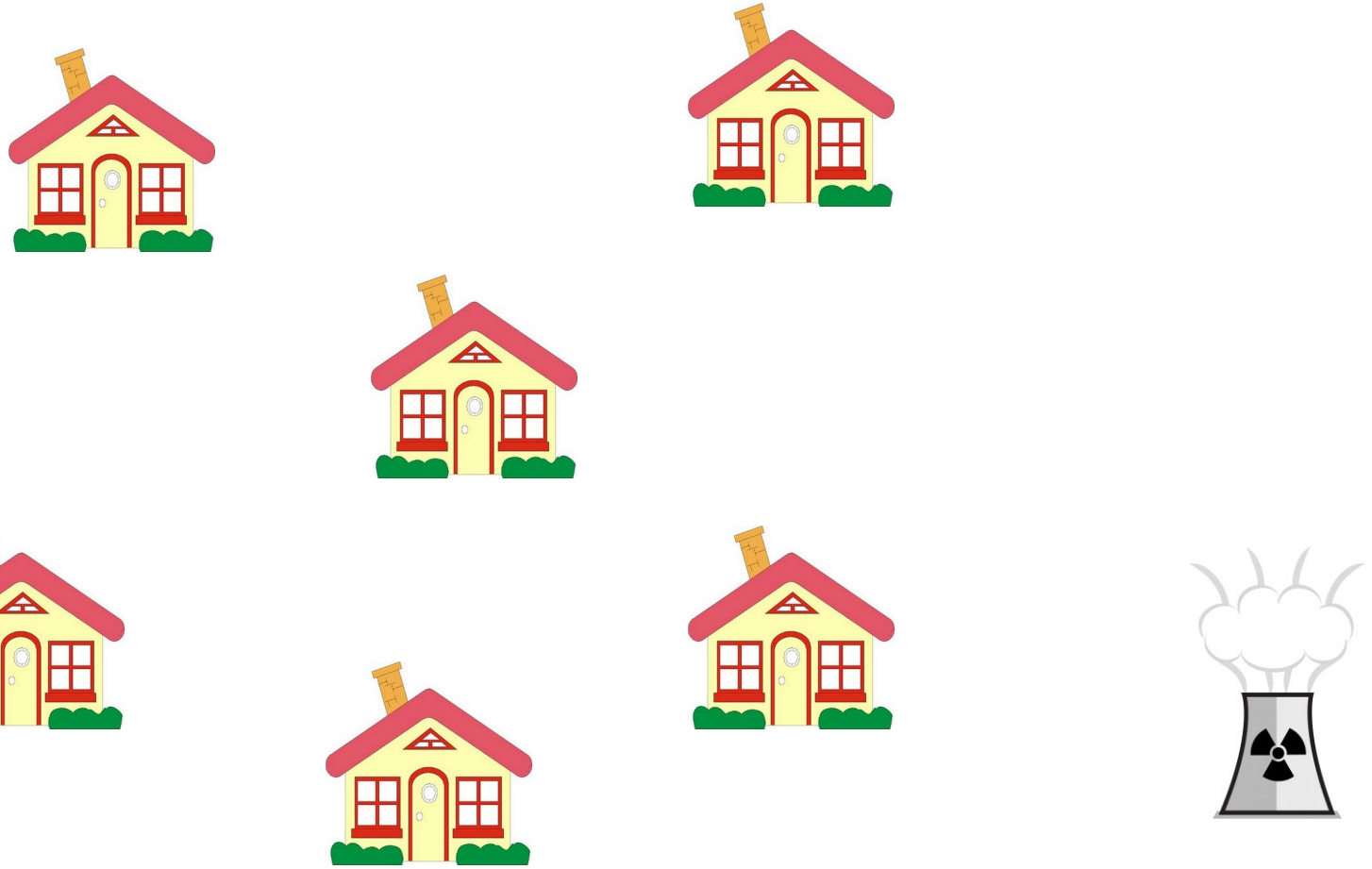
Minimum Spanning Trees

We now know how to compute the minimum cost path from a start node to every other node in a graph.

What other interesting problems can we solve using graphs?

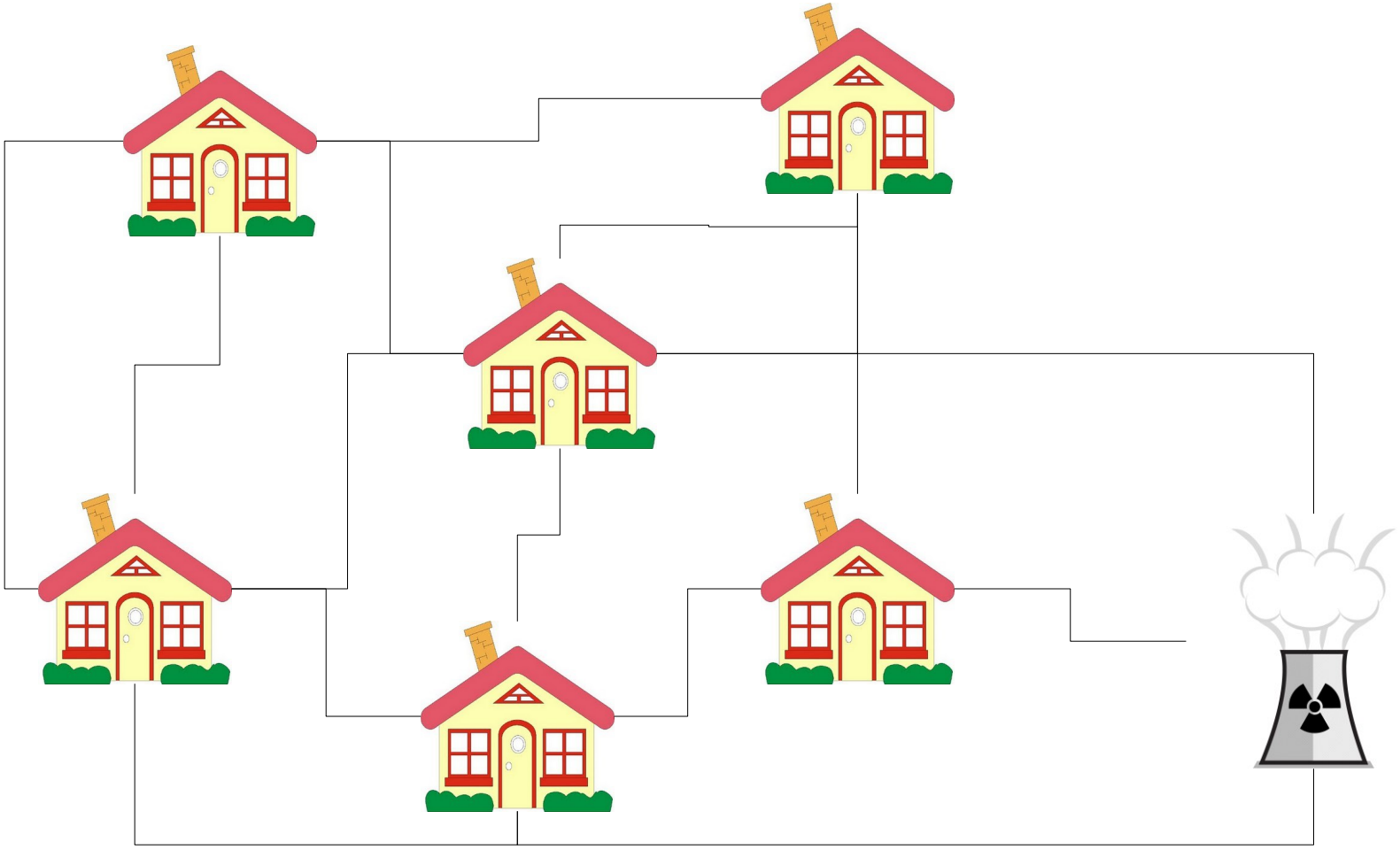
Constructing a Network



http://www.graphicsfactory.com/Clip_Art/Science/Nuclear-Power-Plants-7_381920.html

http://www.habitatstratfordperth.ca/index.php?page=Family&&page_id=3&&id=3

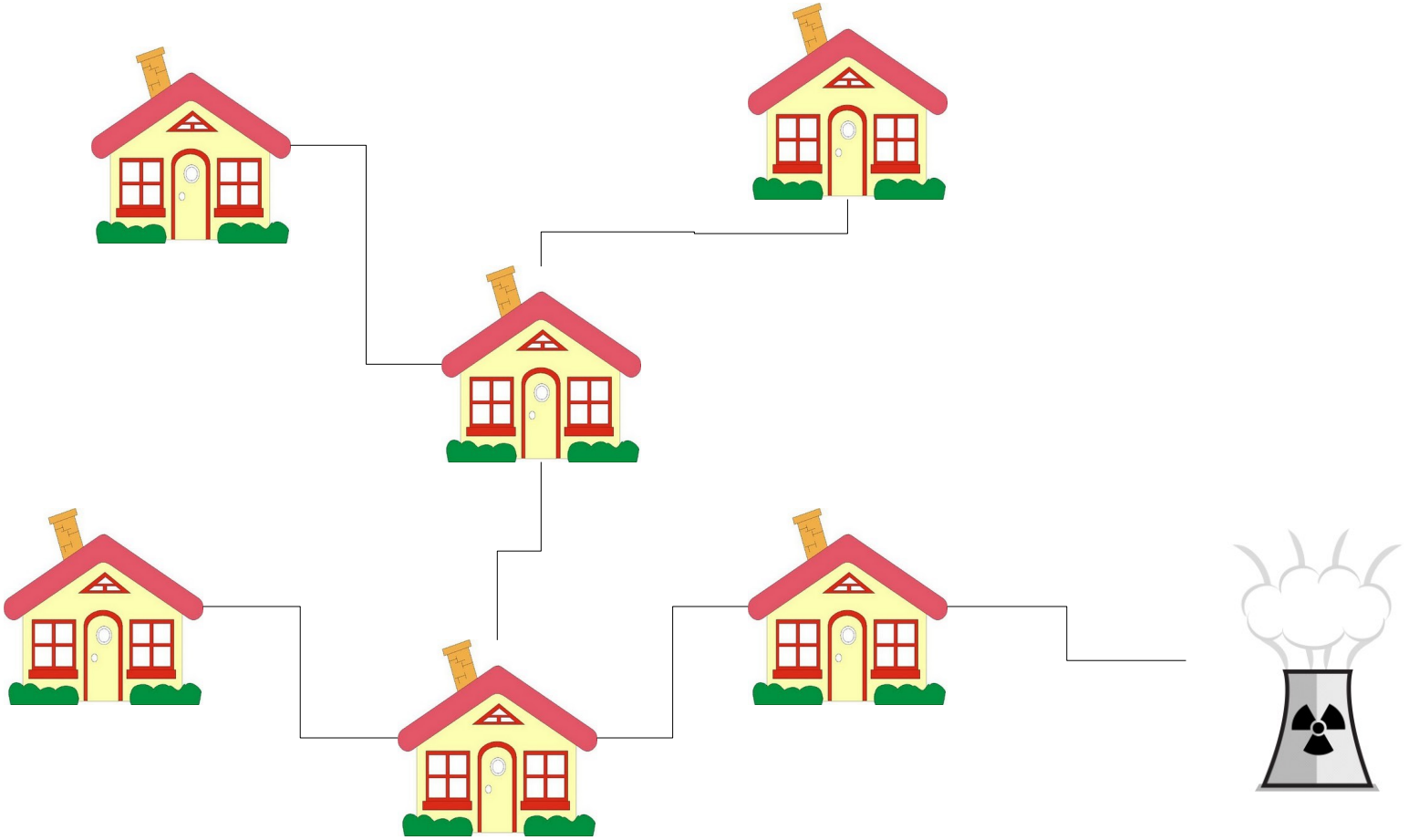
Constructing a Network



http://www.graphicsfactory.com/Clip_Art/Science/Nuclear-Power-Plants-7_381920.html

http://www.habitatstratfordperth.ca/index.php?page=Family&&page_id=3&&id=3

Constructing a Network

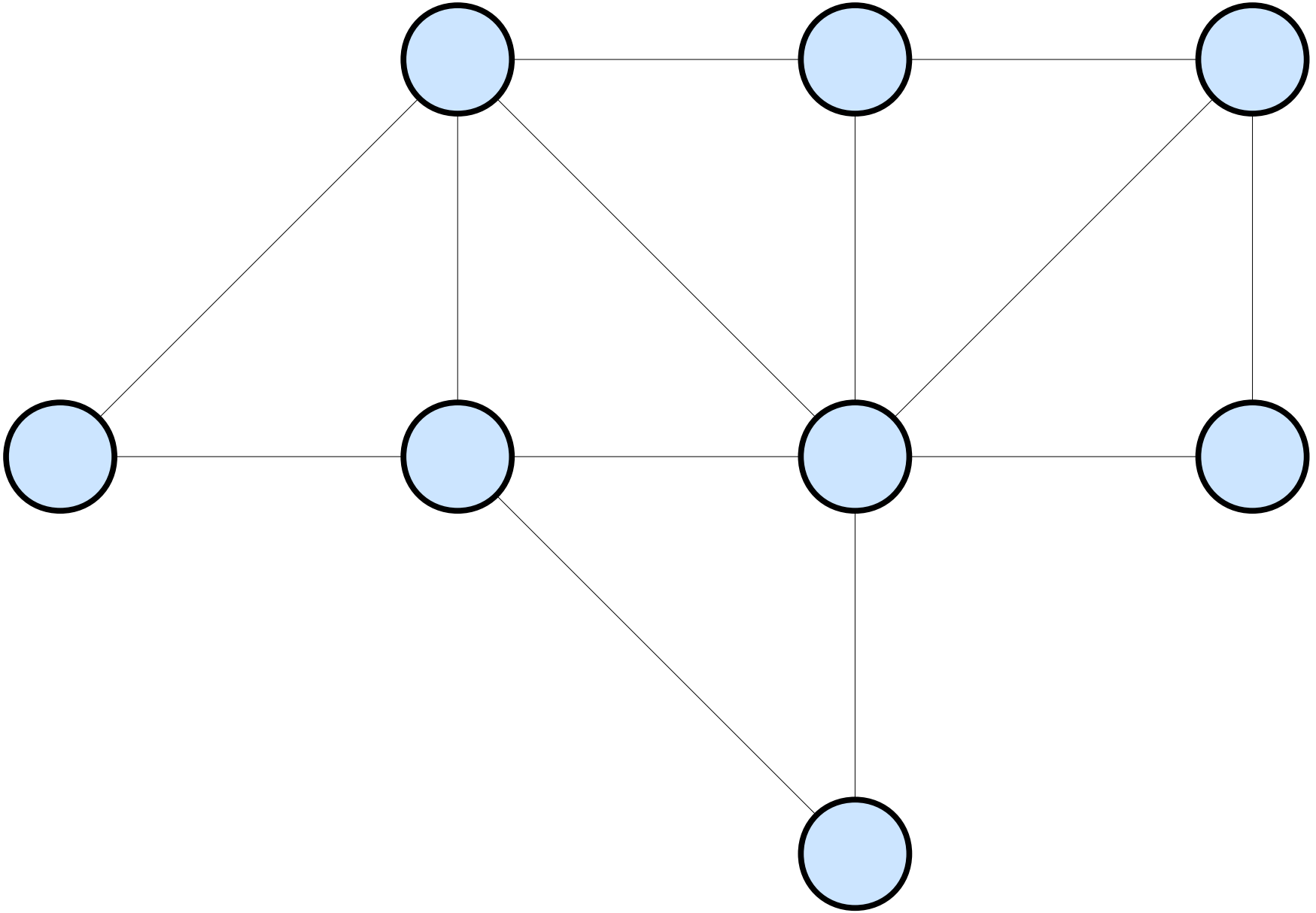


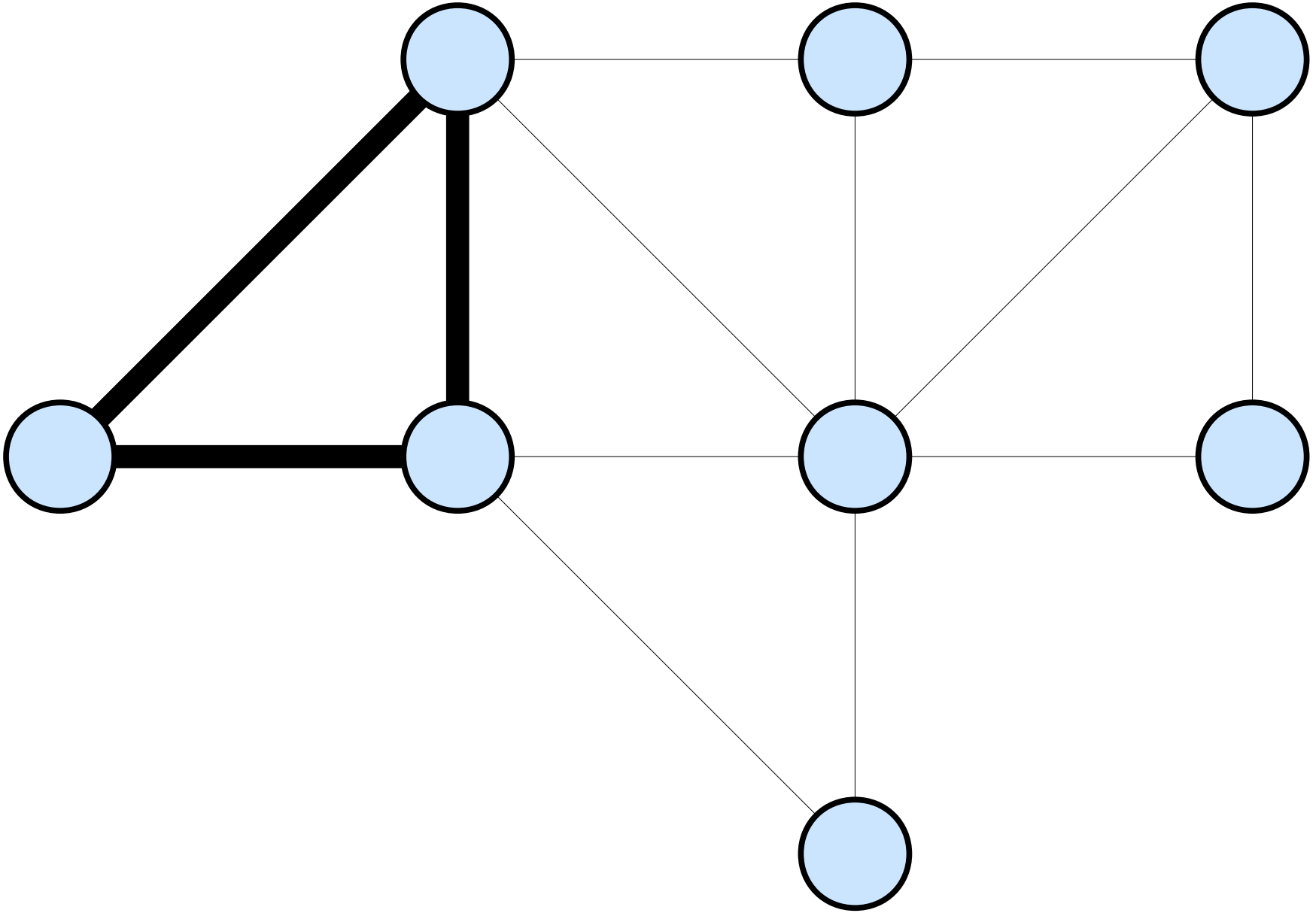
http://www.graphicsfactory.com/Clip_Art/Science/Nuclear-Power-Plants-7_381920.html

