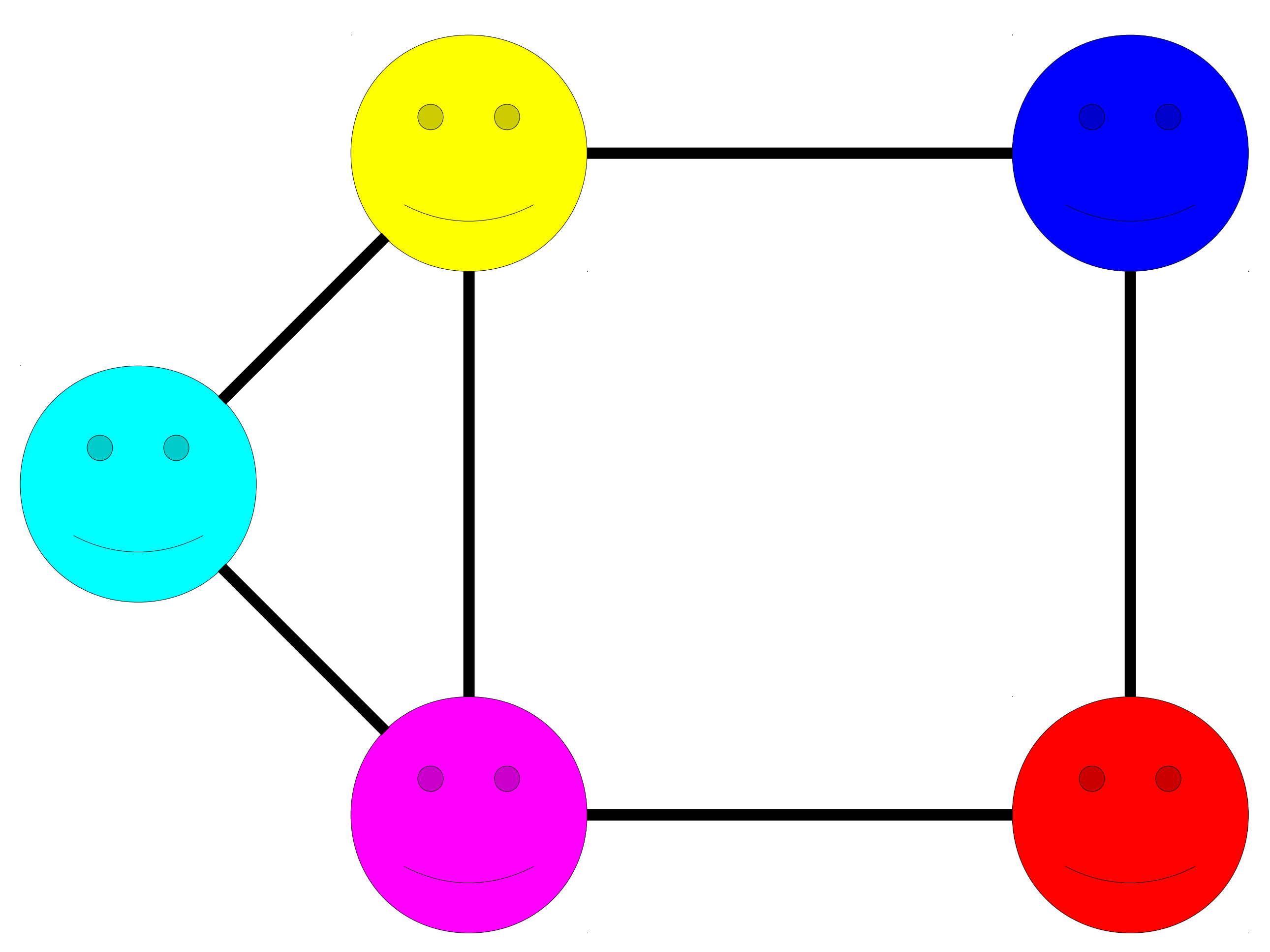
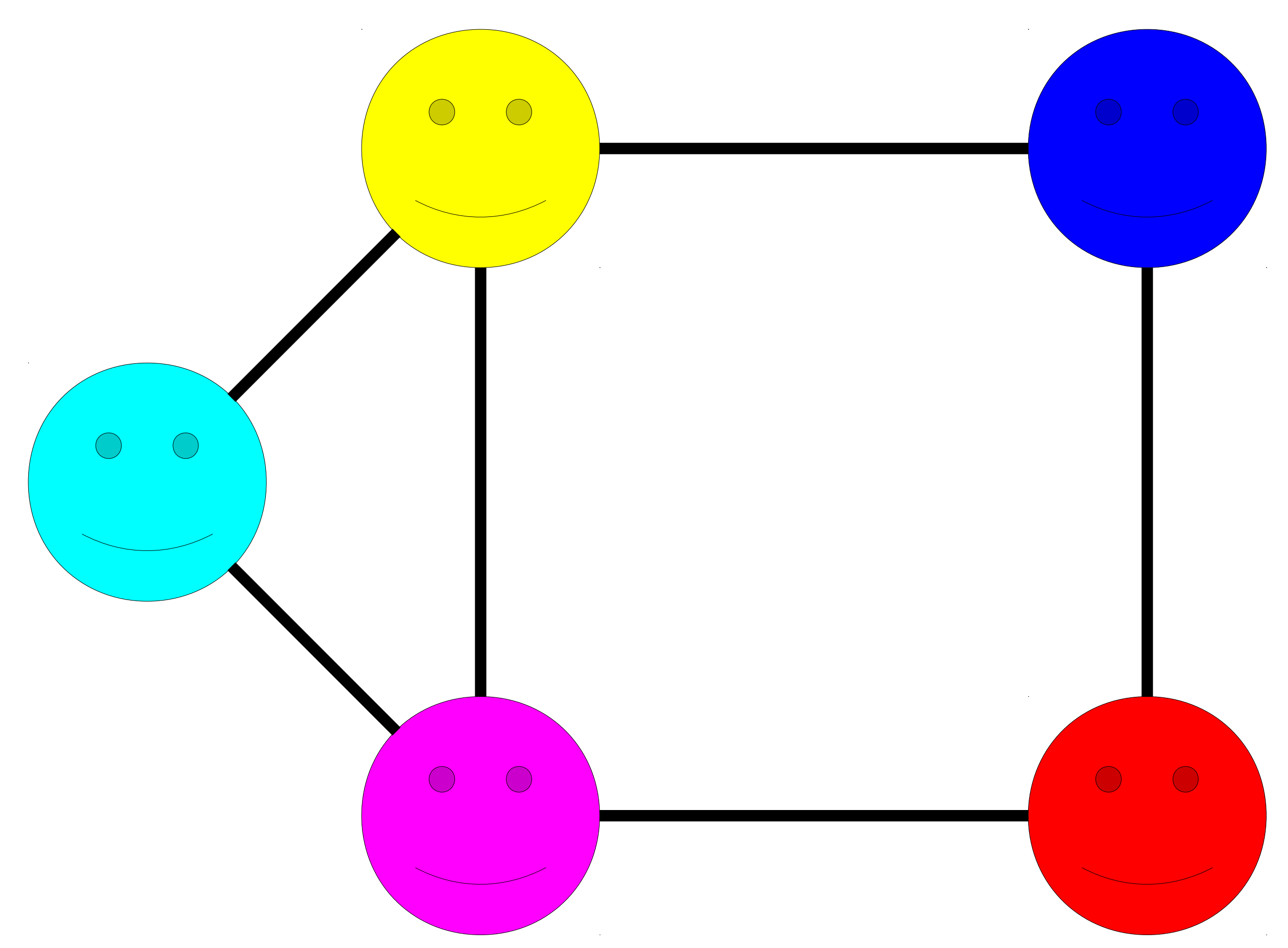
Shortest Paths

A **graph** is a mathematical structure for representing relationships.

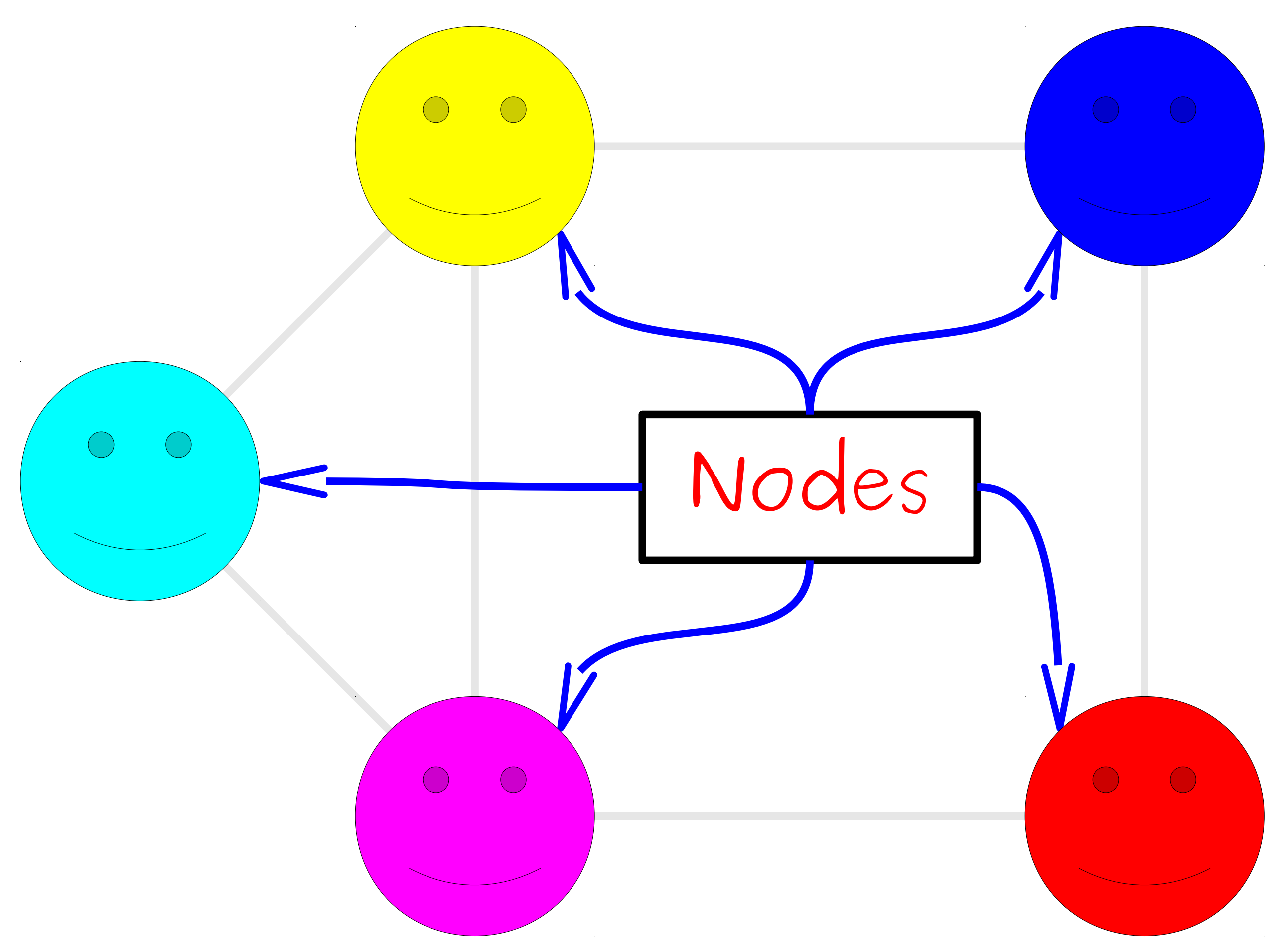


A **graph** is a mathematical structure for representing relationships.



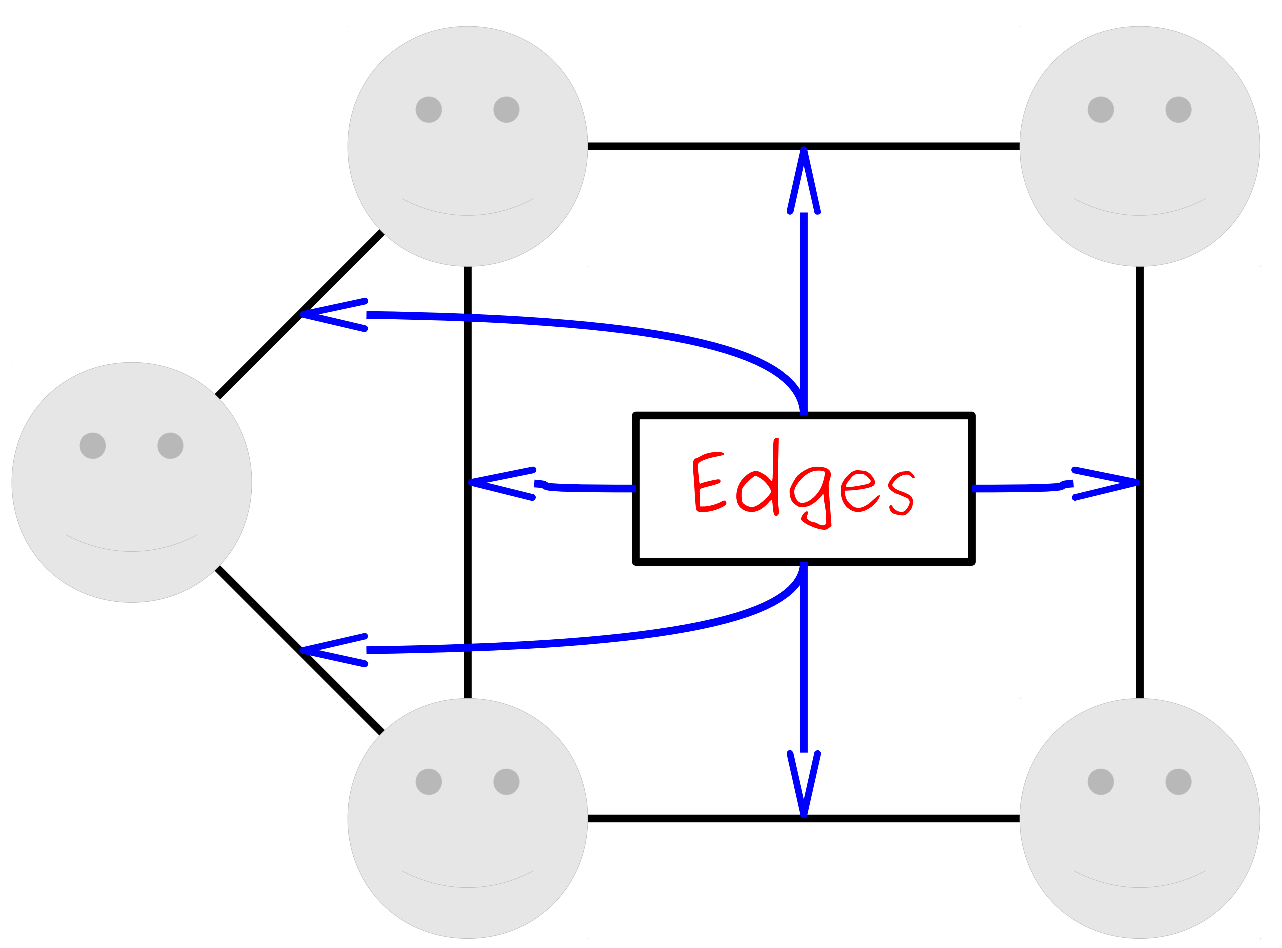
A graph consists of a set of **nodes** connected by **edges**.

A **graph** is a mathematical structure for representing relationships.



A graph consists of a set of **nodes** connected by **edges**.

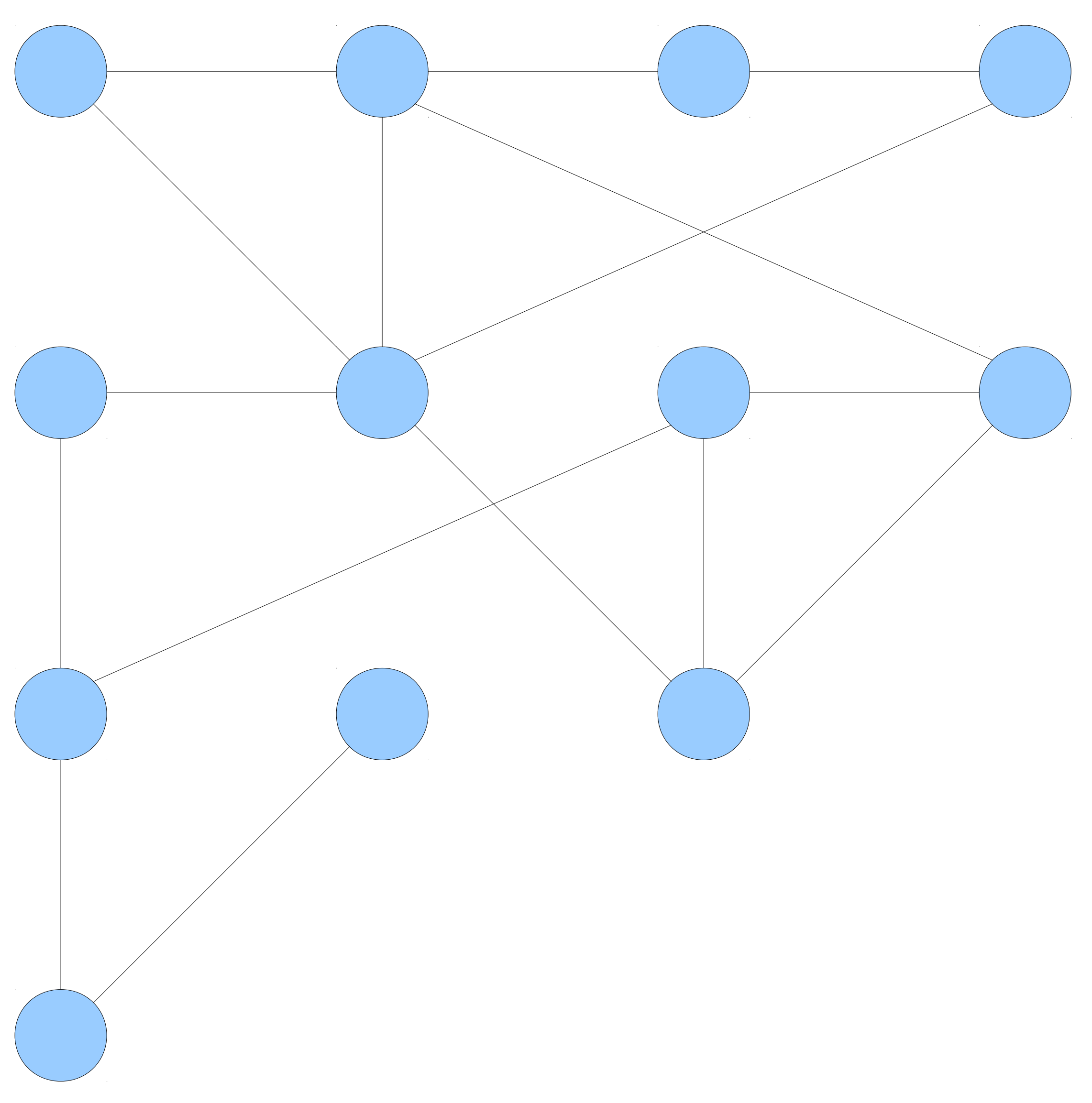
A **graph** is a mathematical structure for representing relationships.



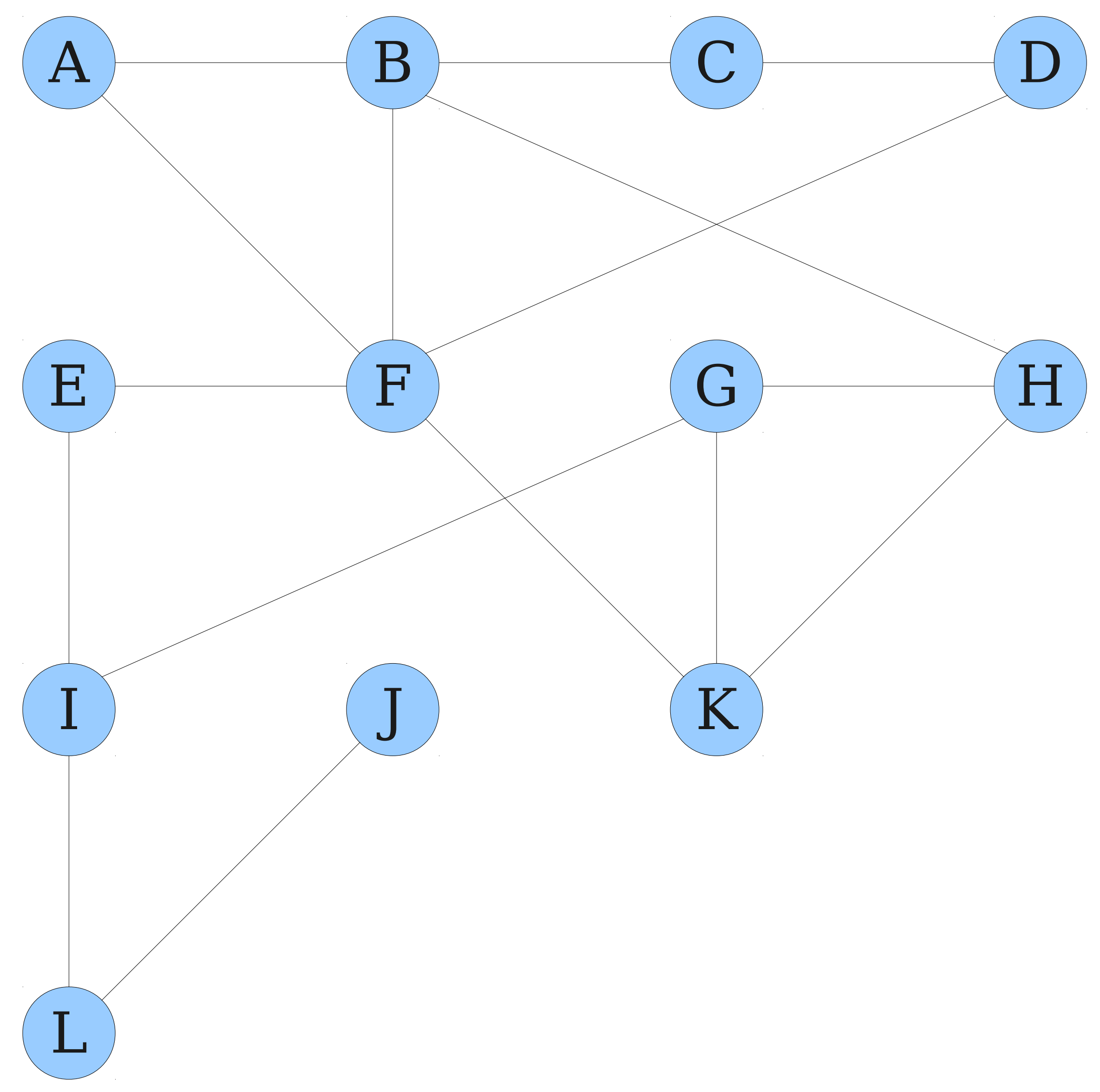
A graph consists of a set of **nodes** connected by **edges**.

