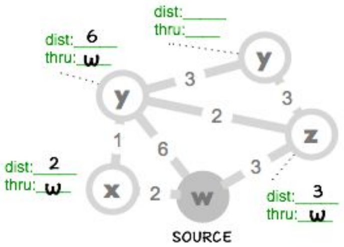
Name(s) Cecilia Beckerbauer Period 4 Date ____________

Activity Guide - Dijkstra's Shortest Path Algorithm

You be the computer!

Below is an adaptation of a famous algorithm to find the shortest path from one node to *all the other nodes* in the graph. It is known as **Dijkstra's algorithm**, after its inventor, Edsger Dijkstra ("Dike-strah"). Today you will act as the computer, interpreting the instructions for the algorithm to produce a **shortest path spanning tree** rooted at your router.



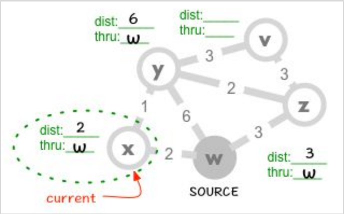
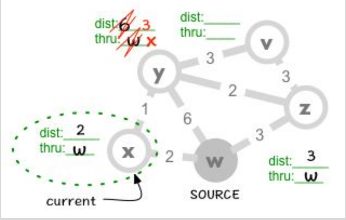
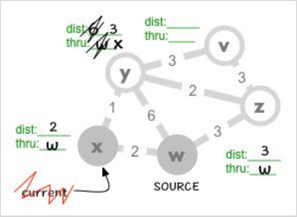
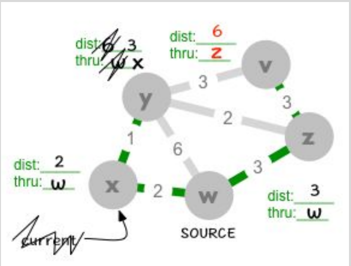
Here's what Dijkstra's algorithm does, in informal English:
The algorithm has you systematically find the shortest path from a **source node** to every other node in the graph by considering one node at a time. Along the way, you keep track of the total distance from the source going through other nodes and update those distances when you find shorter paths. The algorithm below is also written in English but is a bit more structured than usual; it's closer to a real computer program.

Directions:
With a partner, try to follow the algorithm below and track your progress on the graph provided to you. Follow the steps in order, starting at step 0.

Algorithm: Dijkstra's Single Source Shortest Path

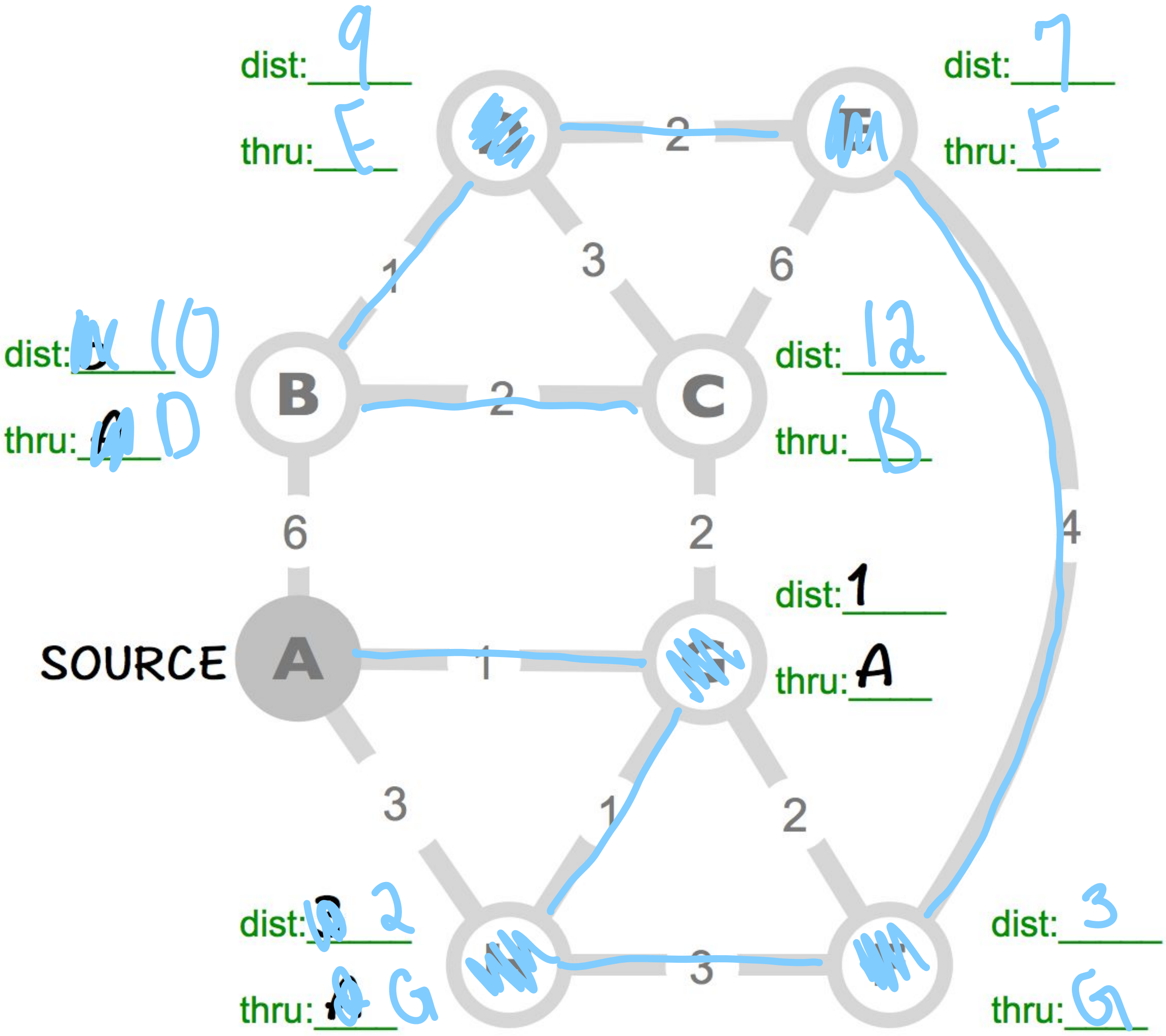
Setup: On the graph, your source node is labeled **Source** and is shaded in. For each of the neighbors of your **Source** node, we've set the initial values for **dist** and **thru.**

