Borůvka's Algorithm
The Oldest MST Algorithm We Know

Presented by
Ali Tavassoly

Course conducted by
DR. Alireza Zarei

Sharif University of Technology

Presentation for Advanced Topics in Algorithm

Presentation Overview

History

This algorithm first published in 1926 by Otakar Borůvka as a method of constructing an efficient electricity network for Moravia.
The algorithm was rediscovered by Choquet in 1938; again by Florek, Lukasiewicz, Perkal, Steinhaus, and Zubrzycki in 1951; and again by Georges Sollin in 1965.
This algorithm is frequently called Sollin's algorithm, especially in the parallel computing literature.

Definition of MST problem

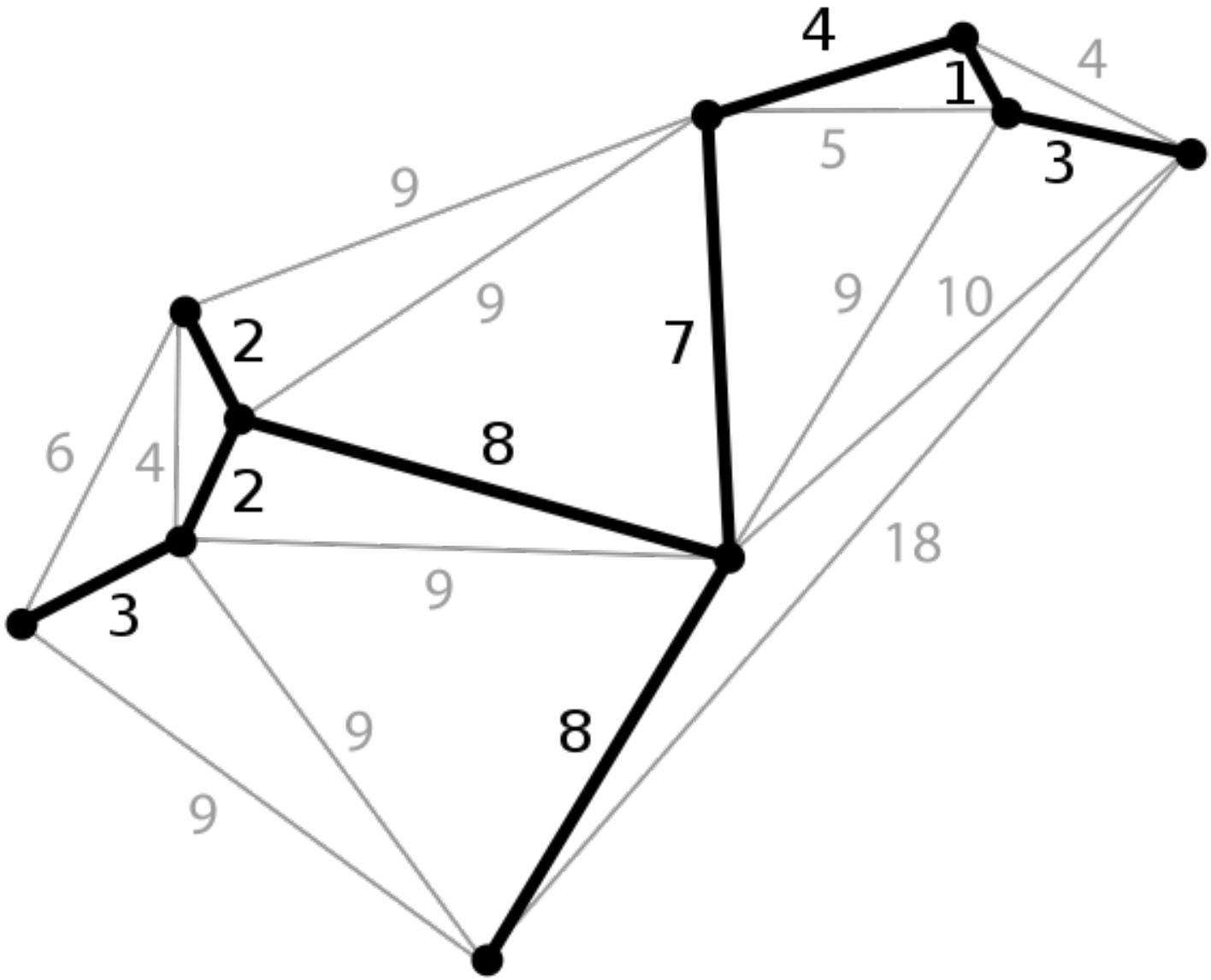
Definition

An edge-weighted graph is a graph where we associate weights or costs with each edge.

Definition

A minimum spanning tree (MST) of an edge-weighted graph is a spanning tree whose weight (the sum of the weights of its edges) is no larger than the weight of any other spanning tree.

Example of MST



Algorithm Explanation

Initial state

Consider n connected components (each vertex is in its component). So at the initial step, we have chosen no edges and we have n trees.

Procedure

At each step, there are some trees in the graph. For each tree, find the minimum weight of an edge connecting this tree to another one. After finding the edges for all the trees, pick them all for our final tree.

Final state

A tree consisting of $n - 1$ edges having minimum total weight among all possible trees.

Overview

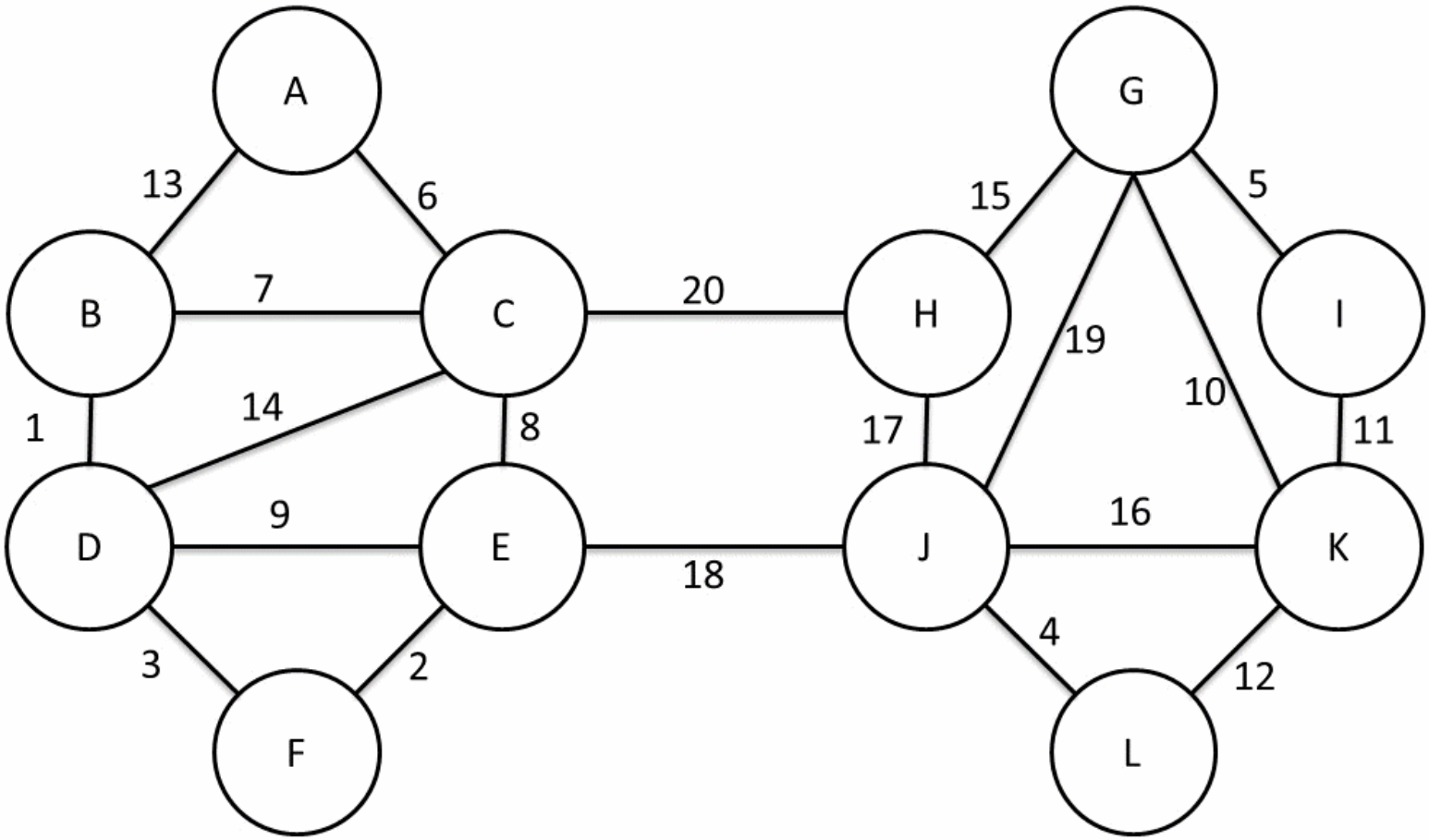
Note

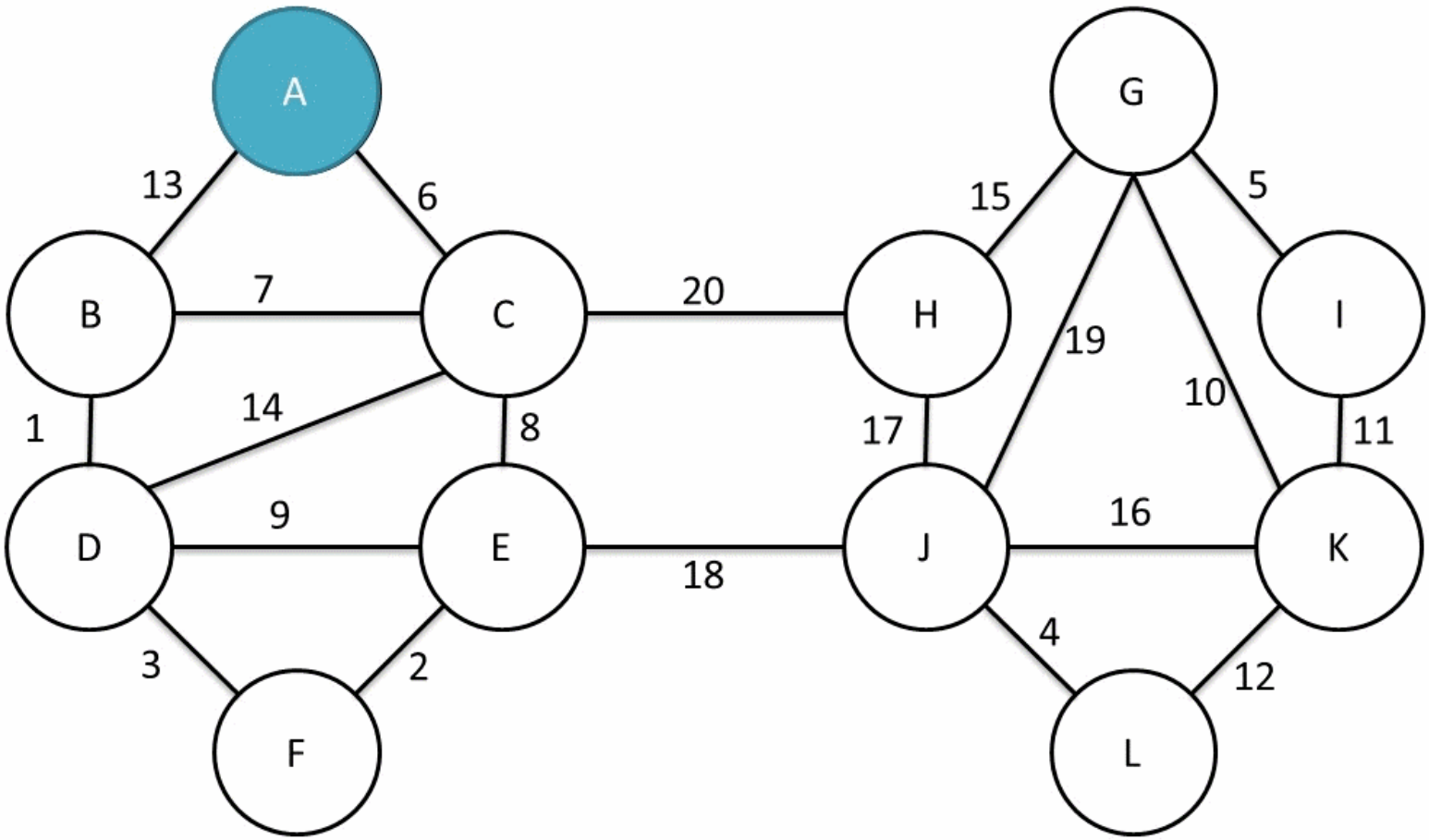
Some edges may be chosen for two trees at the same time. We will obviously pick it only once.