Shortest Paths

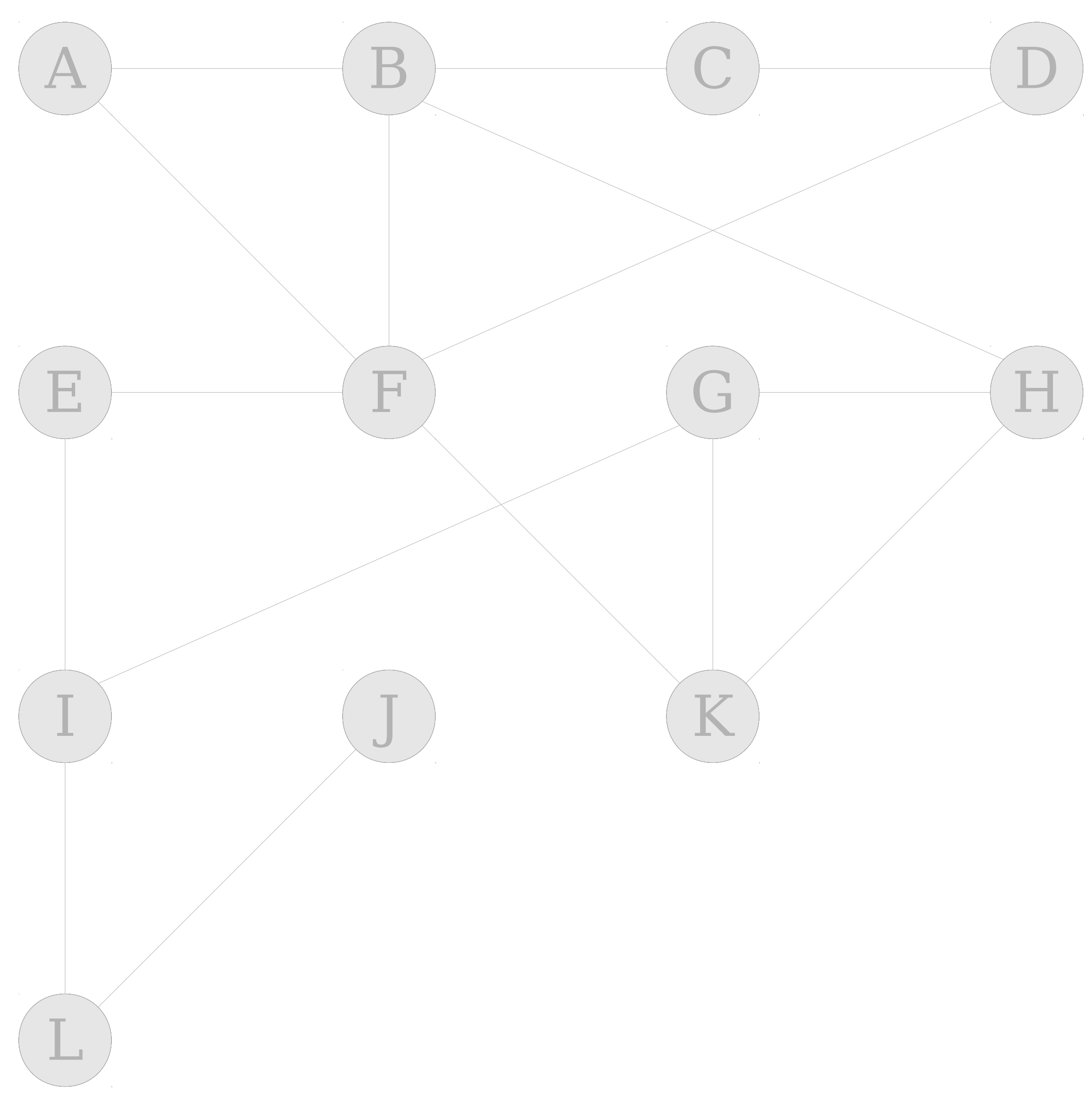
Breadth-First Search



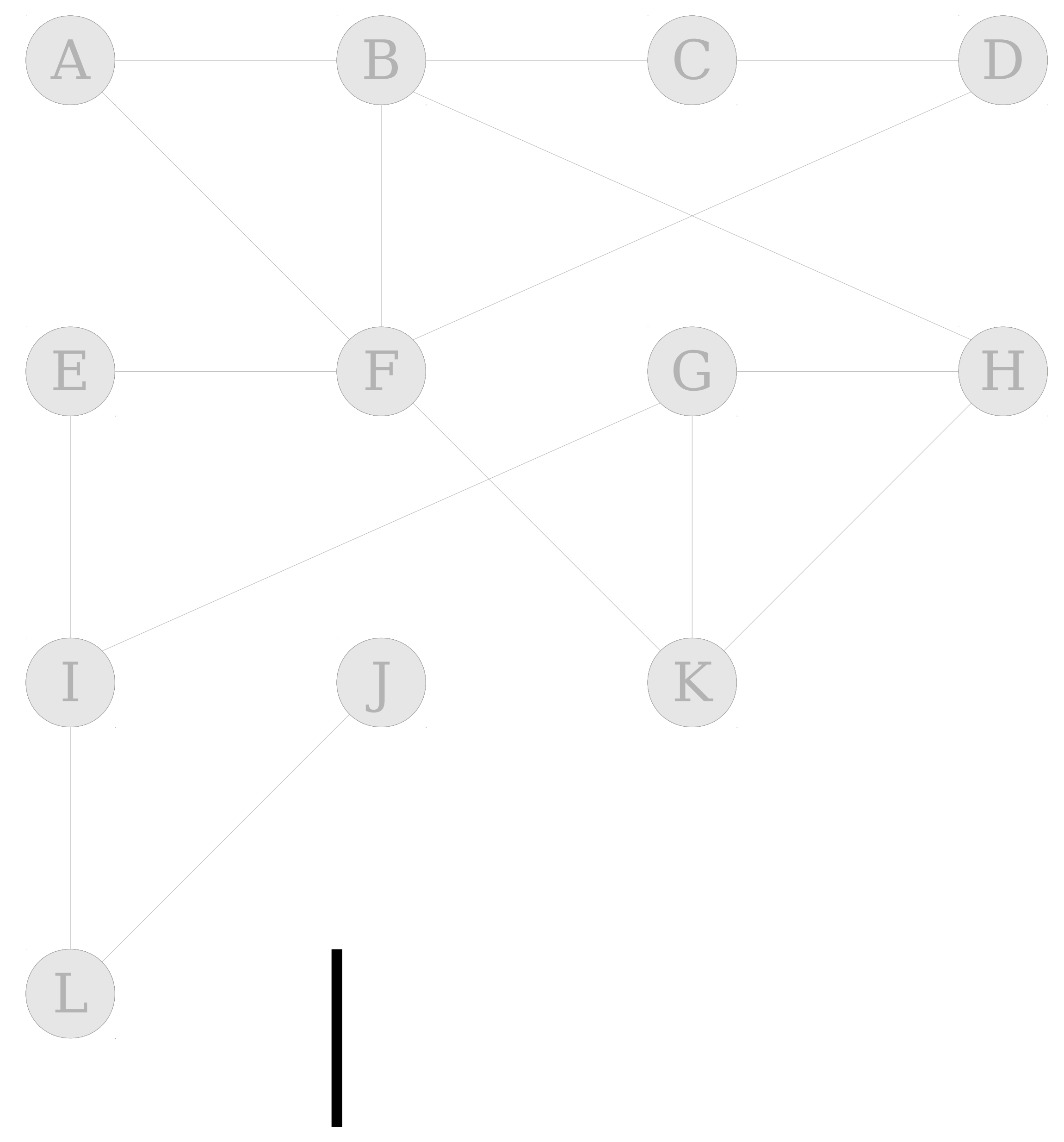
Breadth-First Search



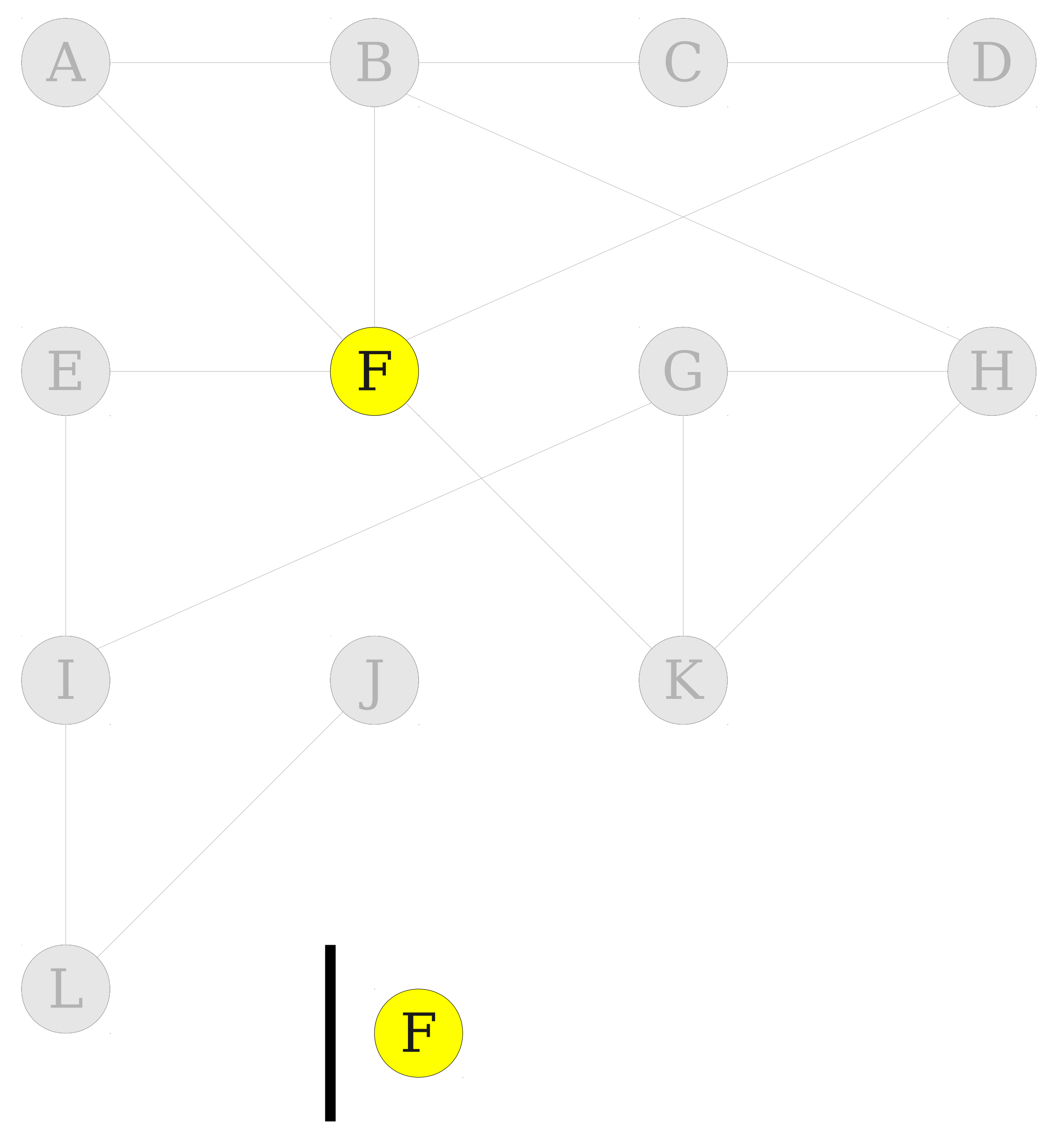
Breadth-First Search



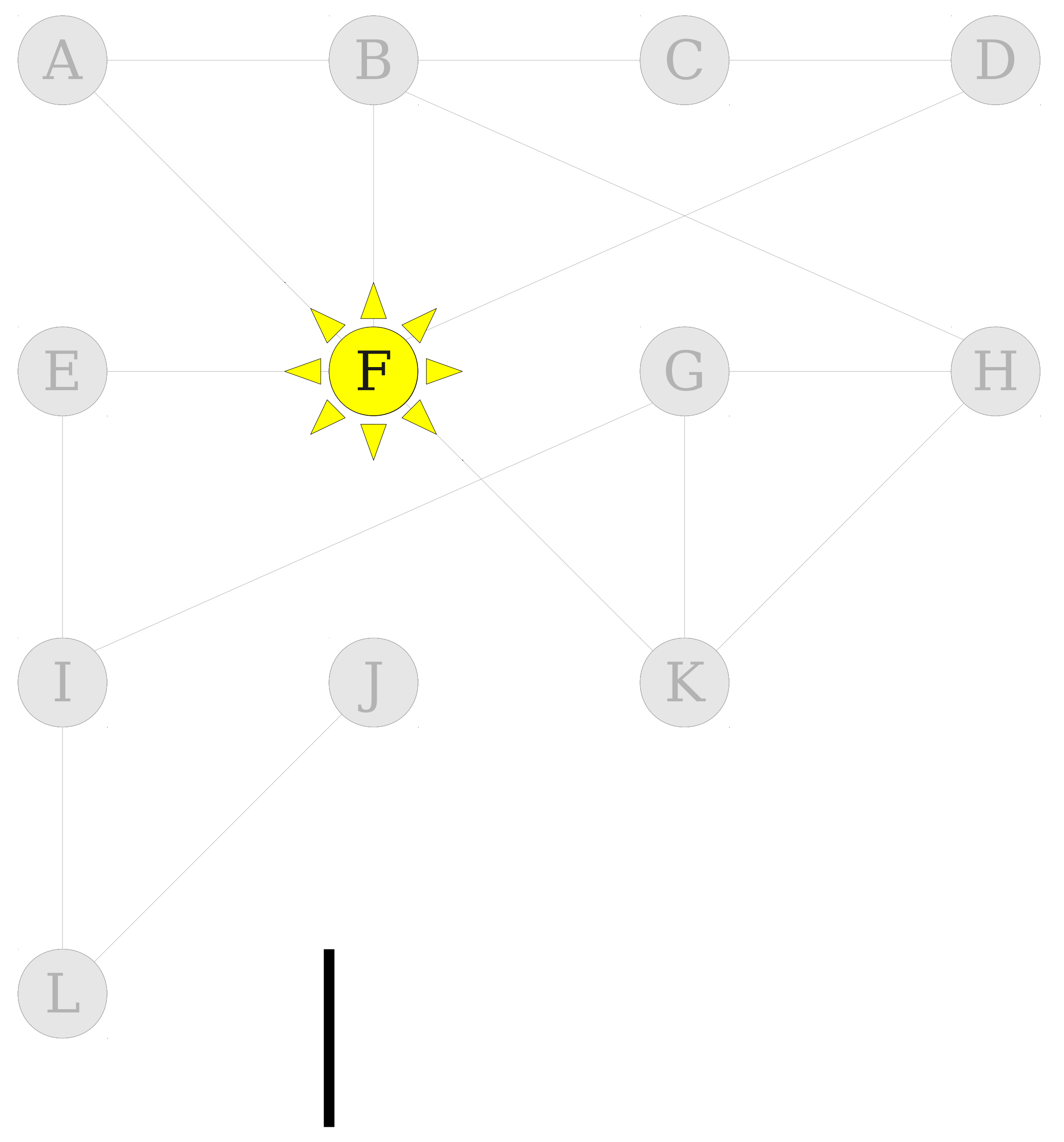
Breadth-First Search



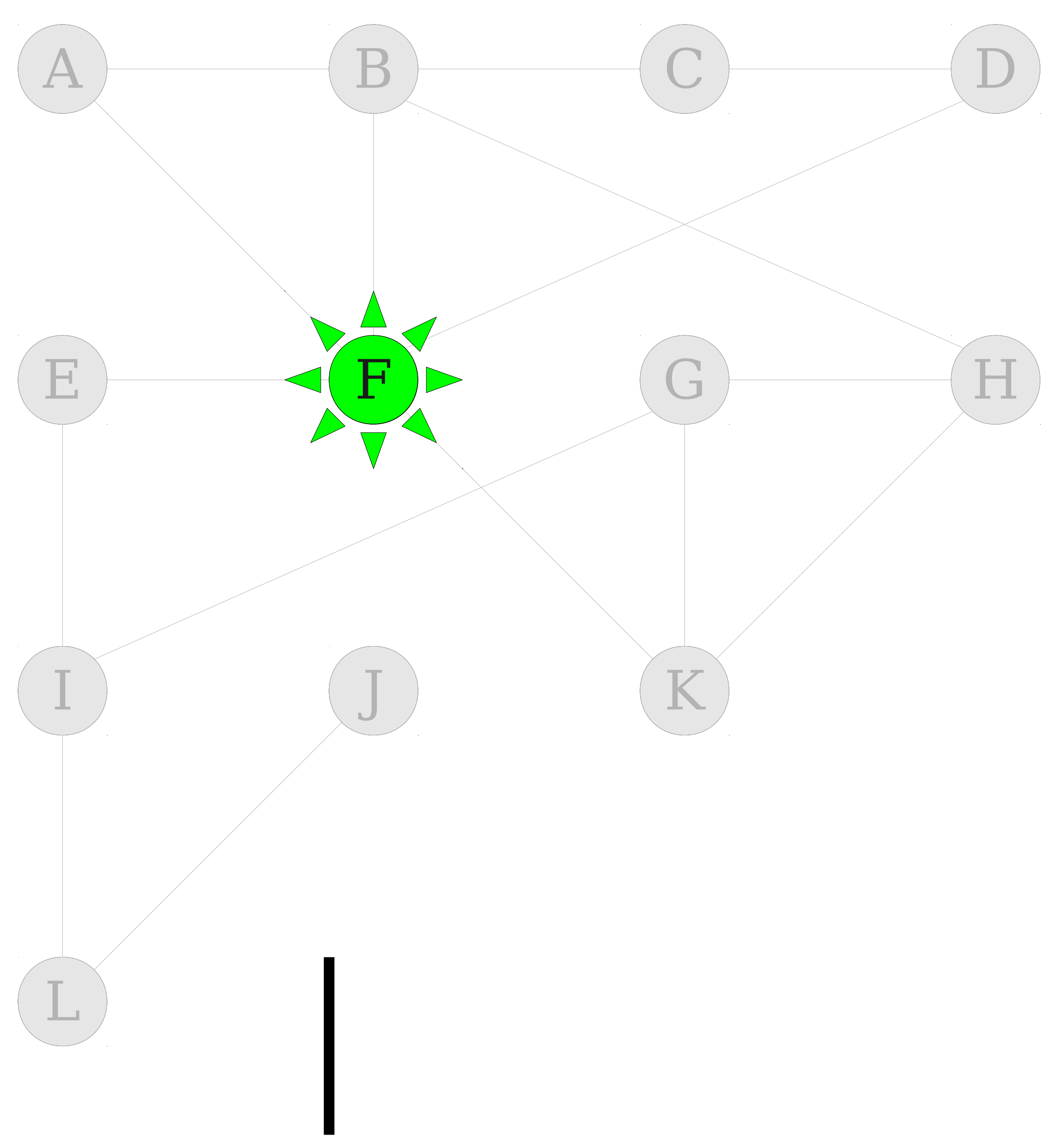
Breadth-First Search



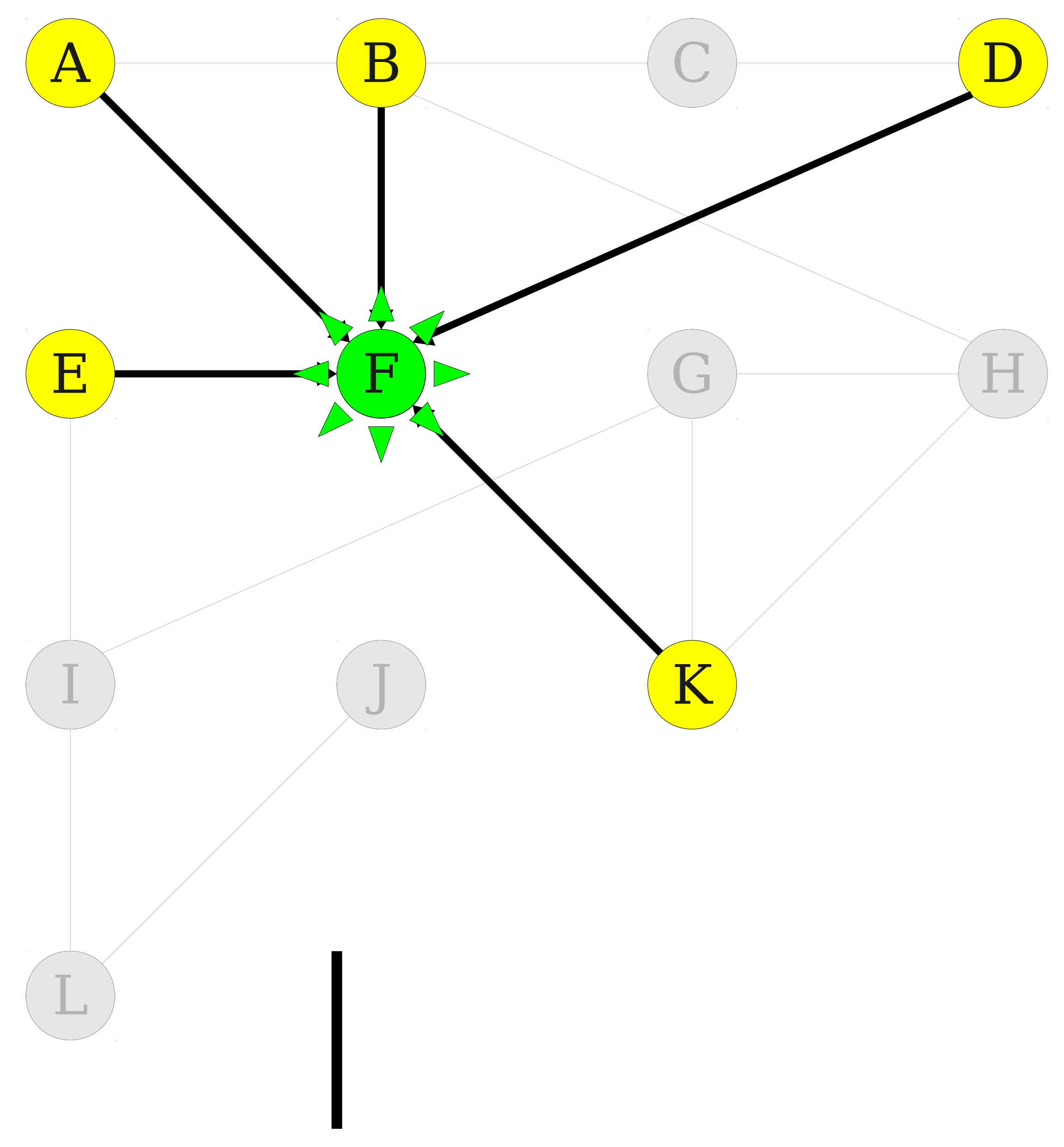
Breadth-First Search



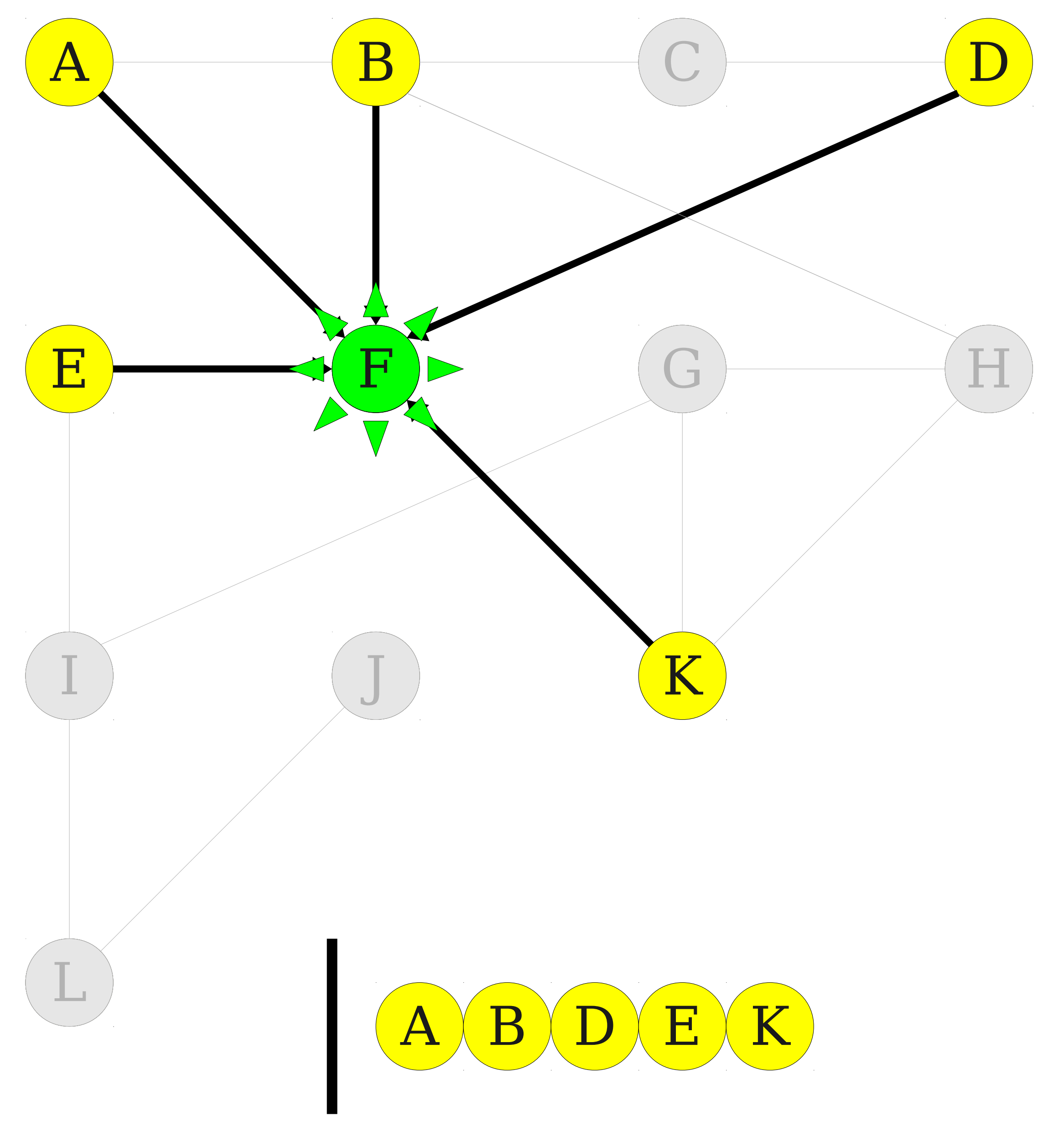
Breadth-First Search



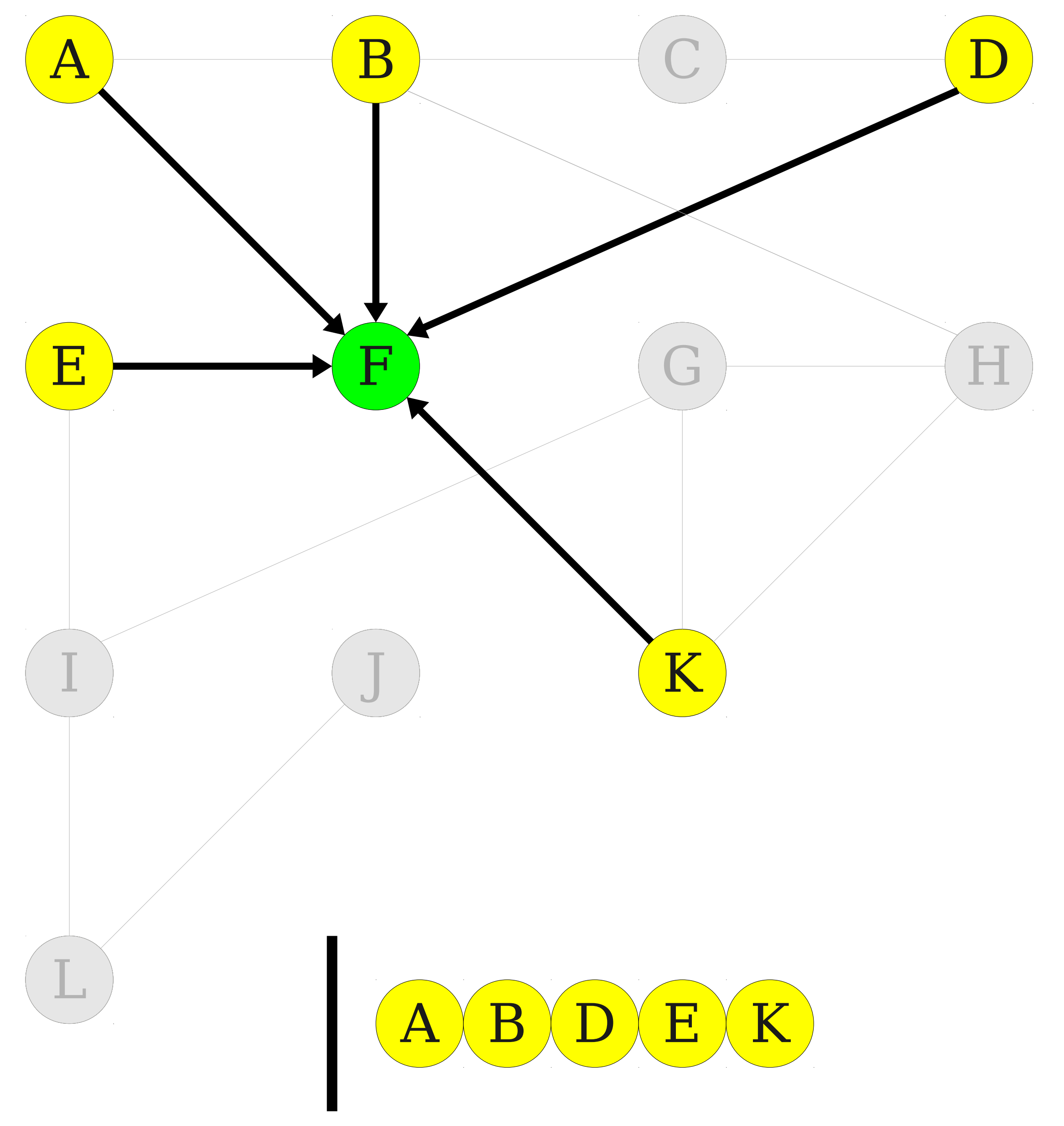
Breadth-First Search



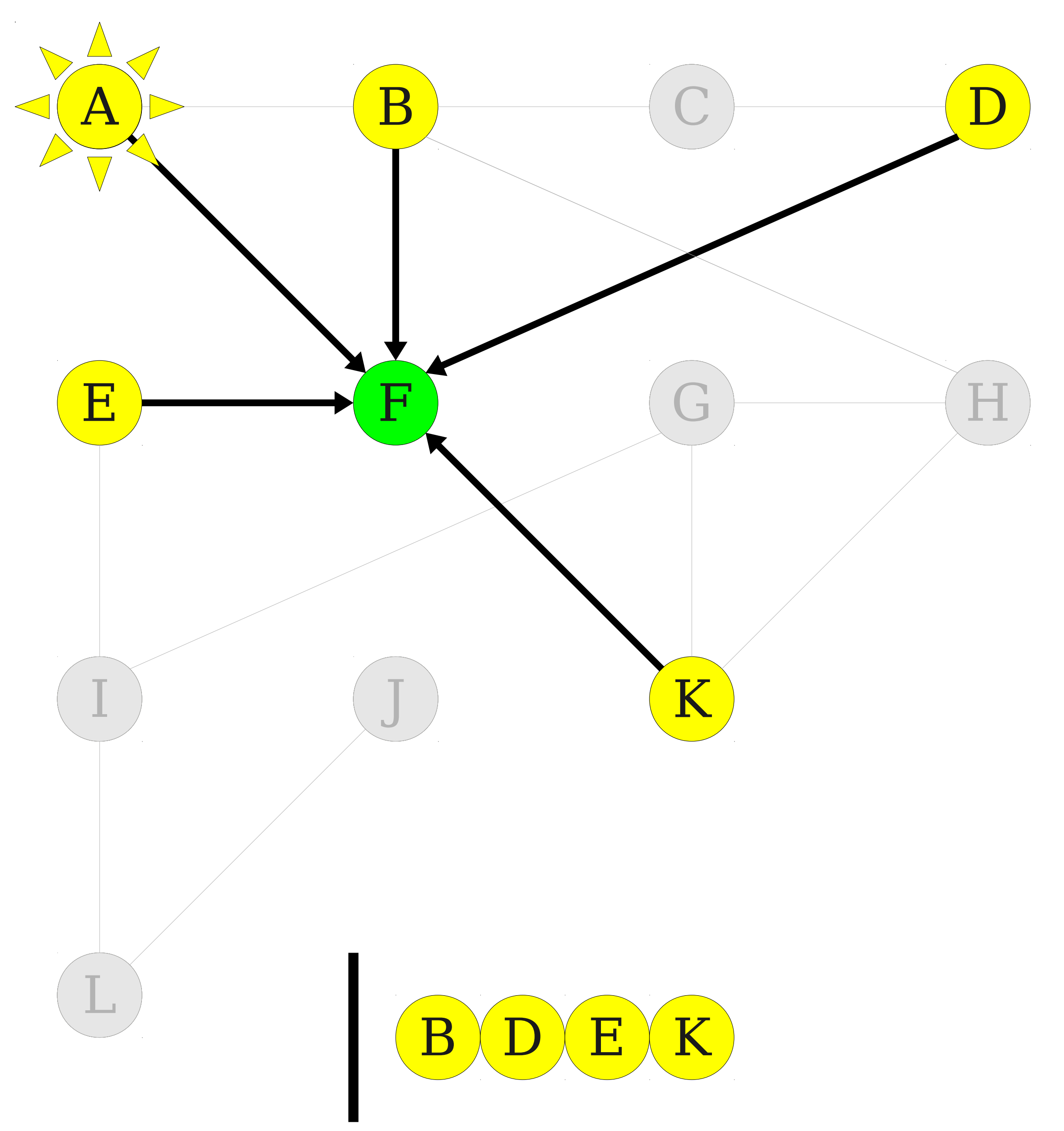
Breadth-First Search



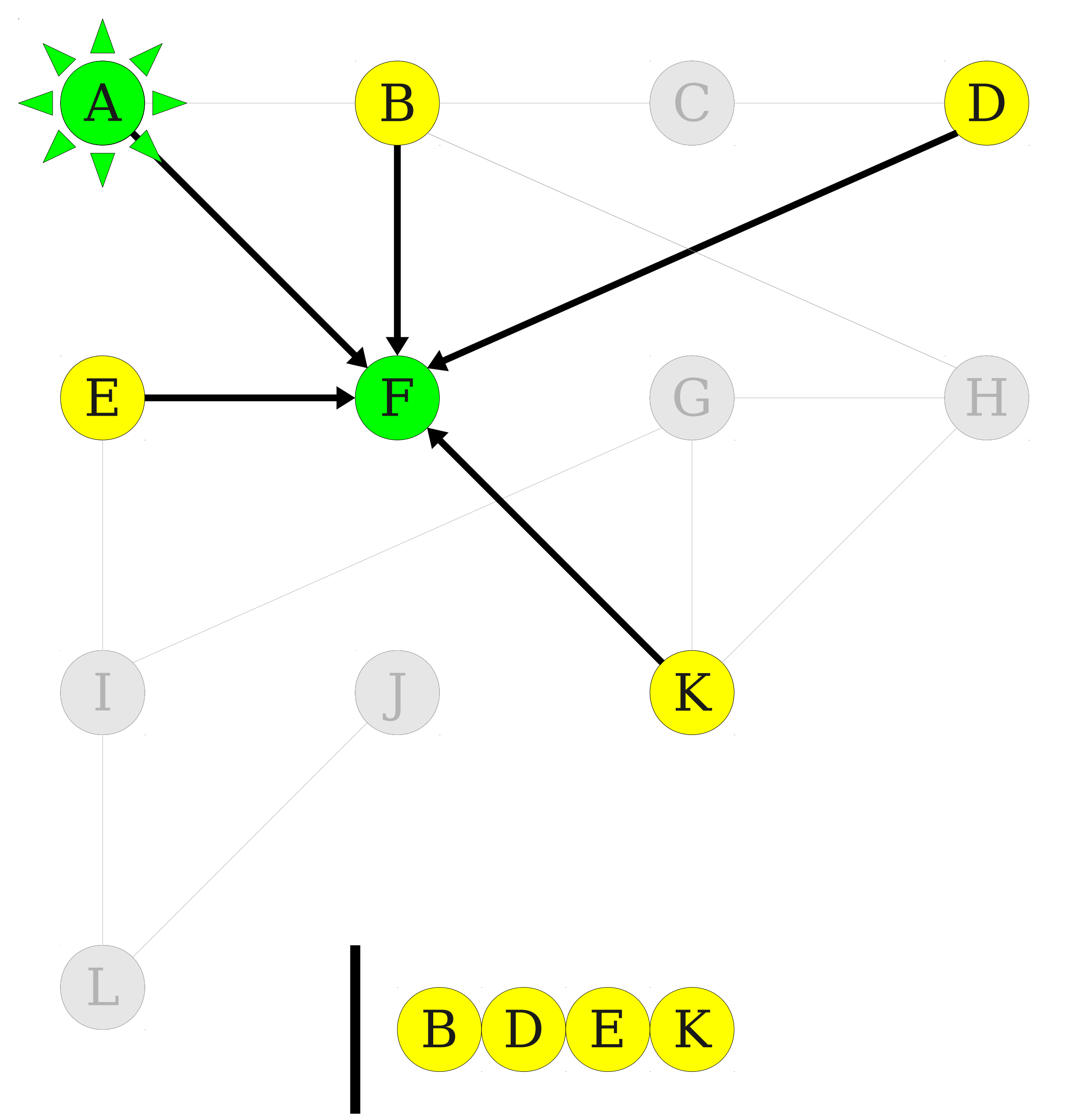
Breadth-First Search



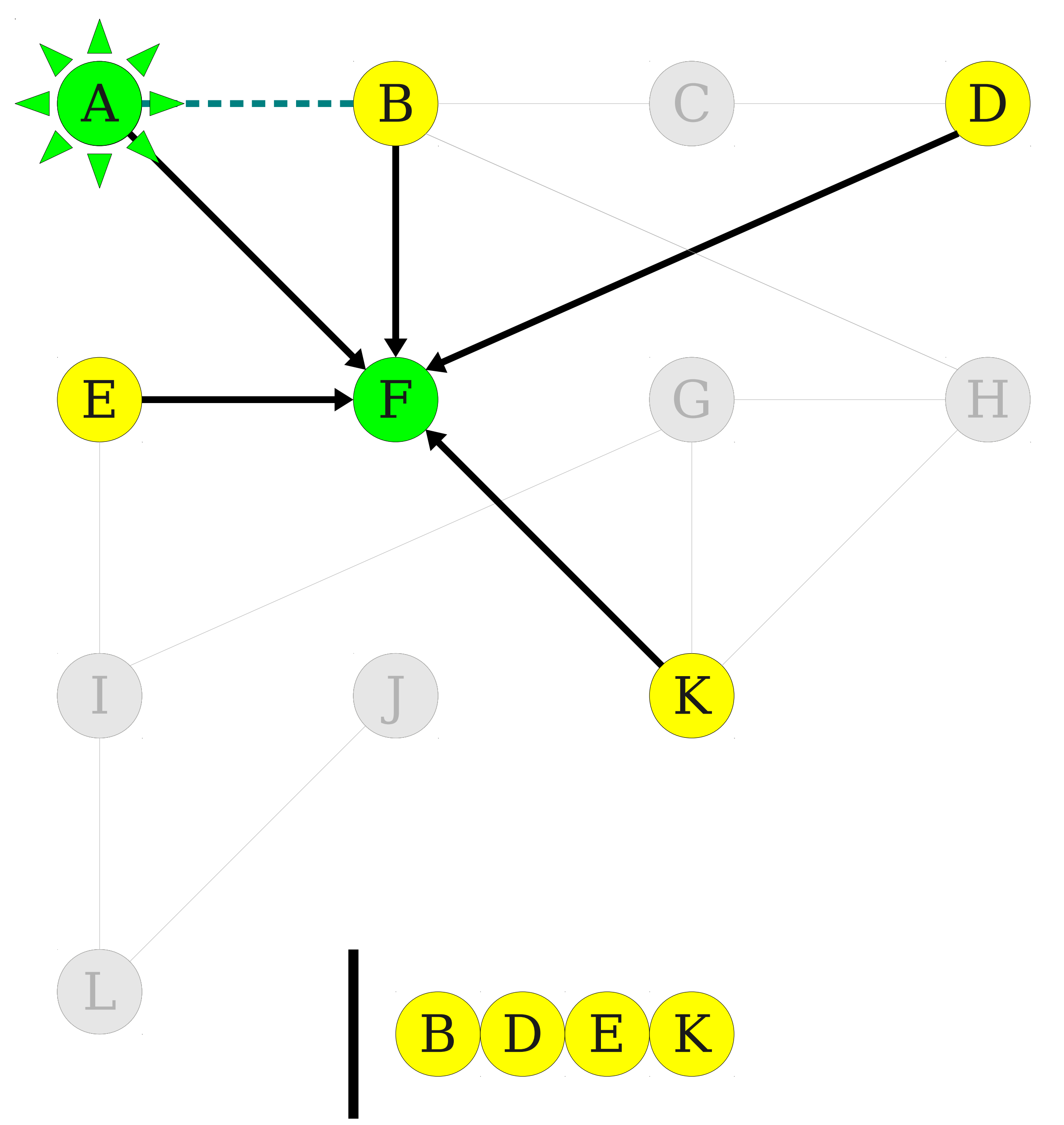
Breadth-First Search



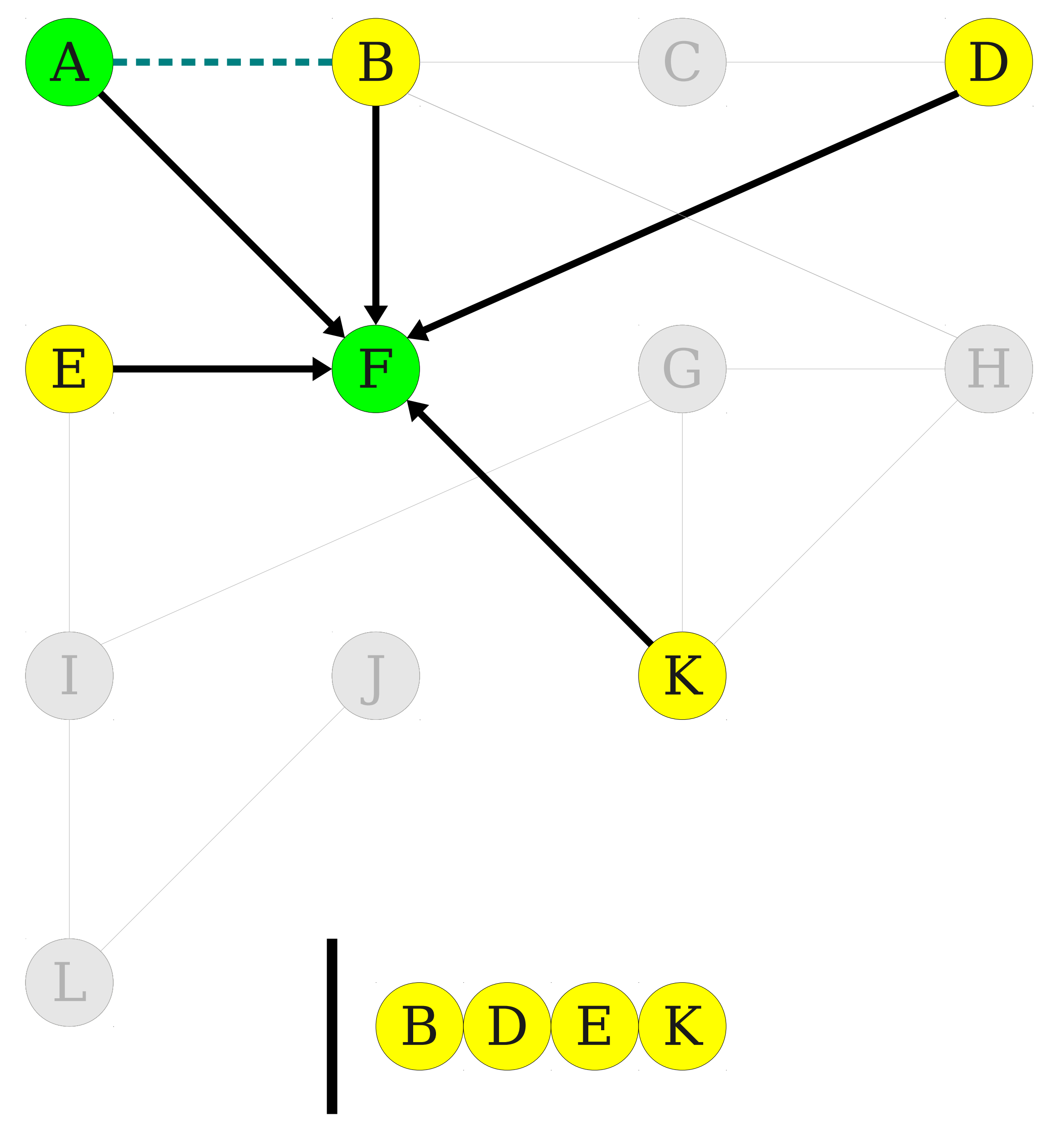
Breadth-First Search



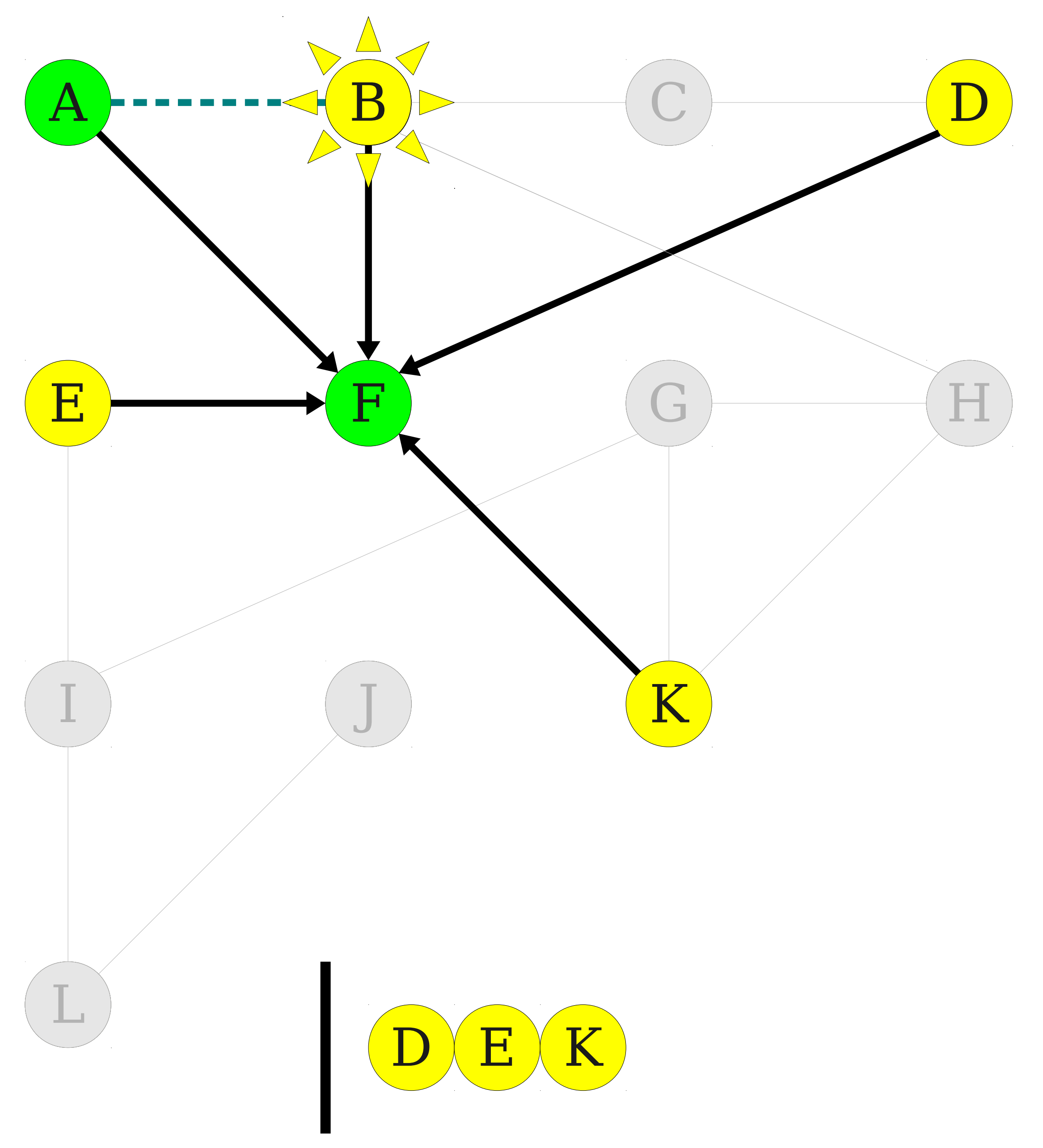
Breadth-First Search



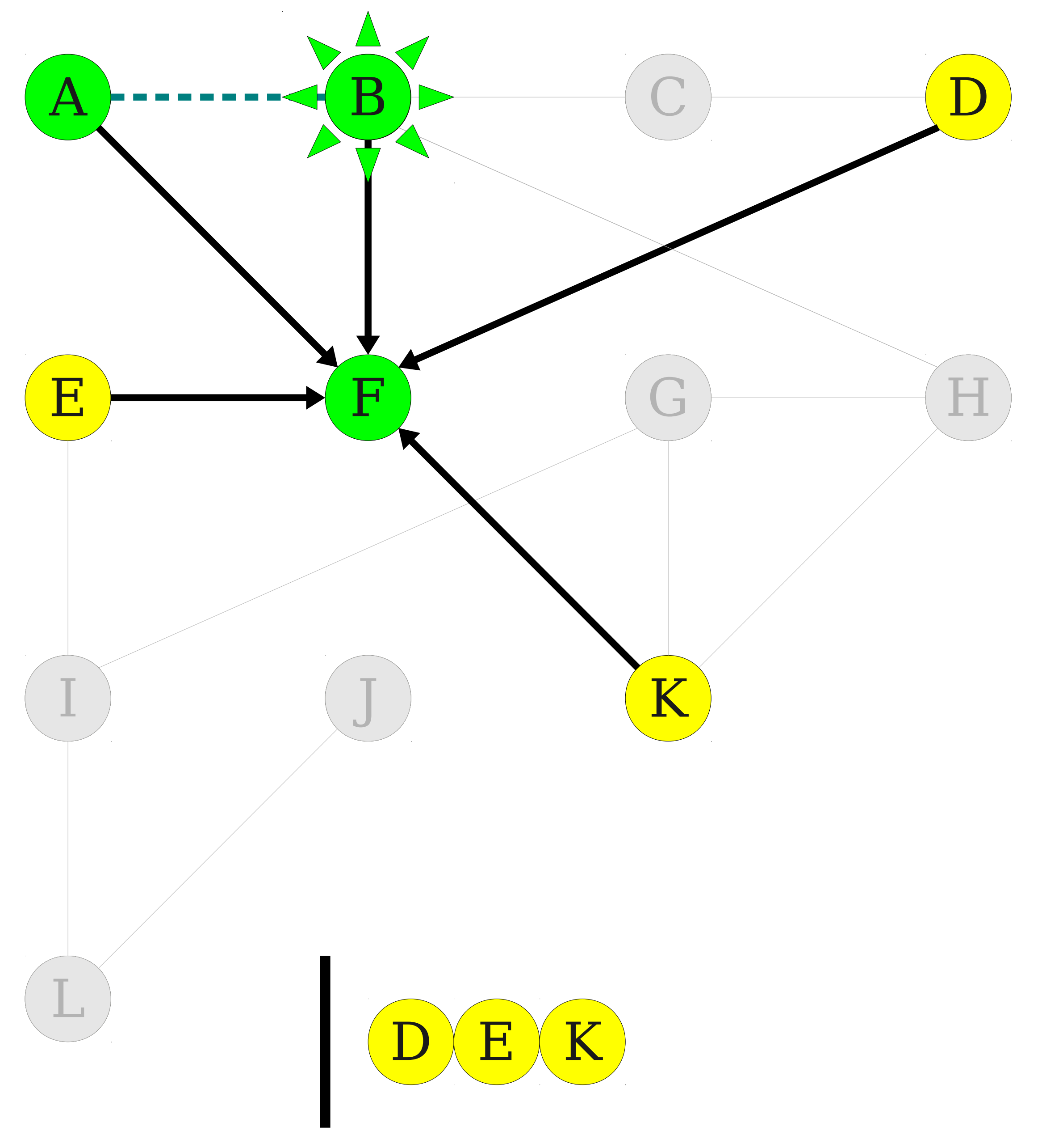
Breadth-First Search



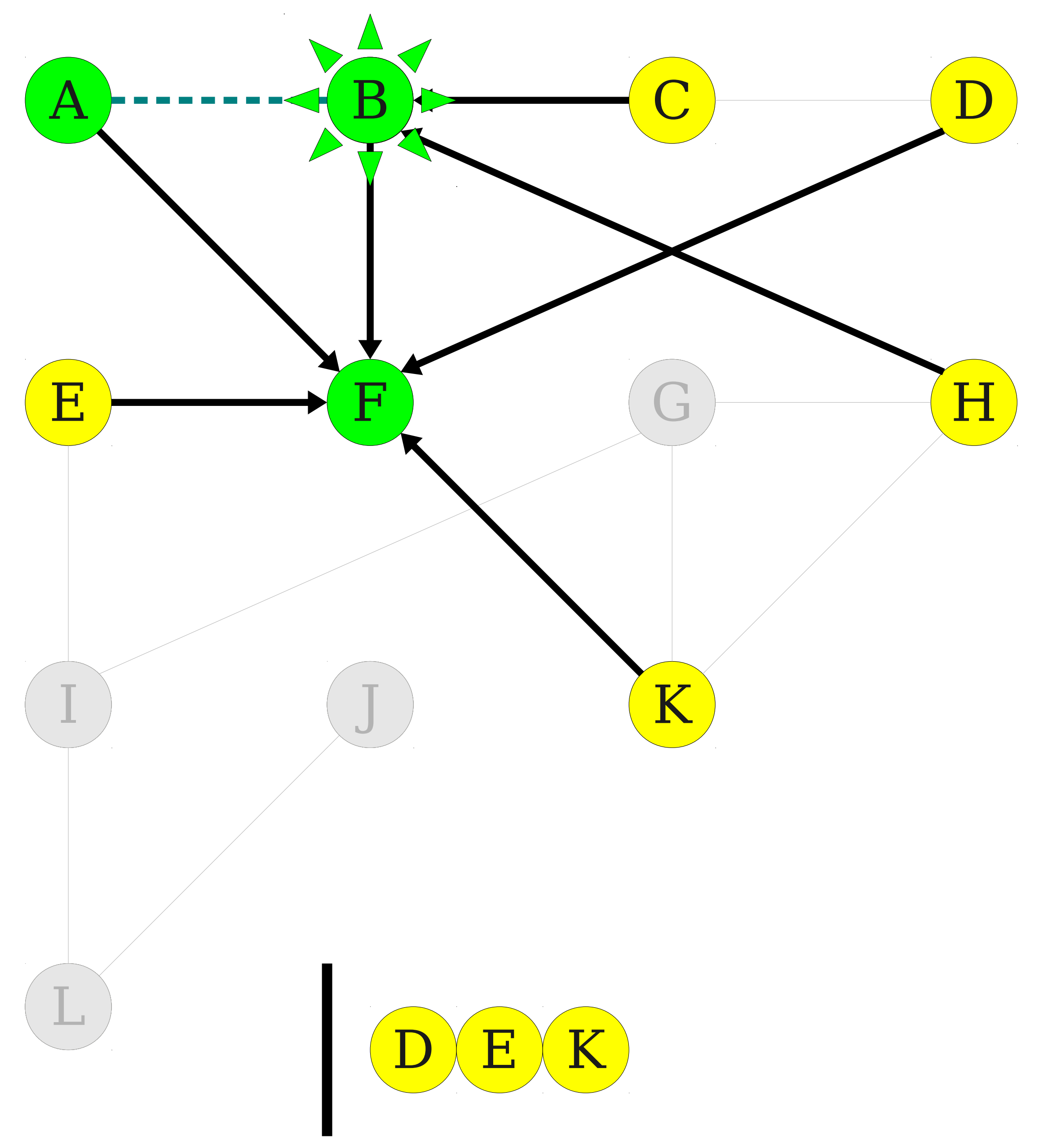
Breadth-First Search



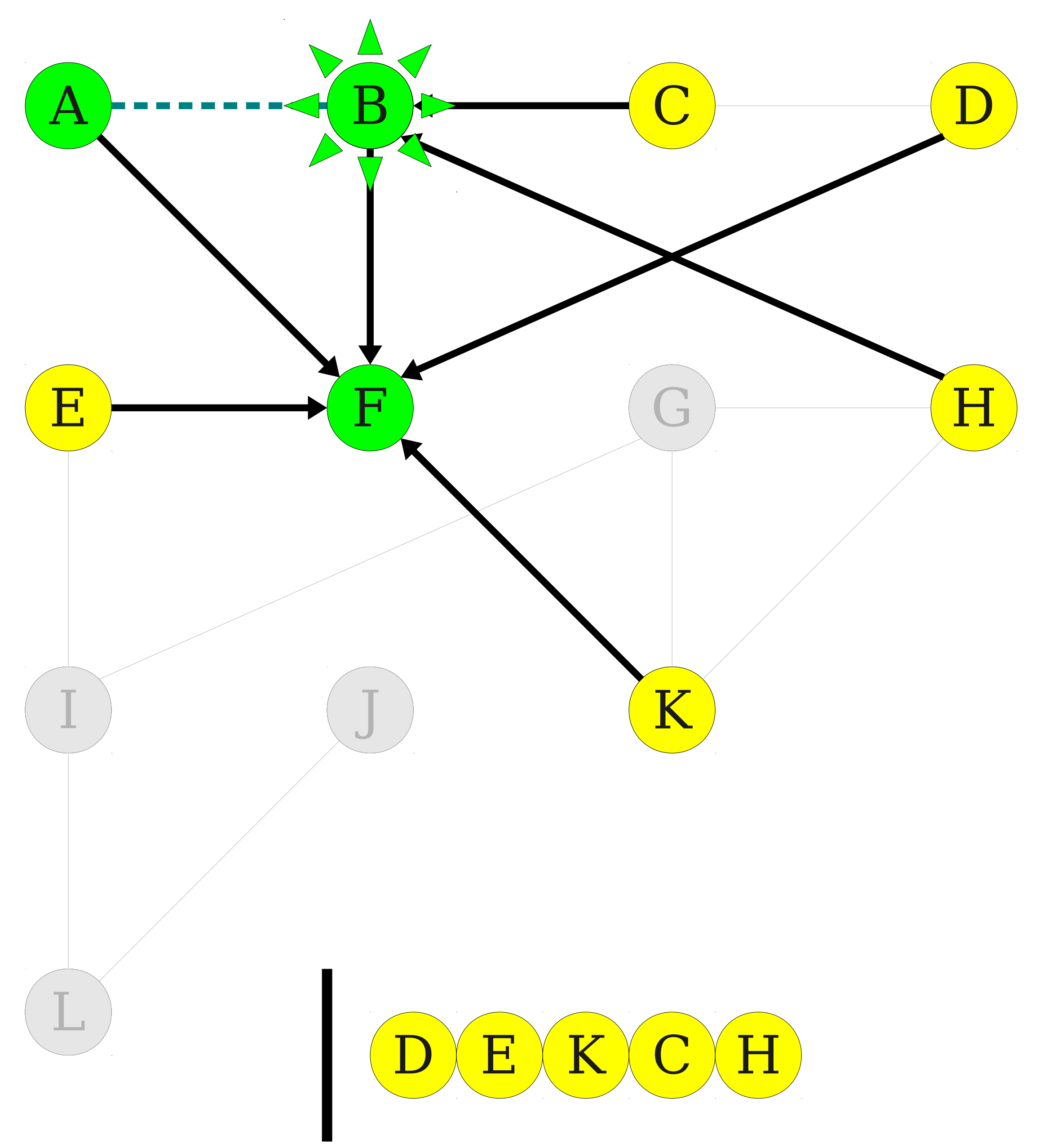
Breadth-First Search



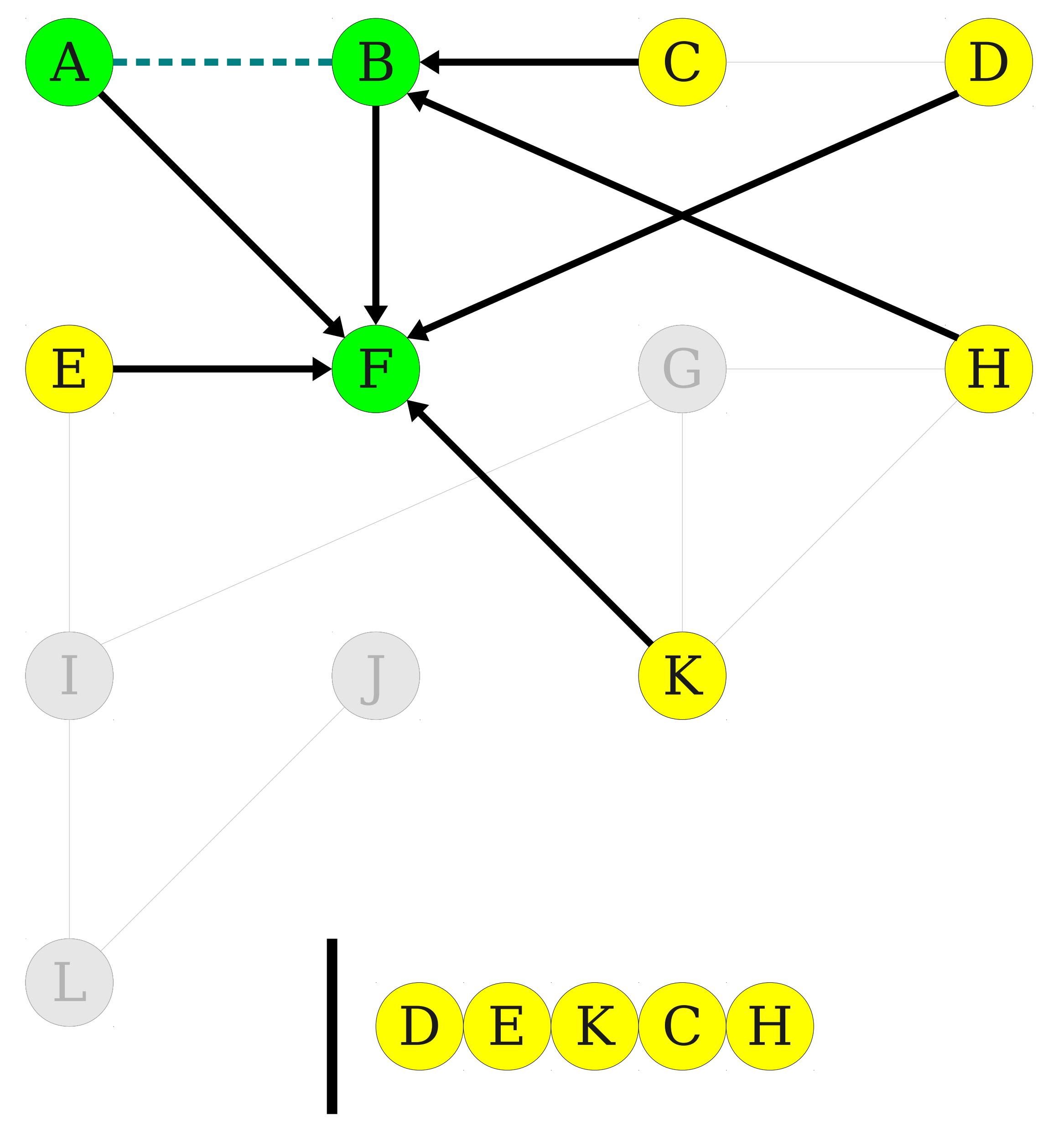
Breadth-First Search



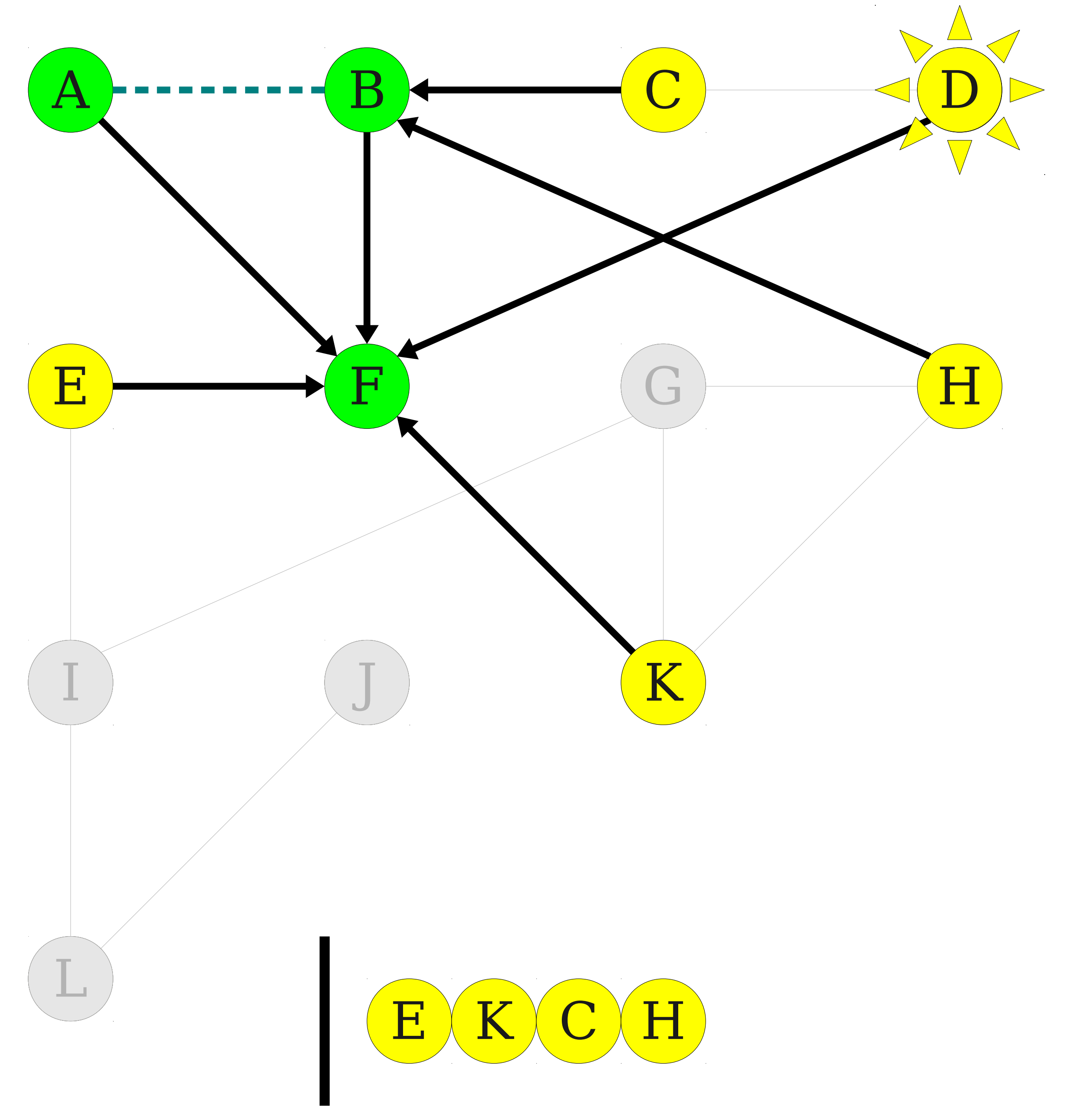
Breadth-First Search



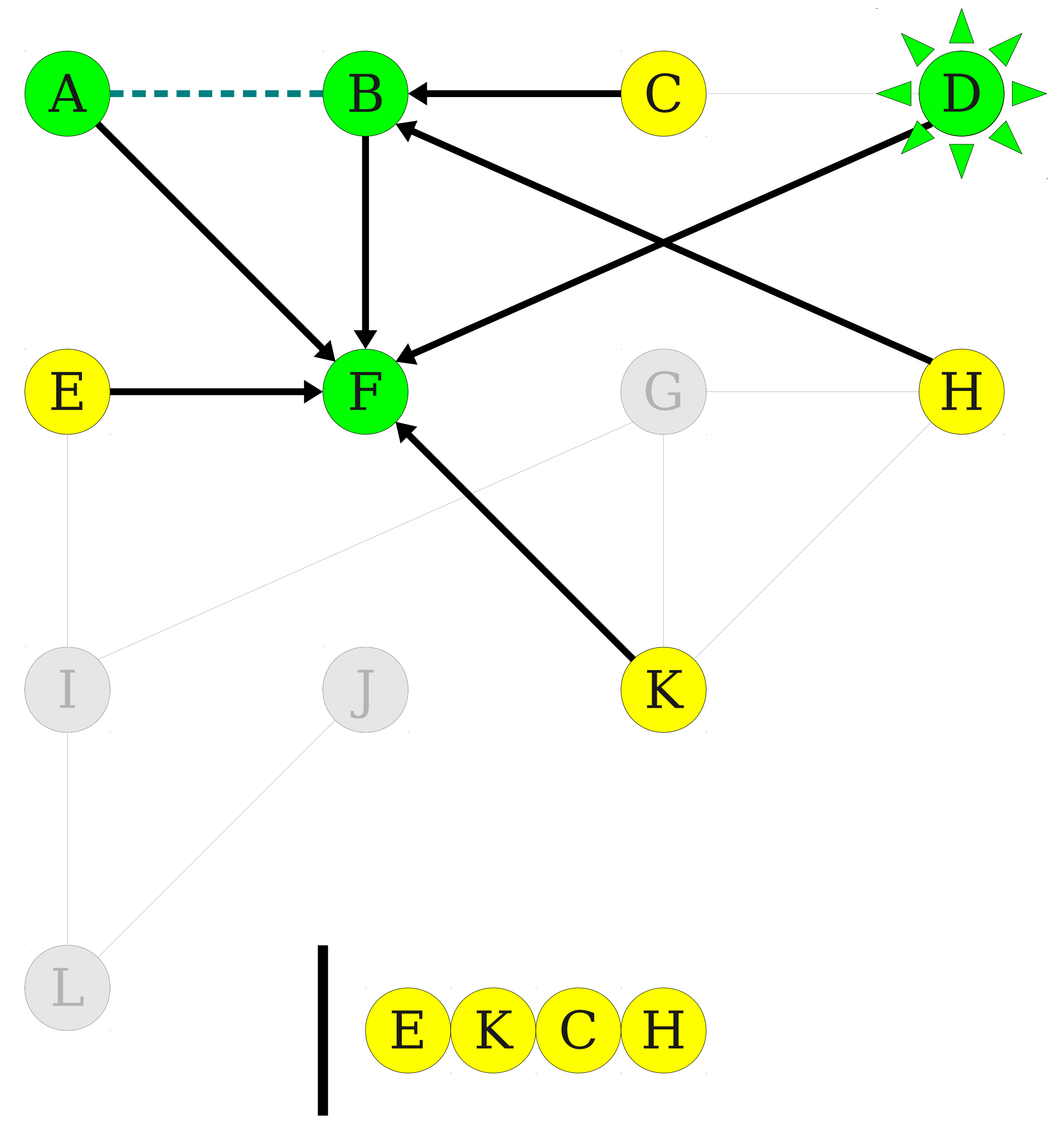
Breadth-First Search



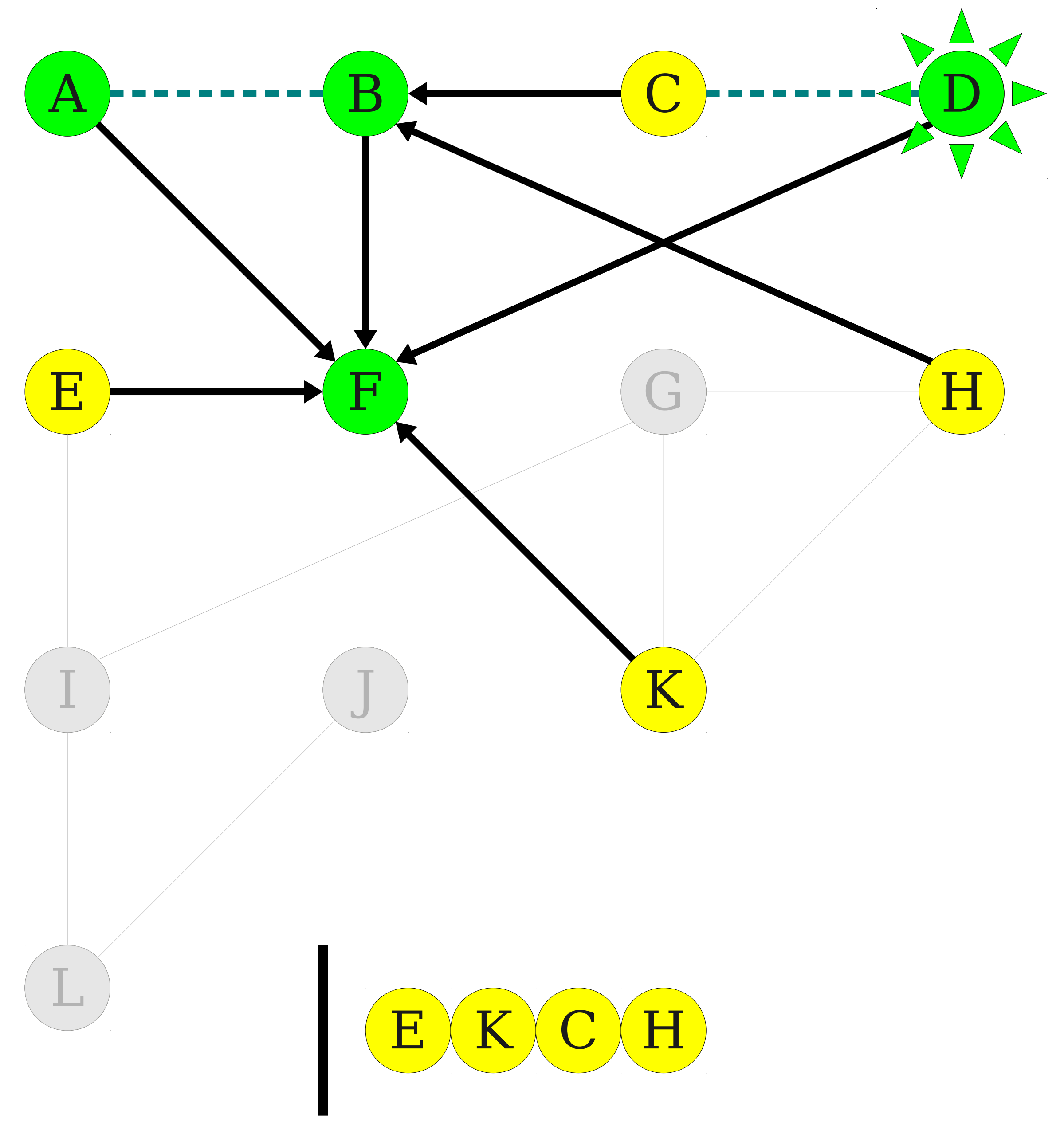
Breadth-First Search



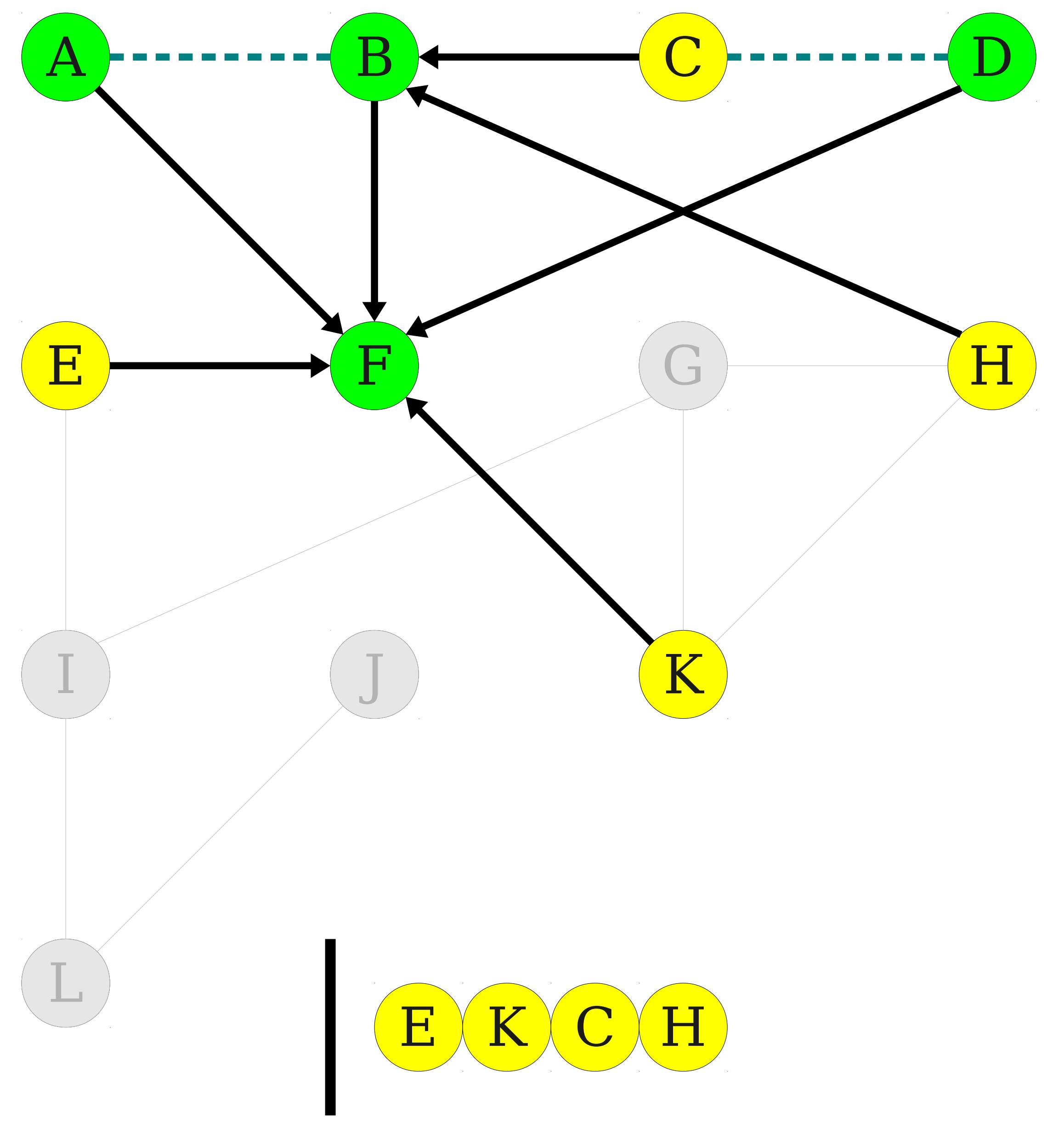
Breadth-First Search



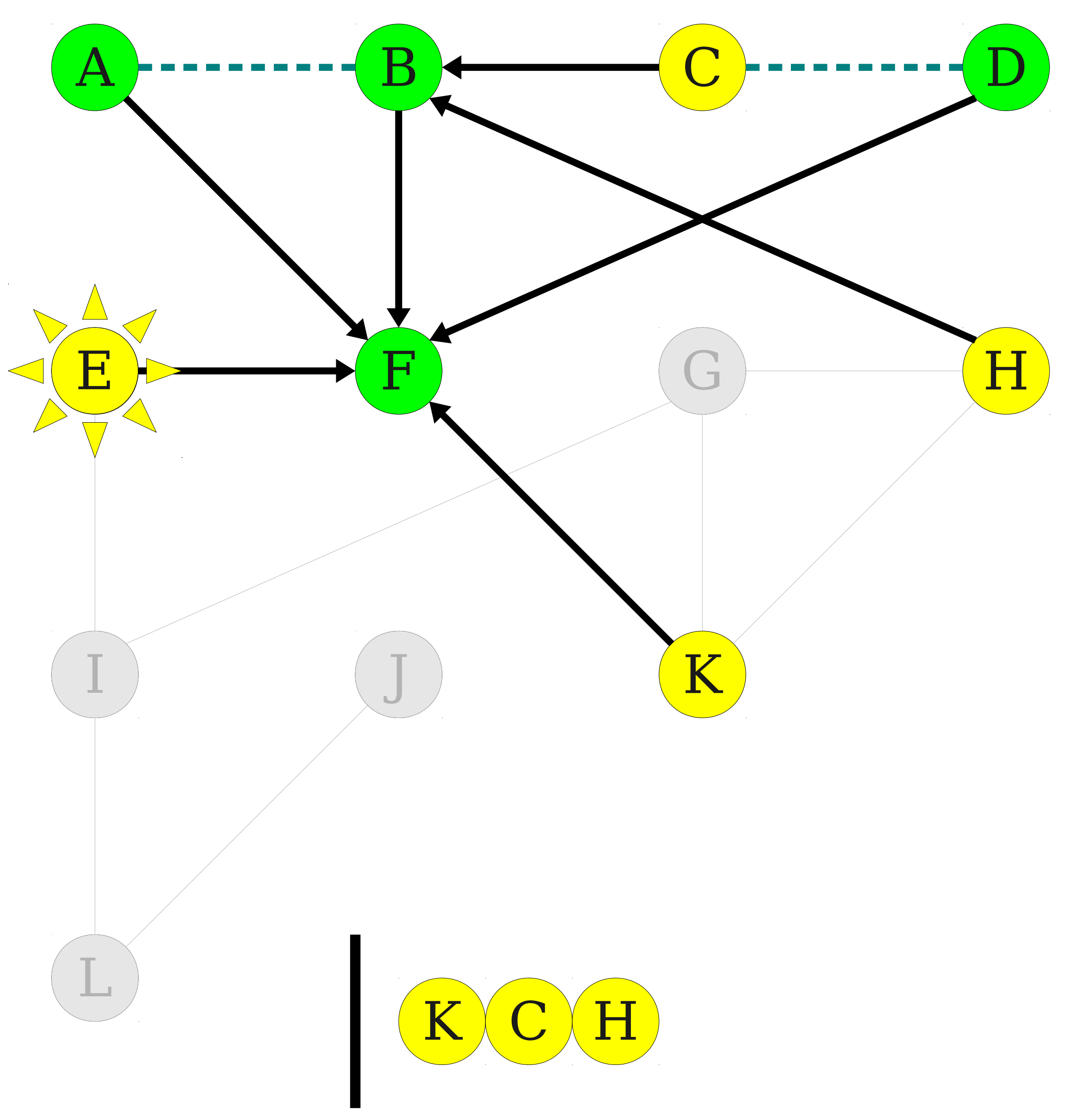
Breadth-First Search



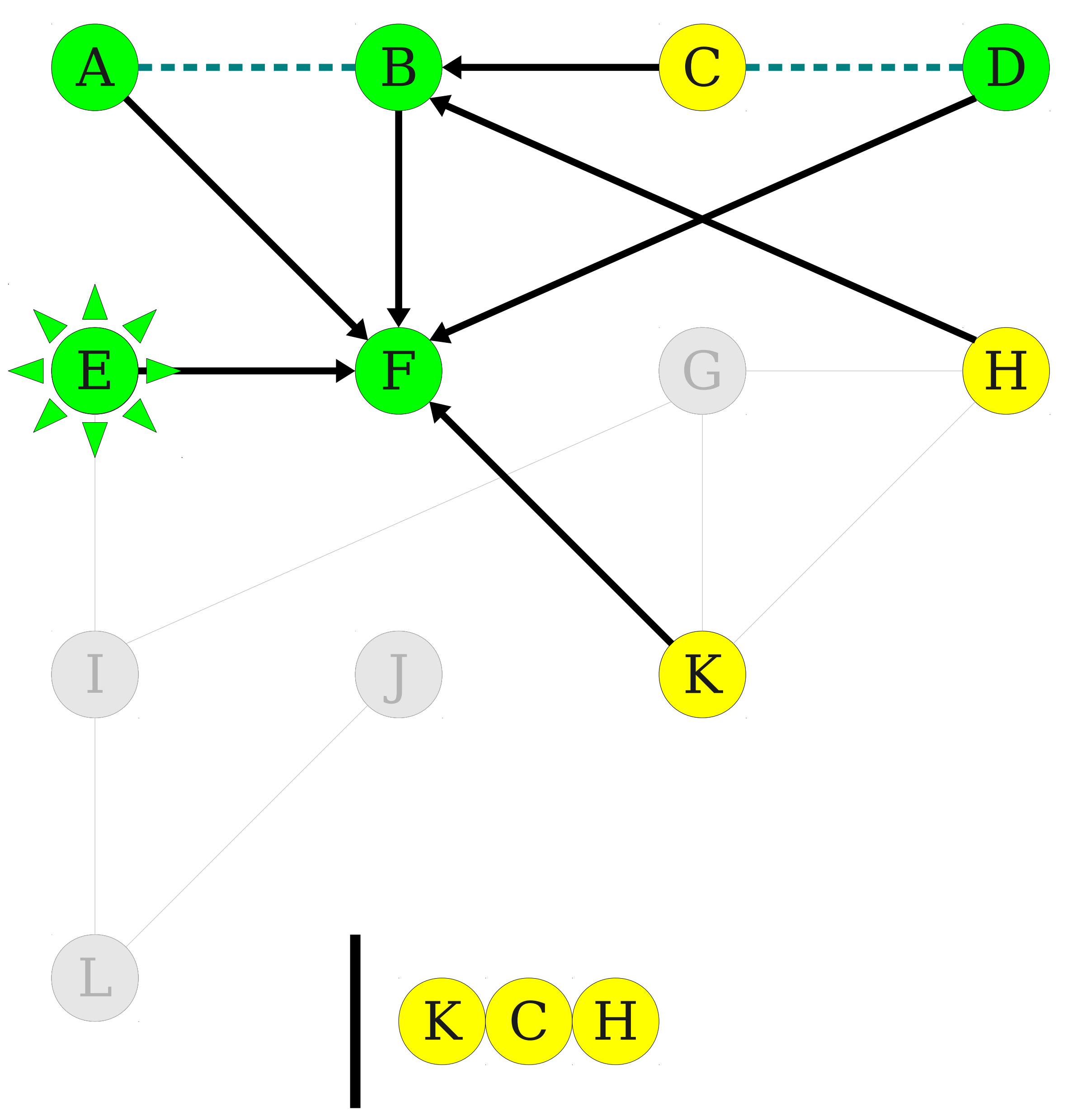
Breadth-First Search



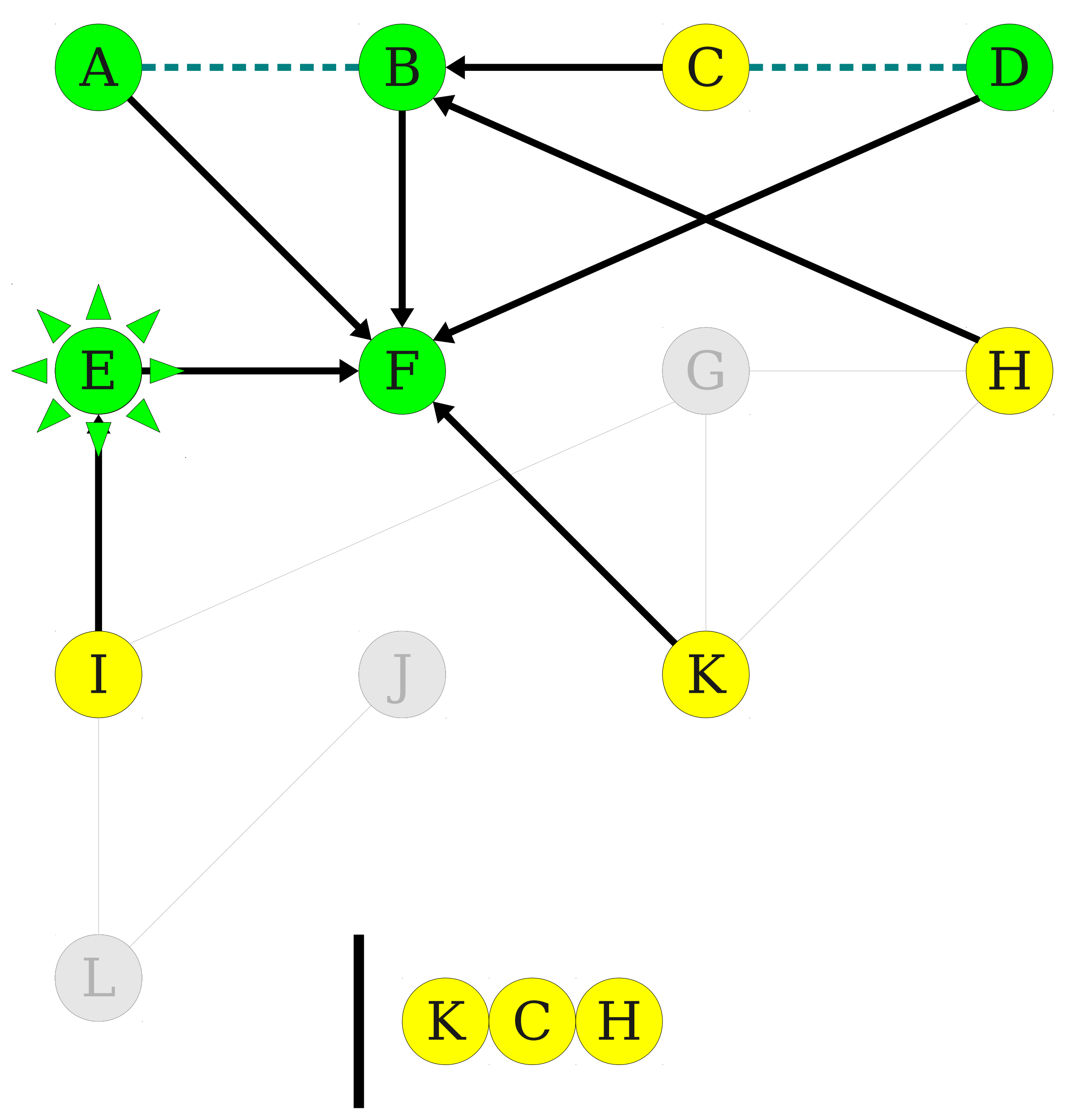
Breadth-First Search



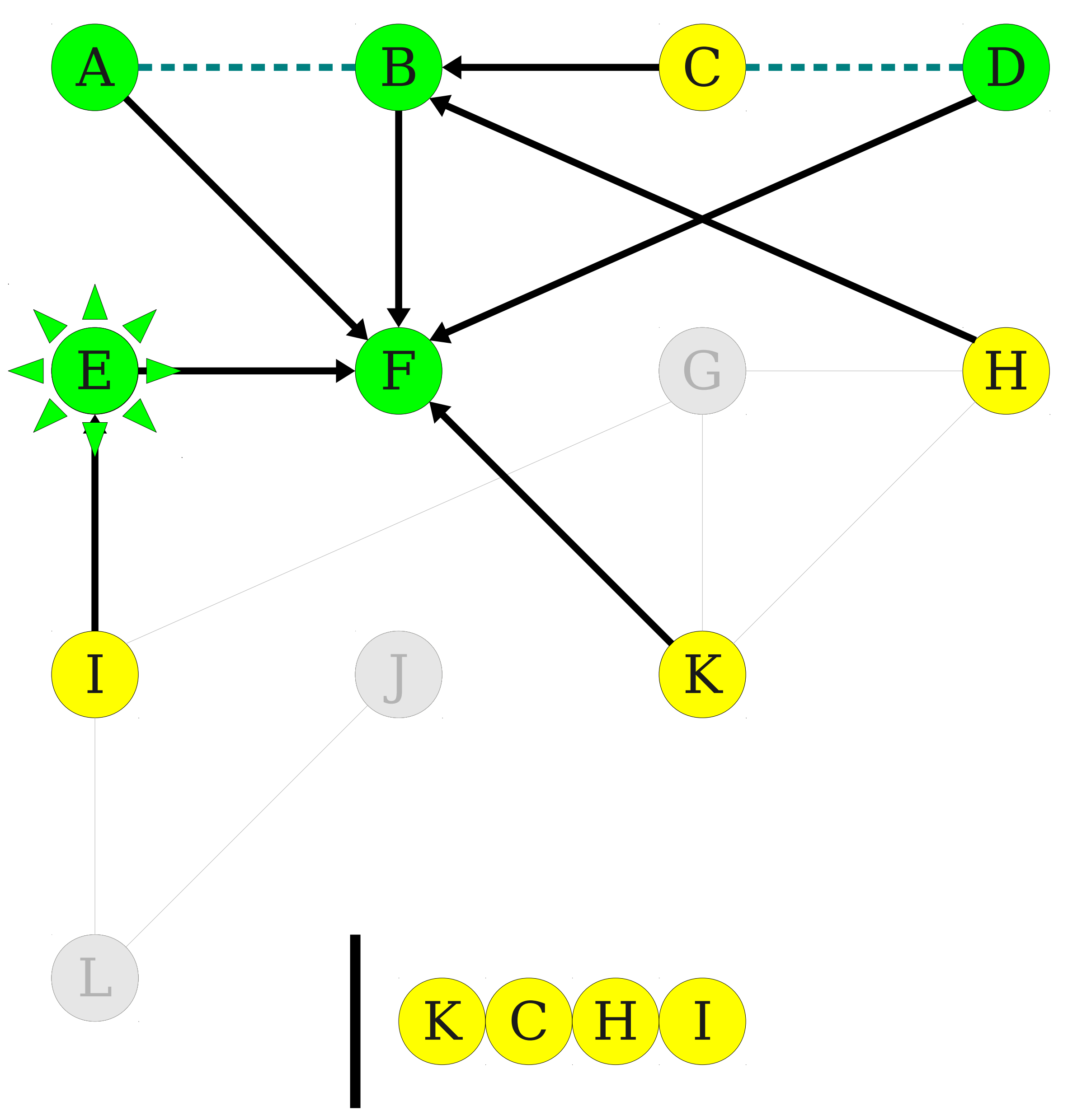
Breadth-First Search



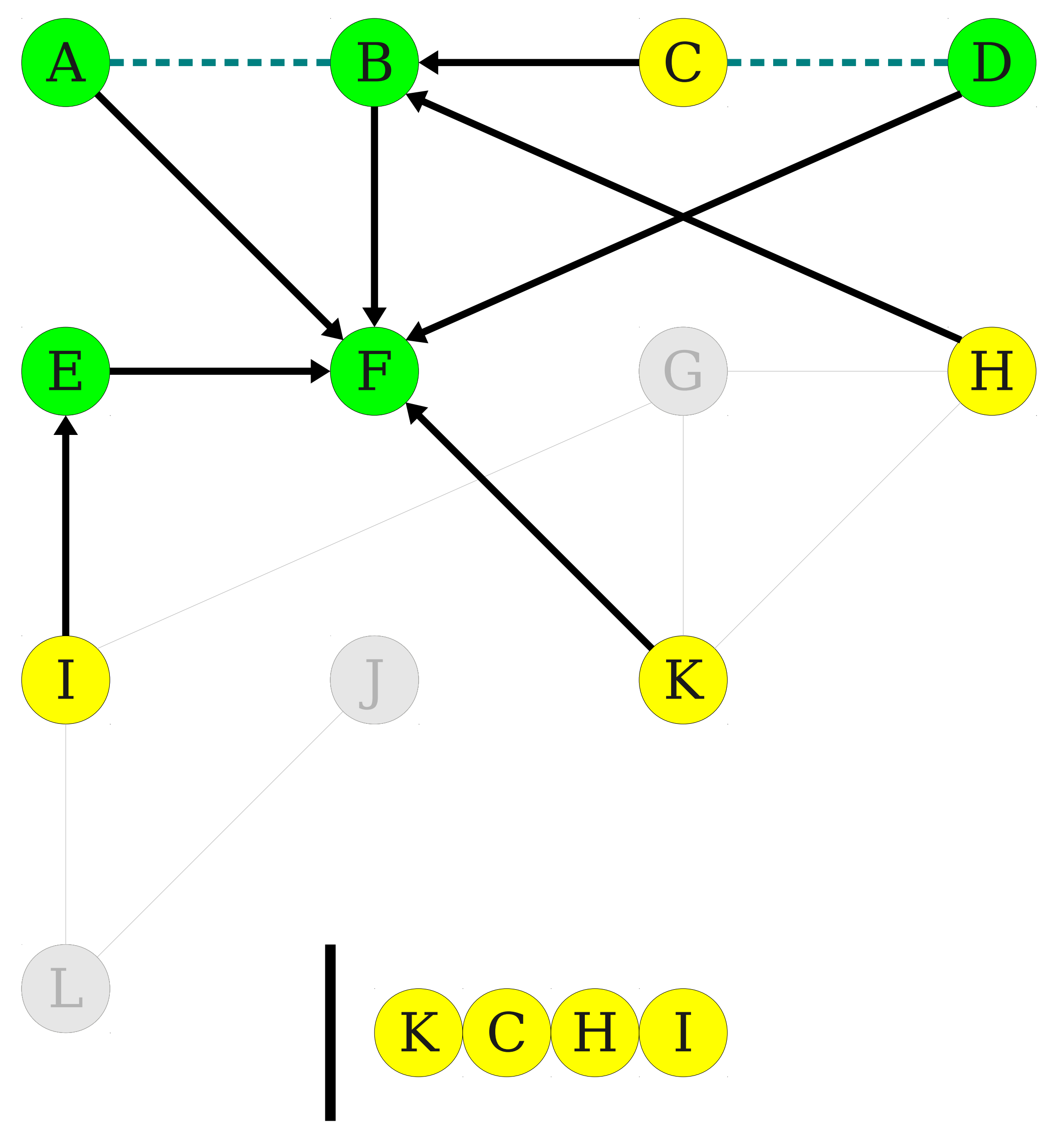
Breadth-First Search



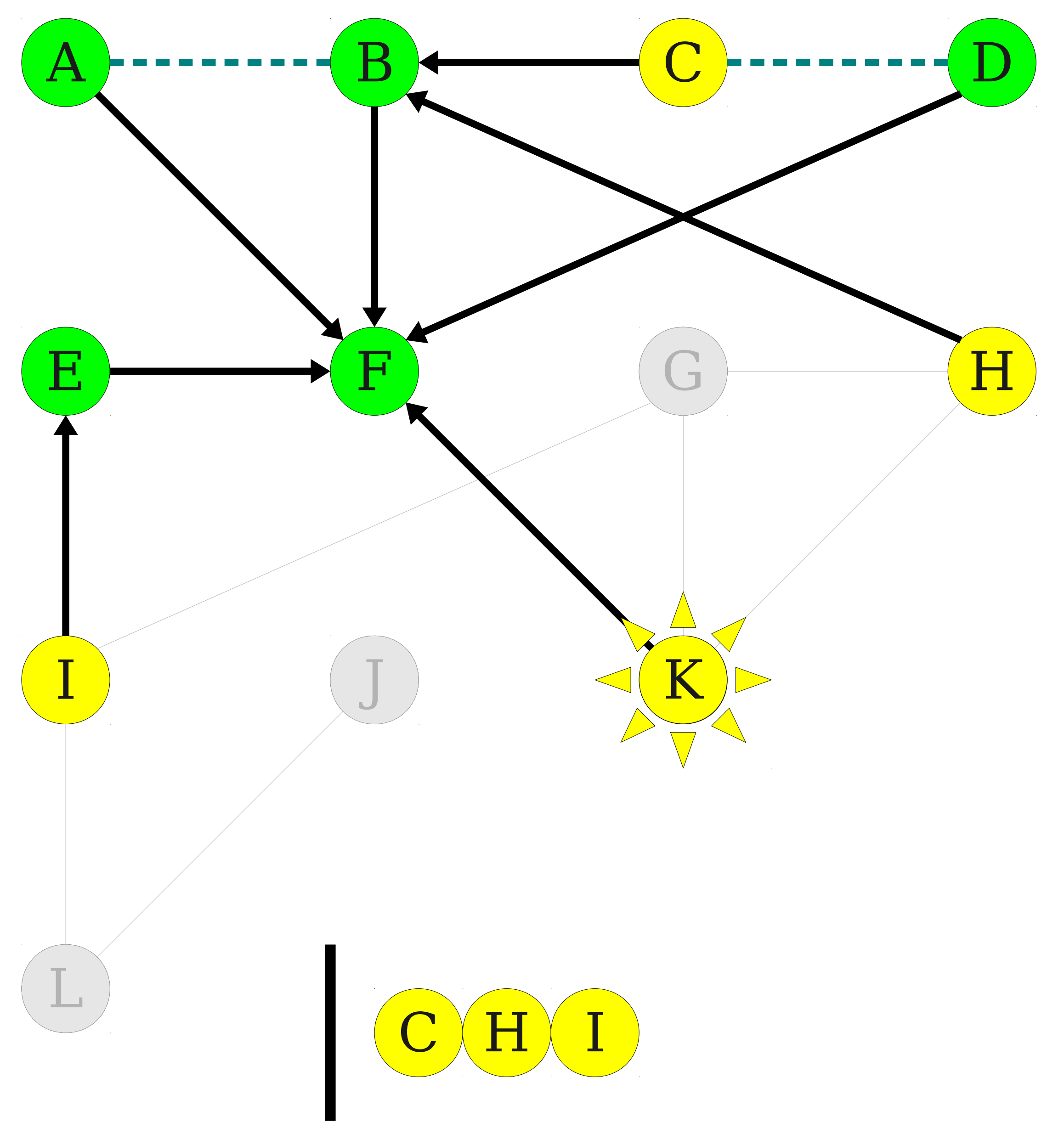
Breadth-First Search



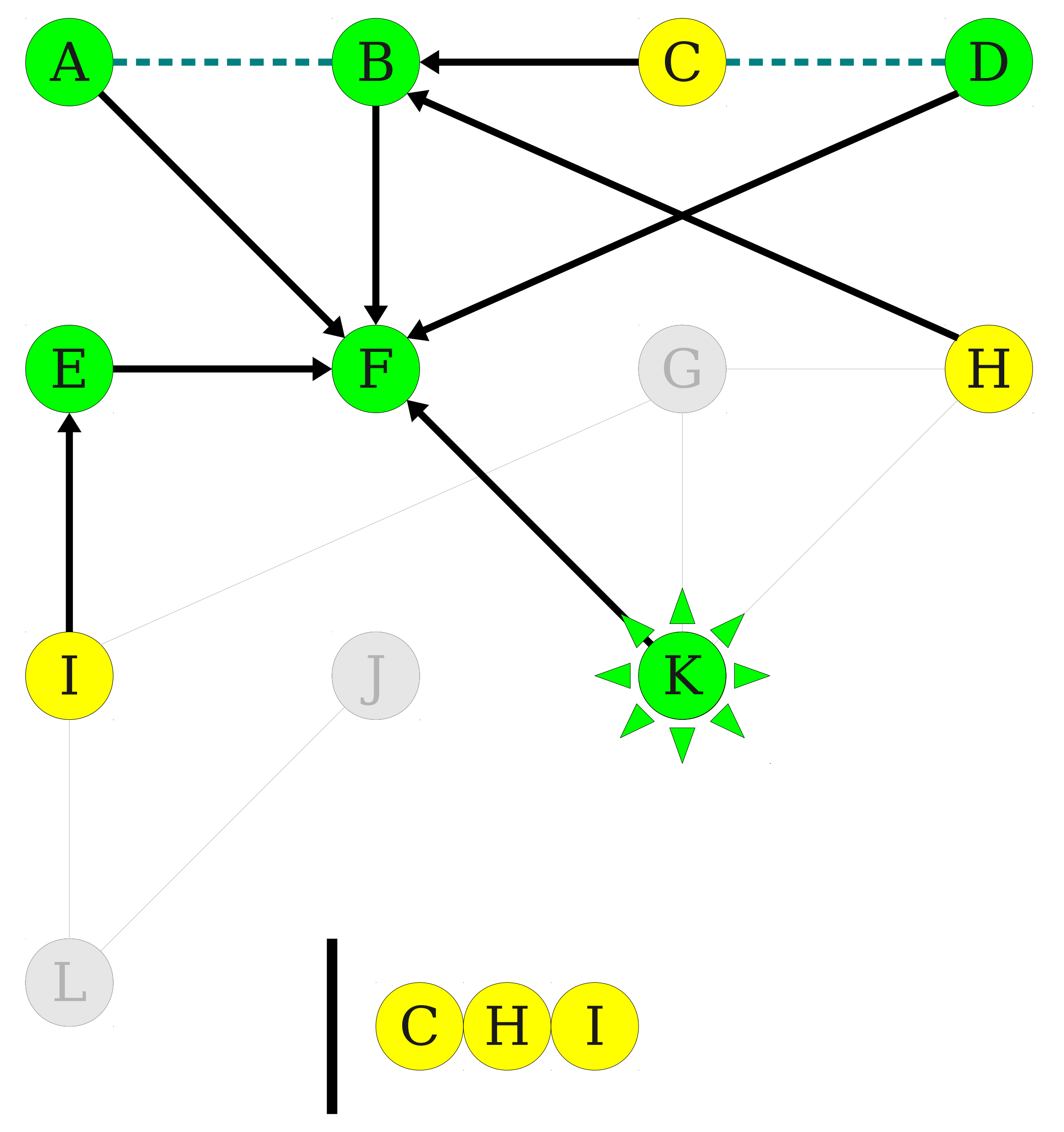
Breadth-First Search



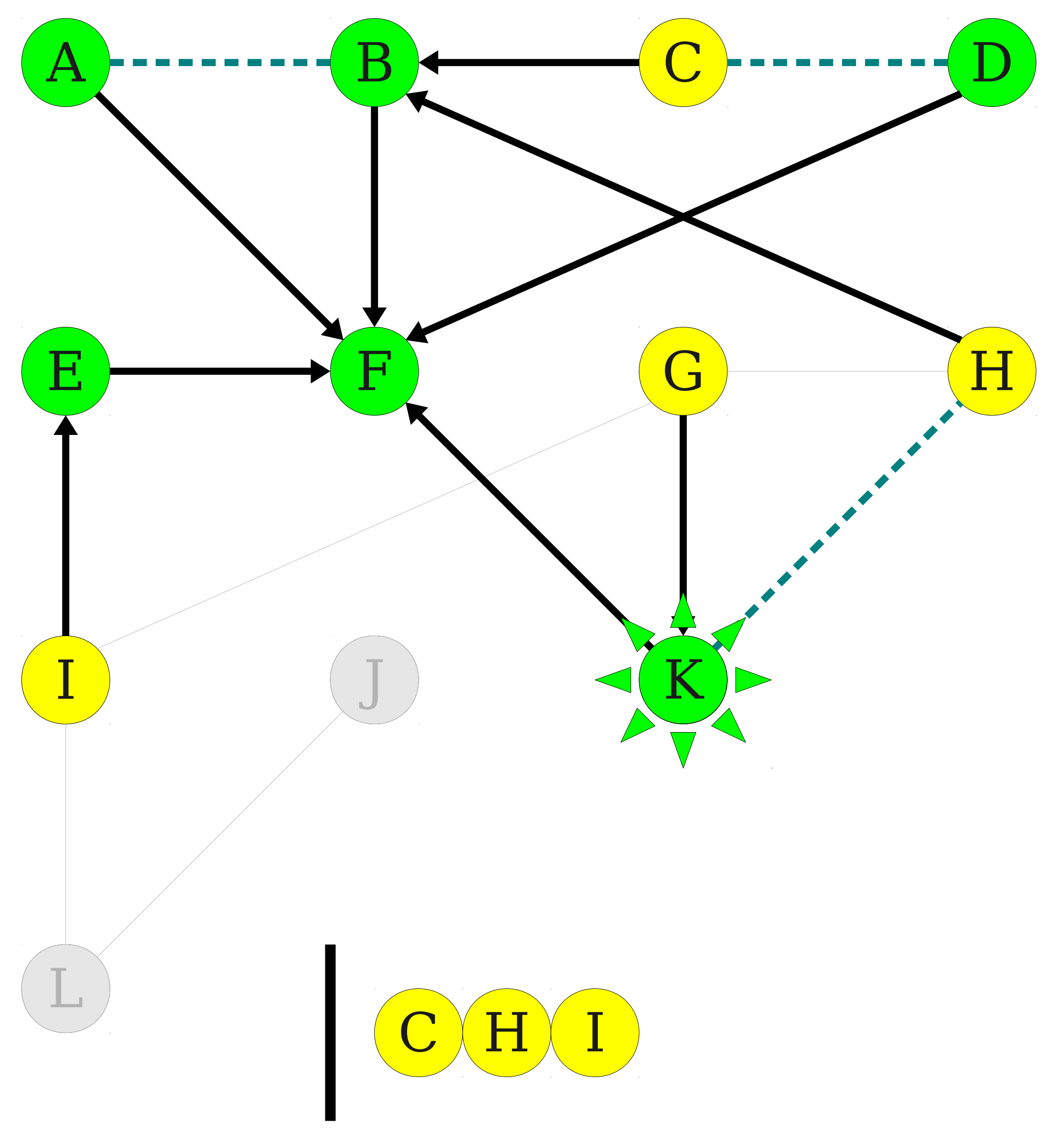