NOTE: dist will be used to keep track of the total best distance from Source

1	Identify the unshaded node with the smallest **dist** value (ignore blanks). If there is a tie for node with smallest **dist** value just pick either one. Label this node **Current.** *In the example on the right,* ***current*** *becomes x, because it is the node with the smallest* ***dist*** *of the three options → x = 2, z = 3, y = 6 and v is blank*	 dist: 6 thru: w dist: thru: v y z x w dist: 2 thru: w dist: 3 thru: w current SOURCE
2	For each unshaded **Neighbor** of **Current** complete step 3. (A **Neighbor** is a node with a direct link to **Current.**)	
3	Compute distance: Add **Current's dist +** weight of edge to **Neighbor.** **Check if** this distance is less than **Neighbor's** current **dist** OR **Neighbor's dist** is blank. If it is less (or blank) a. update **Neighbor's dist** to be this new **dist** b. update **Neighbor's thru** to be the name of the **Current** node. Repeat process for all the neighbors of **Current**.	 dist: ~~6~~ 3 thru: ~~w~~ x dist: thru: v y z x w dist: 2 thru: w dist: 3 thru: w current SOURCE
4	Once done visiting each neighbor of **Current**.... a. Shade in **Current** node b. Cross out the **Current** label. (You're now done with this node)	 dist: ~~6~~ 3 thru: ~~w~~ x dist: thru: v y z x w dist: 2 thru: w dist: 3 thru: w ~~current~~ SOURCE
5	**If** all nodes in the graph are shaded → go to step 6 **otherwise** → go back to step 1	 dist: ~~6~~ 3 thru: ~~w~~ x dist: 6 thru: z v y z x w dist: 2 thru: w dist: 3 thru: w ~~current~~ SOURCE
6	For each node in the graph: highlight the edge between that node and its **thru** node.	
7	The shortest path tree from the source is now highlighted. **Yay**!	