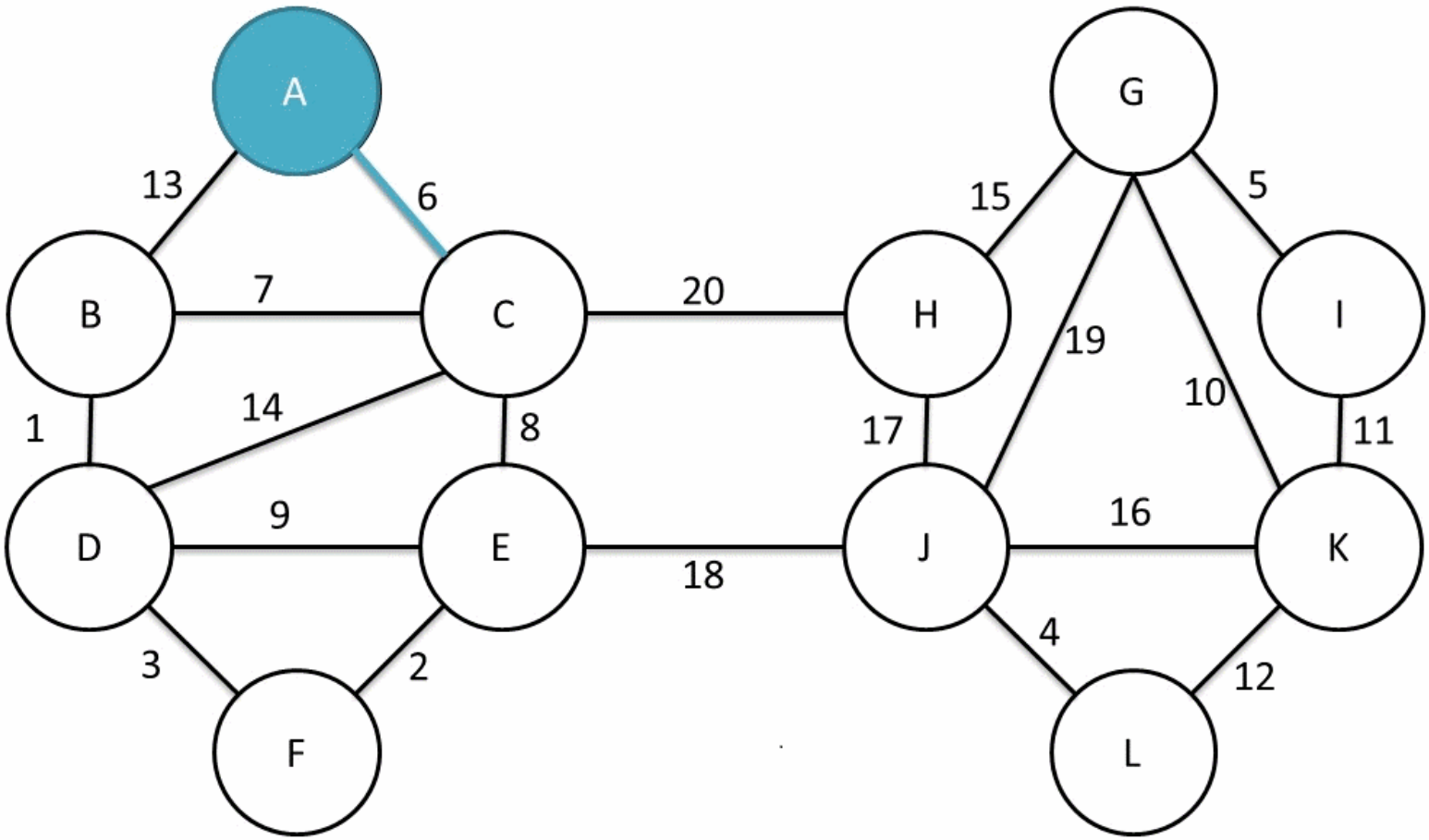
Intuition

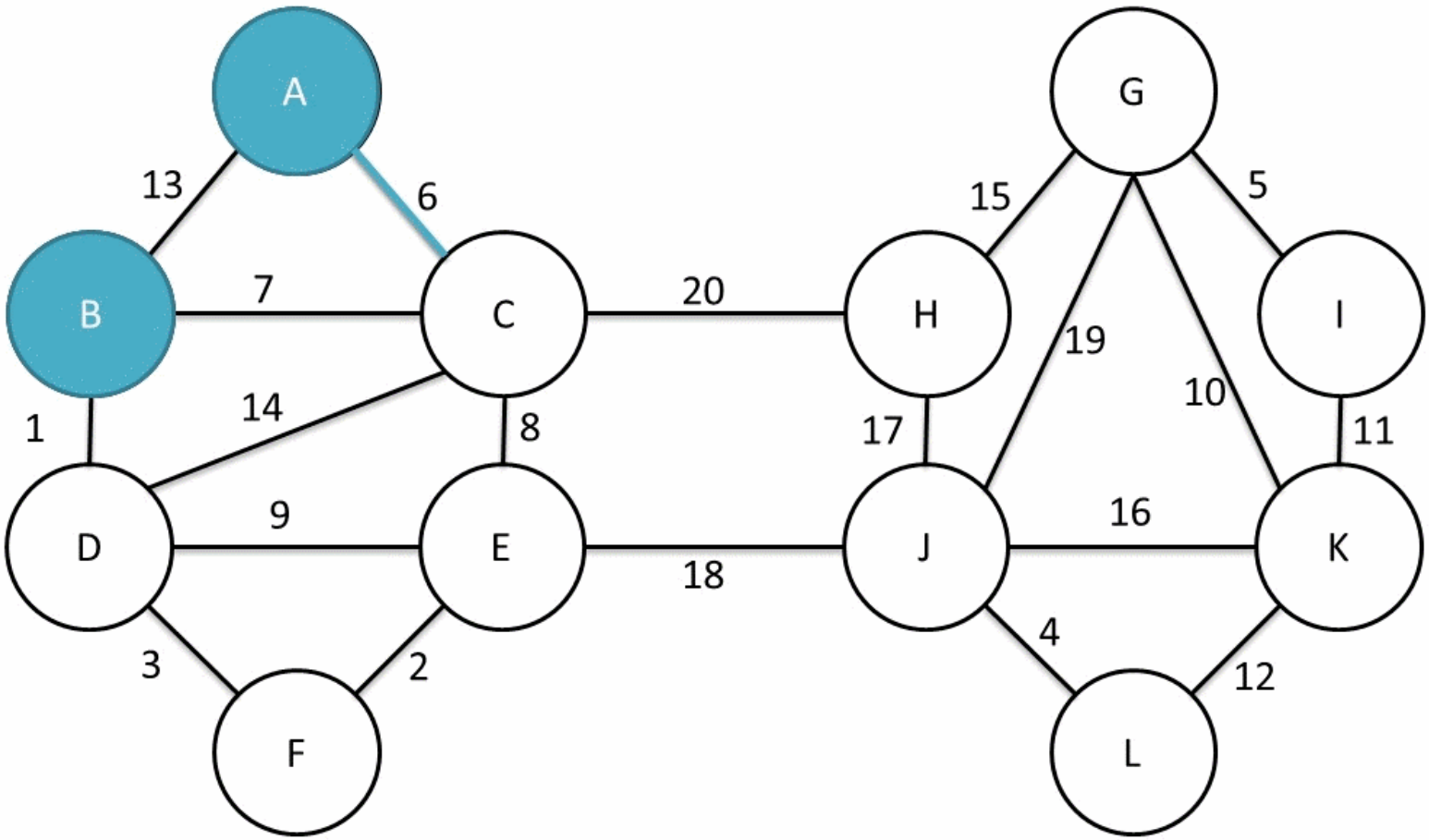
At each step, we are merging two trees, so we will end up with a tree after some steps.
Also, note that we never create a cycle, which is key to the algorithm's effectiveness.

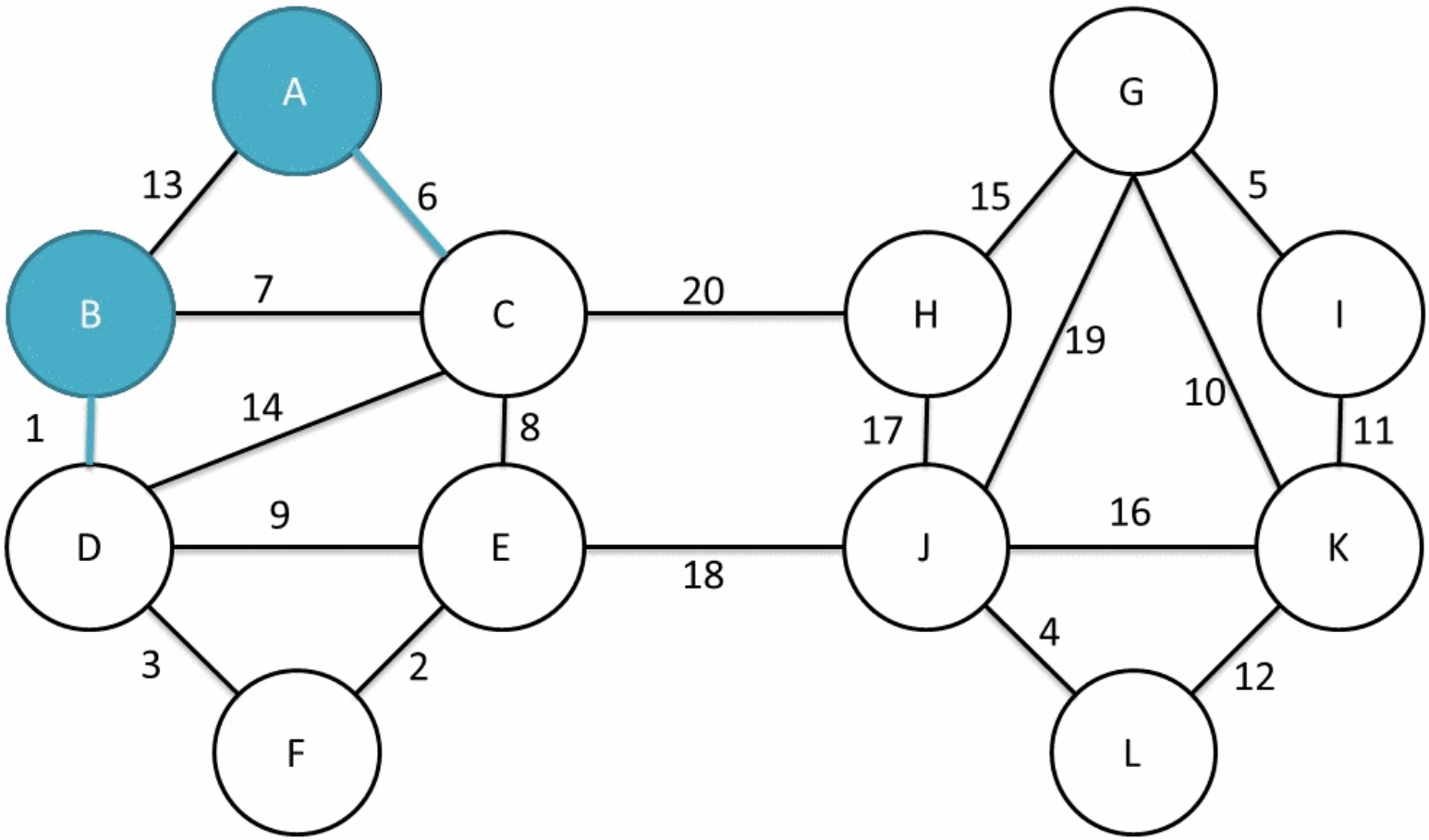
Here is an example of how the algorithm works:

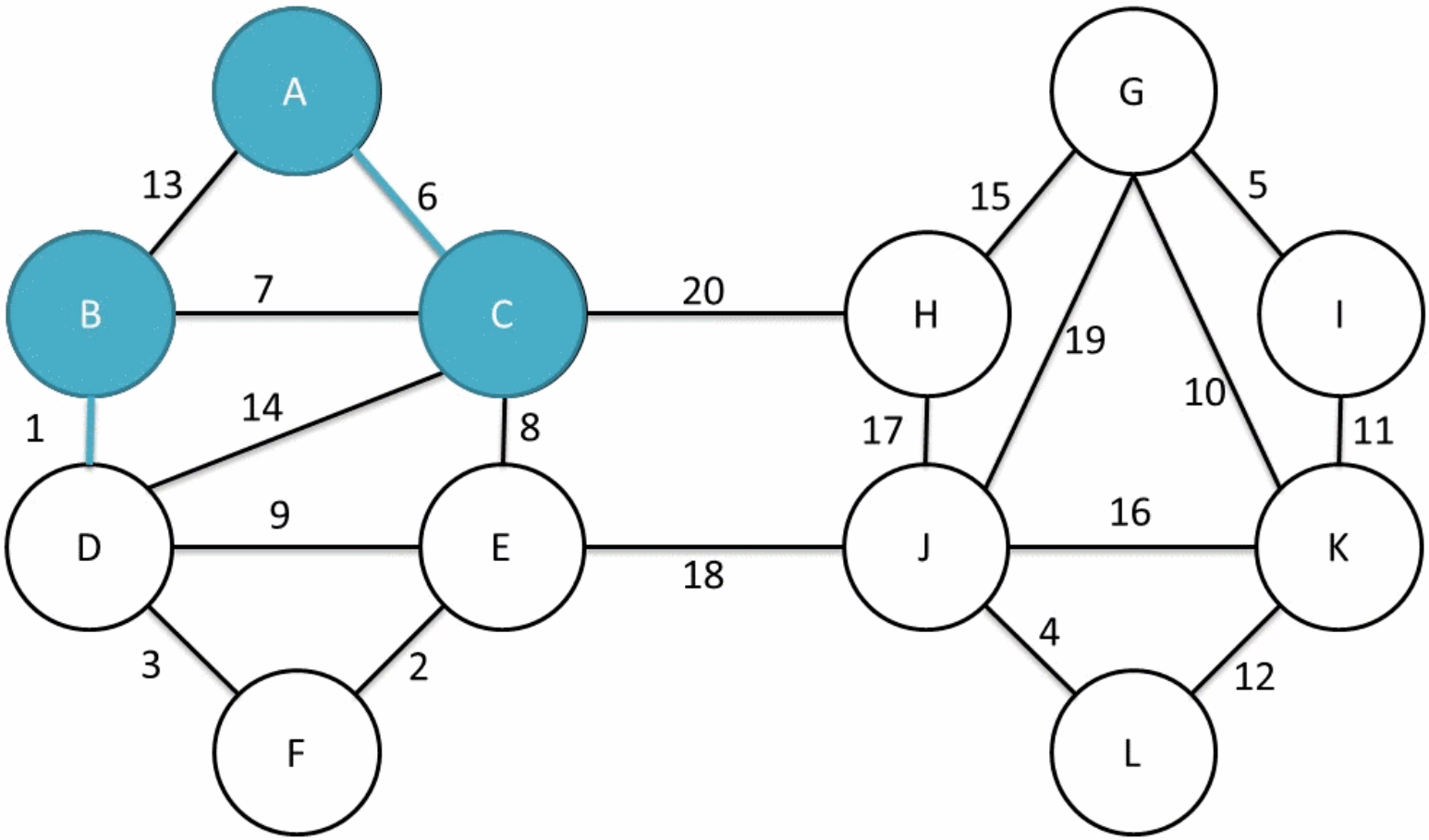


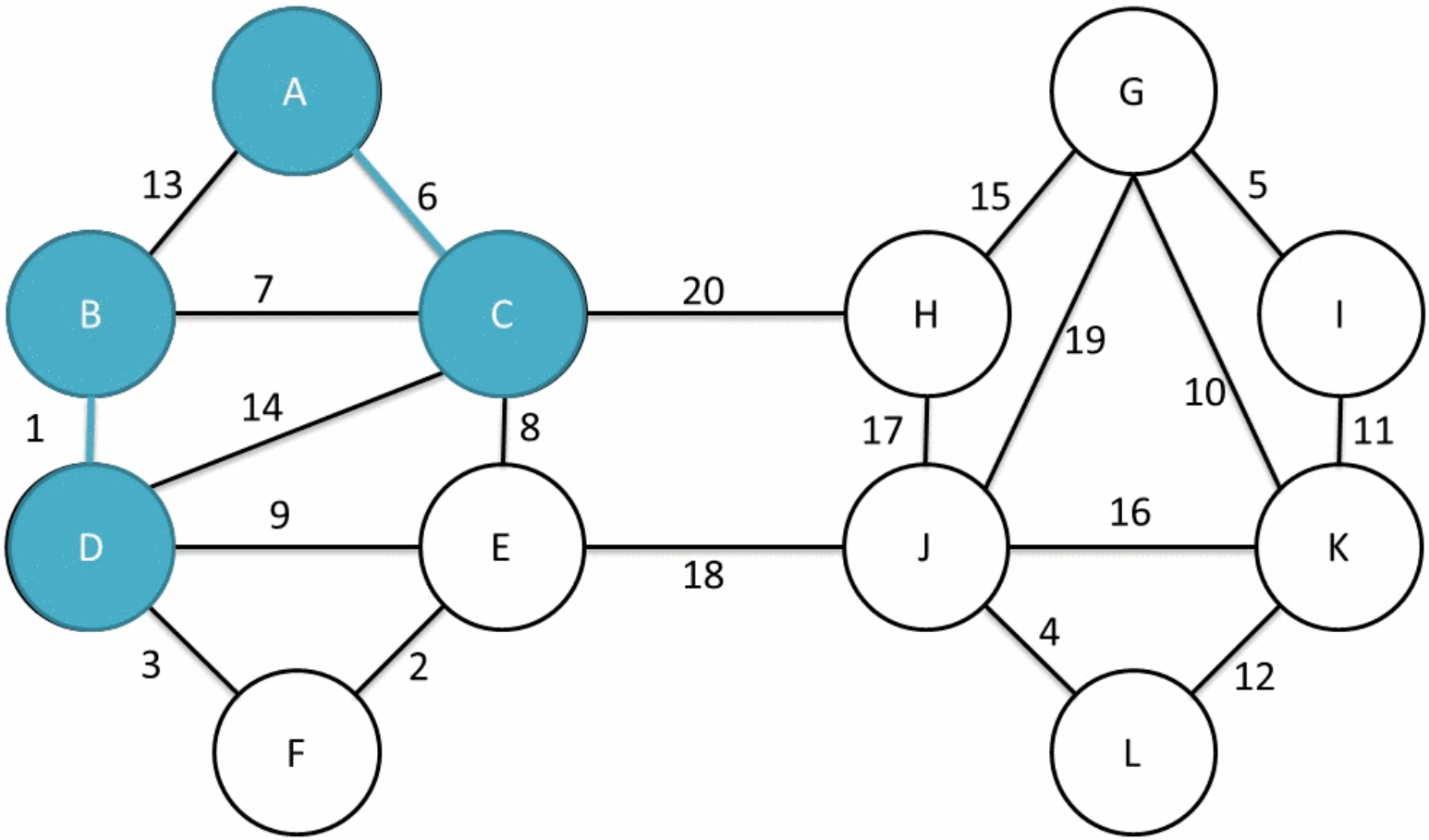

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
5
19
10
11
9
16
18
3
2
4
12

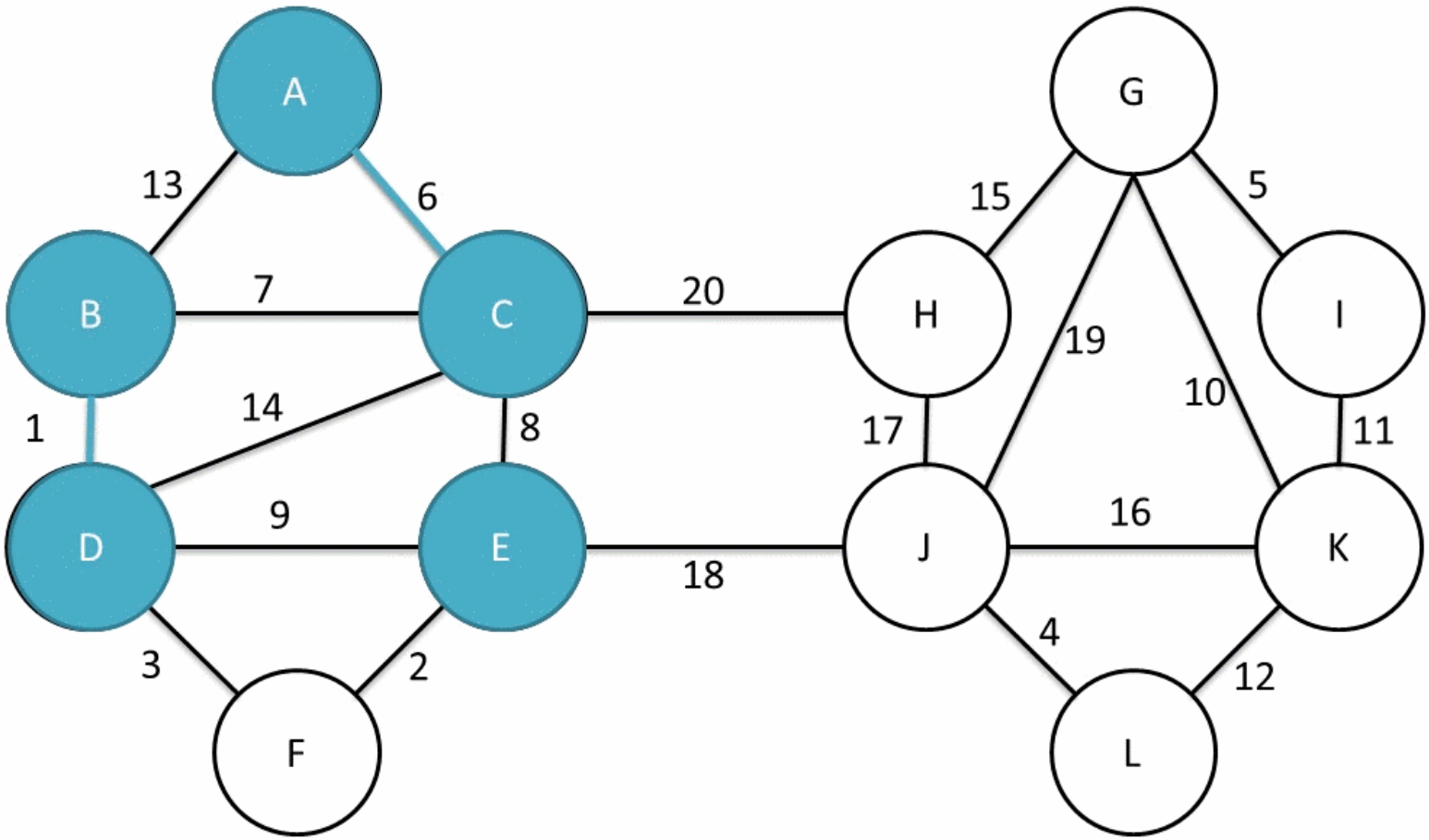

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
5
19
10
11
9
16
18
3
2
4
12