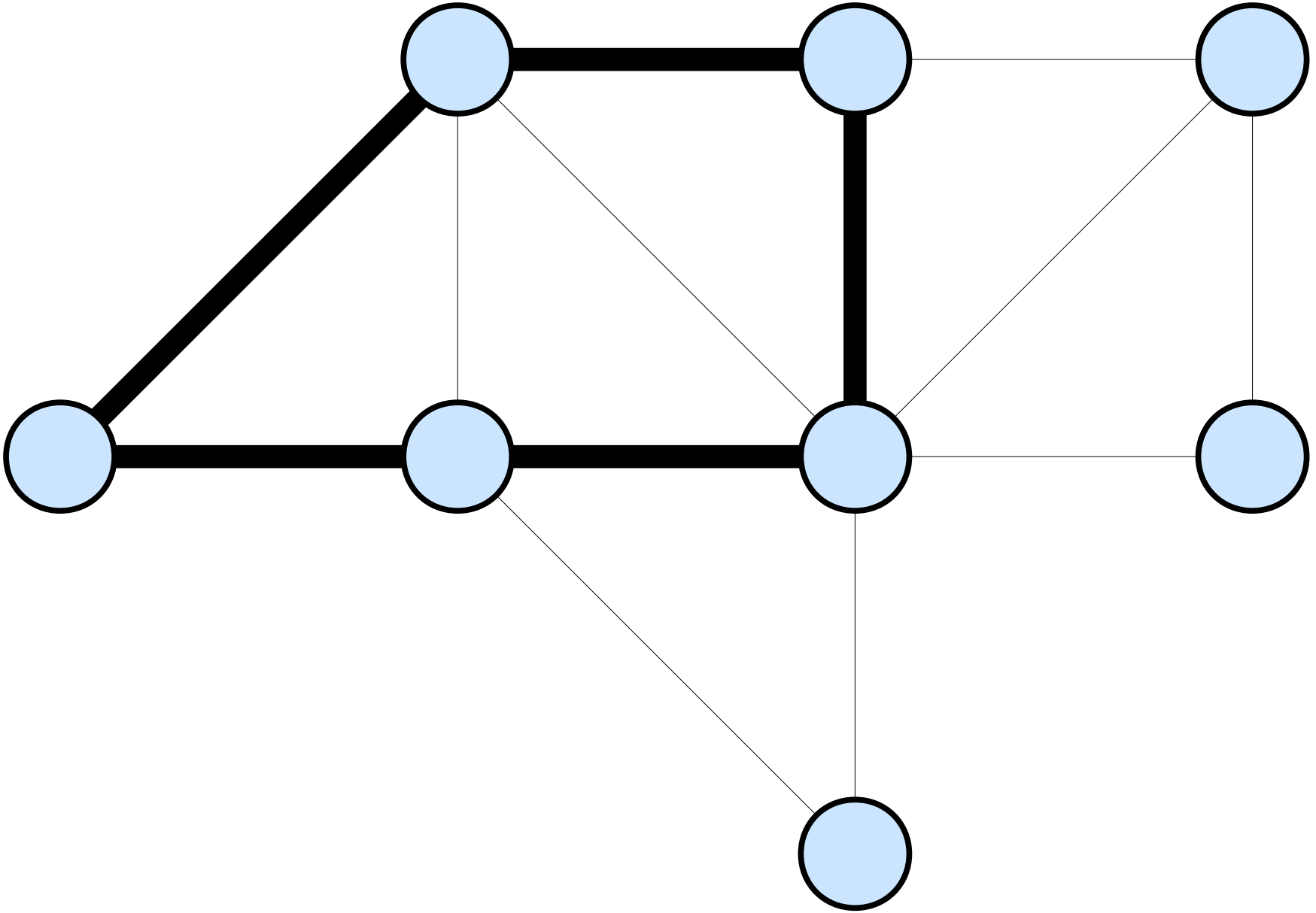
http://www.habitatstratfordperth.ca/index.php?page=Family&&page_id=3&&id=3

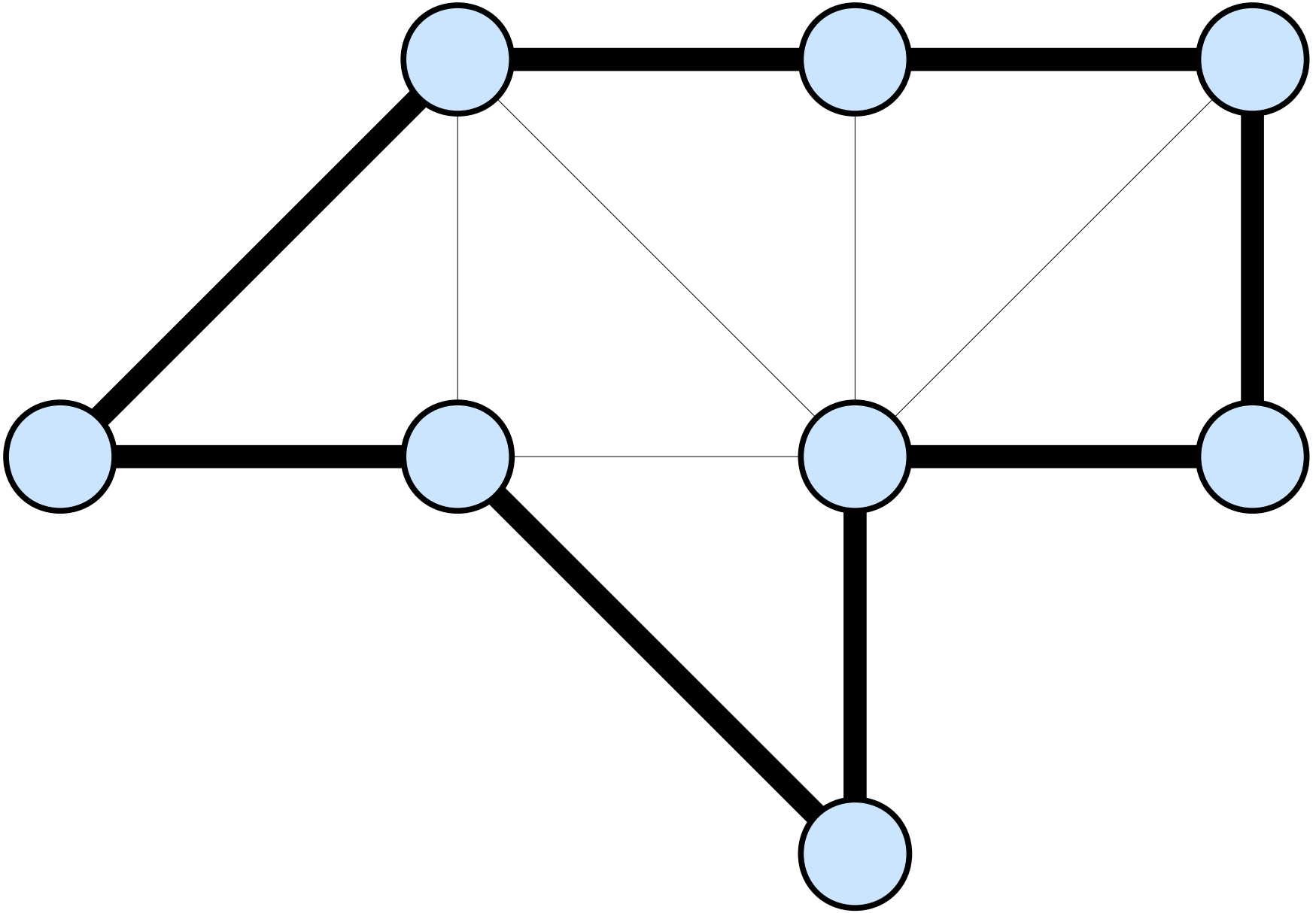
Cycle

A **cycle** in an undirected graph is a path that starts and ends at the same node.



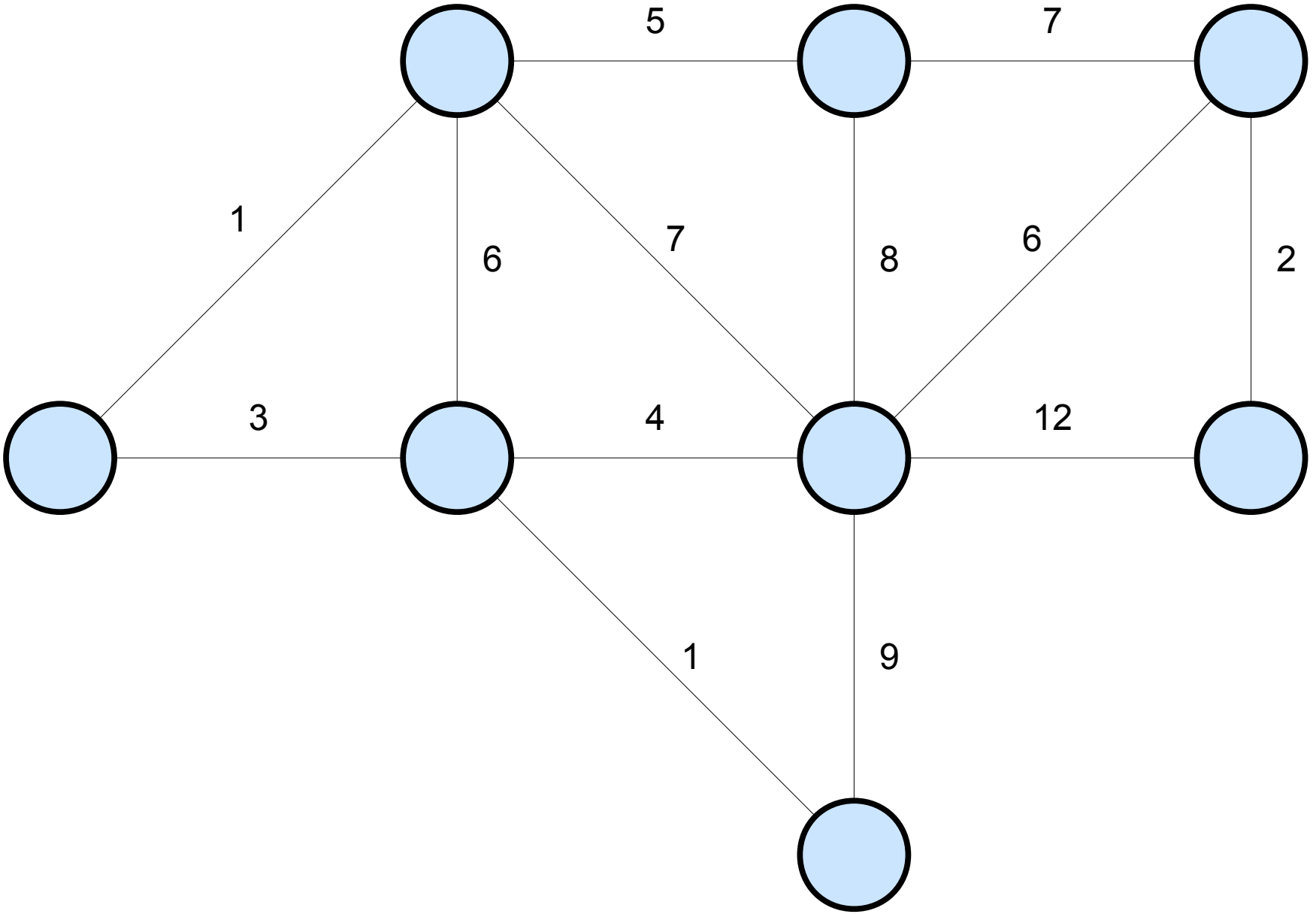


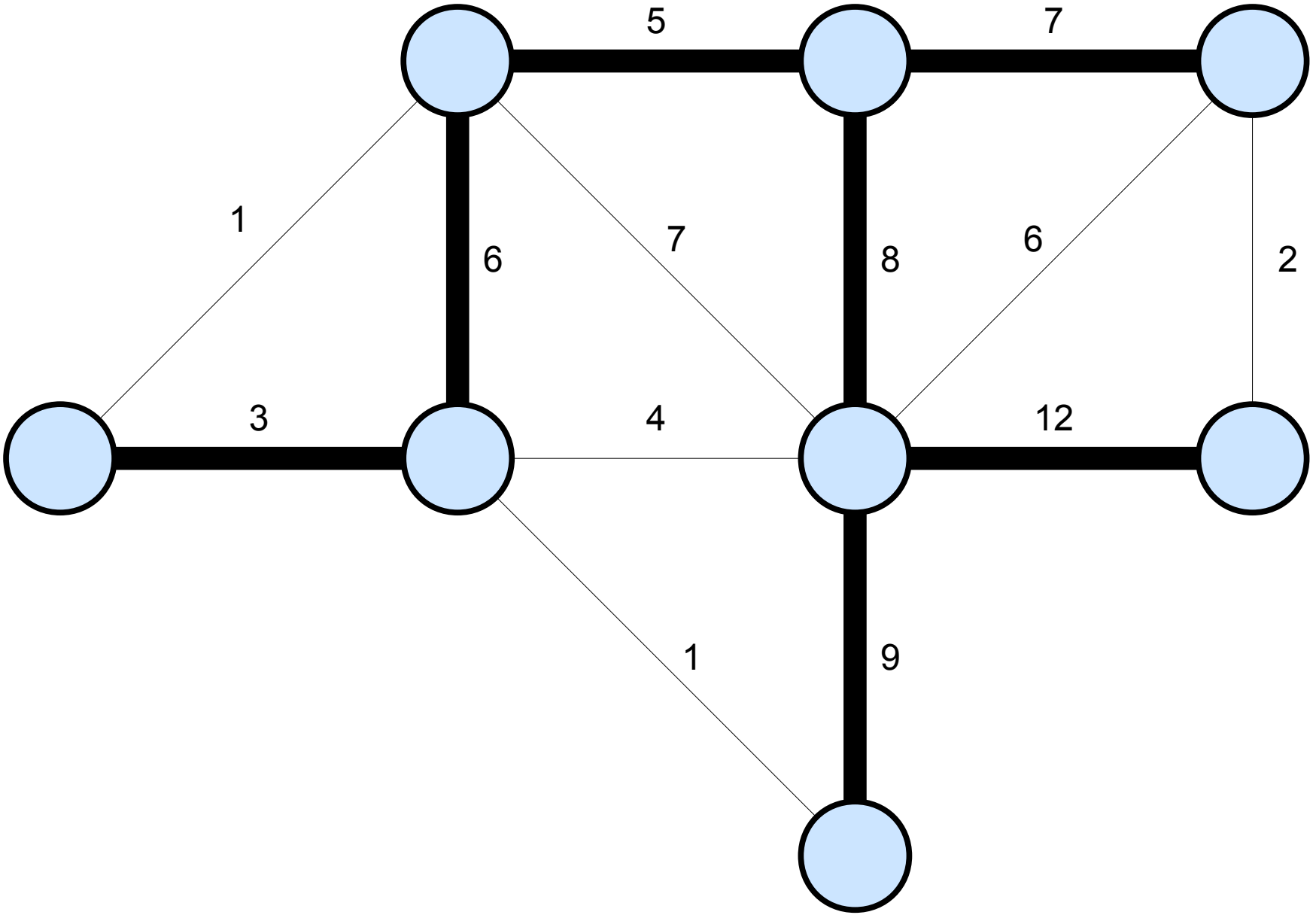


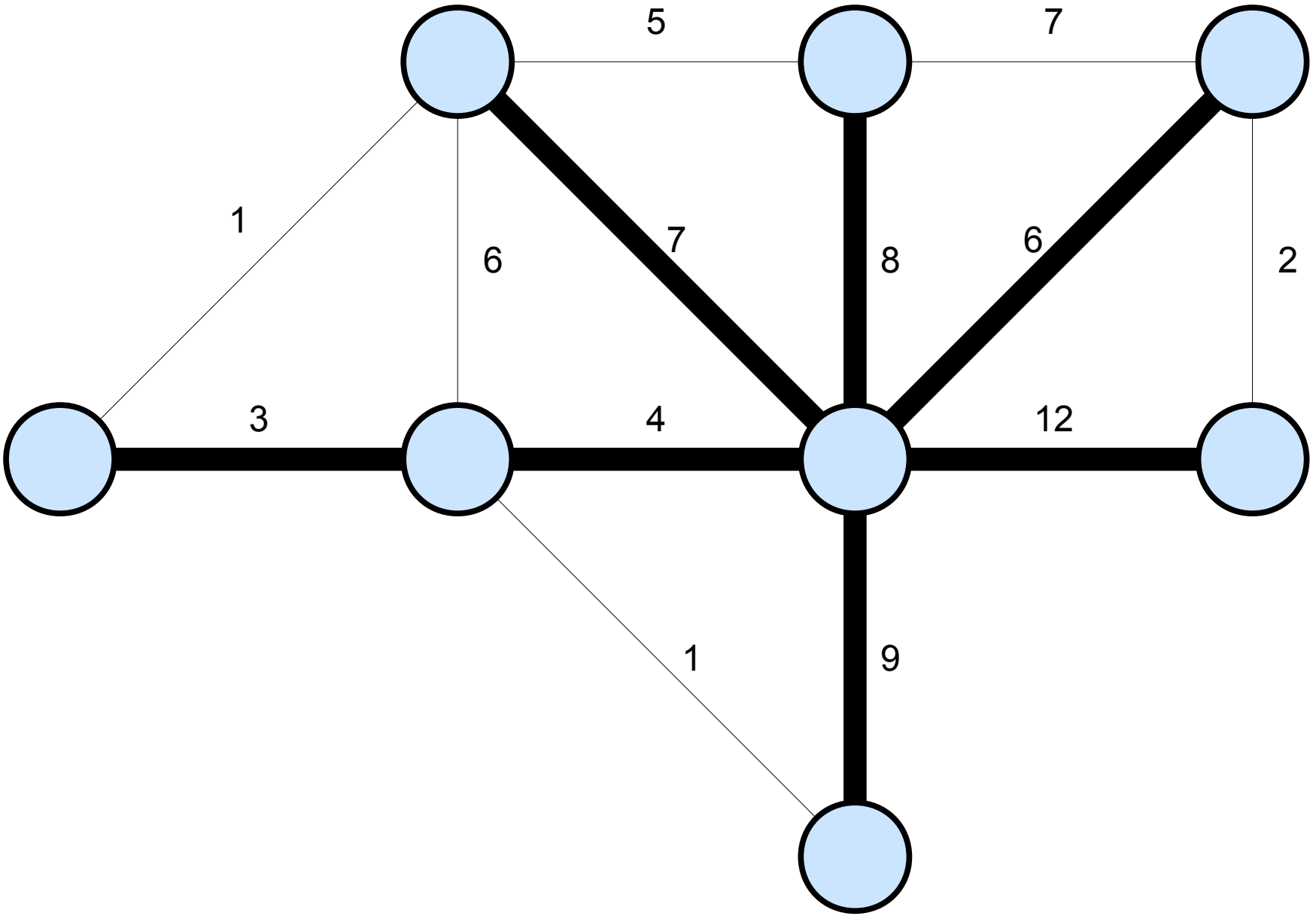


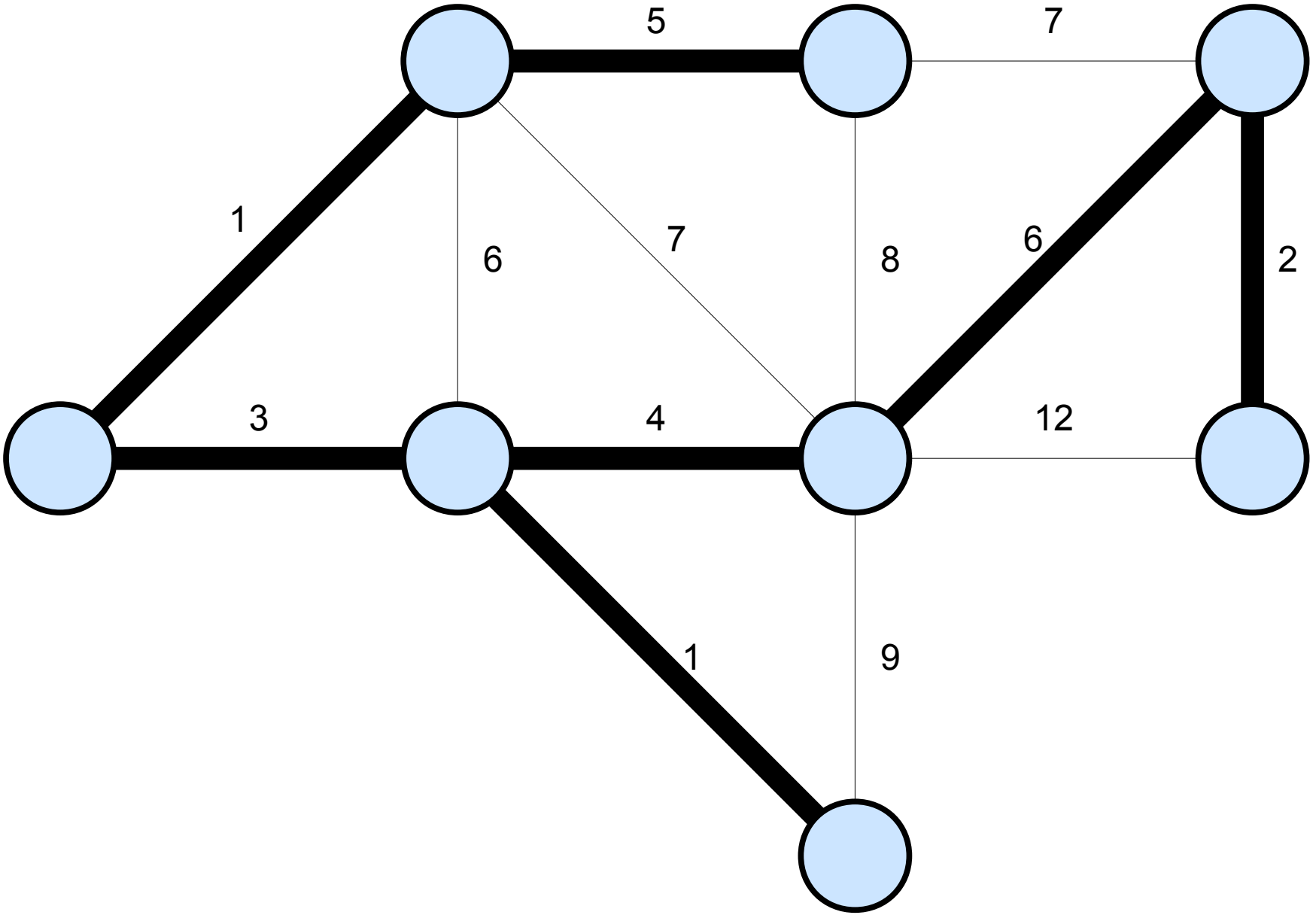
Spanning Trees

A **spanning tree** in an undirected graph is a set of edges, with no cycles, that connects all nodes.