Breadth-First Search



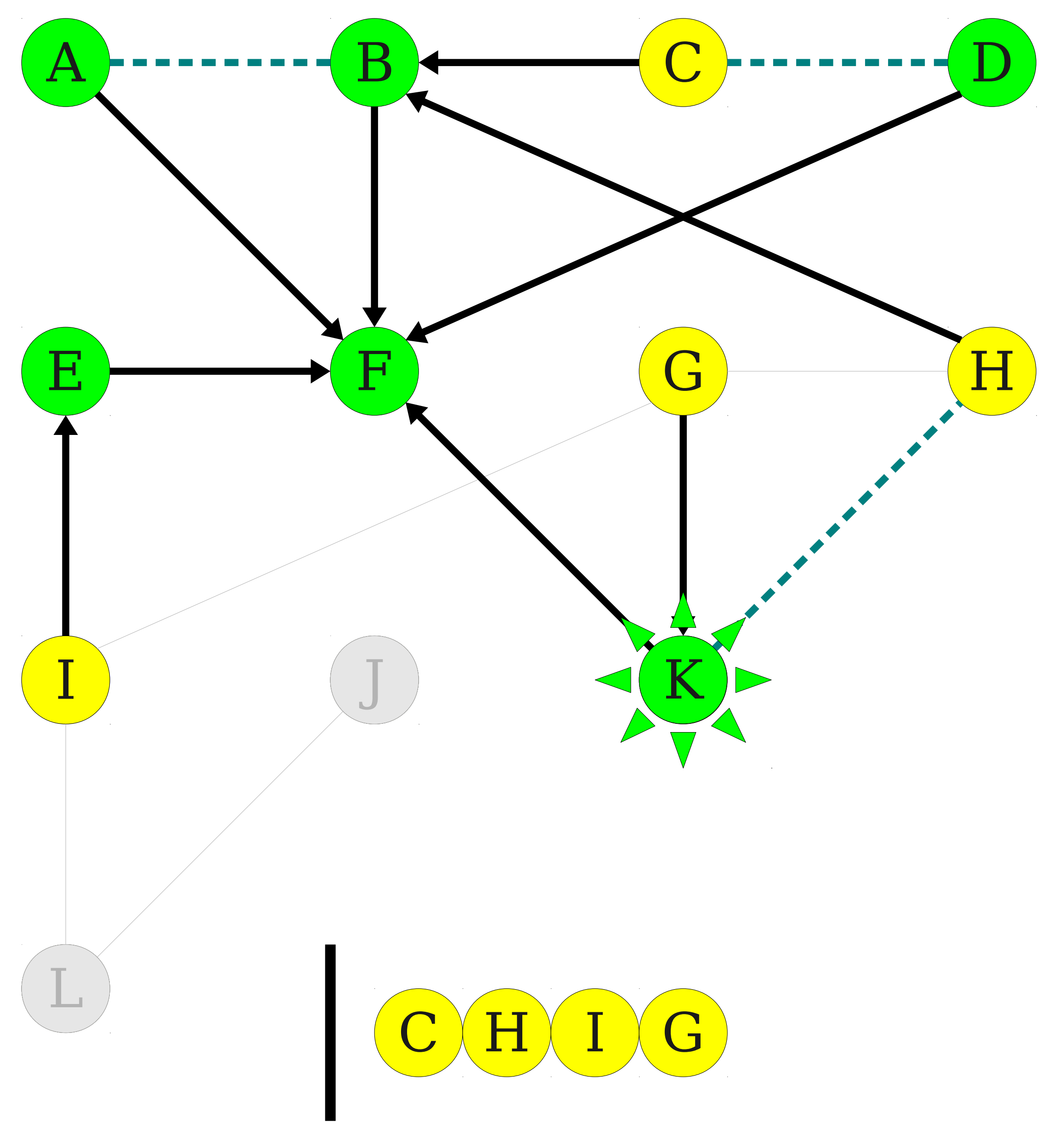
Breadth-First Search



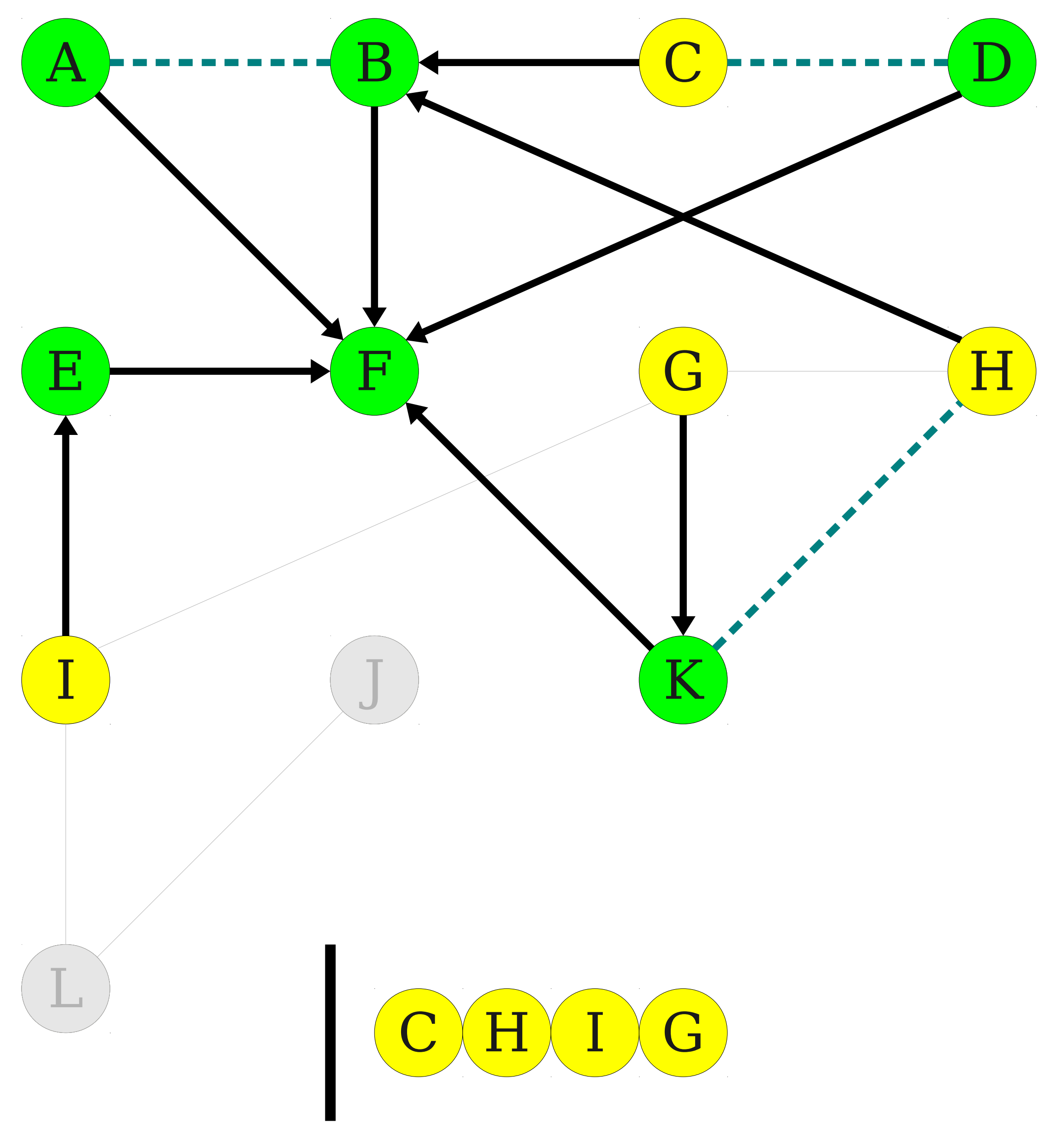
Breadth-First Search



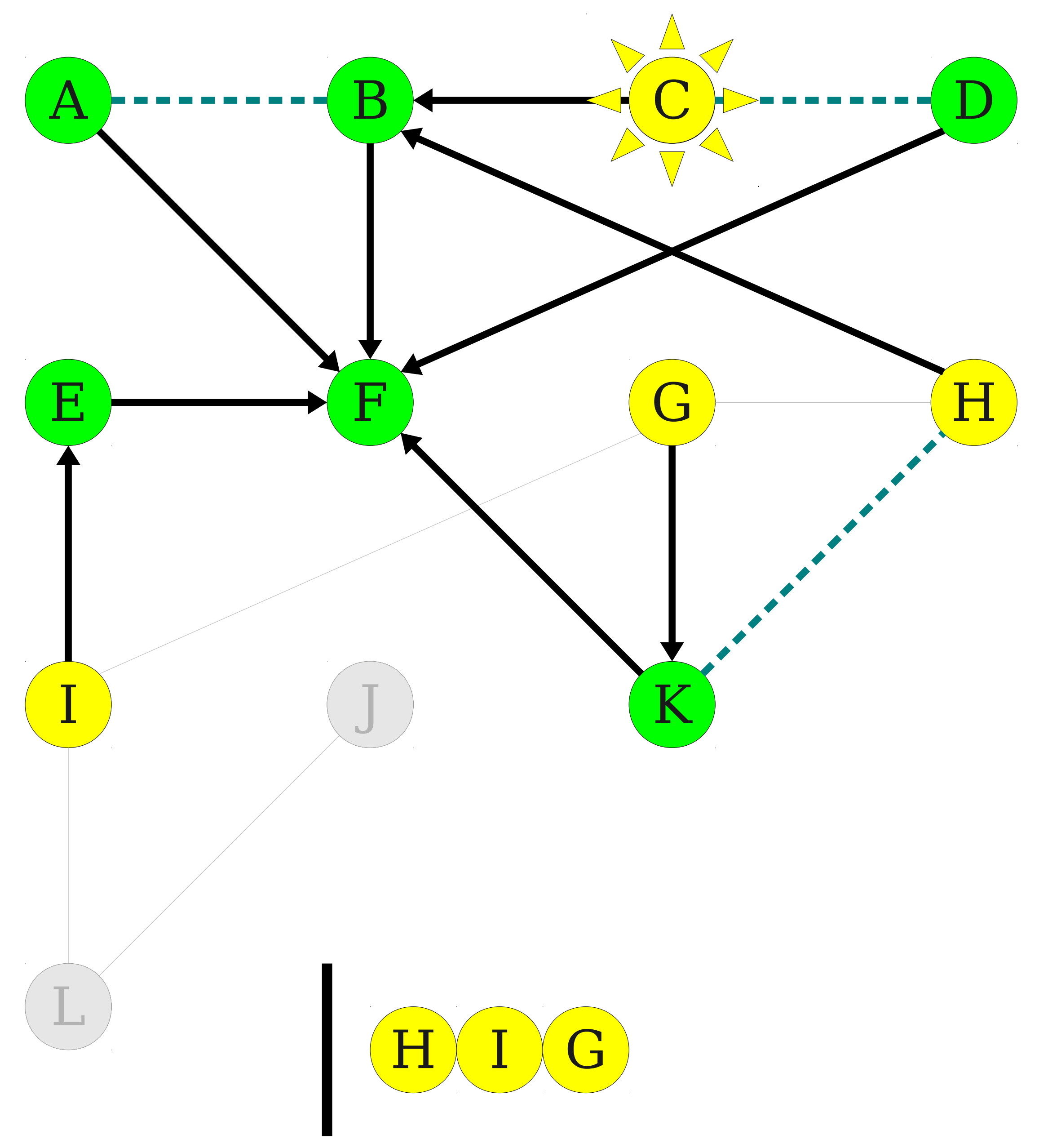
Breadth-First Search



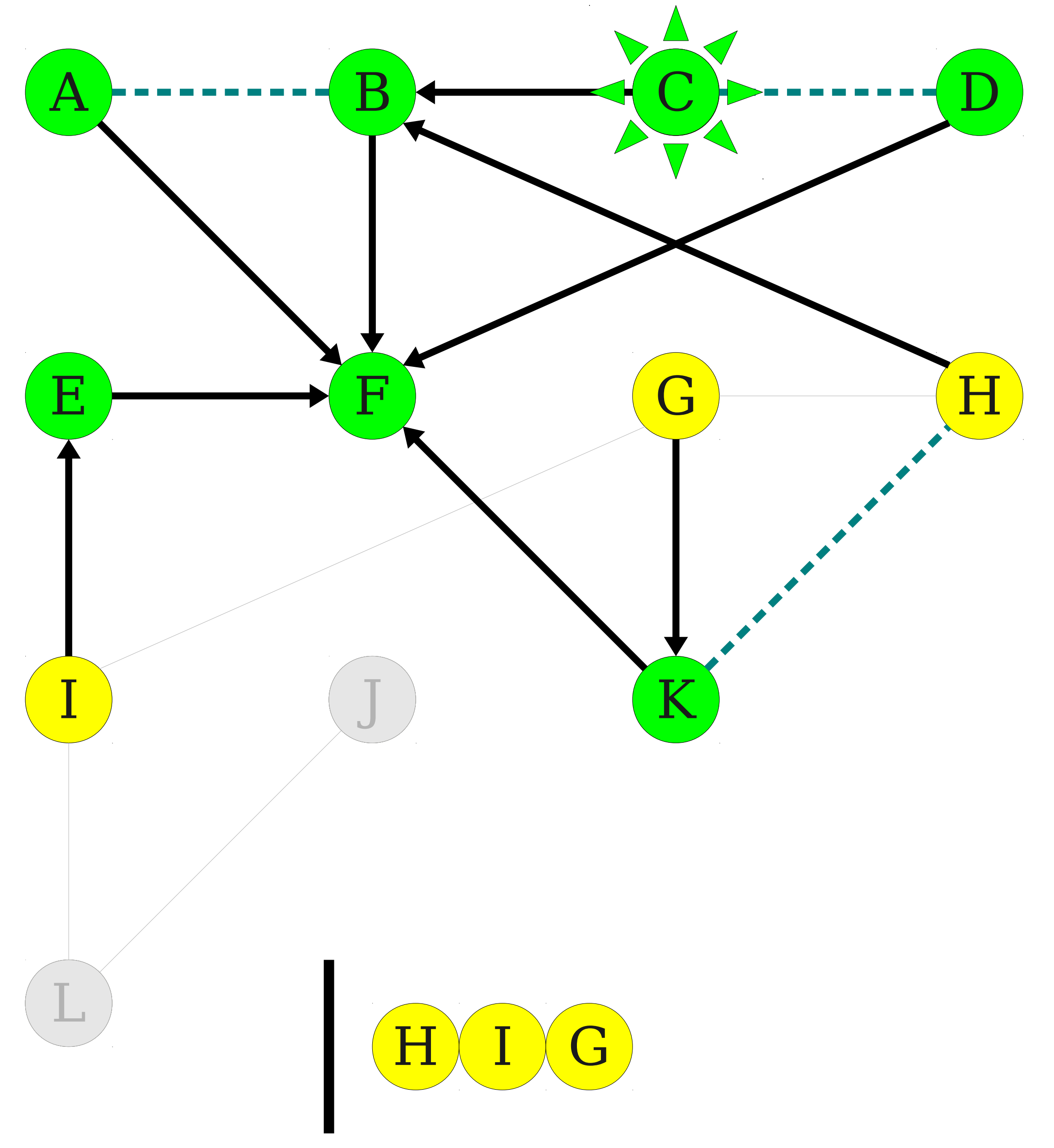
Breadth-First Search



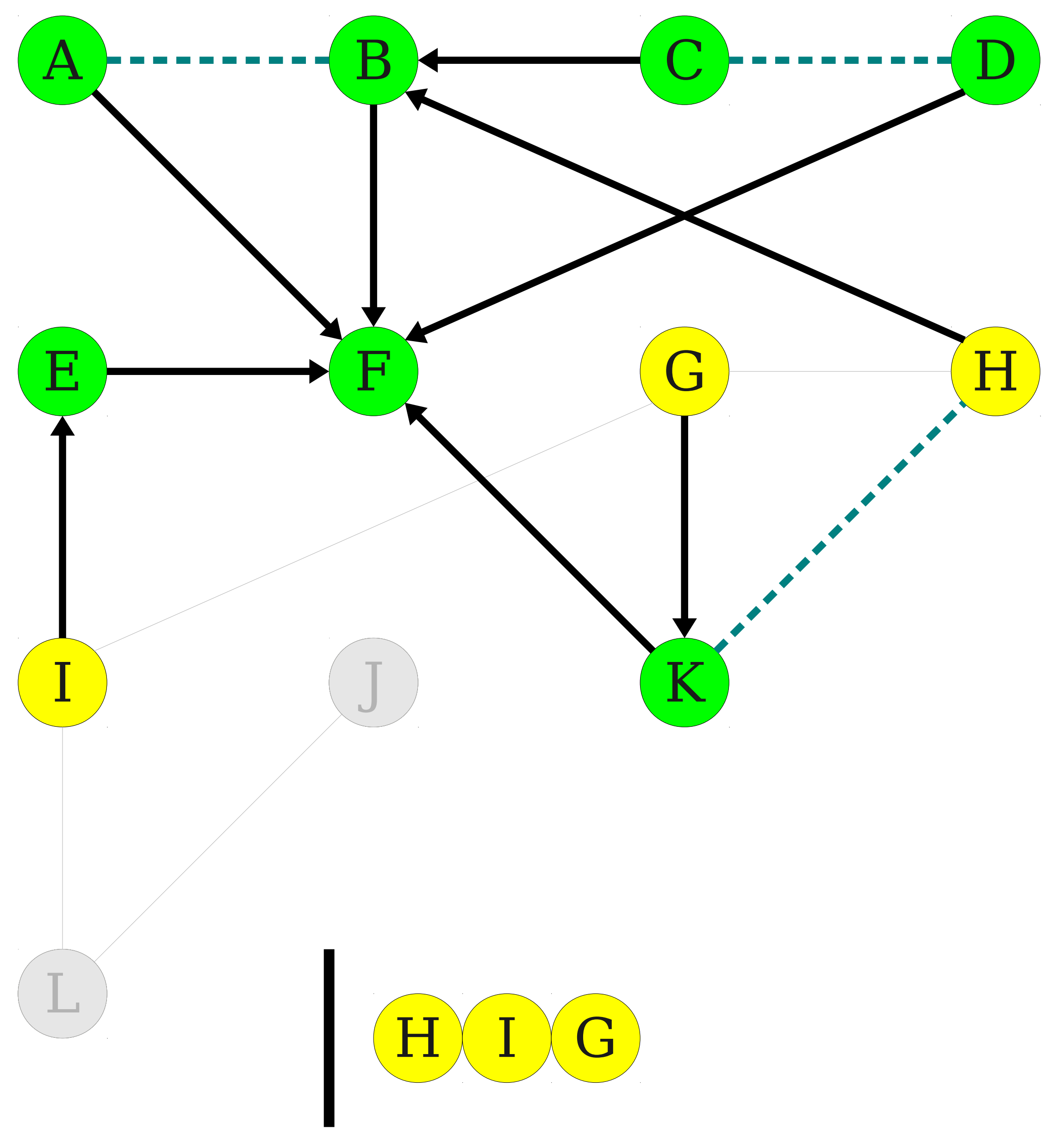
Breadth-First Search



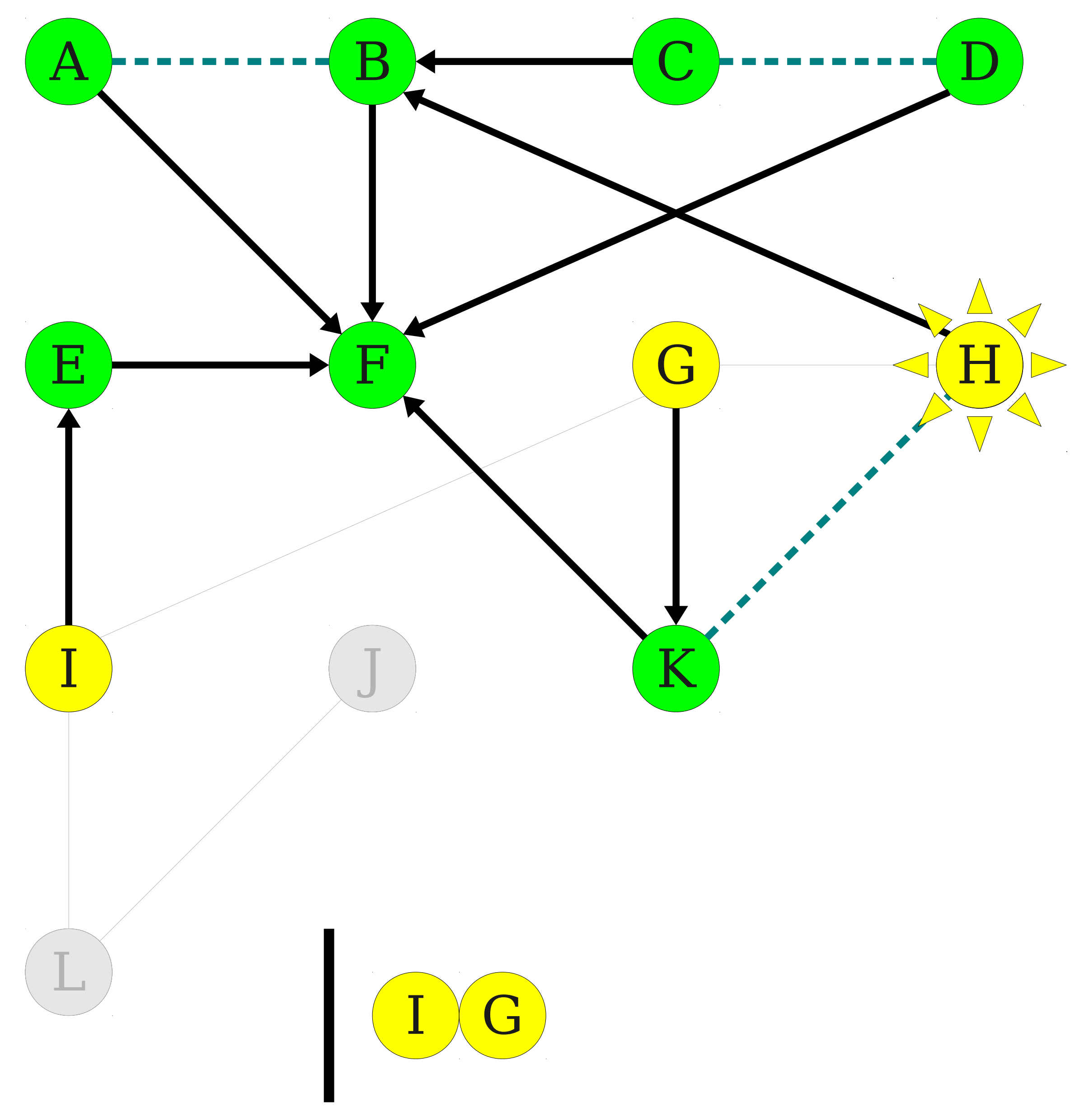
Breadth-First Search



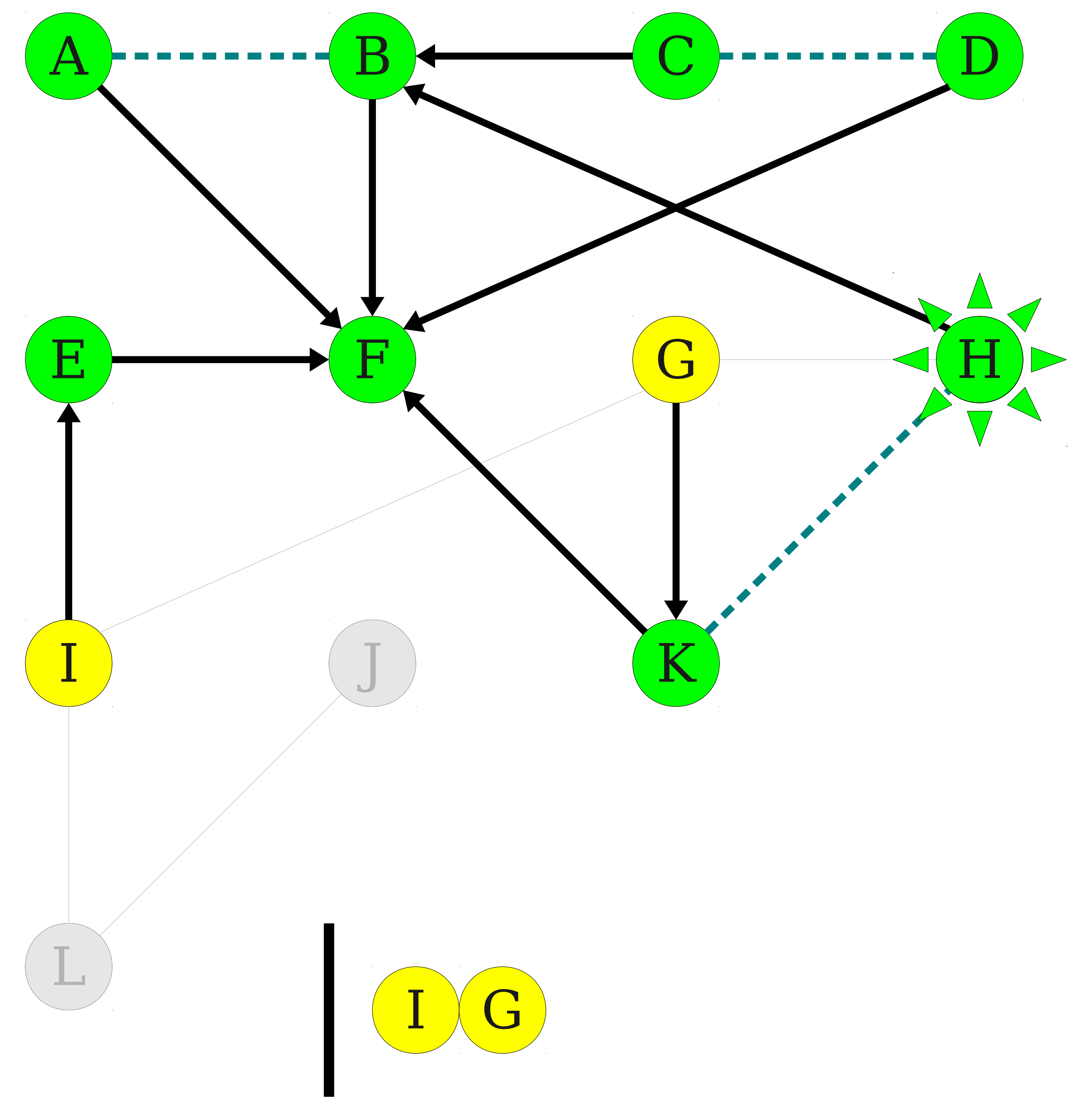
Breadth-First Search



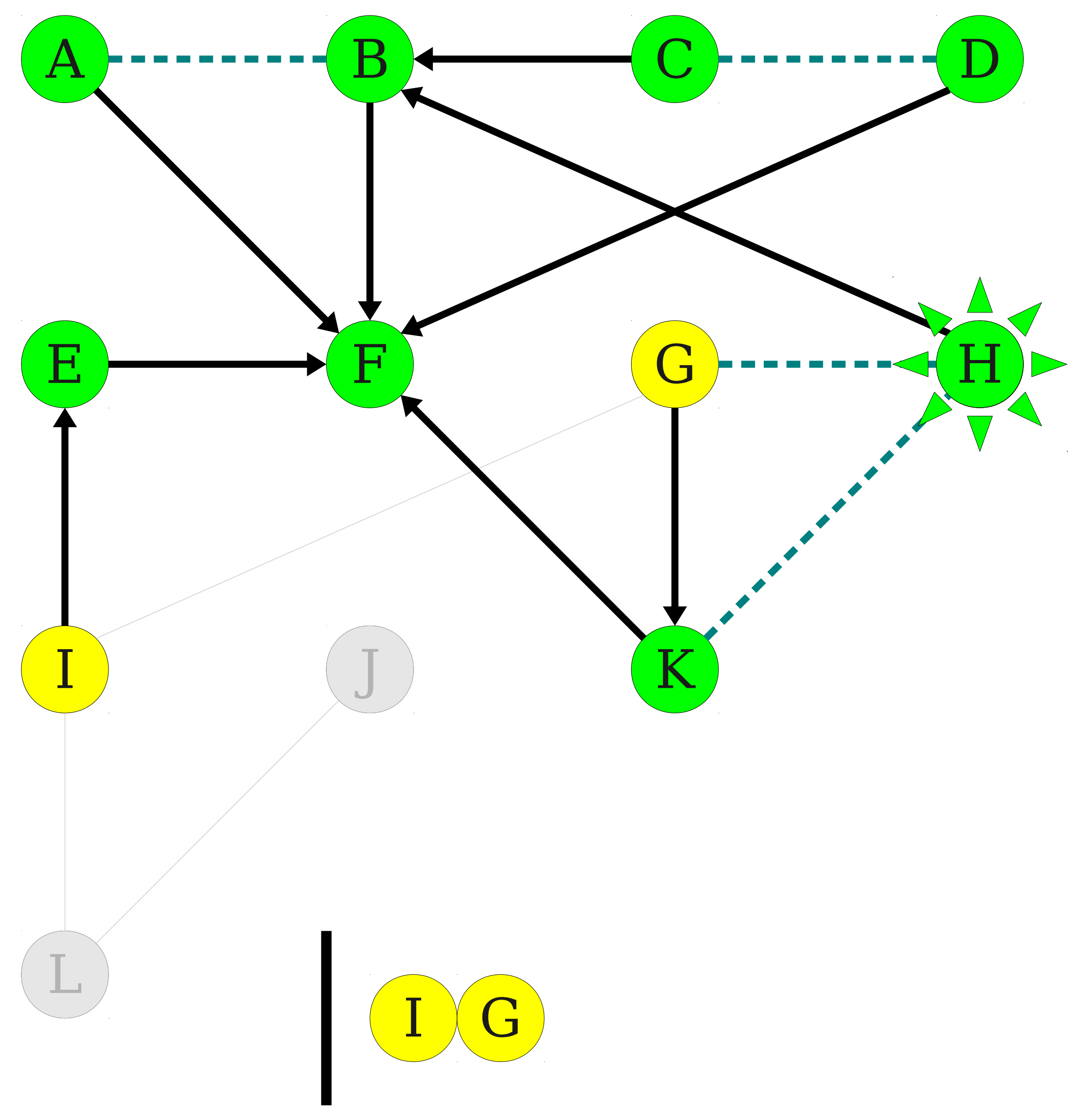
Breadth-First Search



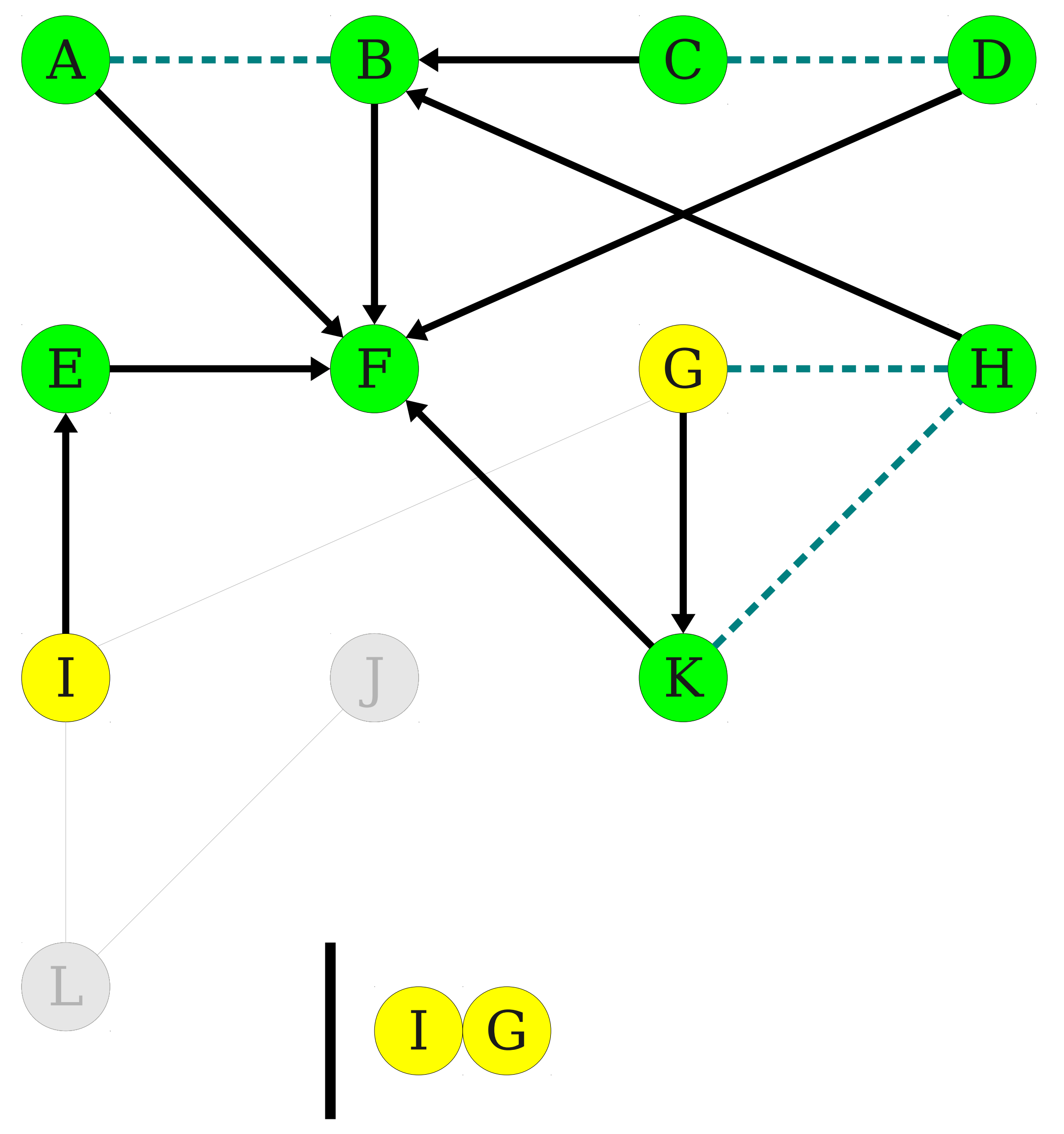
Breadth-First Search



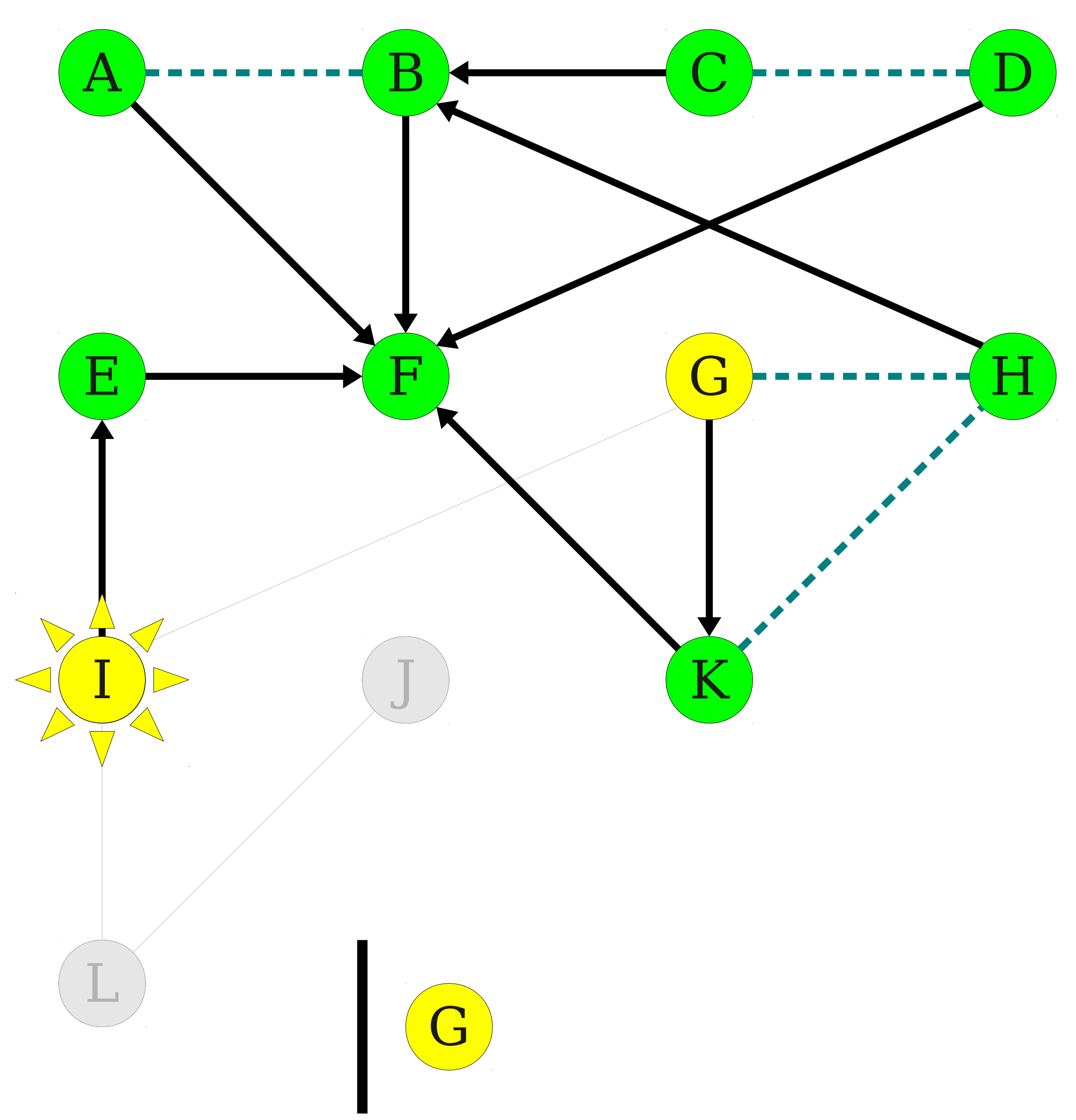
Breadth-First Search



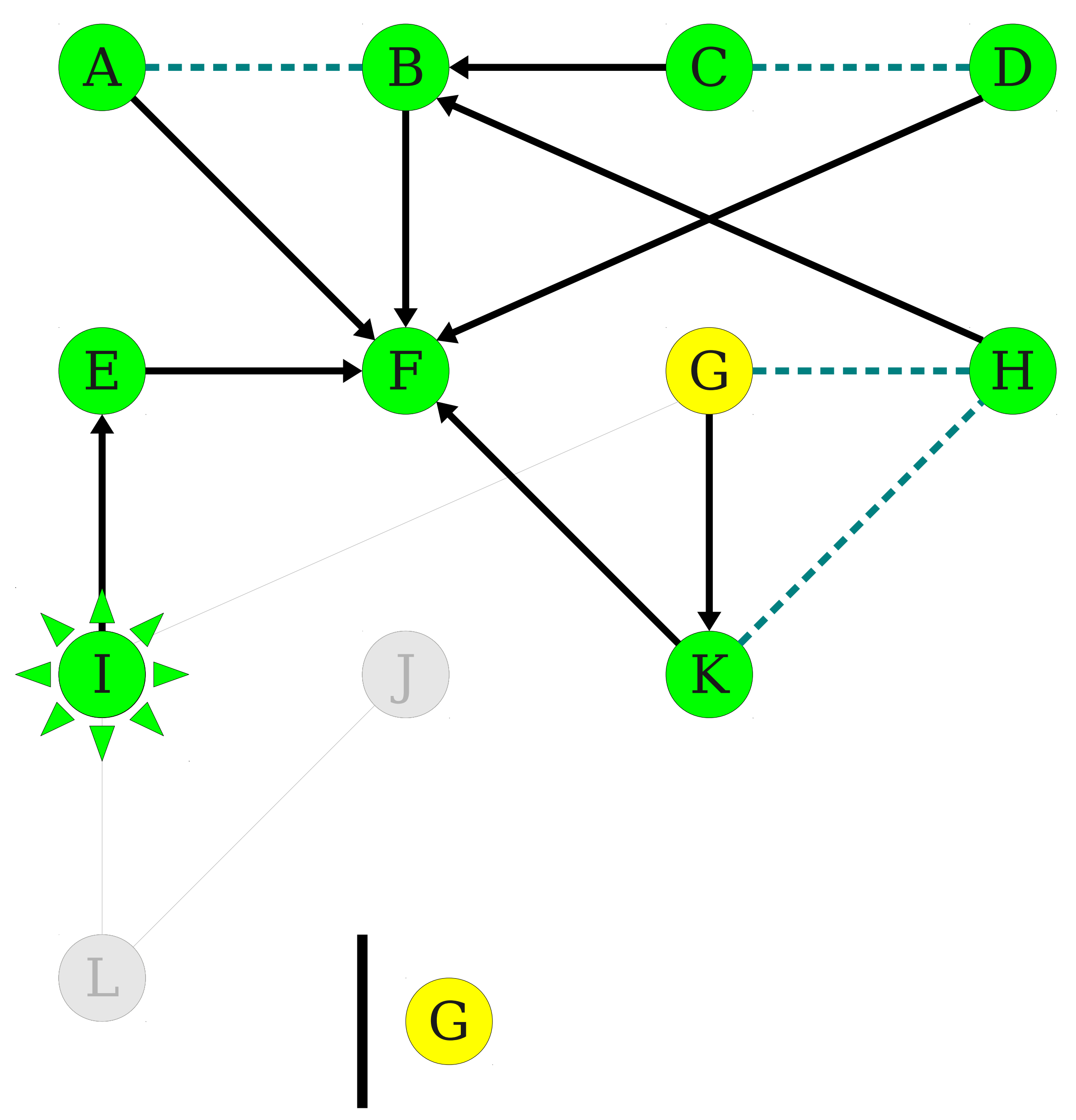
Breadth-First Search



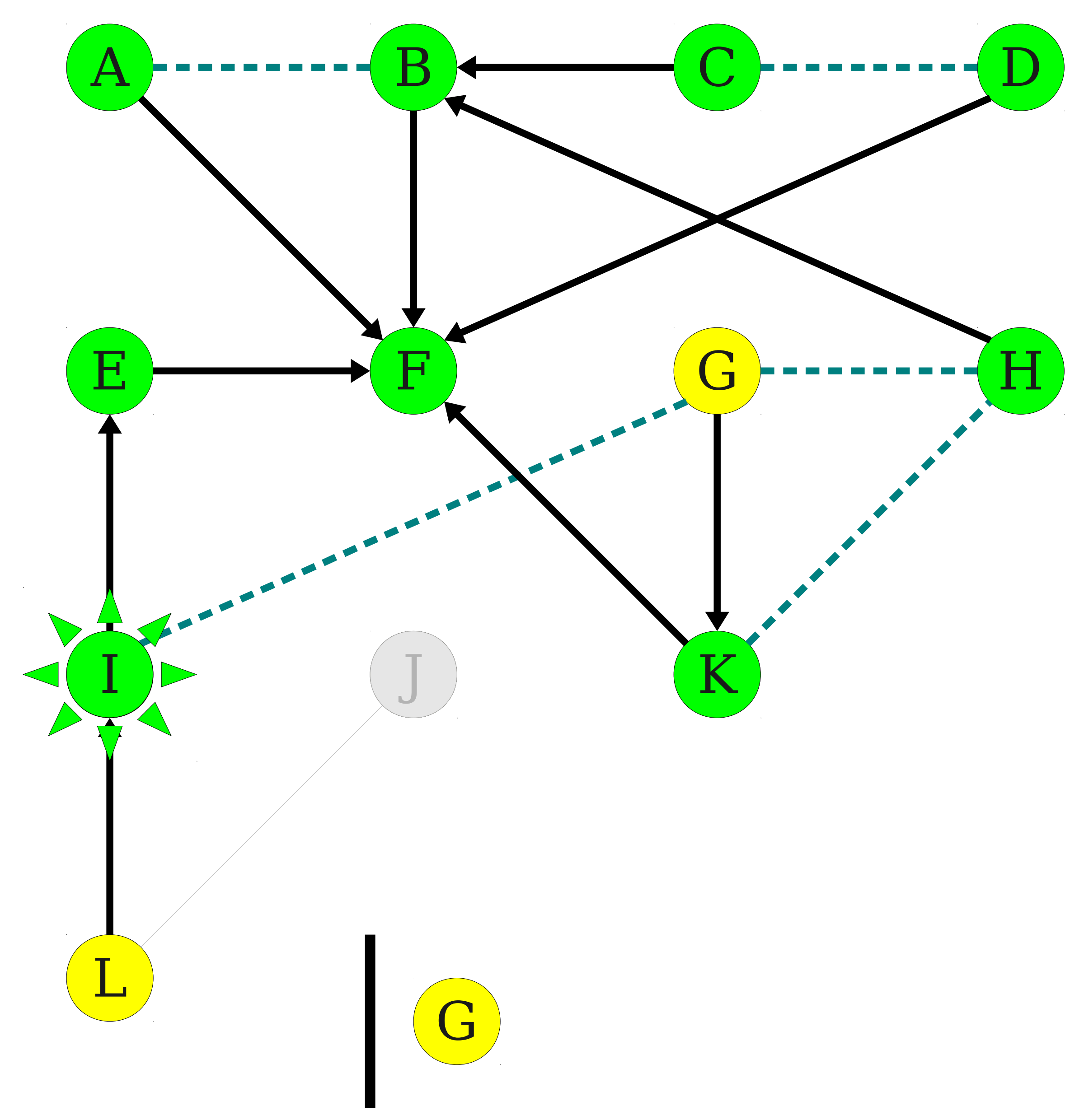
Breadth-First Search



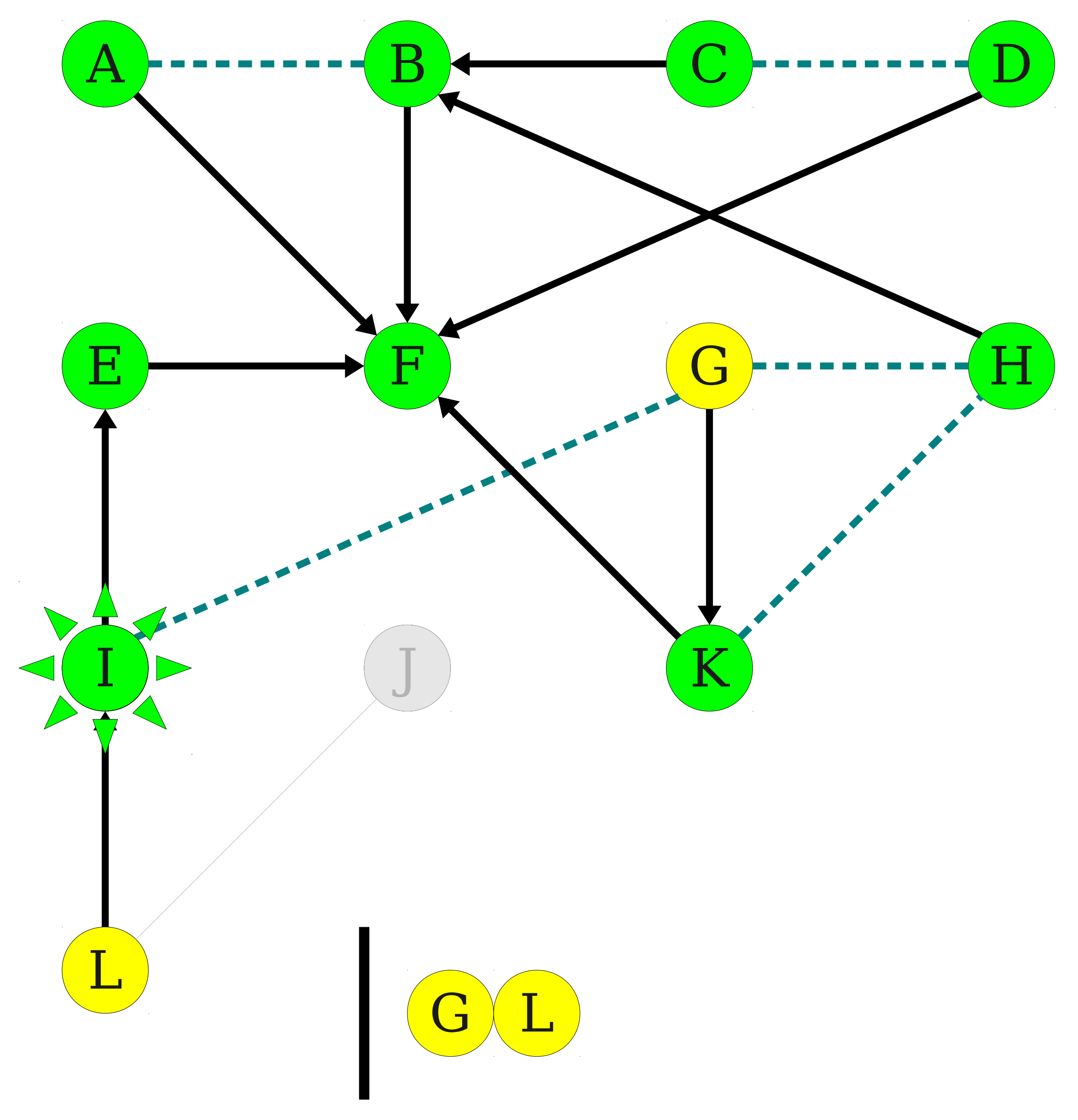
Breadth-First Search



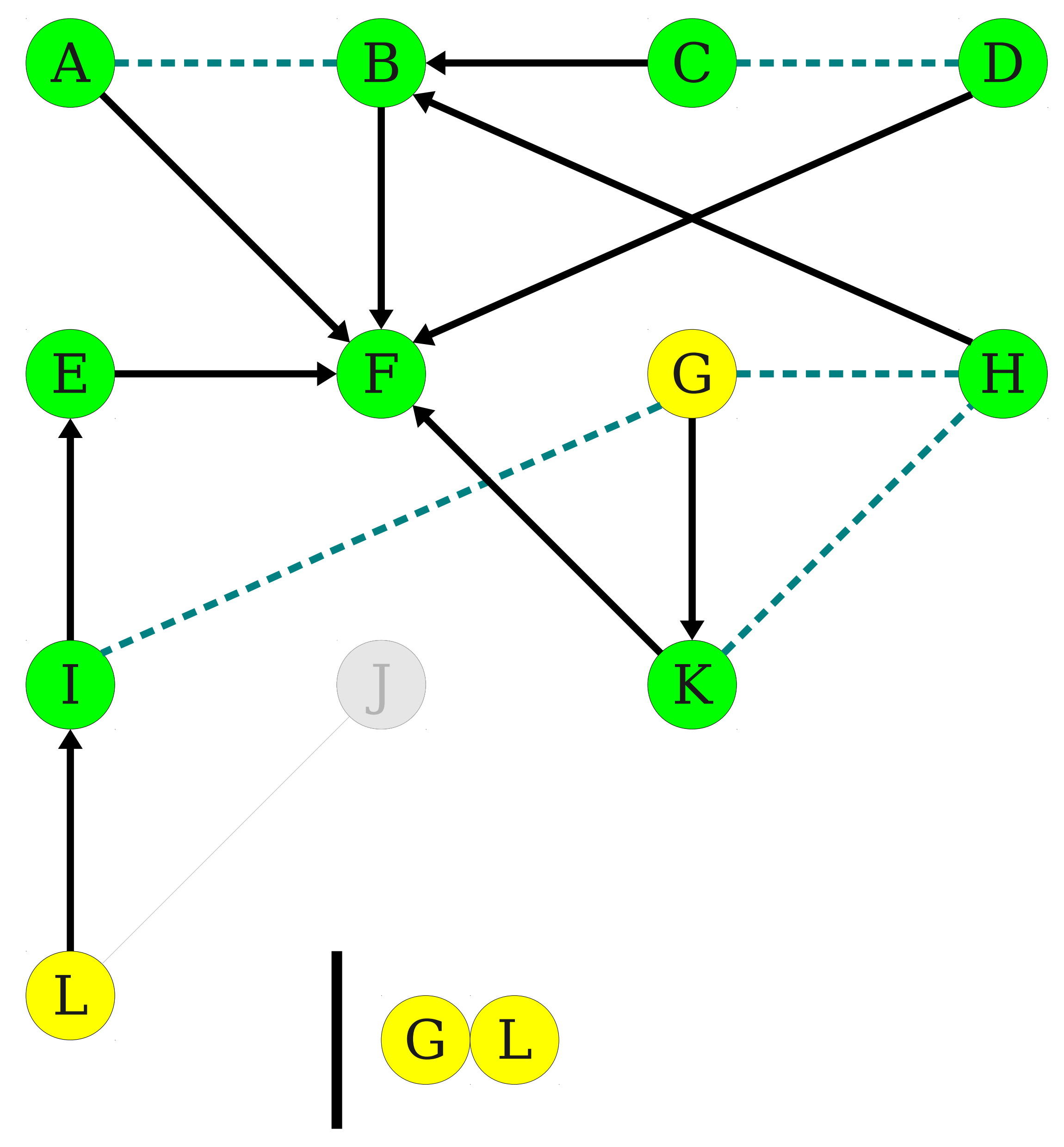
Breadth-First Search



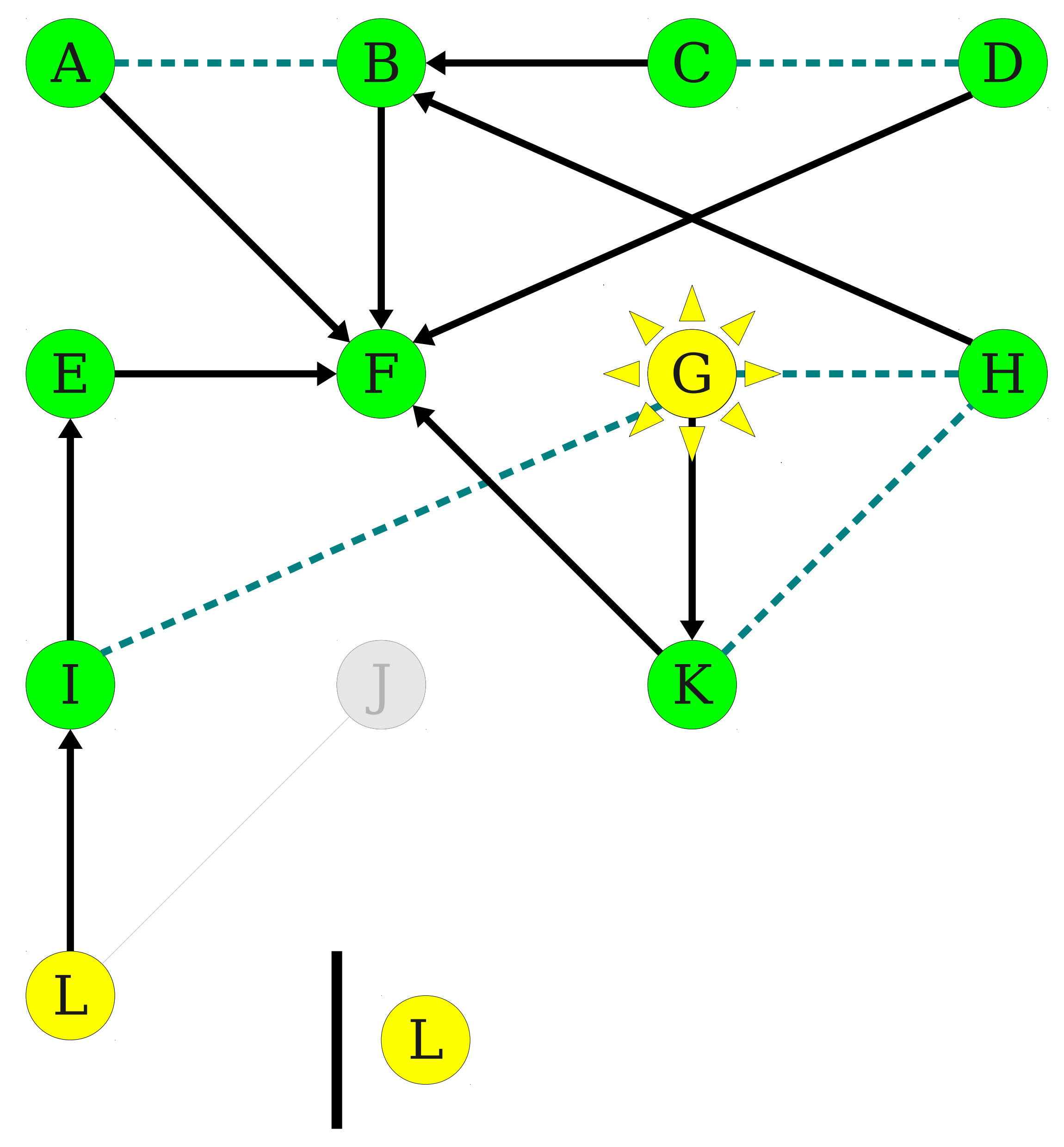
Breadth-First Search



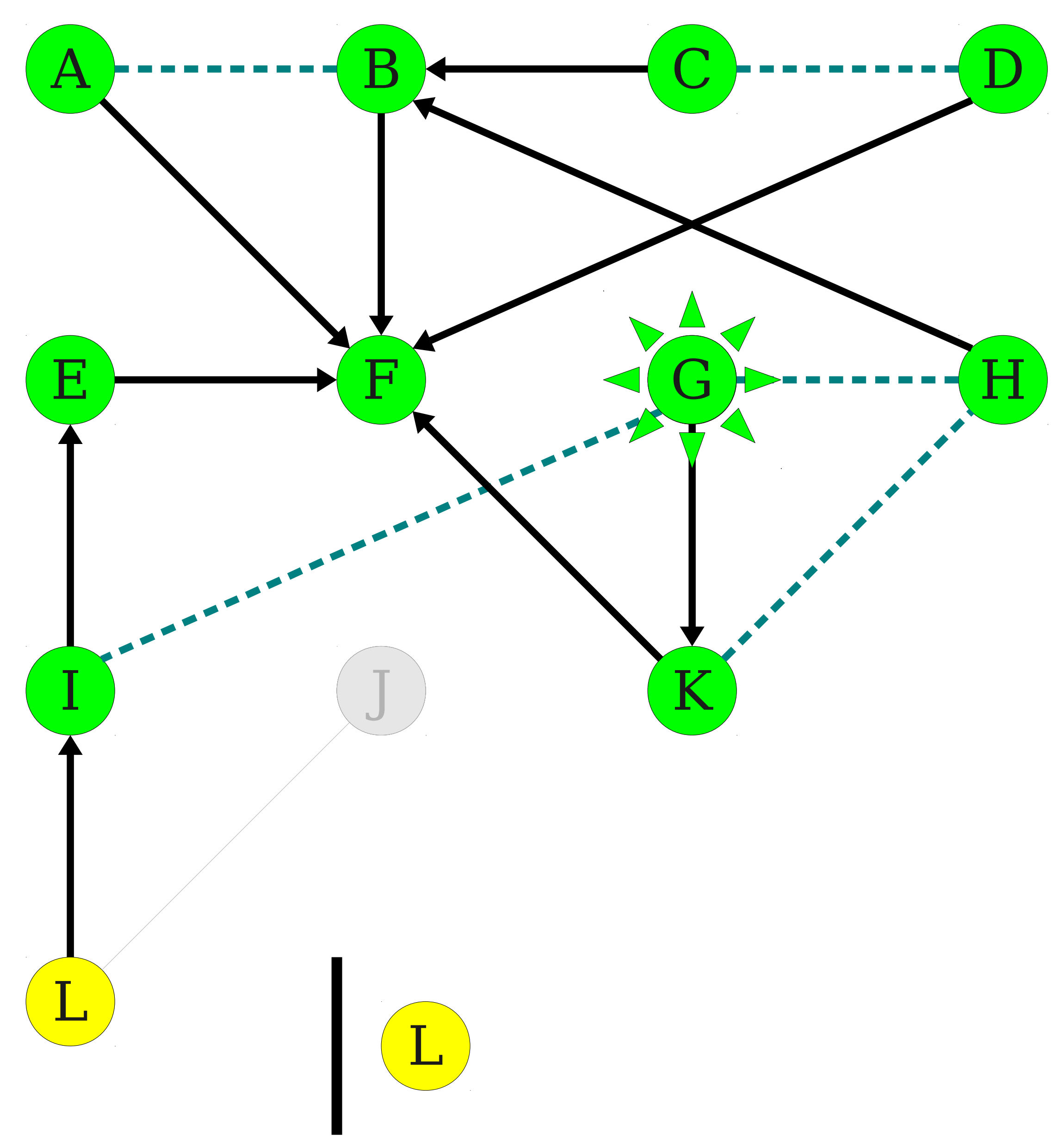
Breadth-First Search



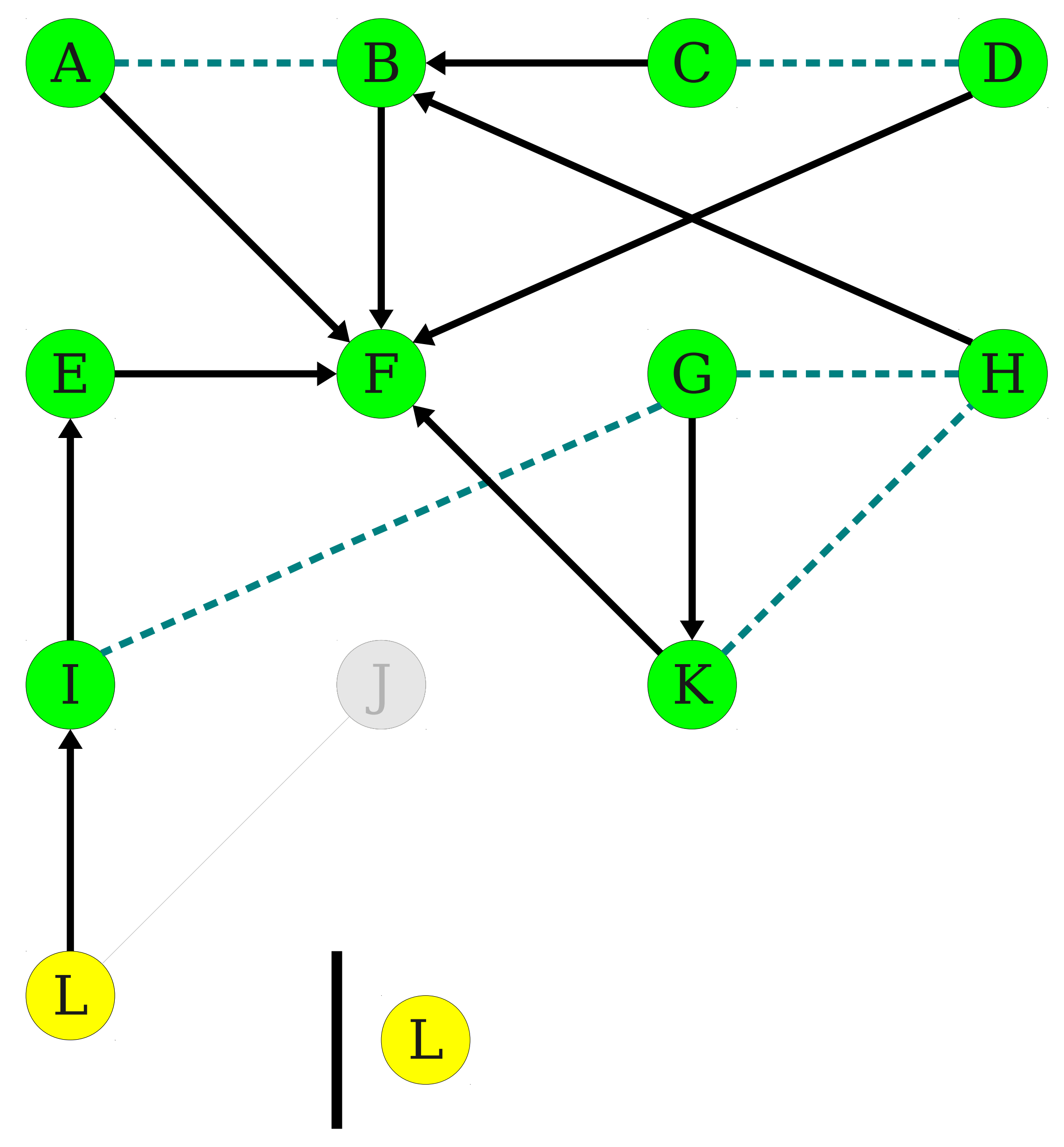
Breadth-First Search



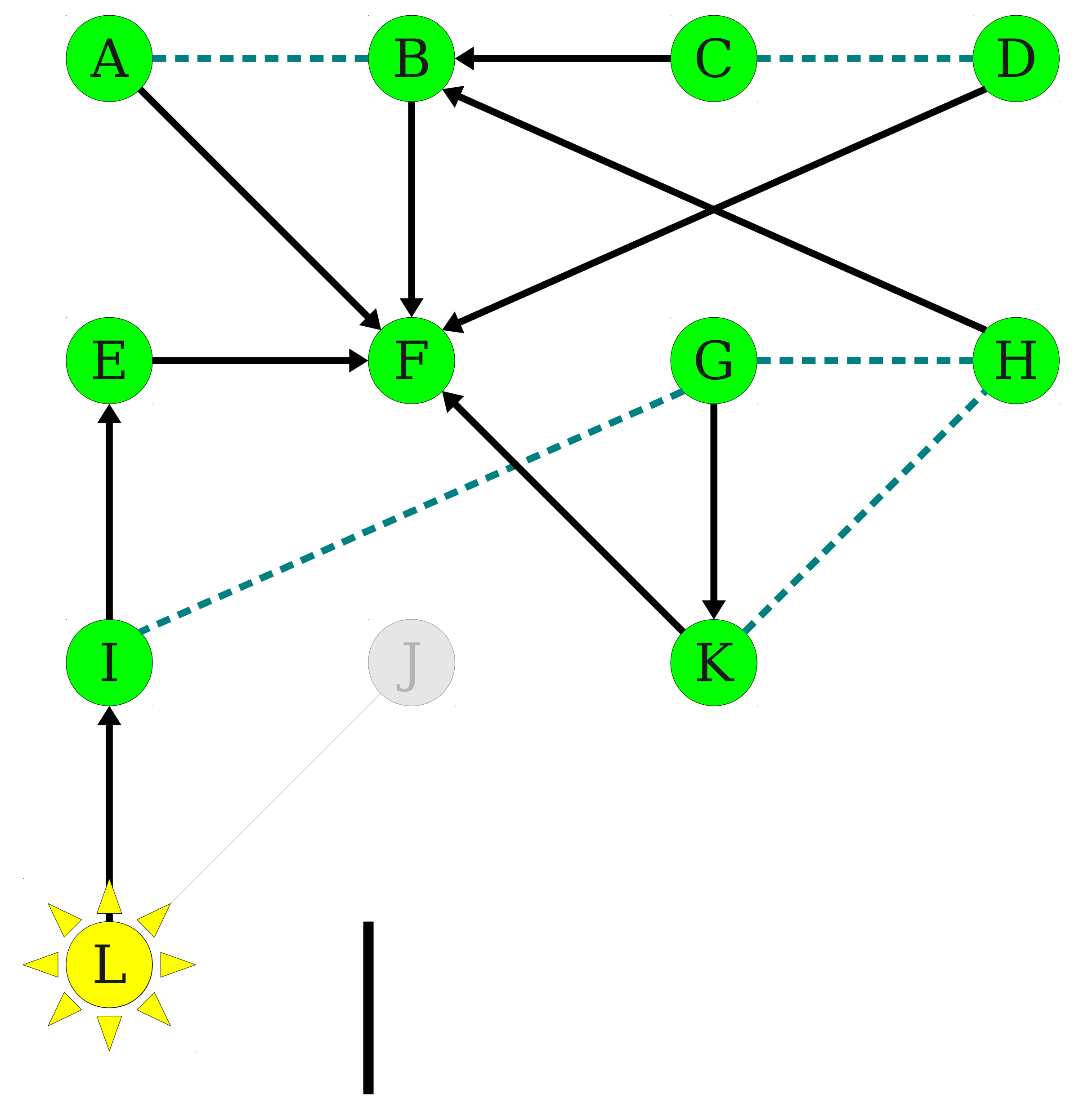
Breadth-First Search



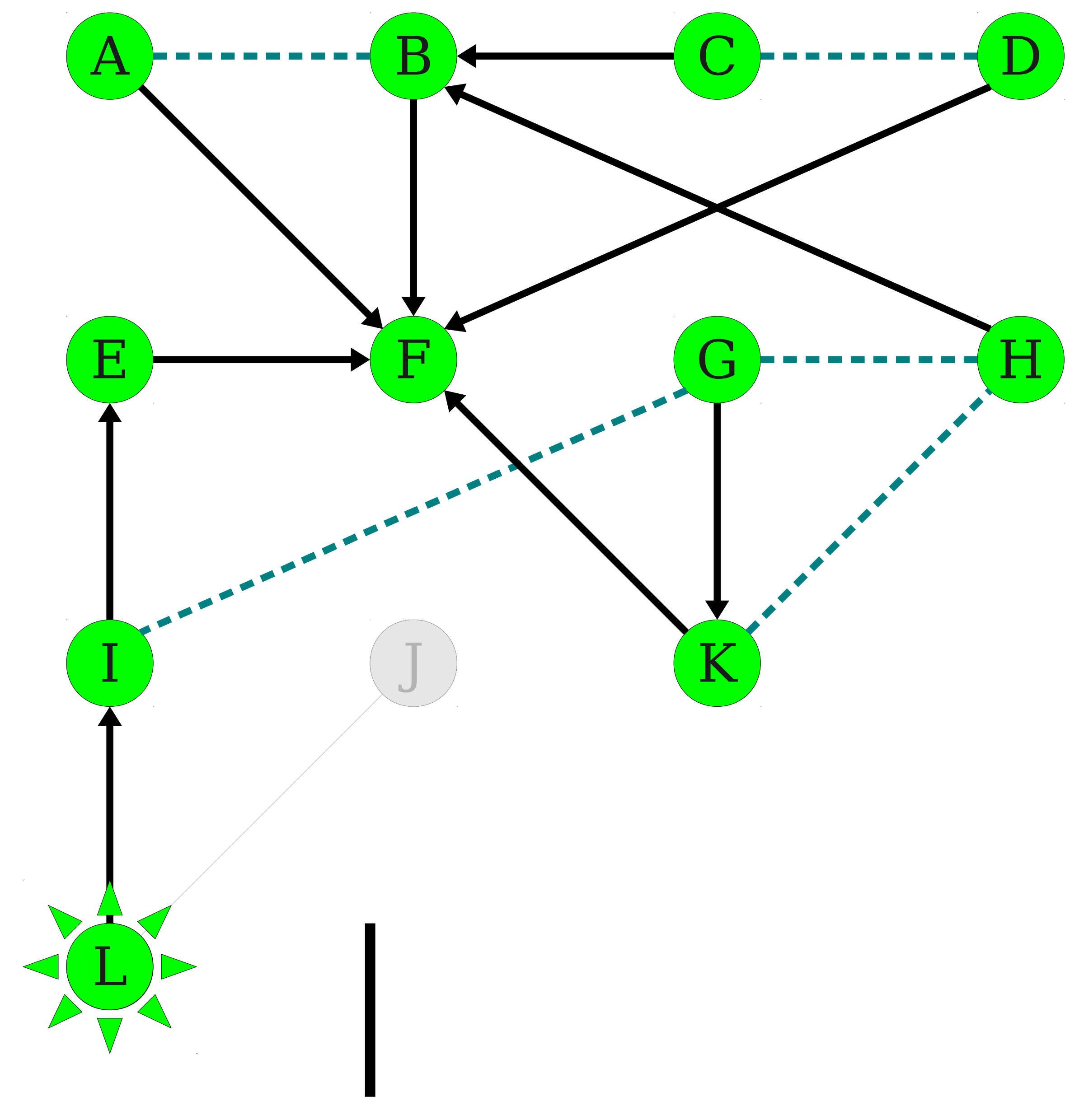
Breadth-First Search



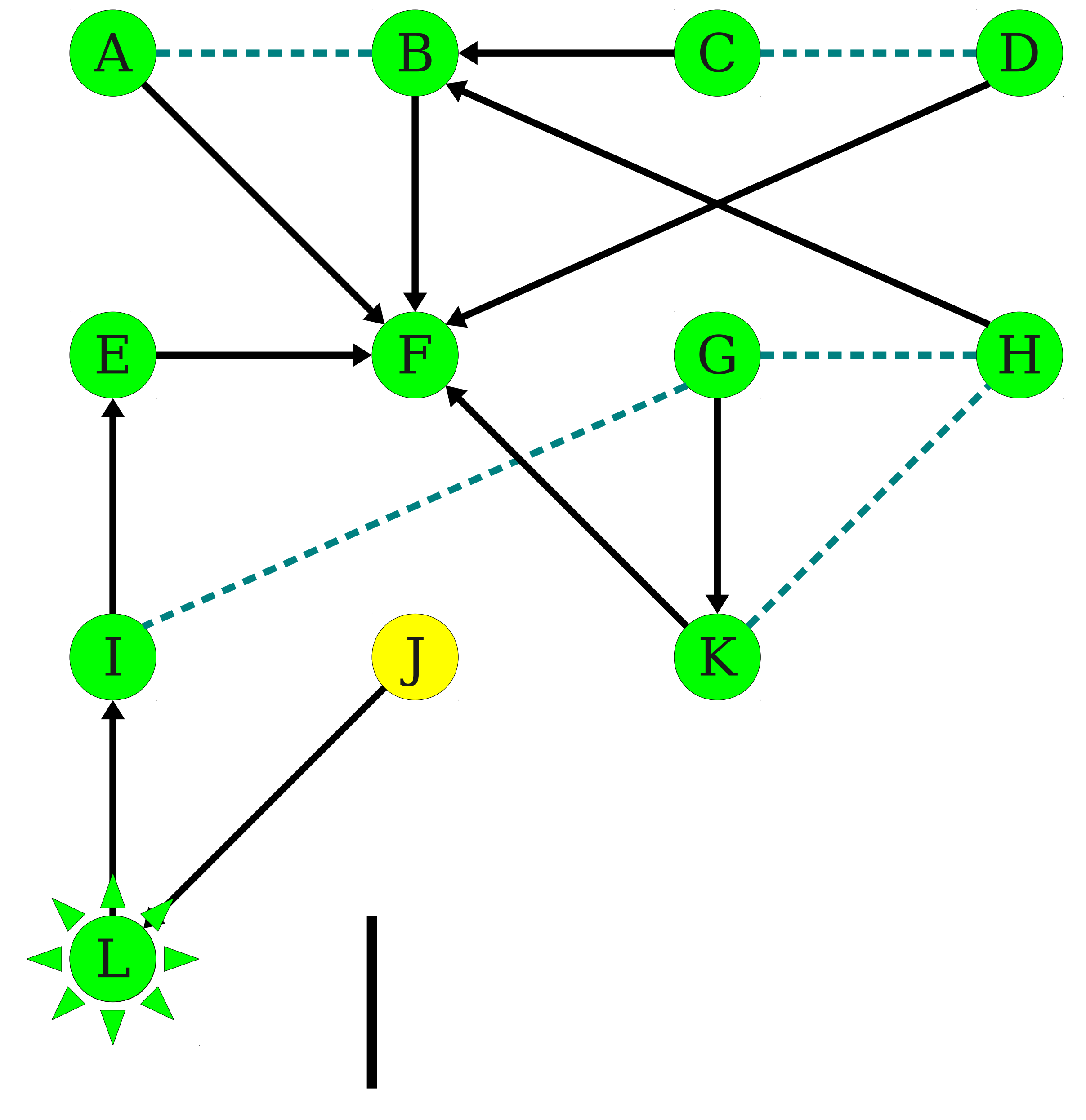
Breadth-First Search



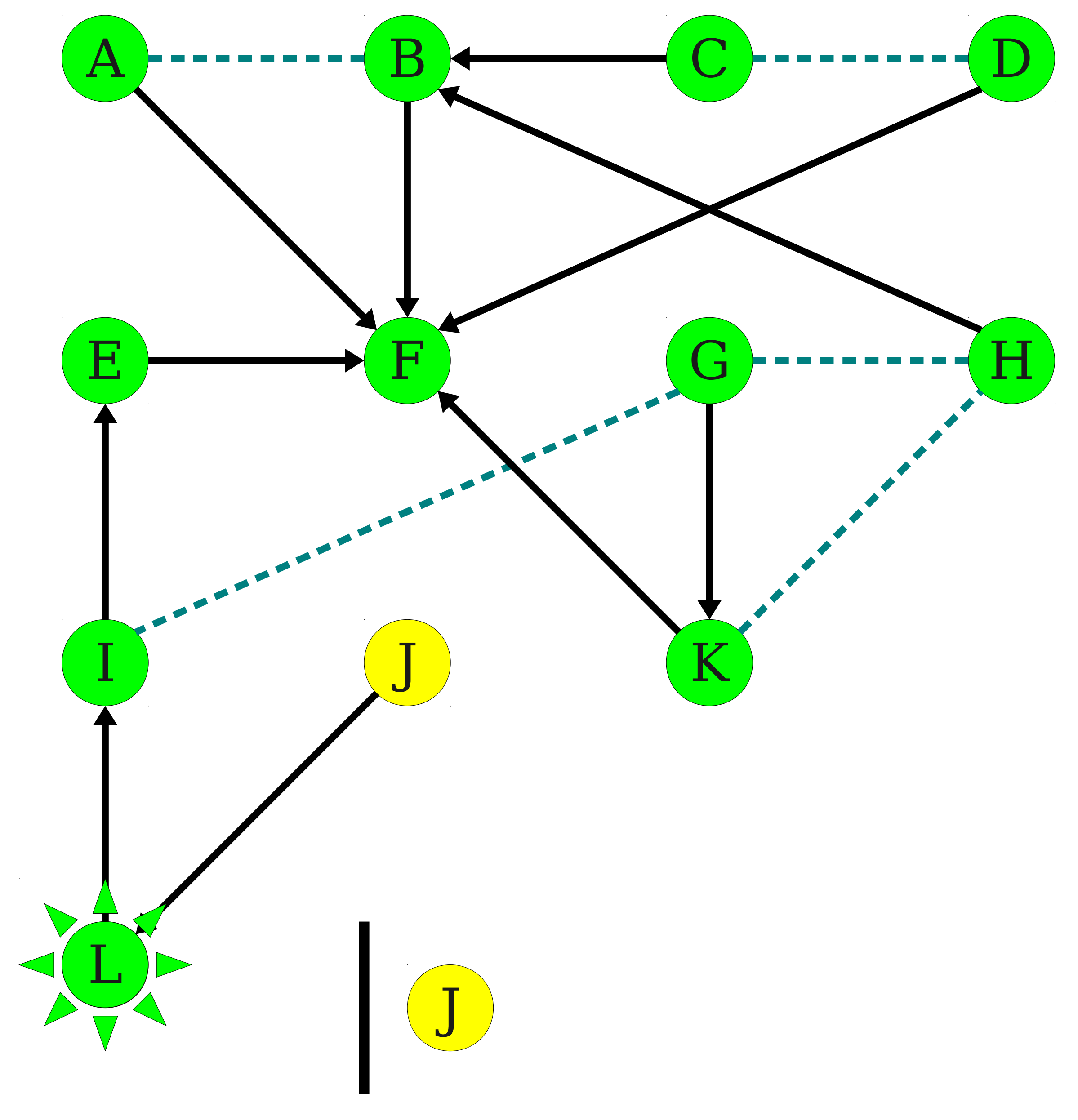
Breadth-First Search



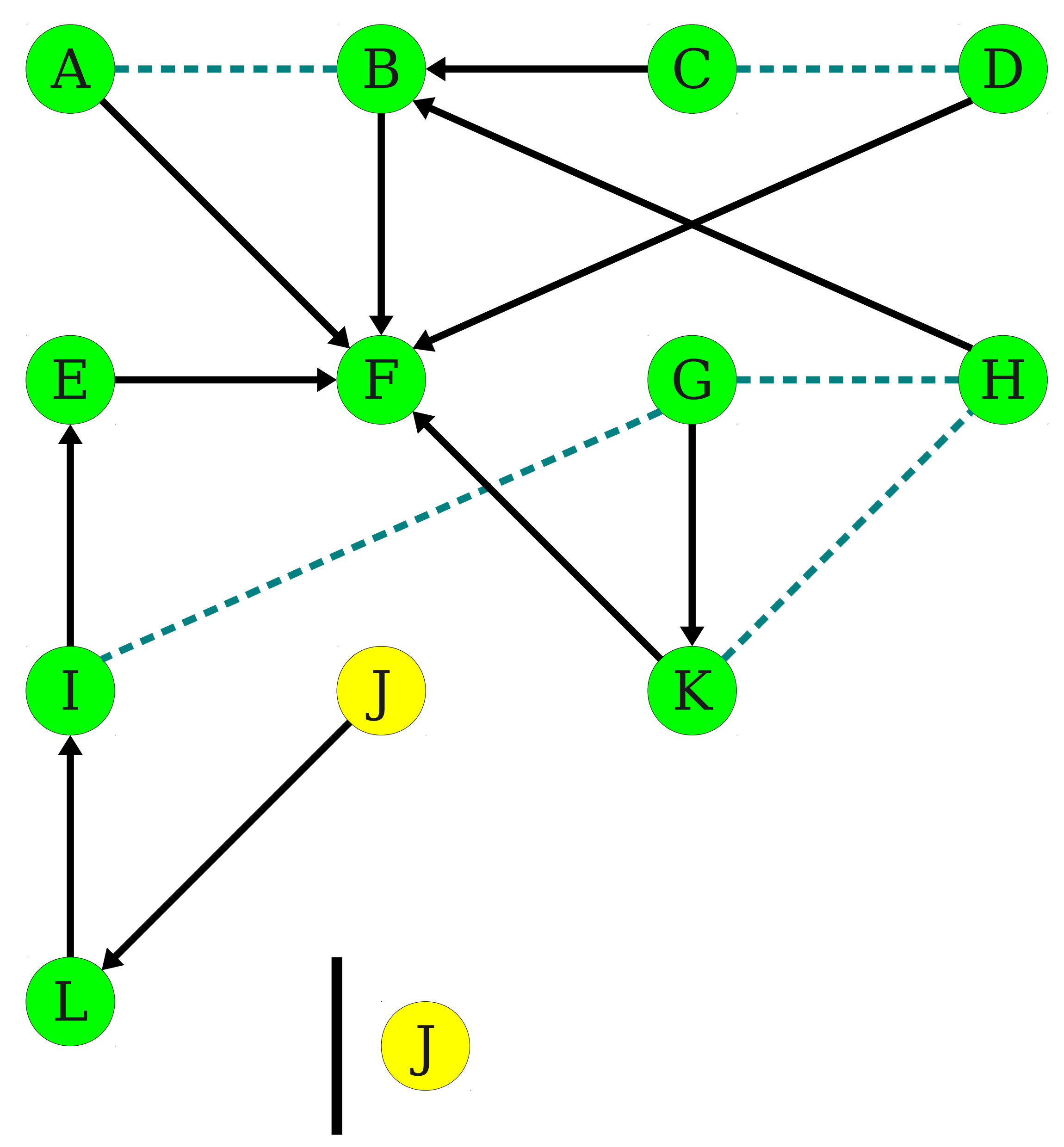
Breadth-First Search



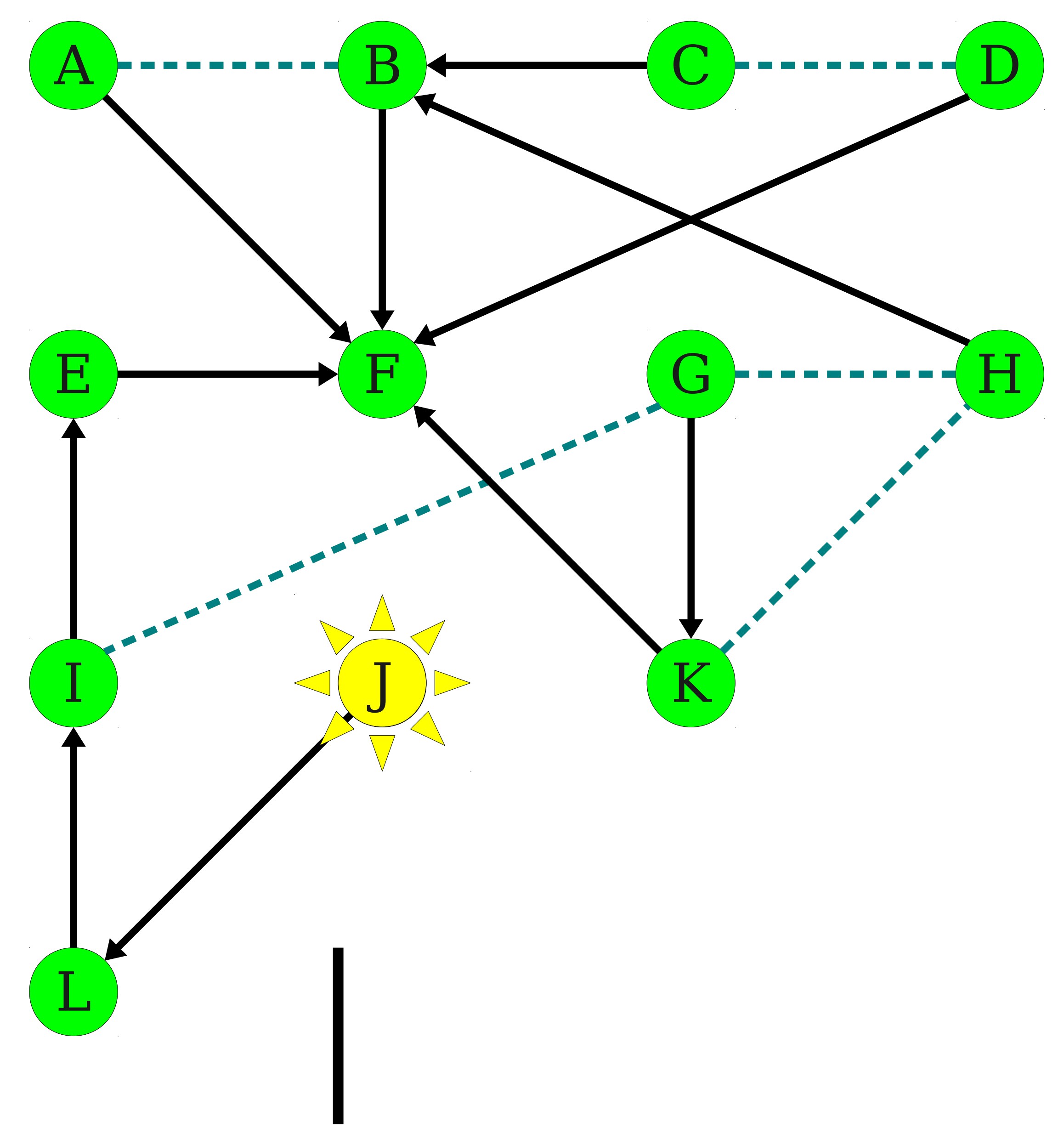
Breadth-First Search



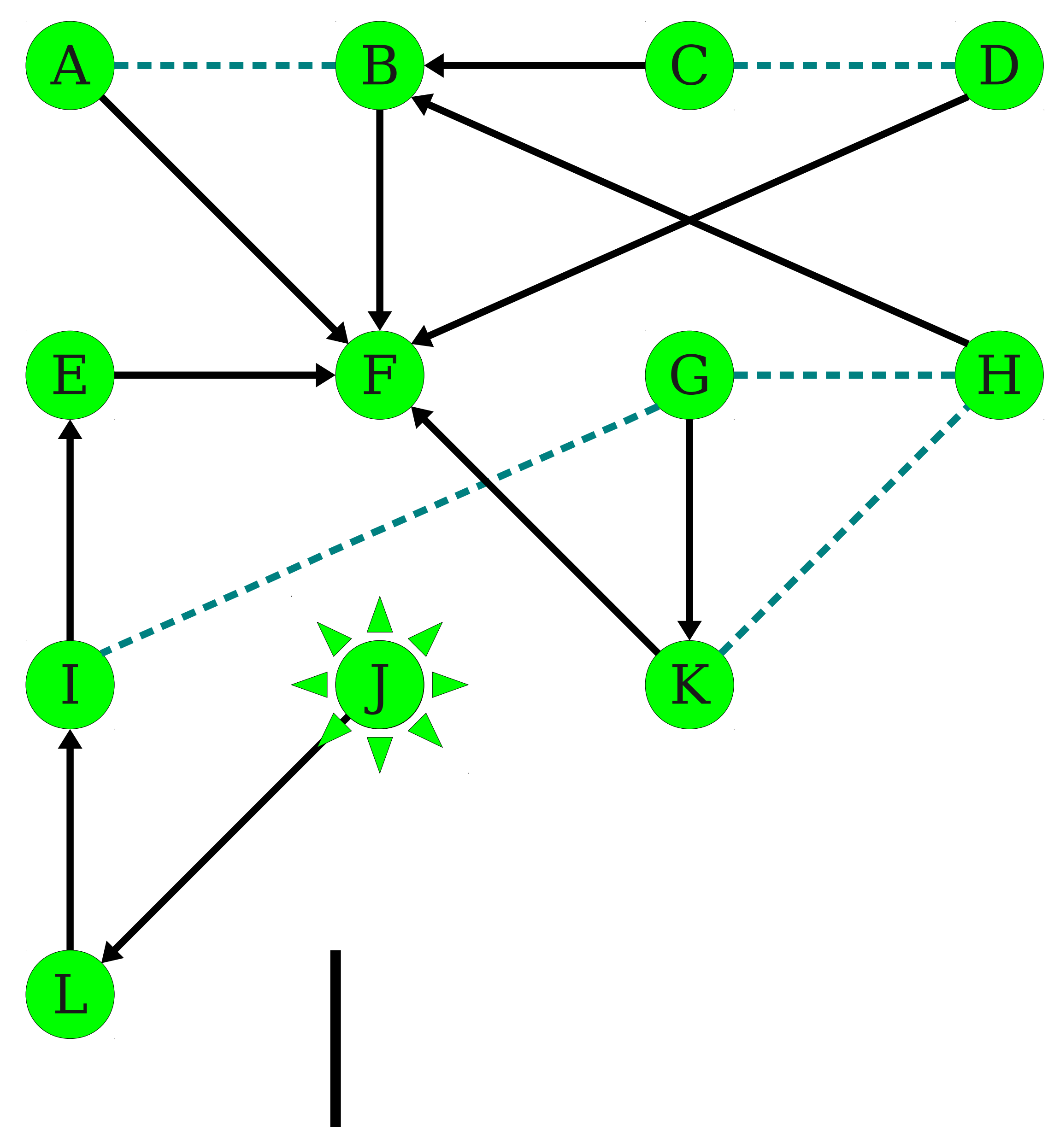
Breadth-First Search



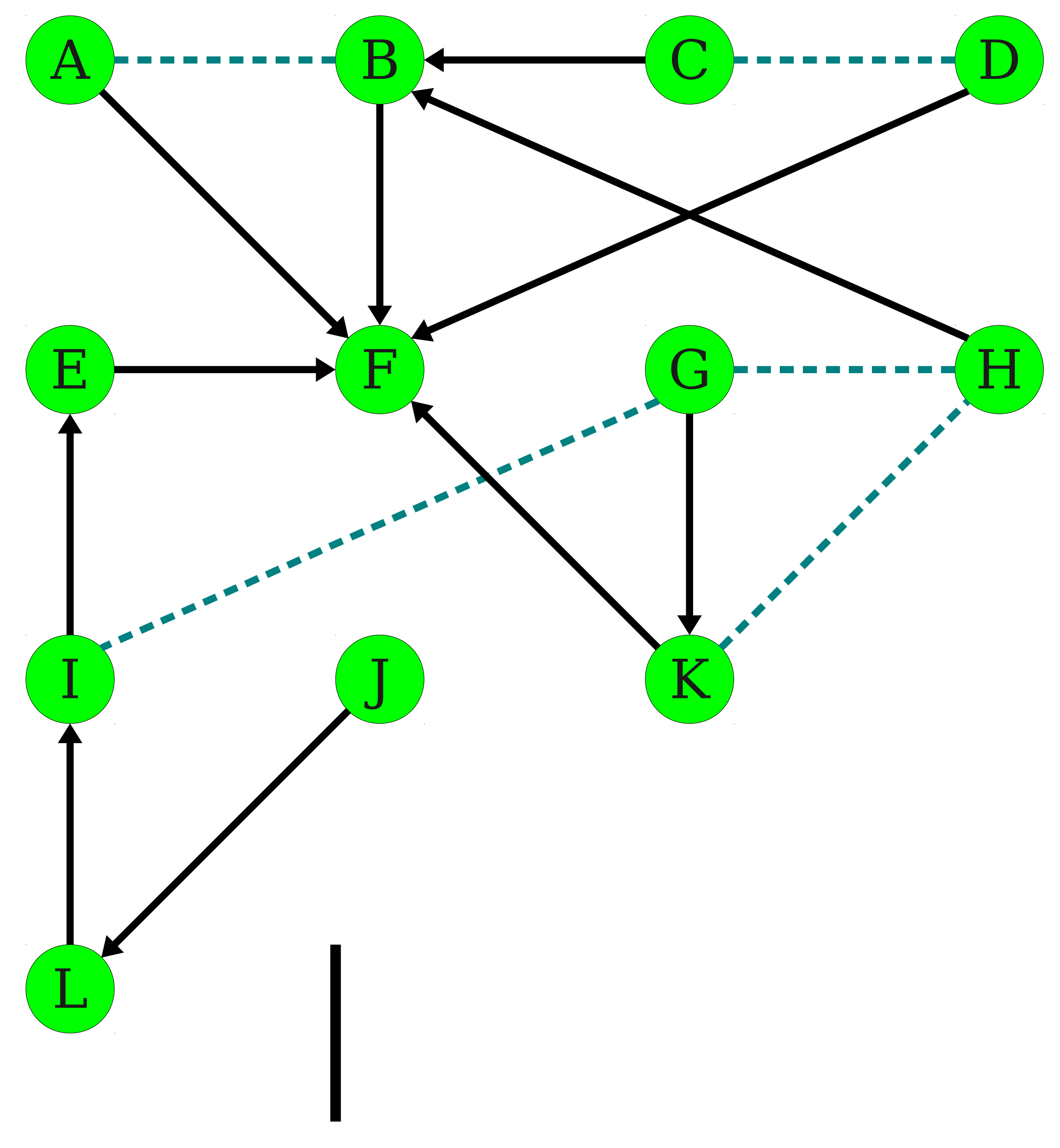
Breadth-First Search



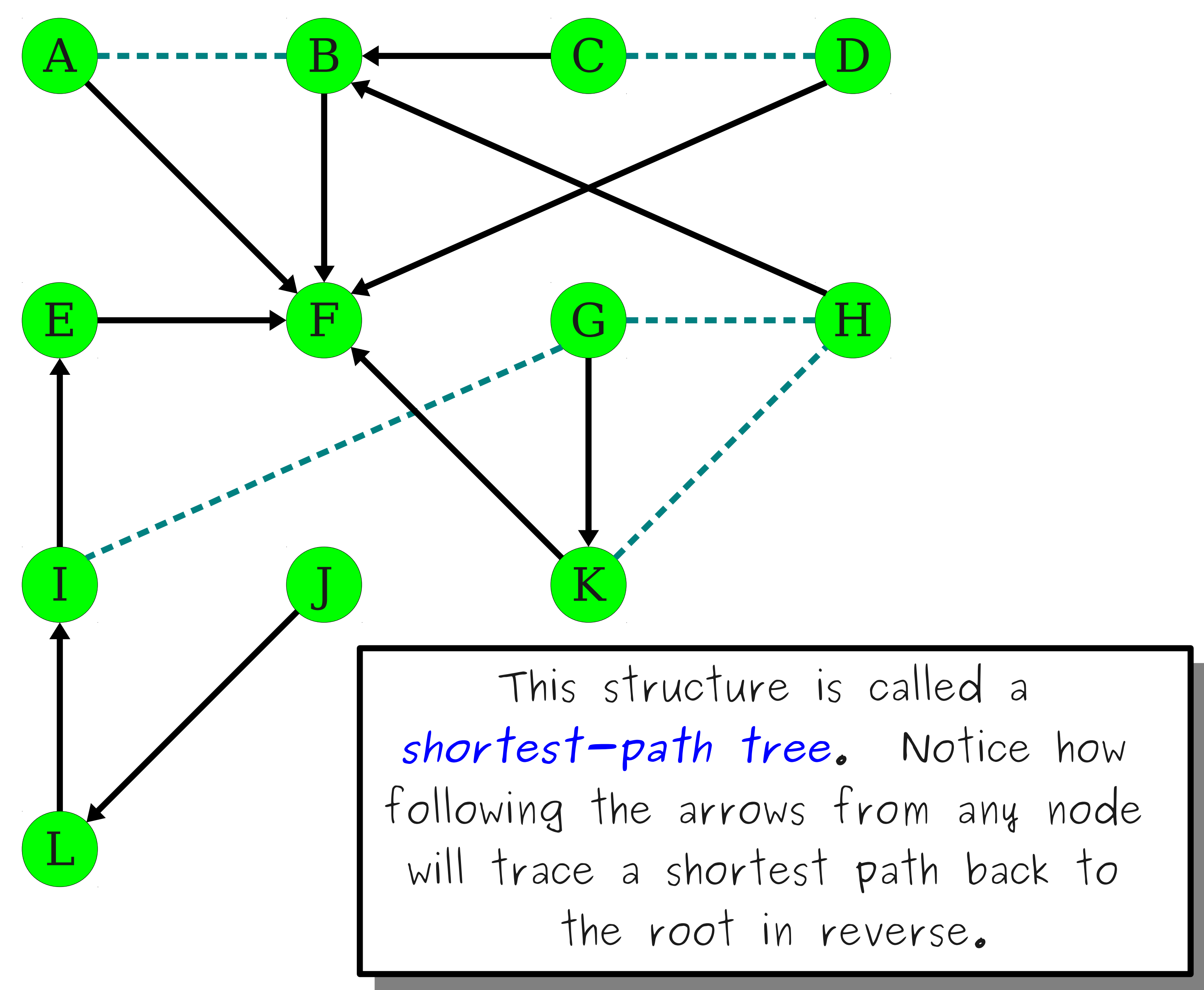
Breadth-First Search

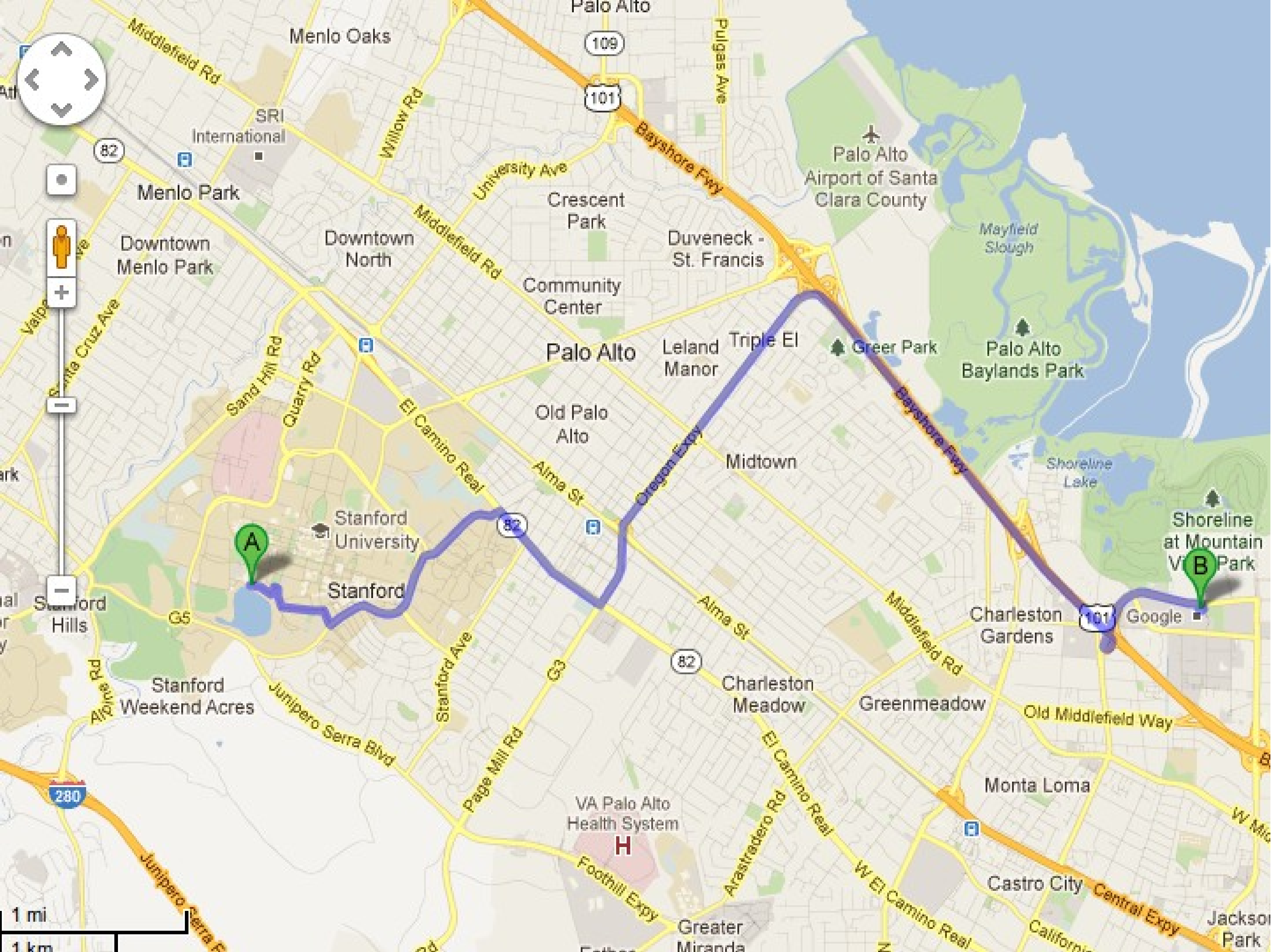


Breadth-First Search



Breadth-First Search

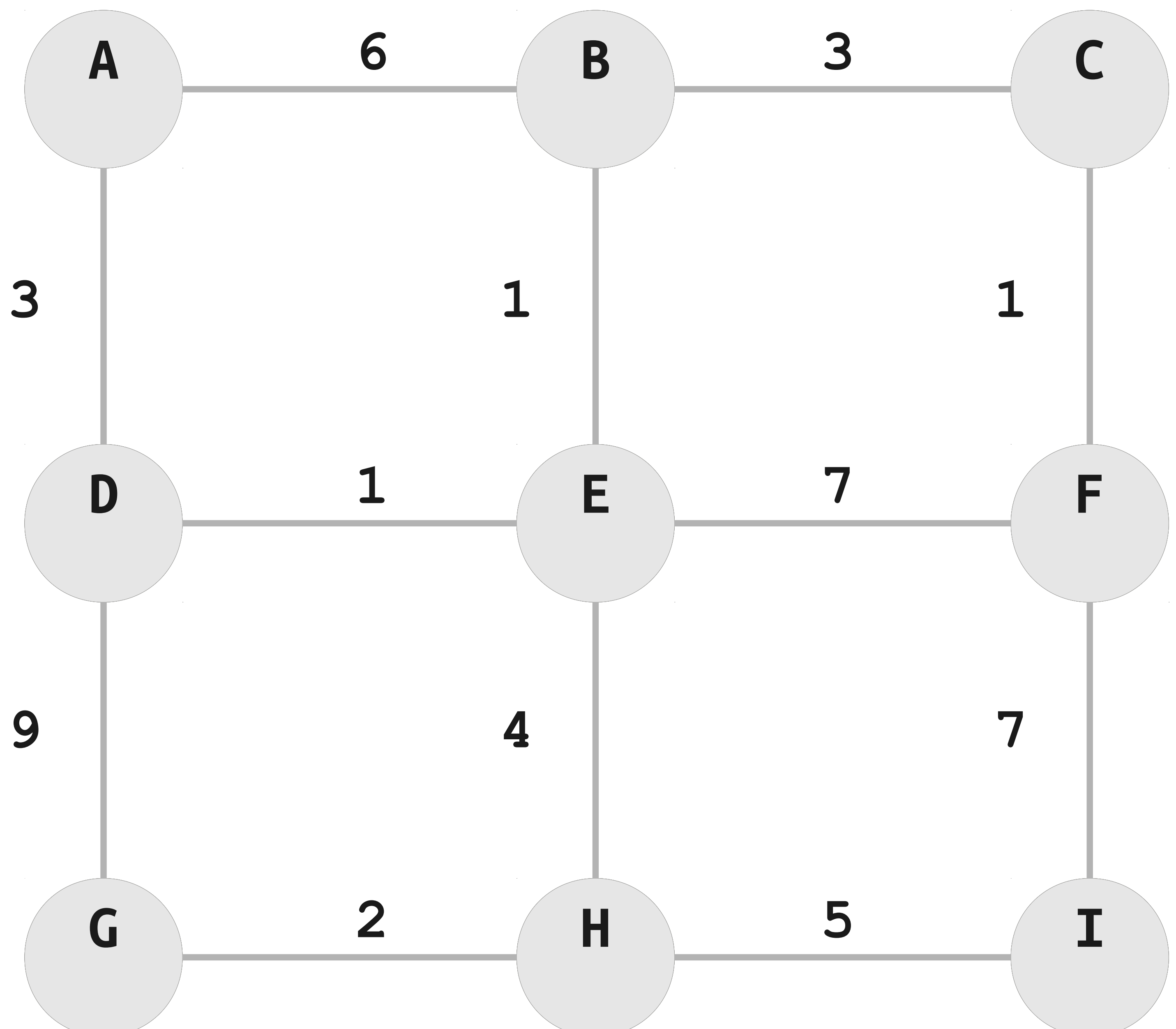


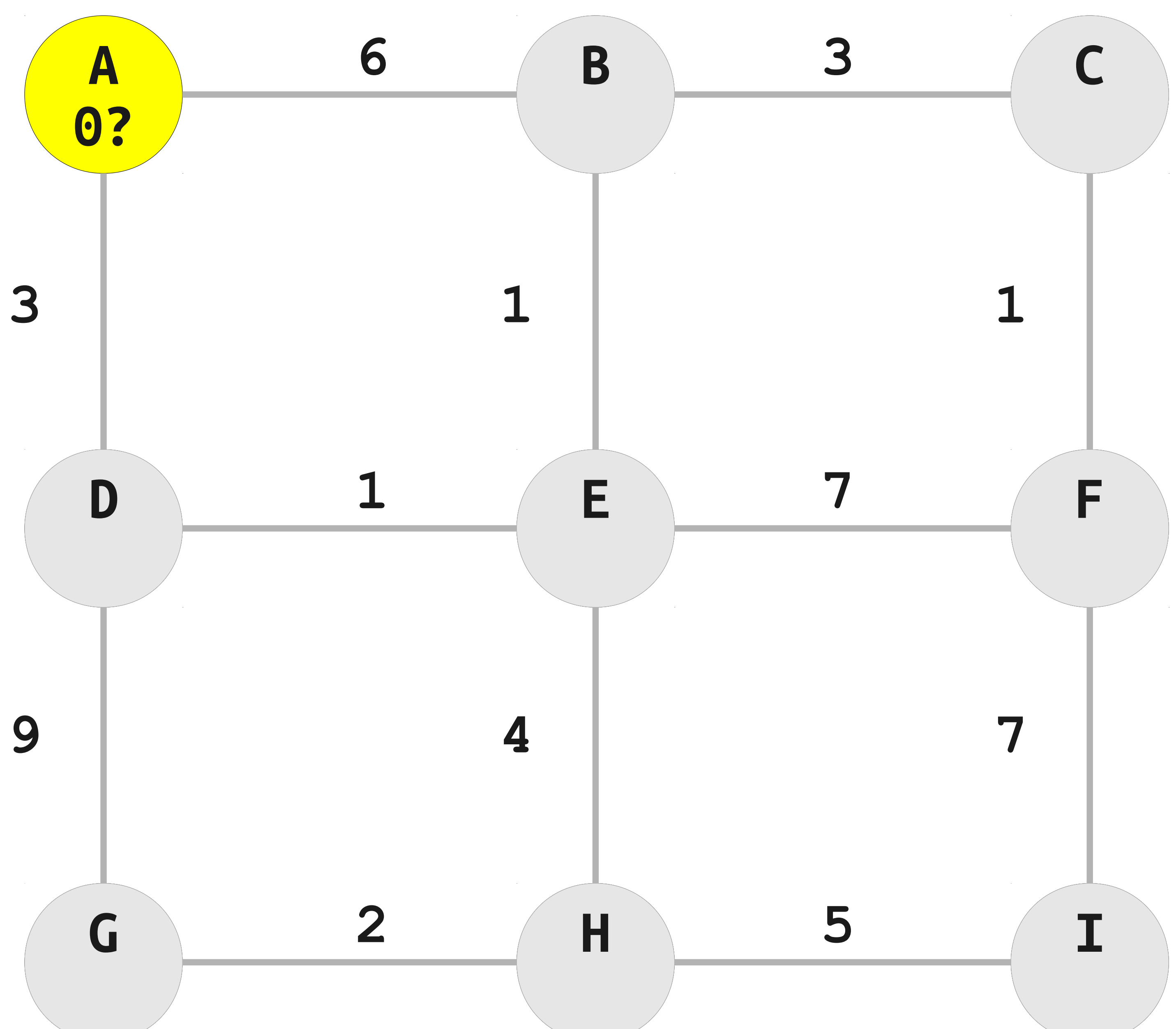