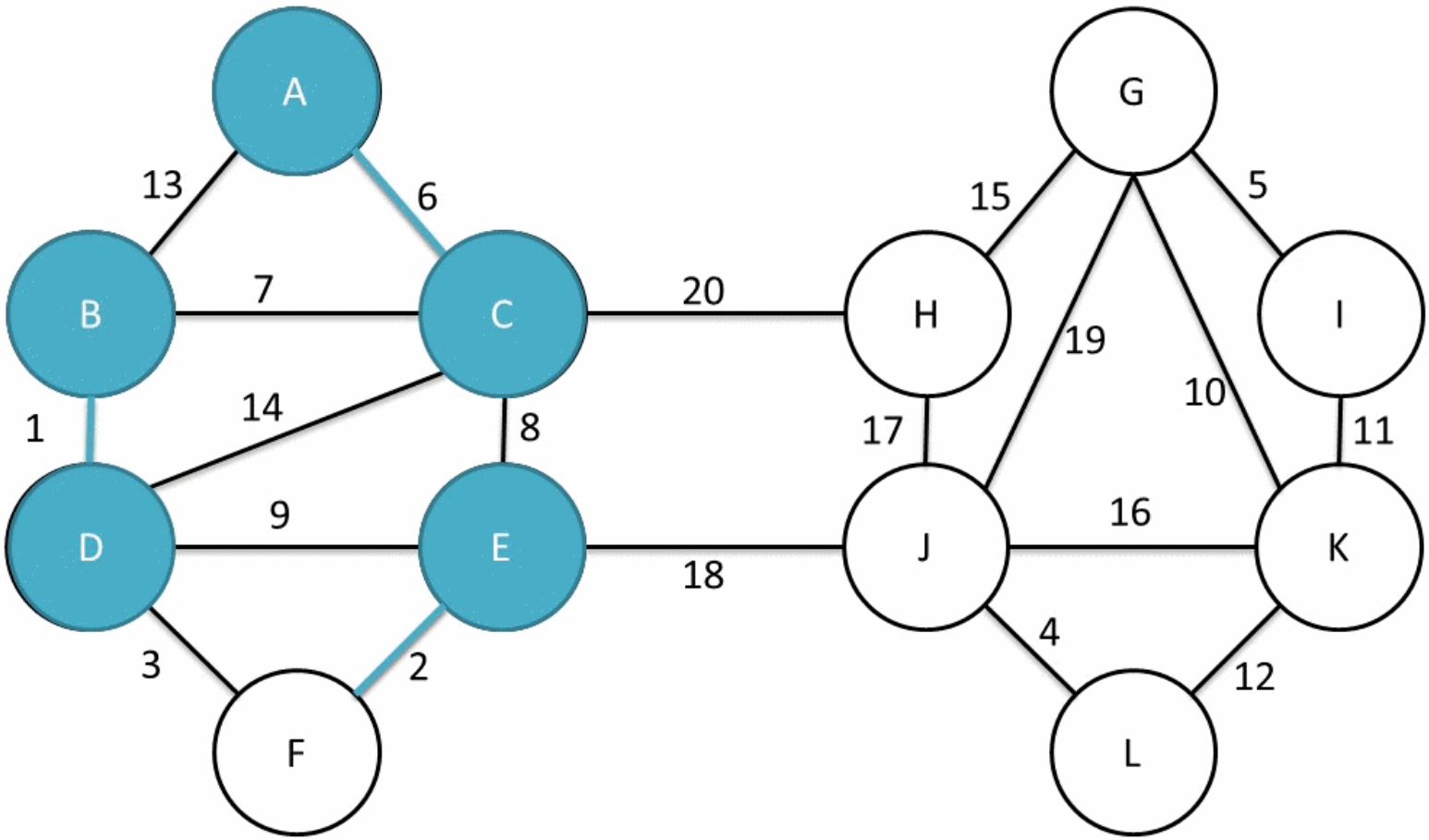

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
19
5
10
11
9
16
18
3
2
4
12

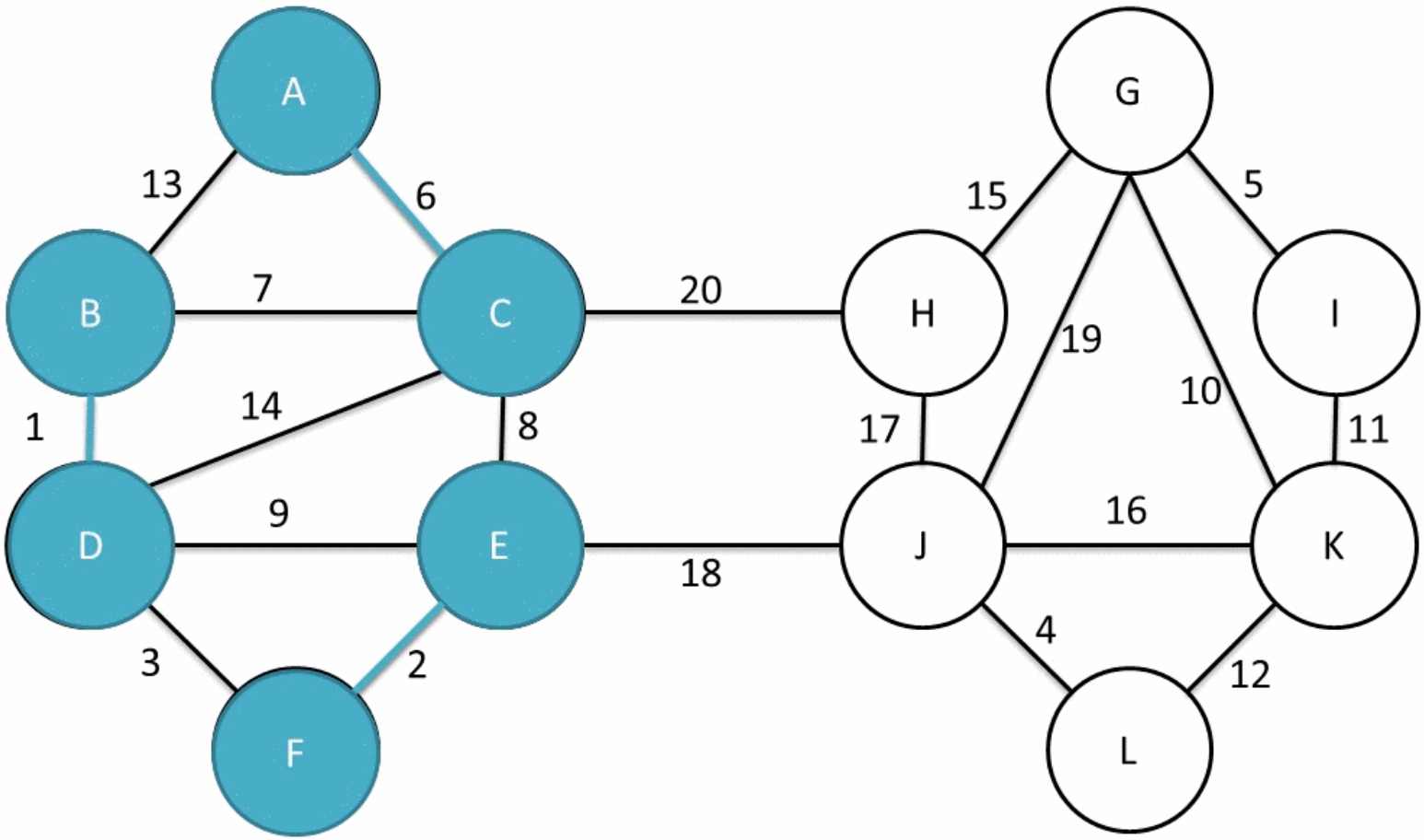

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
5
19
10
11
9
16
18
4
3
2
12

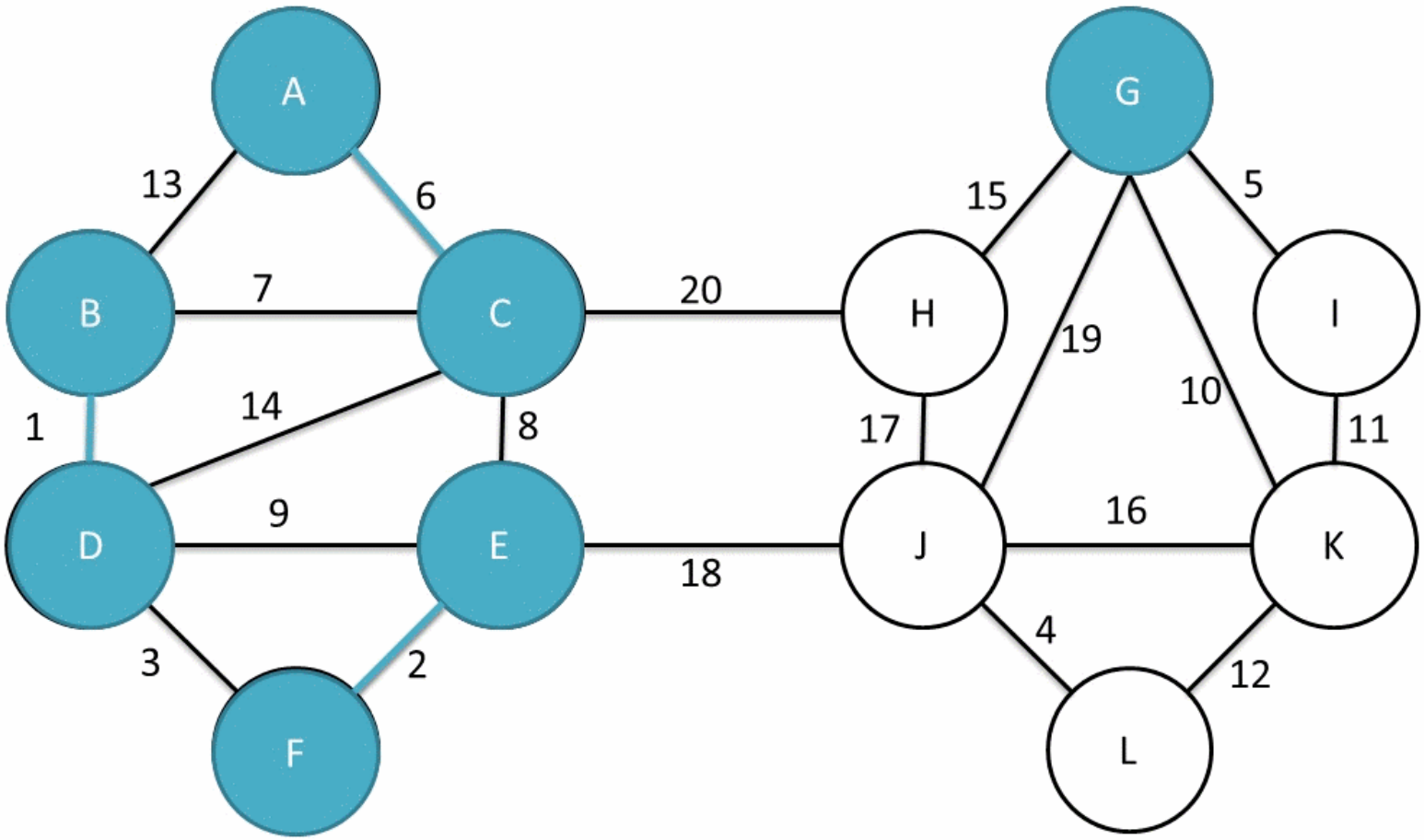

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
19
5
10
11
9
16
18
3
2
4
12

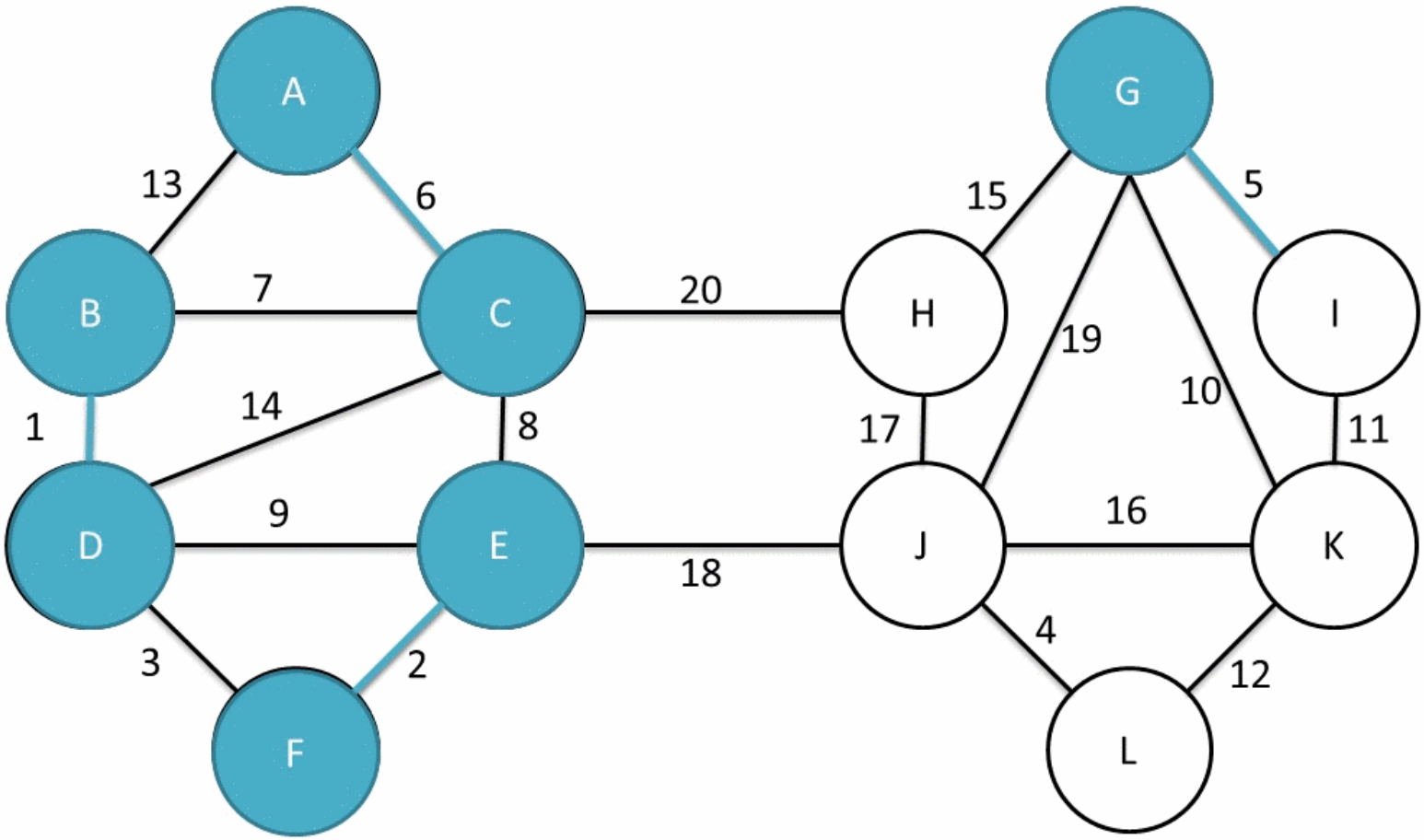

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
17
15
19
5
10
11
9
16
18
3
2
4
12