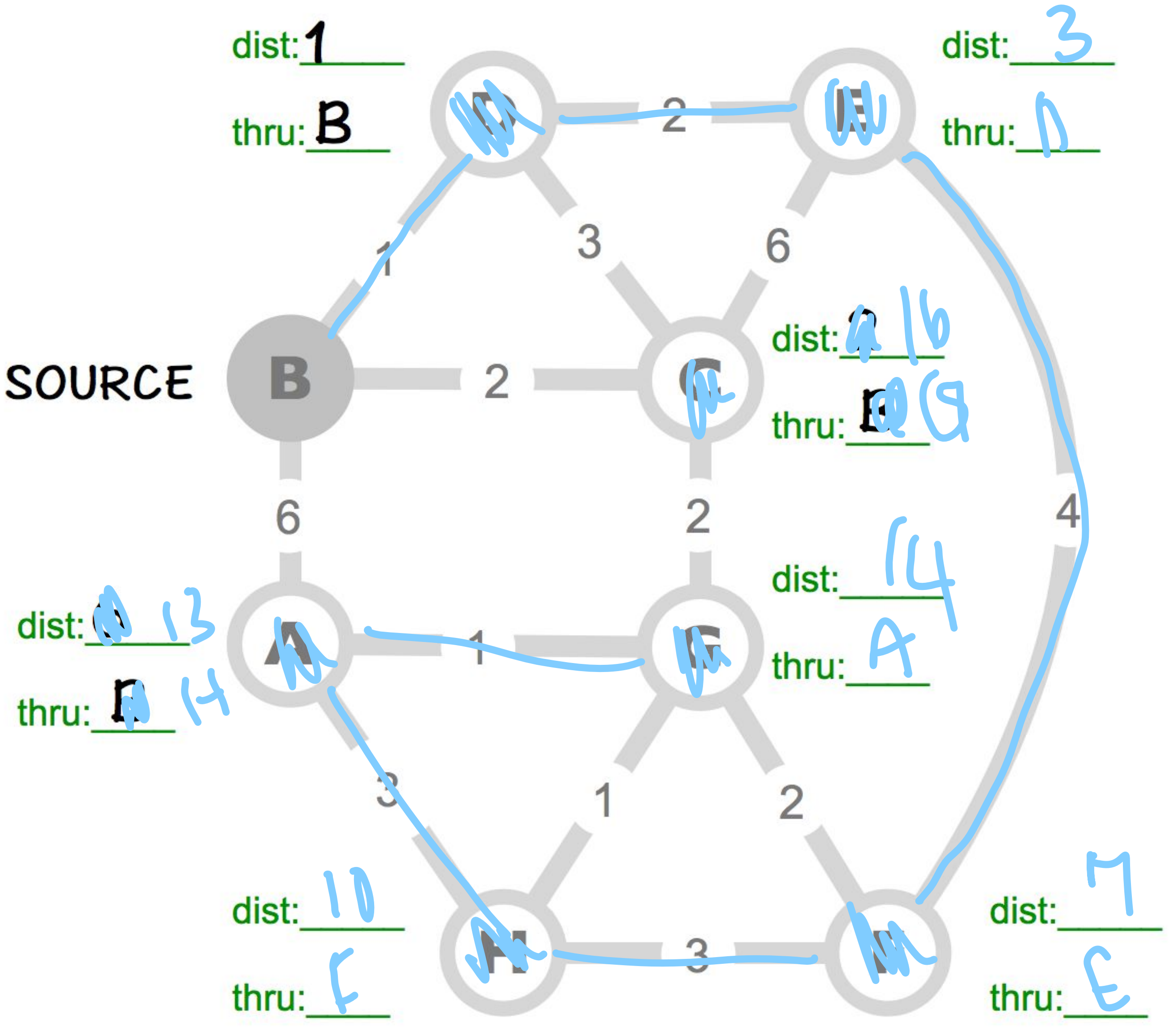
Student Graph A

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **from** values for the immediate neighbors of the source have been filled in.