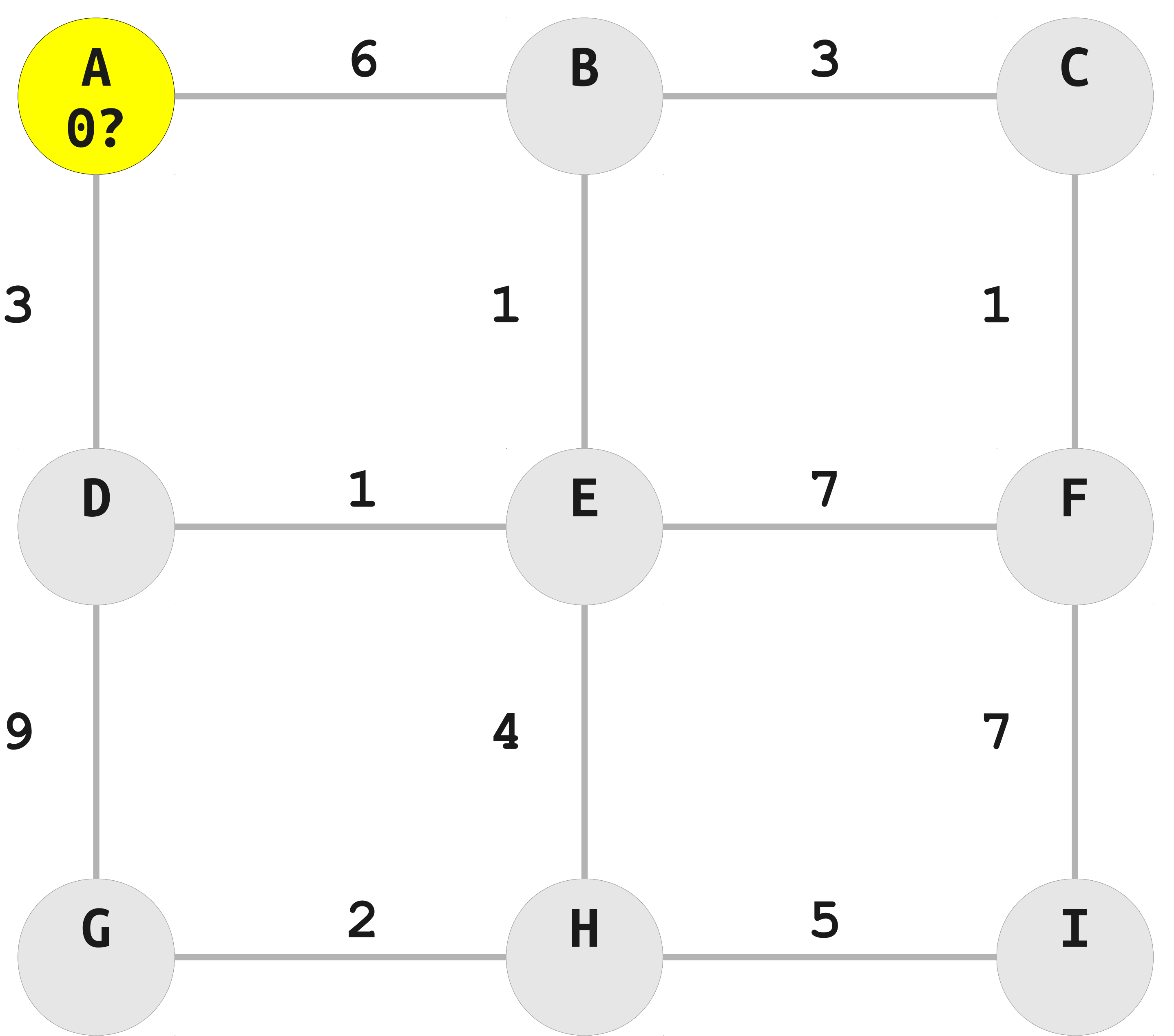
Palo Alto
Menlo Oaks
Middlefield Rd
109
101
Pulgas Ave
SRI
International
Willow Rd
82
Bayshore Fwy
Palo Alto
Airport of Santa
Clara County
Menlo Park
University Ave
Crescent
Park
Mayfield
Slough
Downtown
Menlo Park
Downtown
North
Middlefield Rd
Duveneck -
St. Francis
Community
Center
Santa Cruz Ave
Sand Hill Rd
Quarry Rd
Palo Alto
Leland
Manor
Triple El
Greer Park
Palo Alto
Baylands Park
El Camino Real
Old Palo
Alto
Oregon Expy
Bayshore Fwy
Midtown
Shoreline
Lake
Alma St
Stanford
University
Shoreline
at Mountain
Park
A
B
Stanford
Stanford
Hills
G5
Alma St
Charleston
Gardens
Google
Middlefield Rd
Alpine Rd
Stanford
Weekend Acres
Stanford Ave
G3
Charleston
Meadow
Greenmeadow
Old Middlefield Way
Junipero Serra Blvd
Page Mill Rd
El Camino Real
Monta Loma
280
VA Palo Alto
Health System
H
Arastradero Rd
Foothill Expy
Junipero Serra
W El Camino Real
Castro City
Central Expy
Greater
Miranda
1 mi
1 km

The Shortest Path Problem

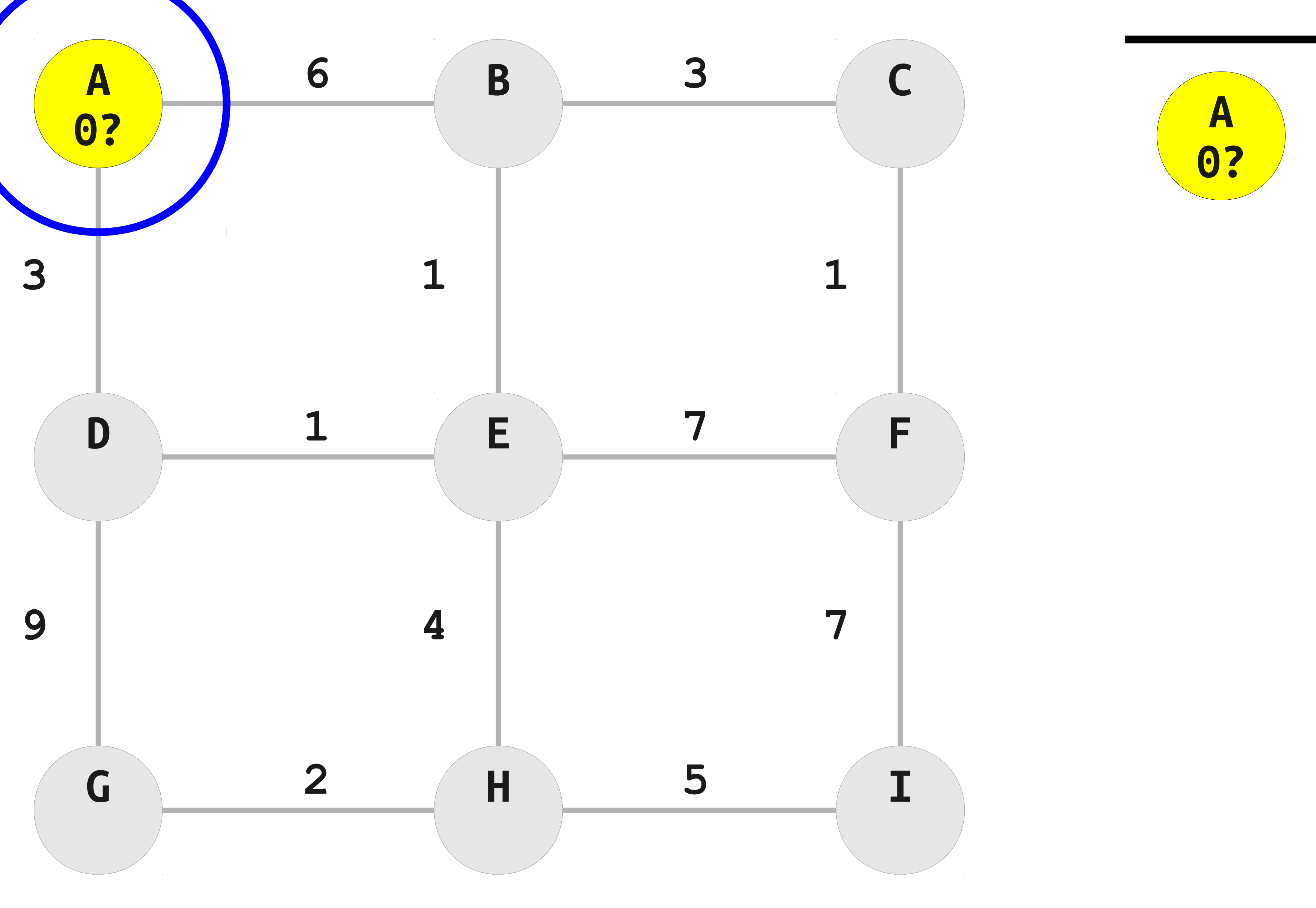
- Suppose that you have a graph representing different locations.
- Each edge has an associated *cost* (or *weight*, *length*, etc.). We'll assume costs are nonnegative.
- Goal: Find the least-cost (or lowest-weight, lowest-length, etc.) path from some node u to a node v.

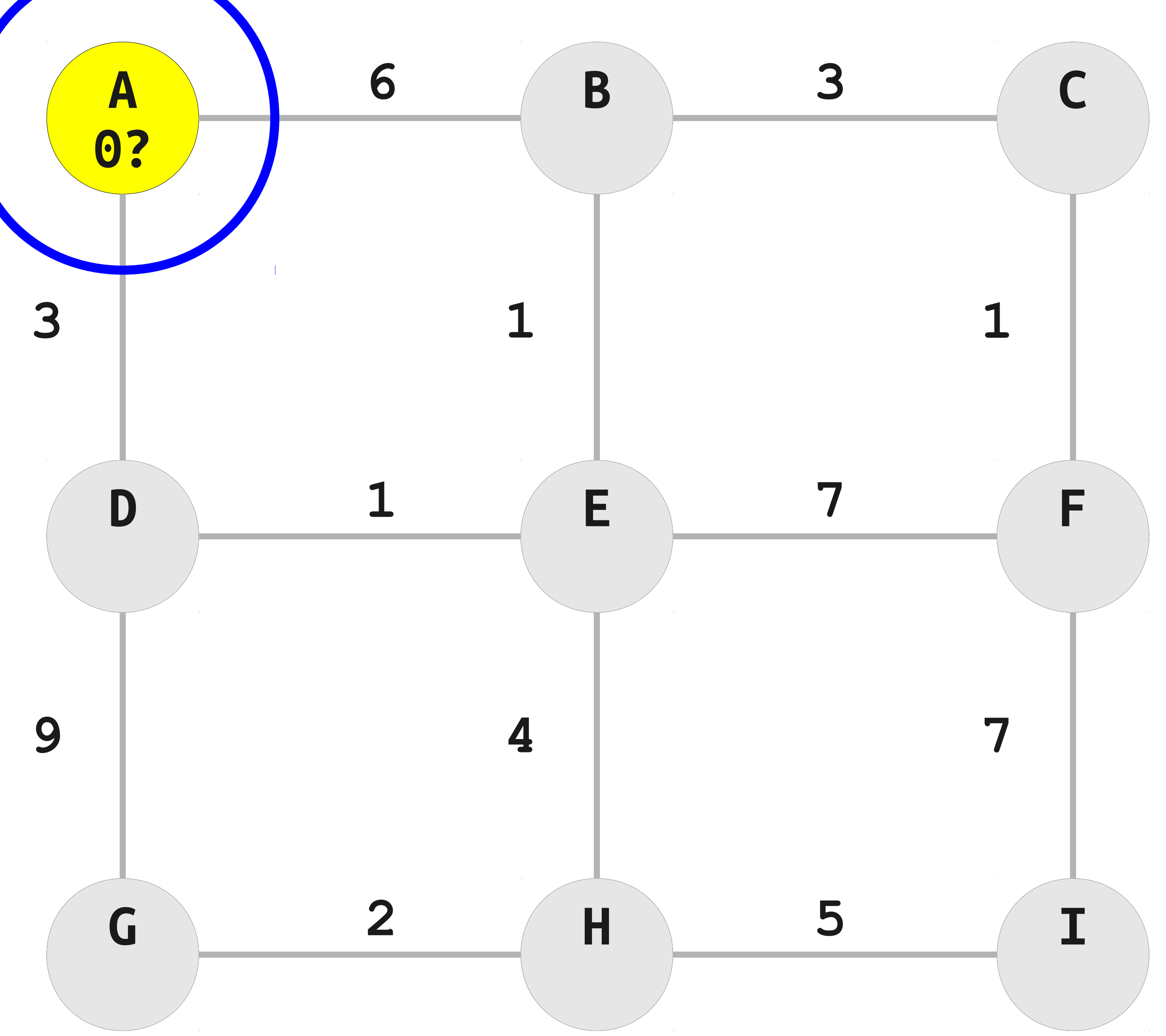

A
6
B
3
C
3
1
1
D
1
E
7
F
9
4
7
G
2
H
5
I


A
0?
6
B
3
C
3
1
1
D
1
E
7
F
9
4
7
G
2
H
5
I

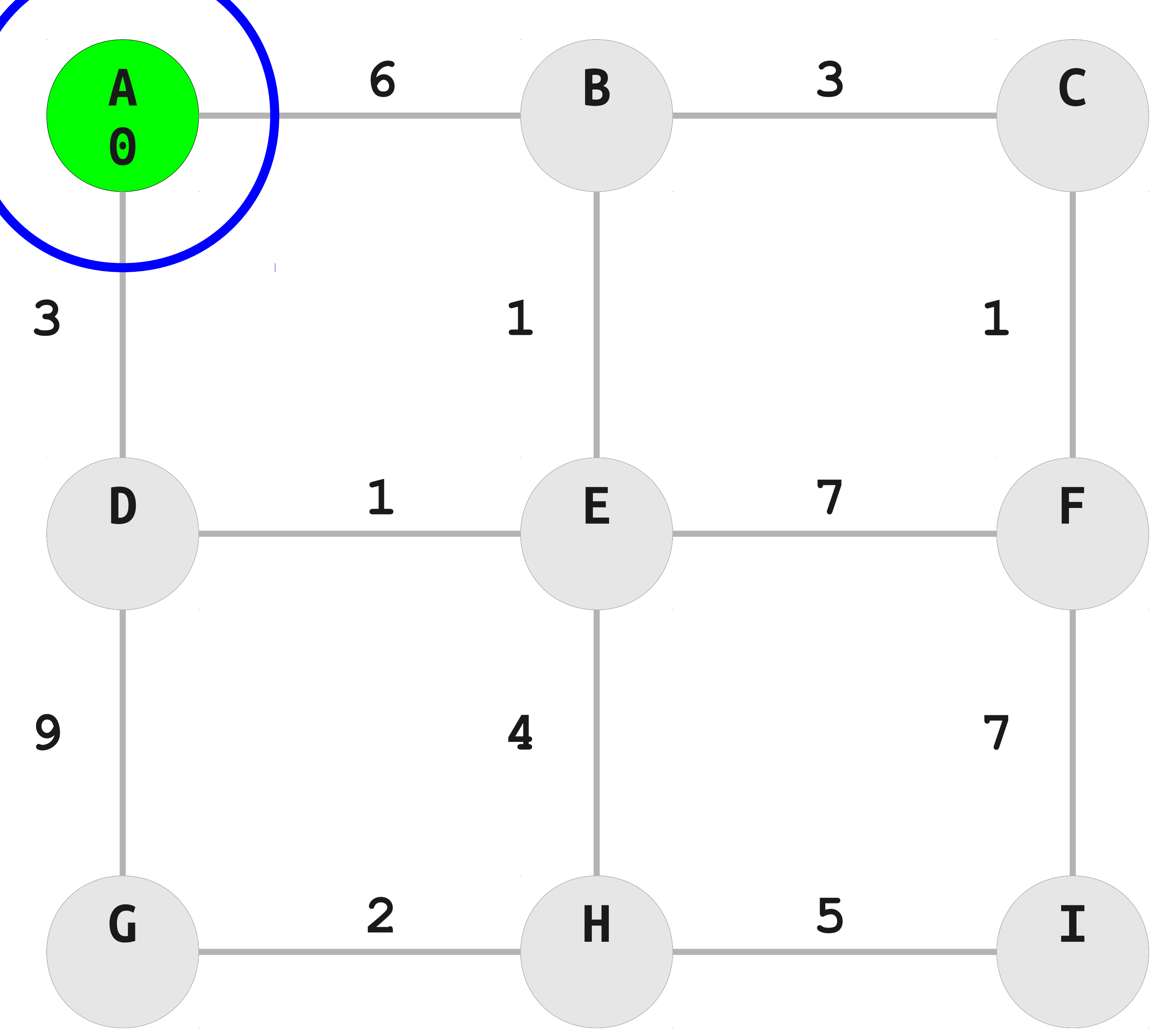


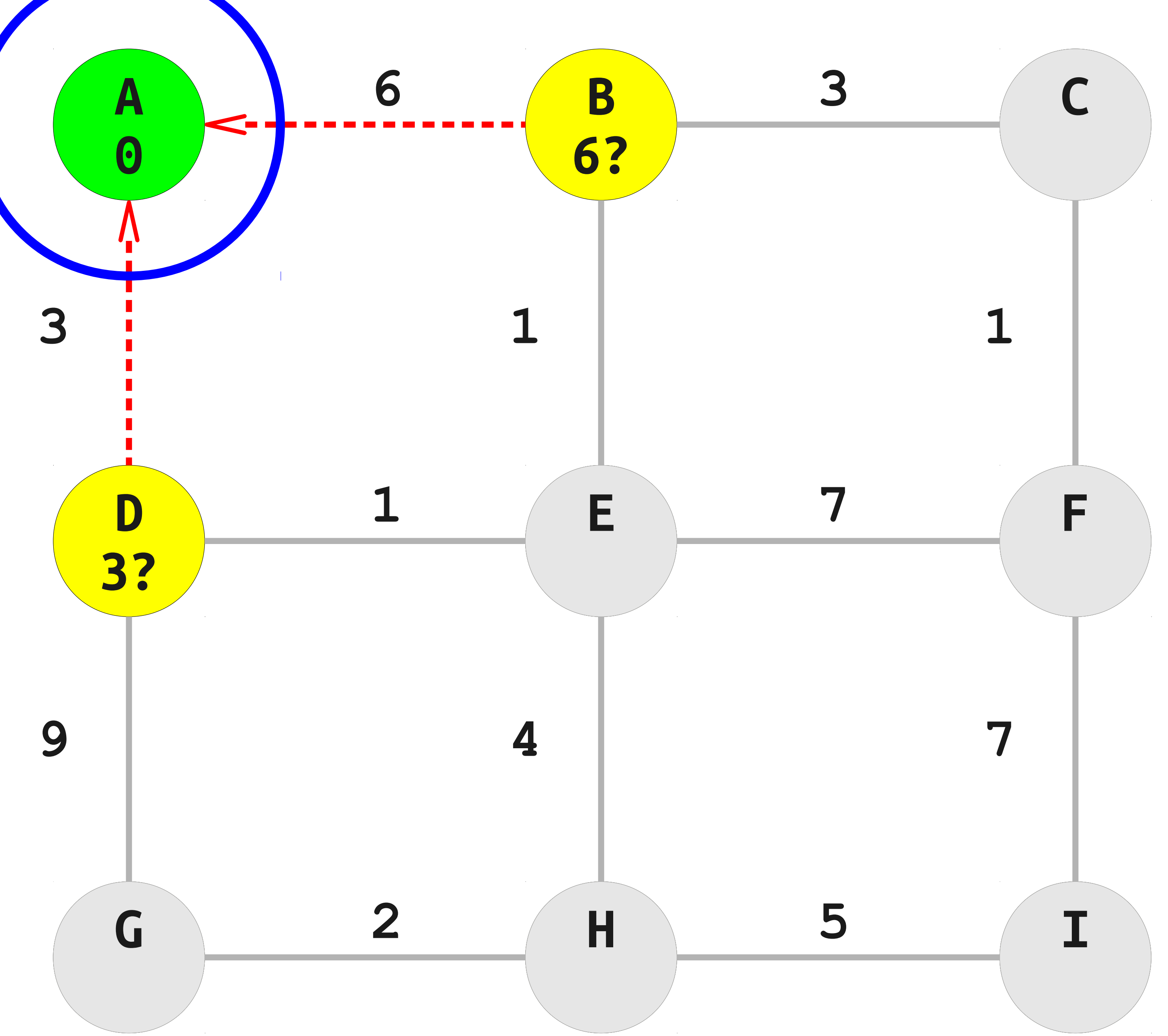
A
0?

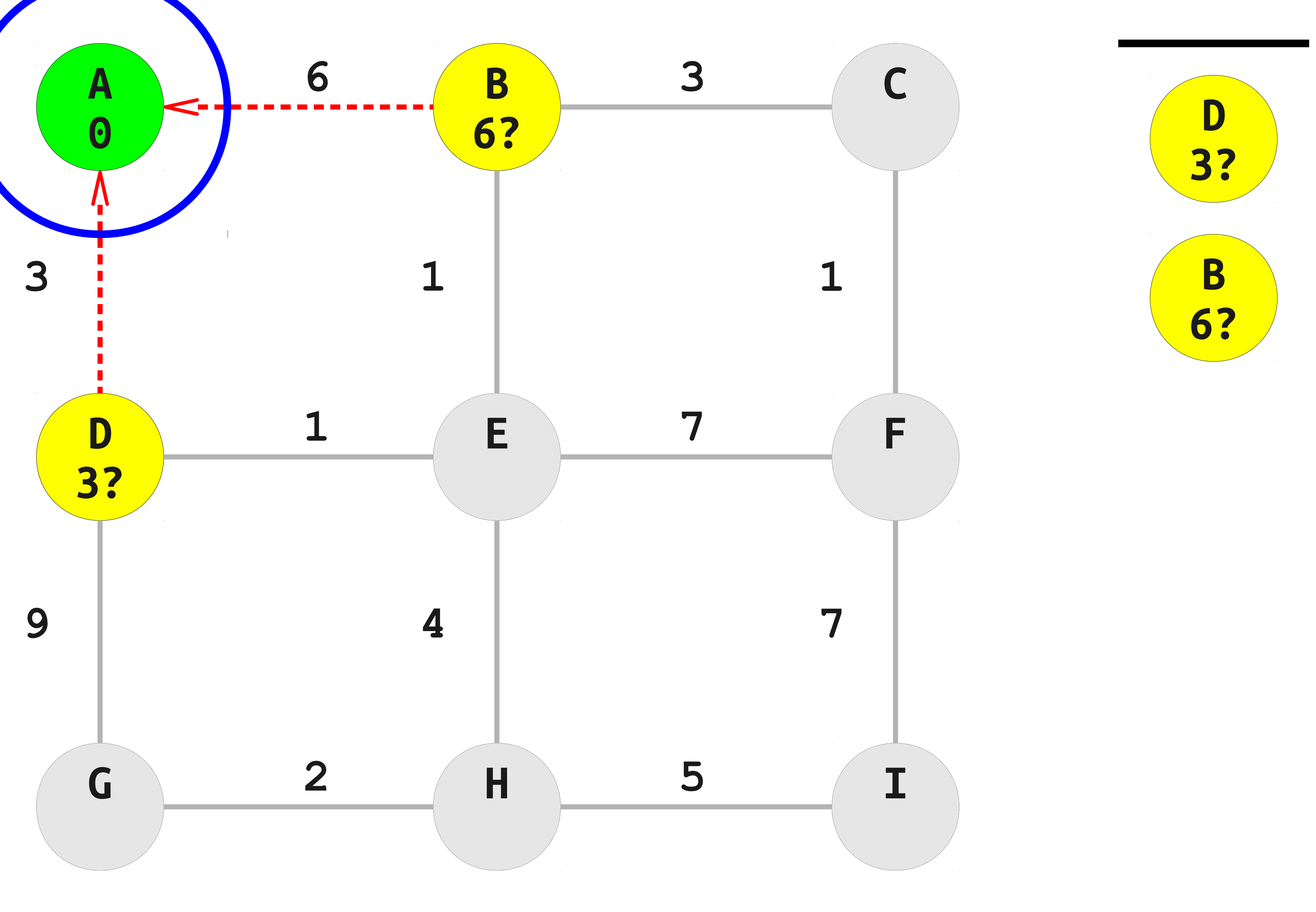

A
0?
6
B
3
C
A
0?
3
1
1
D
1
E
7
F
9
4
7
G
2
H
5
I

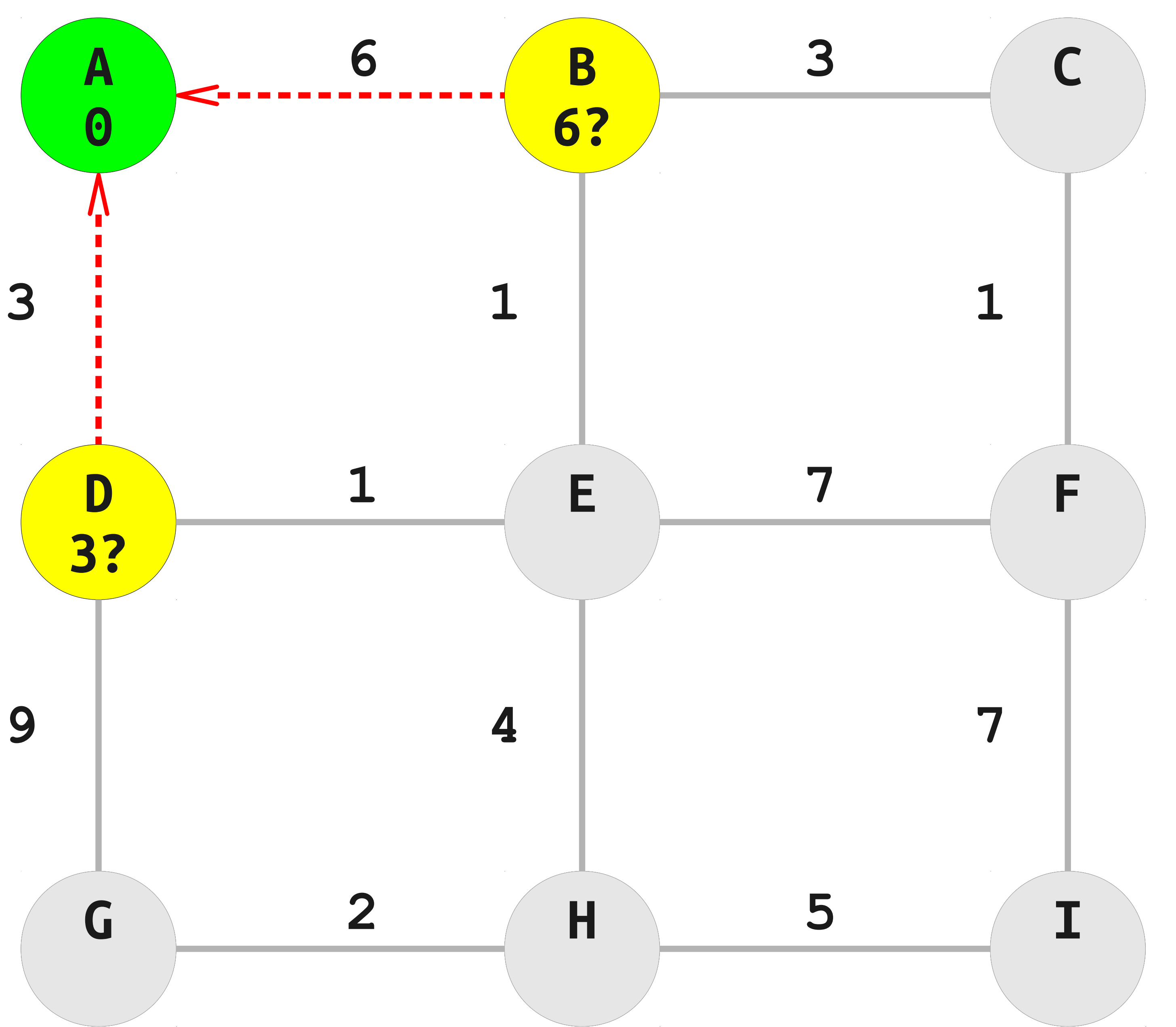


A
0?
6
B
3
C
3
1
1
D
1
E
7
F
9
4
7
G
2
H
5
I

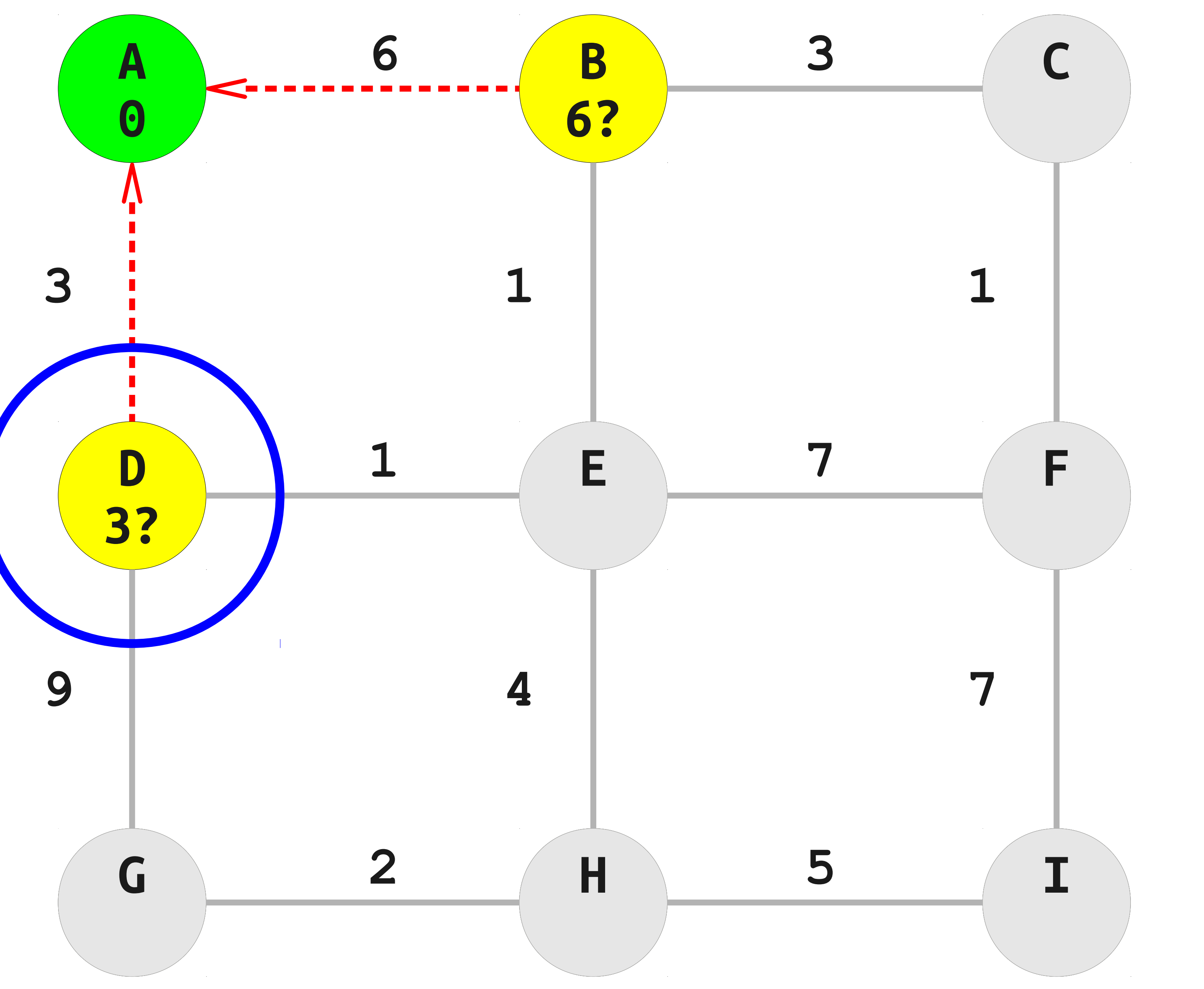

A
0
6
B
3
C
3
1
1
D
1
E
7
F
9
4
7
G
2
H
5
I


A
0
6
B
6?
3
C
3
1
1
D
3?
1
E
7
F
9
4
7
G
2
H
5
I


A
0
6
B
6?
3
C
3
1
1
D
3?
1
E
7
F
9
4
7
G
2
H
5
I
D
3?
B
6?