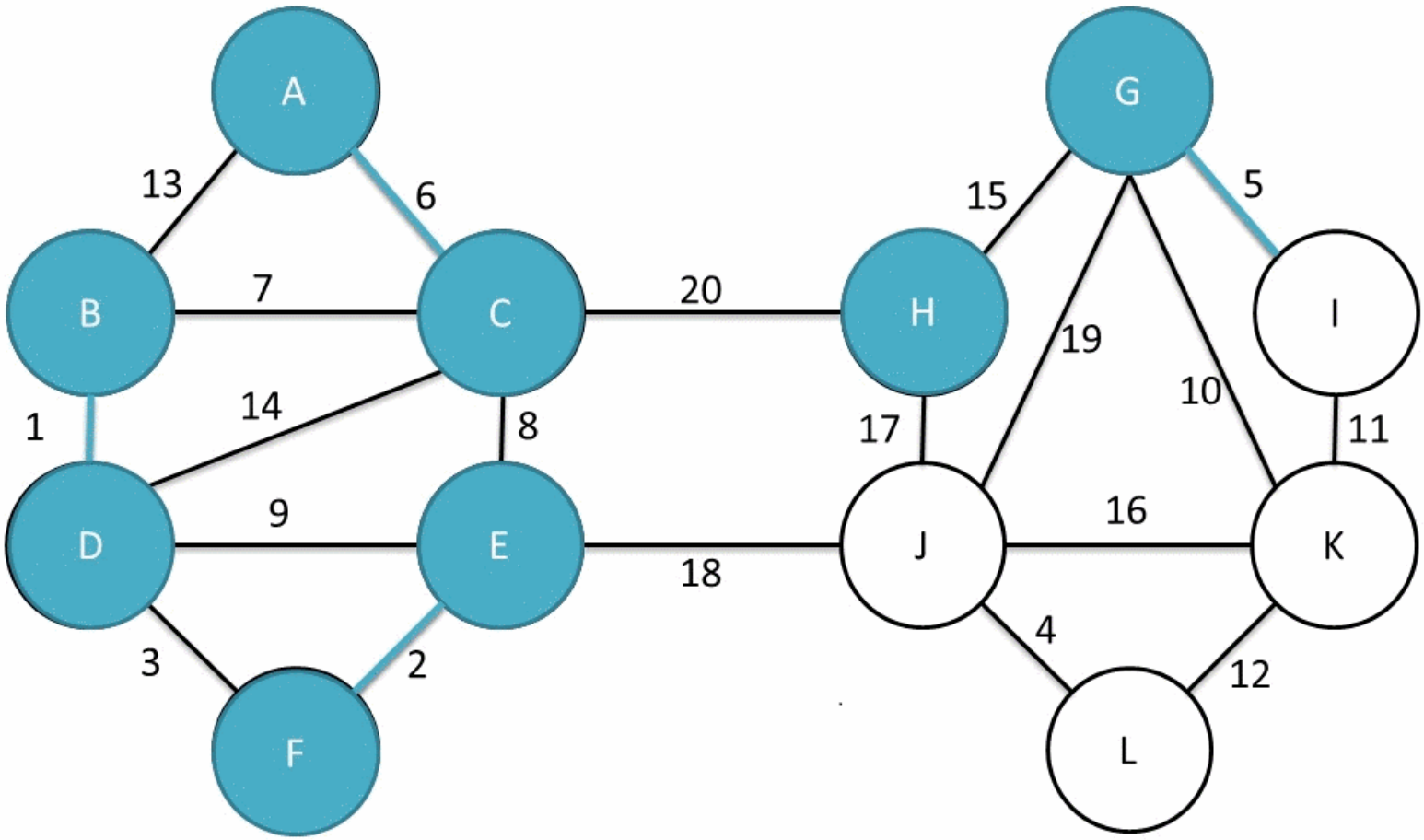

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
18
3
2
15
5
19
10
17
11
16
4
12

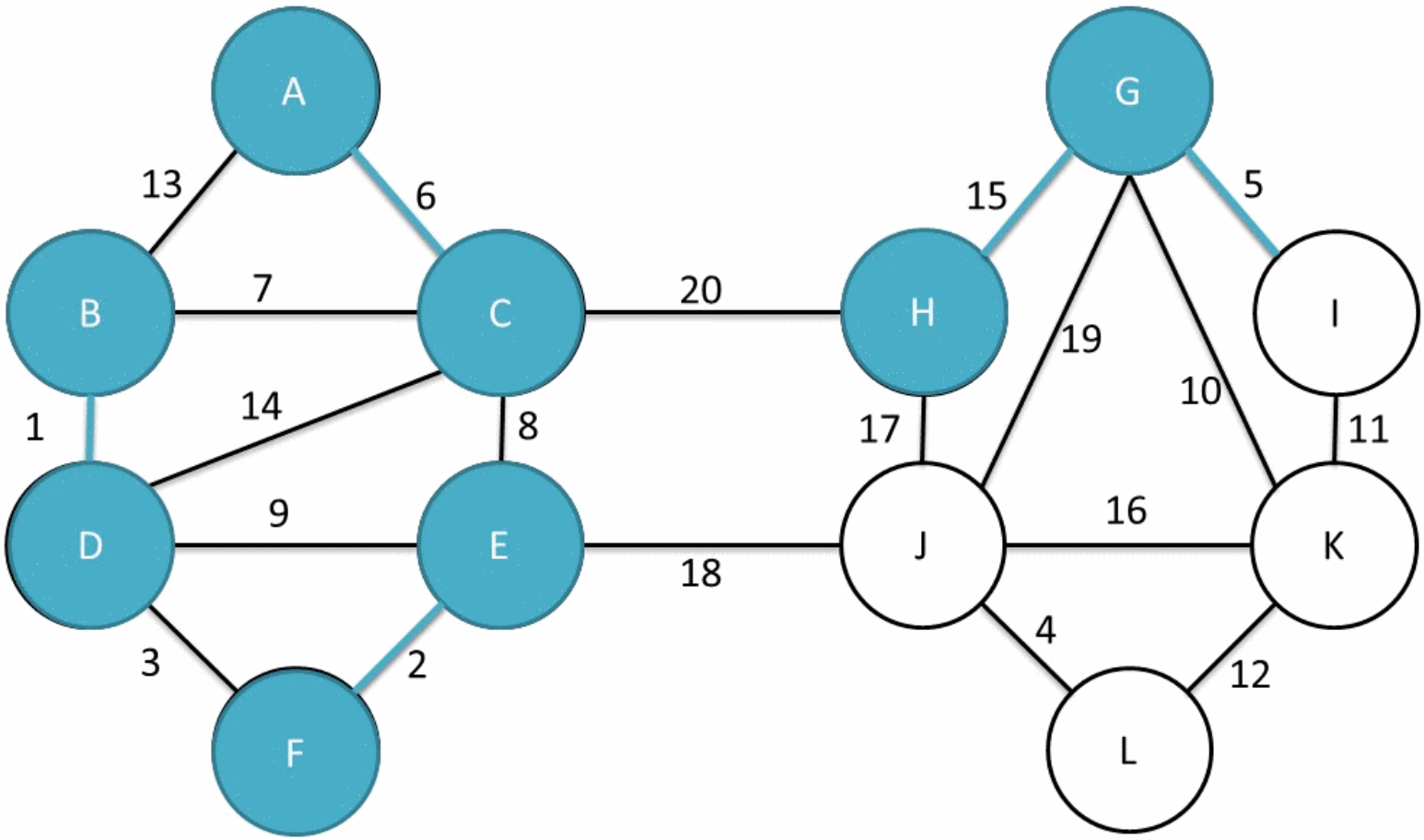

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
18
3
2
15
5
19
10
17
11
16
4
12

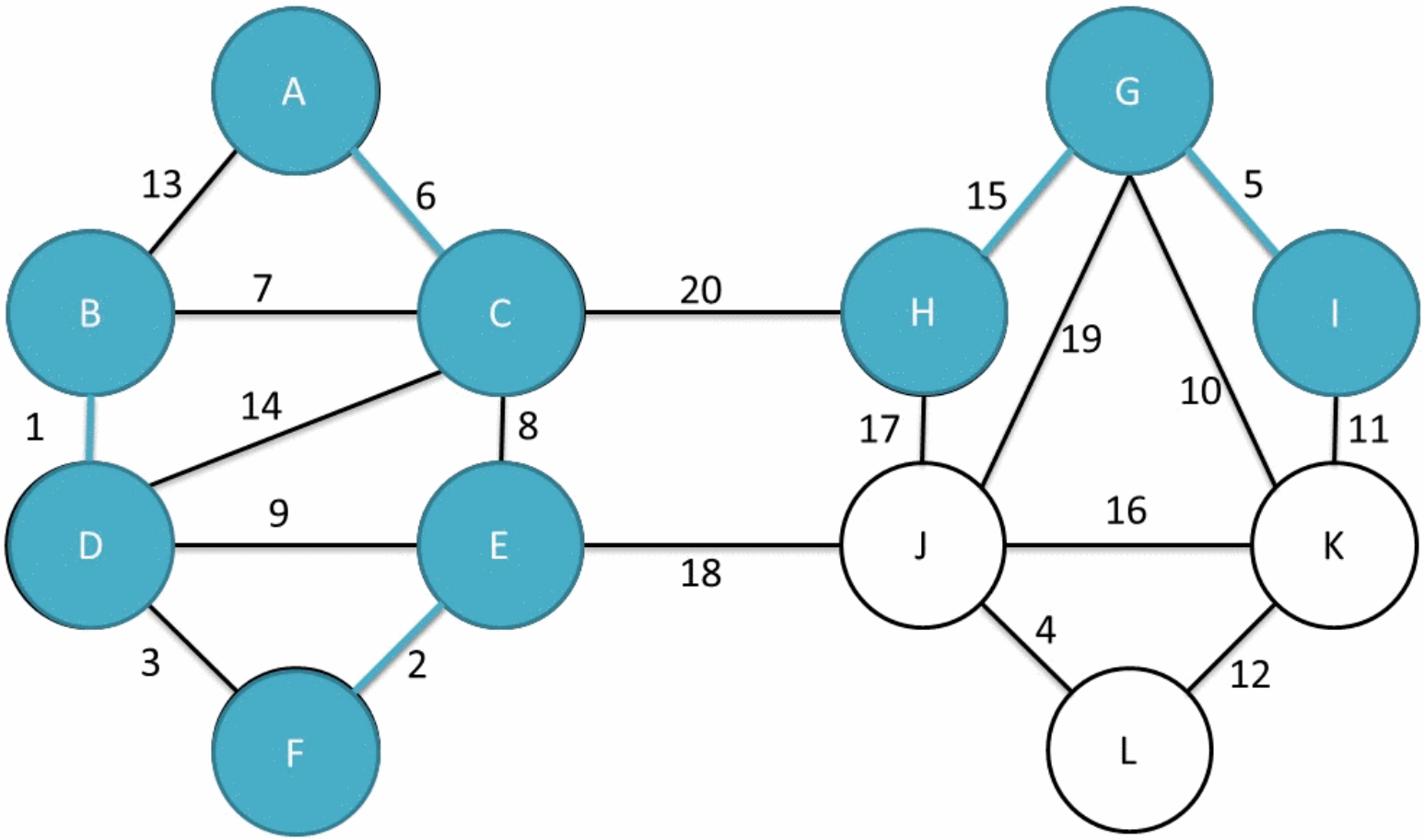

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
18
3
2
15
5
19
10
17
11
16
4
12

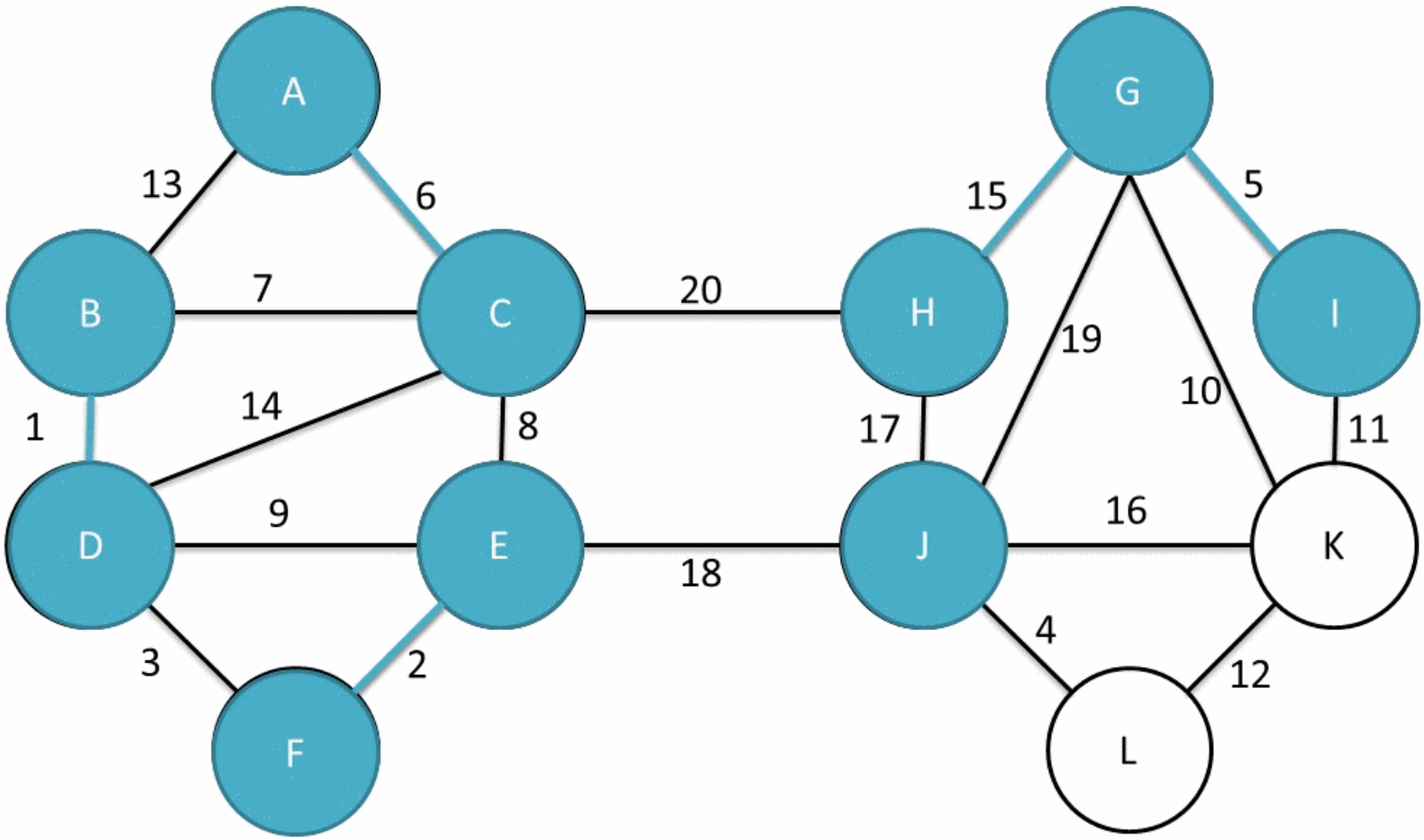

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
18
3
2
15
5
19
10
17
11
16
4
12