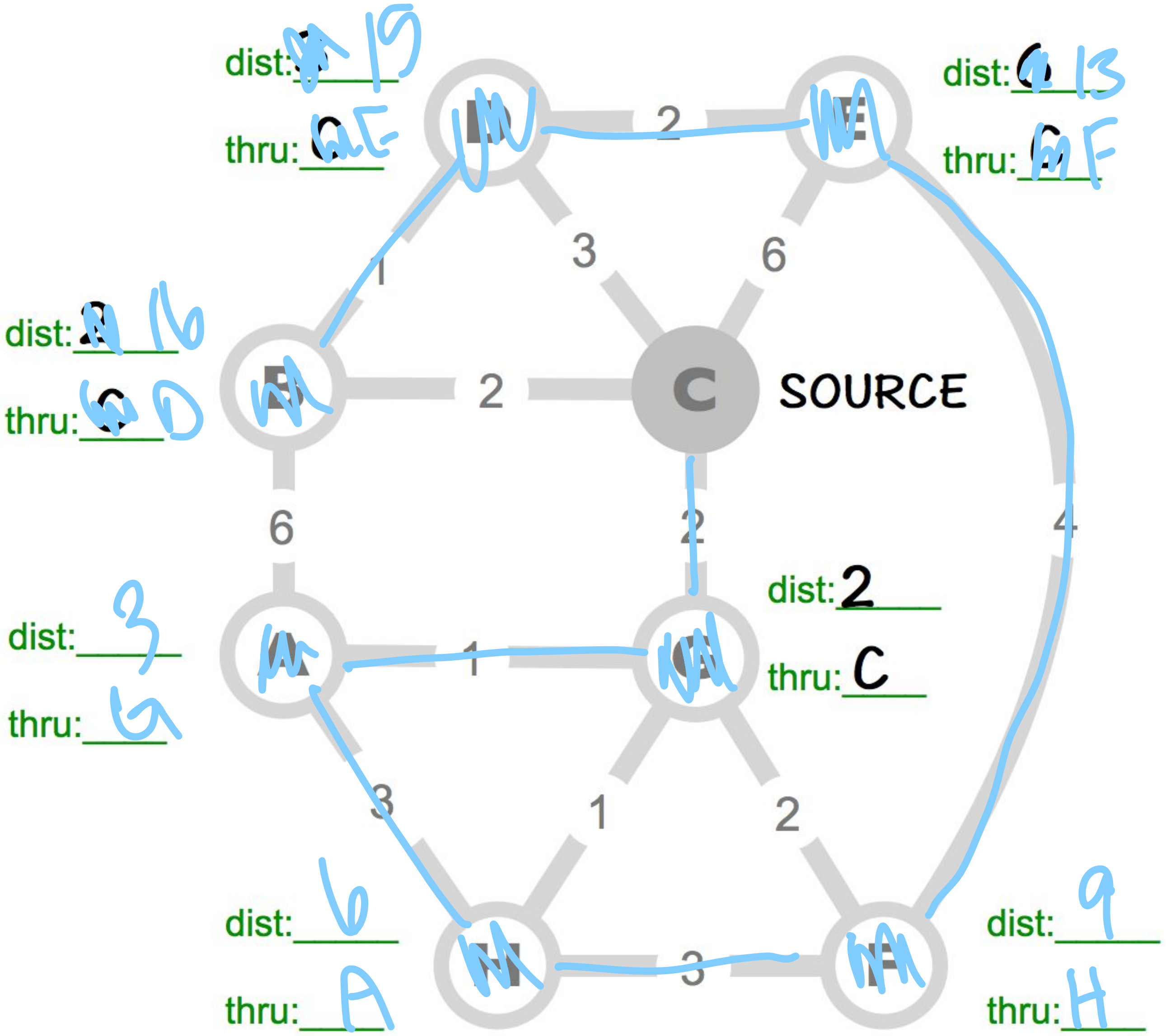
Student Graph B

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.