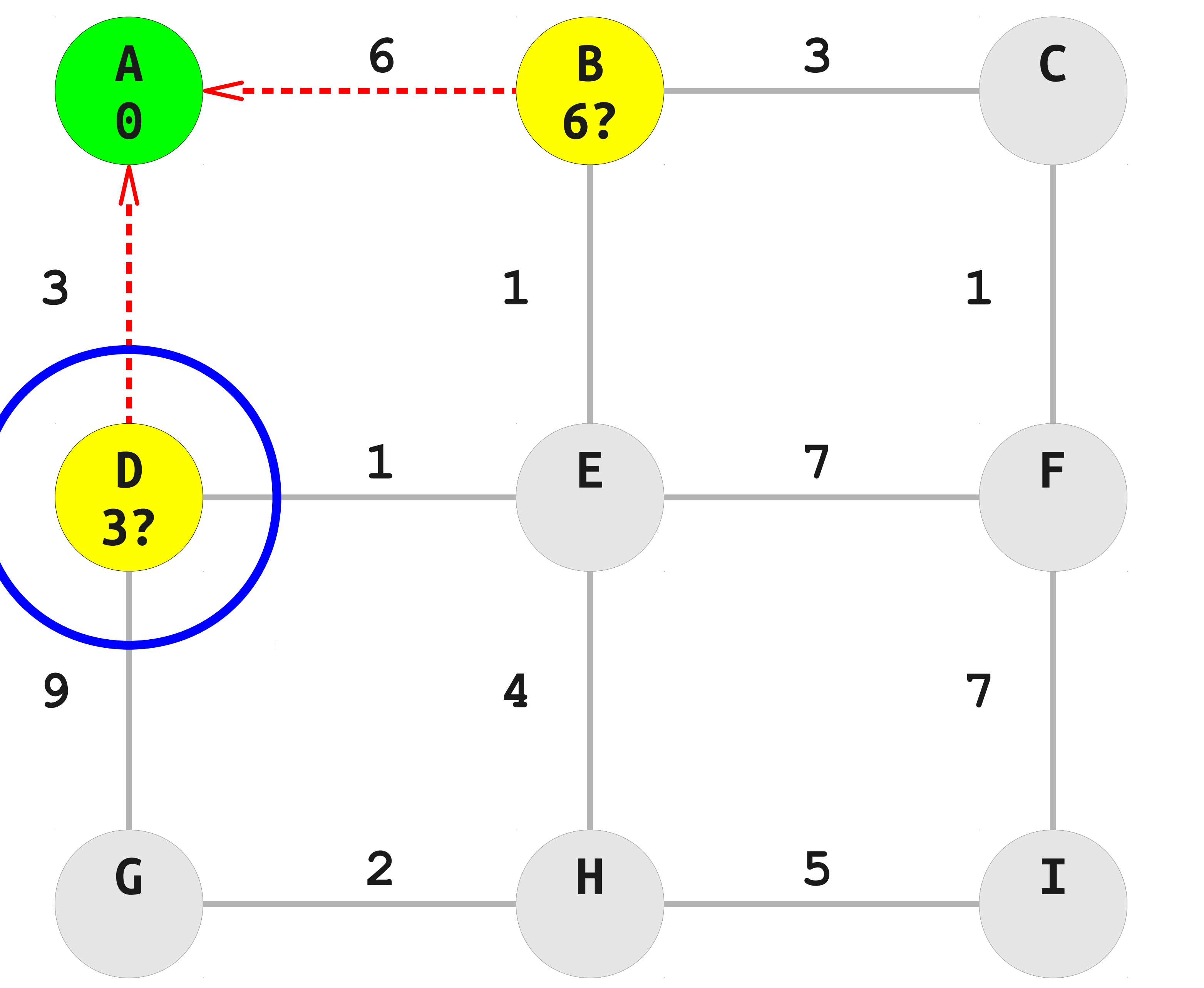

A
0
6
B
6?
3
C
3
1
1
D
3?
1
E
7
F
9
4
7
G
2
H
5
I

D
3?
B
6?

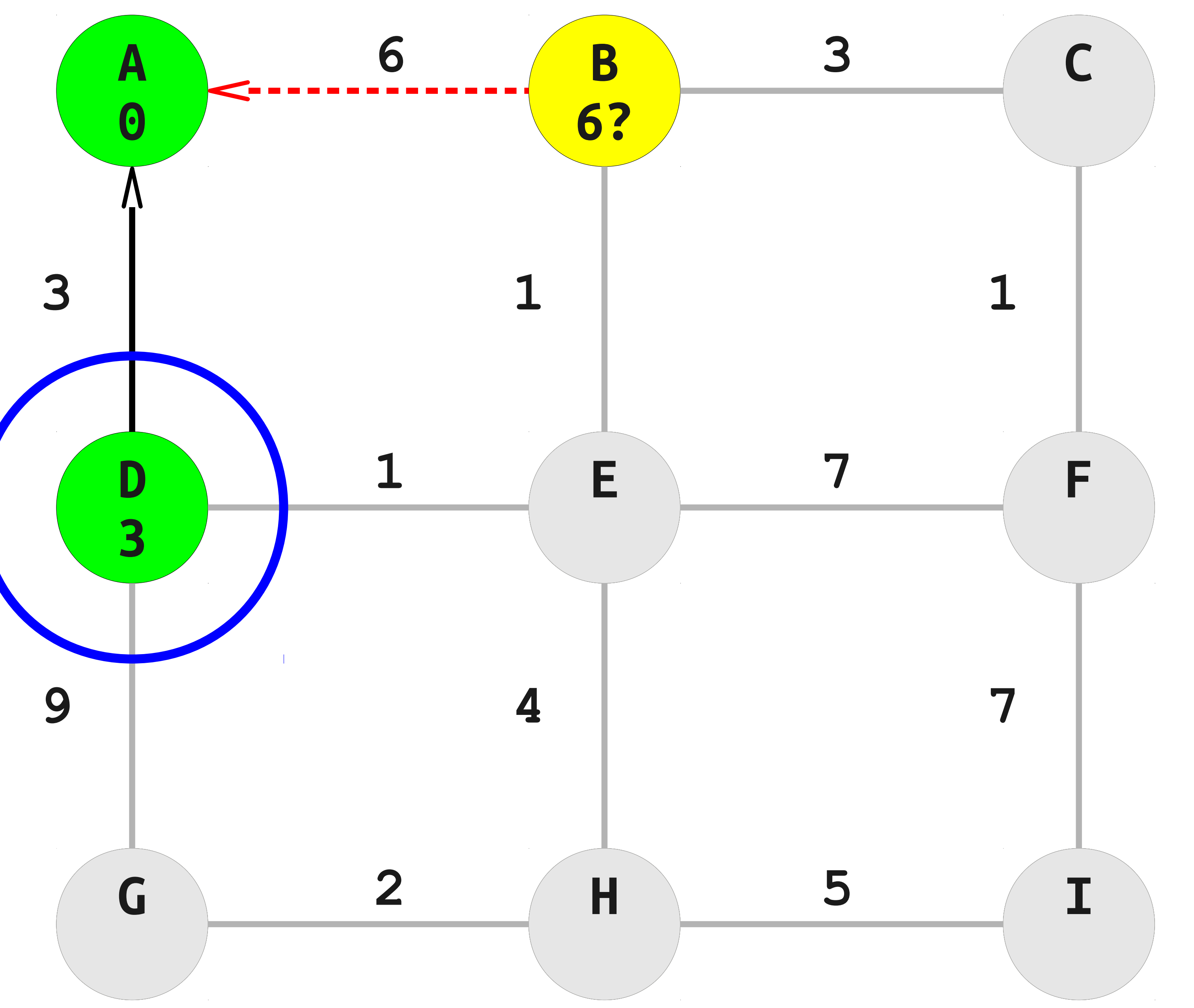


D
3?

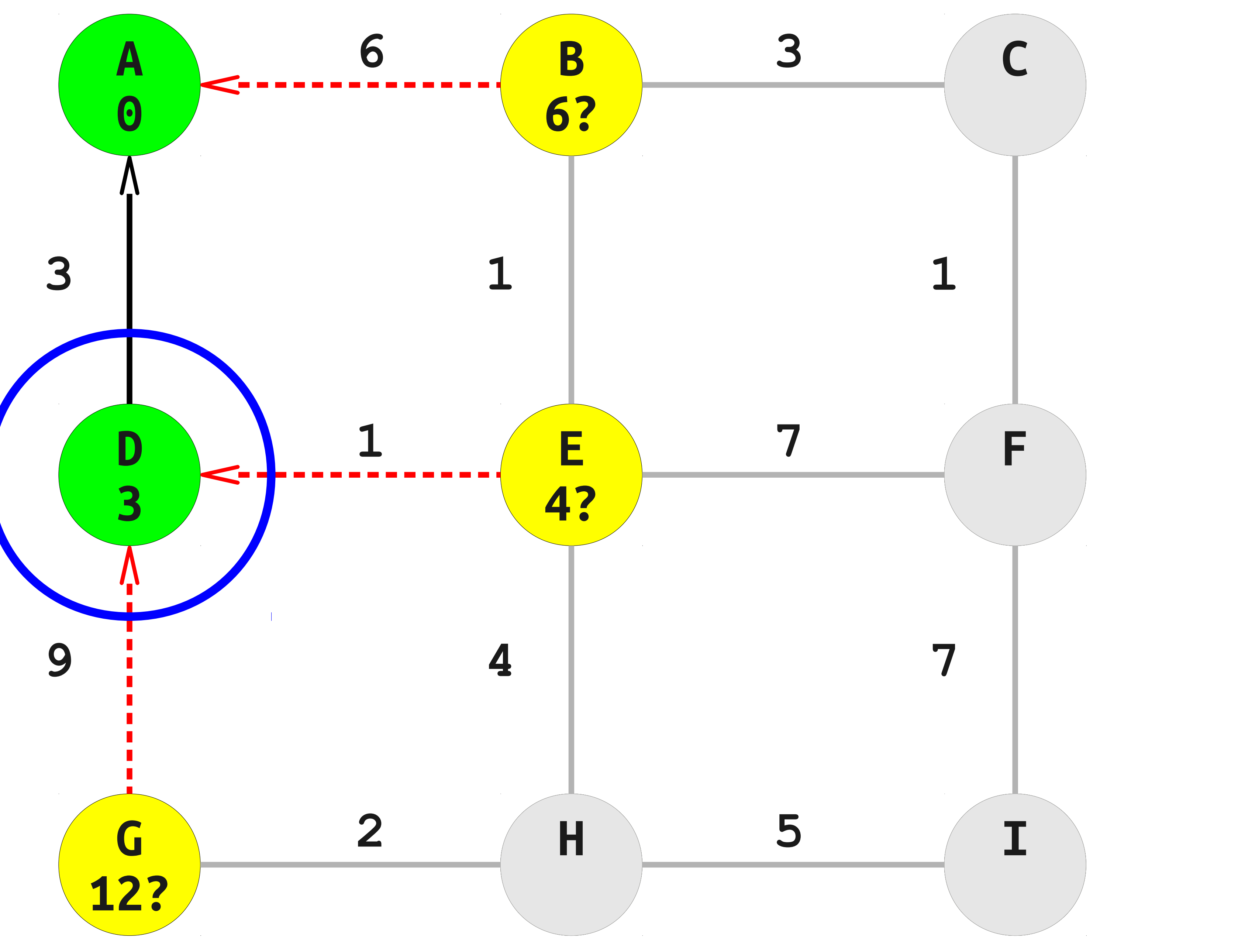
B
6?



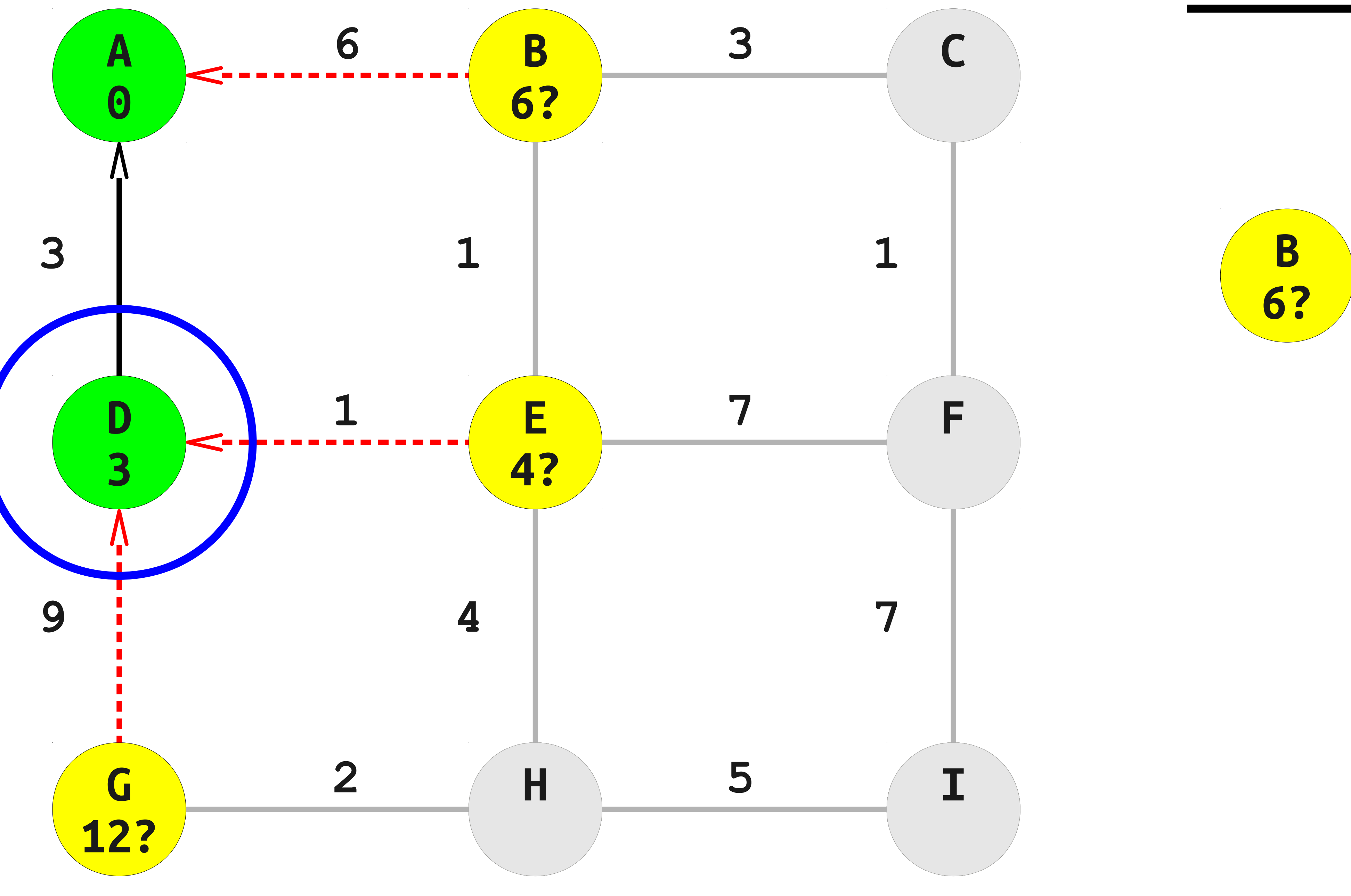
B
6?

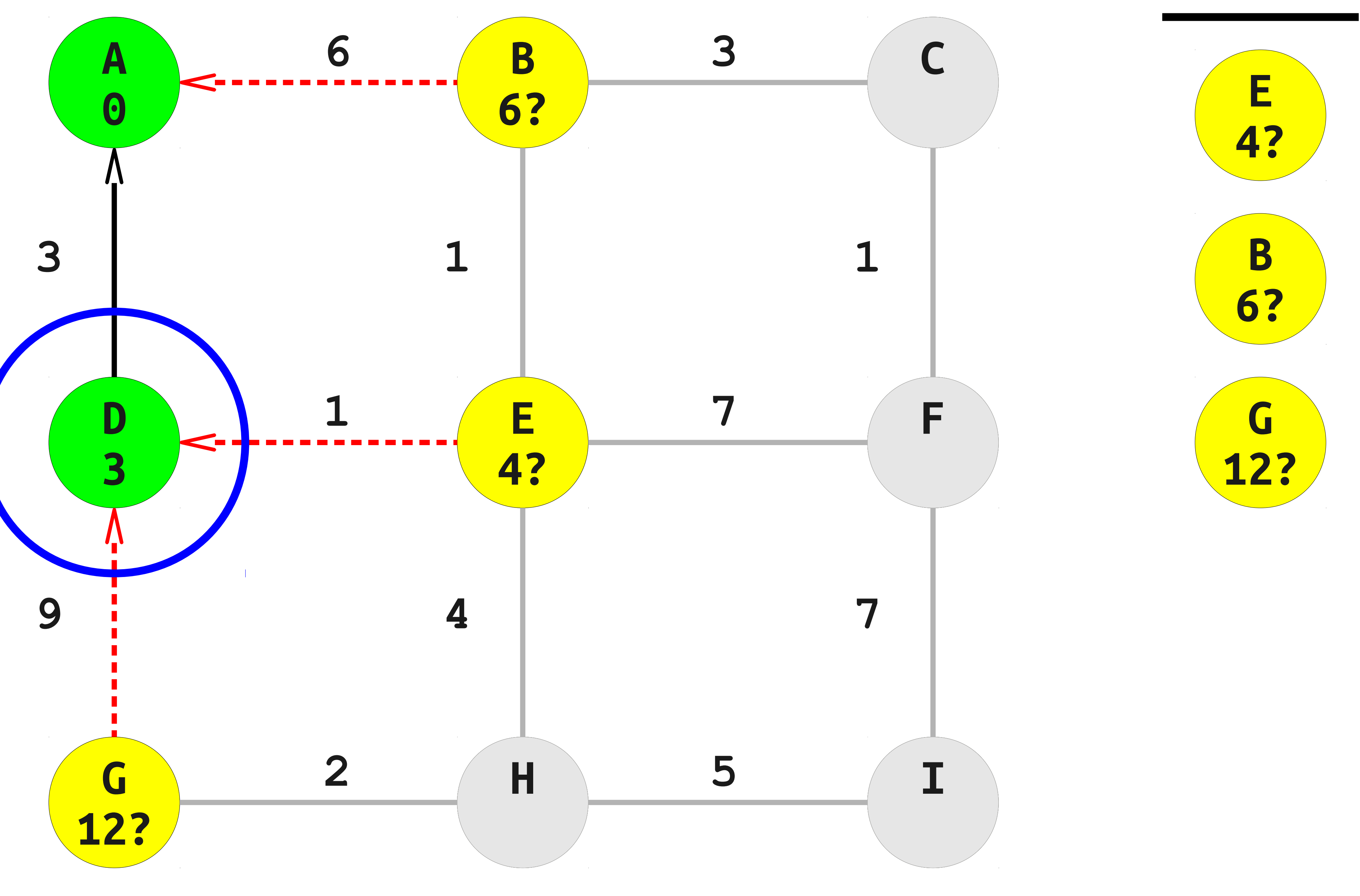


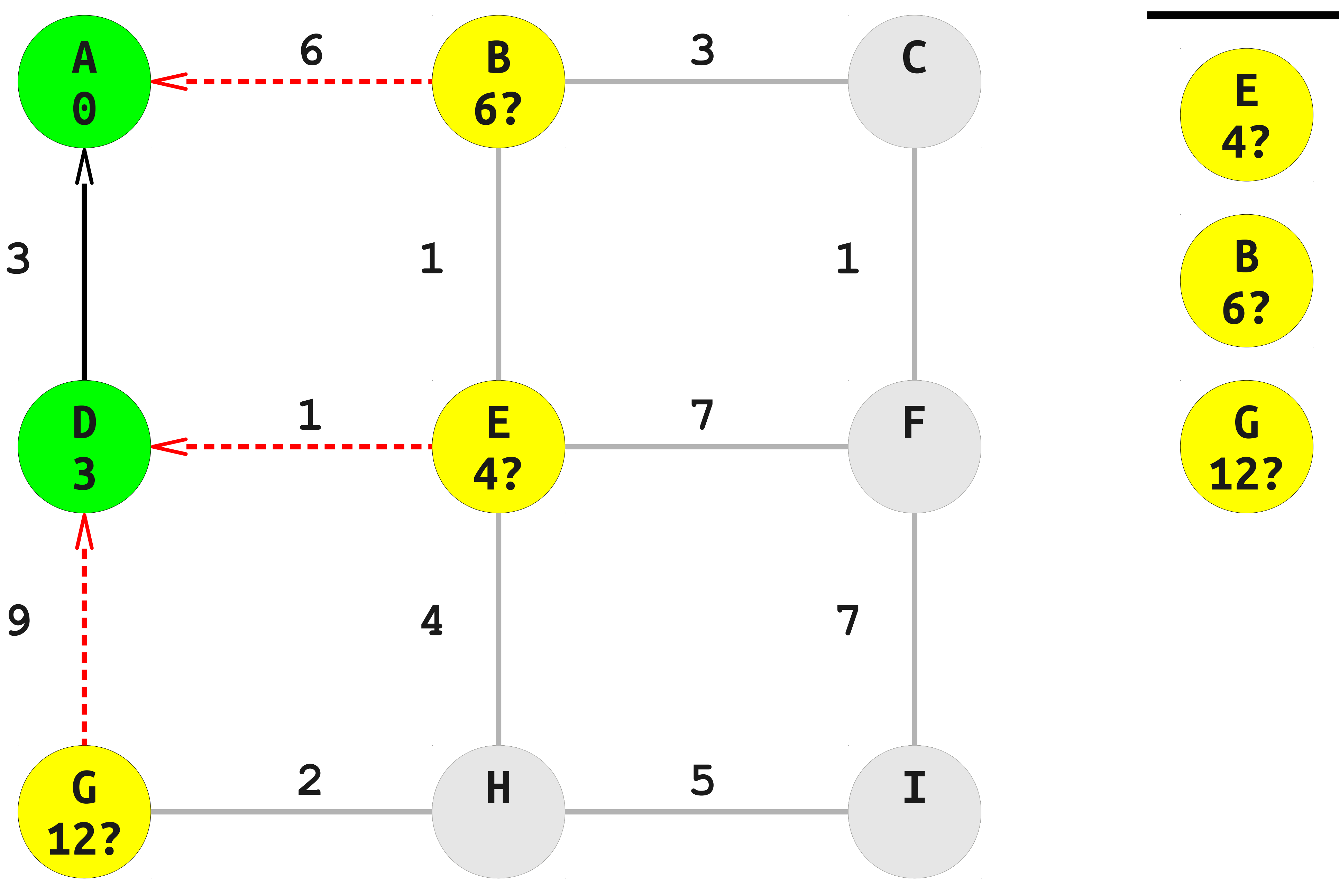
B
6?



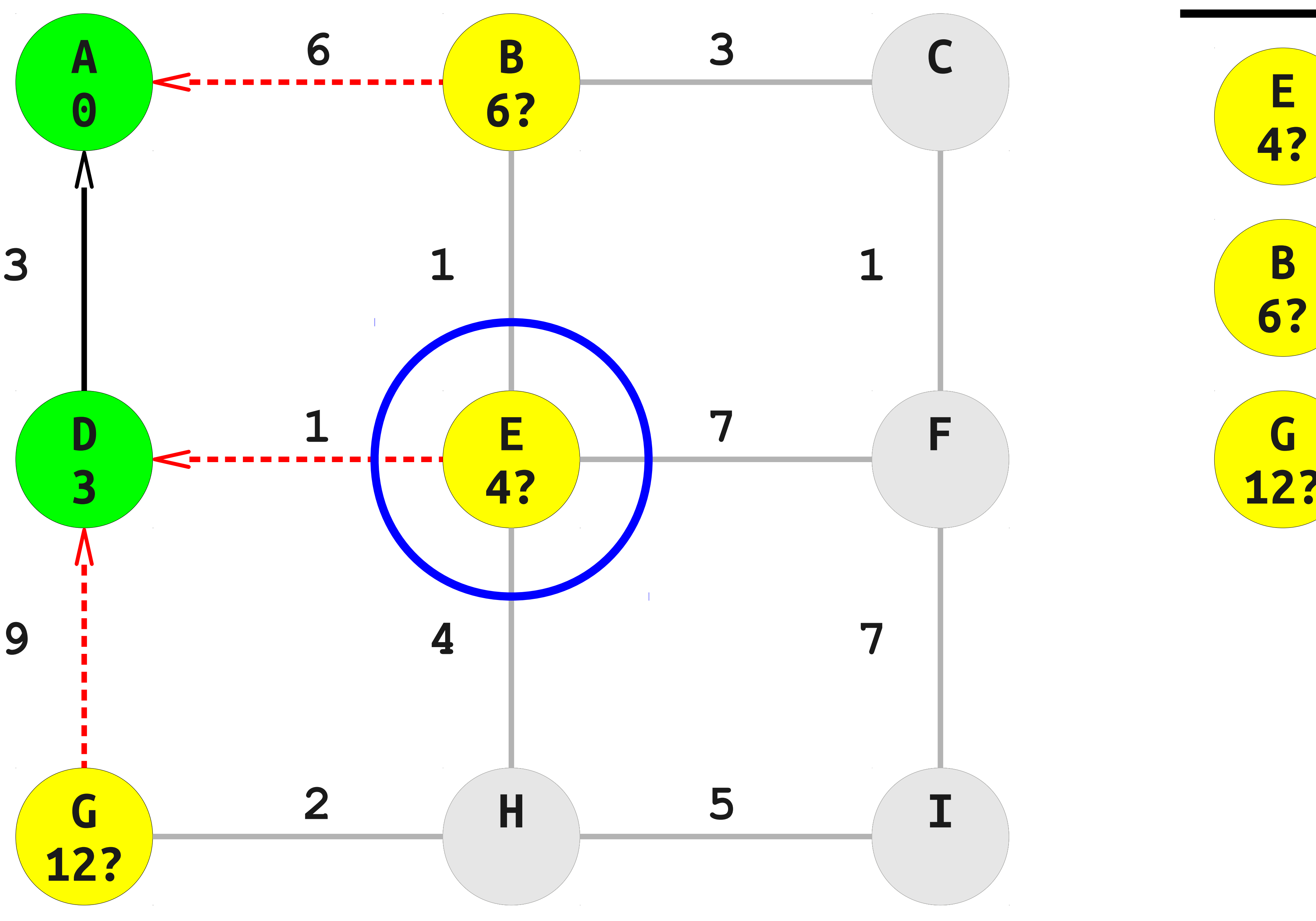
B
6?

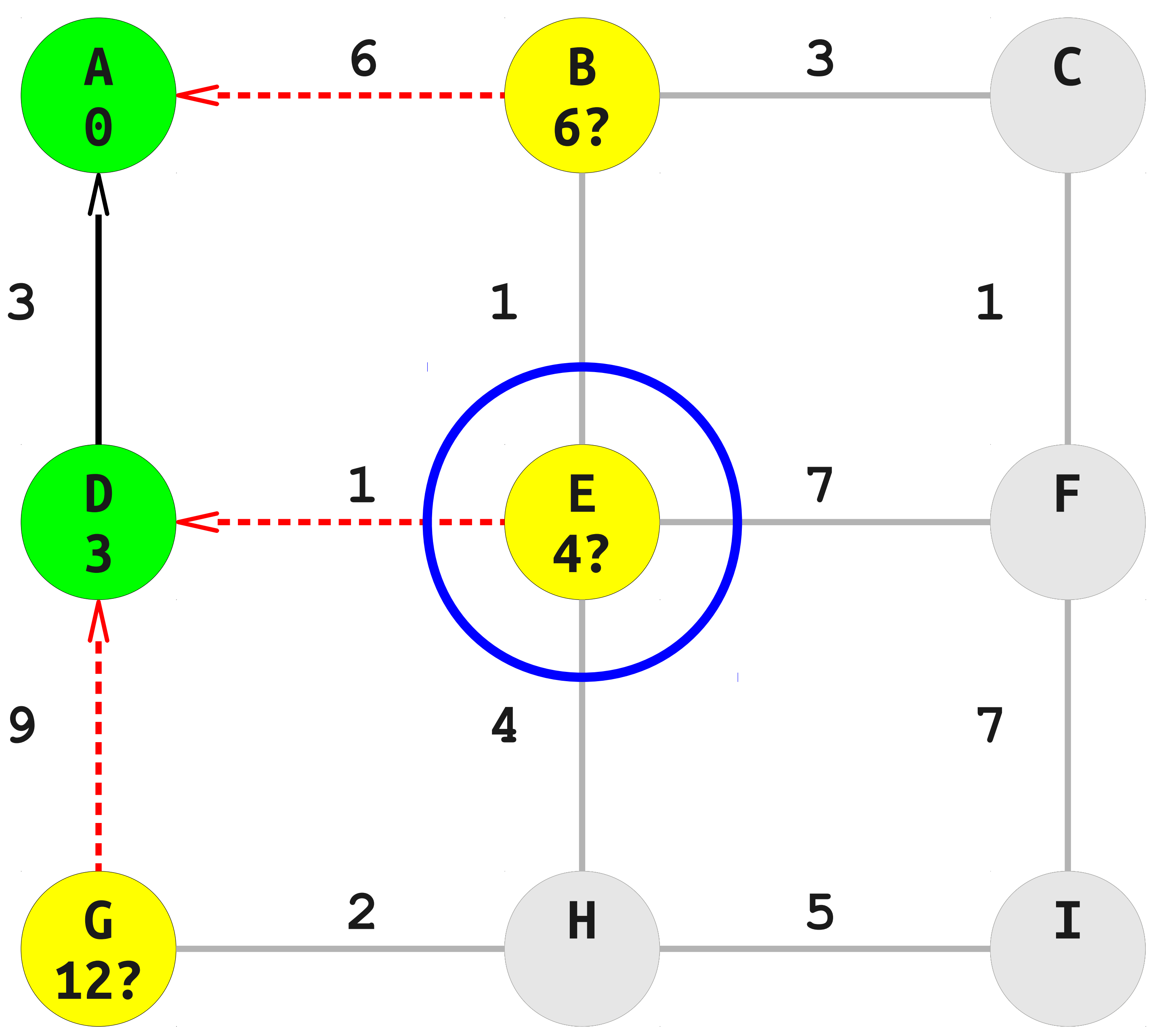

A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4?
7
F
9
4
7
G
12?
2
H
5
I
B
6?


A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4?
7
F
9
4
7
G
12?
2
H
5
I
E
4?
B
6?
G
12?



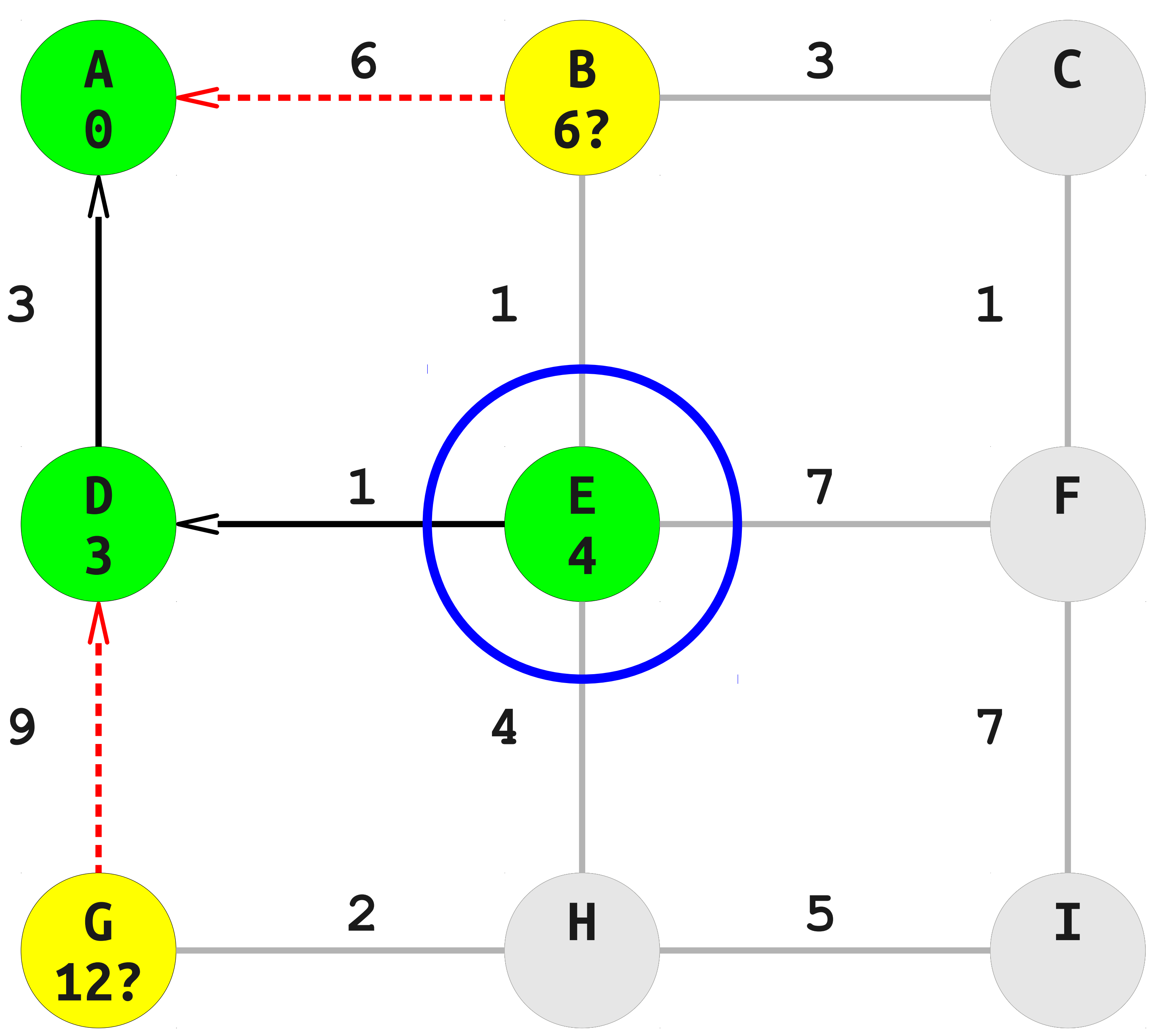
A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4?
7
F
9
4
7
G
12?
2
H
5
I
E
4?
B
6?
G
12?


A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4?
7
F
9
4
7
G
12?
2
H
5
I
E
4?
B
6?
G
12?



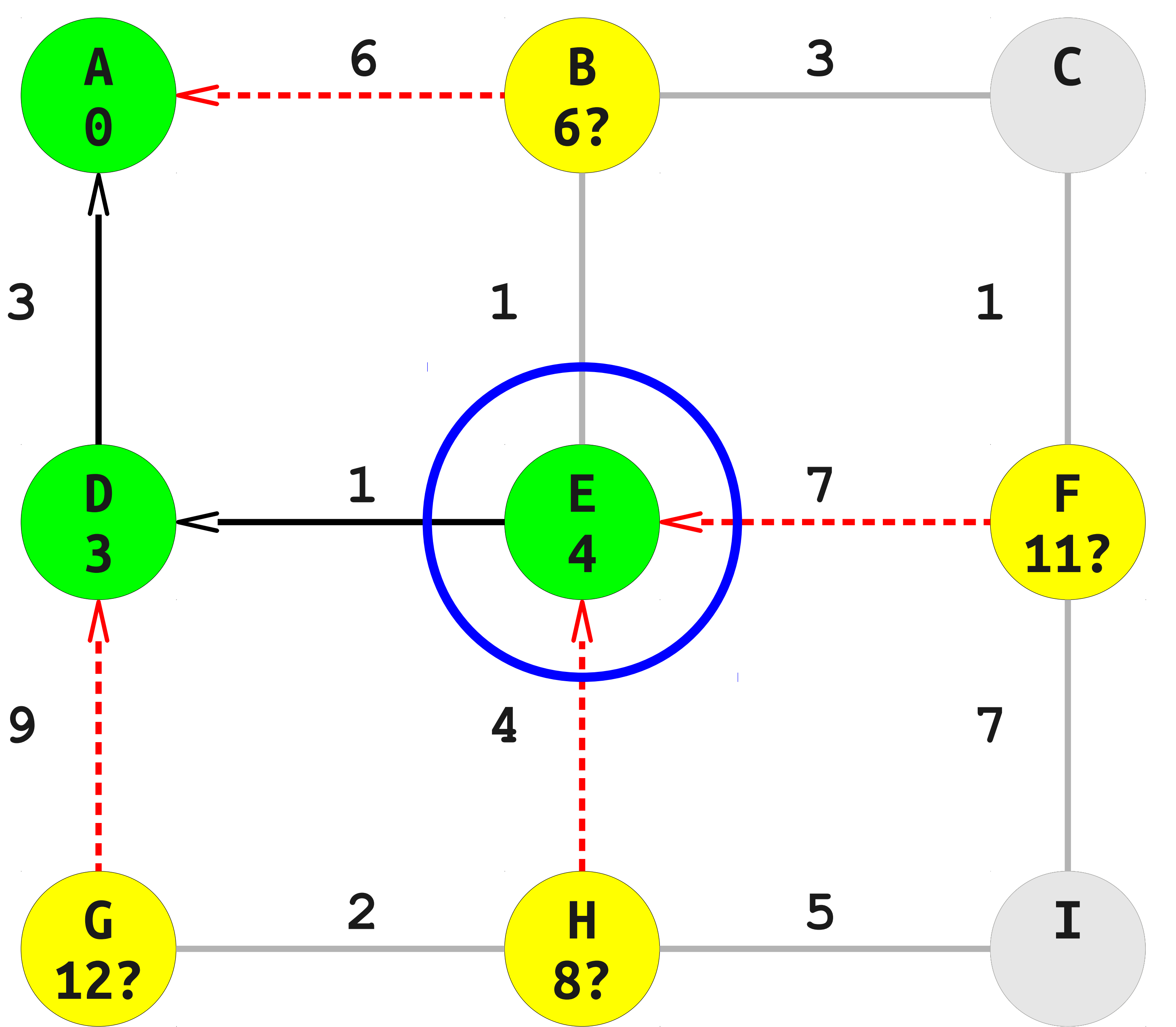
B
6?

G
12?



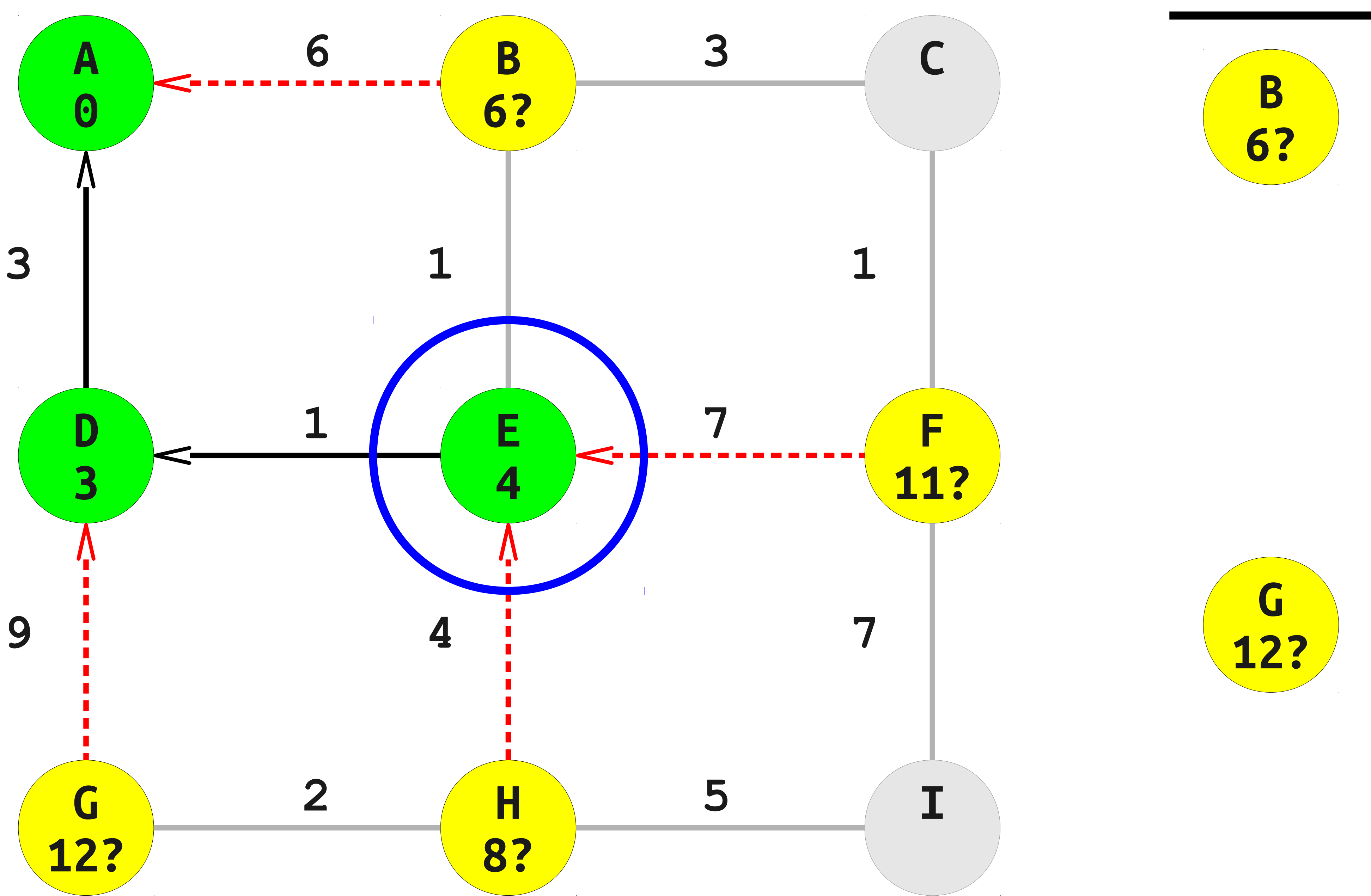
B
6?

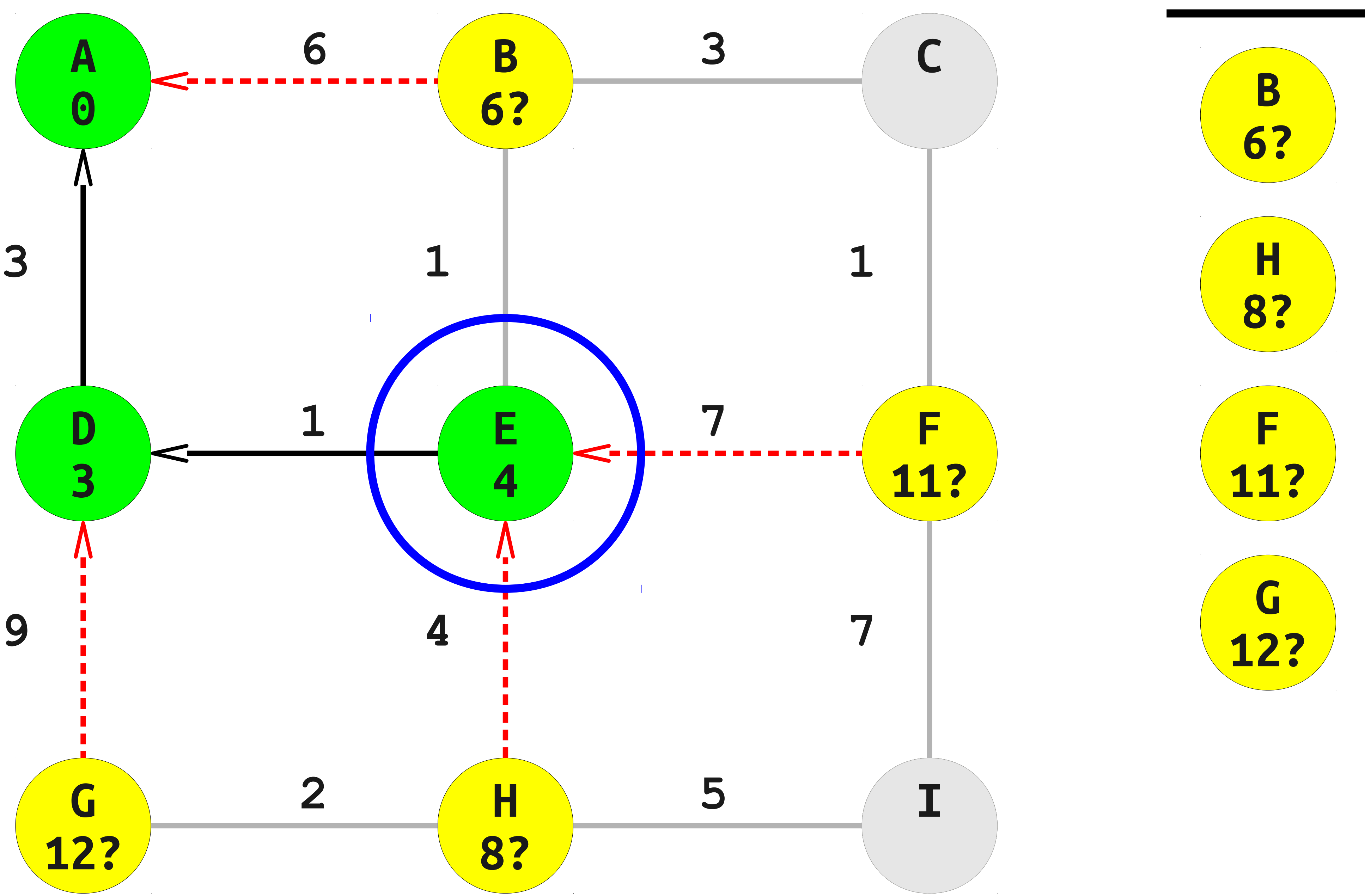
G
12?



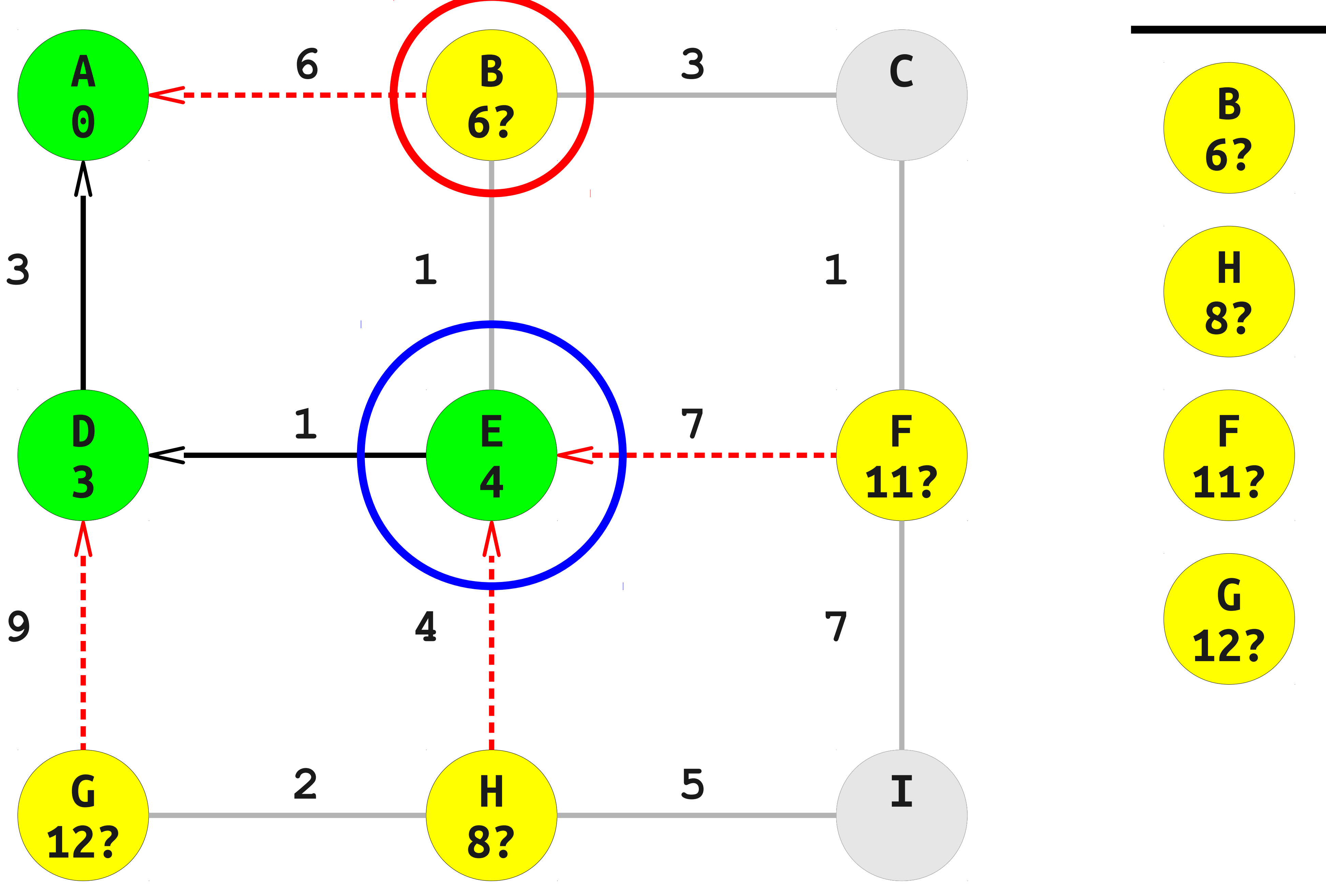
B
6?

G
12?

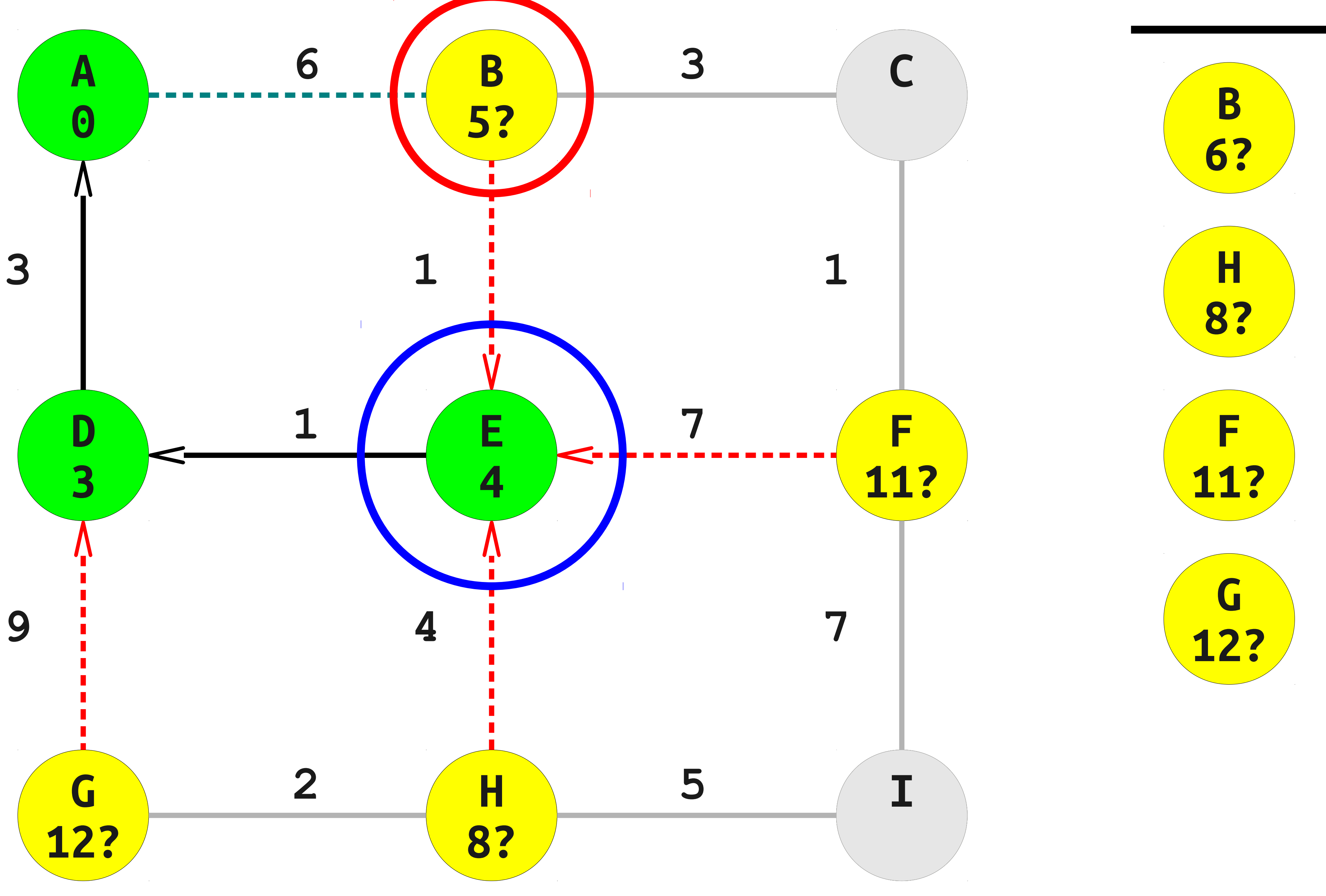

A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
6?
G
12?

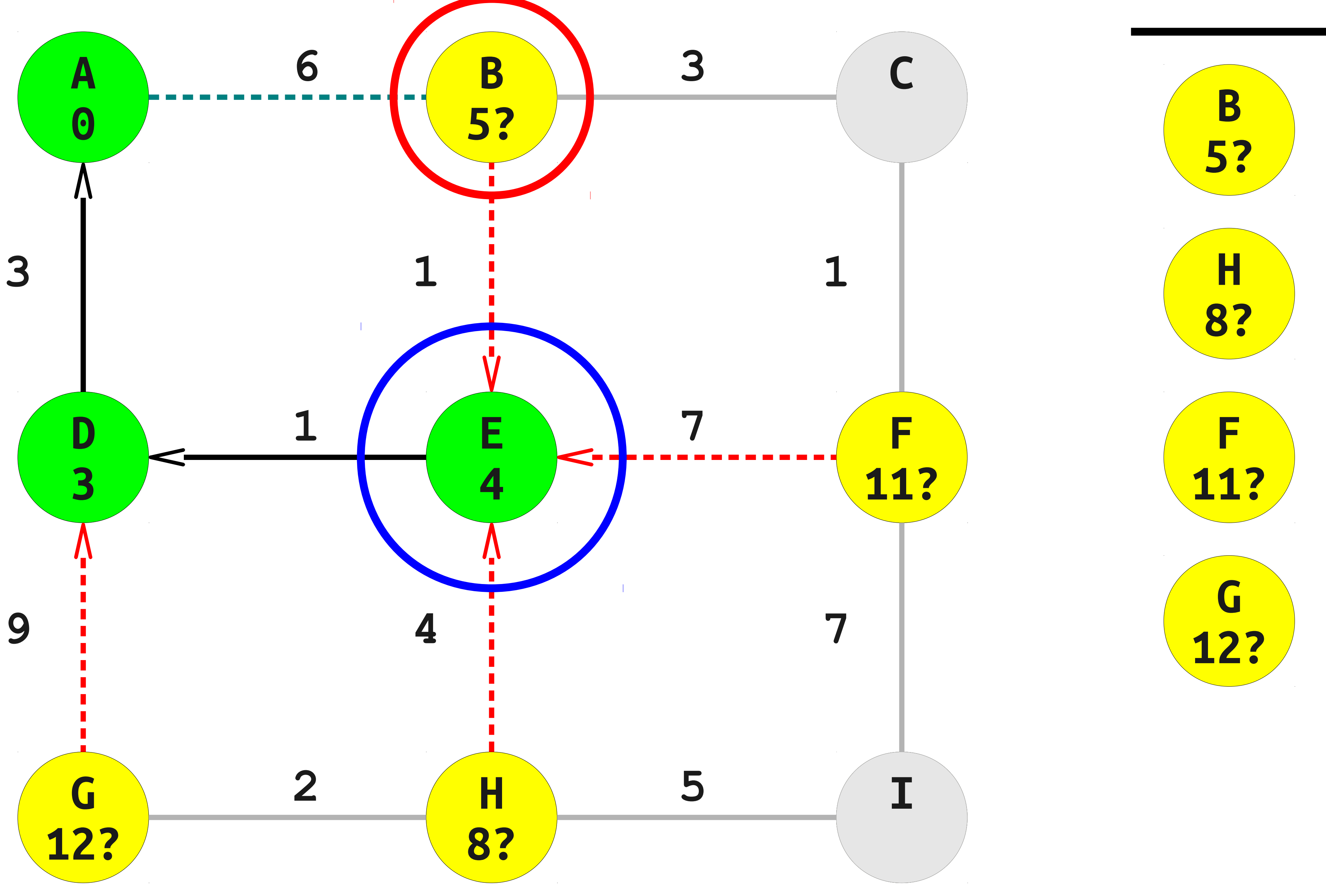


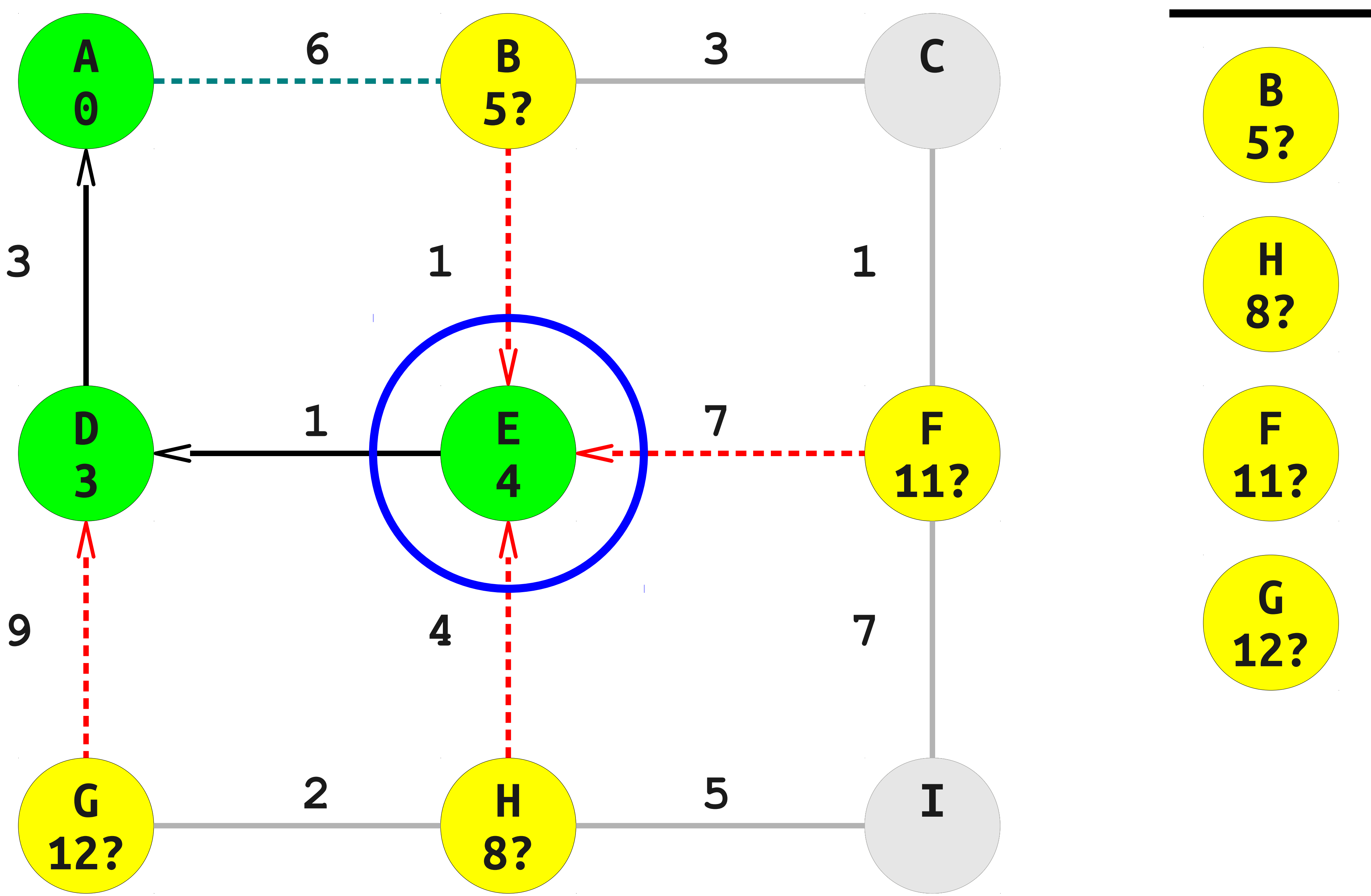
A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
6?
H
8?
F
11?
G
12?

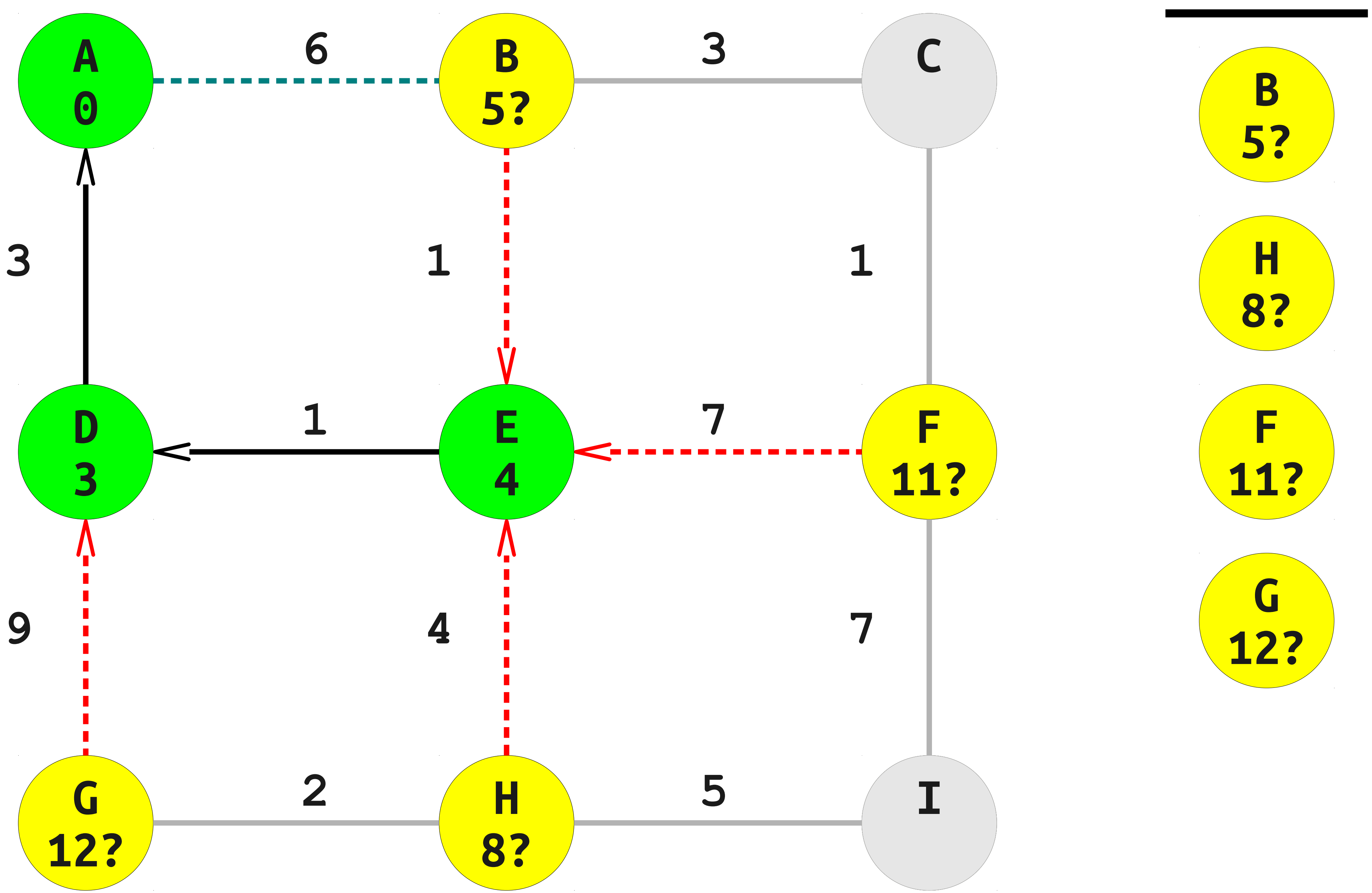


A
0
6
B
6?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
6?
H
8?
F
11?
G
12?

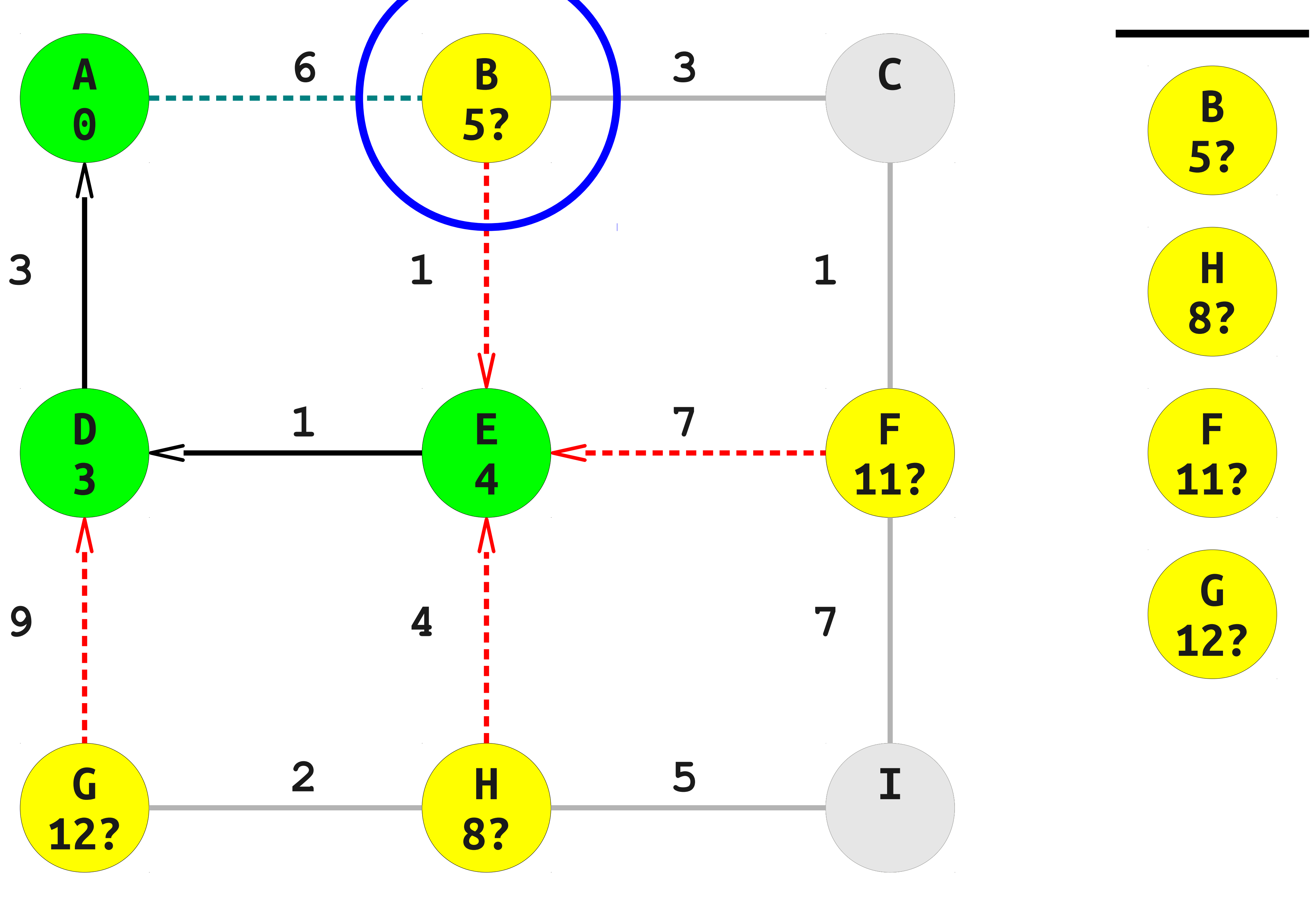

A
0
6
B
5?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
6?
H
8?
F
11?
G
12?

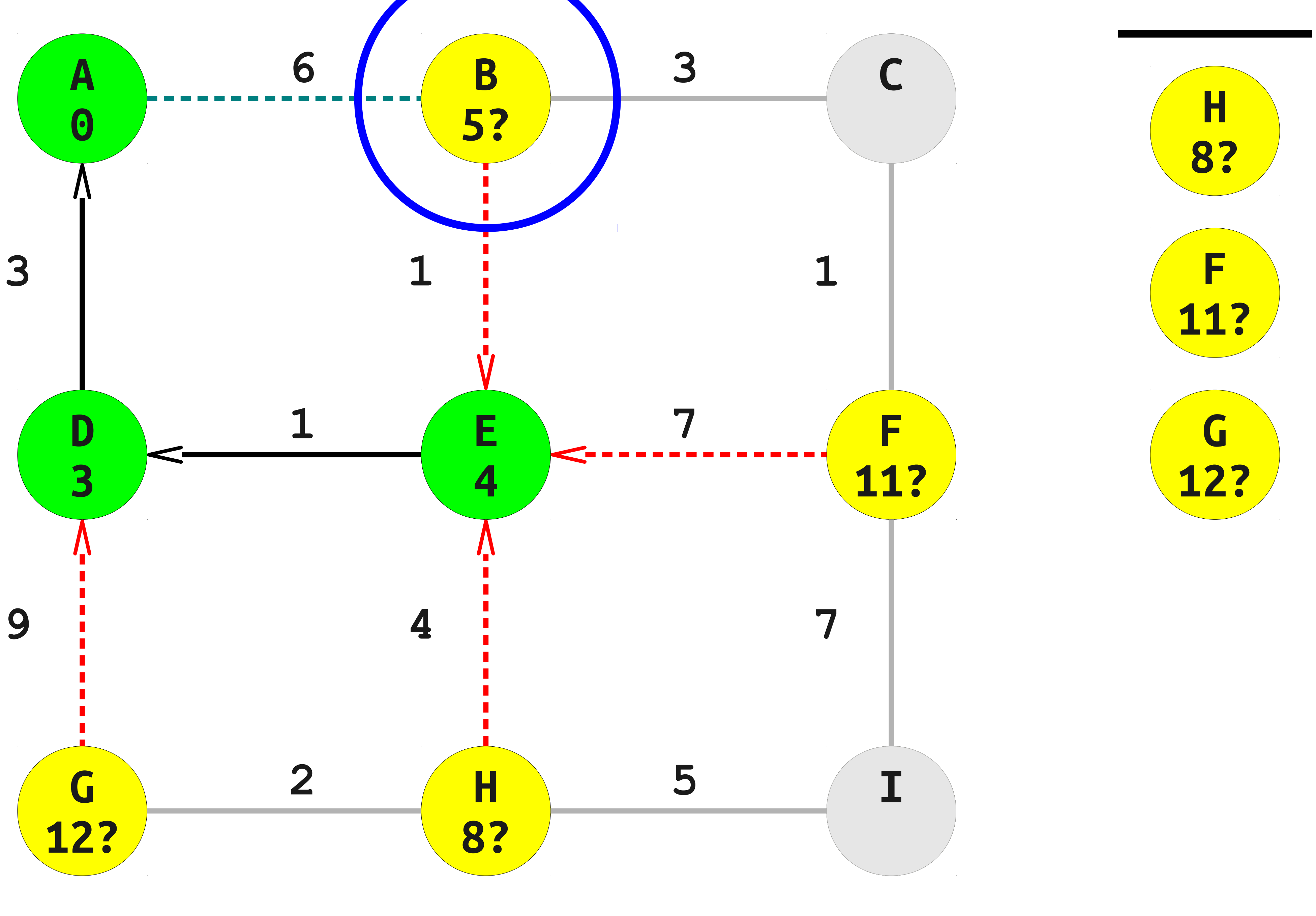

A
0
6
B
5?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
5?
H
8?
F
11?
G
12?