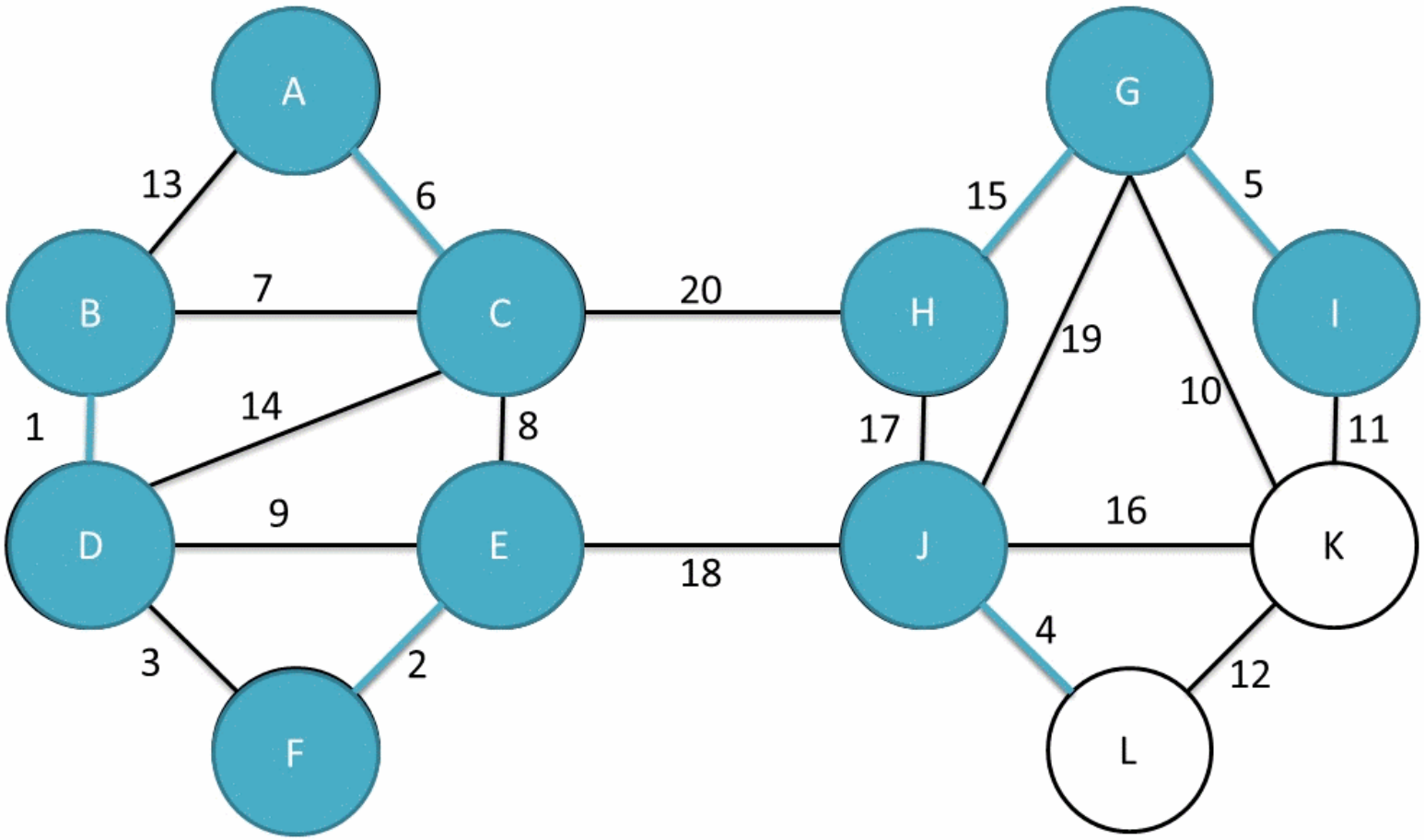

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

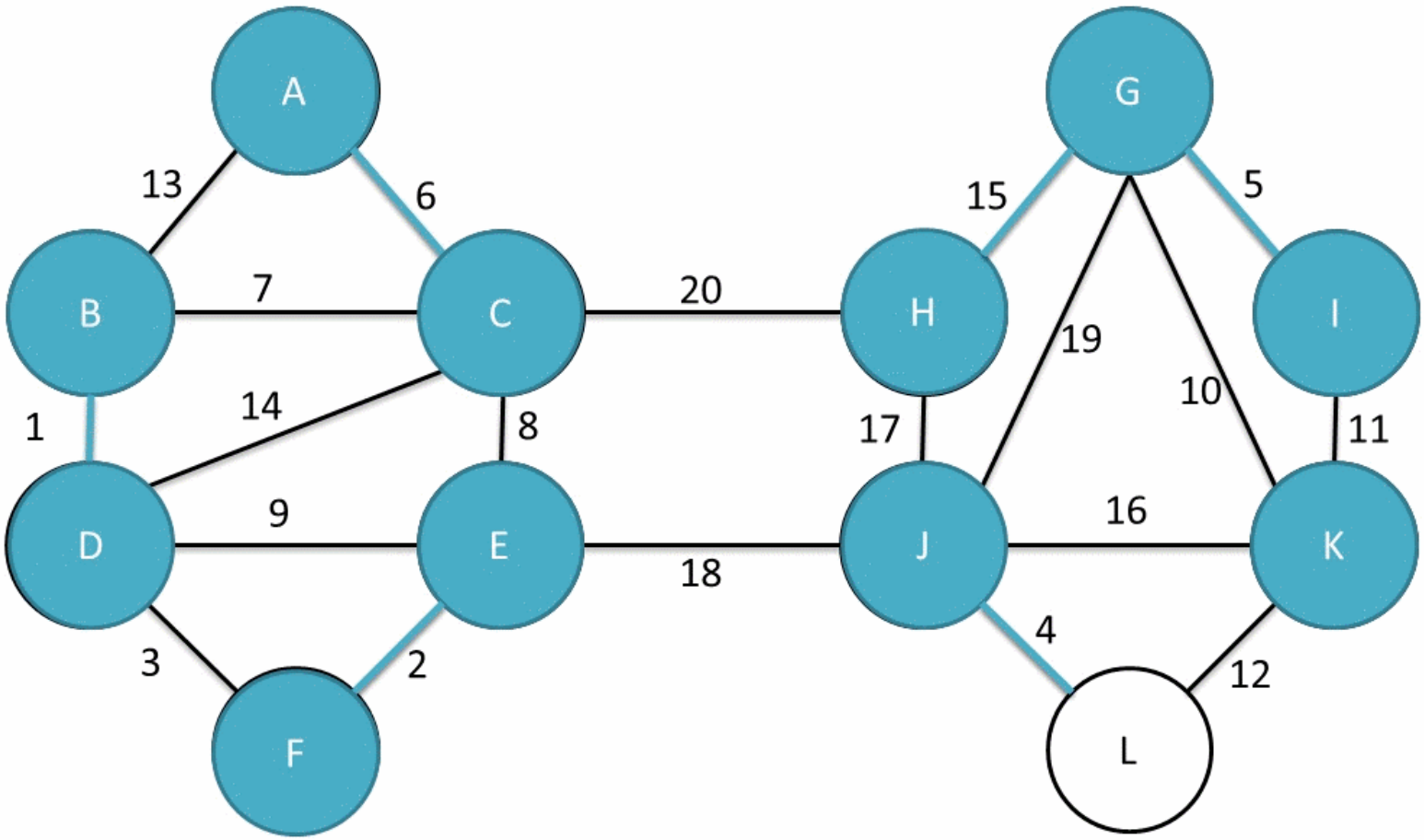

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
18
3
2
15
5
19
10
17
11
16
4
12