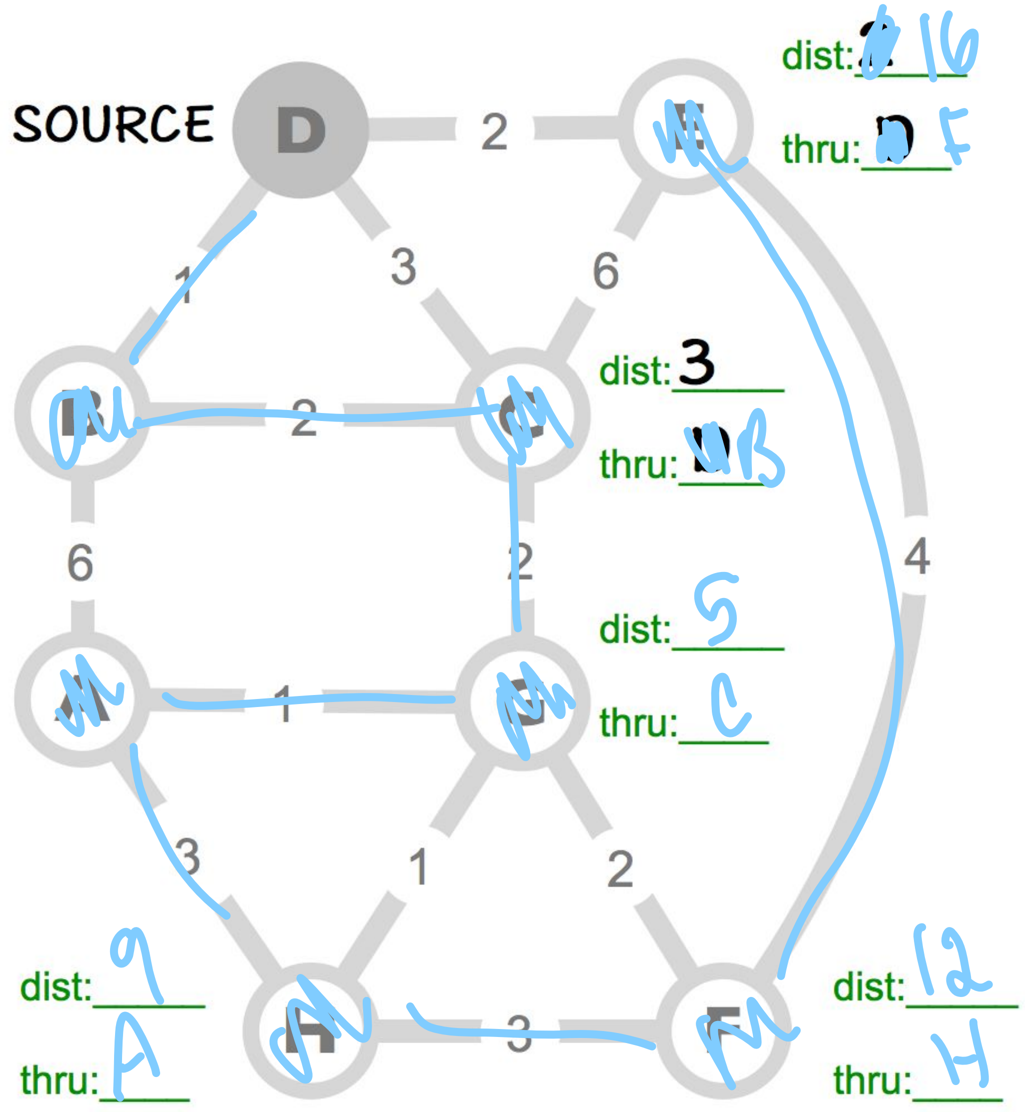
Student Graph C

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.



Student Graph D

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.