A
0
6
B
5?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
5?
H
8?
F
11?
G
12?

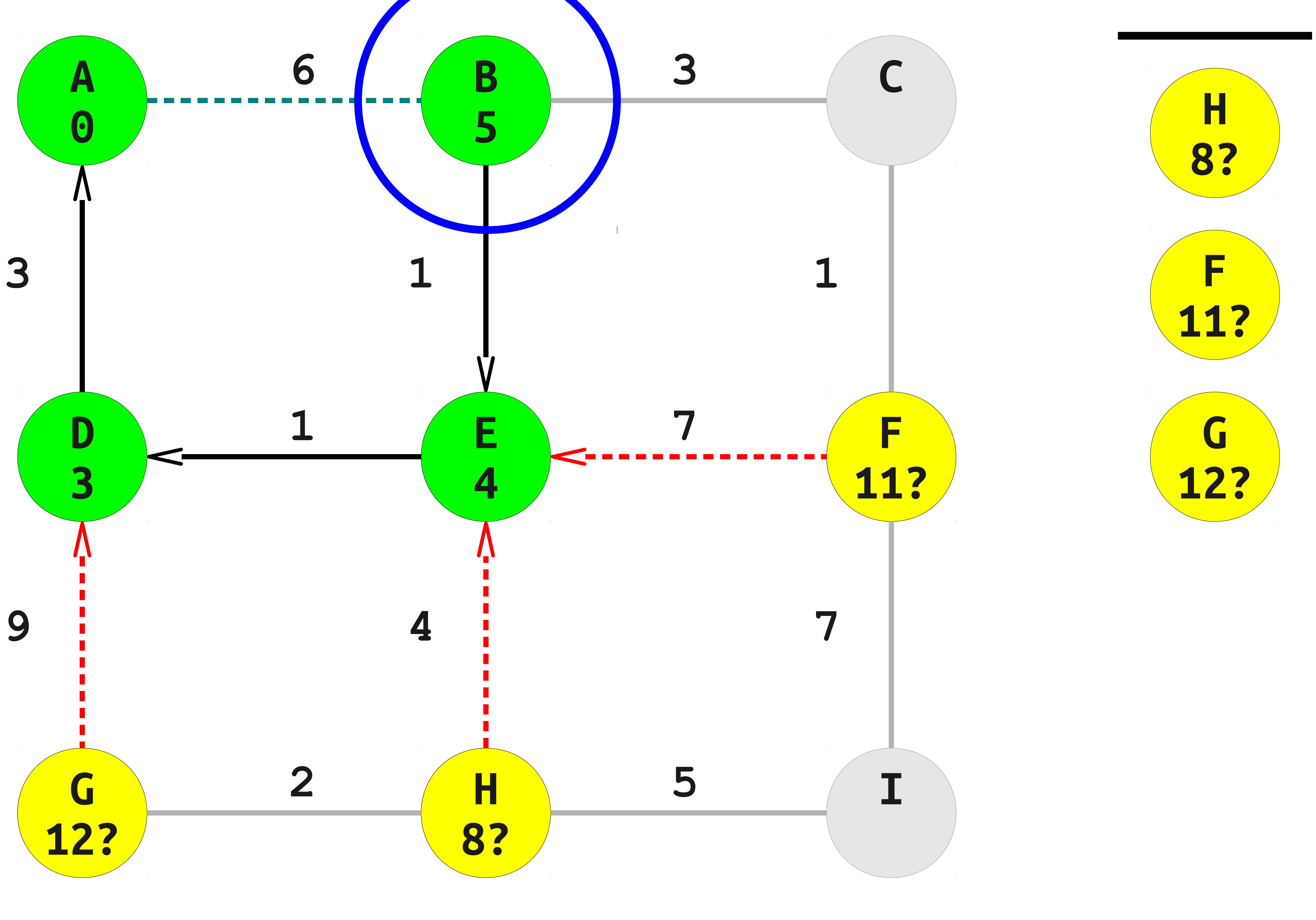


A
0
6
B
5?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
5?
H
8?
F
11?
G
12?

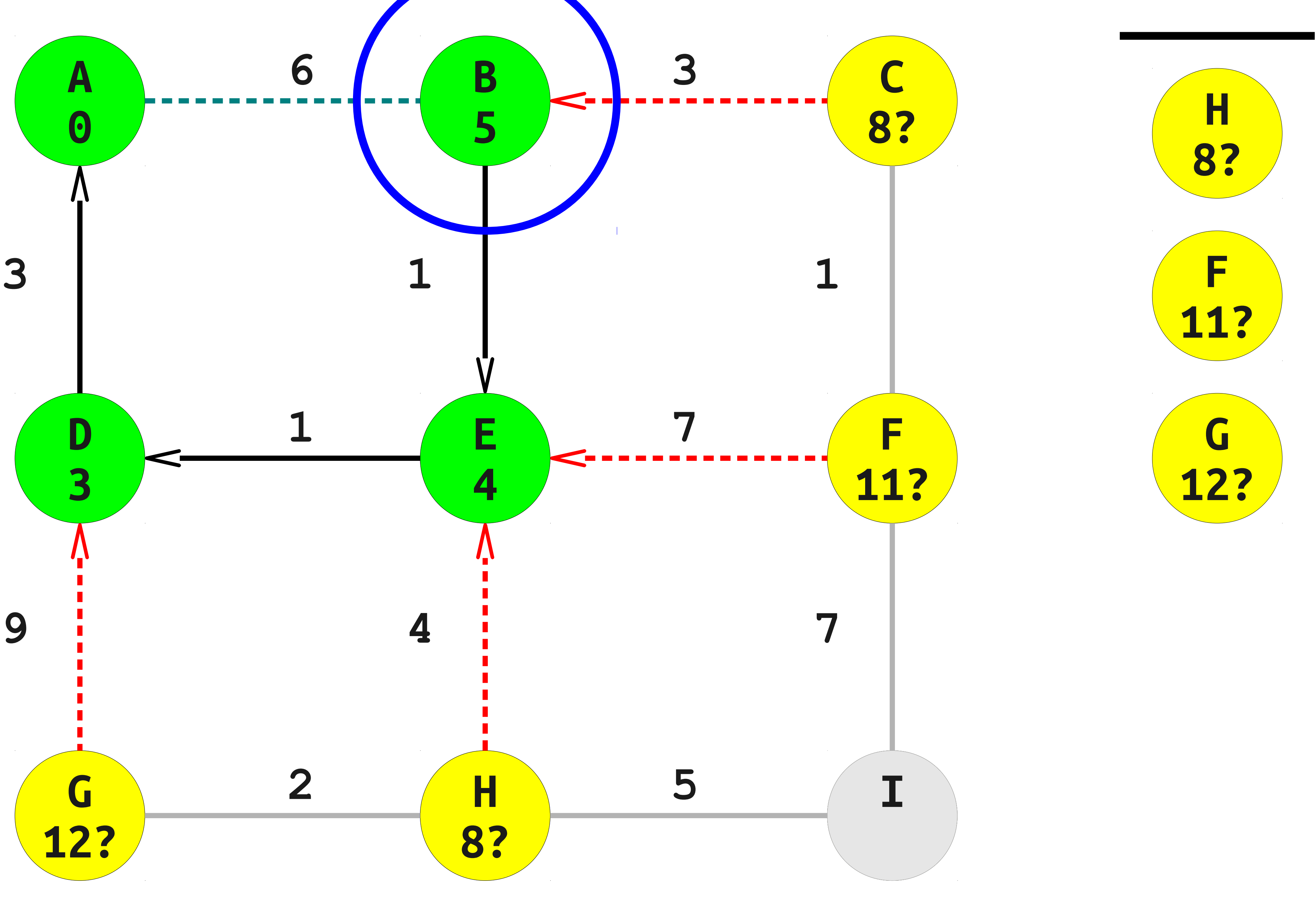

A
0
6
B
5?
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
B
5?
H
8?
F
11?
G
12?

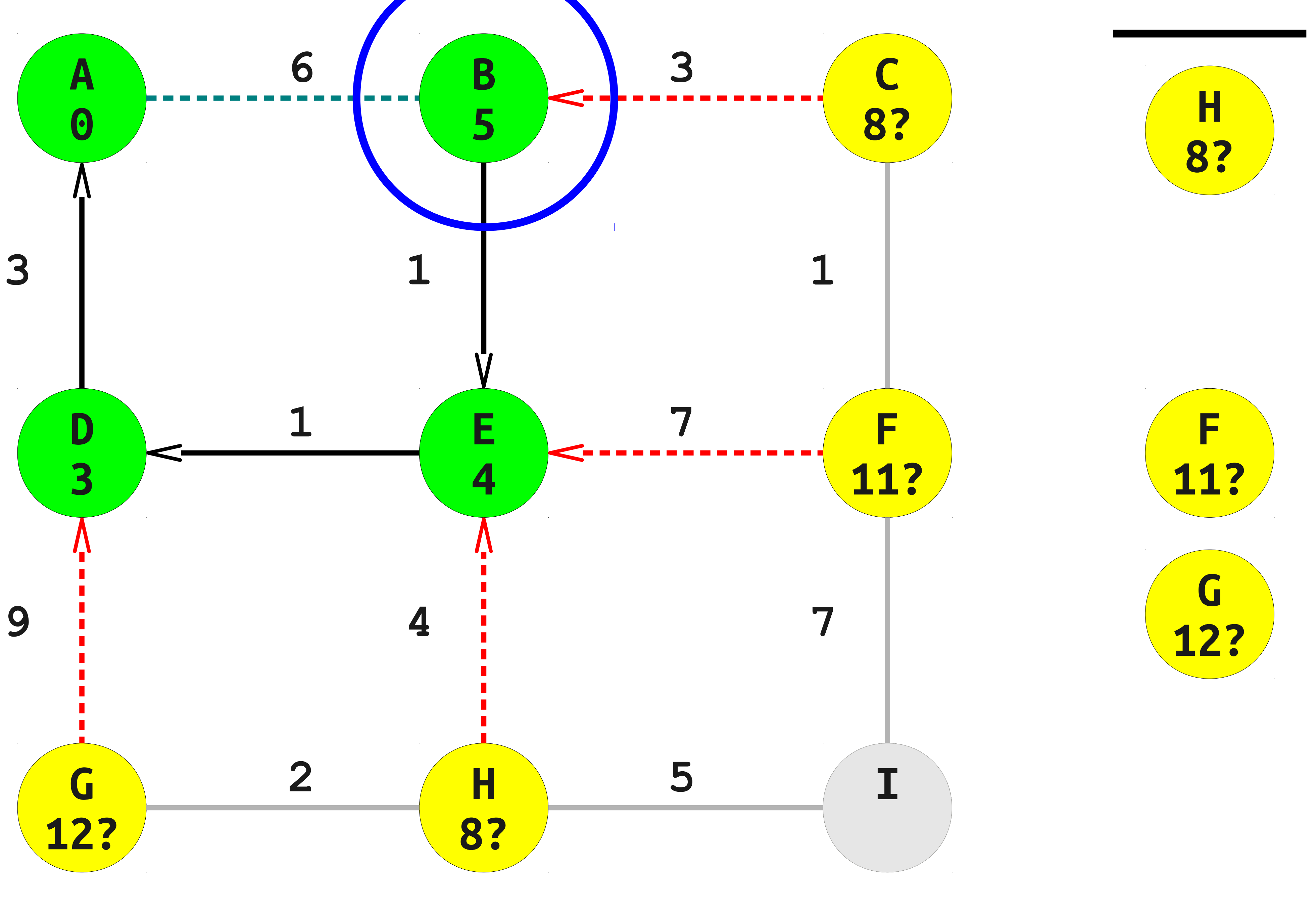


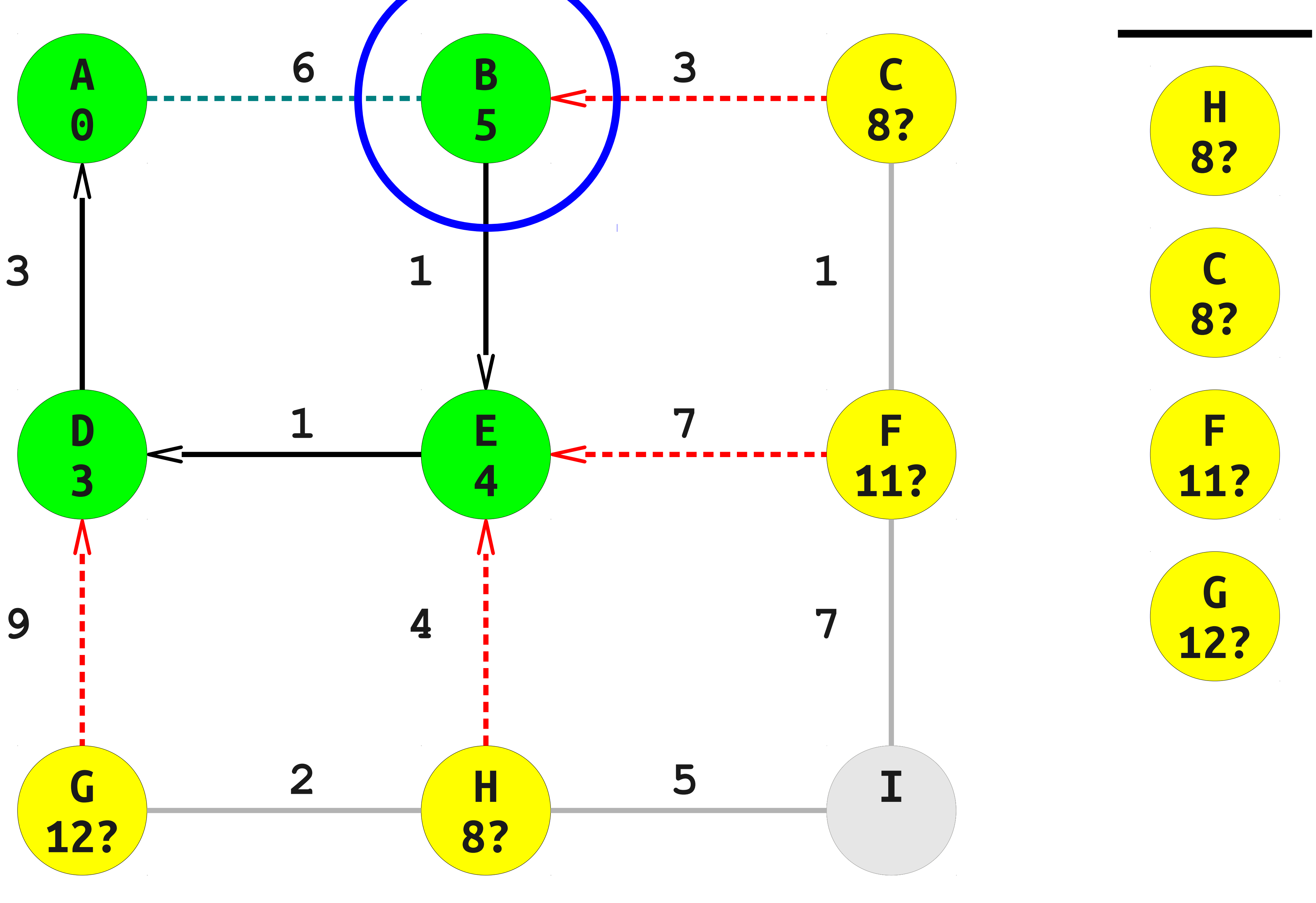
A
0
6
B
5?
3
C
H
8?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
G
12?
9
4
7
G
12?
2
H
8?
5
I



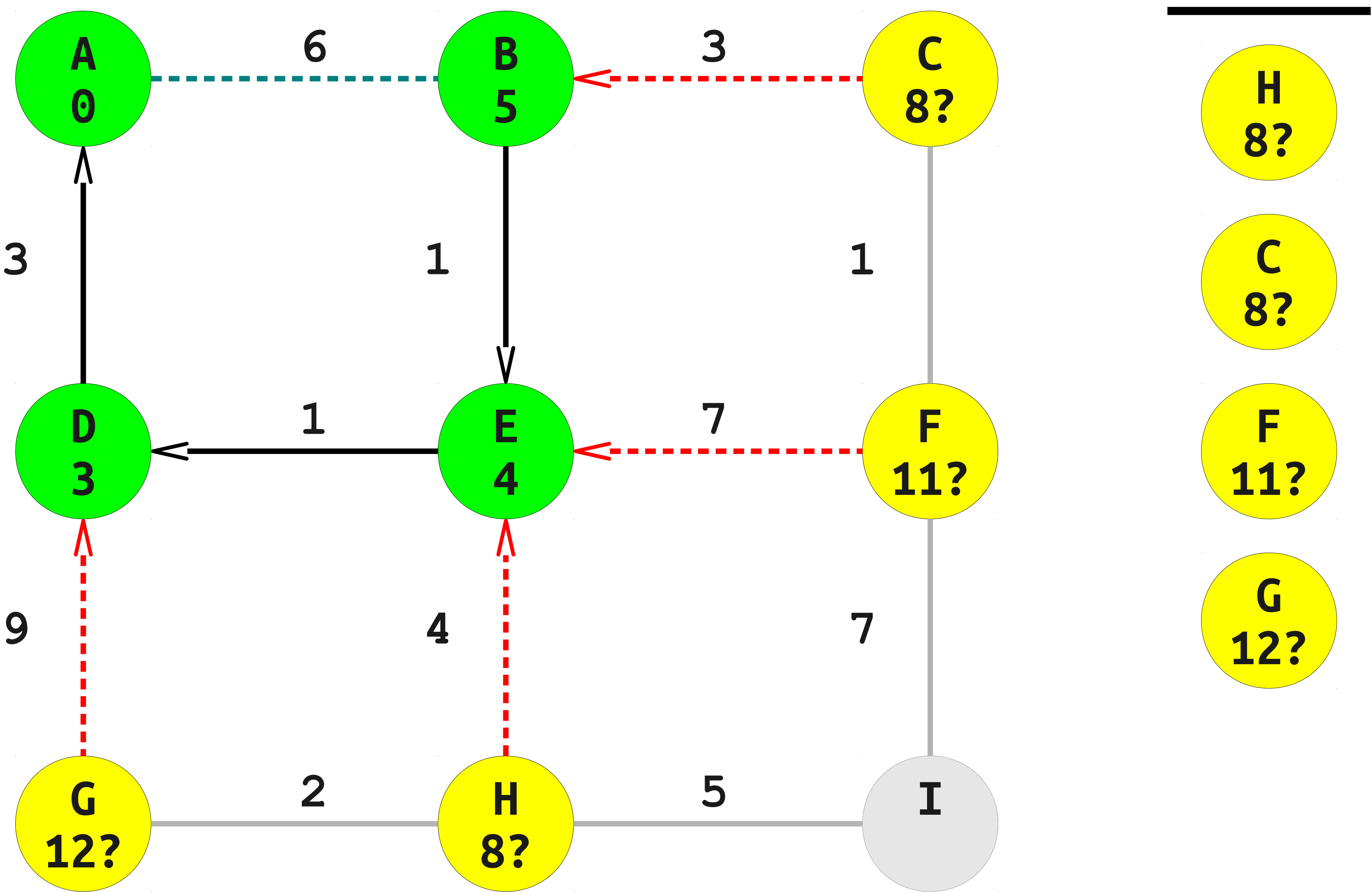
A
0
6
B
5
3
C
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
H
8?
F
11?
G
12?


A
0
6
B
5
3
C
8?
H
8?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
G
12?
9
4
7
G
12?
2
H
8?
5
I

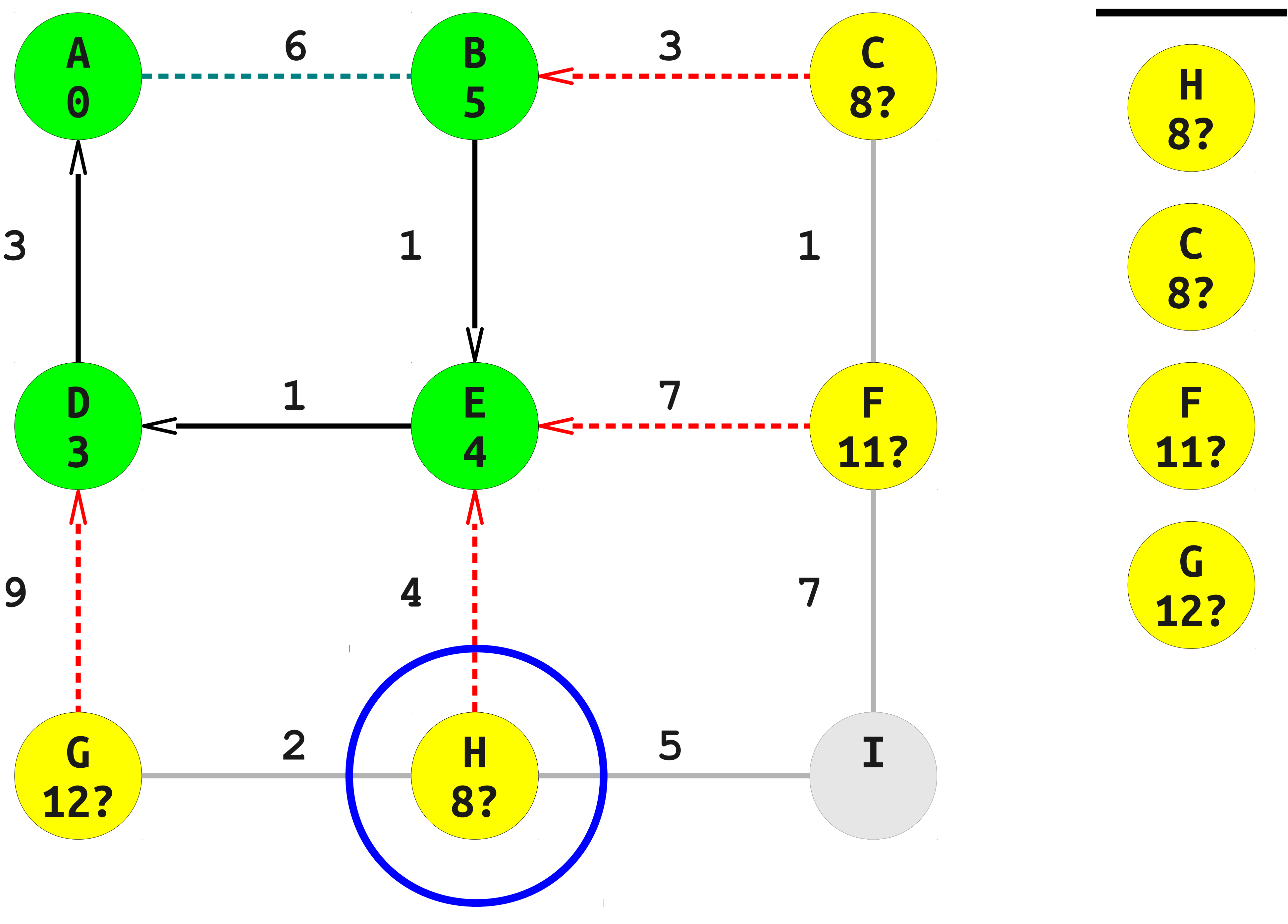

A
0
6
B
5
3
C
8?
H
8?
3
1
1
D
3
1
E
4
7
F
11?
F
11?
G
12?
9
4
7
G
12?
2
H
8?
5
I

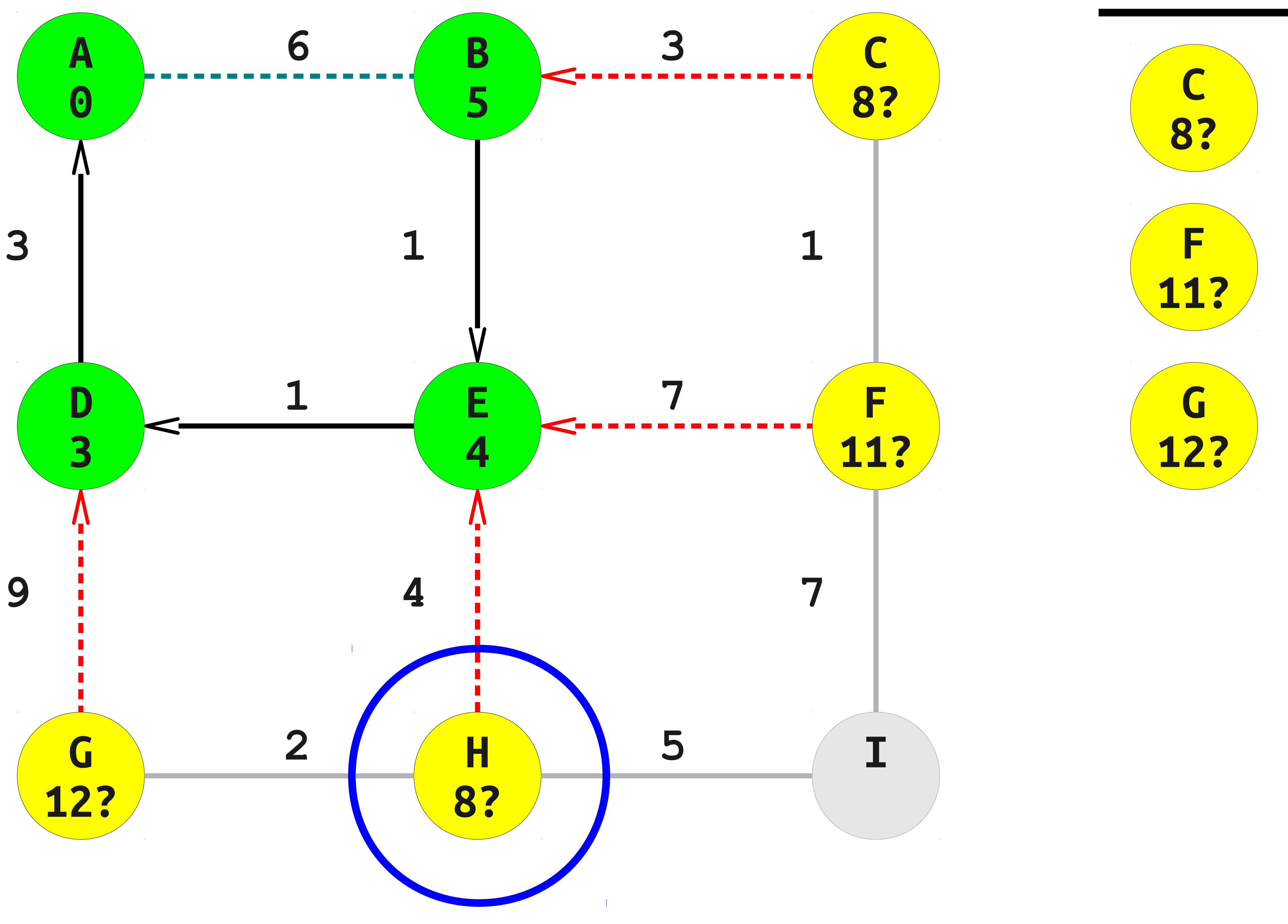


A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
H
8?
C
8?
F
11?
G
12?

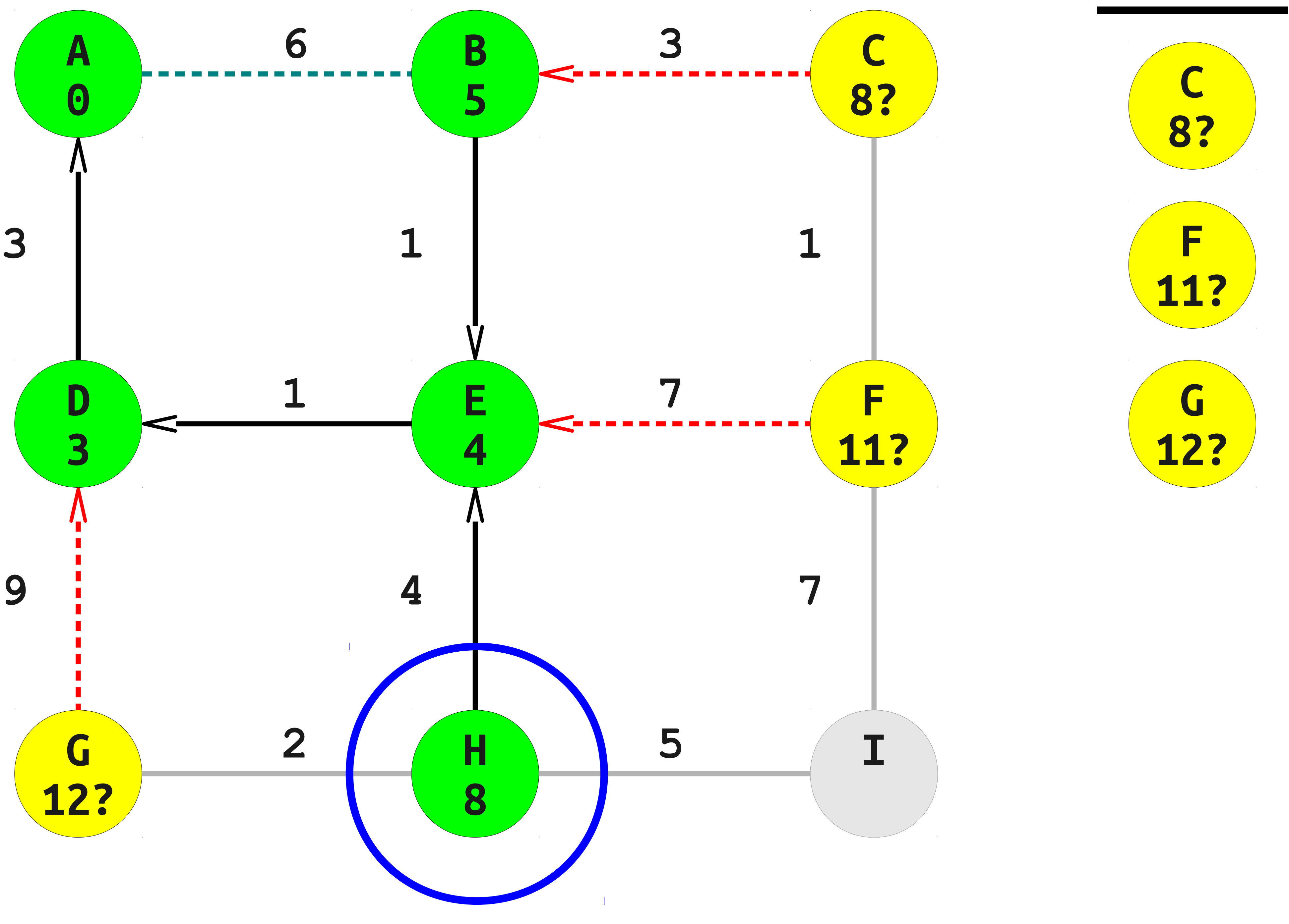


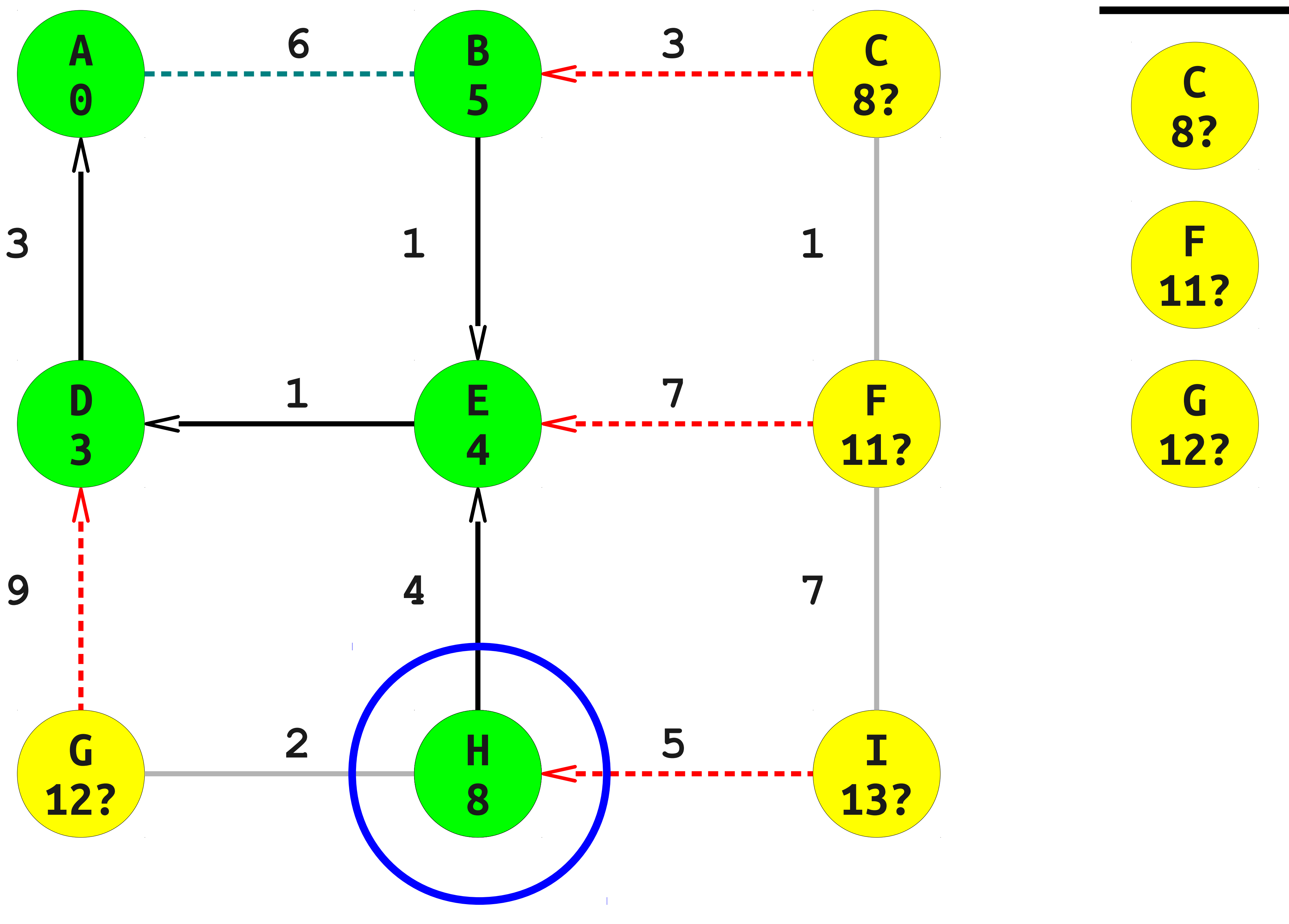
A
0
6
B
5
3
C
8?
H
8?
3
1
1
C
8?
D
3
1
E
4
7
F
11?
F
11?
9
4
7
G
12?
G
12?
2
H
8?
5
I


A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8?
5
I
H
8?
C
8?
F
11?
G
12?

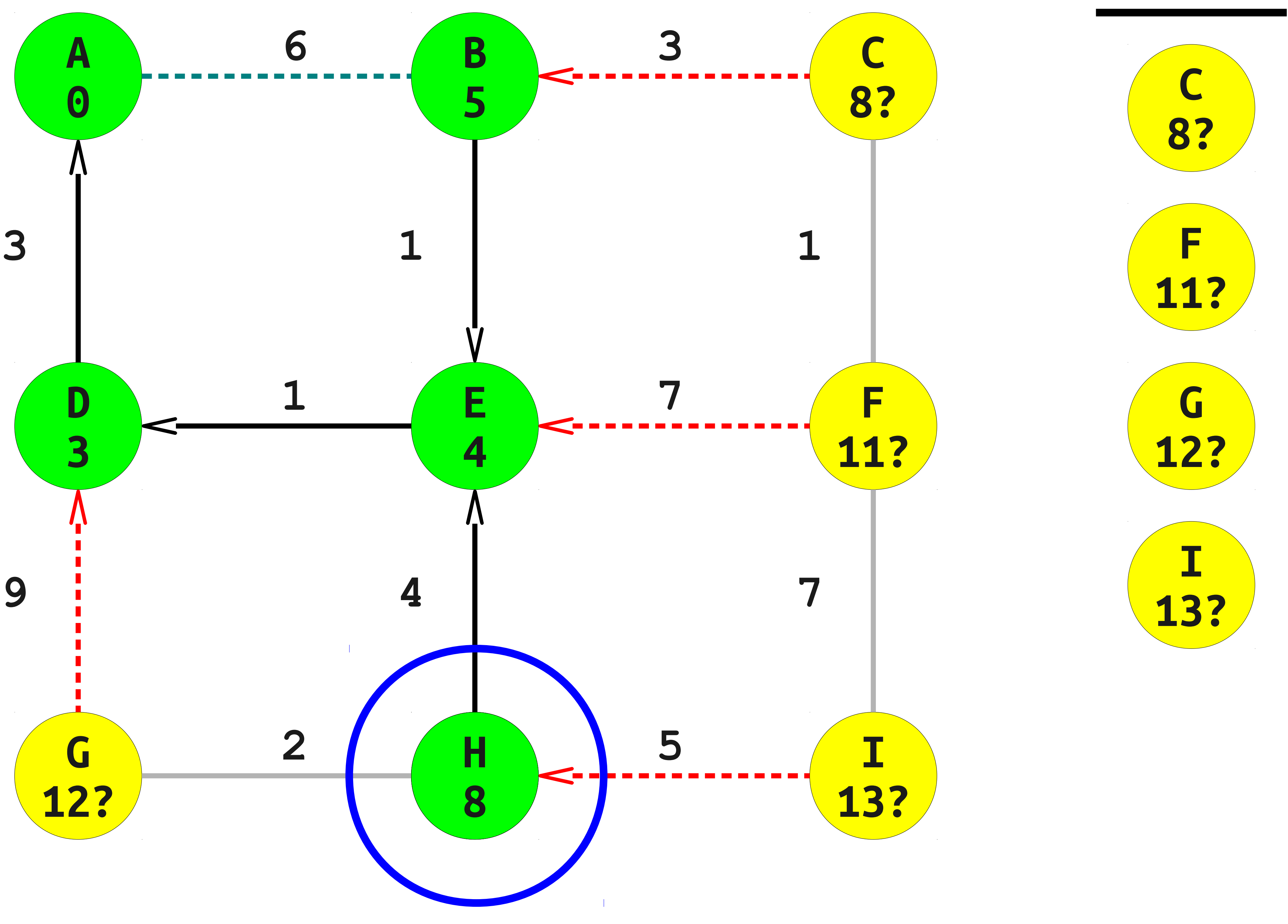


A
0
6
B
5
3
C
8?
C
8?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
G
12?
9
4
7
G
12?
2
H
8?
5
I

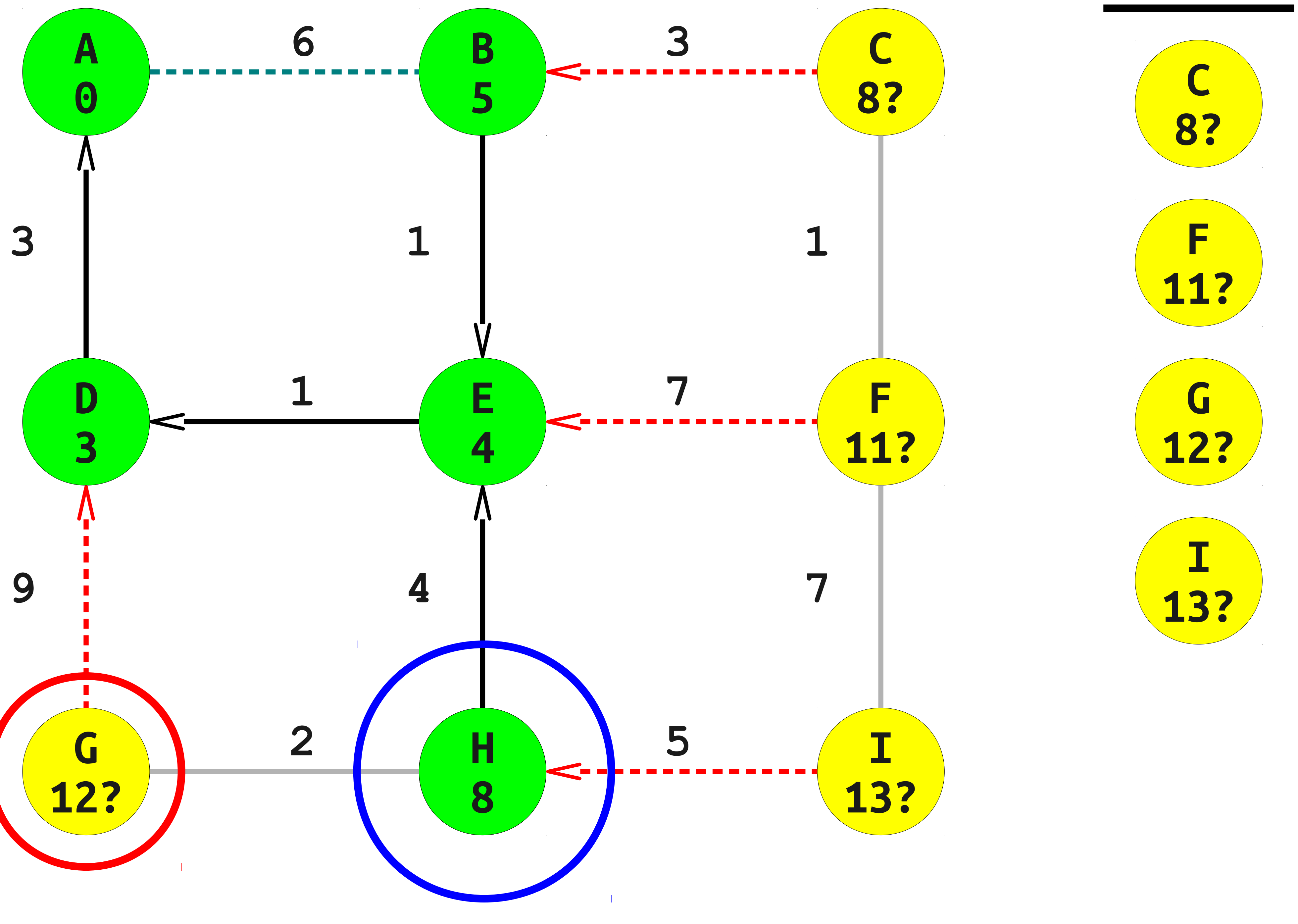

A
0
B
5
C
8?
D
3
E
4
F
11?
G
12?
H
8
I
6
3
3
1
1
1
7
9
4
7
2
5
C
8?
F
11?
G
12?

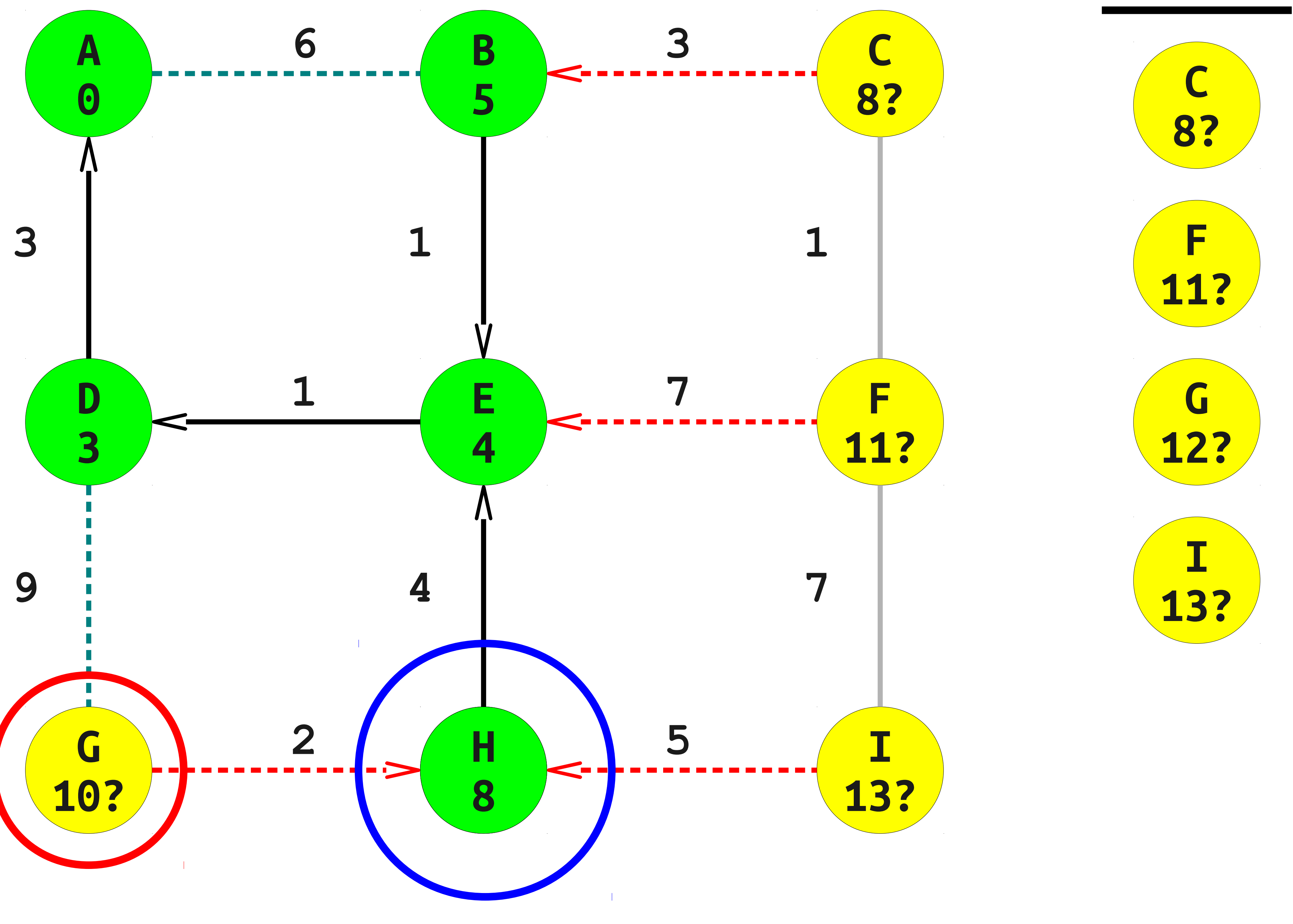


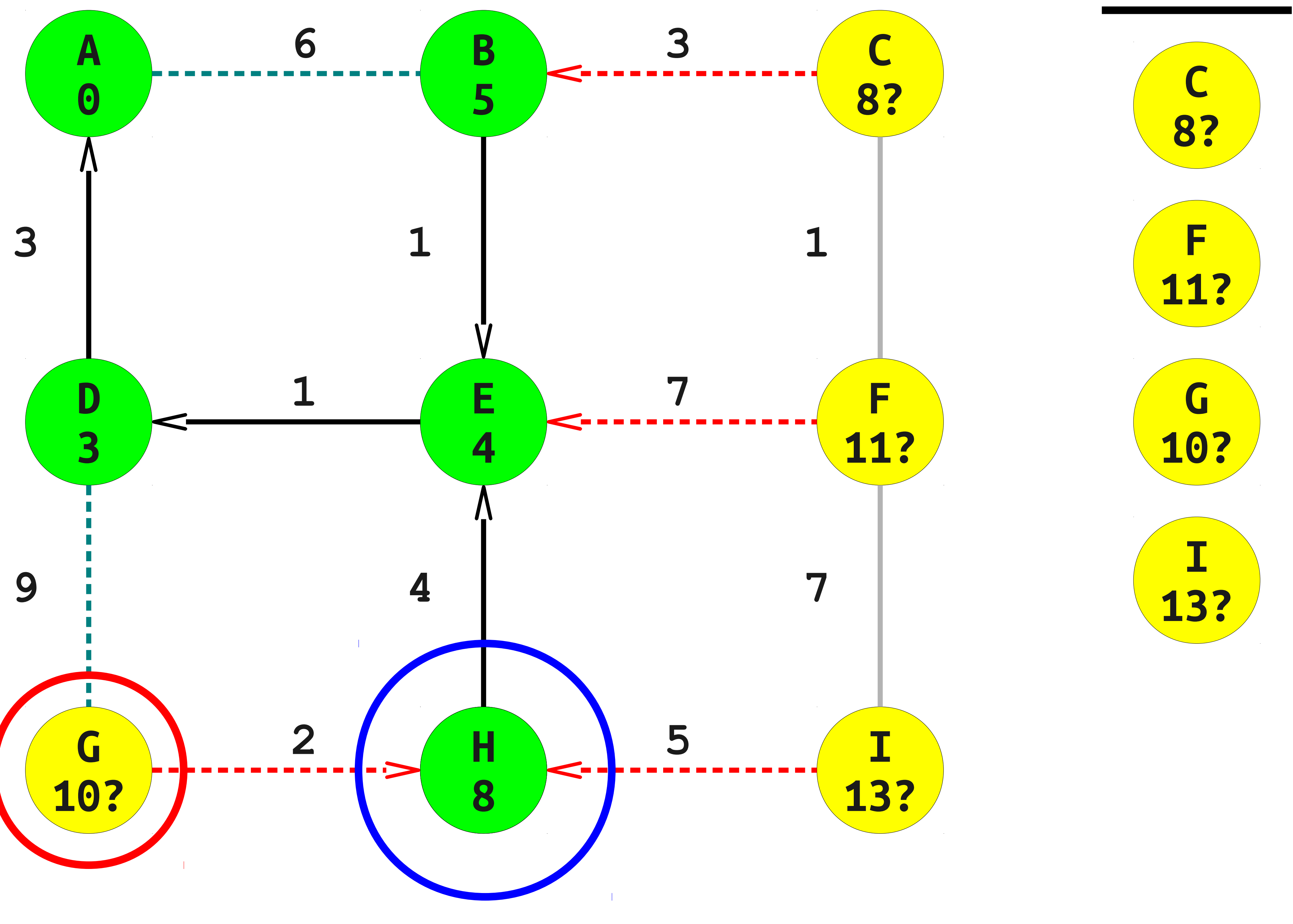
A
0
B
5
C
8?
D
3
E
4
F
11?
G
12?
H
8
I
13?
6
3
3
1
1
1
7
9
4
7
2
5
C
8?
F
11?
G
12?

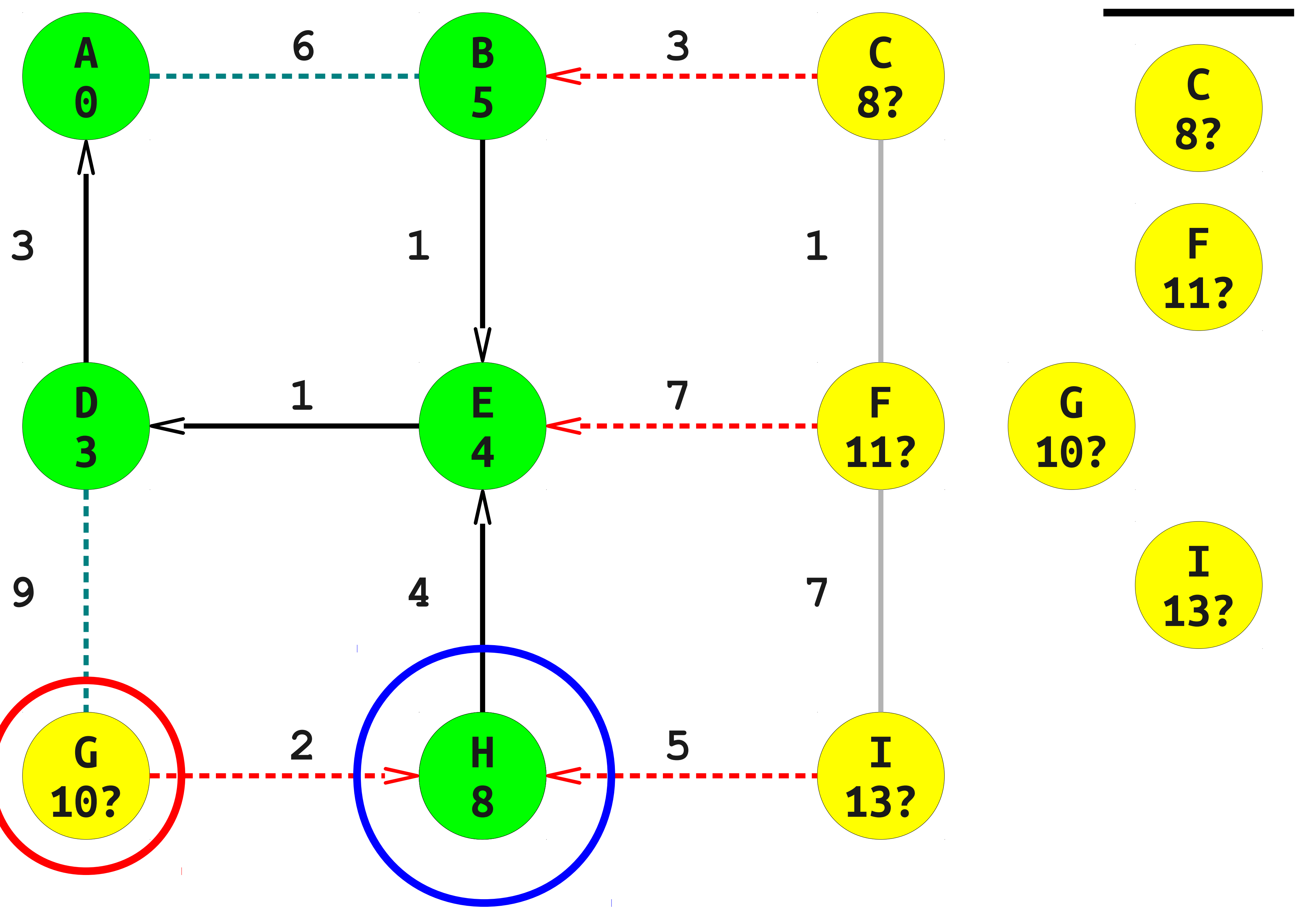


A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8
5
I
13?
C
8?
F
11?
G
12?
I
13?

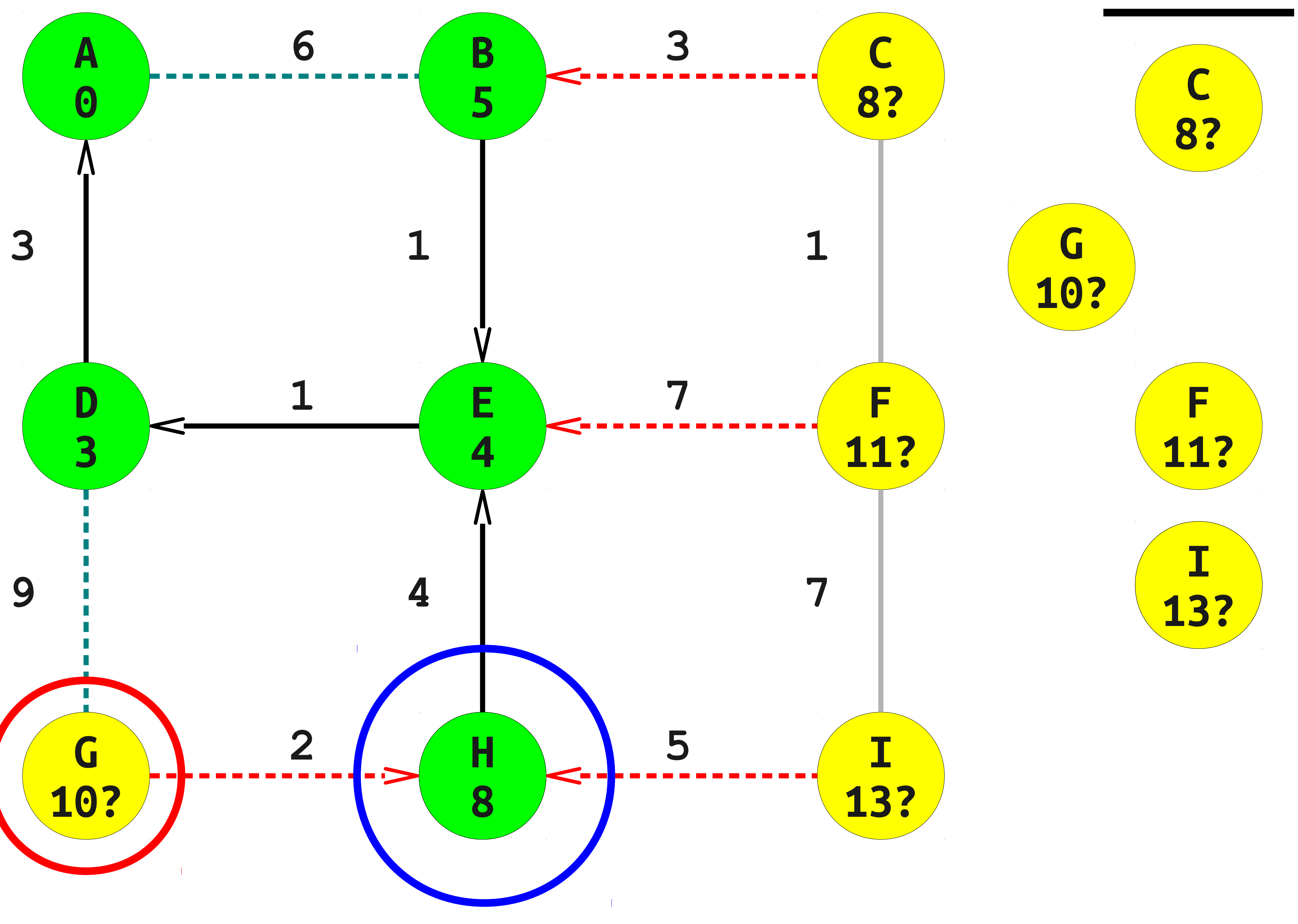

A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
12?
2
H
8
5
I
13?
C
8?
F
11?
G
12?
I
13?

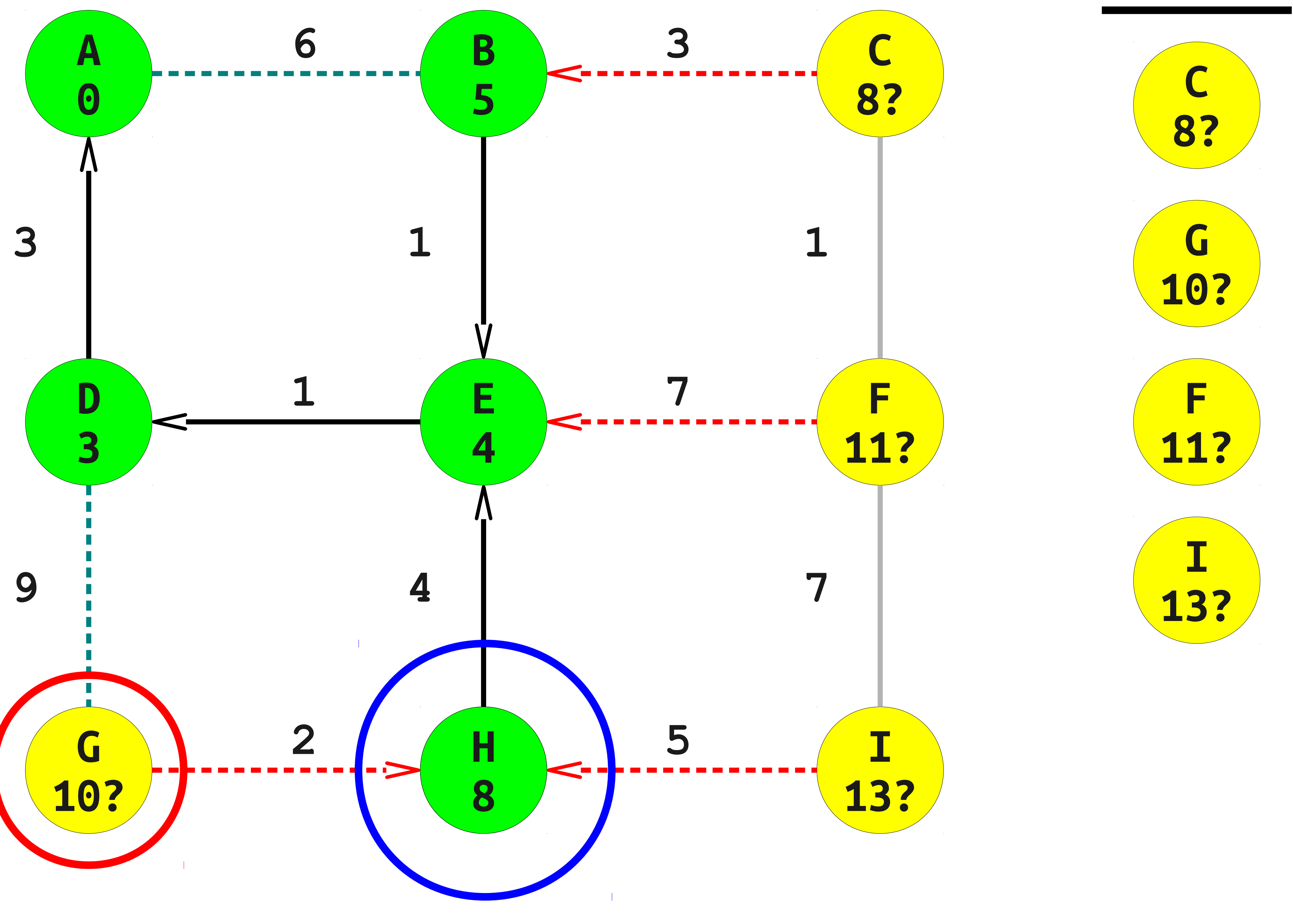

A
0
6
B
5
3
C
8?
C
8?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
G
12?
9
4
7
I
13?
G
10?
2
H
8
5
I
13?

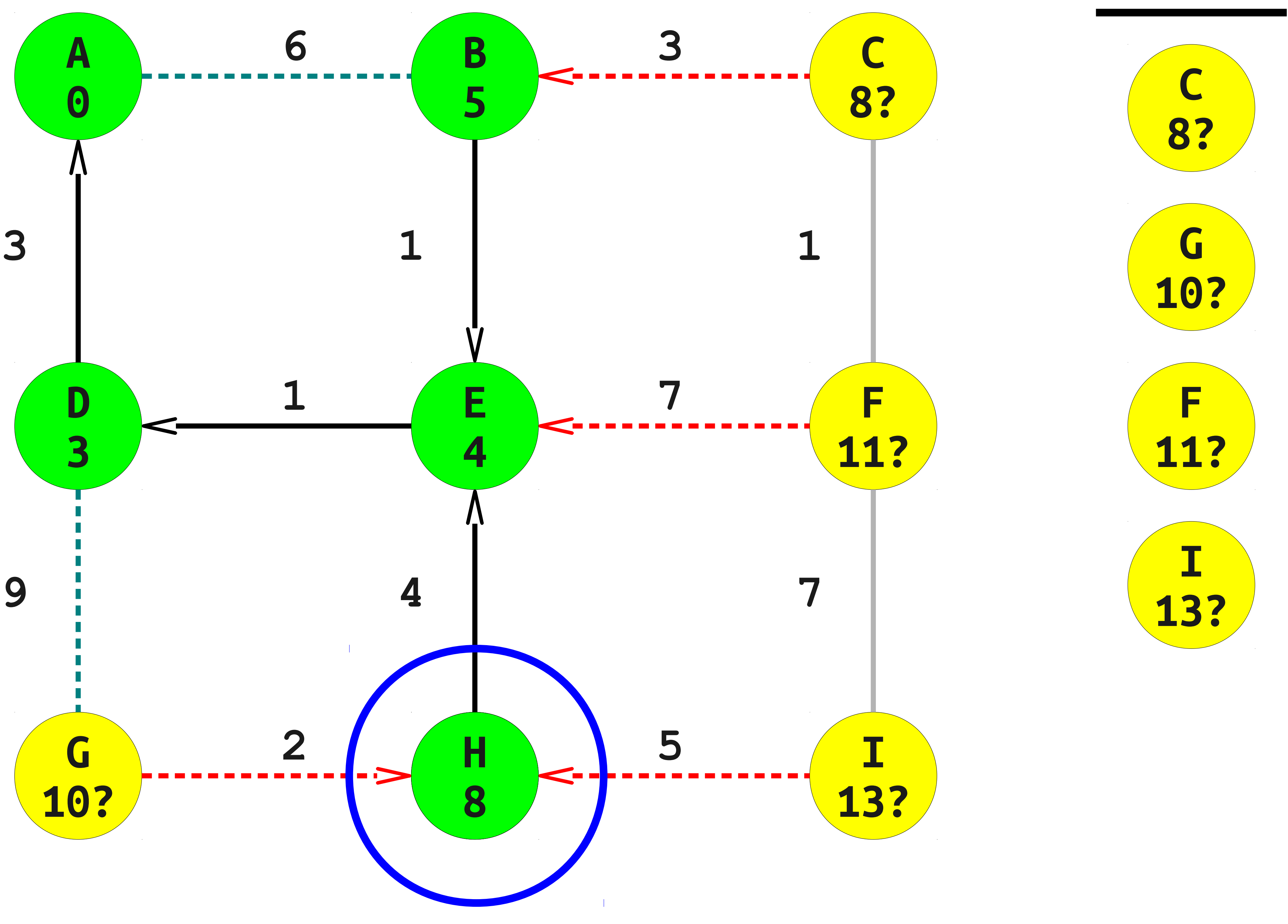

A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
10?
2
H
8
5
I
13?
C
8?
F
11?
G
10?
I
13?