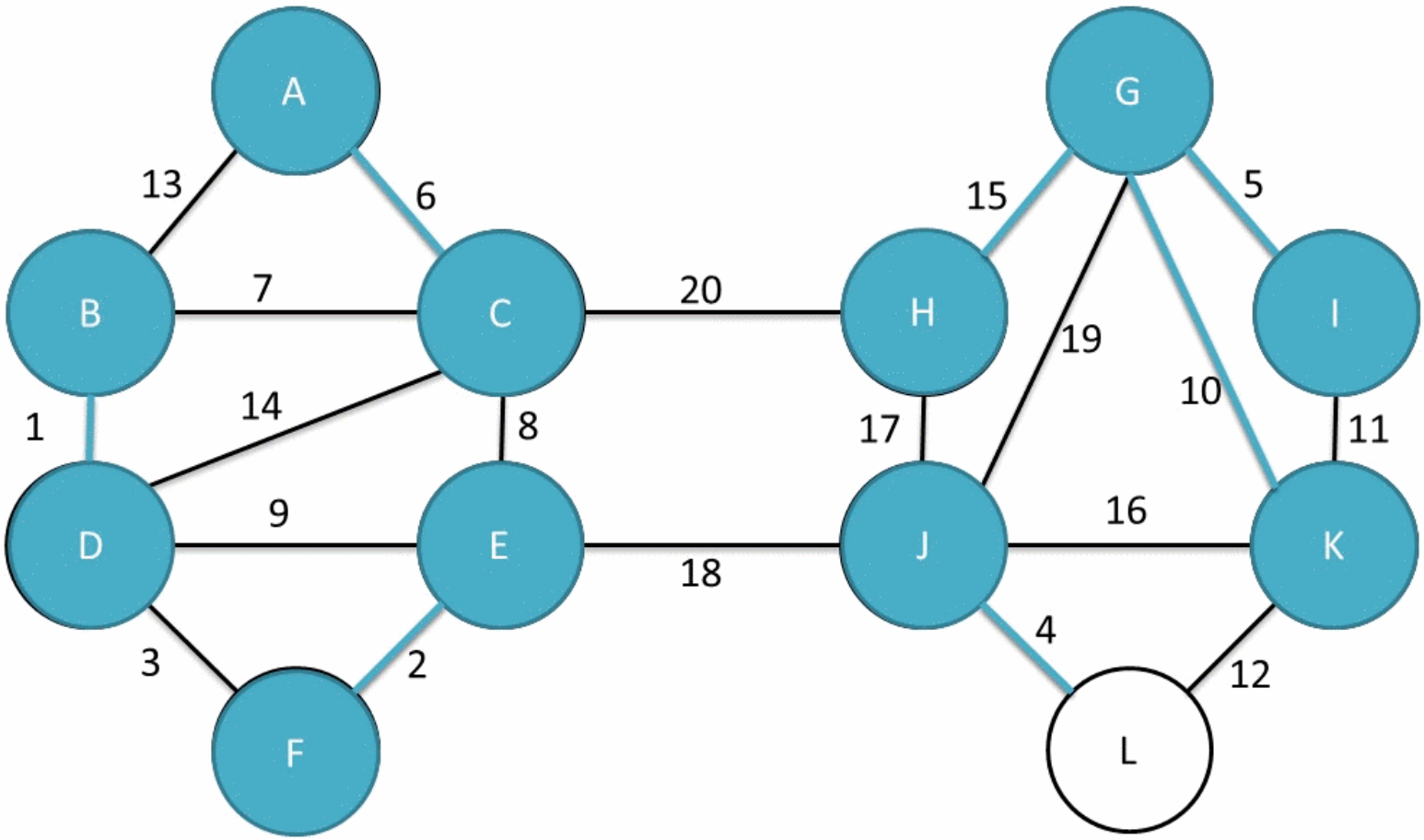

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

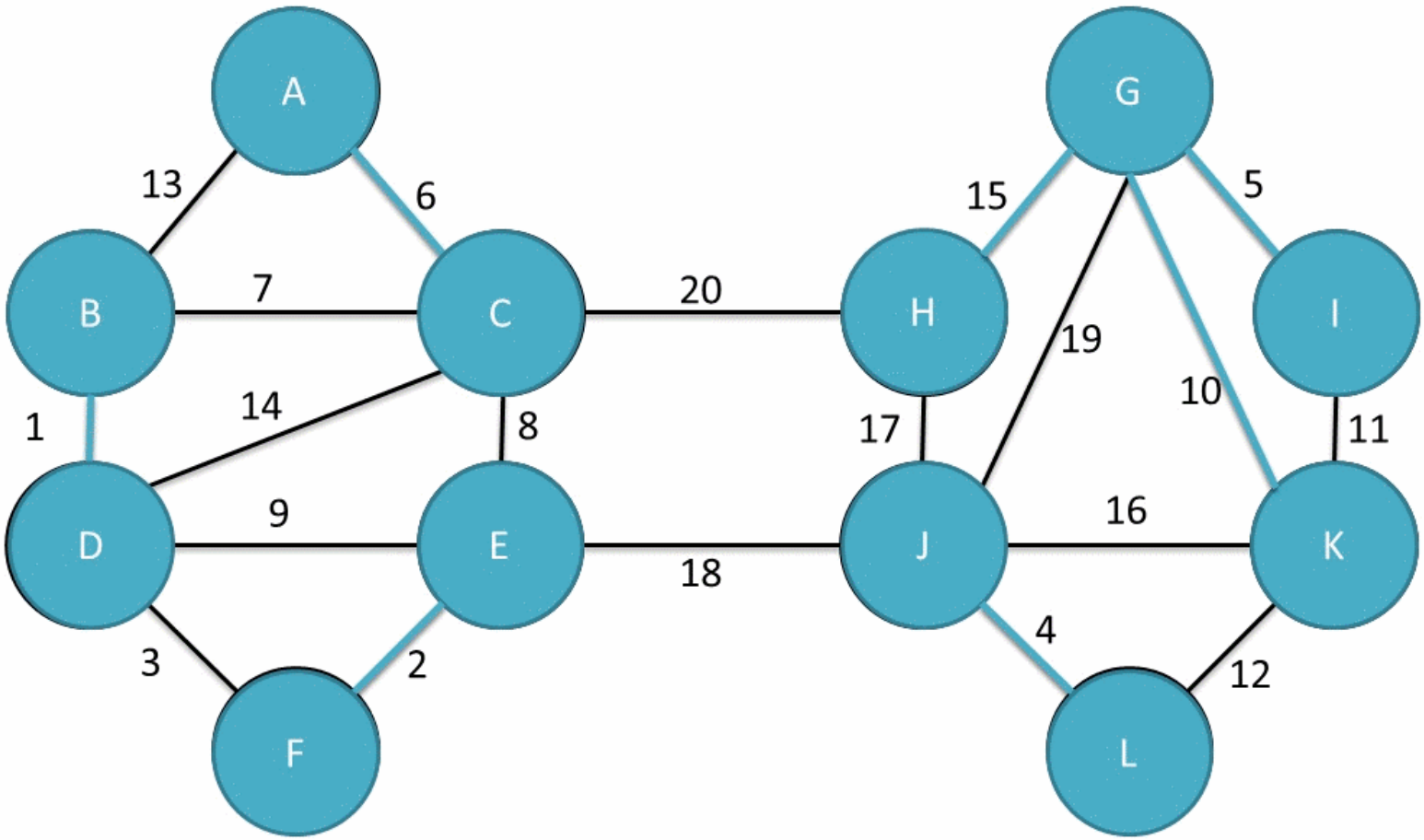

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

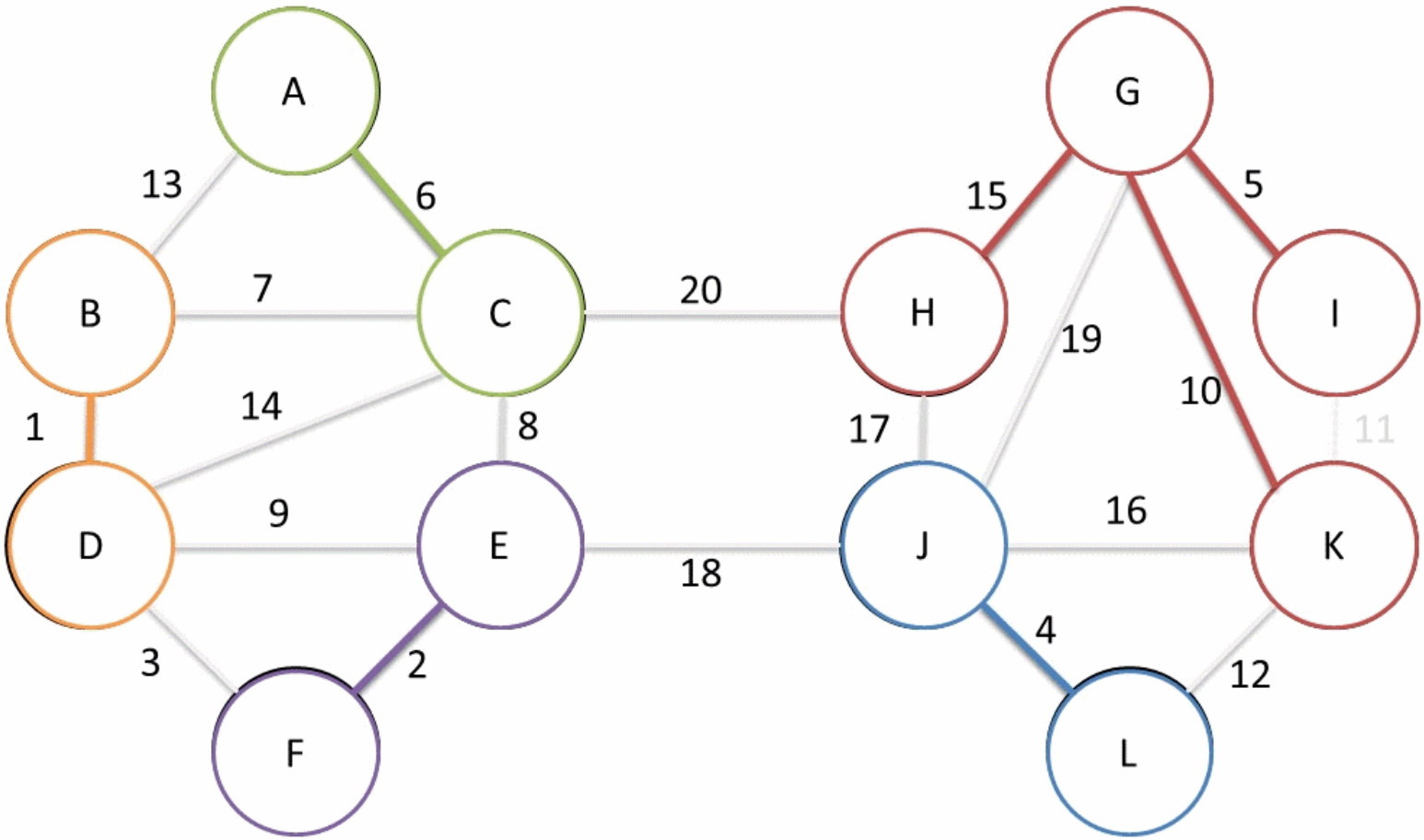

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

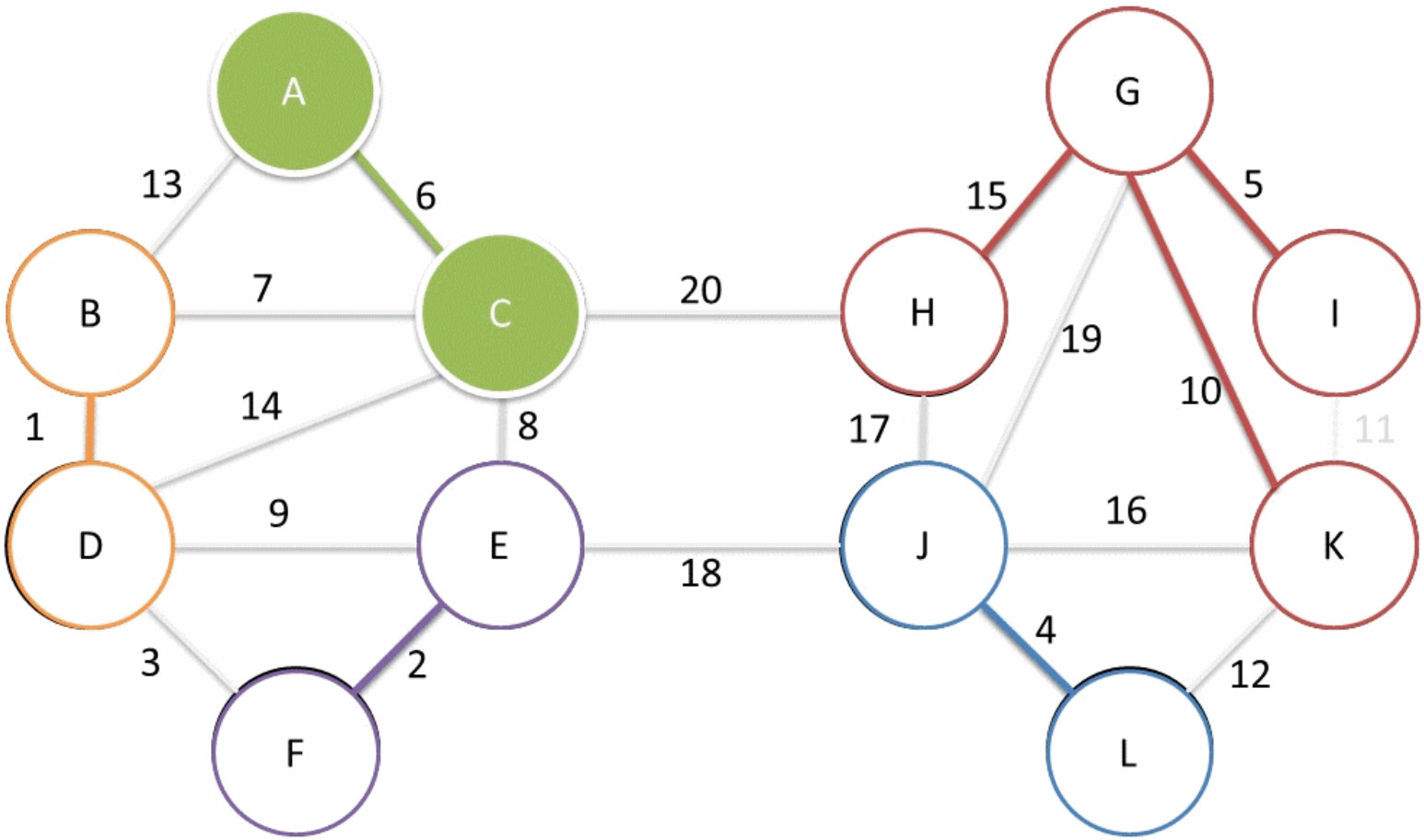

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

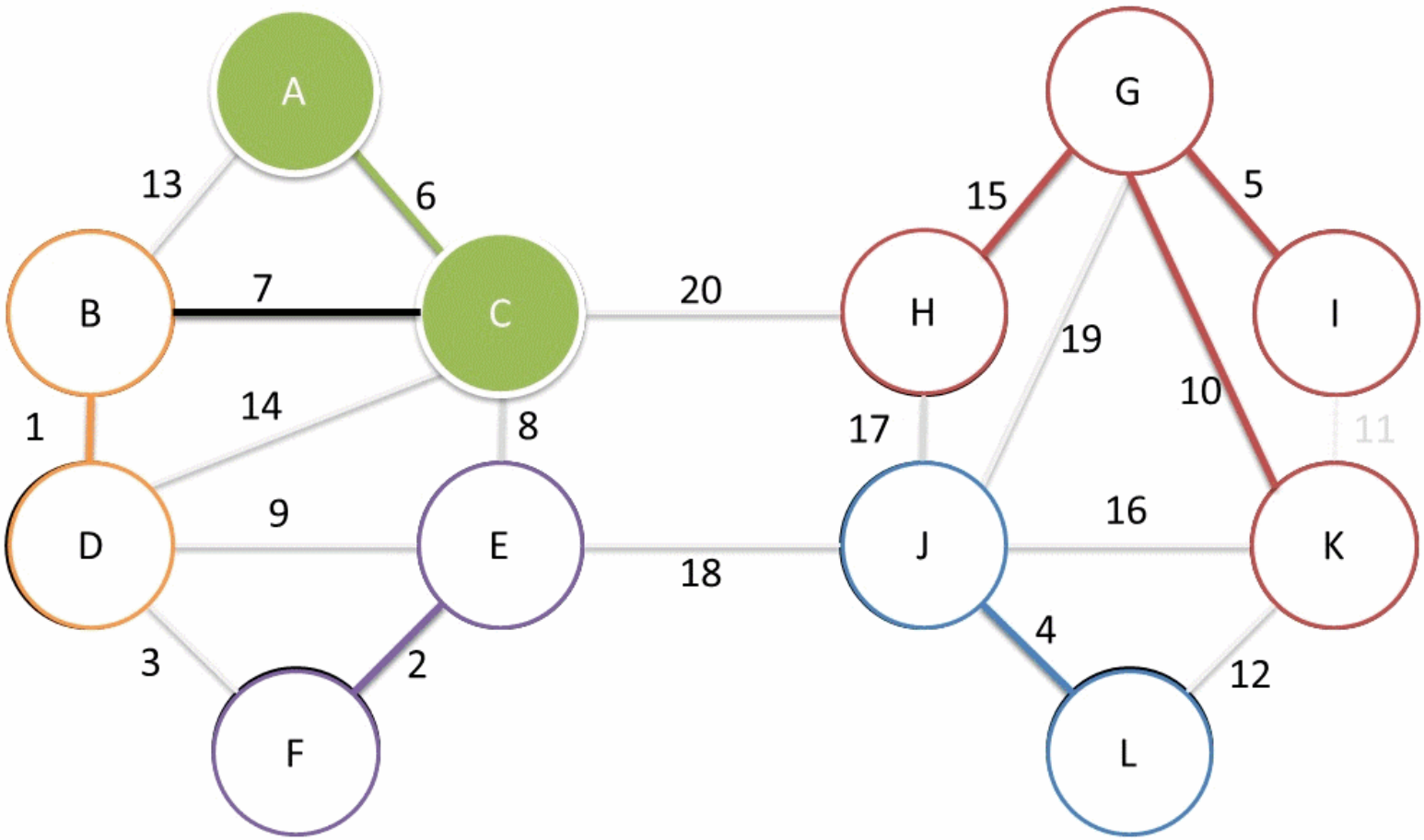

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
18
J
16
K
4
3
2
12
F
L

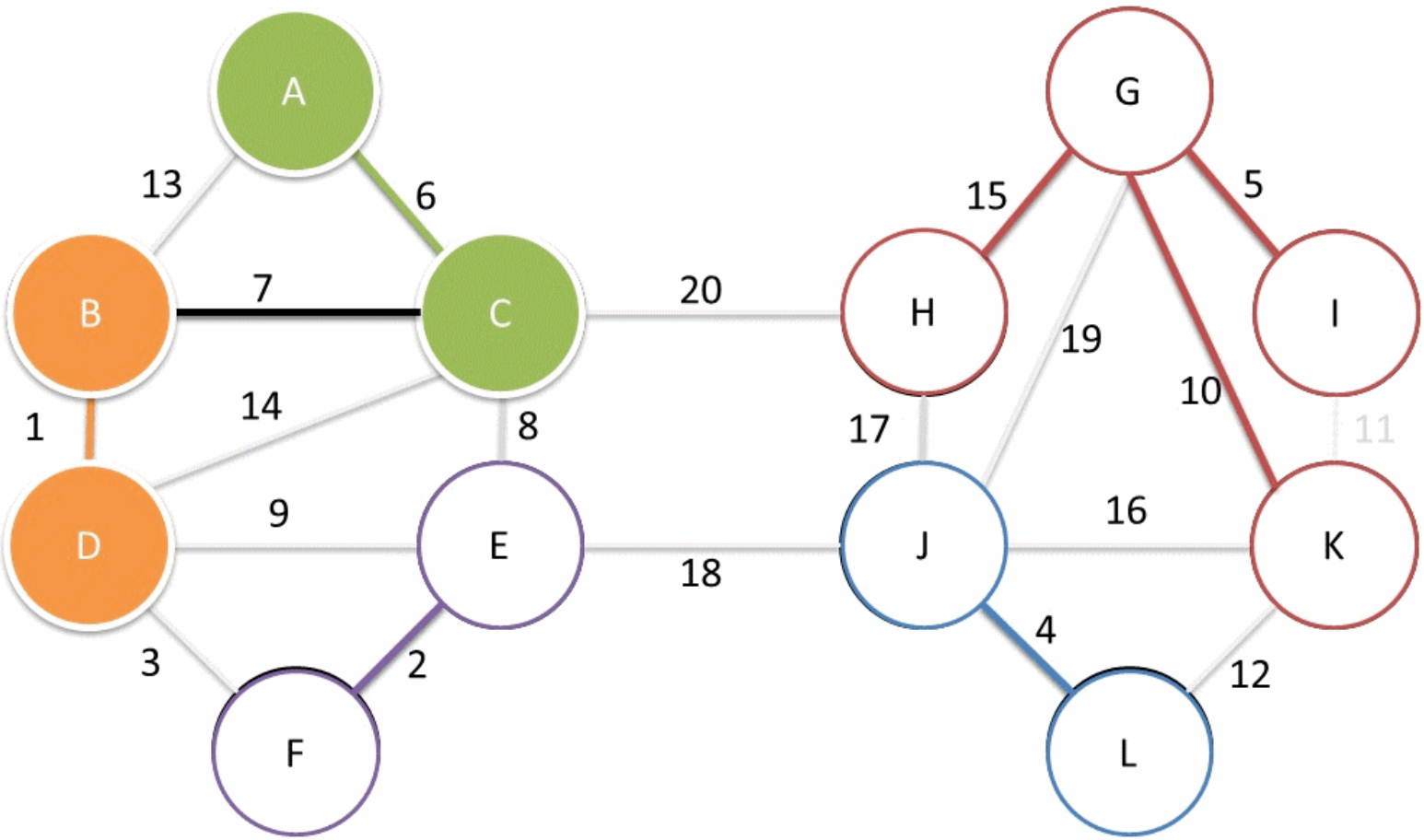

A
G
13
6
15
5
B
7
C
20
H
I
19
14
10
1
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