5
7
1
6
7
8
6
2
3
4
12
1
9


5
7
1
6
7
8
6
2
3
4
12
1
9


5
7
1
6
7
8
6
2
3
4
12
1
9


5
7
1
6
7
8
6
2
3
4
12
1
9

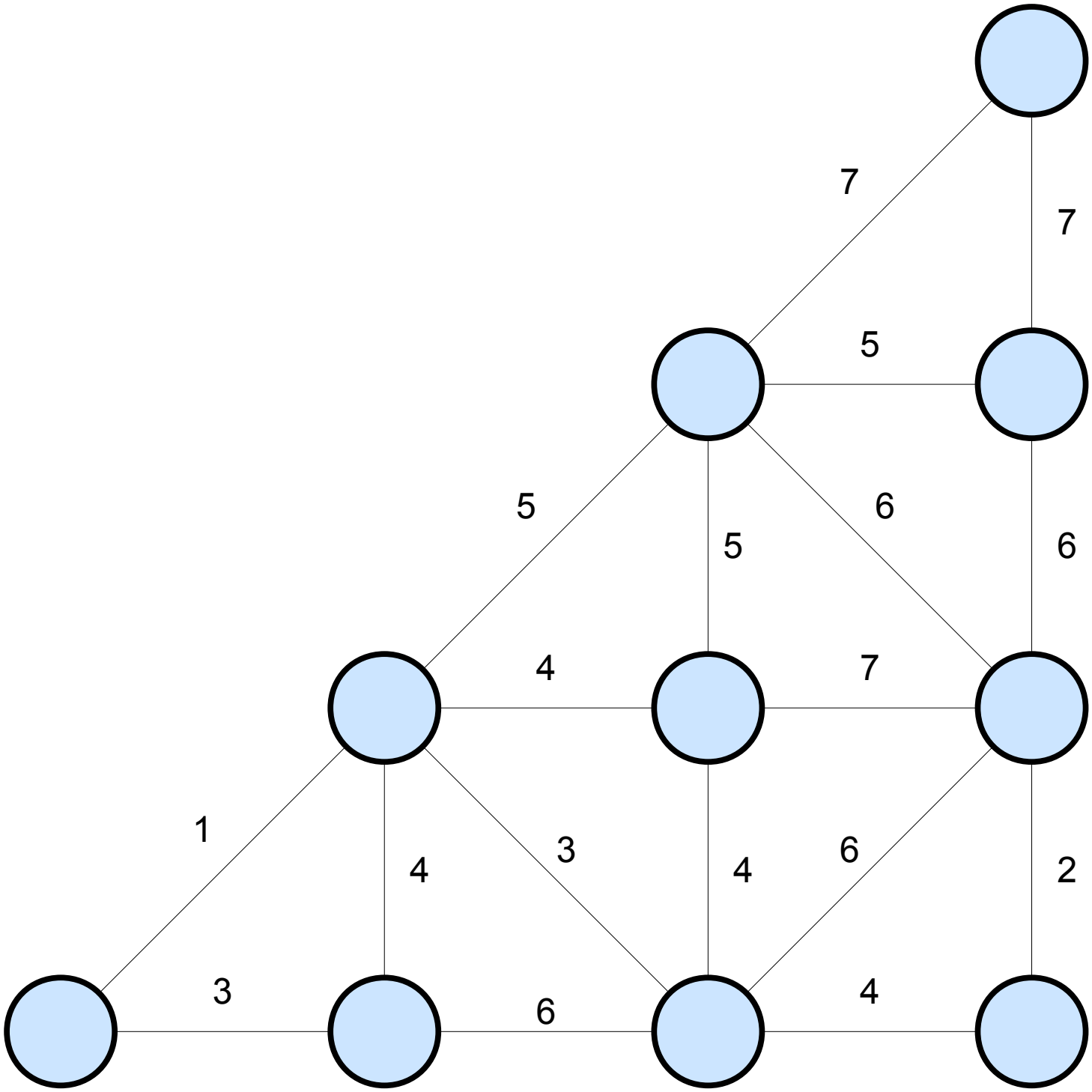
A **minimum spanning tree** (or **MST**) is a spanning tree with the least total cost.

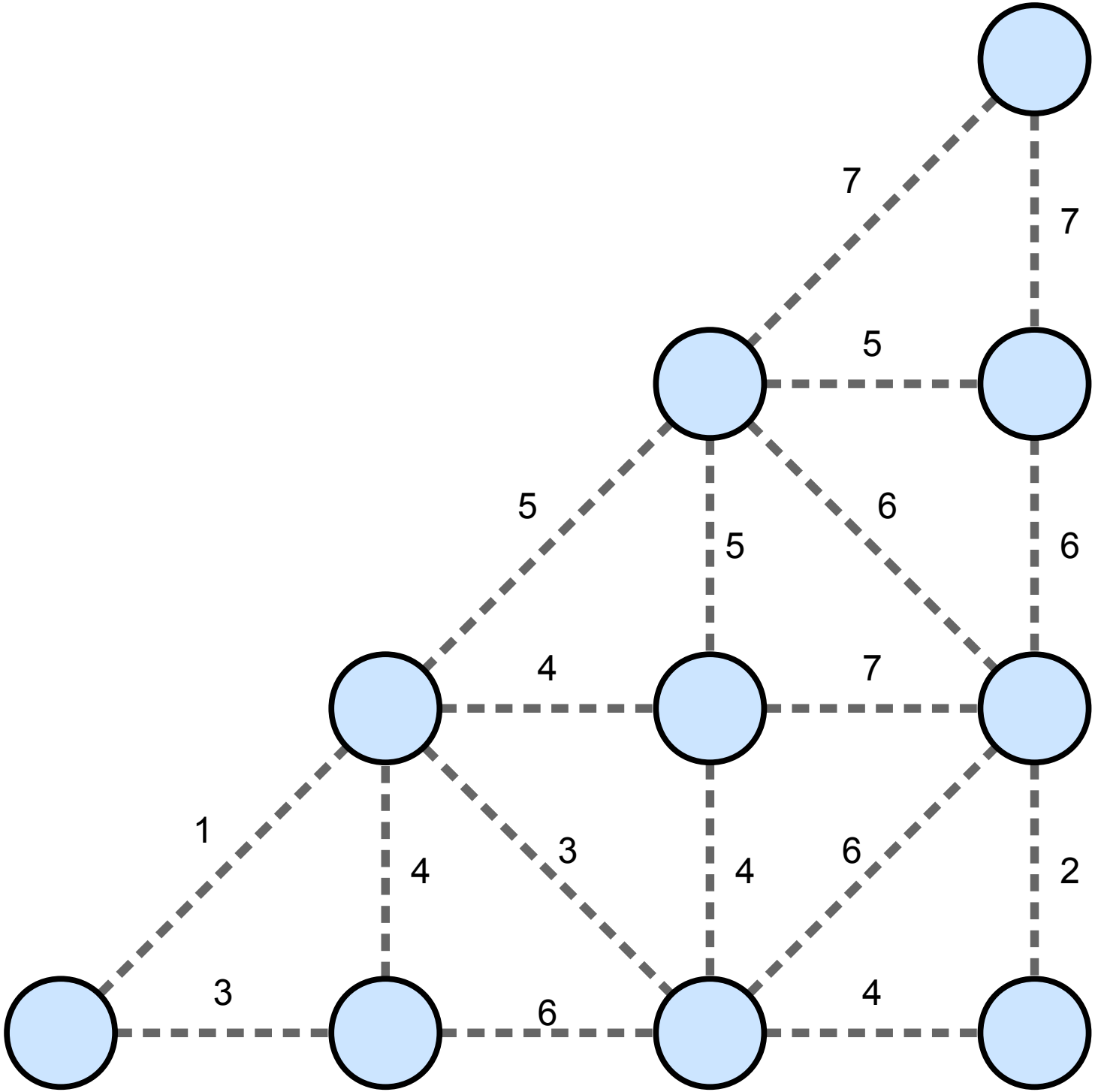
Applications

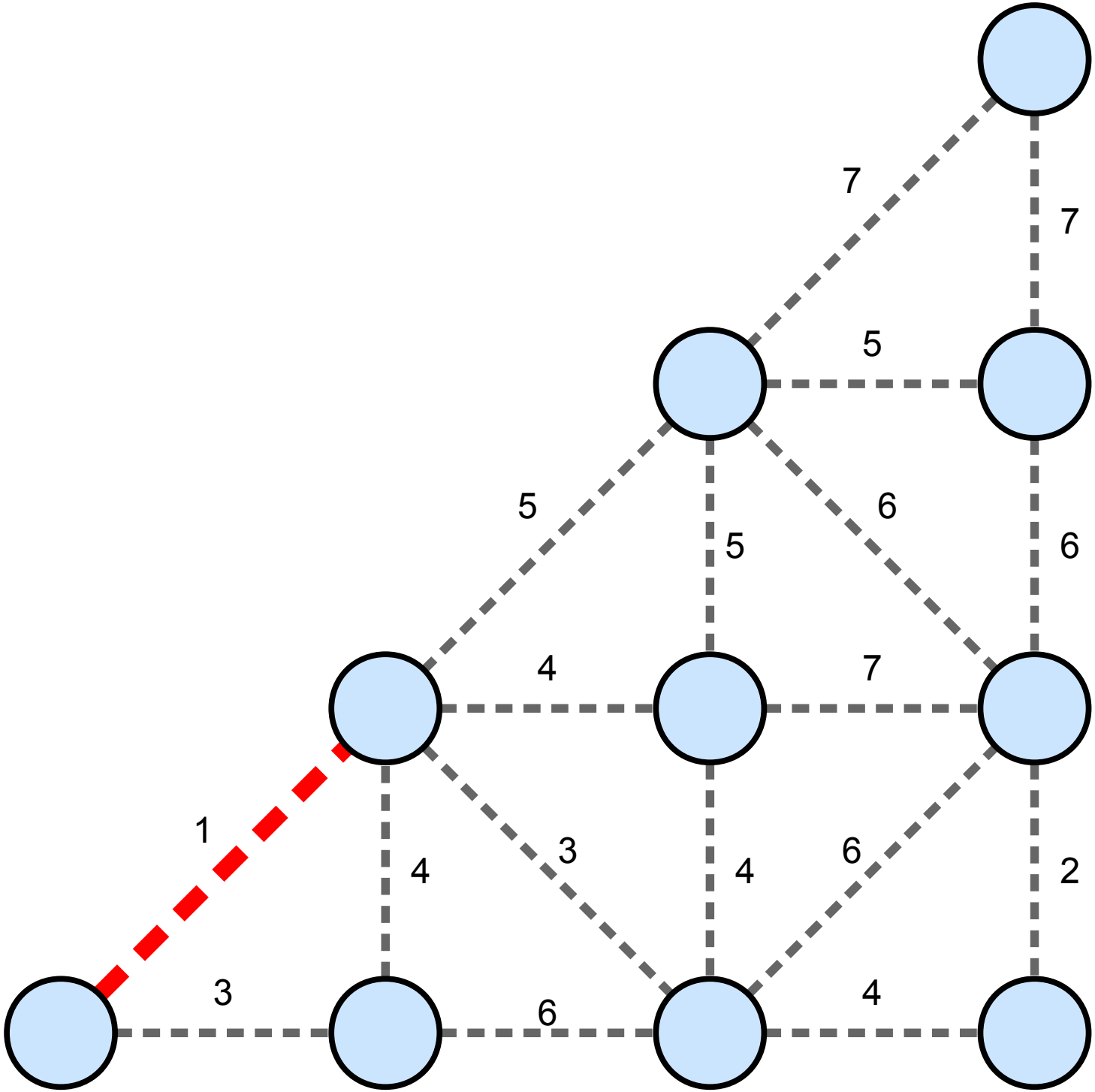
- **Electric Grids**
  - Given a collection of houses, where do you lay wires to connect all houses with the least total cost?
  - This was the initial motivation for studying minimum spanning trees in the early 1920's. (work done by Czech mathematician **Otakar Borůvka**)
- **Data Clustering**
  - More on that later...
- **Maze Generation**
  - More on that later...