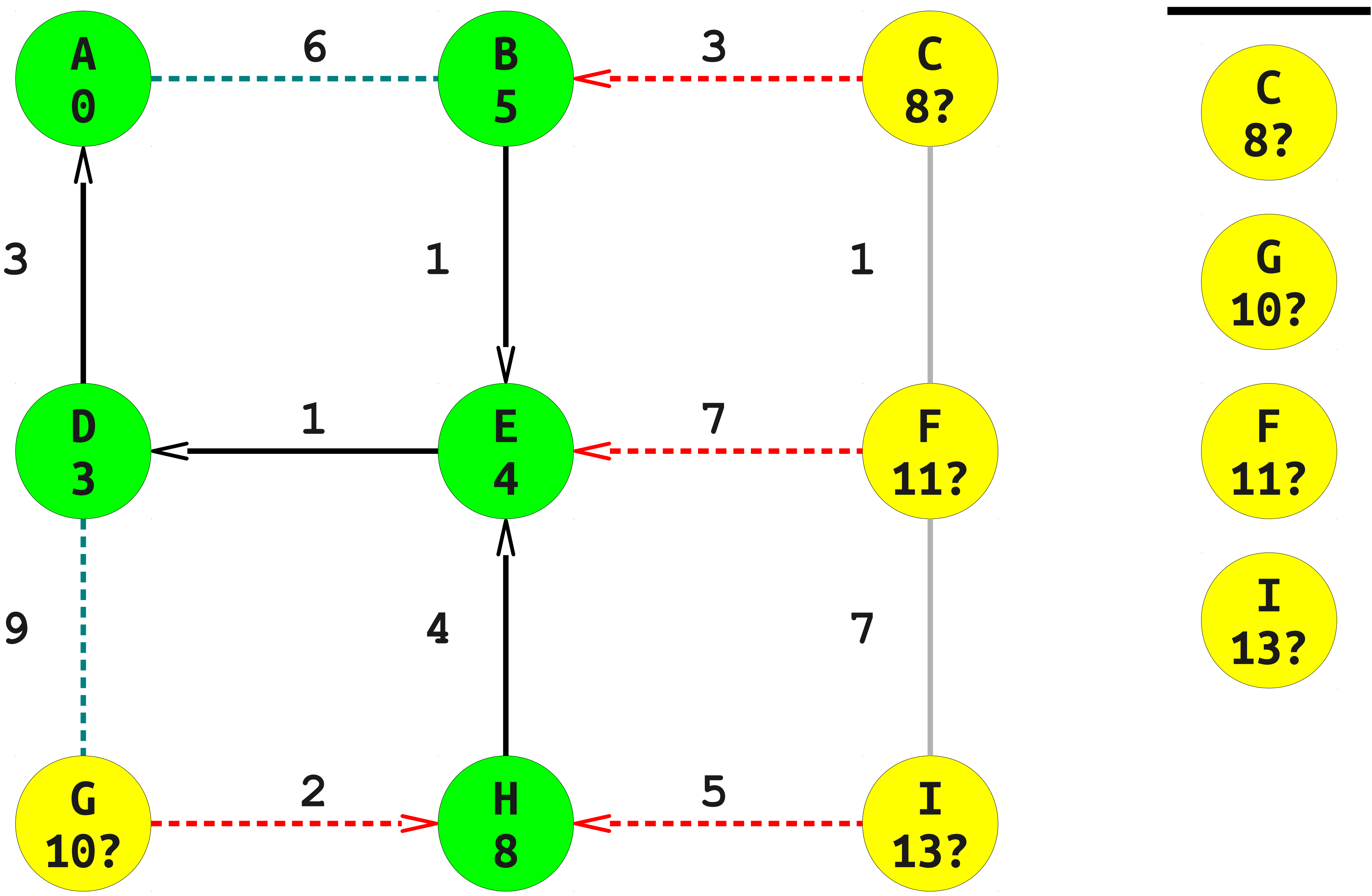


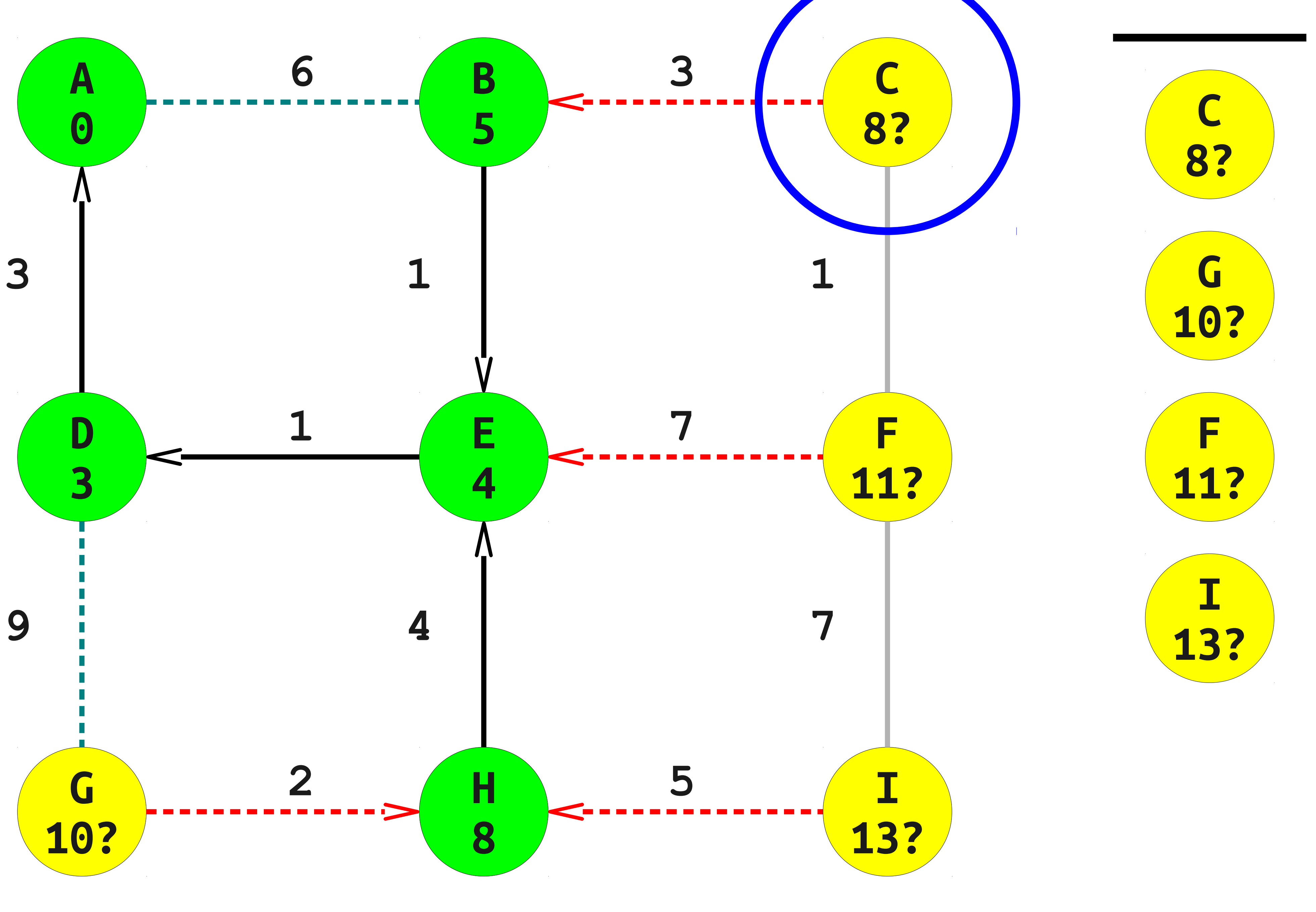
A
0
6
B
5
3
C
8?
C
8?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
G
10?
9
4
7
I
13?
G
10?
2
H
8
5
I
13?


A
0
6
B
5
3
C
8?
C
8?
3
1
1
G
10?
D
3
1
E
4
7
F
11?
F
11?
9
4
7
I
13?
G
10?
2
H
8
5
I
13?

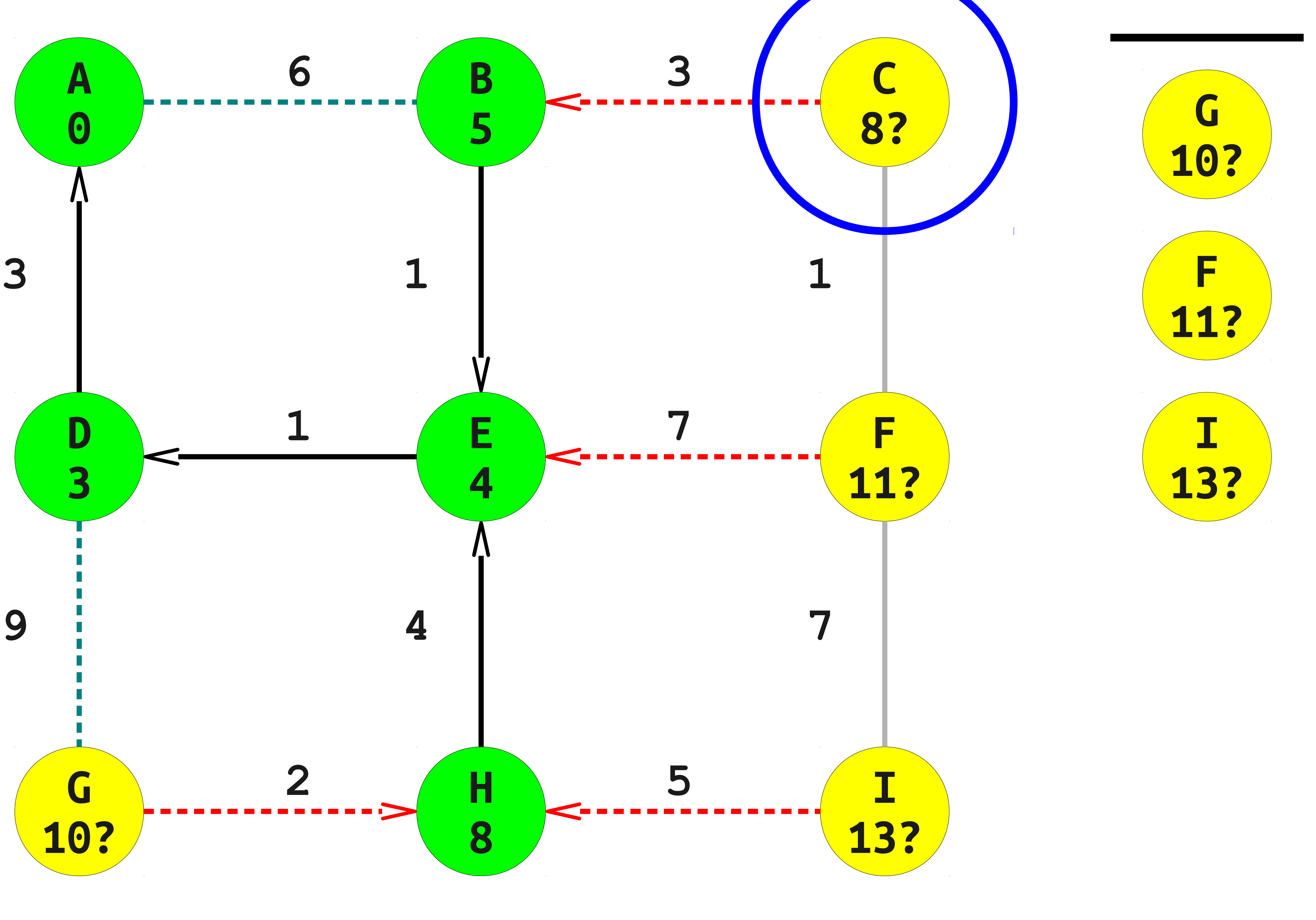

A
0
6
B
5
3
C
8?
C
8?
3
1
1
G
10?
D
3
1
E
4
7
F
11?
F
11?
9
4
7
I
13?
G
10?
2
H
8
5
I
13?

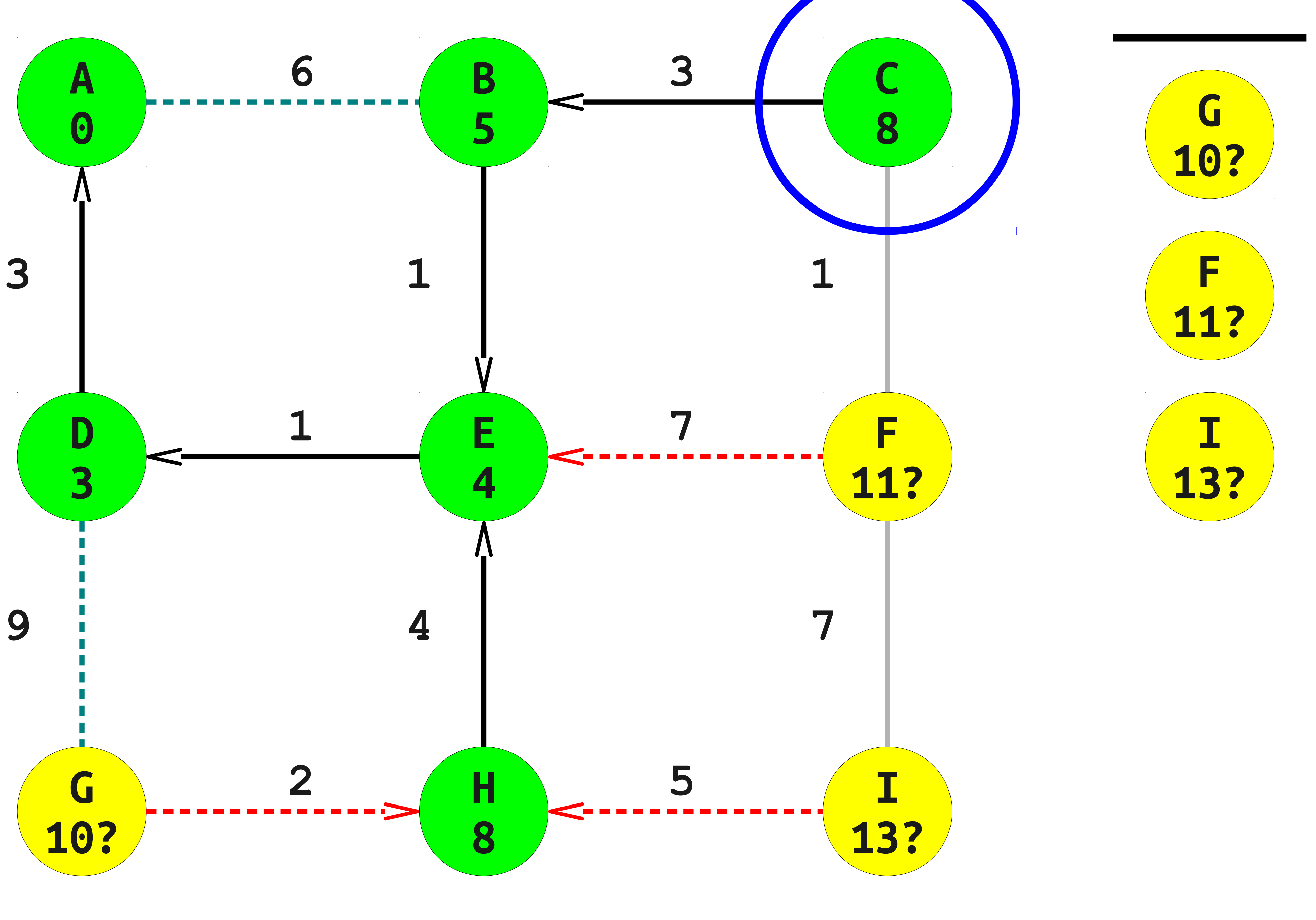

A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
10?
2
H
8
5
I
13?
C
8?
G
10?
F
11?
I
13?

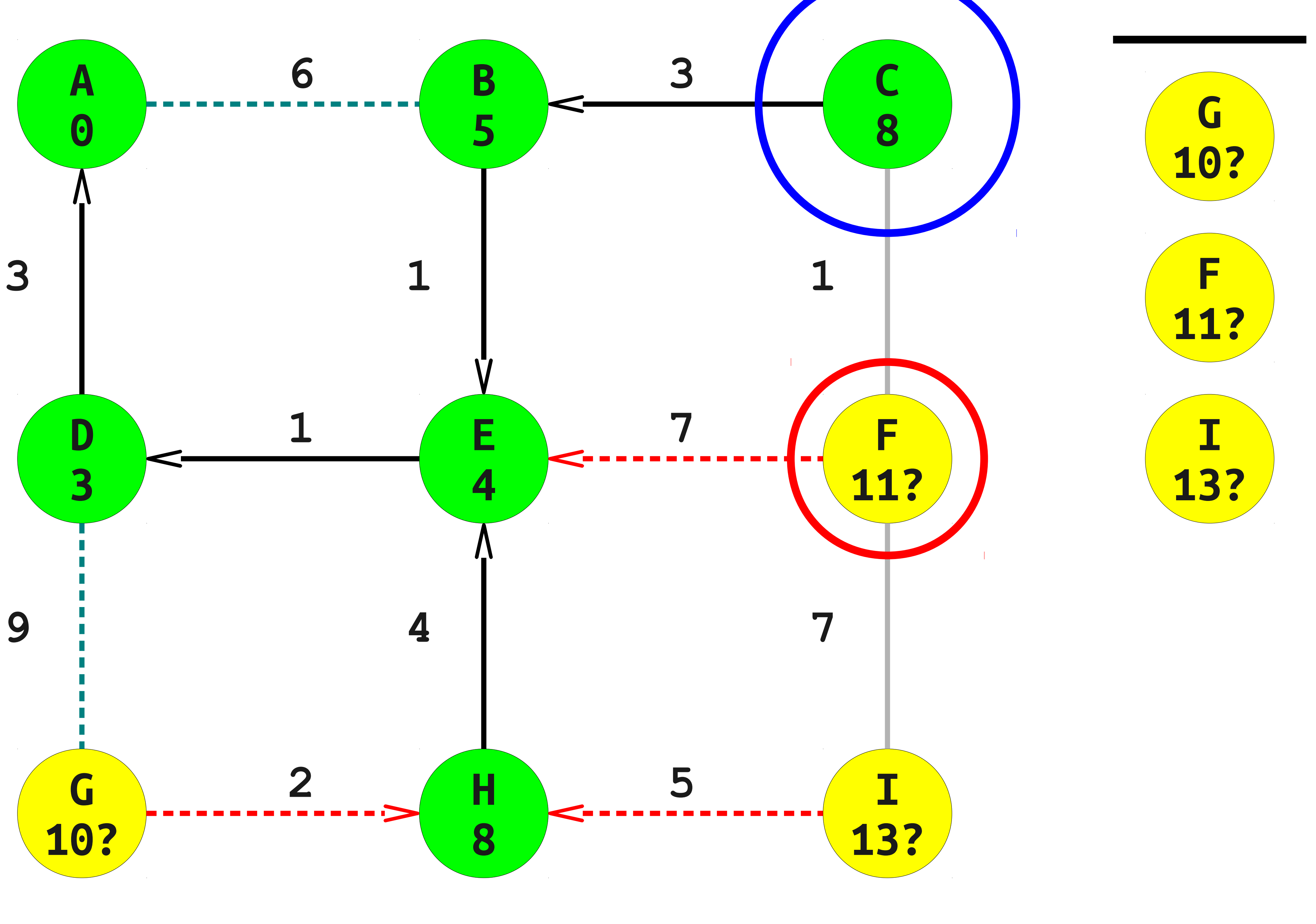

A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
10?
2
H
8
5
I
13?
C
8?
G
10?
F
11?
I
13?



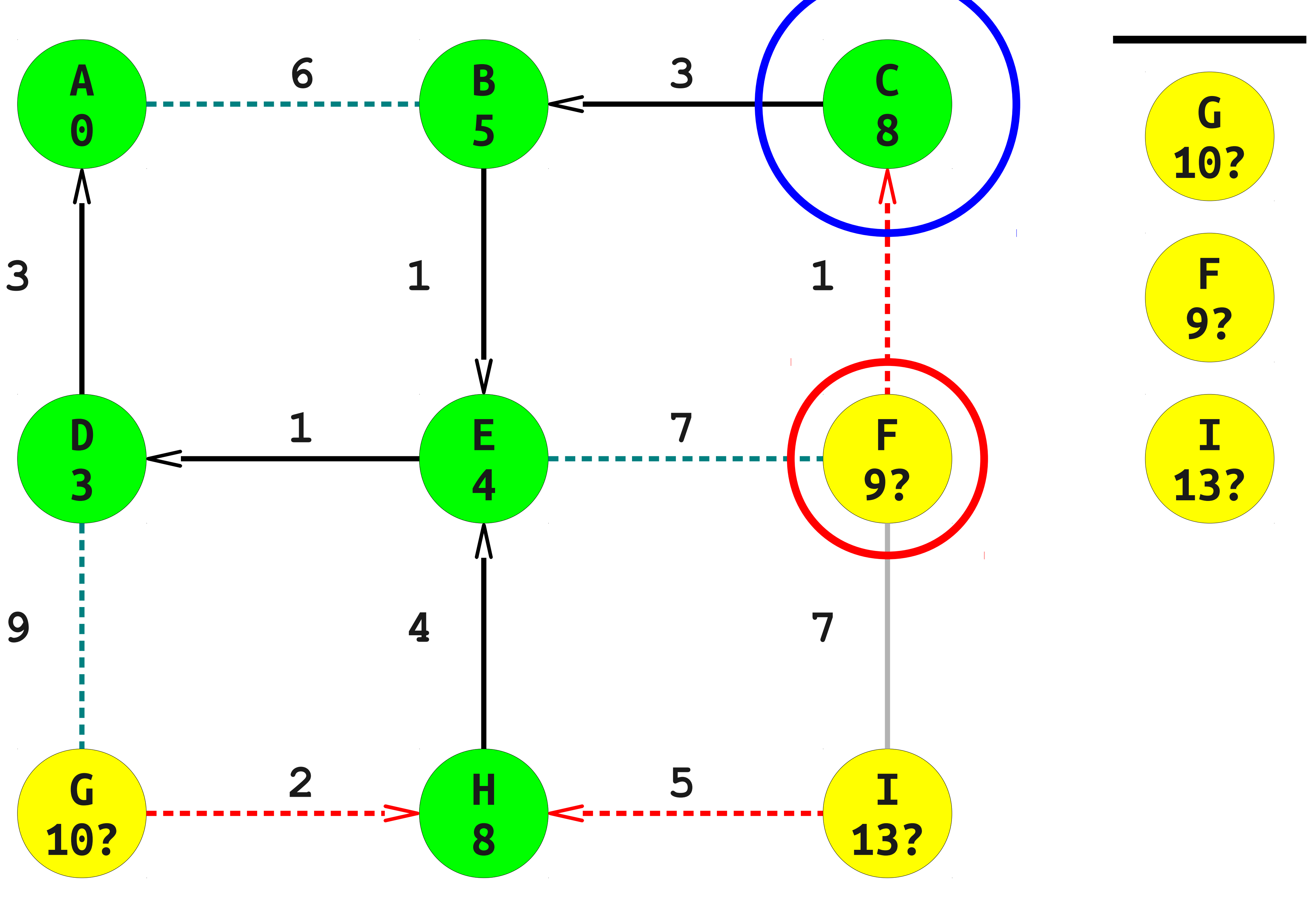
A
0
6
B
5
3
C
8?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
10?
2
H
8
5
I
13?
C
8?
G
10?
F
11?
I
13?


A
0
6
B
5
3
C
8?
G
10?
F
11?
I
13?
3
1
1
D
3
1
E
4
7
F
11?
9
4
7
G
10?
2
H
8
5
I
13?

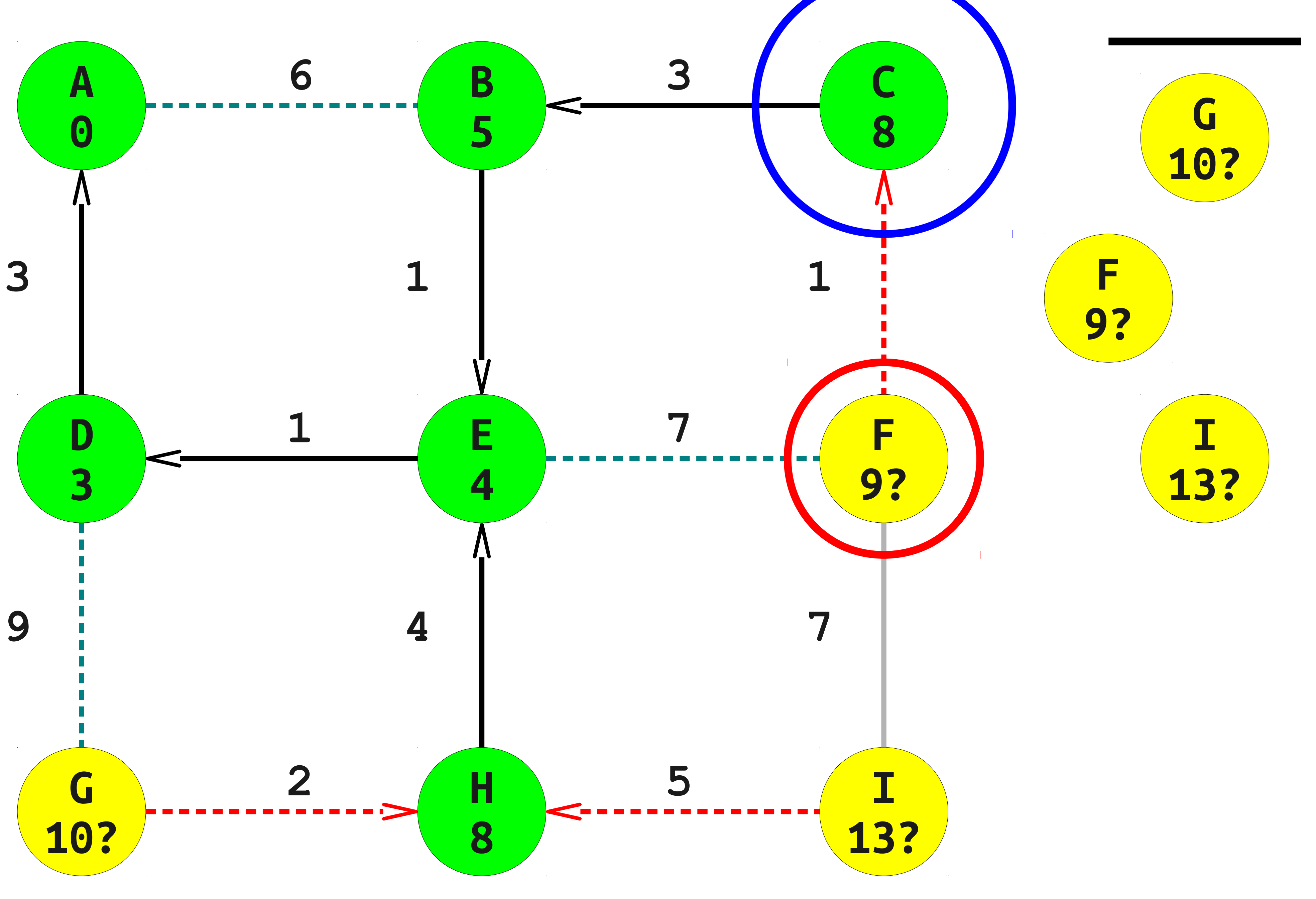

A
0
6
B
5
3
C
8
G
10?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?



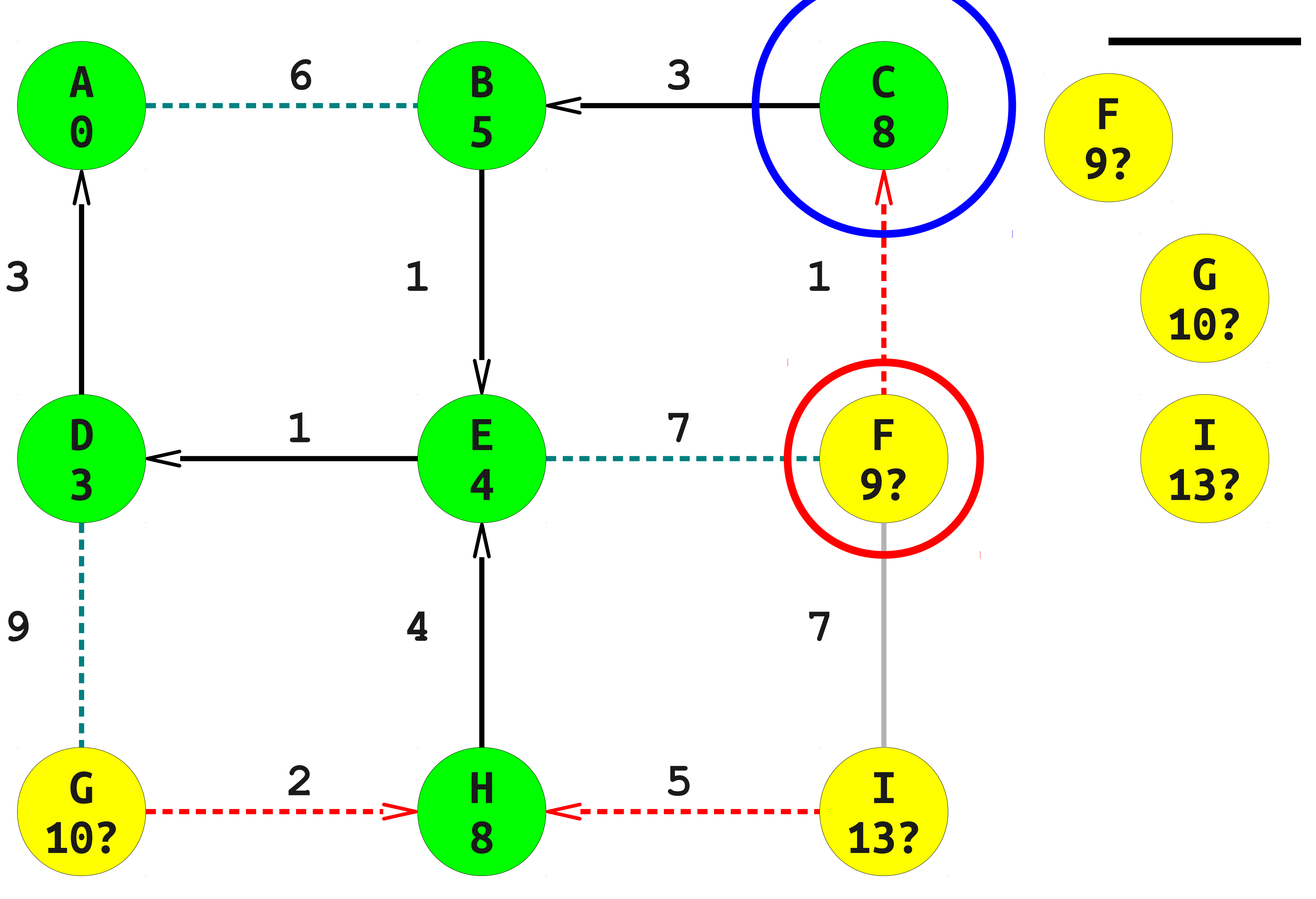
A
0
6
B
5
3
C
8
G
10?
3
1
1
F
11?
D
3
1
E
4
7
F
11?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

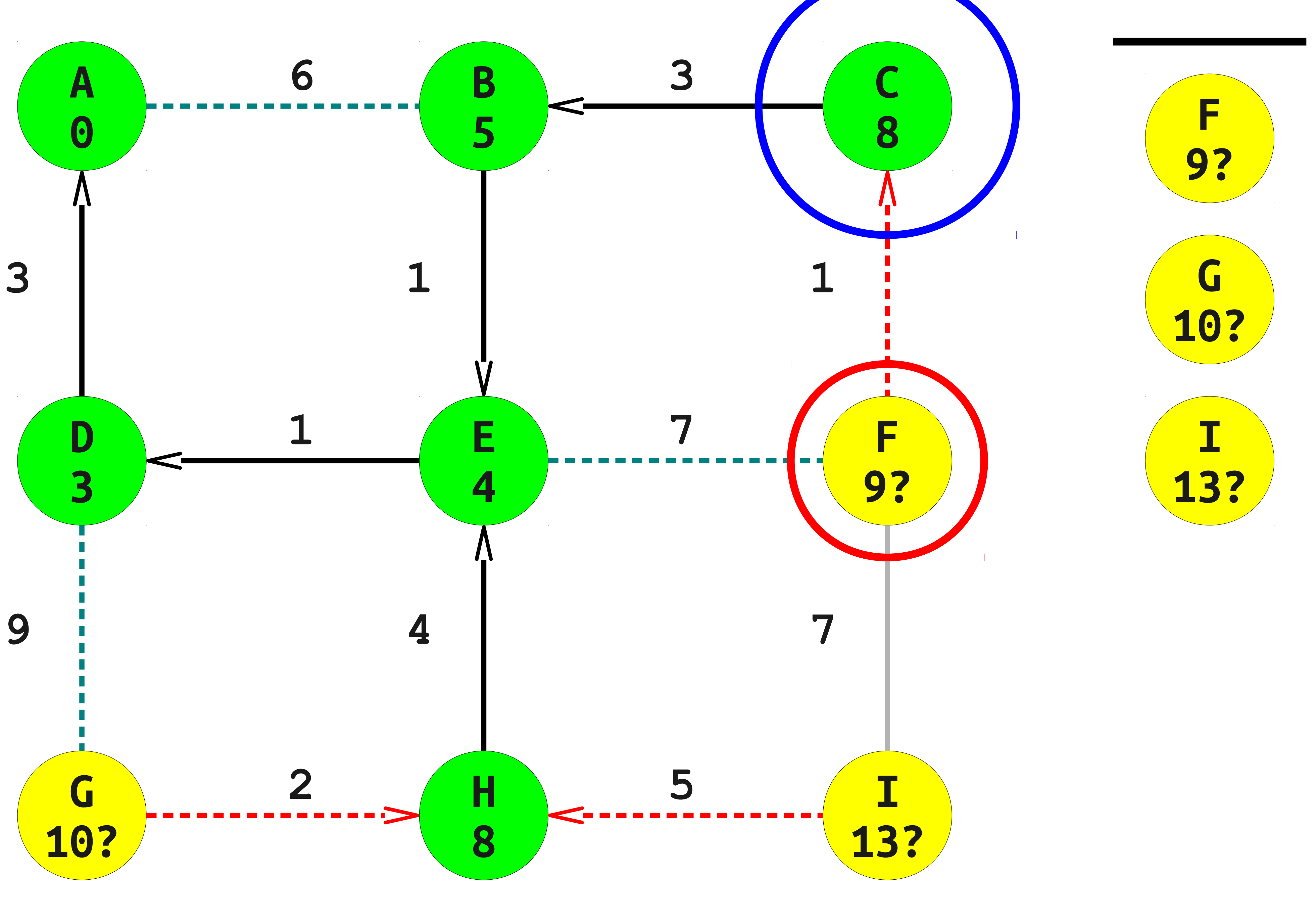


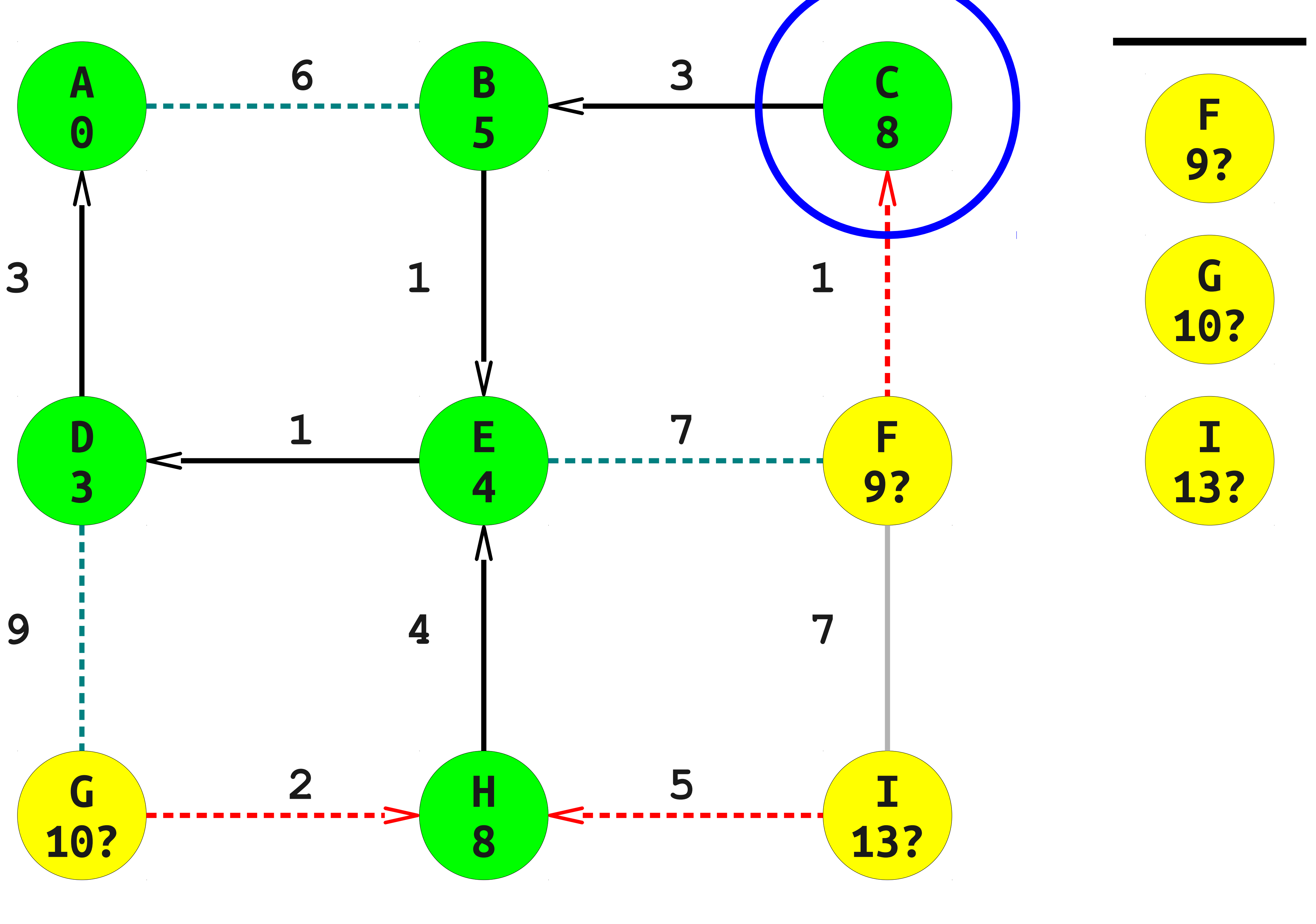
A
0
6
B
5
3
C
8
G
10?
3
1
1
F
9?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

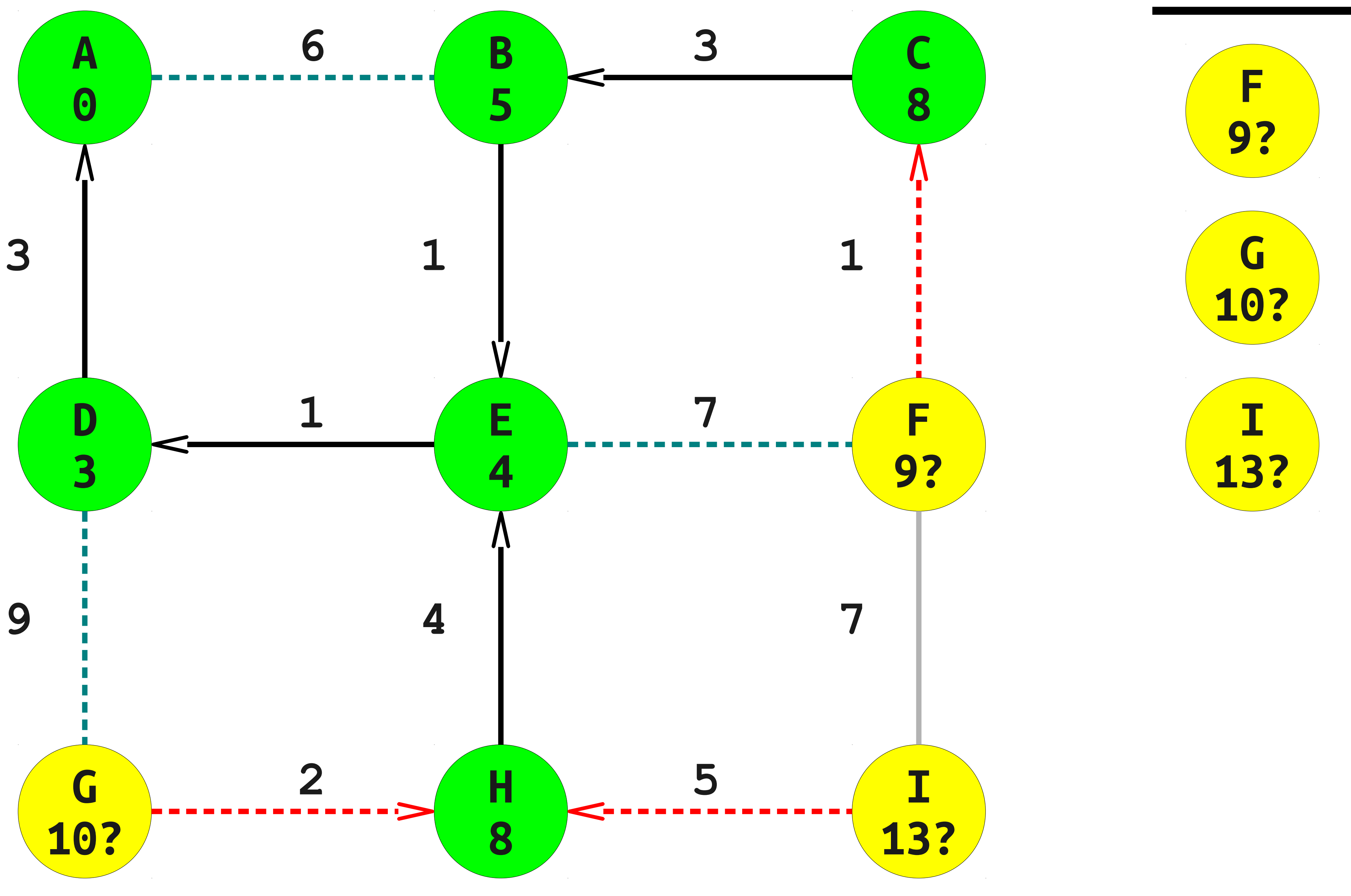


A
0
6
B
5
3
C
8
G
10?
3
1
1
F
9?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

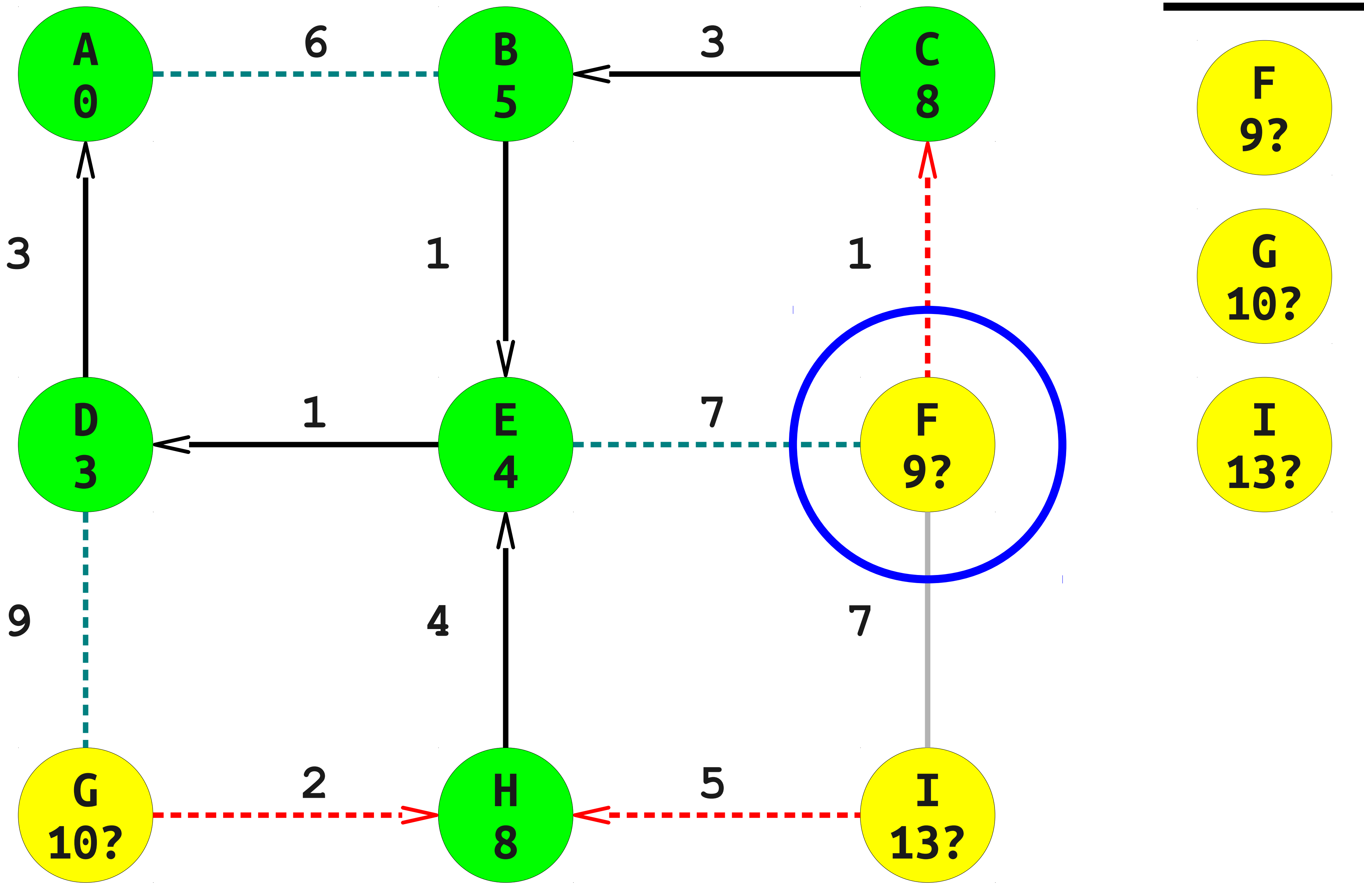

A
0
6
B
5
3
C
8
F
9?
3
1
1
G
10?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?


A
0
6
B
5
3
C
8
F
9?
3
1
1
G
10?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

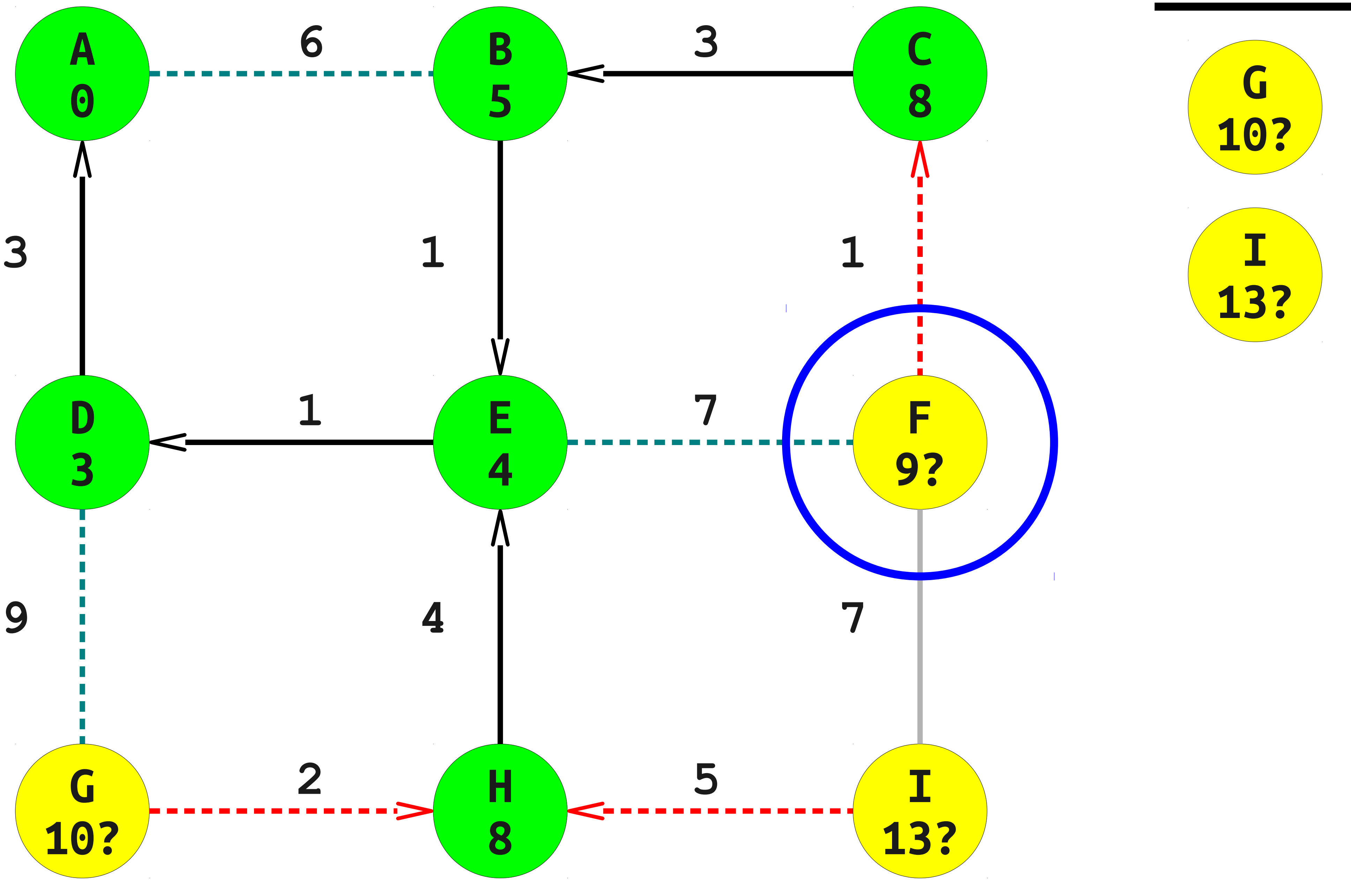

A
0
6
B
5
3
C
8
F
9?
3
1
1
G
10?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?



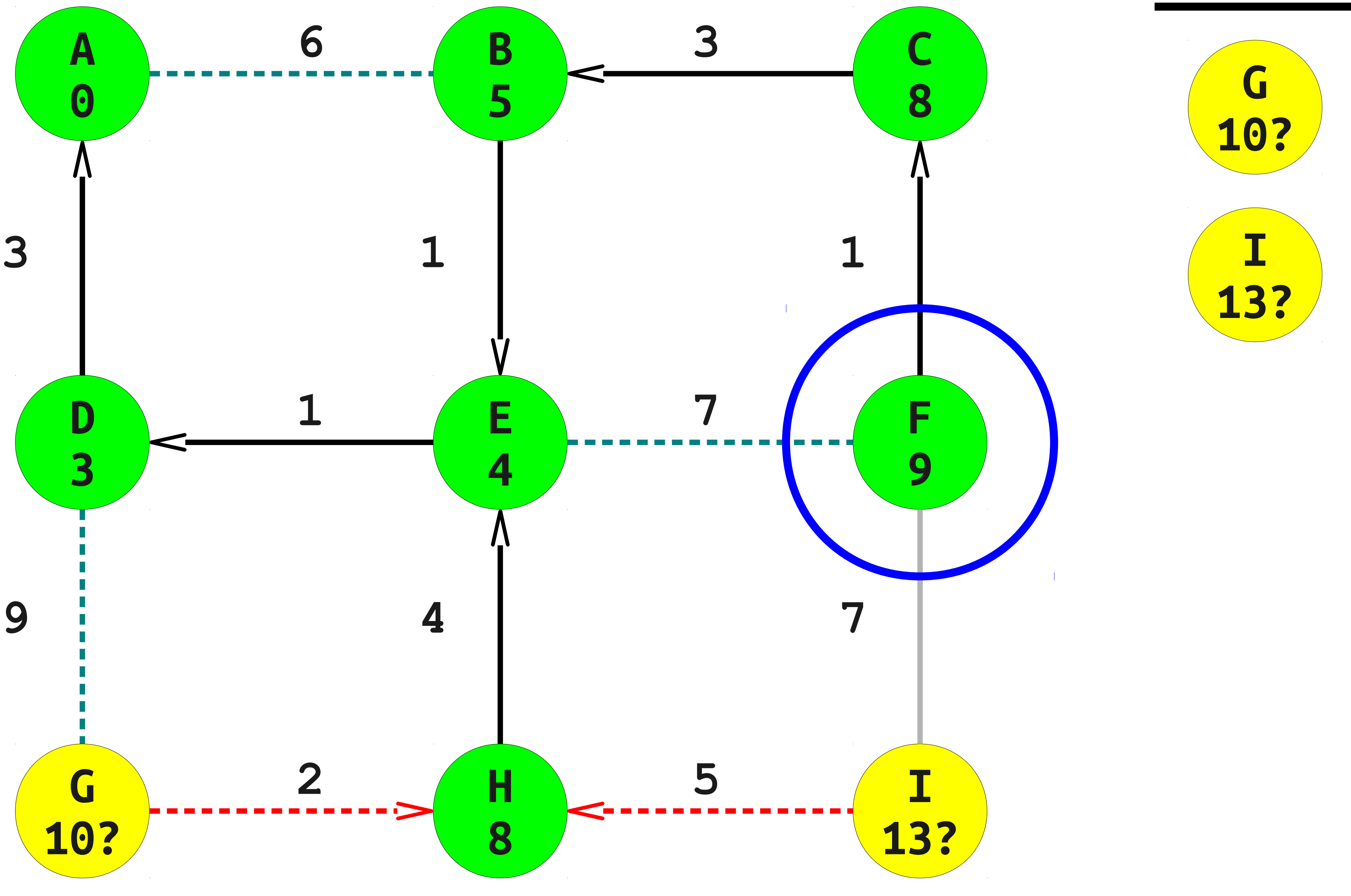
A
0
6
B
5
3
C
8
F
9?
3
1
1
G
10?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

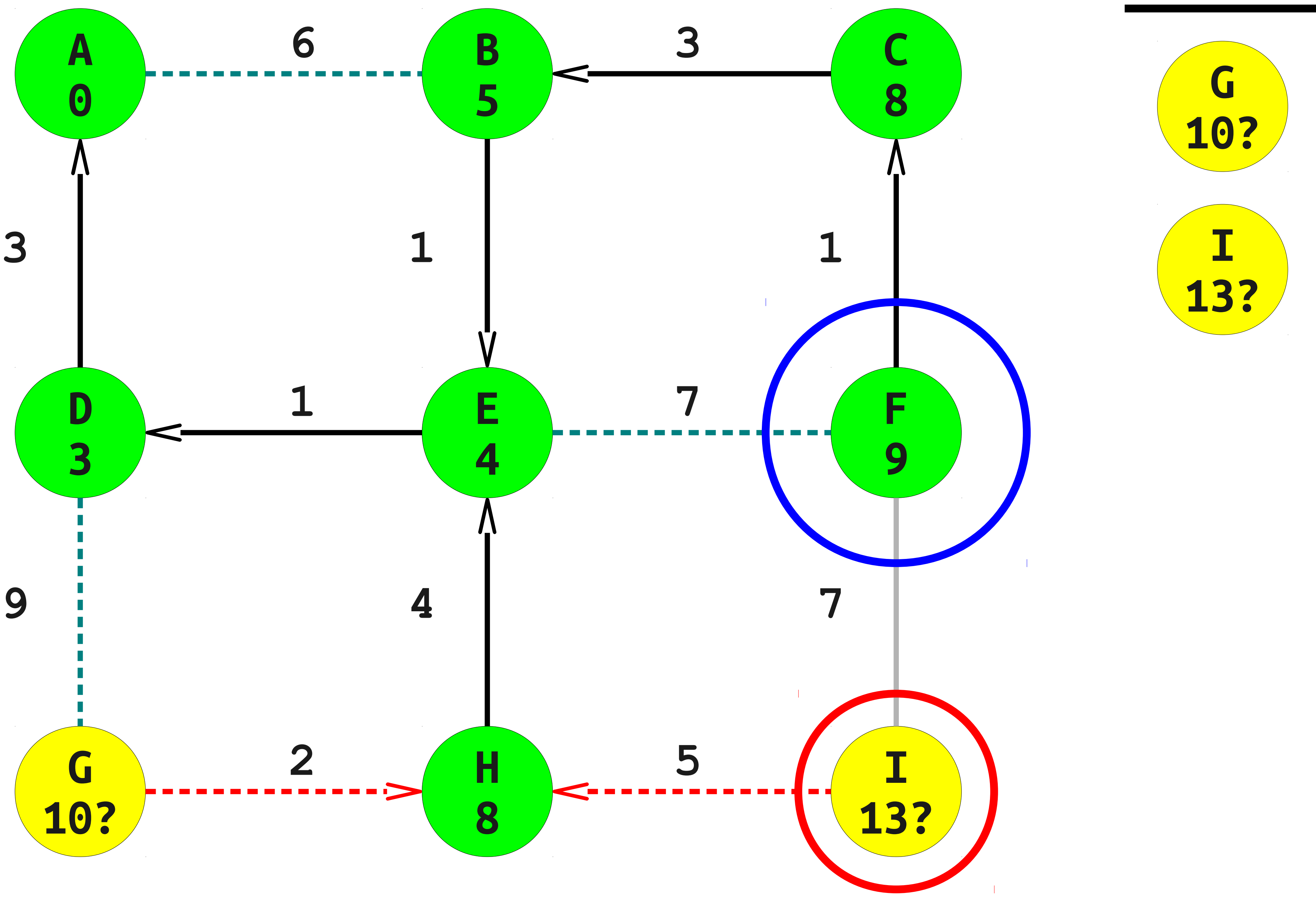


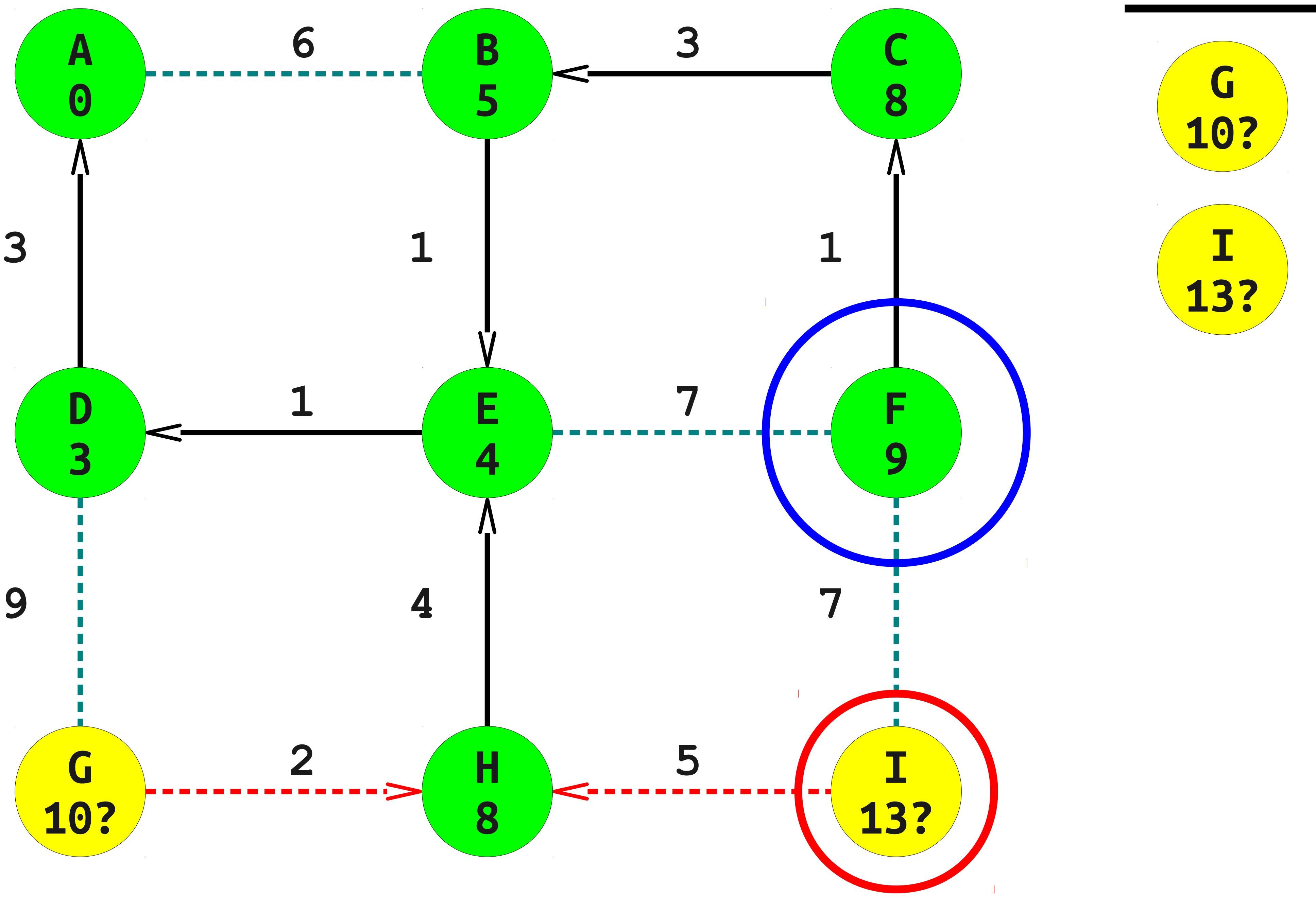
A
0
6
B
5
3
C
8
F
9?
3
1
1
G
10?
D
3
1
E
4
7
F
9?
I
13?
9
4
7
G
10?
2
H
8
5
I
13?

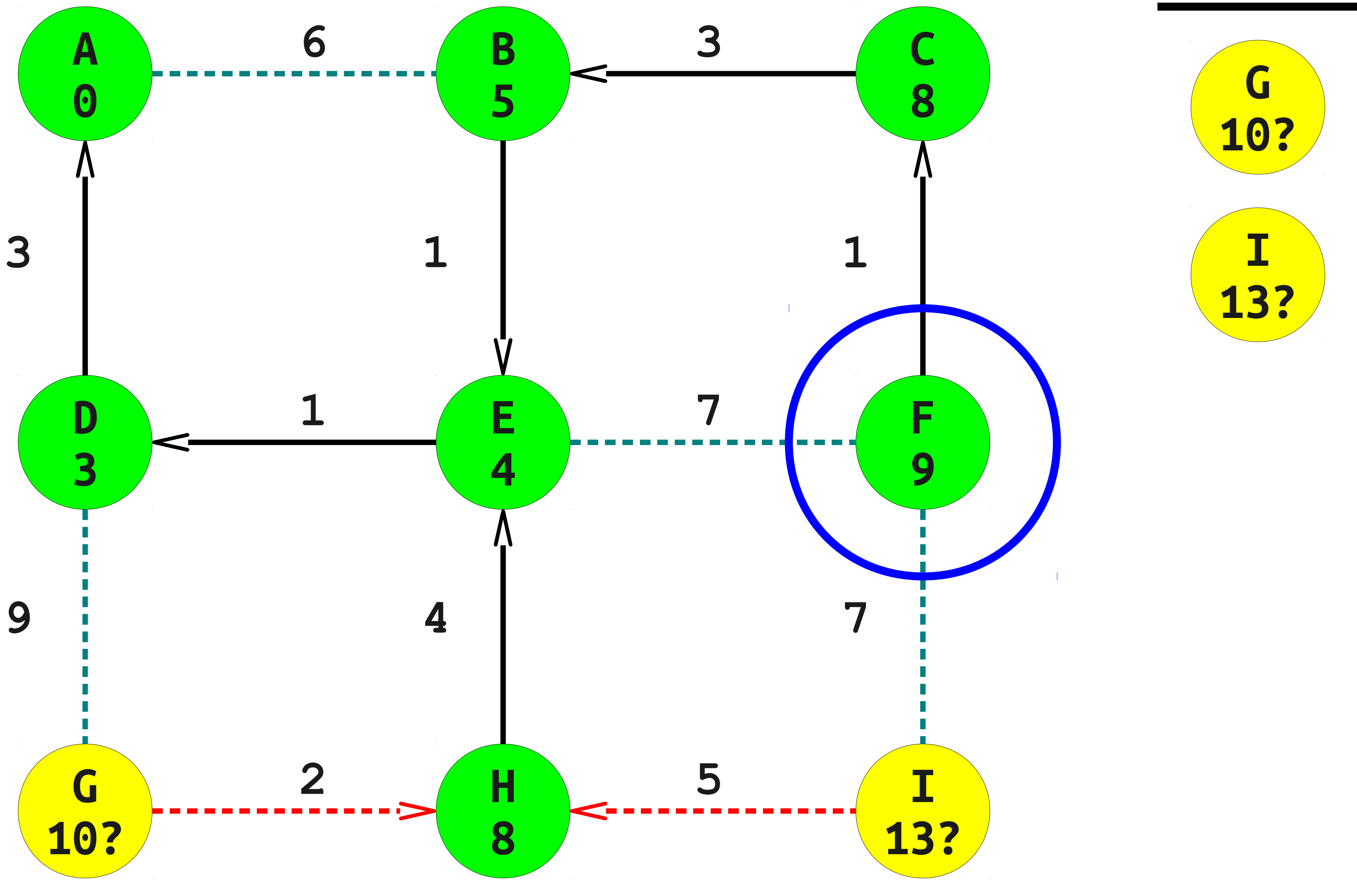


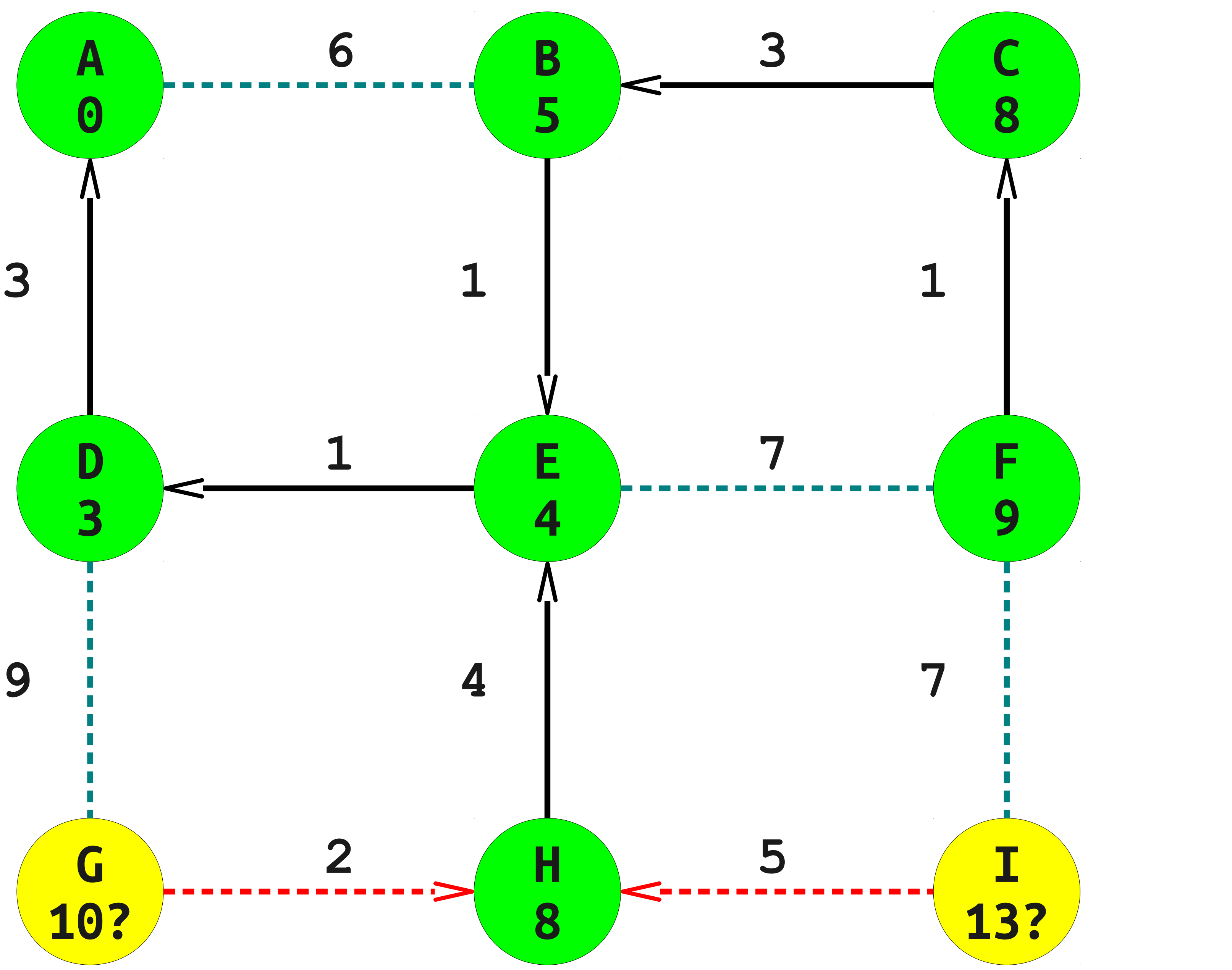
A
0
6
B
5
3
C
8
G
10?
I
13?
3
1
1
D
3
1
E
4
7
F
9?
9
4
7
G
10?
2
H
8
5
I
13?


A
0
6
B
5
3
C
8
G
10?
I
13?
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?