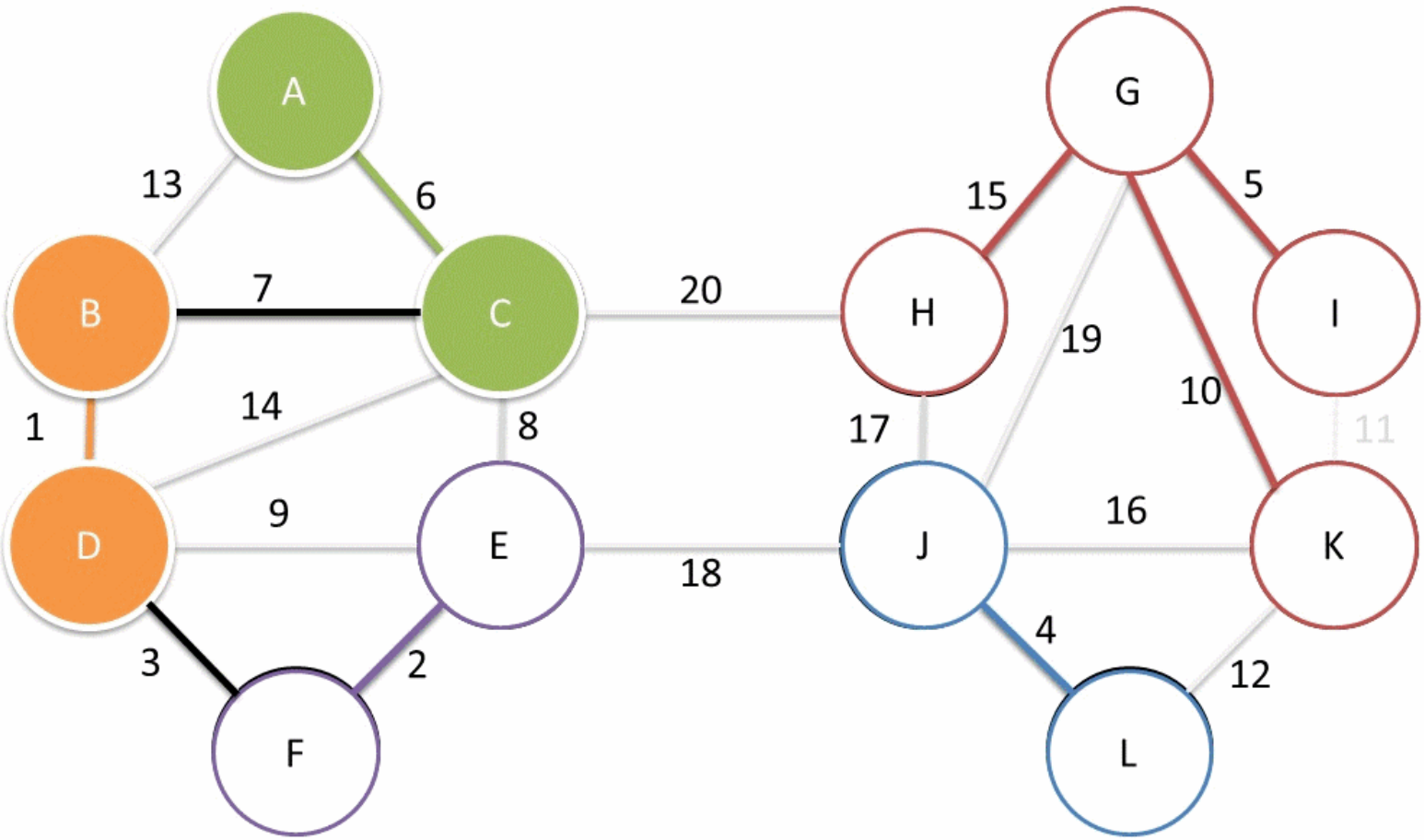

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
18
J
16
K
4
3
2
12
F
L

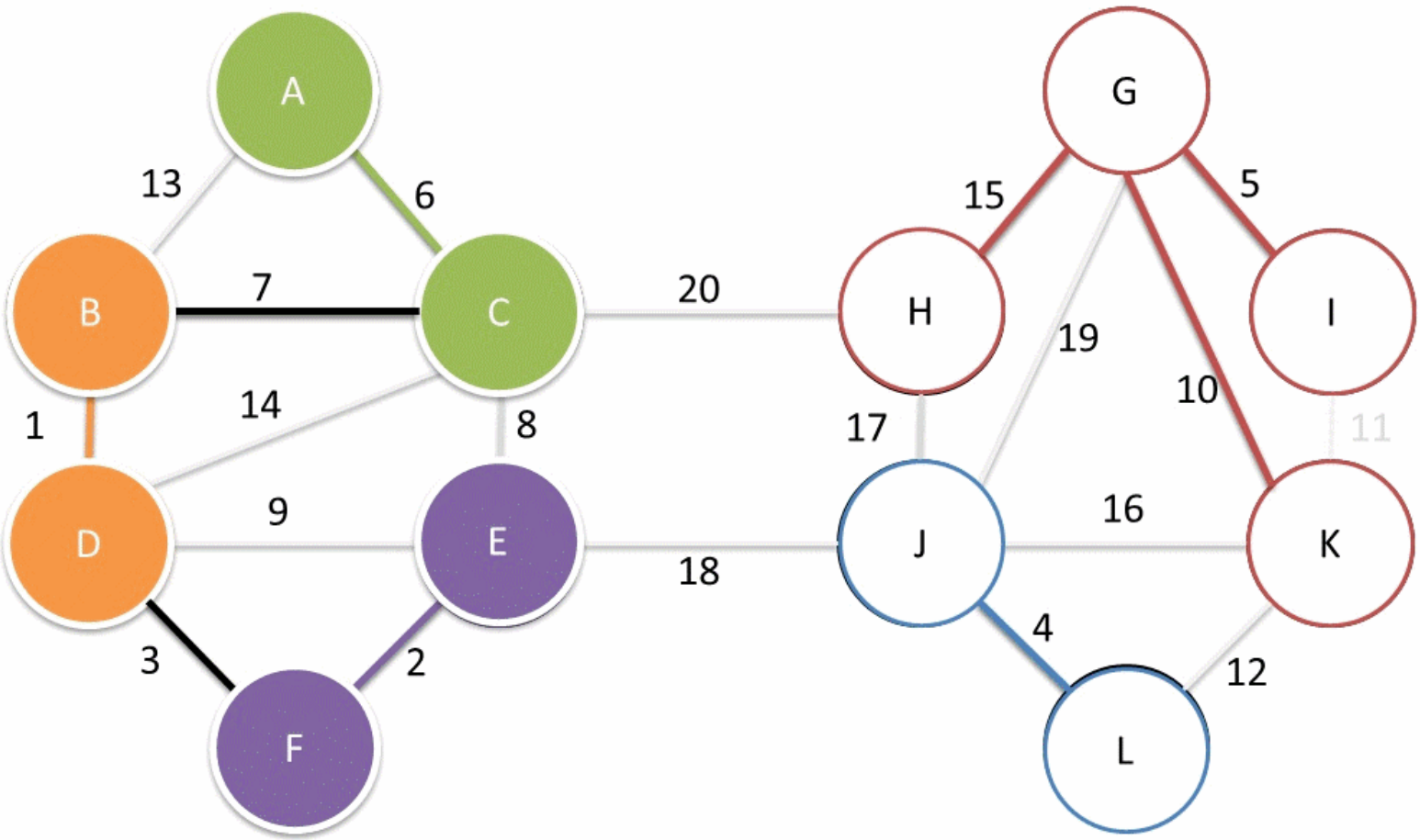

A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
18
J
16
K
4
3
2
12
F
L

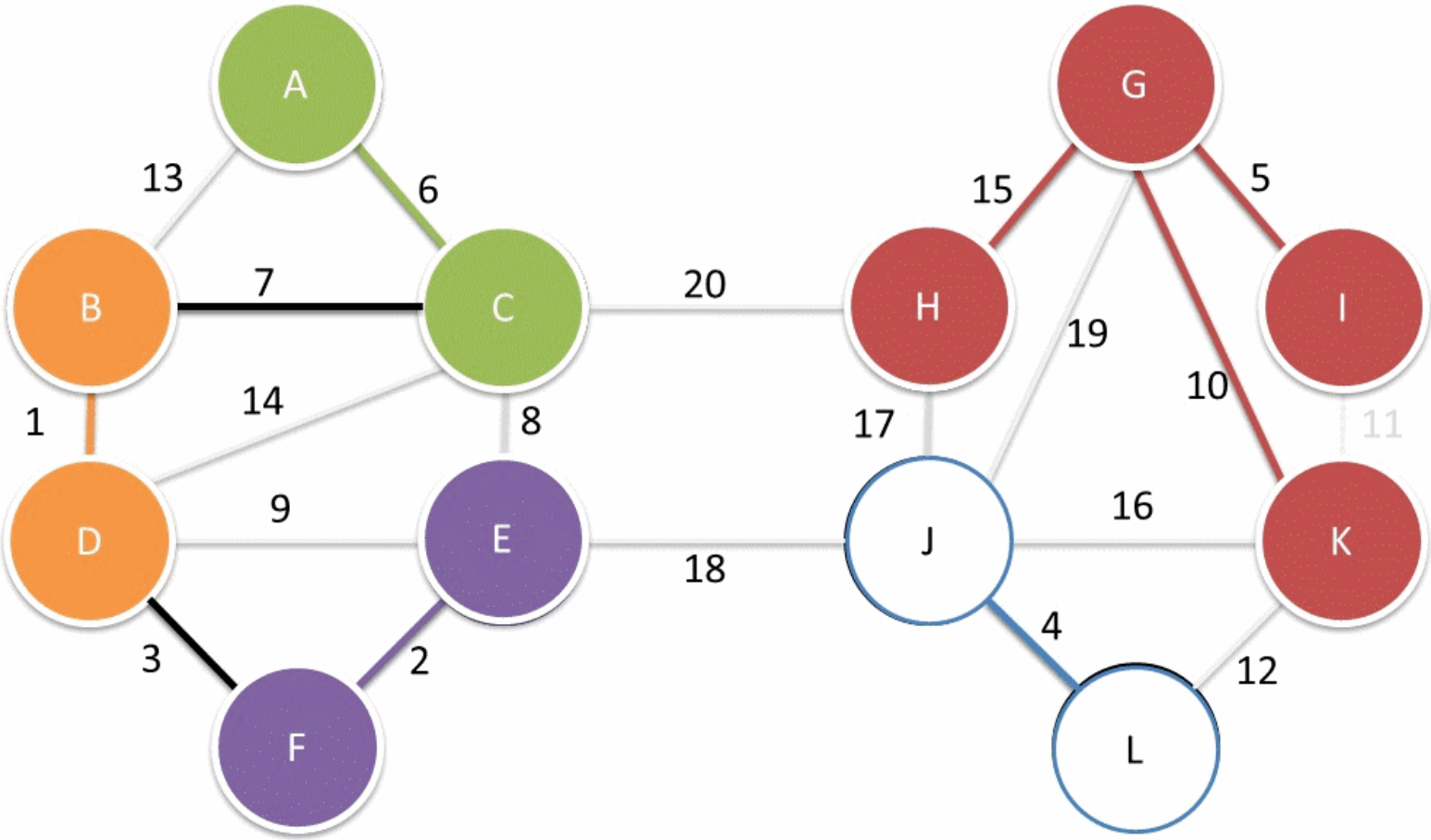

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
14
1
8
9
18
3
2
15
5
19
10
17
11
16
4
12

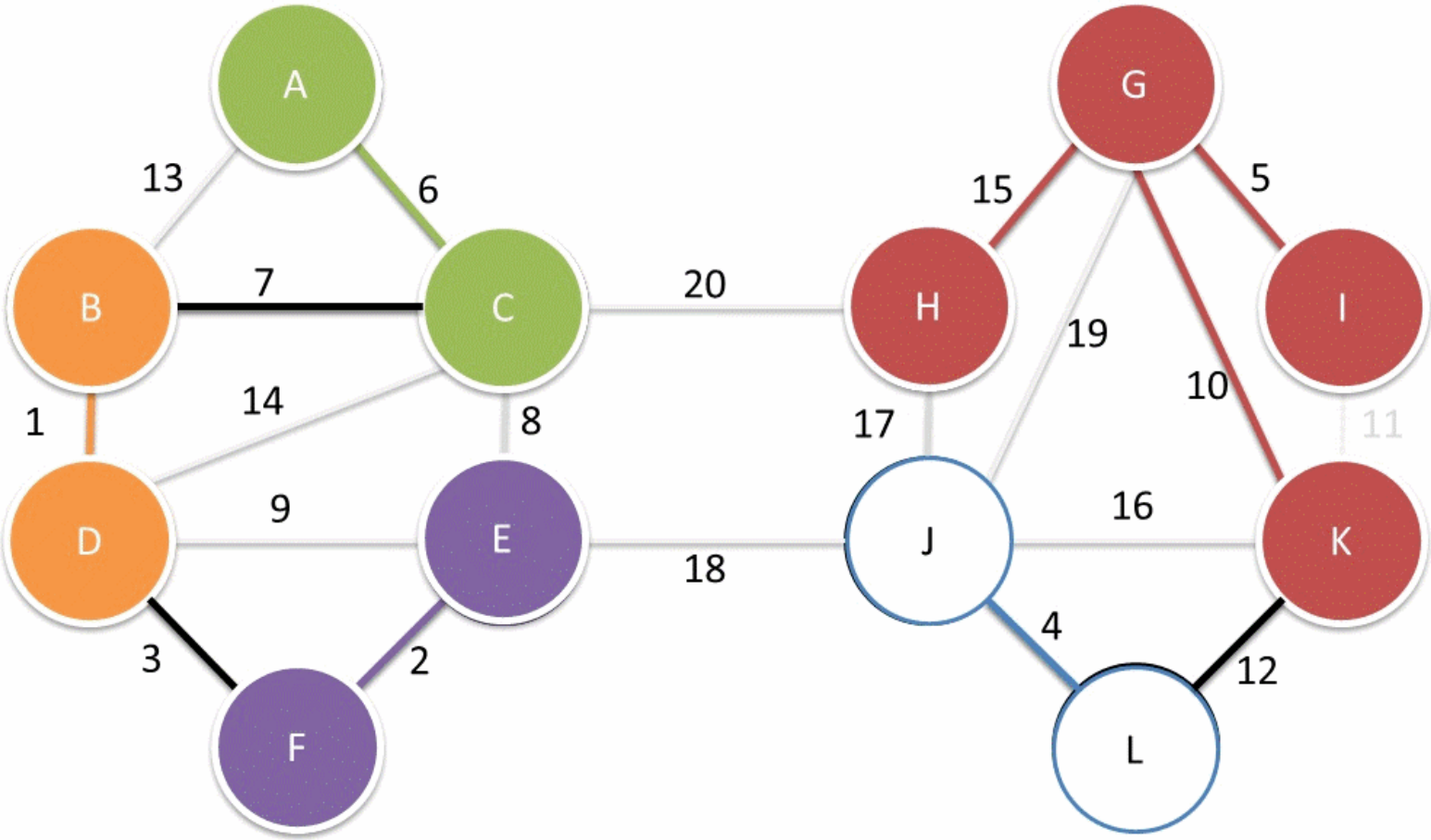

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
8
10
1
17
11
9
16
18
4
3
2
12

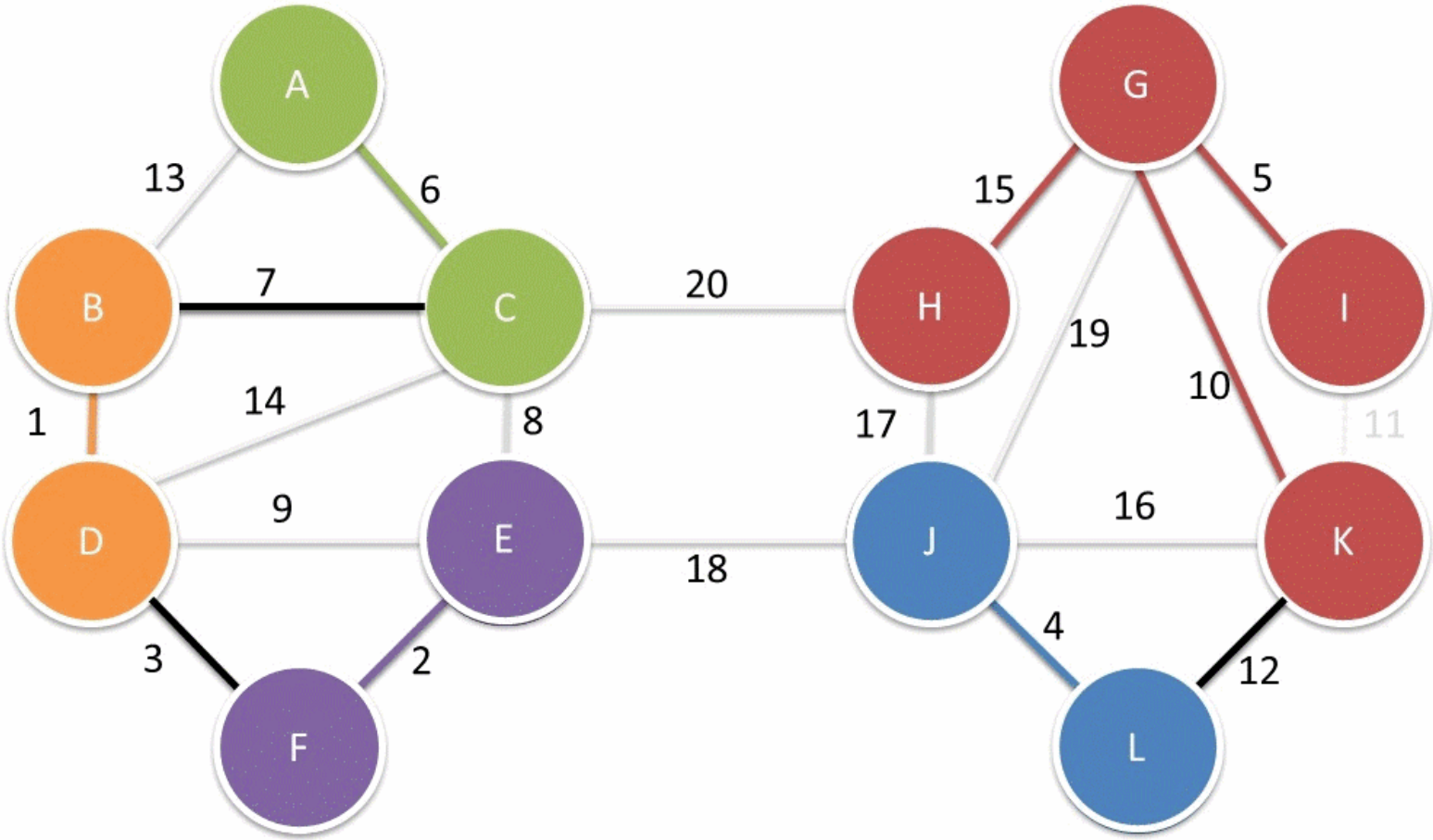

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
1
8
17
10
11
9
16
18
4
3
2
12

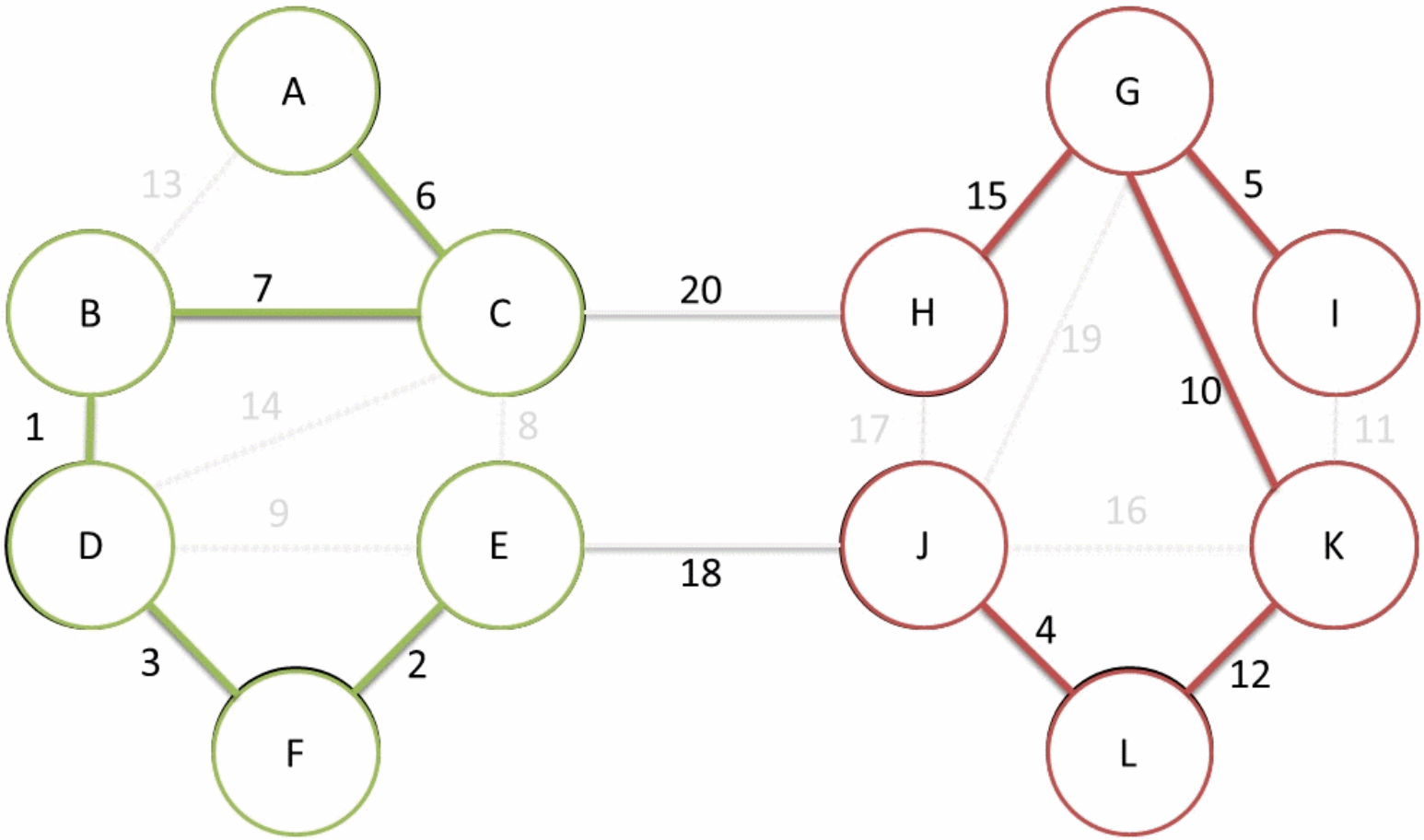

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
1
8
17
10
11
9
16
18
4
3
2
12