E — dist: ~~2~~ 16 thru: ~~D~~ F

B — dist: 1 thru: D

C — dist: 3 thru: ~~D~~ B

A — dist: 6 thru: G

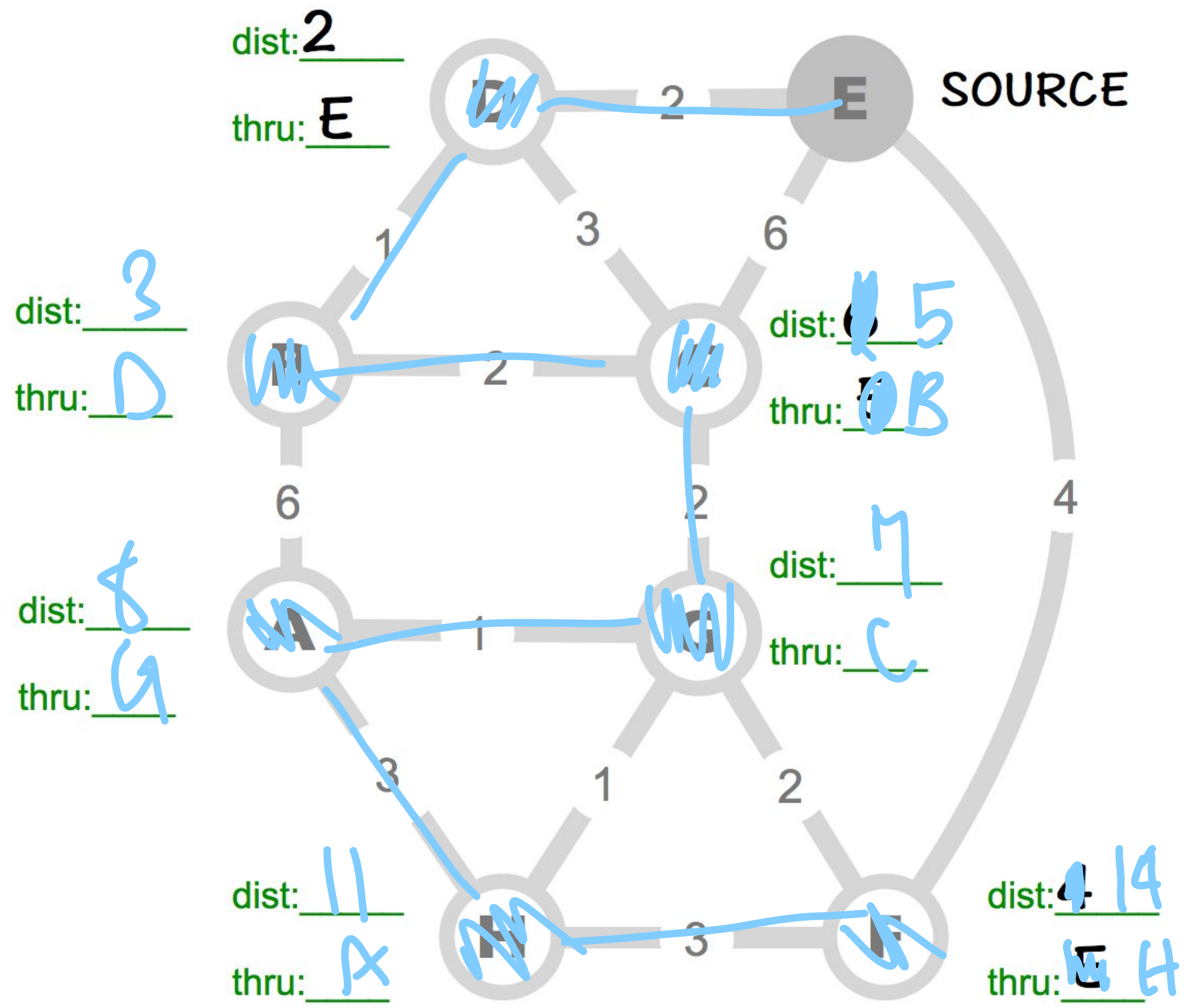
G — dist: 5 thru: C

H — dist: 9 thru: A

F — dist: 12 thru: H

Student Graph E

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.