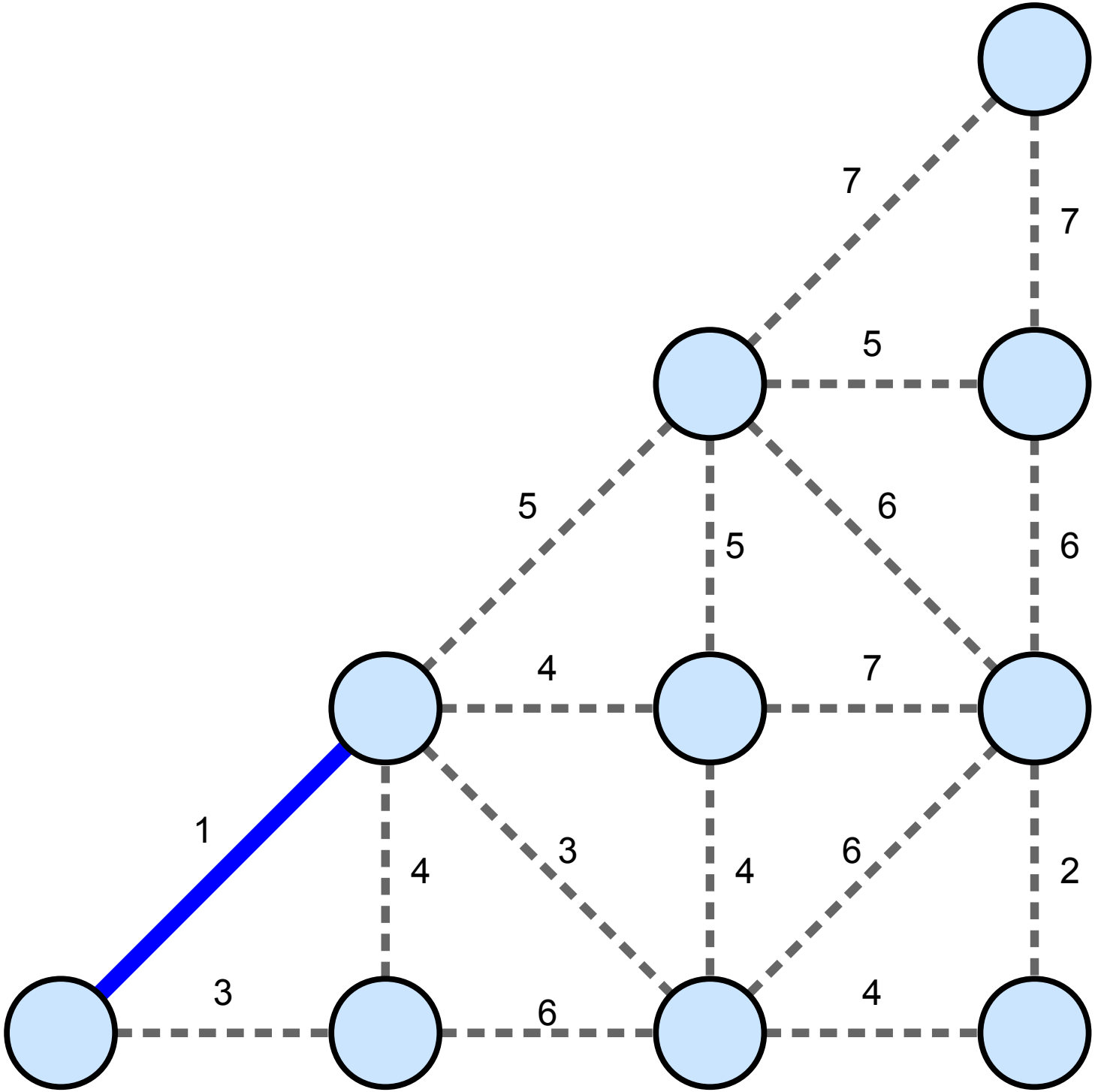
Kruskal's Algorithm

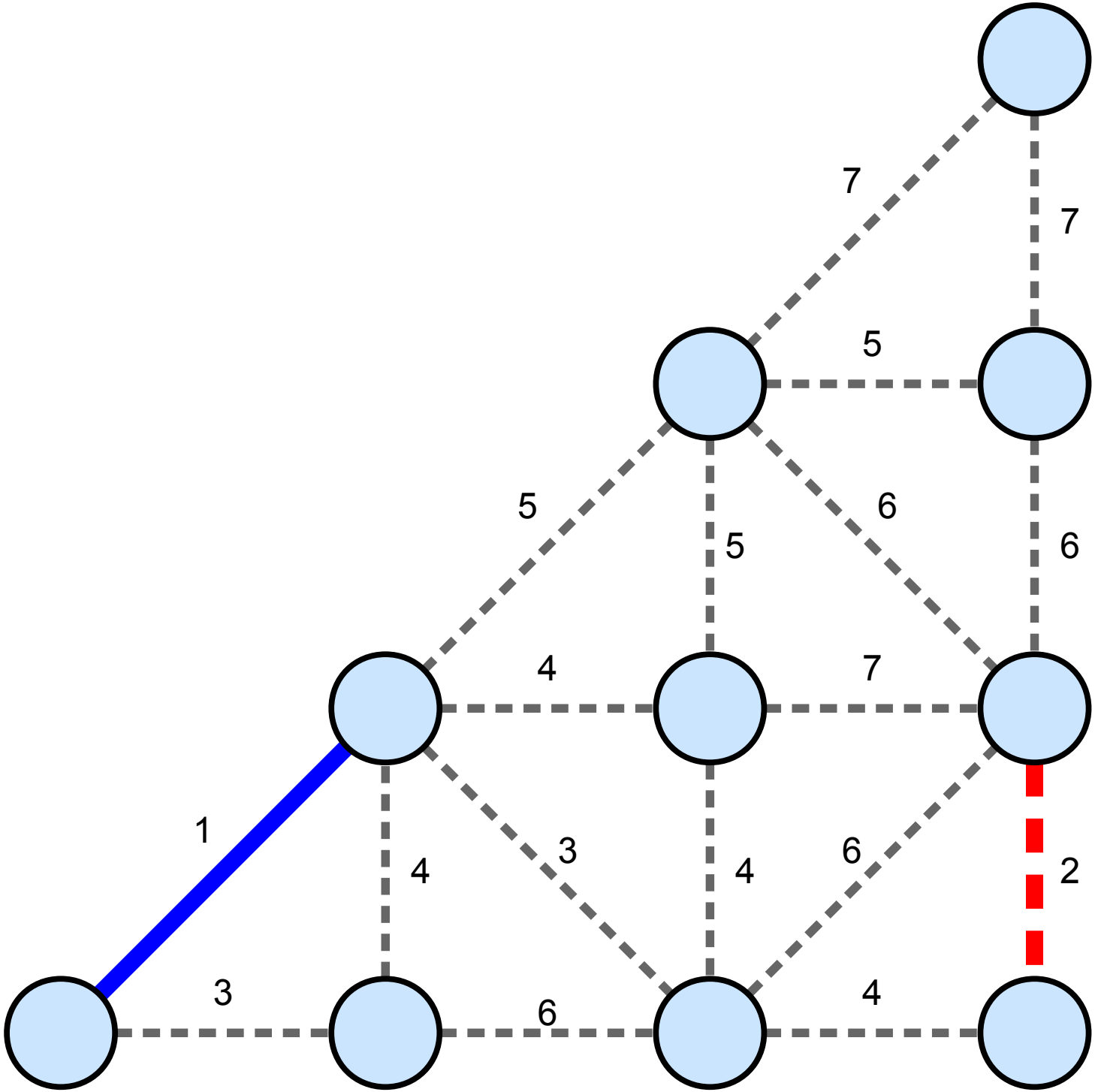
- **Kruskal's algorithm** is an efficient algorithm for finding minimum spanning trees.
- Idea is as follows:
  - Remove all edges from the graph.
  - Place all edges into a priority queue based on their length.
  - While the priority queue is not empty:
    - Dequeue an edge from the priority queue.
    - If the endpoints of the edge aren't already connected to one another, add in that edge.
    - Otherwise, skip the edge.