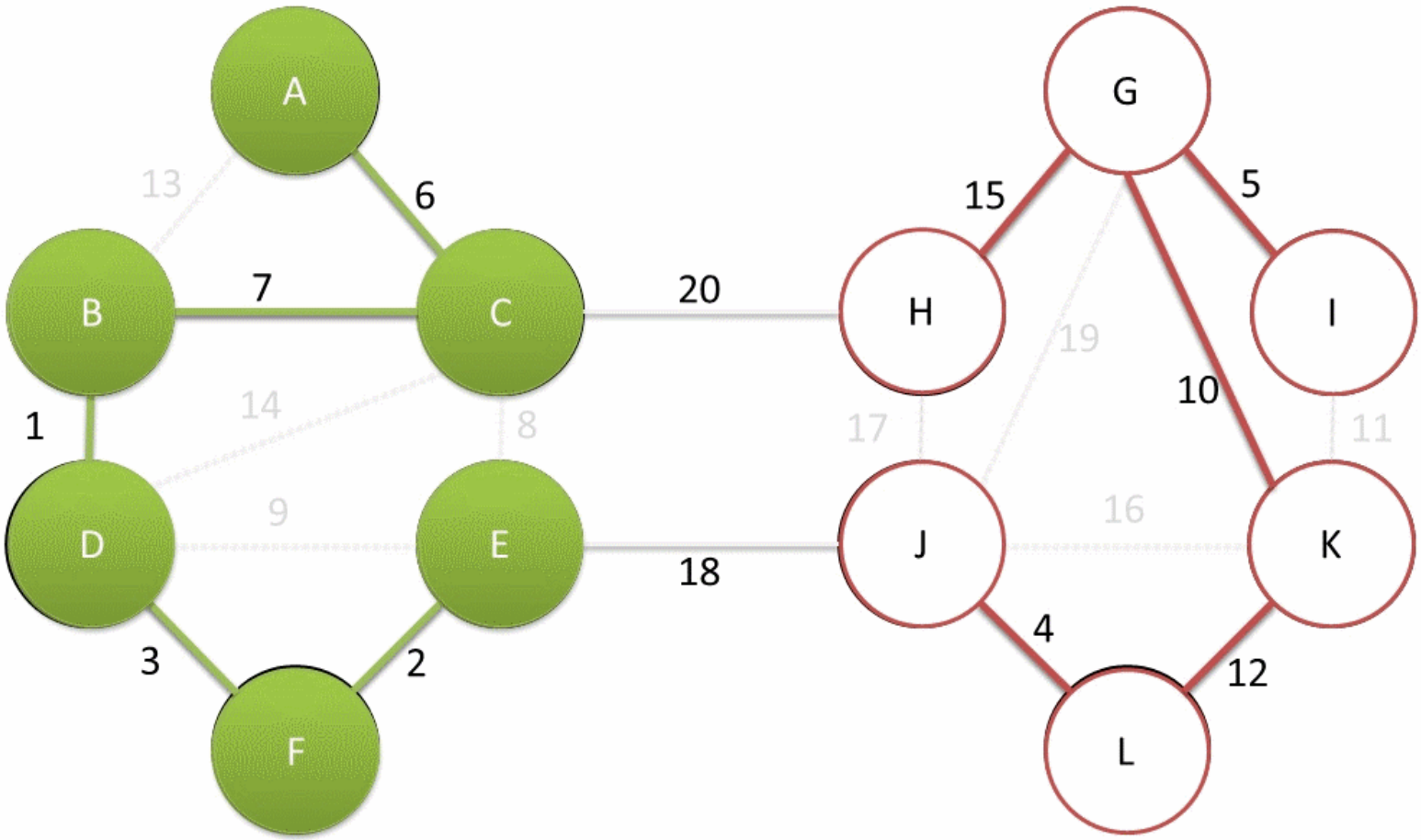

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
1
8
17
10
11
9
16
18
4
3
2
12

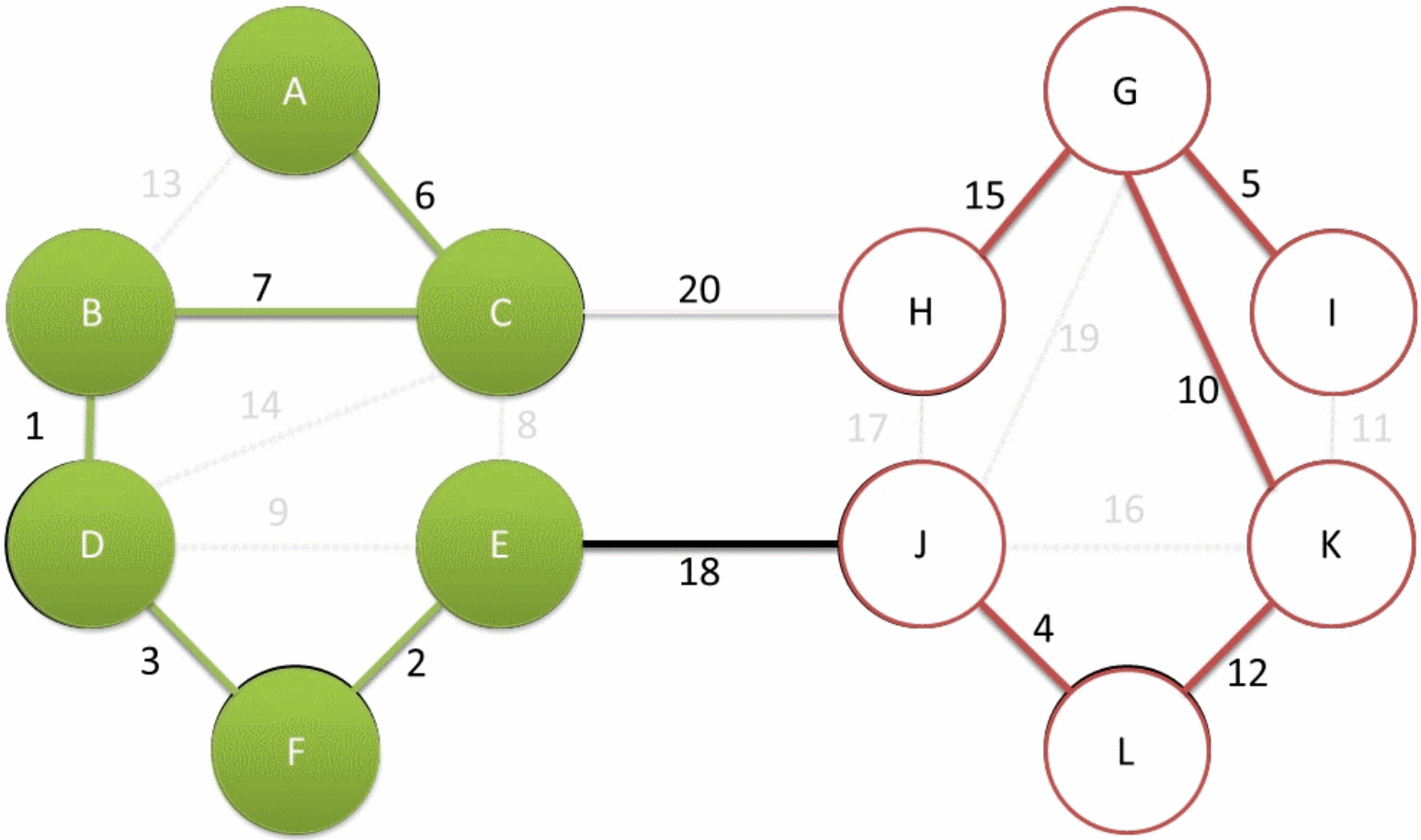

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
1
8
17
10
11
9
16
18
4
3
2
12

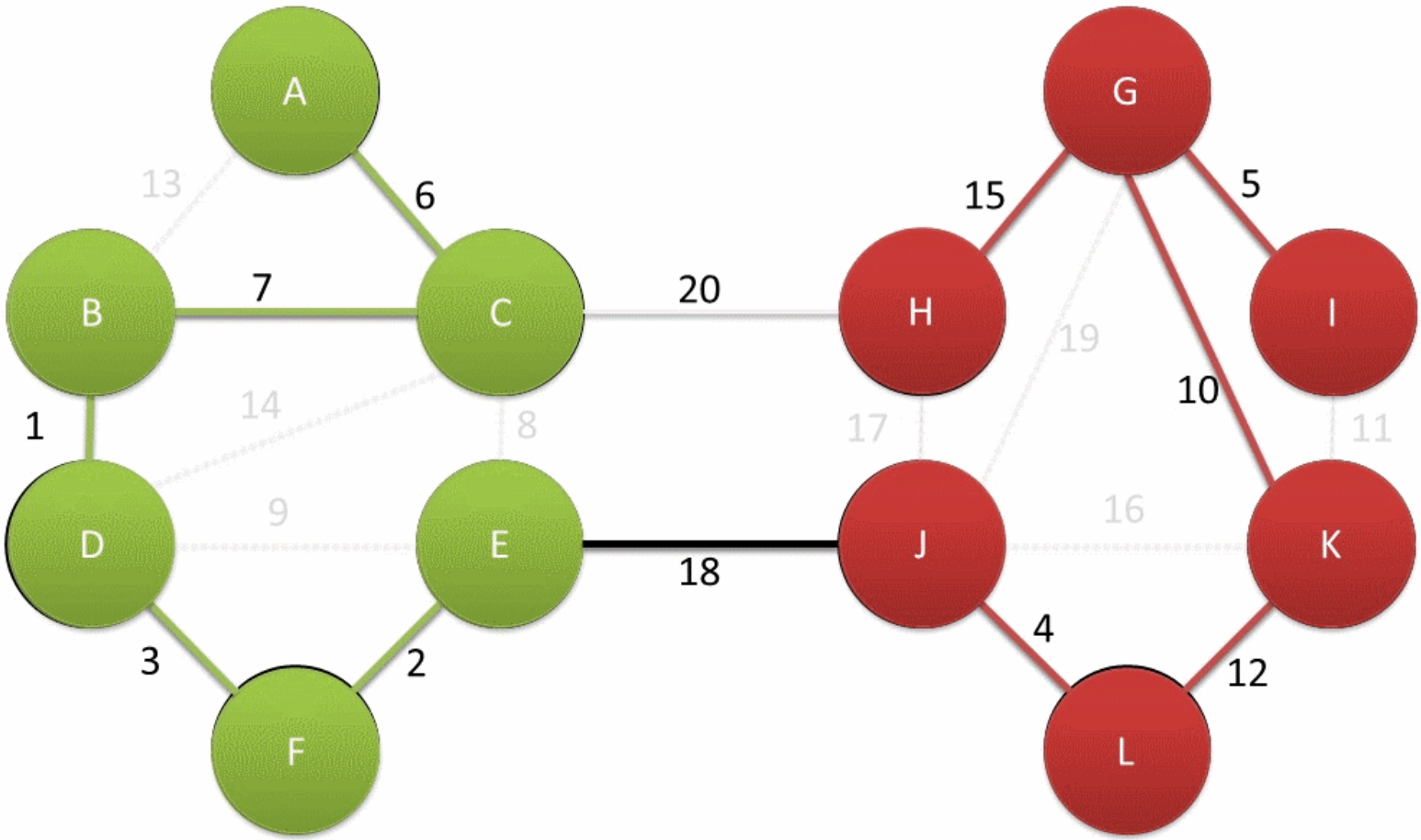

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
14
1
8
17
10
11
9
16
18
4
3
2
12


A
G
13
6
15
5
B
7
C
20
H
I
19
14
10
1
8
17
11
D
9
E
J
16
K
18
4
3
2
12
F
L

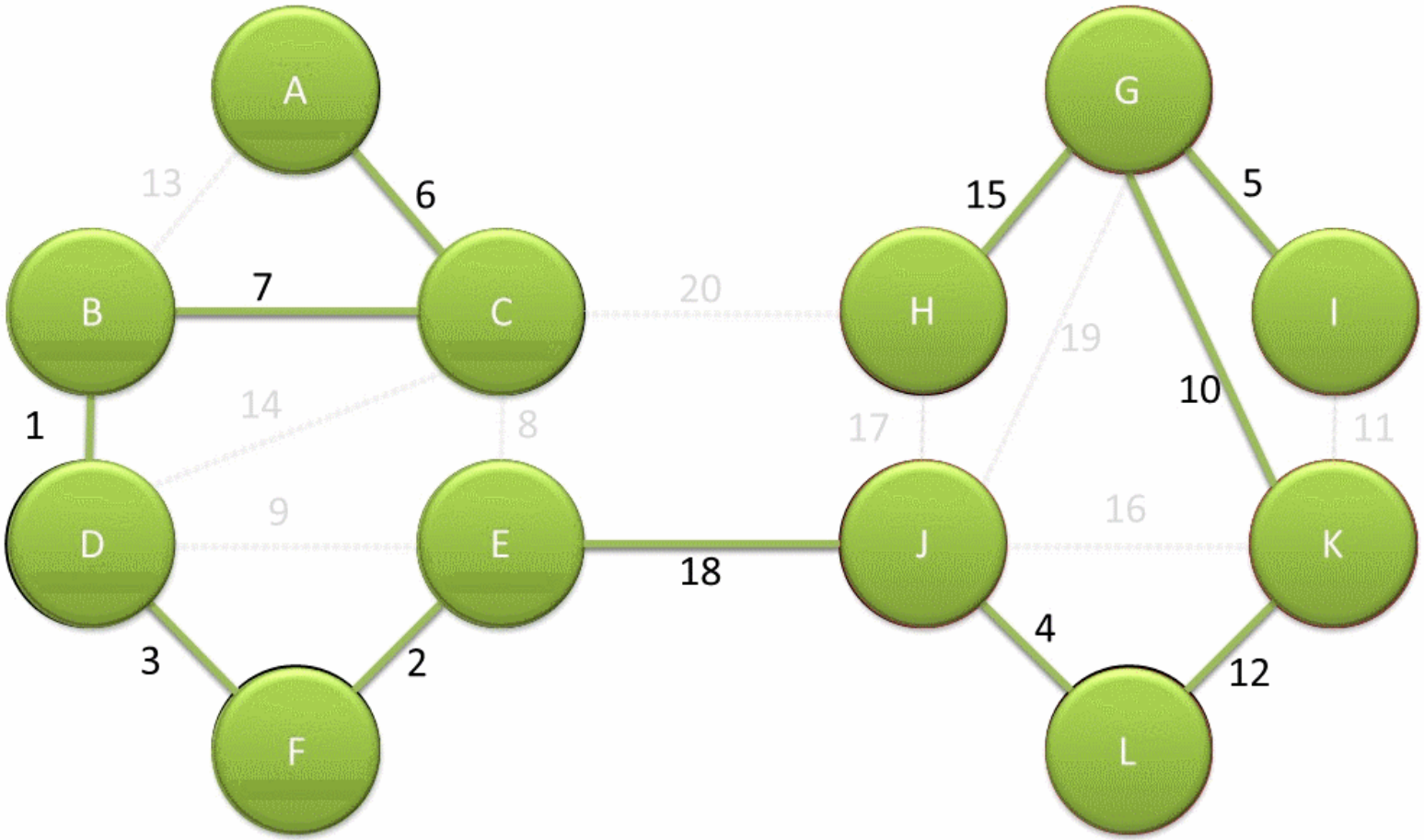

A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
15
5
19
10
1
14
8
17
11
9
16
18
4
3
2
12


A
B
C
D
E
F
G
H
I
J
K
L
13
6
7
20
1
14
8
9
3
2
18
15
5
19
10
17
11
16
4
12


A
G
13
6
15
5
B
7
C
20
H
I
19
14
10
1
8
17
11
D
9
E
18
J
16
K
4
3
2
12
F
L


A
G
13
6
15
5
B
7
C
20
H
I
19
10
1
14
8
17
11
D
9
E
18
J
16
K
4
3
2
12
F
L

And here is the final solution. The MST has a weight of 83 and the chosen edges are:



Borůvka's Algorithm

```
Input: A weighted undirected graph G = (V, E)
Output: F, a minimum spanning forest of G

Initialize a forest F to (V, E') where E' =

completed := false
while not completed do
    Find the connected components of F and assign to each vertex its component
    Initialize the cheapest edge for each component to "None"
    for each edge uv in E, where u and v are in different components of F:
        let wx be the cheapest edge for the component of u
        if is-preferred-over(uv, wx) then
            Set uv as the cheapest edge for the component of u
        let yz be the cheapest edge for the component of v
        if is-preferred-over(uv, yz) then
            Set uv as the cheapest edge for the component of v
    if all components have cheapest edge set to "None" then
        completed := true
    else
        completed := false
        for each component whose cheapest edge is not "None" do
            Add its cheapest edge to E'

function is-preferred-over(edge1, edge2) is
    return (edge2 is "None") or
            (weight(edge1) < weight(edge2)) or
            (weight(edge1) = weight(edge2) and tie-breaking-rule(edge1, edge2))

function tie-breaking-rule(edge1, edge2) is
    The tie-breaking rule; returns true if and only if edge1
    is preferred over edge2 in the case of a tie.
```

Lemma

There would be at most $O(log\ V)$ steps of merging trees.

Lemma

Each step of merging trees, takes $O(V + E)$.

Fact

The time complexity for Borůvka's algorithm is $O((V + E)log\ V)$.

Fact

The memory usage for Borůvka's algorithm is $O(V + E)$.

Lemma

The algorithm leads to a connected graph.

Lemma

The algorithm leads to an acyclic graph.

Fact

The algorithm leads to a tree with V vertices.

Lemma

The algorithm gives the MST (Prove by induction. There always exists some MST containing these edges.).

Other algorithms

Other algorithms for this problem include Prim's algorithm and Kruskal's algorithm.

Fact

Prim algorithm can be done in $O(E + V log\ V)$ using complicated data structures like fibonacci heap.

Fact

Kruskal algorithm can be done in $O((V + E)log\ E)$ since it needs the sorted list of edges.

Application 1

Fast parallel algorithms can be obtained by combining Prim's algorithm with Borůvka's.

Application 2

A faster randomized minimum spanning tree algorithm based in part on Borůvka's algorithm due to Karger, Klein, and Tarjan runs in expected $O(E)$ time.

Application 3

The best known (deterministic) minimum spanning tree algorithm by Bernard Chazelle is also based in part on Borůvka's and runs in $O(E\alpha(E, V))$ time, where α is the inverse Ackermann function.

Fact

These randomized and deterministic algorithms combine steps of Borůvka's algorithm, reducing the number of components that remain to be connected, with steps of a different type that reduce the number of edges between pairs of components.

Figure: Otakar Borůvka

References

- en.wikipedia.org/wiki/Bor%C5%AFvka%27s_algorithm
- en.wikipedia.org/wiki/Otakar_Bor%C5%AFvka
- cs.brown.edu/research/pubs/pdfs/1995/Karger-1995-RLT.pdf
- https://en.wikipedia.org/wiki/Prim%27s_algorithm
- https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

The End

Thank You For Your Attention!