dist: 2
thru: E

dist: 3
thru: D

dist: ~~6~~ 5
thru: ~~E~~ B

dist: 7
thru: C

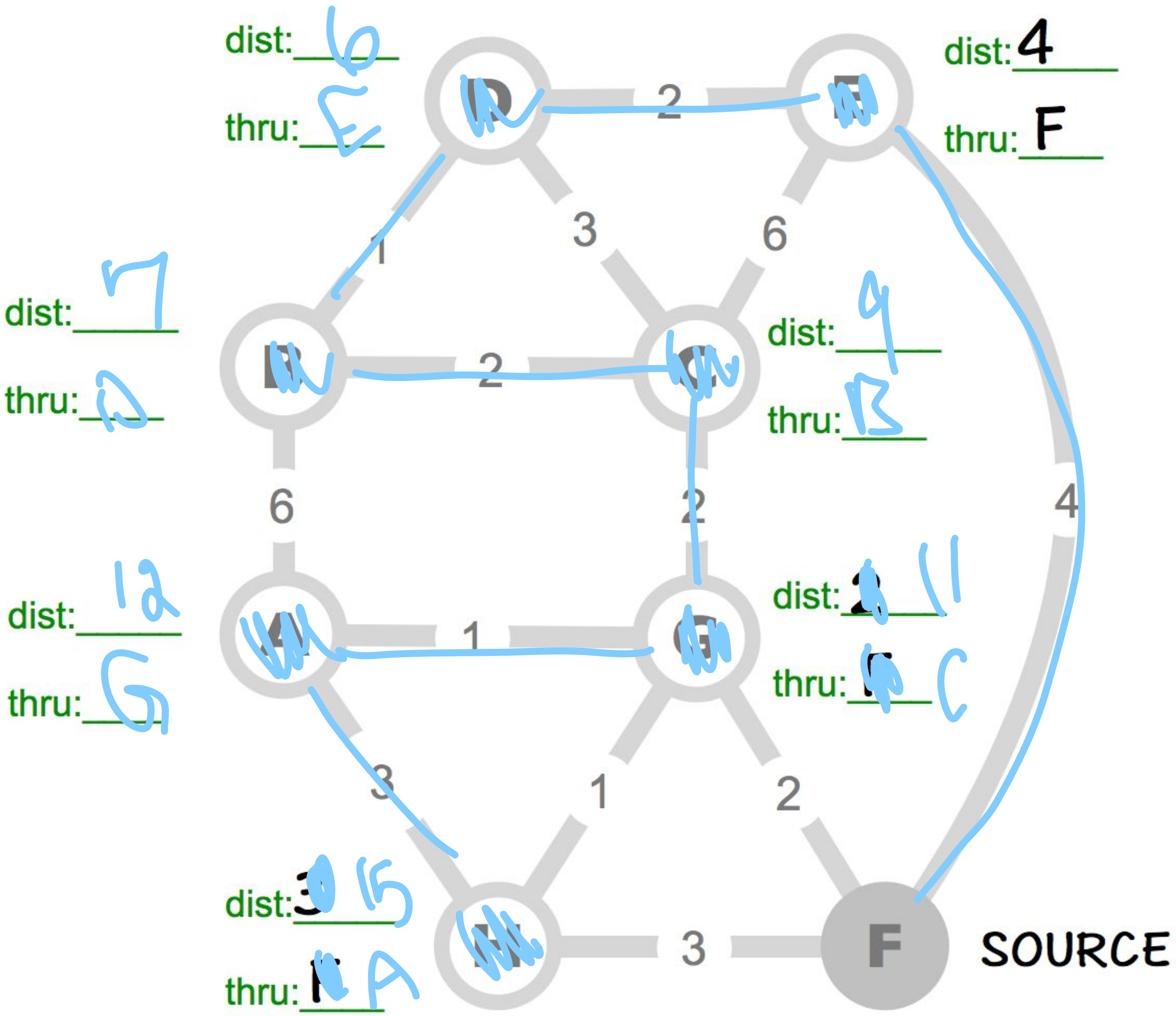
dist: 8
thru: G

dist: 11
thru: A

dist: ~~4~~ 14
thru: ~~E~~ H

Student Graph F

Trace the shortest path algorithm with your partner using this graph.
The source node has been shaded in for you.
The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.

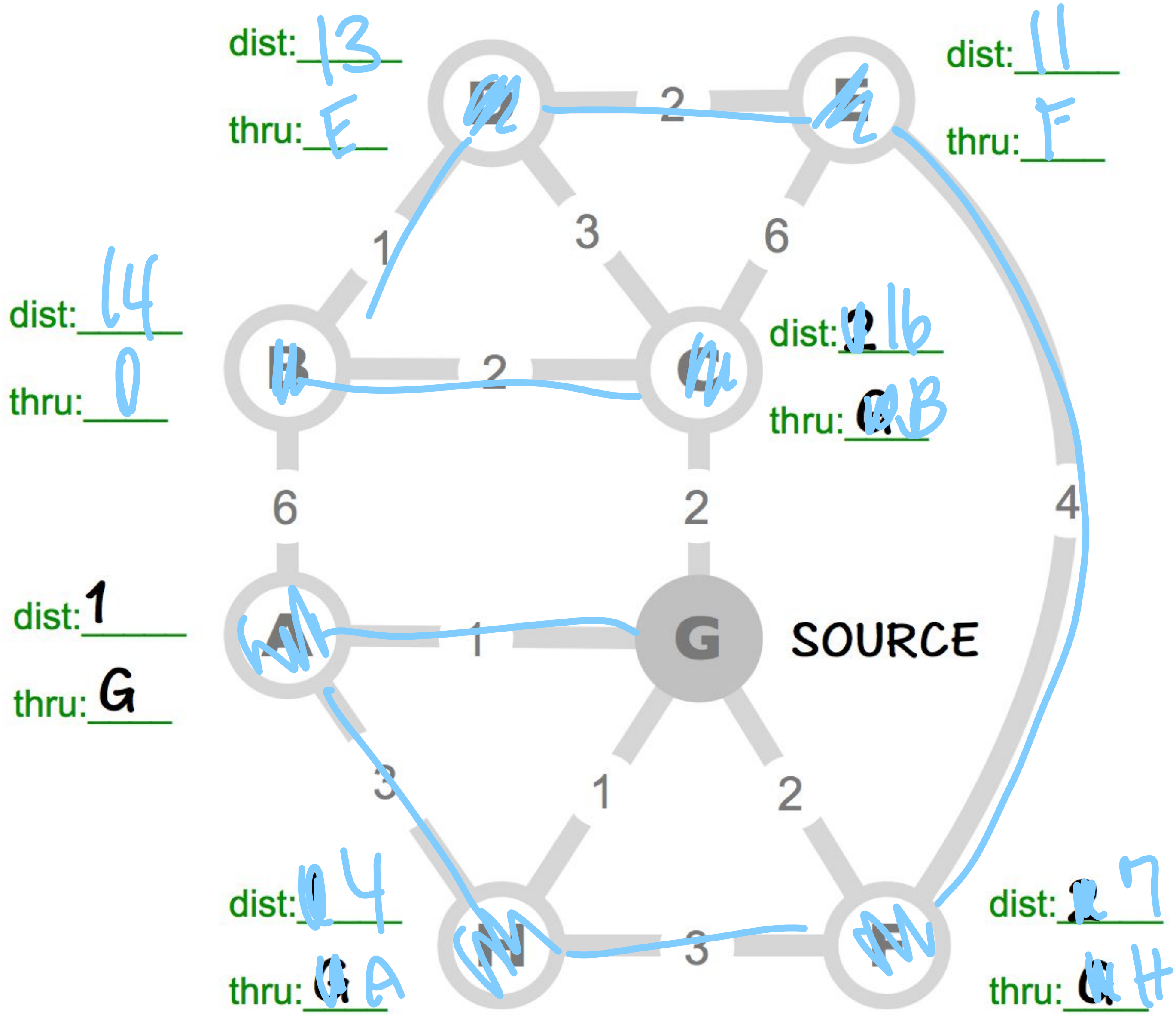


Student Graph G

Trace the shortest path algorithm with your partner using this graph.