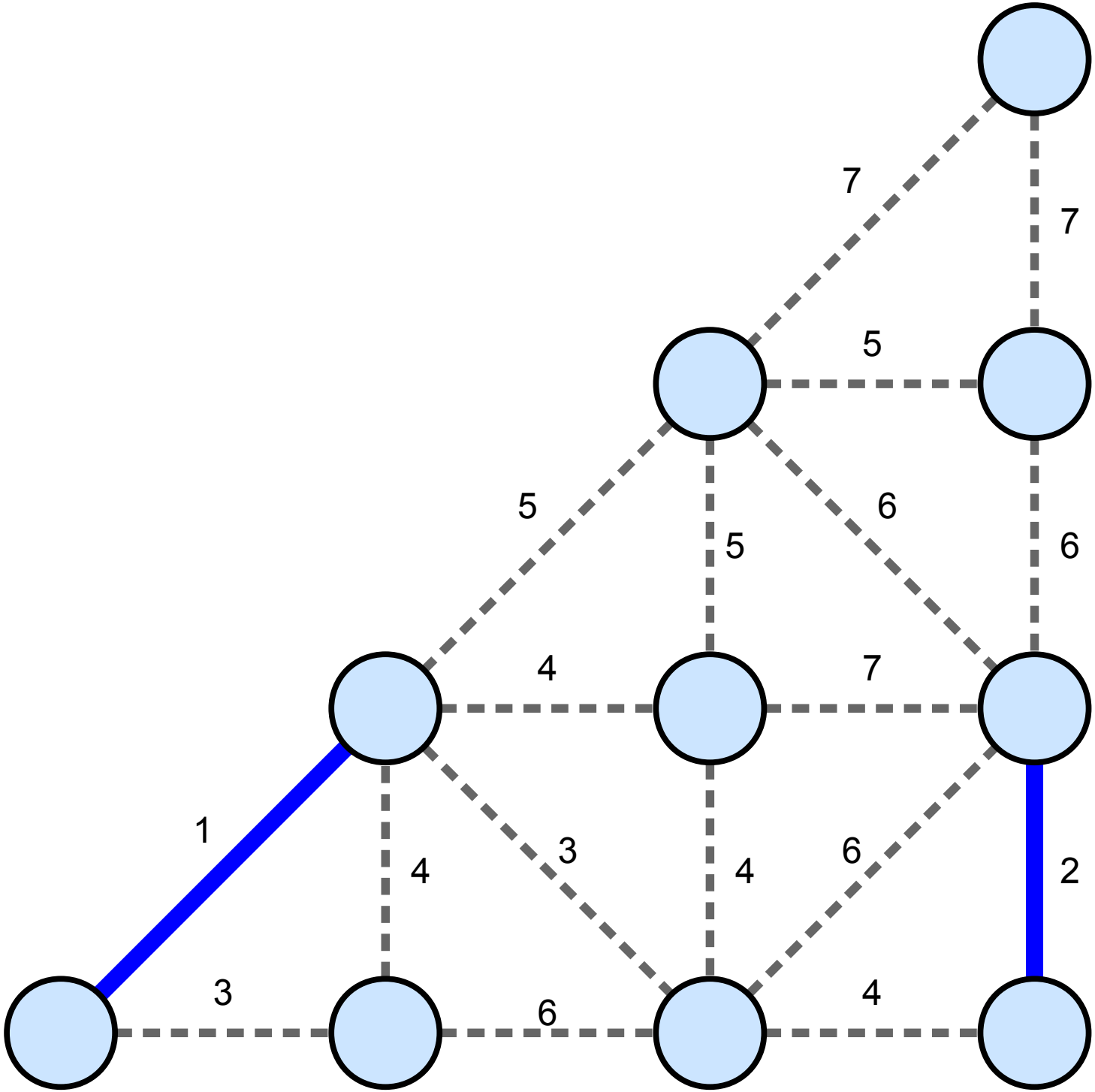

7
7
5
5
6
5
6
4
7
1
4
3
4
6
2
3
6
4

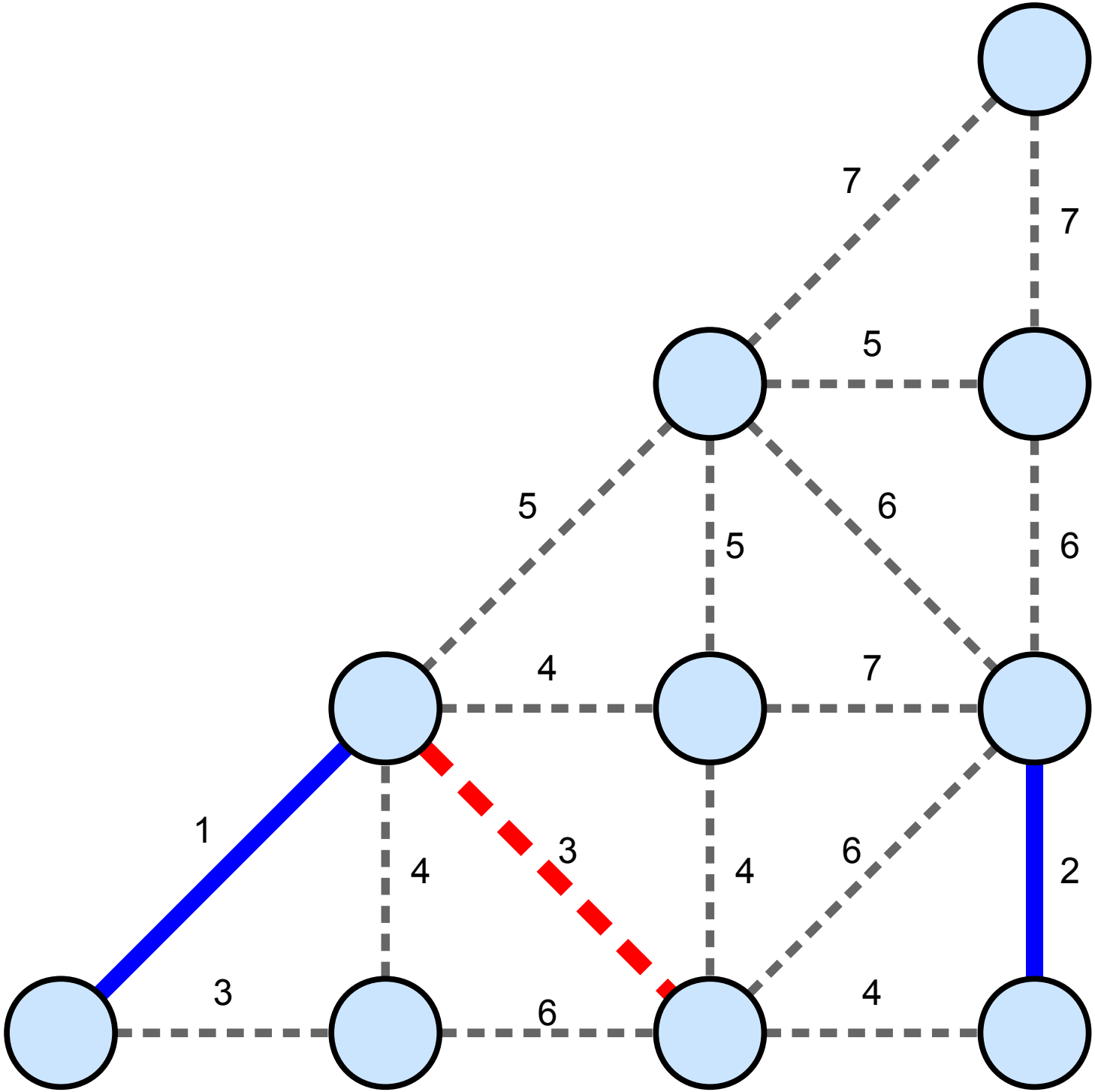

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

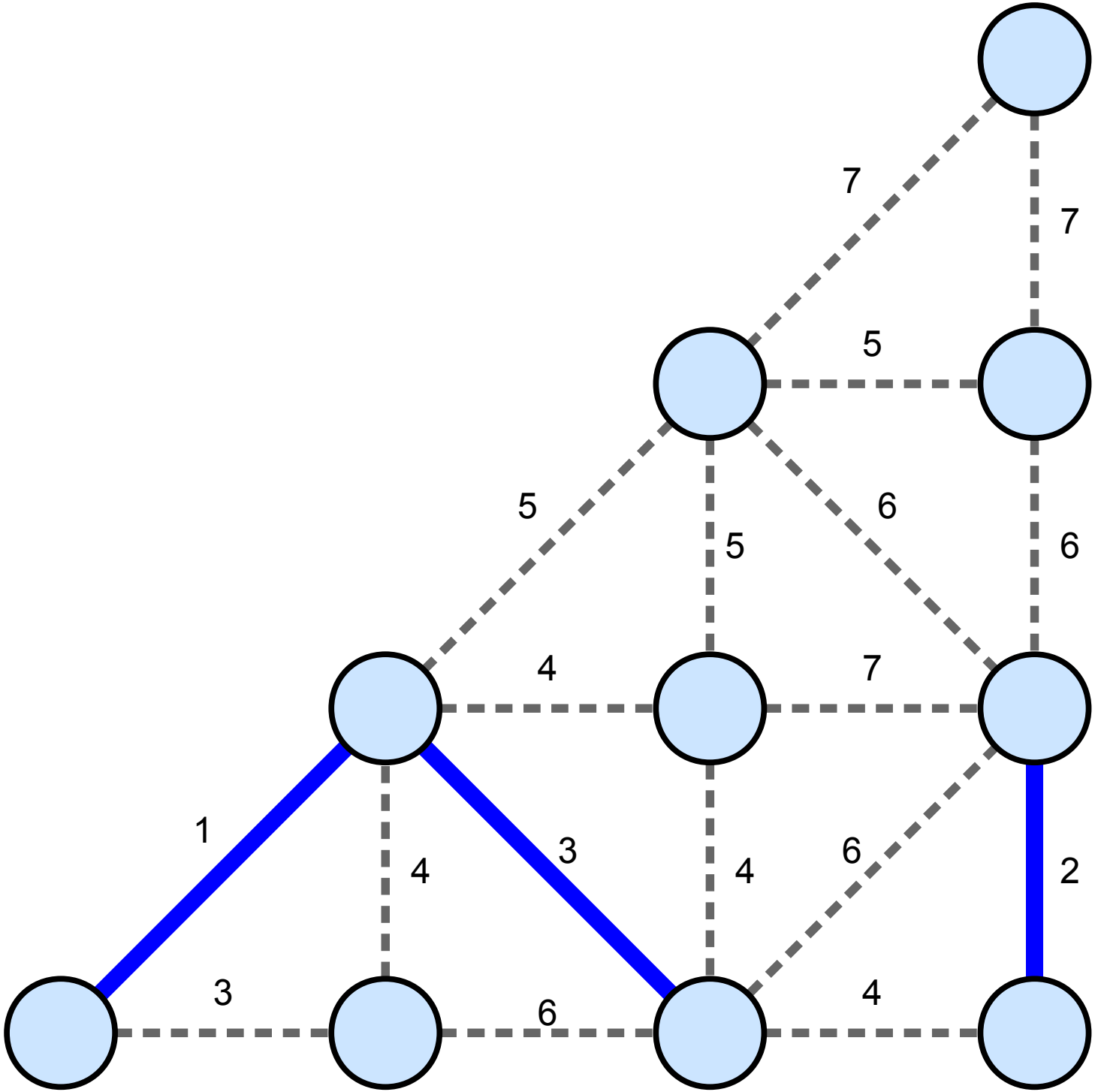

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

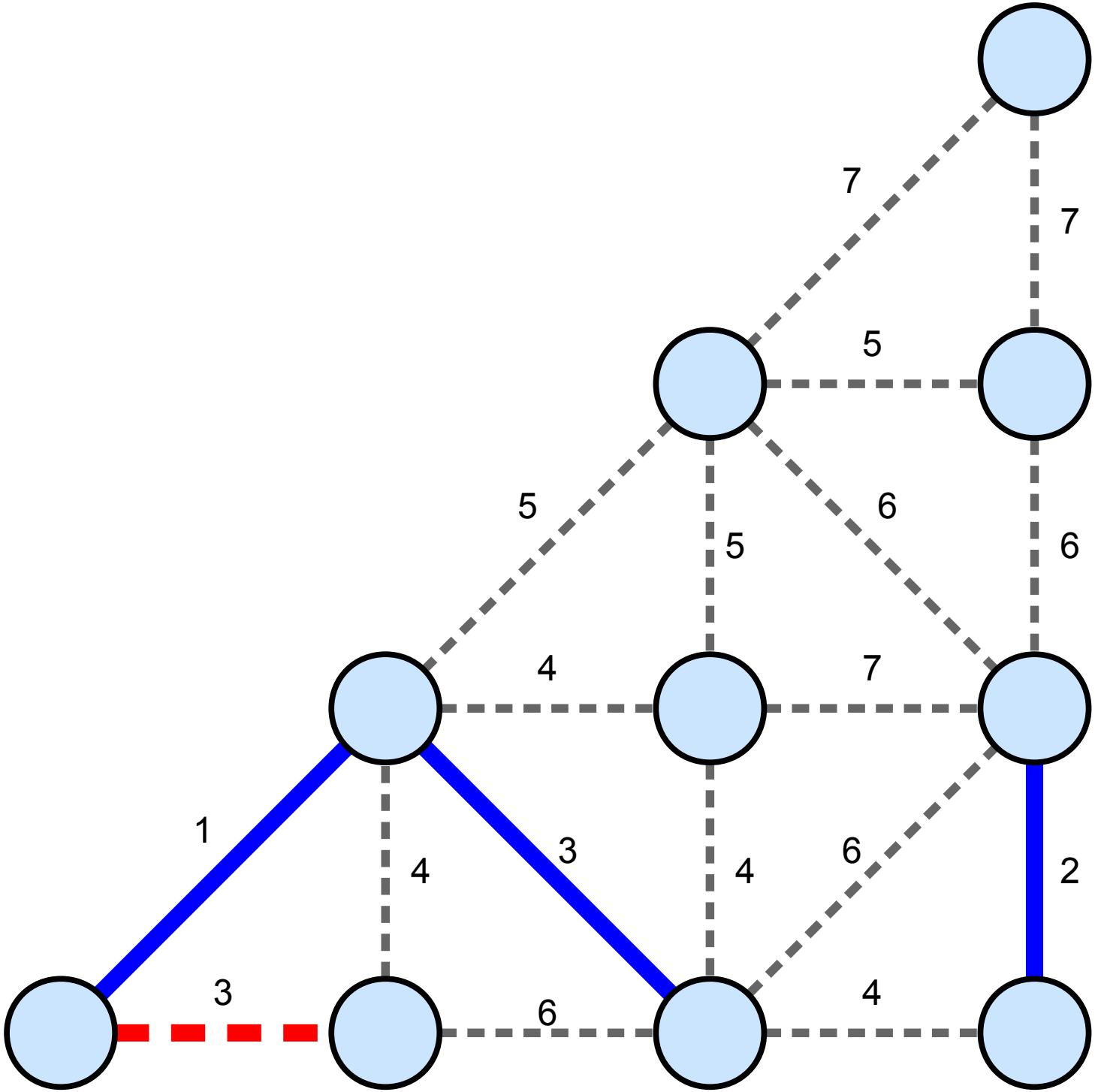

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

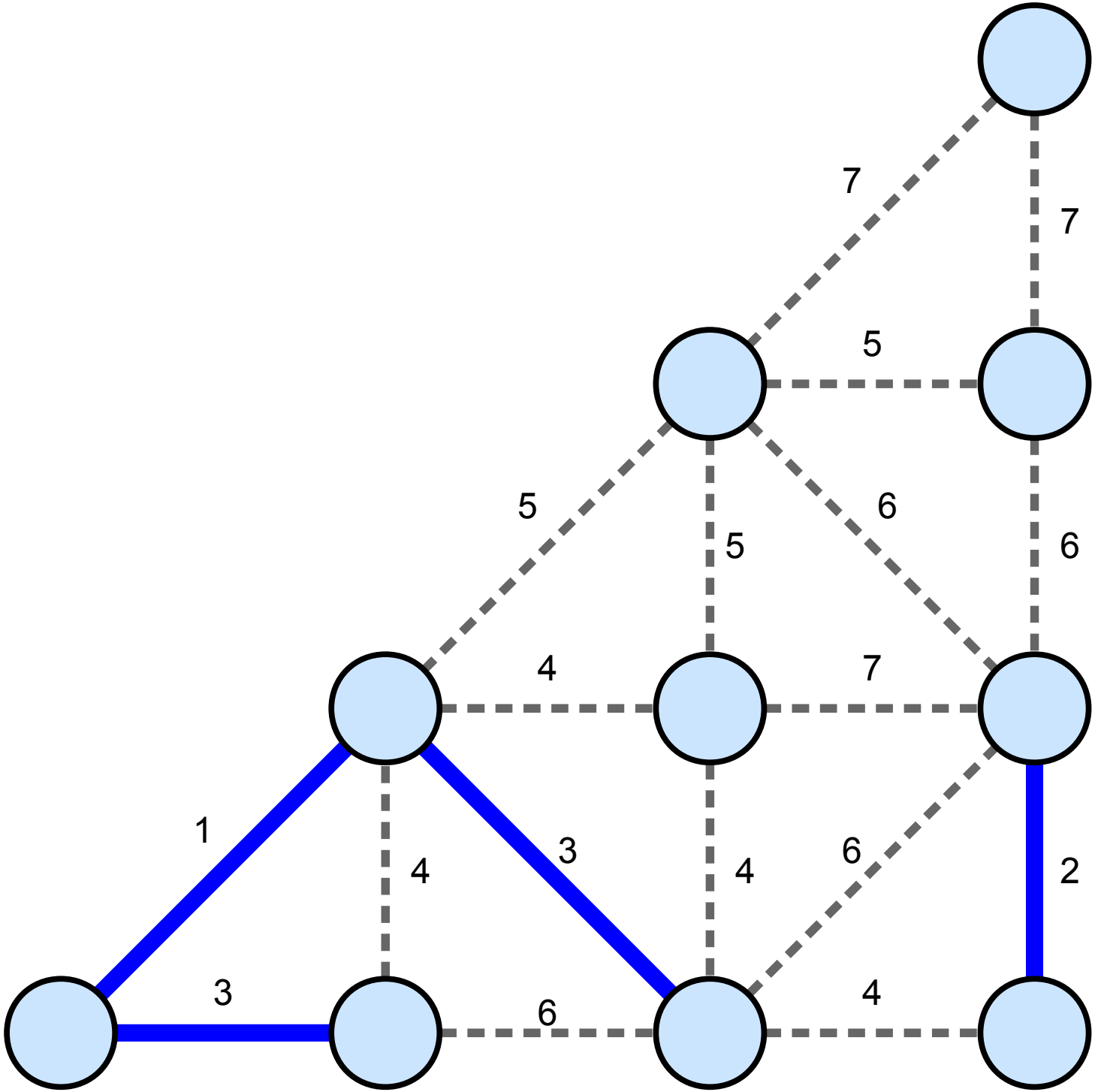

7
7
5
5
6
5
6
4
7
1
4
3
4
6
2
3
6
4

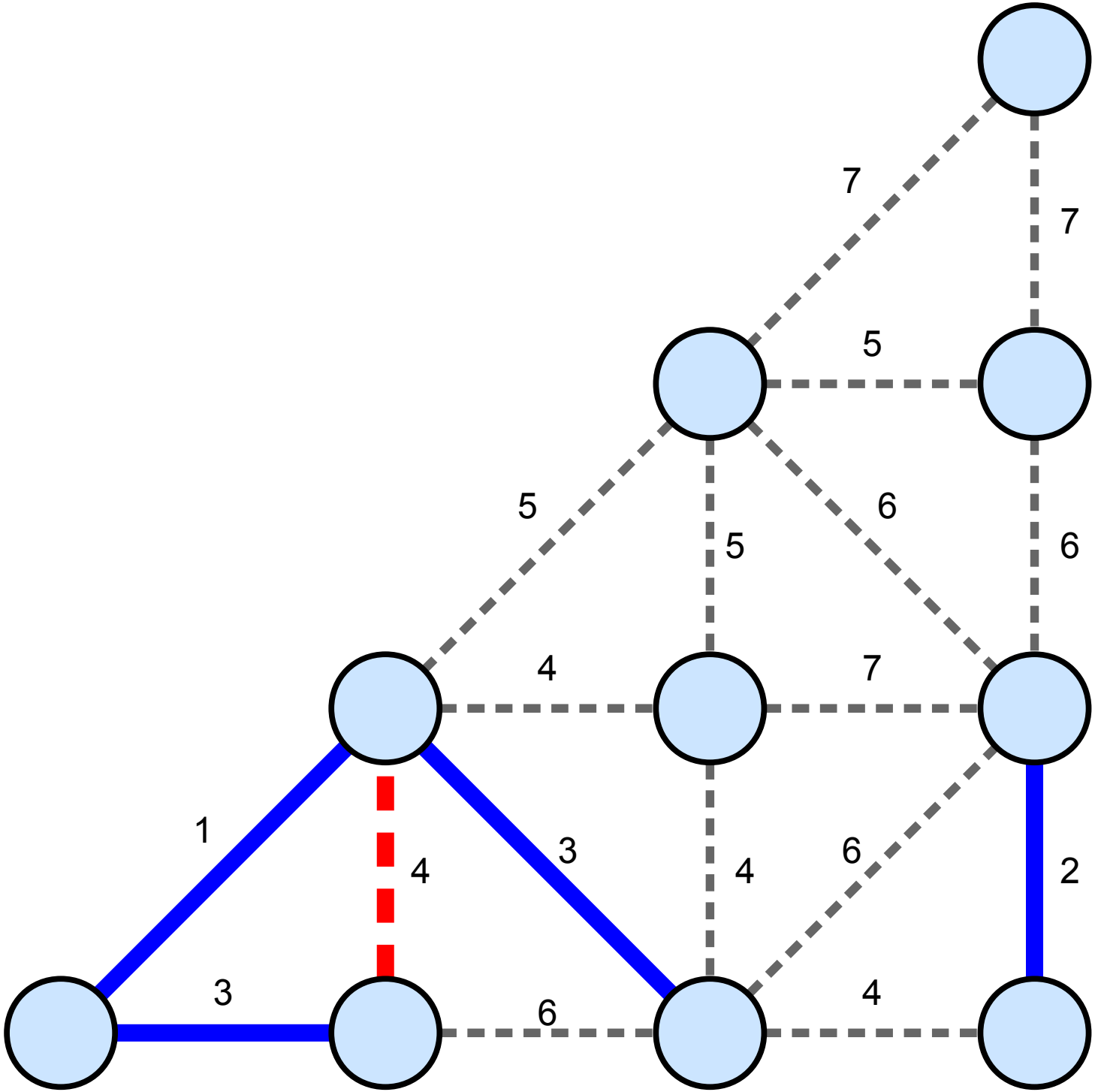

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

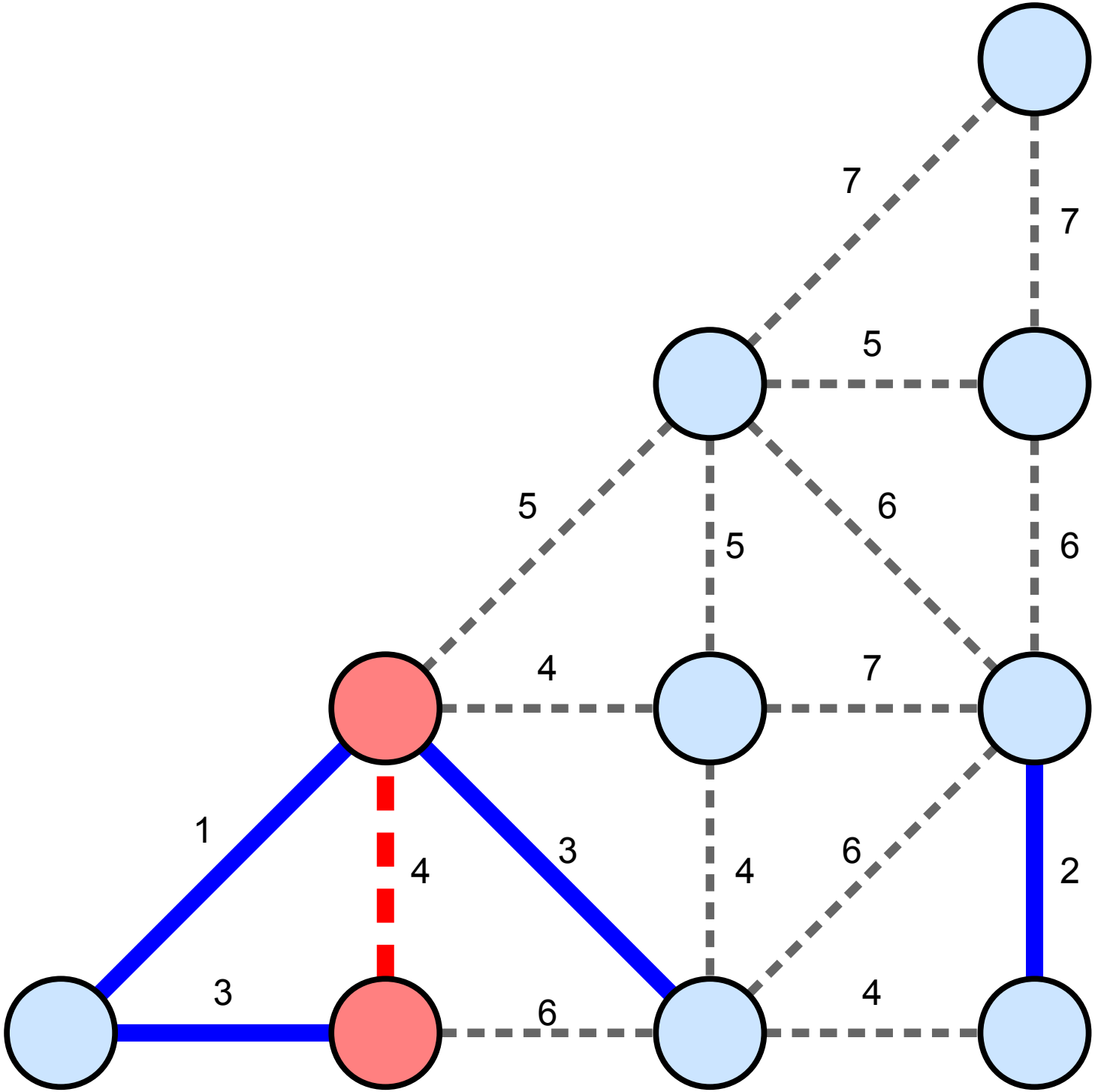