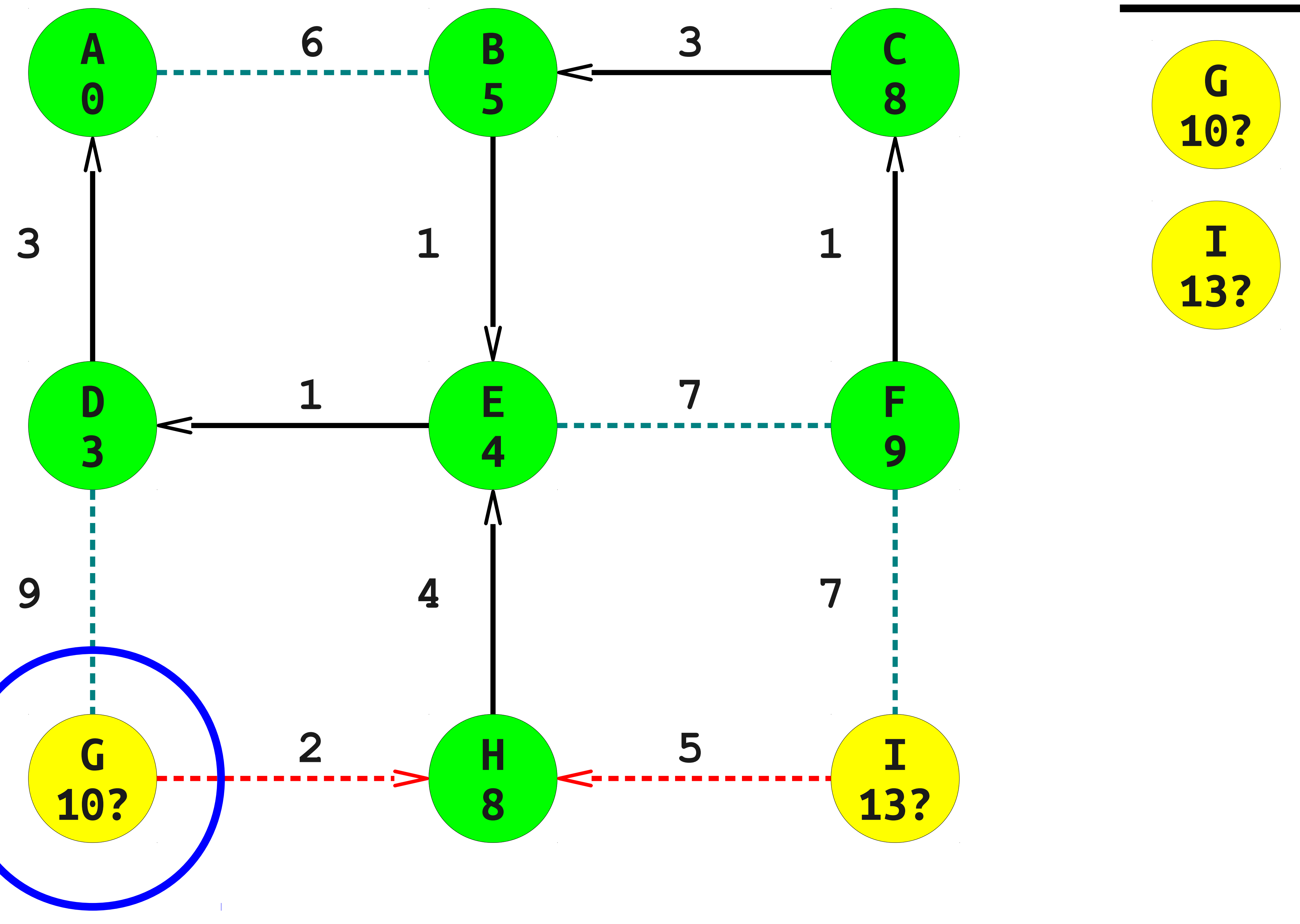

A
0
6
B
5
3
C
8
G
10?
3
1
1
I
13?
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?


A
0
6
B
5
3
C
8
G
10?
3
1
1
I
13?
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?

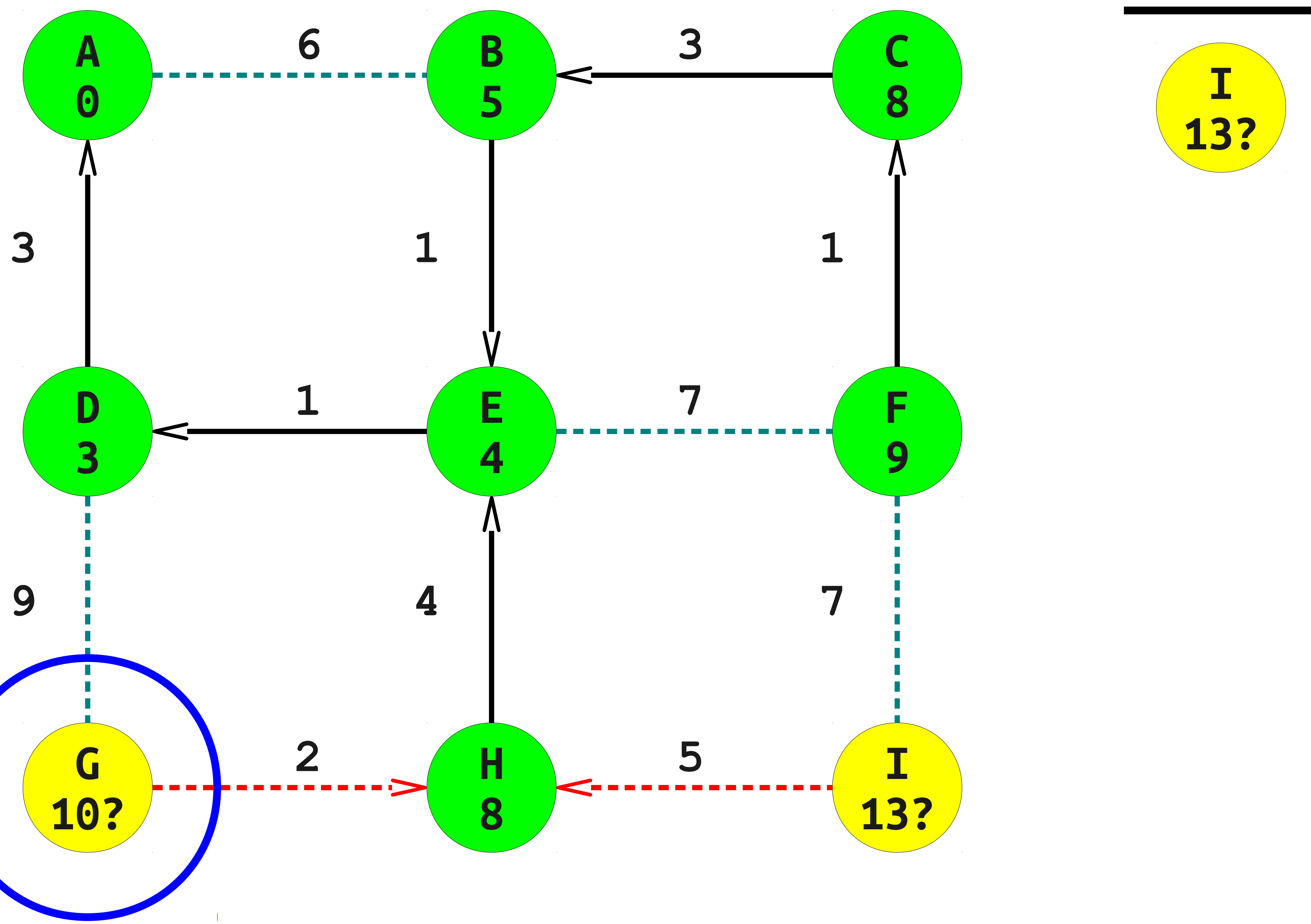

A
0
6
B
5
3
C
8
G
10?
I
13?
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?

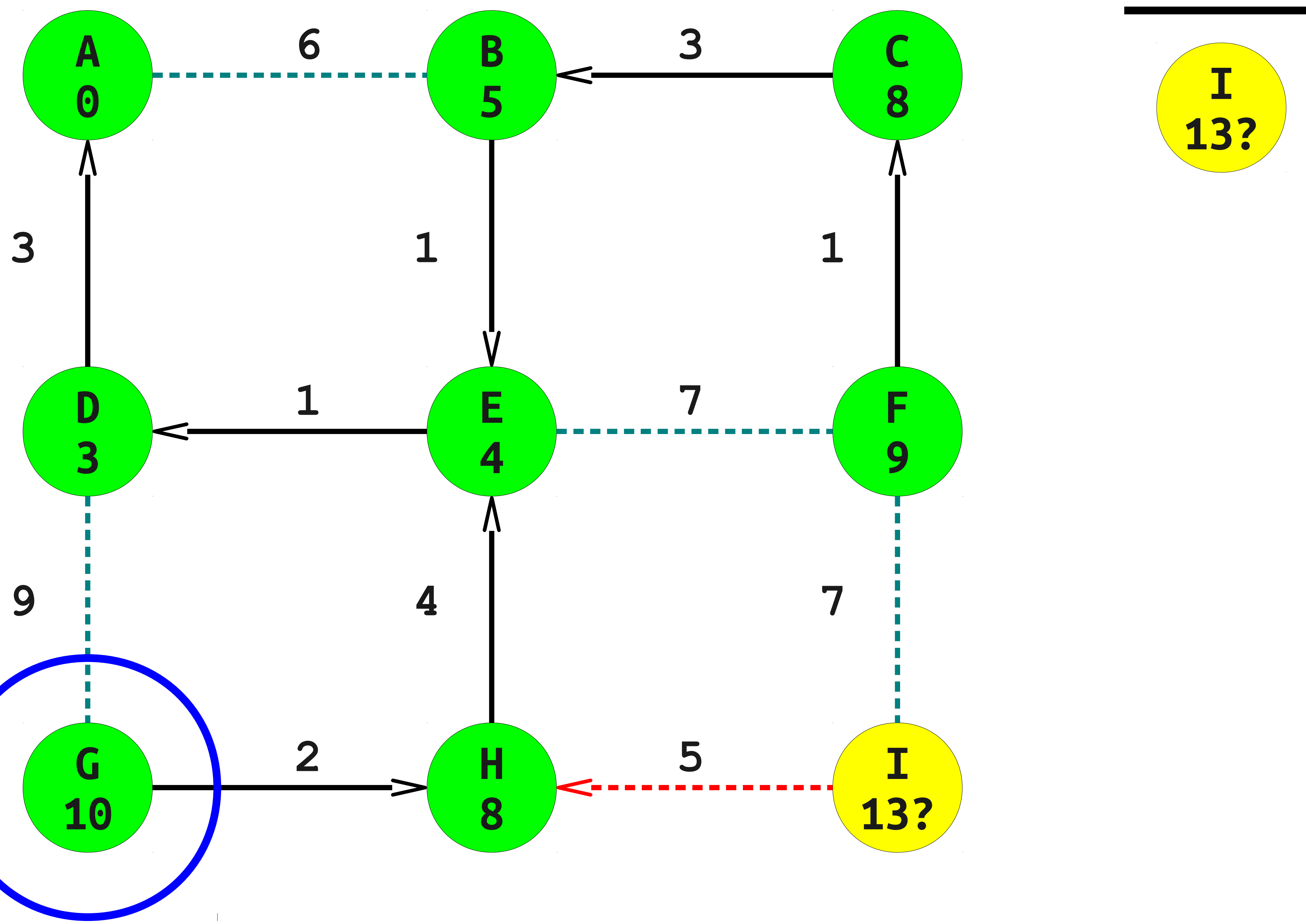


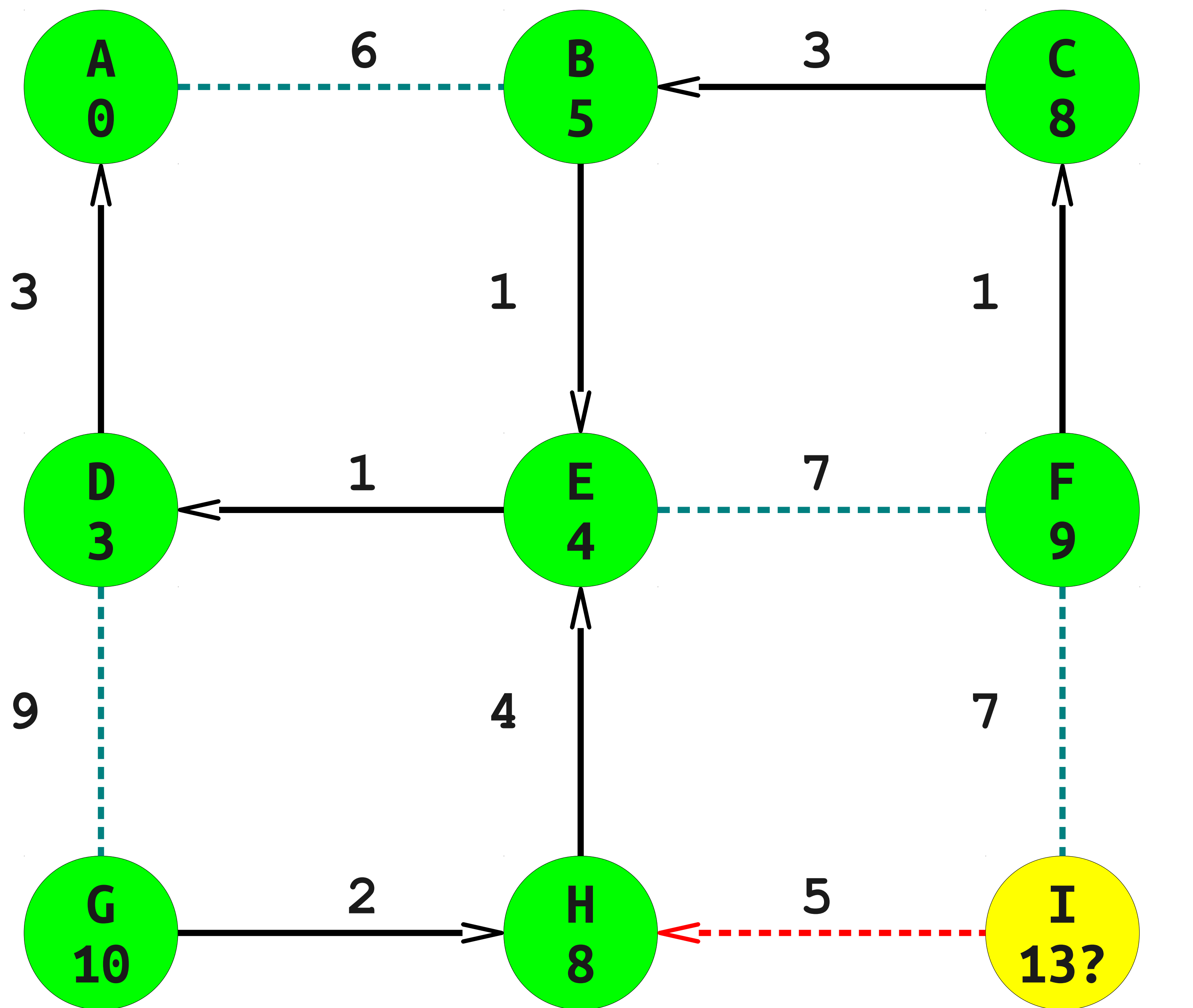
A
0
6
B
5
3
C
8
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?
G
10?
I
13?



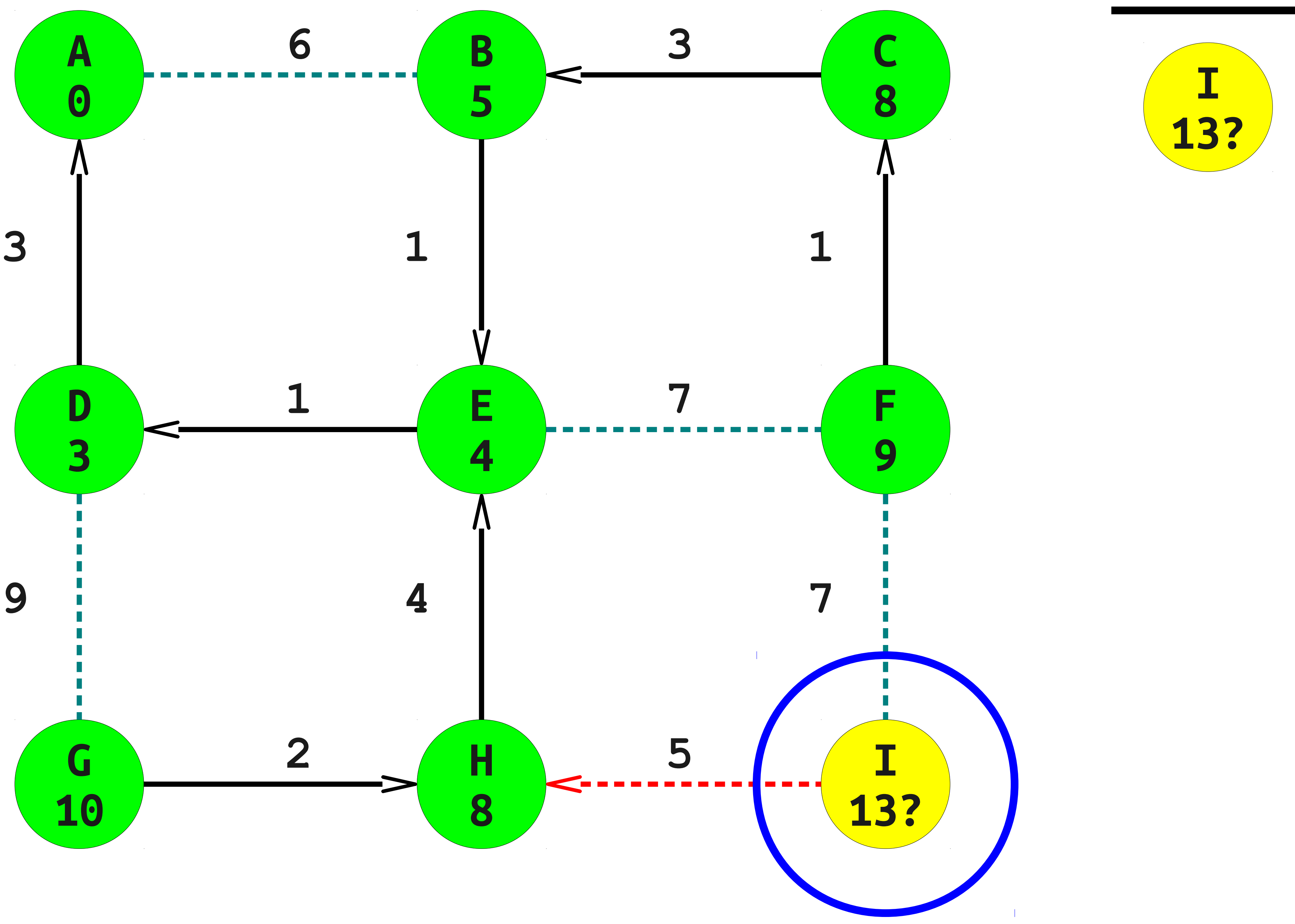
A
0
6
B
5
3
C
8
G
10?
I
13?
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?

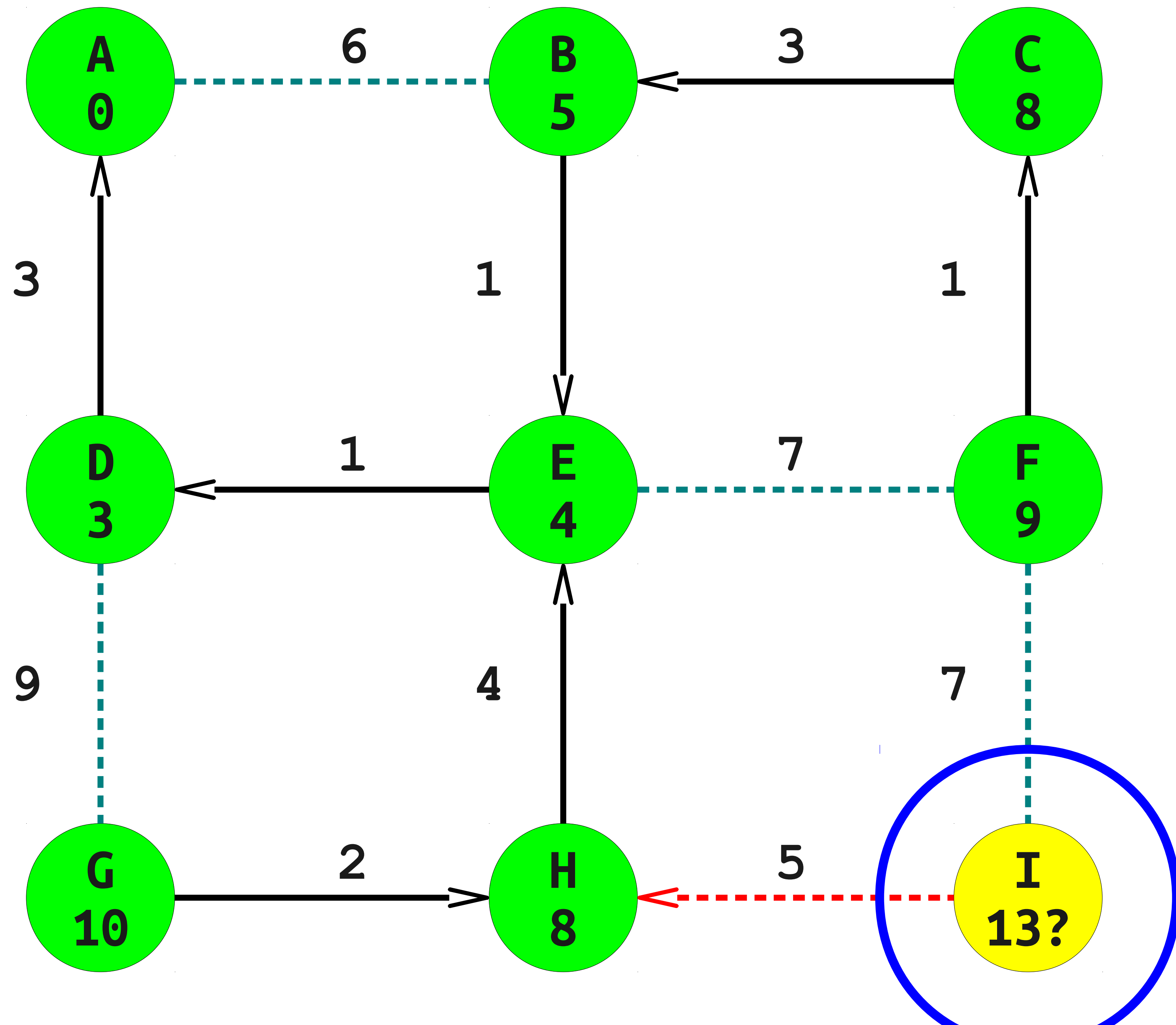

A
0
6
B
5
3
C
8
I
13?
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10?
2
H
8
5
I
13?


A
0
6
B
5
3
C
8
I
13?
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10
2
H
8
5
I
13?

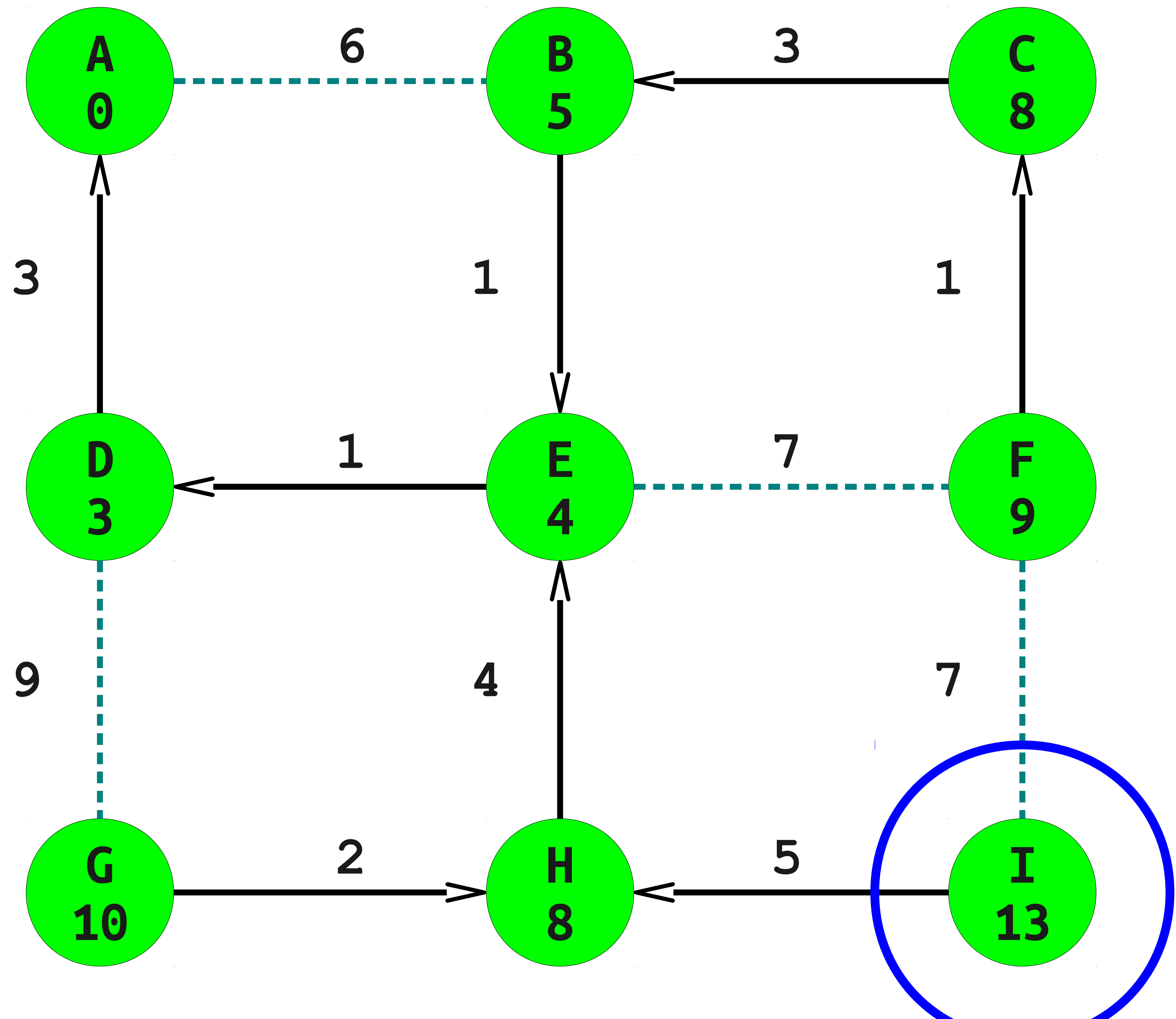


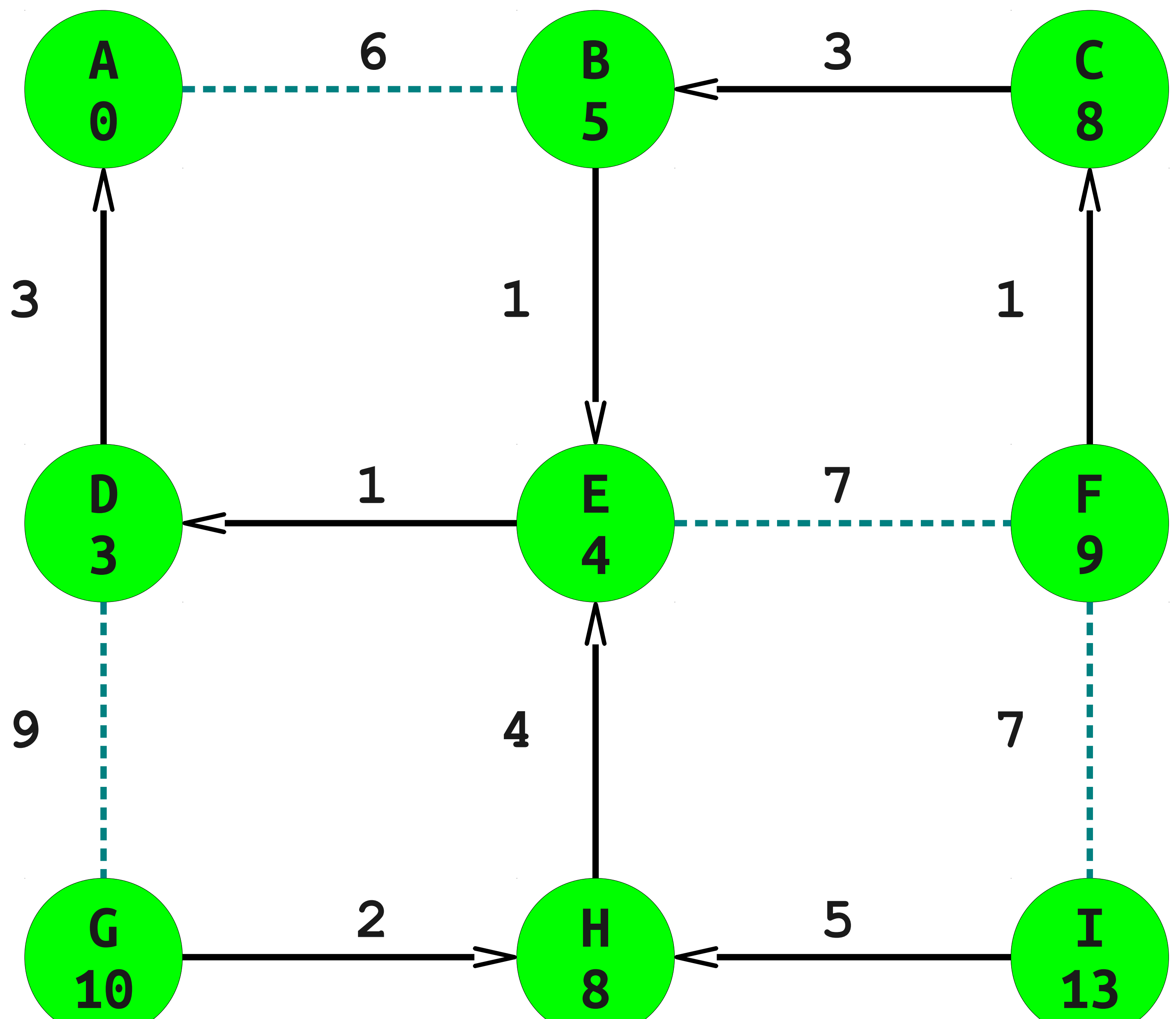
I
13?


A
0
B
5
C
8
I
13?
6
3
3
1
1
D
3
E
4
F
9
1
7
9
4
7
G
10
H
8
I
13?
2
5



A
0
6
B
5
3
C
8
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10
2
H
8
5
I
13?


A
0
6
B
5
3
C
8
3
1
1
D
3
1
E
4
7
F
9
9
4
7
G
10
2
H
8
5
I
13


A
0
B
5
C
8
D
3
E
4
F
9
G
10
H
8
I
13
6
3
3
1
1
1
7
9
4
7
2
5

This approach is called **Dijkstra's algorithm**.

Dijkstra's Algorithm

- Split nodes apart into three groups:
  - Green nodes, where we already have the shortest path;
  - Gray nodes, which we have never seen; and
  - Yellow nodes that we still need to process.
- Dijkstra's algorithm works as follows:
  - Mark all nodes gray except the start node, which is yellow and has cost 0.
  - Until no yellow nodes remain:
    - Choose the yellow node with the lowest total cost.
    - Mark that node green.
    - Mark all its gray neighbors yellow and with the appropriate cost.
    - Update the costs of all adjacent yellow nodes by considering the path through the current node.

An Important Note

- The version of Dijkstra's algorithm I have just described is ***not*** the same as the version described in the course reader.
- This version is more complex than the book's version, but is much faster.

Differences from BFS

- BFS uses a queue, while Dijkstra's algorithm uses a **priority queue**, where priorities are potential distances to nodes.
  - Need a special operation called **decrease-key**, which lowers the priority of an enqueued element.
- BFS never changes a parent pointer once it is set, while Dijkstra's algorithm might change parent pointers.
  - If a possible path to a node is found to be incorrect, then its parent might change.

The next slide will be really useful as a reference, so I changed the color scheme so that you can find it more easily.

- Mark all nodes as gray.
- Mark the initial node s as yellow and at candidate distance **0**.
- Enqueue s into the priority queue.
- While not all nodes have been visited:
  - Dequeue the lowest-cost node u from the priority queue.
  - Color u green. The candidate distance d that is currently stored for node u is the length of the shortest path from s to u.
  - If u is the destination node t, you have found the shortest path from s to t and are done.
  - For each node v connected to u by an edge of length L:
    - If v is gray:
      - Color v yellow.
      - Mark v's distance as $d + L$.
      - Set v's parent to be u.
      - Enqueue v into the priority queue.
    - If v is yellow and the candidate distance to v is greater than $d + L$:
      - Update v's candidate distance to be $d + L$; the older candidate distance is incorrect.
      - Update v's parent to be u.
      - Update v's priority in the priority queue to $d + L$.

This slide is not very important, so we're back to the old color scheme.

Implementation Concerns

- At each point in time, the algorithm must keep track of the following:
  - Which nodes are green (already done), yellow (in the priority queue but not done yet), and gray (never visited before).
  - The guessed distances to each node.
  - The parents of each yellow / green node.
  - The contents of the priority queue.
- There are ***many*** ways that you can store this information. It depends on many factors, including how the graph is represented.

A Few Quick Announcements

Friday Four Square!

Last one of the quarter! Today at 4:15PM
at Gates Computer Science

Please evaluate this course on Axess.

Your feedback really does make a difference. I read every single evaluation. Everything is done anonymously.

Heads-Up: Assignment 7

</announcements>

One Detail with Dijkstra's Algorithm

1?

1? 1?

1?

1?

1? 1?

1?

1?

1? 1?

1

1?

1? 1?

2? 1 2?

2?

1?

1? 1?

2? 1 2?

2?

1?

1 1?

2? 1 2?

2?

2? 1?

2? 1 1?

2? 1 2?

2?

2? 1?

2? 1 1?

2? 1 2?

2?

2? 1

2? 1 1?

2? 1 2?

2?

2?

2? 1 2?

2? 1 1?

2? 1 2?

2?

2?

2? 1 2?

2? 1 1

2? 1 2?

2?

2?

2? 1 2?

2? 1 1 2?

2? 1 2?

2?

2?

2? 1 2

2? 1 1 2?

2? 1 2?

2?

2? 3?

2? 1 2 3?

2? 1 1 2?

2? 1 2?

2?

				2?	3?								
			2?	1	2	3?							
		2?	1	★	1	2?			★				
			2?	1	2								
				2?									

2? 3?

2? 1 2 3?

2? 1 ☆ 1 2? ☆

2? 1 2 3?

2? 3?

				2?	3?								
			2?	1	2	3?							
		2?	1	★	1	2			★				
			2?	1	2	3?							
				2?	3?								

		2?	3?				
	2?	1	2	3?			
2?	1	★	1	2	3?		★
	2?	1	2	3?			
		2?	3?				

				2?	3?								
			2?	1	2	3?							
		2?	1	★	1	2	3?		★				
			2	1	2	3?							
				2?	3?								

2? 3?
2? 1 2 3?
2? 1 1 2 3?
3? 2 1 2 3?
3? 2? 3?

2? 3?

2? 1 2 3?

2? 1 ★ 1 2 3? ★

3? 2 1 2 3?

3? 2 3?

3?

				2?	3?								
		3?	2?	1	2	3?							
	3?	2	1	★	1	2	3?		★				
		3?	2	1	2	3?							
			3?	2	3?								
				3?									

			3?	2?	3?								
		3?	2	1	2	3?							
	3?	2	1	★	1	2	3?		★				
		3?	2	1	2	3?							
			3?	2	3?								
				3?									

3?

3? 2 3?

3? 2 1 2 3?

3? 2 1 ★ 1 2 3? ★

3? 2 1 2 3?

3? 2 3?

3?

				3?									
			3?	2	3?								
		3?	2	1	2	3?							
	3?	2	1	★	1	2	3?		★				
		3?	2	1	2	3?							
			3?	2	3	4?							
				3?	4?								

				3?									
			3?	2	3?								
		3?	2	1	2	3?							
	3?	2	1	★	1	2	3?		★				
		3?	2	1	2	3	4?						
			3?	2	3	4?							
				3?	4?								

				3?									
			3?	2	3?								
		3?	2	1	2	3?							
	3?	2	1	☆	1	2	3?		☆				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				3?									
			3?	2	3?								
	4?	3?	2	1	2	3?							
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				3?	4?								
			3?	2	3	4?							
	4?	3?	2	1	2	3?							
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				3?	4?								
		4?	3?	2	3	4?							
	4?	3	2	1	2	3?							
4?	3	2	1		1	2	3?						
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				3?	4?								
		4?	3?	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

			4?	3?	4?								
		4?	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				4?									
			4?	3	4?								
		4?	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3?	2	3	4?							
				3?	4?								

				4?									
			4?	3	4?								
		4?	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3	2	3	4?							
			4?	3?	4?								

				4?									
			4?	3	4?								
		4?	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3?		★				
	4?	3	2	1	2	3	4?						
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
			4?	3	4?								
		4?	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3	4?	★				
	4?	3	2	1	2	3	4?						
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?								
	5?	4	3	2	3	4?							
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3	4?	★				
	4?	3	2	1	2	3	4?						
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
	4?	3	2	1	2	3	4?						
4?	3	2	1	★	1	2	3	4?	★				
	4?	3	2	1	2	3	4?						
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
	4?	3	2	1	2	3	4?	5?					
4?	3	2	1	★	1	2	3	4	5? ★				
	4?	3	2	1	2	3	4?	5?					
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
	4?	3	2	1	2	3	4	5?					
4?	3	2	1	★	1	2	3	4	5? ★				
	4?	3	2	1	2	3	4?	5?					
		4?	3	2	3	4?							
			4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
	4?	3	2	1	2	3	4	5?					
4?	3	2	1	★	1	2	3	4	5? ★				
	4?	3	2	1	2	3	4?	5?					
	5?	4	3	2	3	4?							
		5?	4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4?	3	2	1	★	1	2	3	4	5? ★				
	4?	3	2	1	2	3	4?	5?					
	5?	4	3	2	3	4?							
		5?	4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4?	3	2	1	2	3	4?	5?					
	5?	4	3	2	3	4?							
		5?	4?	3	4?								
				4?									

				4?									
		5?	4?	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4?	5?					
	5?	4	3	2	3	4?							
		5?	4?	3	4?								
				4?									

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4?	5?					
	5?	4	3	2	3	4?							
		5?	4?	3	4?								
				4?									

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4?	5?						
		5?	4?	3	4?								
				4?									

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4?	5?						
		5?	4?	3	4	5?							
				4?	5?								

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4?	5?						
		5?	4?	3	4	5?							
			5?	4	5?								

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4?	5?						
		5?	4	3	4	5?							
			5?	4	5?								

			5?	4?									
		5?	4	3	4?	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5?						
		5?	4	3	4	5?							
			5?	4	5?								

			5?	4?	5?								
		5?	4	3	4	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5?						
		5?	4	3	4	5?							
			5?	4	5?								

			5?	4	5?								
		5?	4	3	4	5?							
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5?						
		5?	4	3	4	5?							
			5?	4	5?								

			5?	4	5?	6?							
		5?	4	3	4	5	6?						
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5?						
		5?	4	3	4	5?							
			5?	4	5?								

			5?	4	5?	6?							
		5?	4	3	4	5	6?						
	5?	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5? ★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5	6?					
		5?	4	3	4	5?	6?						
			5?	4	5?								

<table>
<tr><td></td><td></td><td></td><td>5?</td><td>4</td><td>5?</td><td>6?</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>6?</td><td>5?</td><td>4</td><td>3</td><td>4</td><td>5</td><td>6?</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6?</td><td>5</td><td>4</td><td>3</td><td>2</td><td>3</td><td>4</td><td>5?</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>5?</td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5?</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>3</td><td>2</td><td>1</td><td>★</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5? ★</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5?</td><td>4</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5?</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5?</td><td>4</td><td>3</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6?</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>5?</td><td>4</td><td>3</td><td>4</td><td>5?</td><td>6?</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>5?</td><td>4</td><td>5?</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

			5?	4	5?	6?							
	6?	5?	4	3	4	5	6?						
6?	5	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	5★?				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5	6?					
	6?	5	4	3	4	5?	6?						
		6?	5?	4	5?								

			5?	4	5?	6?							
	6?	5?	4	3	4	5	6?						
6?	5	4	3	2	3	4	5?						
5?	4	3	2	1	2	3	4	5?					
4	3	2	1	★	1	2	3	4	★				
5?	4	3	2	1	2	3	4	5?					
	5?	4	3	2	3	4	5	6?					
	6?	5	4	3	4	5?	6?						
		6?	5?	4	5?								

How Dijkstra's Works

- Dijkstra's algorithm works by incrementally computing the shortest path to intermediary nodes in the graph in case they prove to be useful.
- Most of these nodes are completely in the wrong direction.
- No “big-picture” conception of how to get to the destination – the algorithm explores outward in all directions.
- Could we give the algorithm a hint?

Heuristics

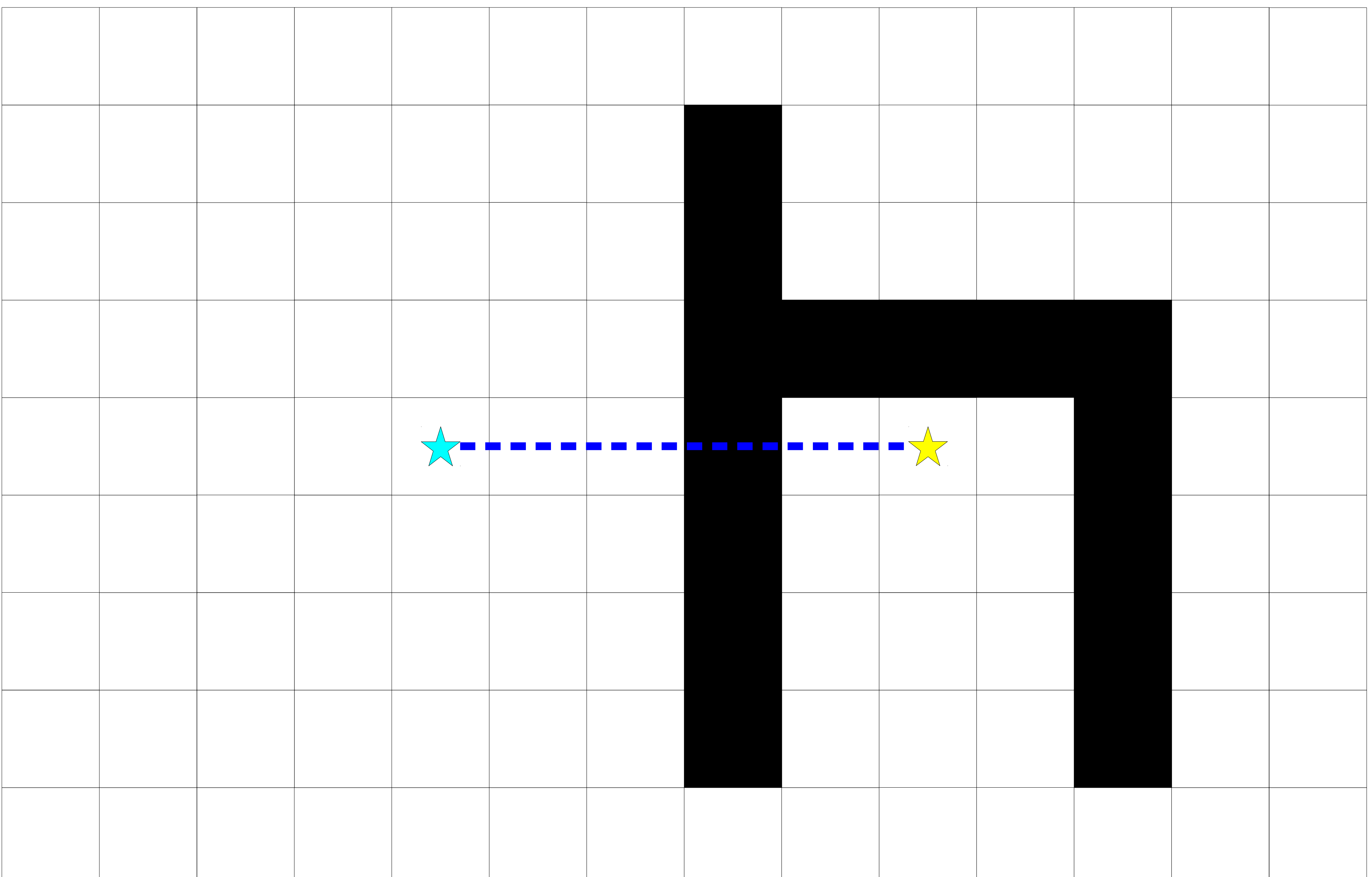
- In the context of graph searches, a **heuristic function** is a function that guesses the distance from some known node to the destination node.
- The guess doesn't have to be correct, but it should try to be as accurate as possible.
- Examples: For Google Maps, a heuristic for estimating distance might be the straight-line “as the crow flies” distance.

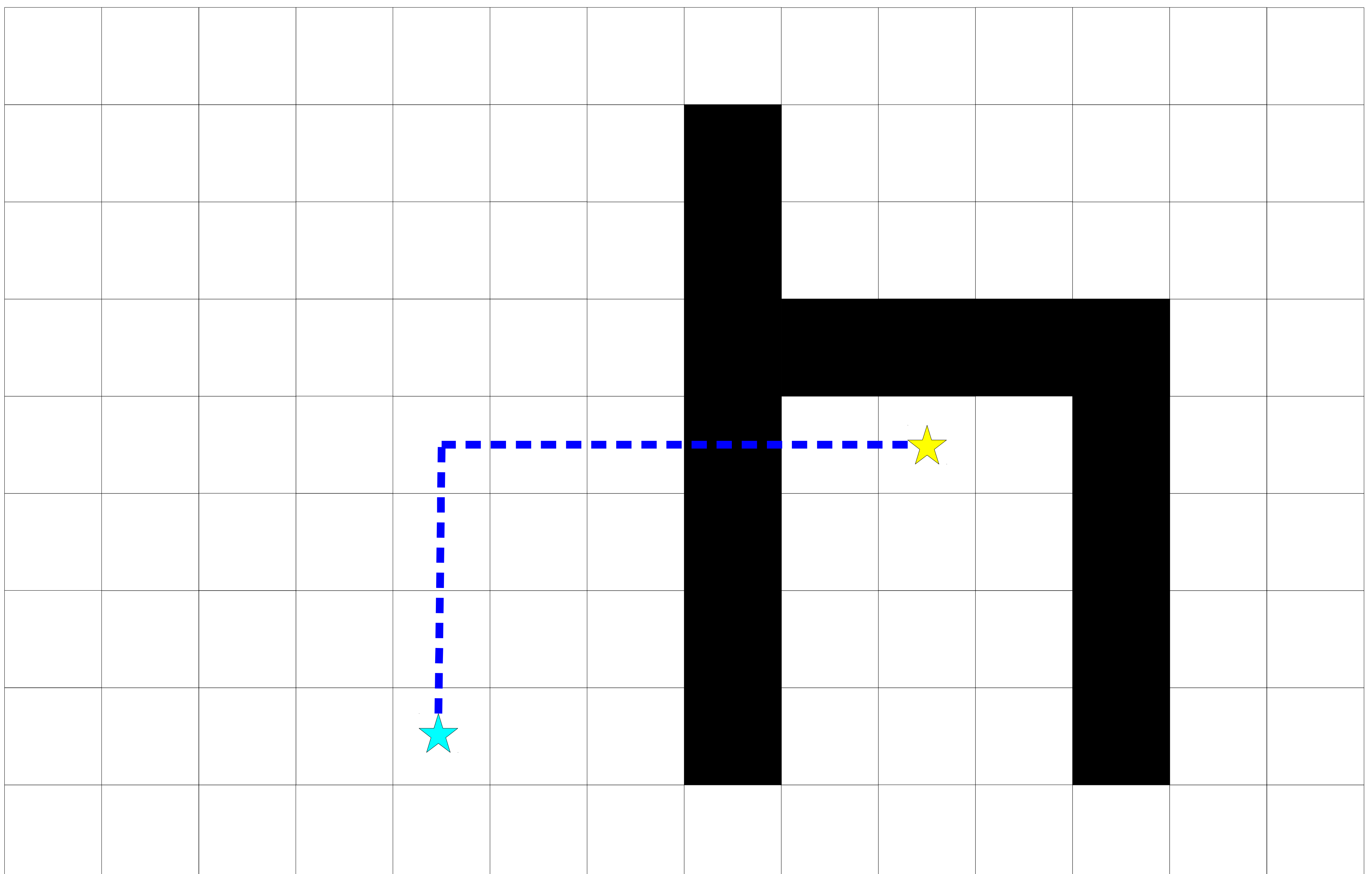
Admissible Heuristics

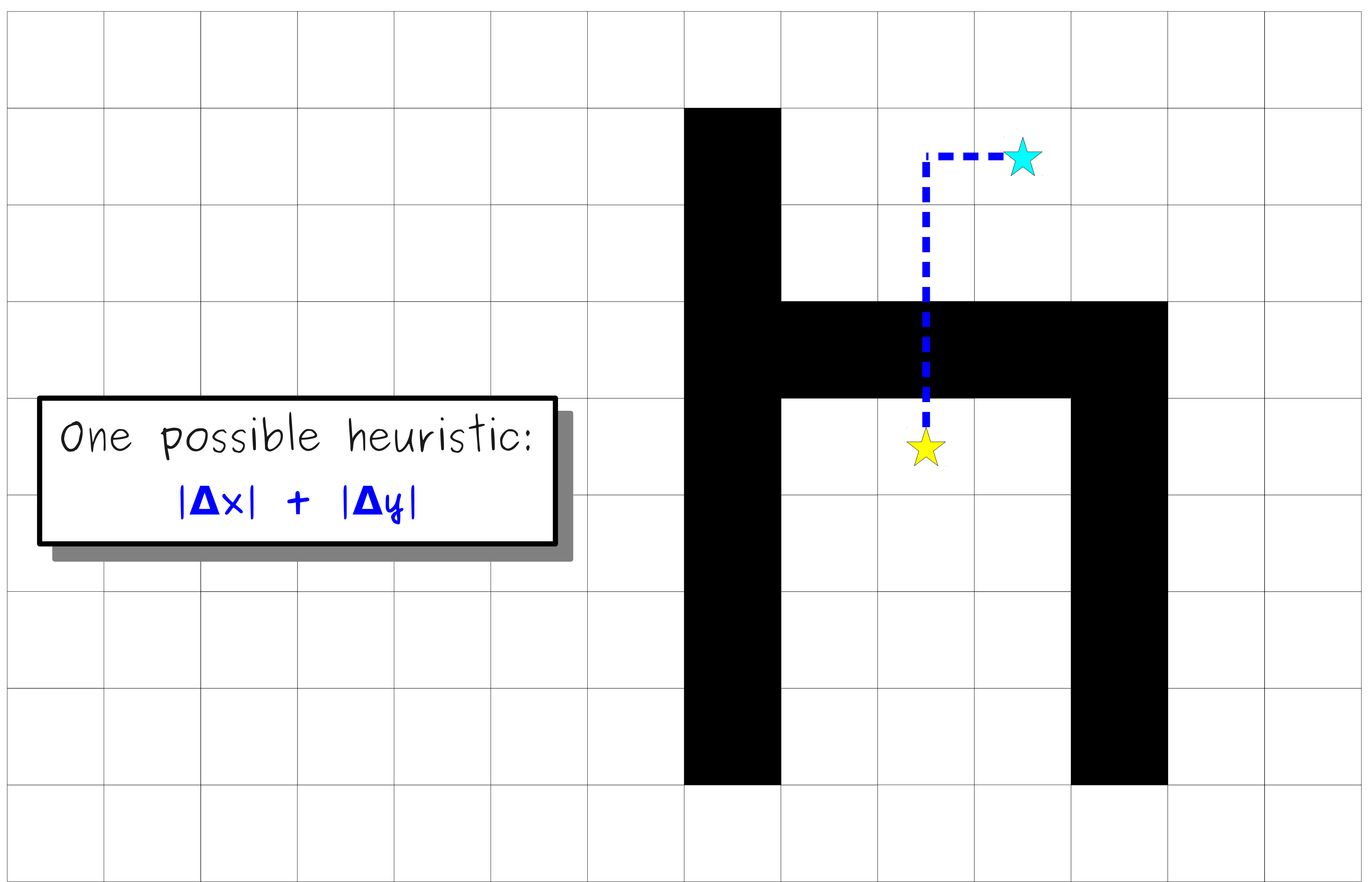
- A heuristic function is called an **admissible heuristic** if it never overestimates the distance from any node to the destination.
- In other words:

predicted-distance ≤ actual-distance

h

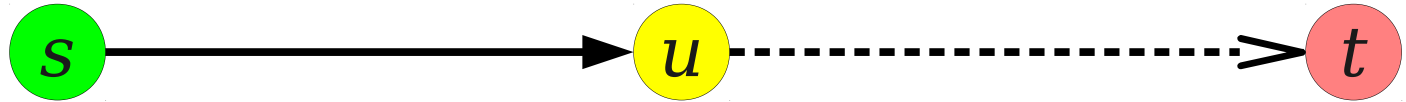





One possible heuristic:
|Δx| + |Δy|

Why Heuristics Matter

- We can modify Dijkstra's algorithm by introducing heuristic functions.
- Given any node u, there are two associated costs:



- The actual distance from the start node s.
- The heuristic distance from u to the end node t.
- Key idea: Run Dijkstra's algorithm, but use the following priority in the priority queue:

priority(u) = distance(s, u) + heuristic(u, t)

- This modification of Dijkstra's algorithm is called the **A* search algorithm**.

1?

1? 1?

1?

1?

1?

1?

1?

1?

1?

1 +
4?

1?

1?

1?

1 +
4?

1?

1 +
6?

1?

1 +
4?

1?

1 + 6?

1?

1 + 4?

1?

1 + 6?

1?

1 + 4?

1 + 6?

1 + 6?

1?

1 + 4?

1 + 6?

1 + 6?

1 + 6?

1 + 4?

1 + 6?

1 + 6?

1 + 6?

1 + 4?

1 + 6?

1 + 6?

1 + 6?

1

1 + 6?

1 + 6?
2 + 5?
1 + 6?
1
2 + 3?
1 + 6?
2 + 5?

1 + 6?

2 + 5?

1 + 6?

1

2 + 3?

1 + 6?

2 + 5?

1 + 6?
2 + 5?
1 + 6?
1
2
1 + 6?
2 + 5?

1 + 6?
2 + 5?
3 + 4?

1 + 6?
1
2
3 + 2?

1 + 6?
2 + 5?
3 + 4?

	1 + 6?	2 + 5?	3 + 4?	
1 + 6?		1	2	3 + 2?
	1 + 6?	2 + 5?	3 + 4?	

	1 + 6?	2 + 5?	3 + 4?			
1 + 6?	★	1	2	3		★
	1 + 6?	2 + 5?	3 + 4?			

	1 + 6?	2 + 5?	3 + 4?	4 + 3?		
1 + 6?		1	2	3	4 + 1?	
	1 + 6?	2 + 5?	3 + 4?	4 + 3?		

	1 + 6?	2 + 5?	3 + 4?	4 + 3?		
1 + 6?		1	2	3	4 + 1?	
	1 + 6?	2 + 5?	3 + 4?	4 + 3?		

	1 + 6?	2 + 5?	3 + 4?	4 + 3?	
1 + 6?		1	2	3	4
	1 + 6?	2 + 5?	3 + 4?	4 + 3?	

	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	
1 + 6?		1	2	3	4	5 + 0?
	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	

	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	
1 + 6?	☆	1	2	3	4	5 + 0?
	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	

	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	
1 + 6?		1	2	3	4	
	1 + 6?	2 + 5?	3 + 4?	4 + 3?	5 + 2?	

1 + 6?

1 + 6?

1 + 4?

1 + 6?

1 + 6?

1 + 6?

1

1 + 6?

1 + 6? 2 + 5?

1 + 6? 1 2 + 3?

1 + 6? 2 + 5?

1 + 6? 2 + 5?

1 + 6? 1 2

1 + 6? 2 + 5?

1 + 6?
2 + 5?
3 + 4?

1 + 6?
1
2

1 + 6?
2 + 5?
3 + 4?

1 + 6?	2 + 5?	3 + 4?

1 ★ 1 2 ★

1 + 6?	2 + 5?	3 + 4?

	2 + 7?	1 + 6?	2 + 5?	3 + 4?	
2 + 7?	1	★	1	2	
	2 + 7?	1 + 6?	2 + 5?	3 + 4?	

	2 + 7?	1 + 6?	2 + 5?	3 + 4?			
2 + 7?	1	★	1	2			★
	2 + 7?	1 + 6?	2	3 + 4?			

2 + 7? 1 + 6? 2 + 5? 3 + 4?

2 + 7? 1 1 2

2 + 7? 1 + 6? 2 3 + 4?

3 + 6?

2 + 7? | 1 + 6? | 2 | 3 + 4?

2 + 7? | 1 | 1 | 2

2 + 7? | 1 + 6? | 2 | 3 + 4?

3 + 6?

3 + 6?

2 + 7? 1 + 6? 2 3 + 4?

2 + 7? 1 1 2

2 + 7? 1 + 6? 2 3 + 4?

3 + 6?

3 + 6?

2 + 7? 1 2 3 + 4?

2 + 7? 1 ★ 1 2 ★

2 + 7? 1 + 6? 2 3 + 4?

3 + 6?

2 + 7? 3 + 6?

2 + 7? 1 2 3 + 4?

2 + 7? 1 1 2

2 + 7? 1 + 6? 2 3 + 4?

3 + 6?

2 + 7?
3 + 6?
2 + 7?
1
2
3 + 4?
2 + 7?
1
1
2
2 + 7?
1
2
3 + 4?
3 + 6?

2 + 7? 3 + 6?

2 + 7? 1 2 3 + 4?

2 + 7? 1 1 2

2 + 7? 1 2 3 + 4?

2 + 7? 3 + 6?

		2 + 7?	3 + 6?	
	2 + 7?	1	2	3
2 + 7?	1		1	2
	2 + 7?	1	2	3 + 4?
		2 + 7?	3 + 6?	

		2 + 7?	3 + 6?	4 + 5?
	2 + 7?	1	2	3
2 + 7?	1		1	2
	2 + 7?	1	2	3 + 4?
		2 + 7?	3 + 6?	

2 + 7? 3 + 6? 4 + 5?

2 + 7? 1 2 3

2 + 7? 1 1 2

2 + 7? 1 2 3

2 + 7? 3 + 6?

		2 + 7?	3 + 6?	4 + 5?	
	2 + 7?	1	2	3	
2 + 7?	1		1	2	
	2 + 7?	1	2	3	
		2 + 7?	3 + 6?	4 + 5?	

		2 + 7?	3 + 6?	4 + 5?
	2 + 7?	1	2	3
2 + 7?	1		1	2
	2	1	2	3
		2 + 7?	3 + 6?	4 + 5?

		2 + 7?	3 + 6?	4 + 5?
	2 + 7?	1	2	3
2 + 7?	1		1	2
3 + 8?	2	1	2	3
	3 + 8?	2 + 7?	3 + 6?	4 + 5?

		2	3 + 6?	4 + 5?			
	2 + 7?	1	2	3			
2 + 7?	1	★	1	2			★
3 + 8?	2	1	2	3			
	3 + 8?	2 + 7?	3 + 6?	4 + 5?			

3 + 8?

3 + 8? 2 3 + 6? 4 + 5?

2 + 7? 1 2 3

2 + 7? 1 1 2

3 + 8? 2 1 2 3

3 + 8? 2 + 7? 3 + 6? 4 + 5?

		3 + 8?		
	3 + 8?	2	3 + 6?	4 + 5?
	2 + 7?	1	2	3
2 + 7?	1		1	2
3 + 8?	2	1	2	3
	3 + 8?	2 + 7?	3 + 6?	4

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
			2 + 7?	1	2	3							
		2 + 7?	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2 + 7?	3 + 6?	4	5 + 4?						
						5 + 6?							

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
			2 + 7?	1	2	3							
		2 + 7?	1	★	1	2			★				
		3 + 8?	2	1	2	3							
			3 + 8?	2	3 + 6?	4	5 + 4?						
						5 + 6?							

		3 + 8?					
	3 + 8?	2	3 + 6?	4 + 5?			
	2 + 7?	1	2	3			
2 + 7?	1	★	1	2			★
3 + 8?	2	1	2	3			
	3 + 8?	2	3 + 6?	4	5 + 4?		
		3 + 8?		5 + 6?			

3 + 8?

3 + 8? 2 3 + 6? 4 + 5?

2 + 7? 1 2 3

2 + 7? 1 1 2

3 + 8? 2 1 2 3

3 + 8? 2 3 4 5 + 4?

3 + 8? 5 + 6?

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
			2 + 7?	1	2	3							
		2 + 7?	1	★	1	2			★				
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
			2 + 7?	1	2	3							
		2	1	★	1	2			★				
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
		3 + 8?	2 + 7?	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

				3 + 8?									
			3 + 8?	2	3 + 6?	4 + 5?							
		3 + 8?	2	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

			3 + 8?			
		3 + 8?	2	3	4 + 5?	
	3 + 8?	2	1	2	3	
3 + 8?	2	1		1	2	
	3 + 8?	2	1	2	3	
		3 + 8?	2	3	4	5 + 4?
			3 + 8?	4 + 7?	5 + 6?	

				3 + 8?	4 + 7?								
			3 + 8?	2	3	4 + 5?							
		3 + 8?	2	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

				3 + 8?	4 + 7?								
			3 + 8?	2	3	4							
		3 + 8?	2	1	2	3							
	3 + 8?	2	1	★	1	2			★				
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5 + 4?						
				3 + 8?	4 + 7?	5 + 6?							

			3 + 8?	4 + 7?	5 + 6?	
		3 + 8?	2	3	4	5 + 4?
	3 + 8?	2	1	2	3	
3 + 8?	2	1		1	2	
	3 + 8?	2	1	2	3	
		3 + 8?	2	3	4	5 + 4?
			3 + 8?	4 + 7?	5 + 6?	

				3 + 8?	4 + 7?	5 + 6?							
			3 + 8?	2	3	4	5 + 4?						
		3 + 8?	2	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5						
				3 + 8?	4 + 7?	5 + 6?							

				3 + 8?	4 + 7?	5 + 6?							
			3 + 8?	2	3	4	5 + 4?						
		3 + 8?	2	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3							
			3 + 8?	2	3	4	5	6 + 3?					
				3 + 8?	4 + 7?	5 + 6?	6 + 5?						

			3 + 8?	4 + 7?	5 + 6?		
		3 + 8?	2	3	4	5 + 4?	
	3 + 8?	2	1	2	3		
3 + 8?	2	1		1	2		
	3 + 8?	2	1	2	3		
		3 + 8?	2	3	4	5	6
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	

			3 + 8?	4 + 7?	5 + 6?			
		3 + 8?	2	3	4	5 + 4?		
	3 + 8?	2	1	2	3			
3 + 8?	2	1		1	2			
	3 + 8?	2	1	2	3		7 + 2?	
		3 + 8?	2	3	4	5	6	7 + 2?
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	

			3 + 8?	4 + 7?	5 + 6?			
		3 + 8?	2	3	4	5		
	3 + 8?	2	1	2	3			
3 + 8?	2	1		1	2			
	3 + 8?	2	1	2	3		7 + 2?	
		3 + 8?	2	3	4	5	6	7 + 2?
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	

				3 + 8?	4 + 7?	5 + 6?	6 + 5?						
			3 + 8?	2	3	4	5	6 + 3?					
		3 + 8?	2	1	2	3							
	3 + 8?	2	1		1	2							
		3 + 8?	2	1	2	3		7 + 2?					
			3 + 8?	2	3	4	5	6	7 + 2?				
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?					

				3 + 8?	4 + 7?	5 + 6?	6 + 5?						
			3 + 8?	2	3	4	5	6 + 3?					
		3 + 8?	2	1	2	3							
	3 + 8?	2	1	★	1	2			★				
		3 + 8?	2	1	2	3		7					
			3 + 8?	2	3	4	5	6	7 + 2?				
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?					

			3 + 8?	4 + 7?	5 + 6?	6 + 5?		
		3 + 8?	2	3	4	5	6 + 3?	
	3 + 8?	2	1	2	3			
3 + 8?	2	1		1	2		8 + 1?	
	3 + 8?	2	1	2	3		7	8 + 1?
		3 + 8?	2	3	4	5	6	7 + 2?
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	

			3 + 8?	4 + 7?	5 + 6?	6 + 5?		
		3 + 8?	2	3	4	5	6 + 3?	
	3 + 8?	2	1	2	3			
3 + 8?	2	1		1	2		8 + 1?	
	3 + 8?	2	1	2	3		7	8 + 1?
		3 + 8?	2	3	4	5	6	7
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	

			3 + 8?	4 + 7?	5 + 6?	6 + 5?			
		3 + 8?	2	3	4	5	6 + 3?		
	3 + 8?	2	1	2	3				
3 + 8?	2	1		1	2		8 + 1?		
	3 + 8?	2	1	2	3		7	8 + 1?	
		3 + 8?	2	3	4	5	6	7	8 + 3?
			3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?	

				3 + 8?	4 + 7?	5 + 6?	6 + 5?						
			3 + 8?	2	3	4	5	6 + 3?					
		3 + 8?	2	1	2	3							
	3 + 8?	2	1	★	1	2		8	★				
		3 + 8?	2	1	2	3		7	8 + 1?				
			3 + 8?	2	3	4	5	6	7	8 + 3?			
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?				

				3 + 8?	4 + 7?	5 + 6?	6 + 5?						
			3 + 8?	2	3	4	5	6 + 3?					
		3 + 8?	2	1	2	3		9 + 2?					
	3 + 8?	2	1	★	1	2		8	9 + 0? ★				
		3 + 8?	2	1	2	3		7	8 + 1?				
			3 + 8?	2	3	4	5	6	7	8 + 3?			
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?				

				3 + 8?	4 + 7?	5 + 6?	6 + 5?						
			3 + 8?	2	3	4	5	6					
		3 + 8?	2	1	2	3		9 + 2?					
	3 + 8?	2	1	★	1	2		8	★ 9 + 0?				
		3 + 8?	2	1	2	3		7	8 + 1?				
			3 + 8?	2	3	4	5	6	7	8 + 3?			
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?				

				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?					
			3 + 8?	2	3	4	5	6	7 + 2?				
		3 + 8?	2	1	2	3		7 + 2?					
	3 + 8?	2	1	★	1	2		8	9 + 0? ★				
		3 + 8?	2	1	2	3		7	8 + 1?				
			3 + 8?	2	3	4	5	6	7	8 + 3?			
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?				

				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?					
			3 + 8?	2	3	4	5	6	7 + 2?				
		3 + 8?	2	1	2	3		7 + 2?					
	3 + 8?	2	1		1	2		8					
		3 + 8?	2	1	2	3		7	8 + 1?				
			3 + 8?	2	3	4	5	6	7	8 + 3?			
				3 + 8?	4 + 7?	5 + 6?	6 + 5?	7 + 4?	8 + 3?				

For Comparison: What Dijkstra's Algorithm Would Have Searched

8	7	6	5	4	5	6	7	8	9?				
7	6	5	4	3	4	5	6	7	8	9?			
6	5	4	3	2	3	4	5	6	7	8	9?		
5	4	3	2	1	2	3		7	8	9?			
4	3	2	1	★	1	2		8	★				
5	4	3	2	1	2	3		7	8	9?			
6	5	4	3	2	3	4	5	6	7	8	9?		
7	6	5	4	3	4	5	6	7	8	9?			
8	7	6	5	4	5	6	7	8	9?				

A* Search

- As long as the heuristic is admissible (and satisfies one other technical condition), A* will always find the shortest path from the source to the destination node.
- Can be *dramatically* faster than Dijkstra's algorithm.
  - Focuses work in areas likely to be productive.
  - Avoids solutions that appear worse until there is evidence they may be appropriate.

Close Cousins

- Dijkstra's algorithm and A* search are very closely related:
  - Dijkstra's uses a node's candidate distance as its priority.
  - A* uses a node's candidate distance ***plus a heuristic value*** as its priority.
- Interesting fact: If you use the **zero heuristic** (which always predicts a node is at distance 0 from the endpoint), A* search is completely identical to Dijkstra's algorithm!

Dijkstra's Algorithm

- Mark all nodes as gray.
- Mark the initial node ***s*** as yellow and at candidate distance **0**.
- Enqueue ***s*** into the priority queue with priority **0**.
- While not all nodes have been visited:
  - Dequeue the lowest-cost node ***u*** from the priority queue.
  - Color ***u*** green. The candidate distance ***d*** that is currently stored for node ***u*** is the length of the shortest path from ***s*** to ***u***.
  - If ***u*** is the destination node ***t***, you have found the shortest path from ***s*** to ***t*** and are done.
  - For each node ***v*** connected to ***u*** by an edge of length ***L***:
    - If ***v*** is gray:
      - Color ***v*** yellow.
      - Mark ***v***'s distance as ***d* + *L***.
      - Set ***v***'s parent to be ***u***.
      - Enqueue ***v*** into the priority queue with priority ***d* + *L***.
    - If ***v*** is yellow and the candidate distance to ***v*** is greater than ***d* + *L***:
      - Update ***v***'s candidate distance to be ***d* + *L***.
      - Update ***v***'s parent to be ***u***.
      - Update ***v***'s priority in the priority queue to ***d* + *L***.

A* Search

- Mark all nodes as gray.
- Mark the initial node s as yellow and at candidate distance **0**.
- Enqueue s into the priority queue with priority $h(s, t)$.
- While not all nodes have been visited:
  - Dequeue the lowest-cost node u from the priority queue.
  - Color u green. The candidate distance d that is currently stored for node u is the length of the shortest path from s to u.
  - If u is the destination node t, you have found the shortest path from s to t and are done.
  - For each node v connected to u by an edge of length L:
    - If v is gray:
      - Color v yellow.
      - Mark v's distance as $d + L$.
      - Set v's parent to be u.
      - Enqueue v into the priority queue with priority $d + L + h(v, t)$.
    - If v is yellow and the candidate distance to v is greater than $d + L$:
      - Update v's candidate distance to be $d + L$.
      - Update v's parent to be u.
      - Update v's priority in the priority queue to $d + L + h(v, t)$.

Next Time

- **More Graph Algorithms**
  - Minimum Spanning Trees
  - Kruskal's Algorithm
  - Data Clustering