The source node has been shaded in for you.

The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.

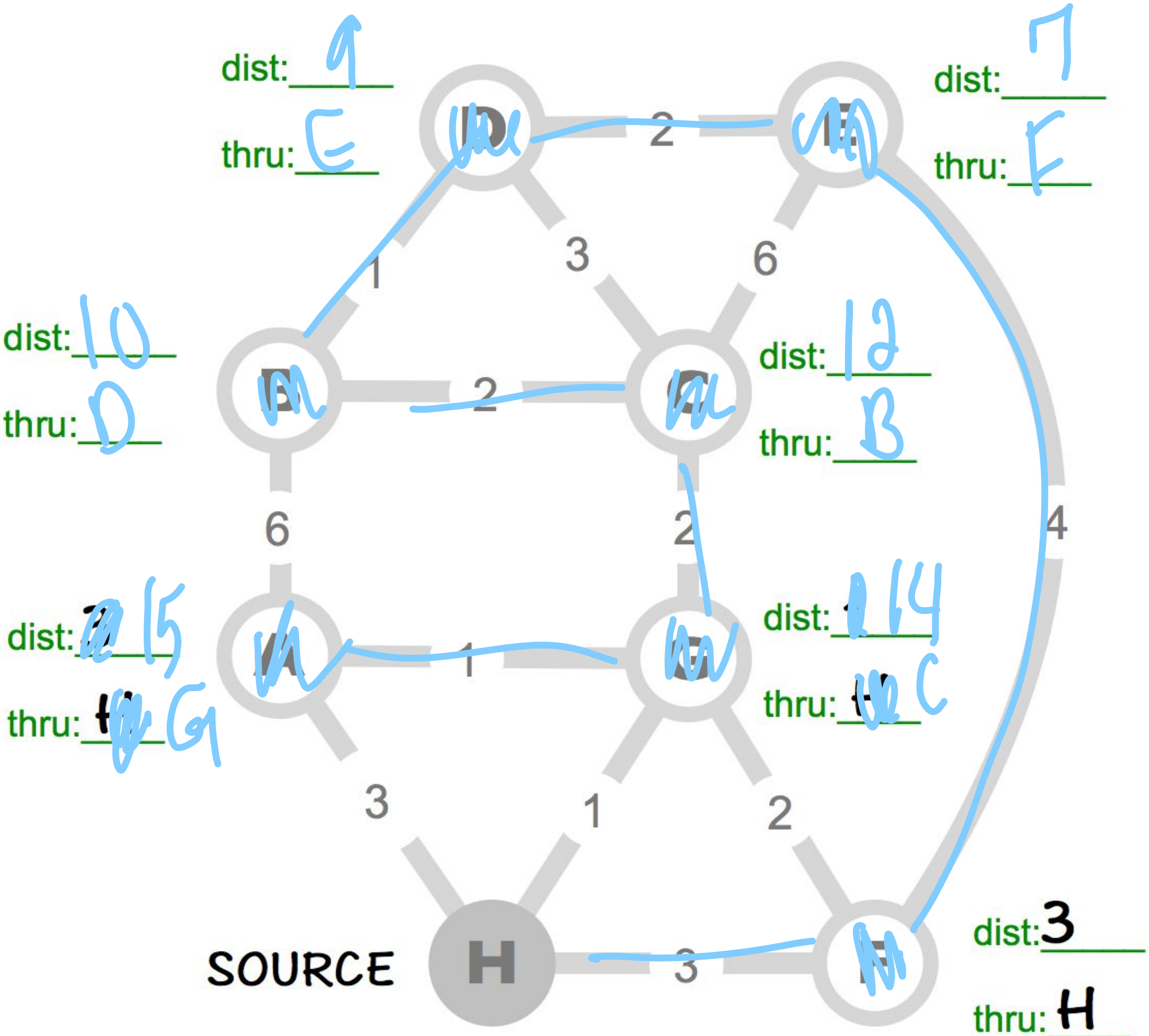


Student Graph H

Trace the shortest path algorithm with your partner using this graph.

The source node has been shaded in for you.

The **dist**ance and **thru** values for the immediate neighbors of the source have been filled in.


dist: 9
thru: E
D
2
E
dist: 7
thru: F
1
3
6
dist: 10
thru: D
B
2
C
dist: 12
thru: B
6
2
4
dist: ~~2~~ 15
thru: ~~H~~ G
A
1
G
dist: ~~1~~ 14
thru: ~~H~~ C
3
1
2
SOURCE
H
3
F
dist: 3
thru: H