7
7
5
5
5
6
6
4
7
1
3
4
4
6
2
3
6
4

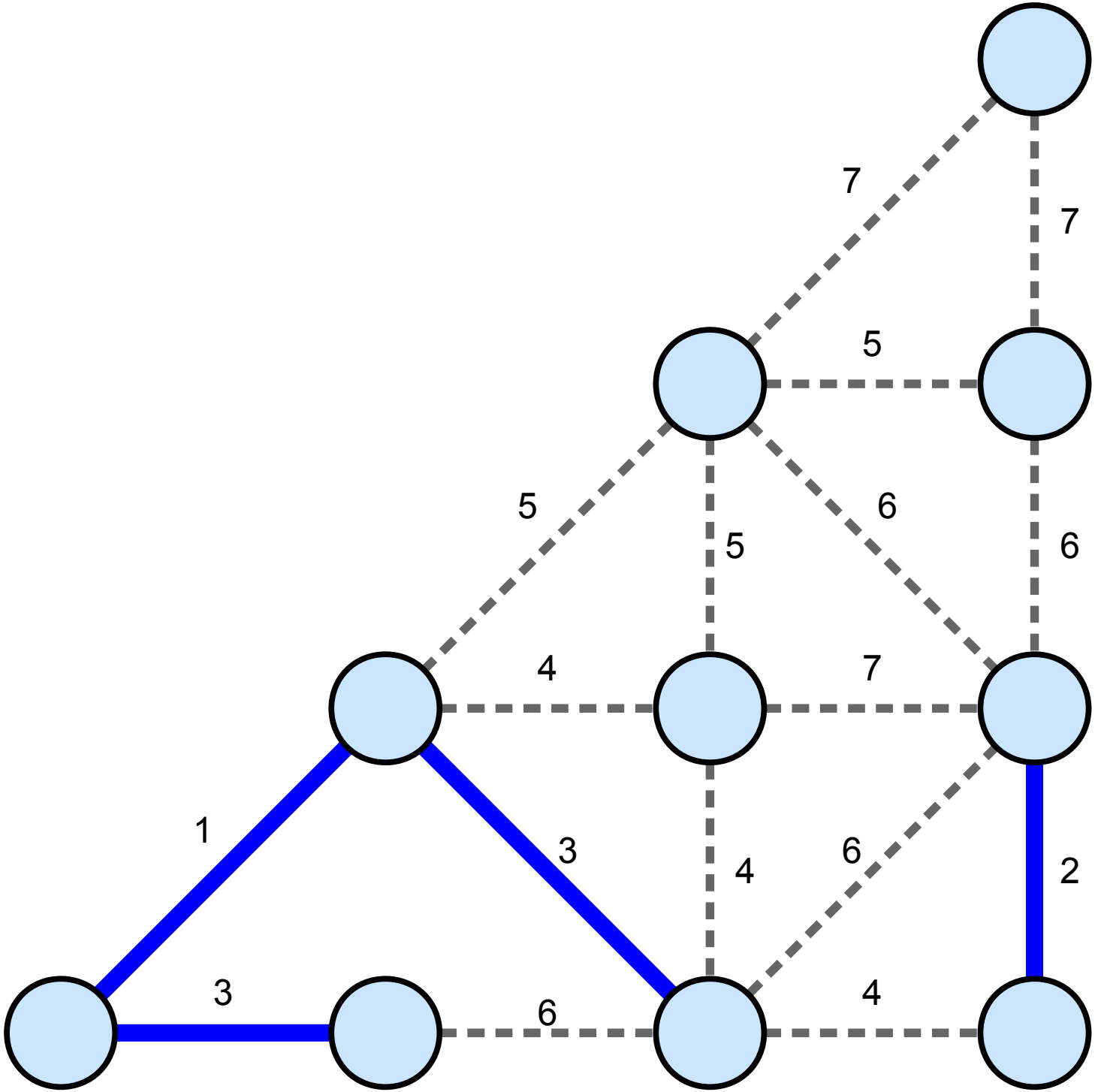

7
7
5
5
5
6
6
5
4
7
1
4
3
4
6
2
3
6
4

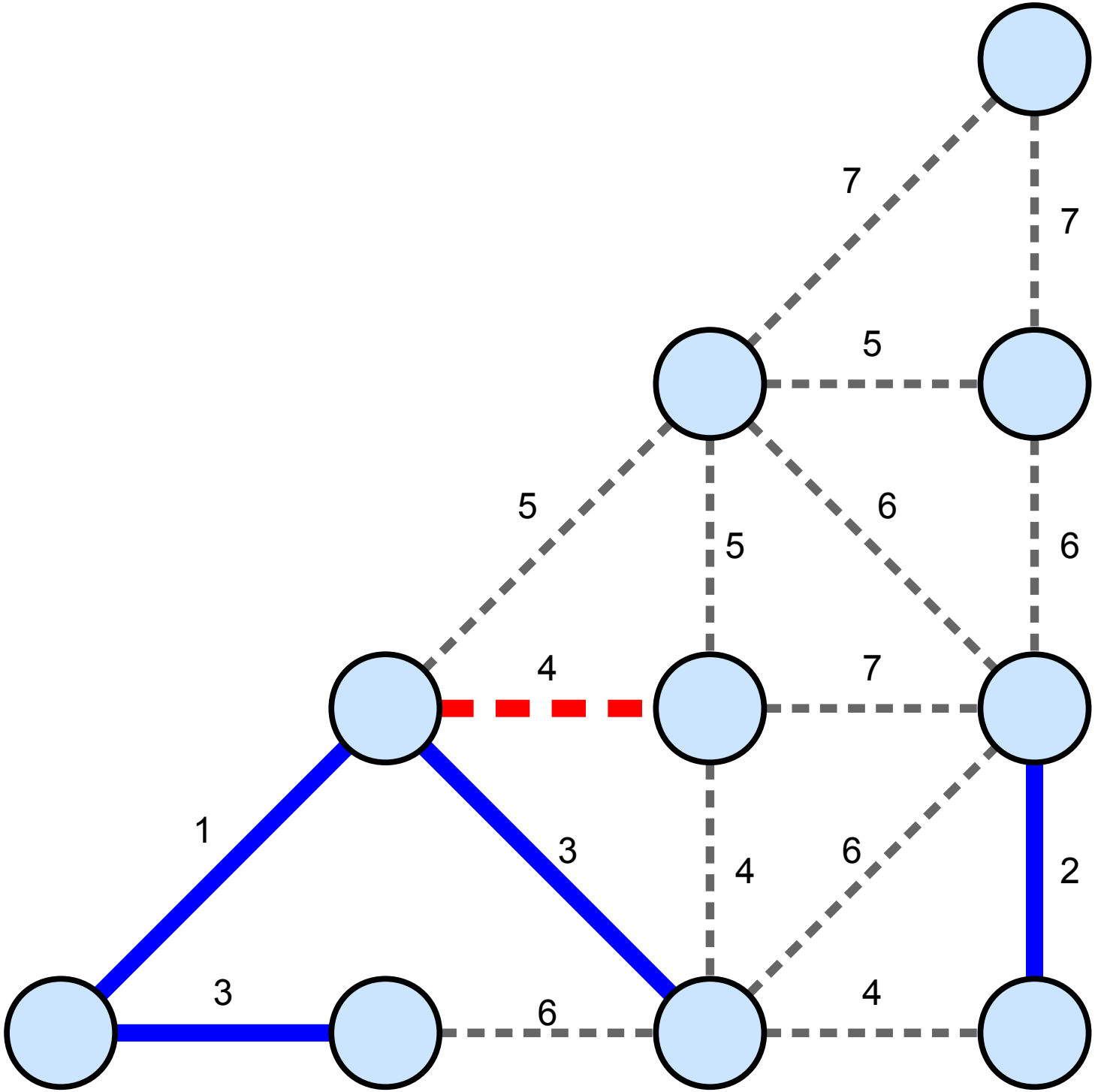

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

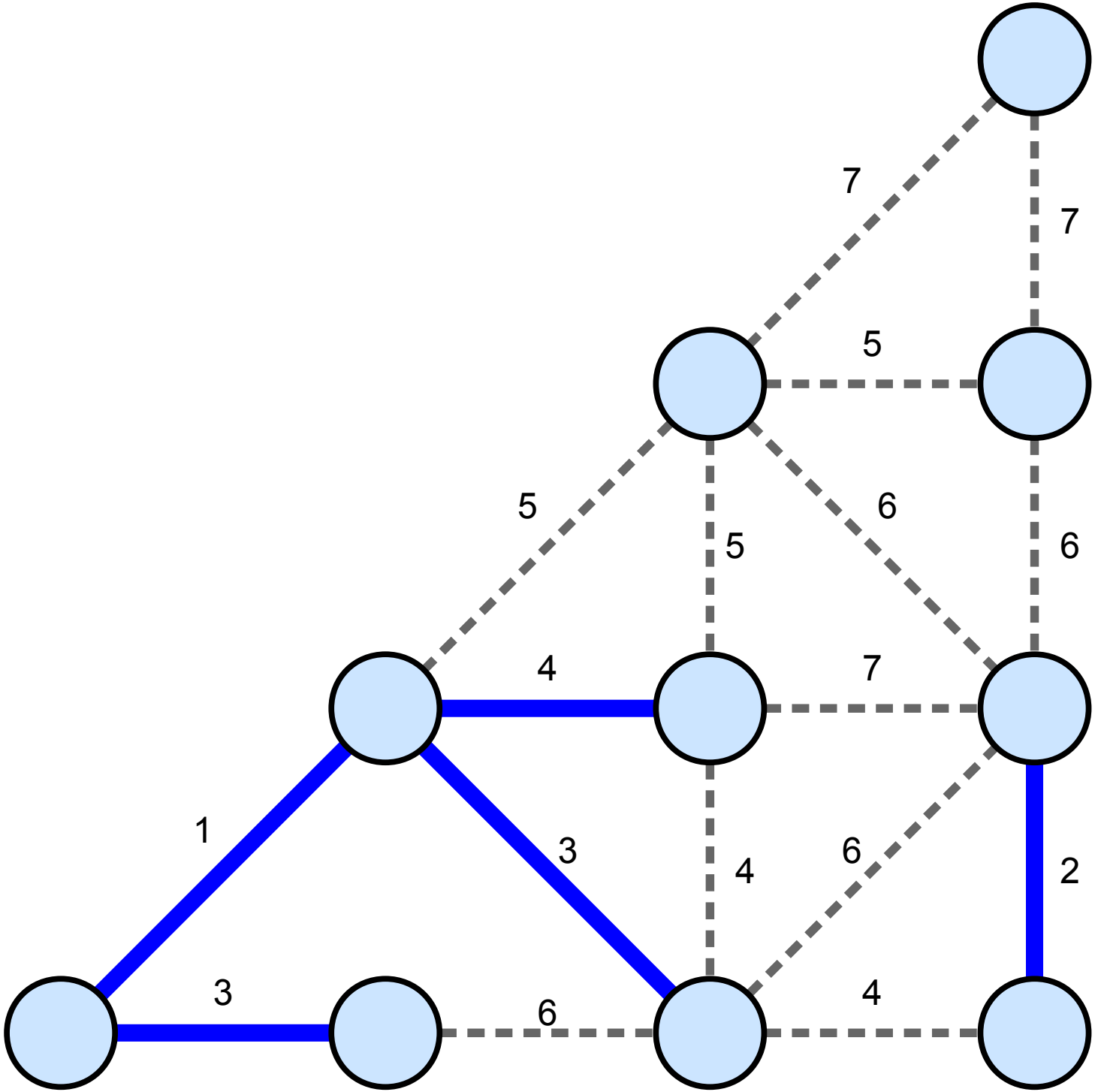

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

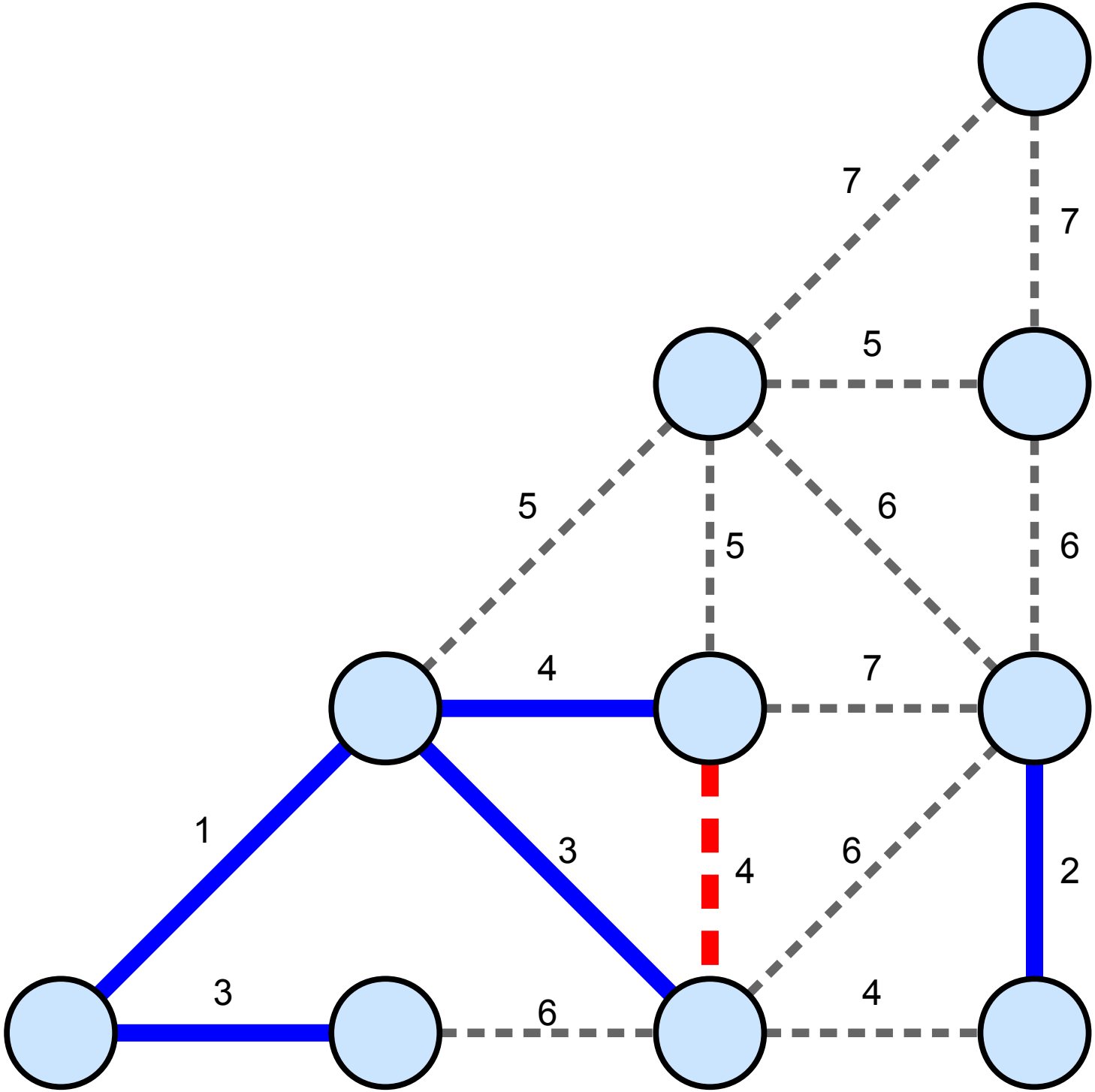

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

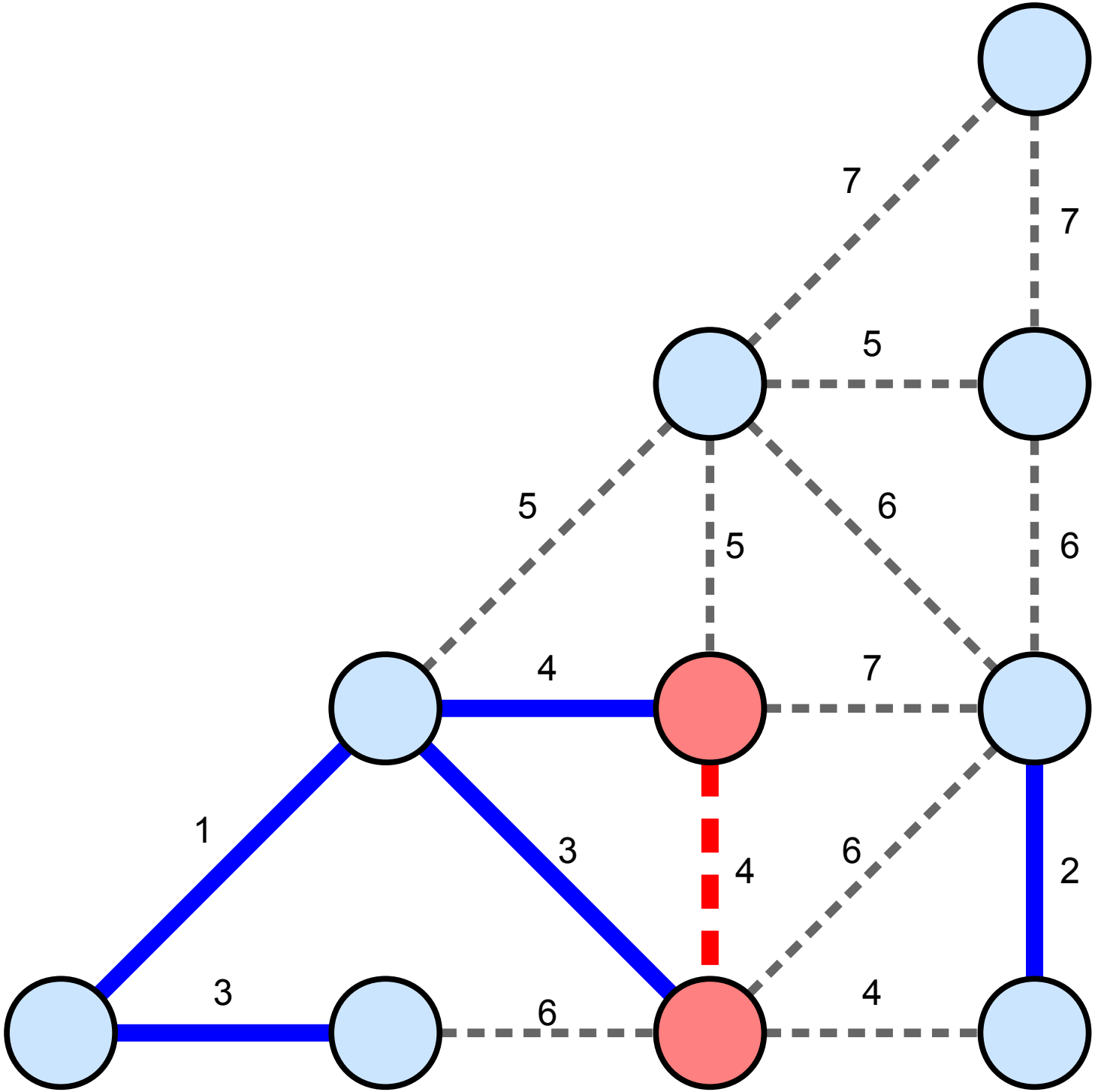

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4


7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

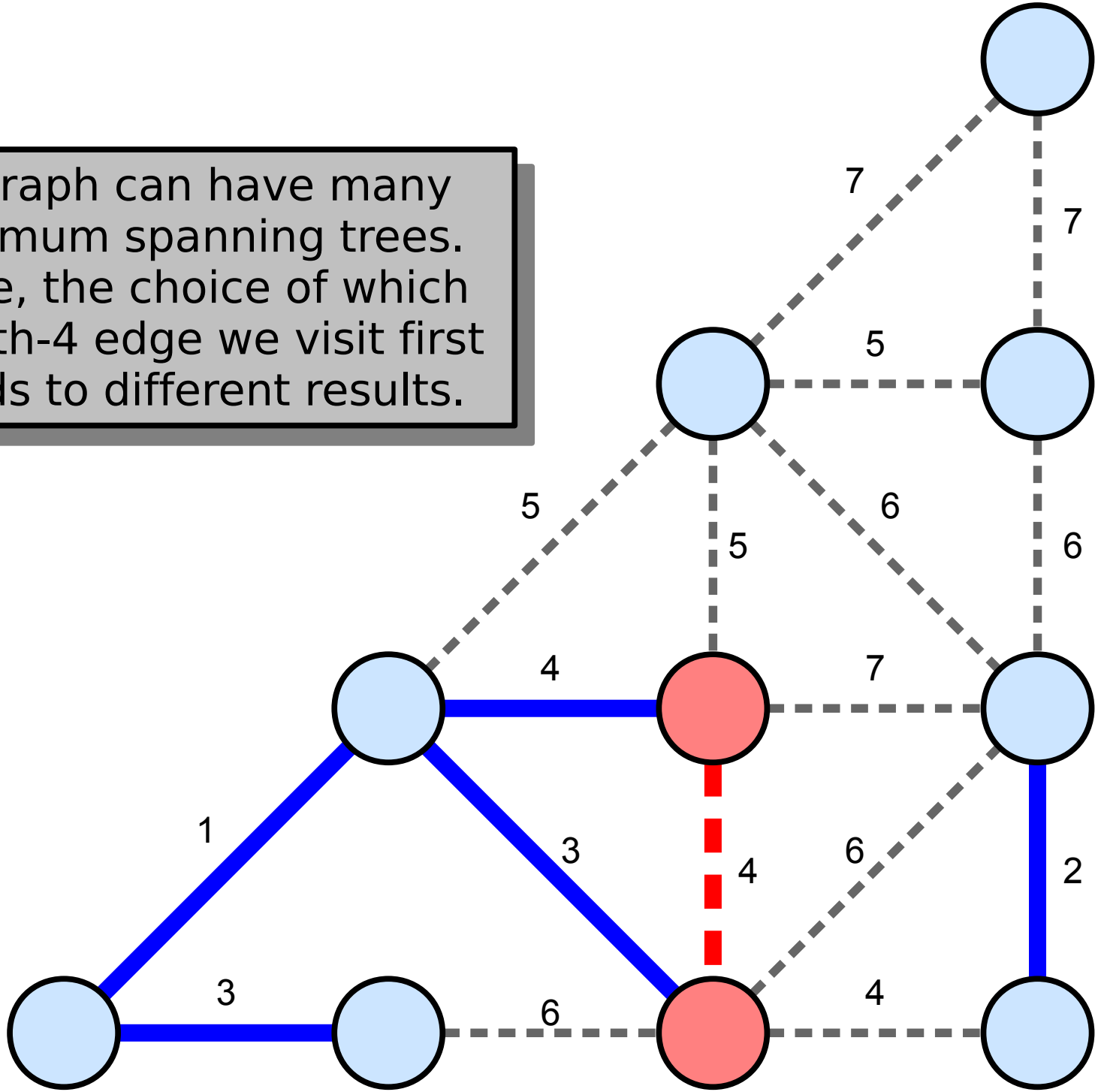

7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

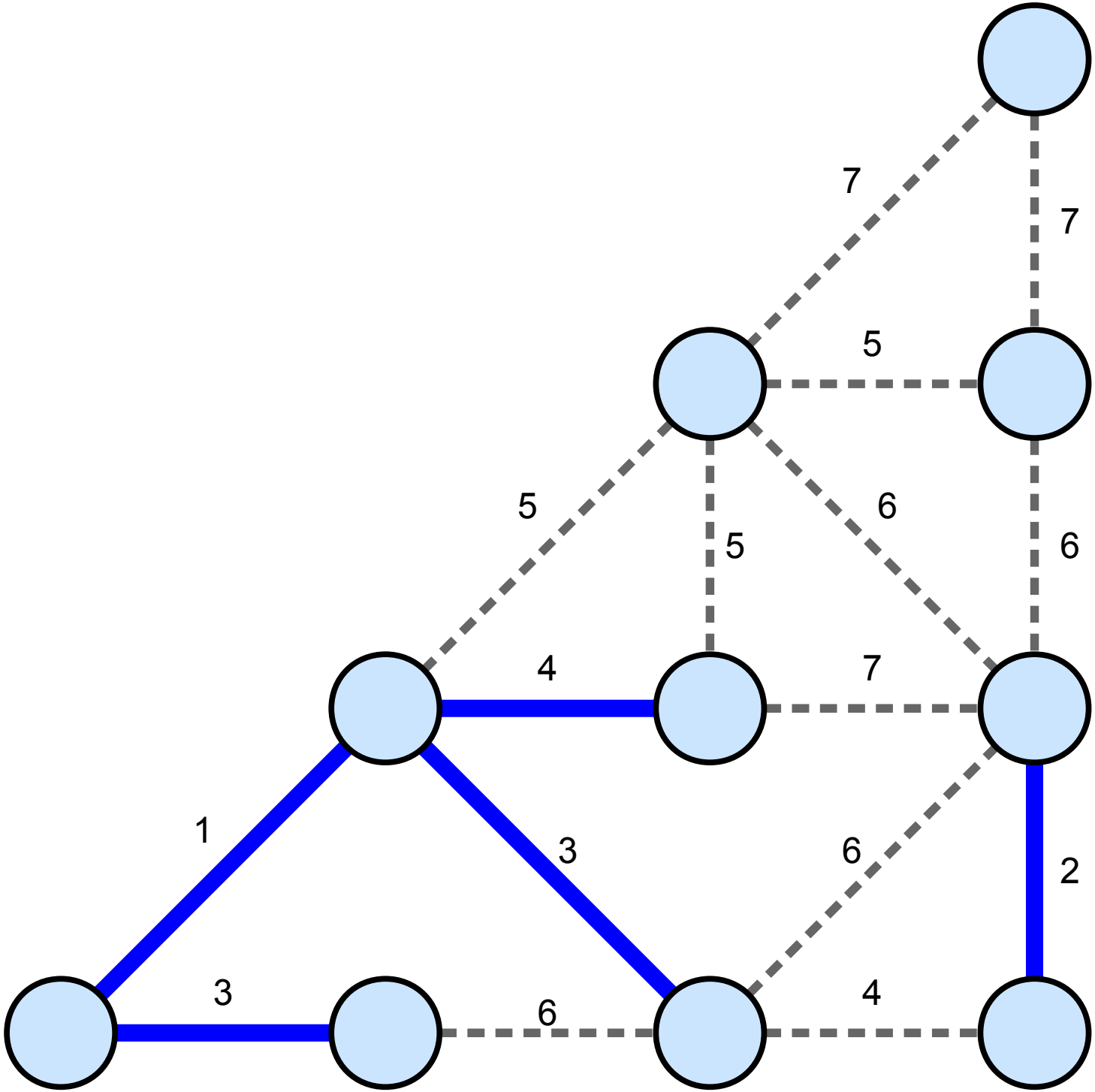

7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

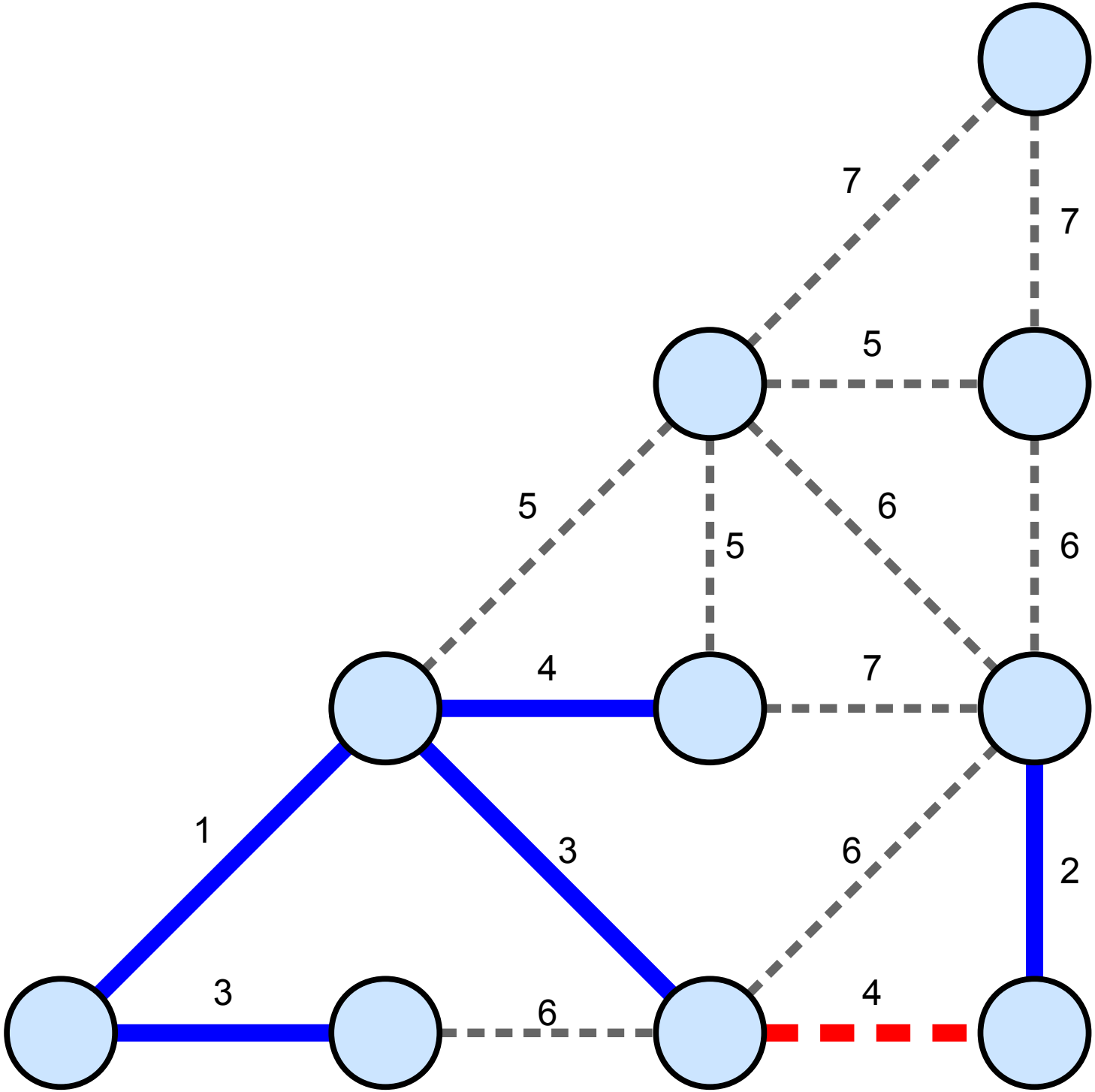

7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

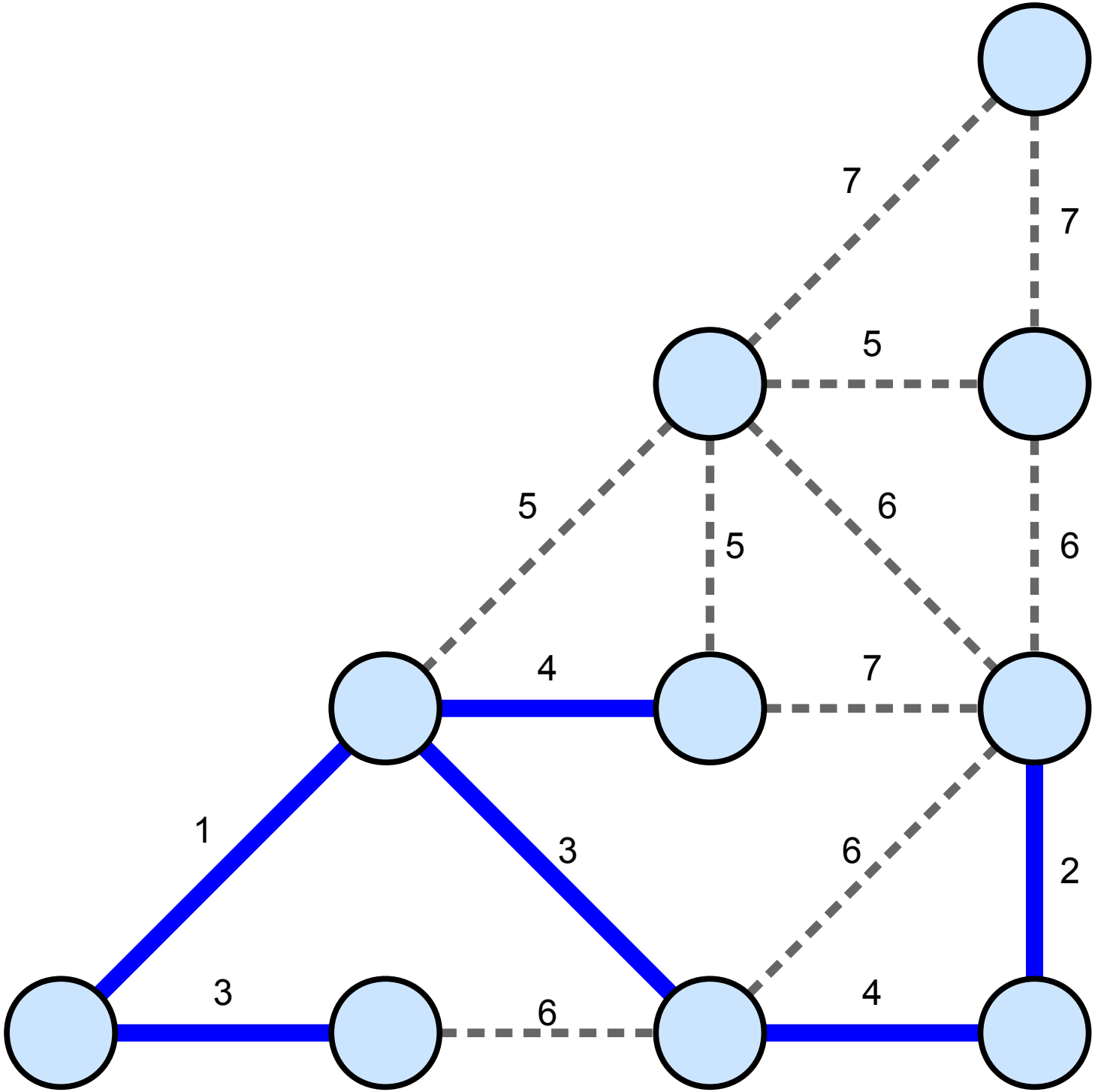

7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

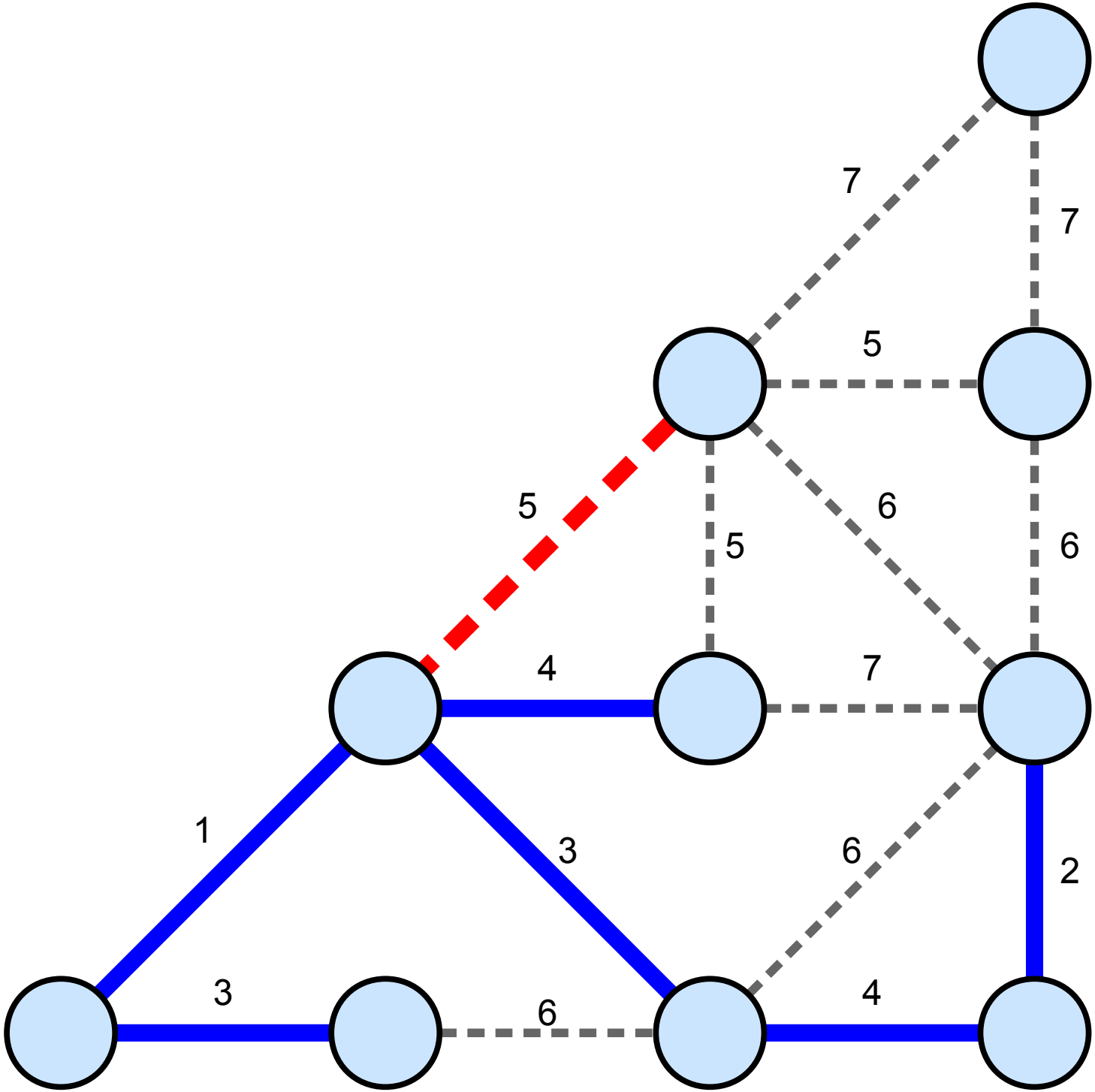
A graph can have many minimum spanning trees. Here, the choice of which length-4 edge we visit first leads to different results.

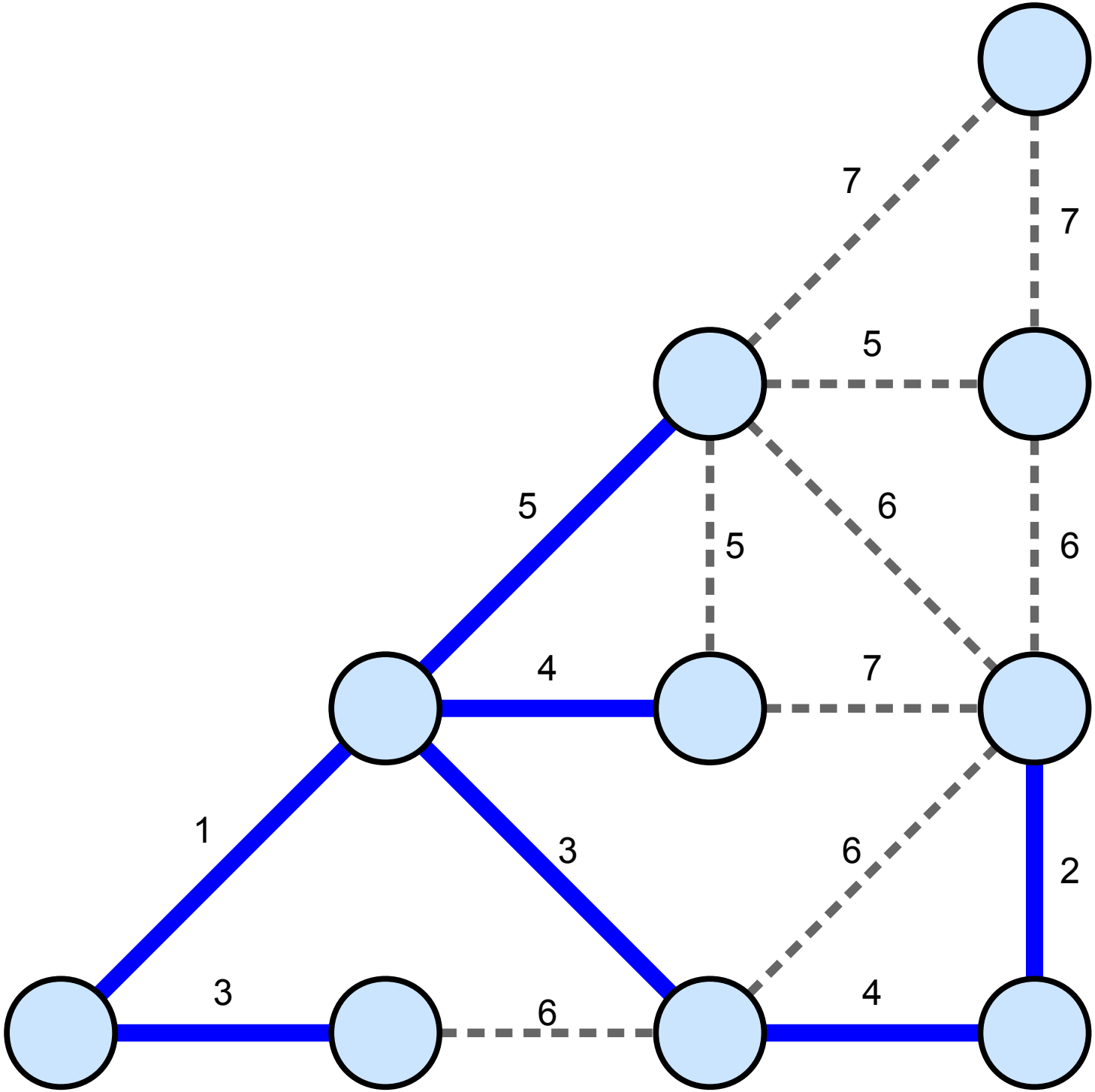


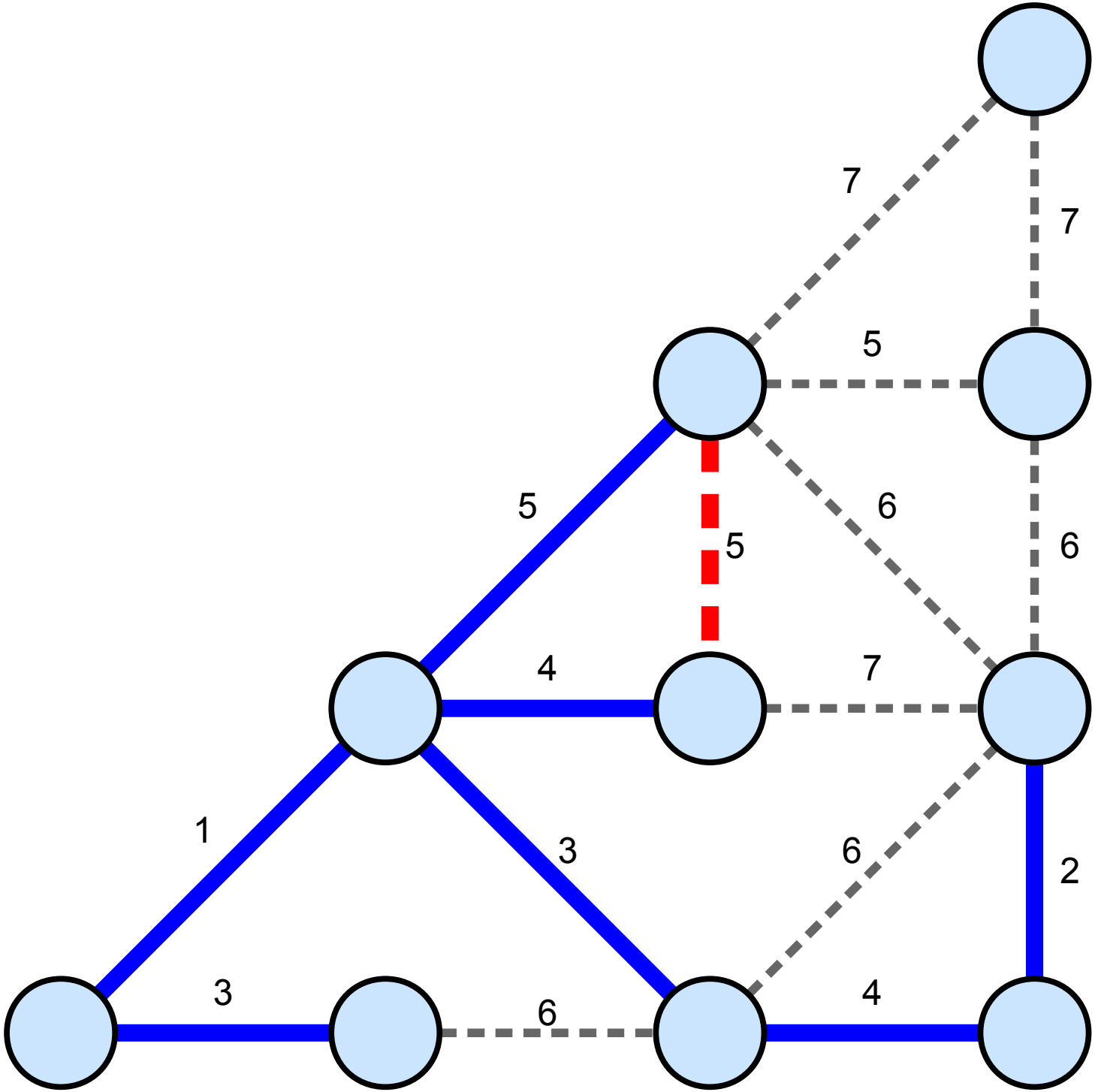

7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

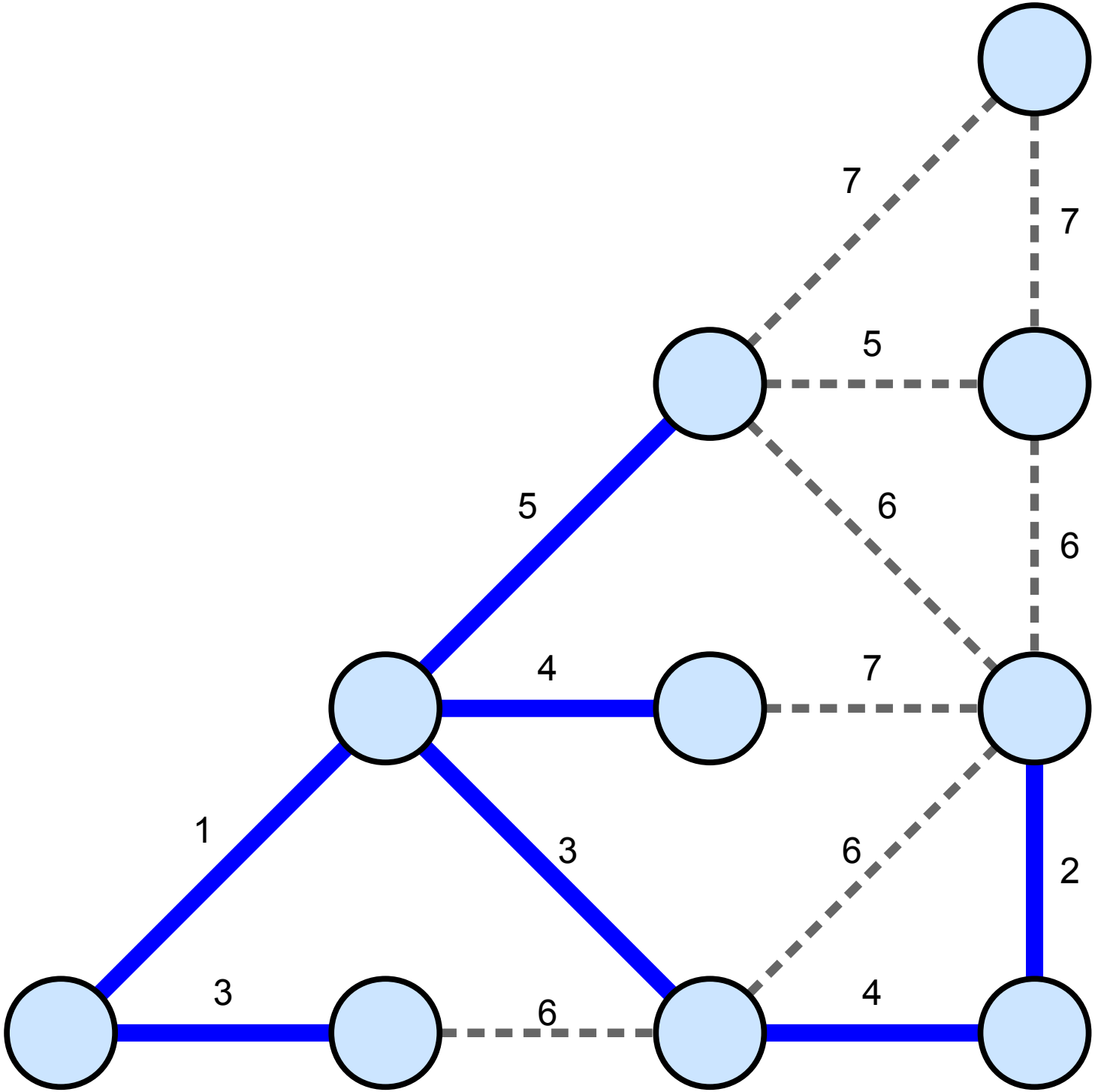

7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

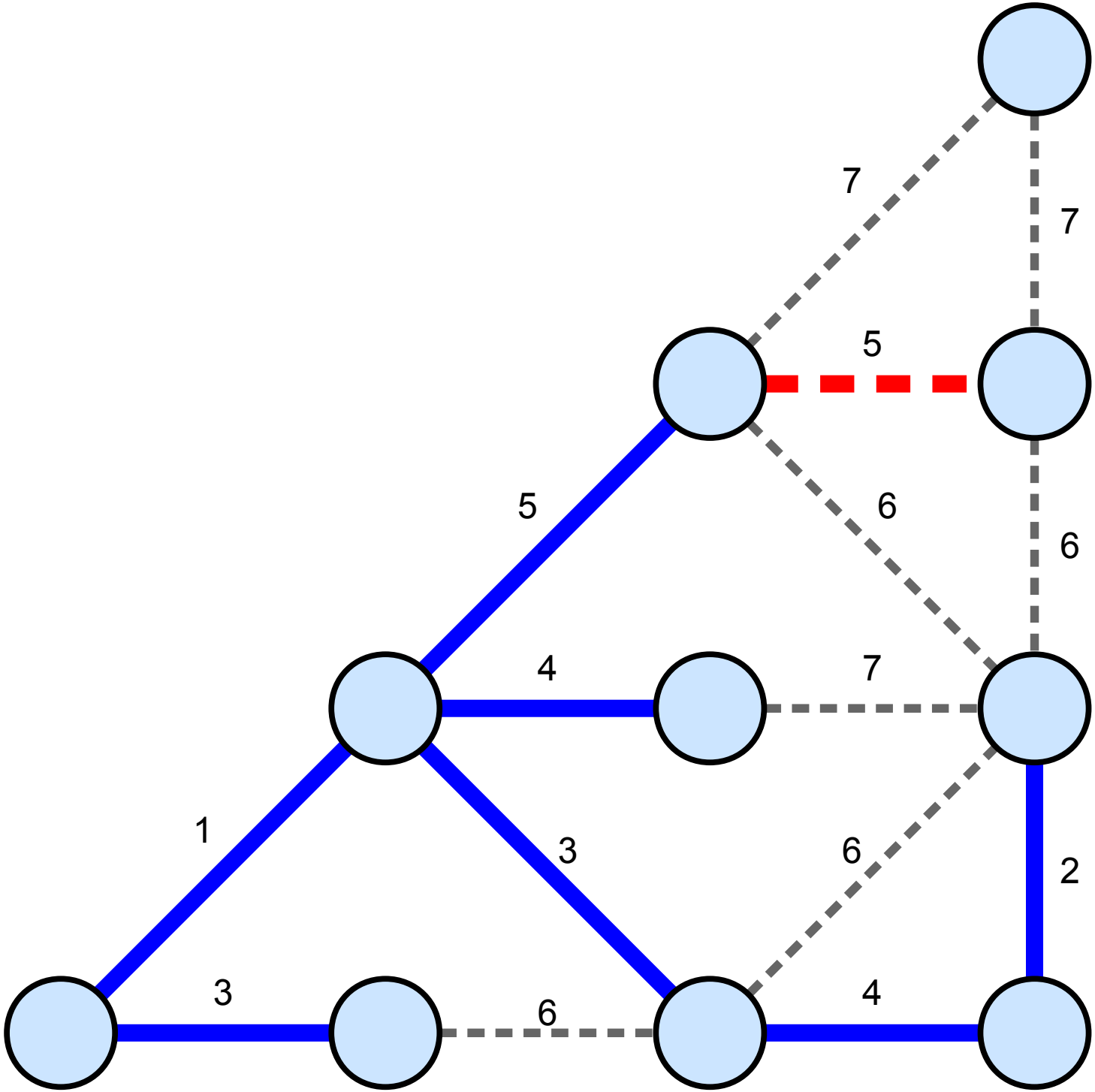

7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

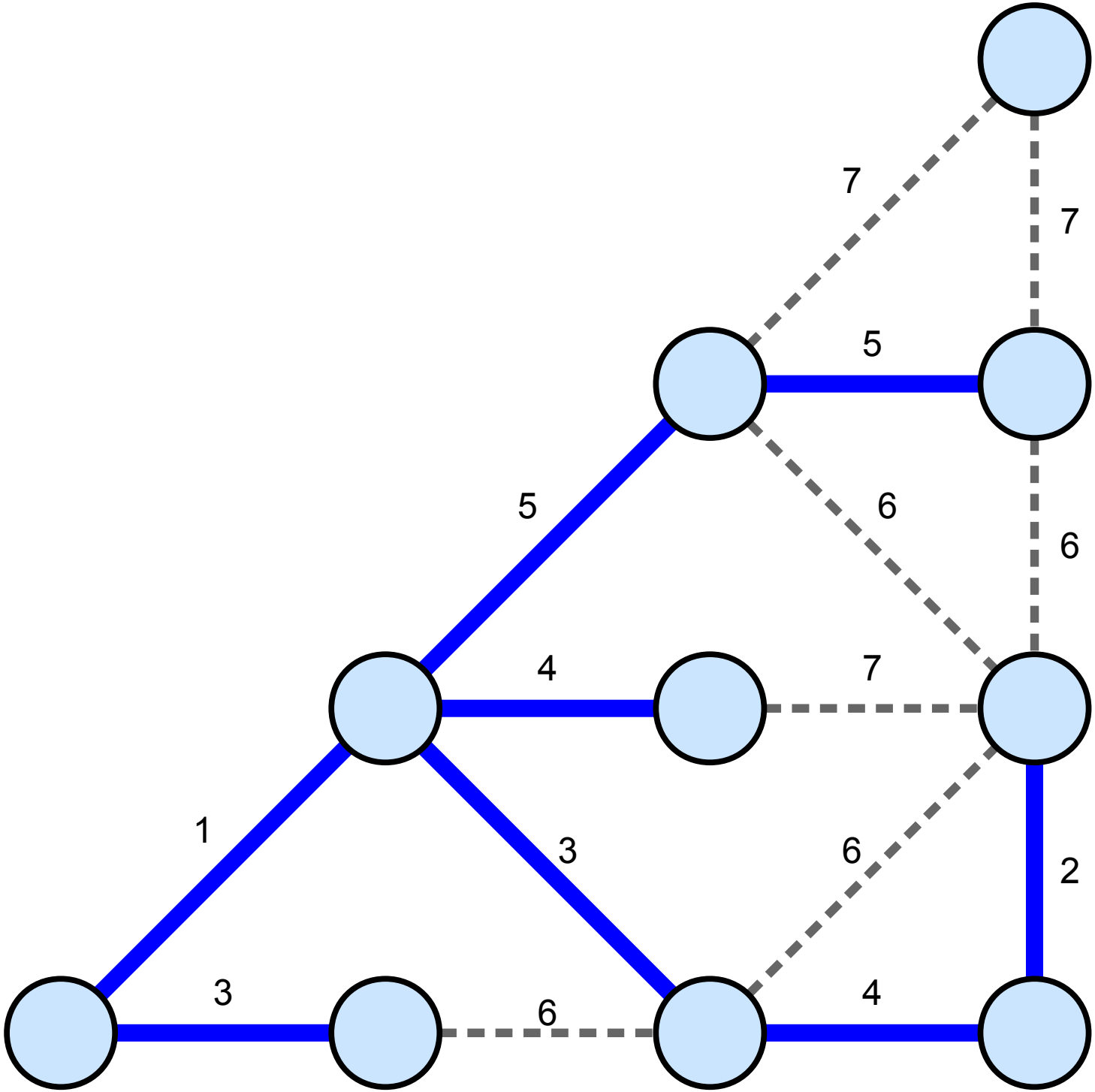

7
7
5
5
6
5
6
4
7
1
3
6
6
2
3
6
4

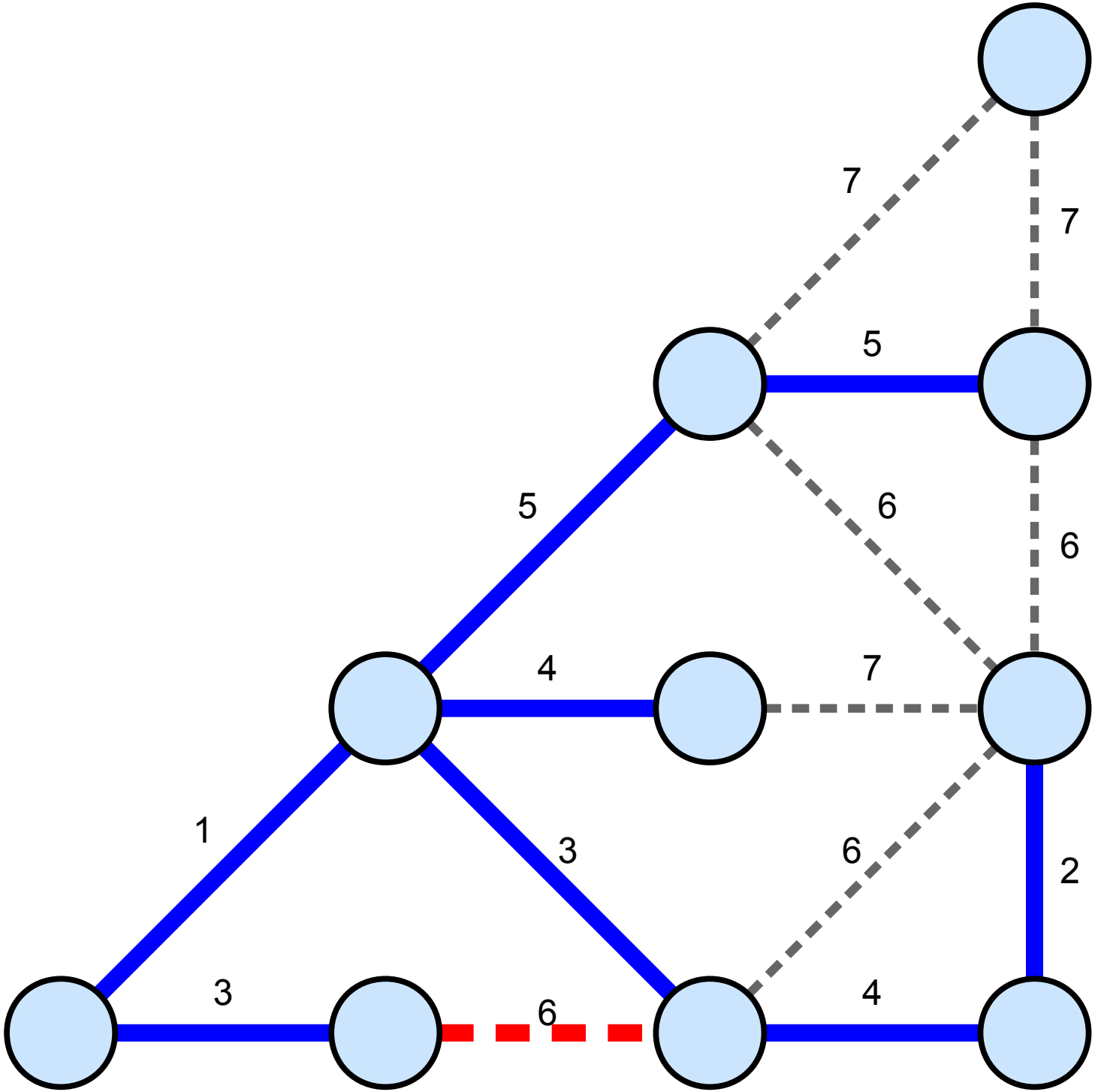

7
7
5
5
6
5
6
4
7
1
3
6
2
3
6
4

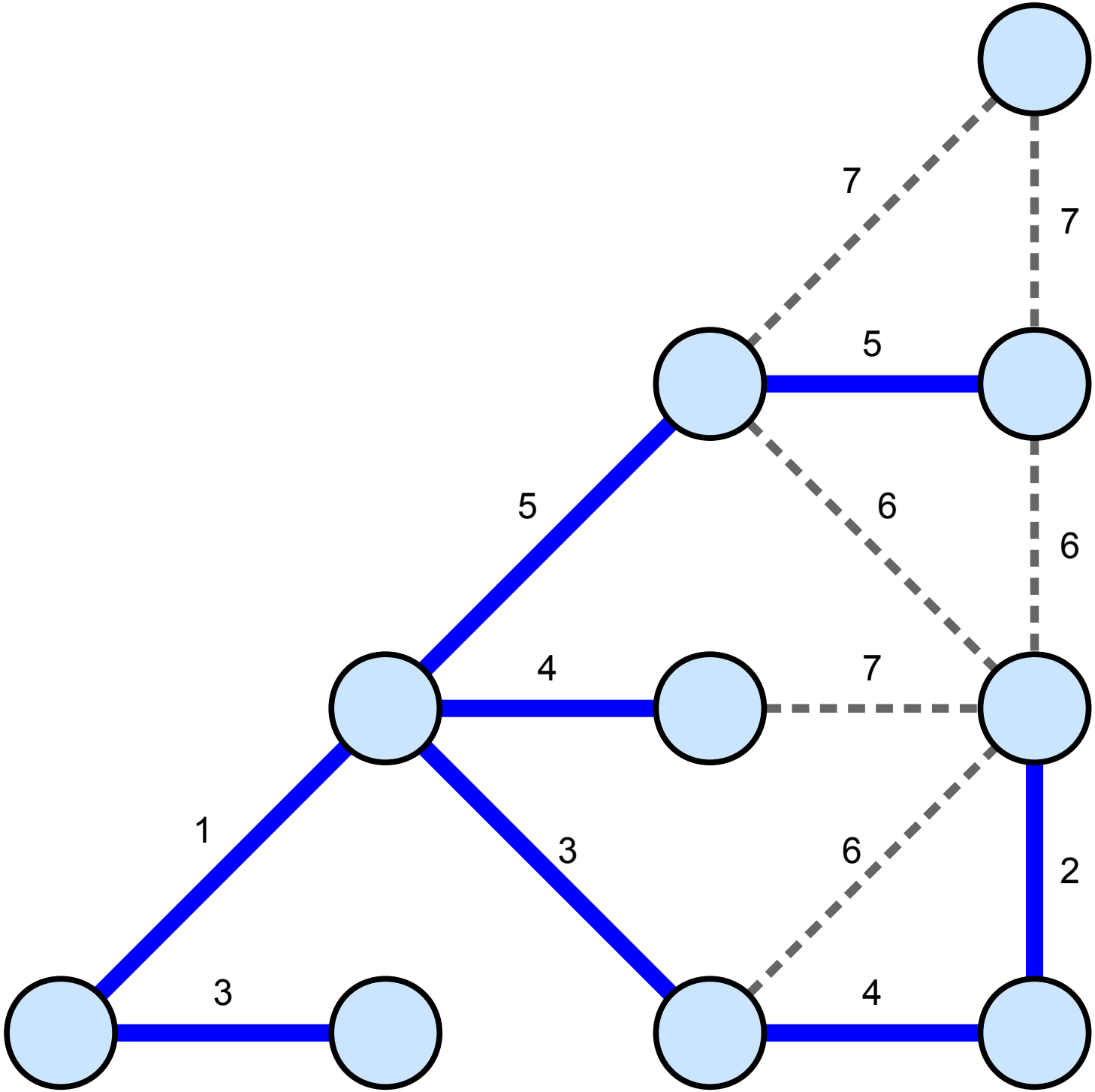

7
7
5
5
6
6
5
4
7
1
3
6
2
3
6
4

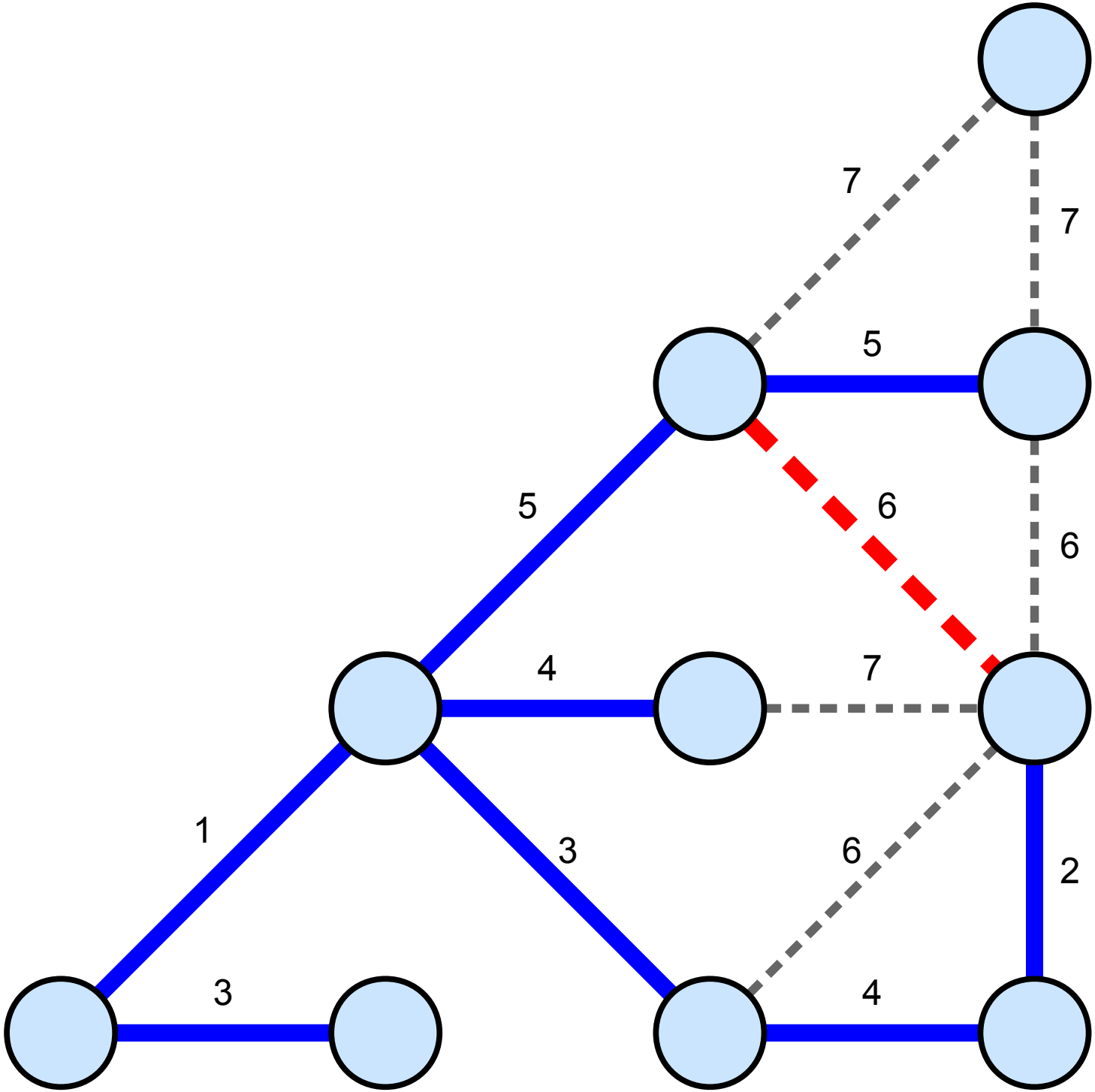

7
7
5
5
6
6
4
7
1
3
6
2
3
6
4

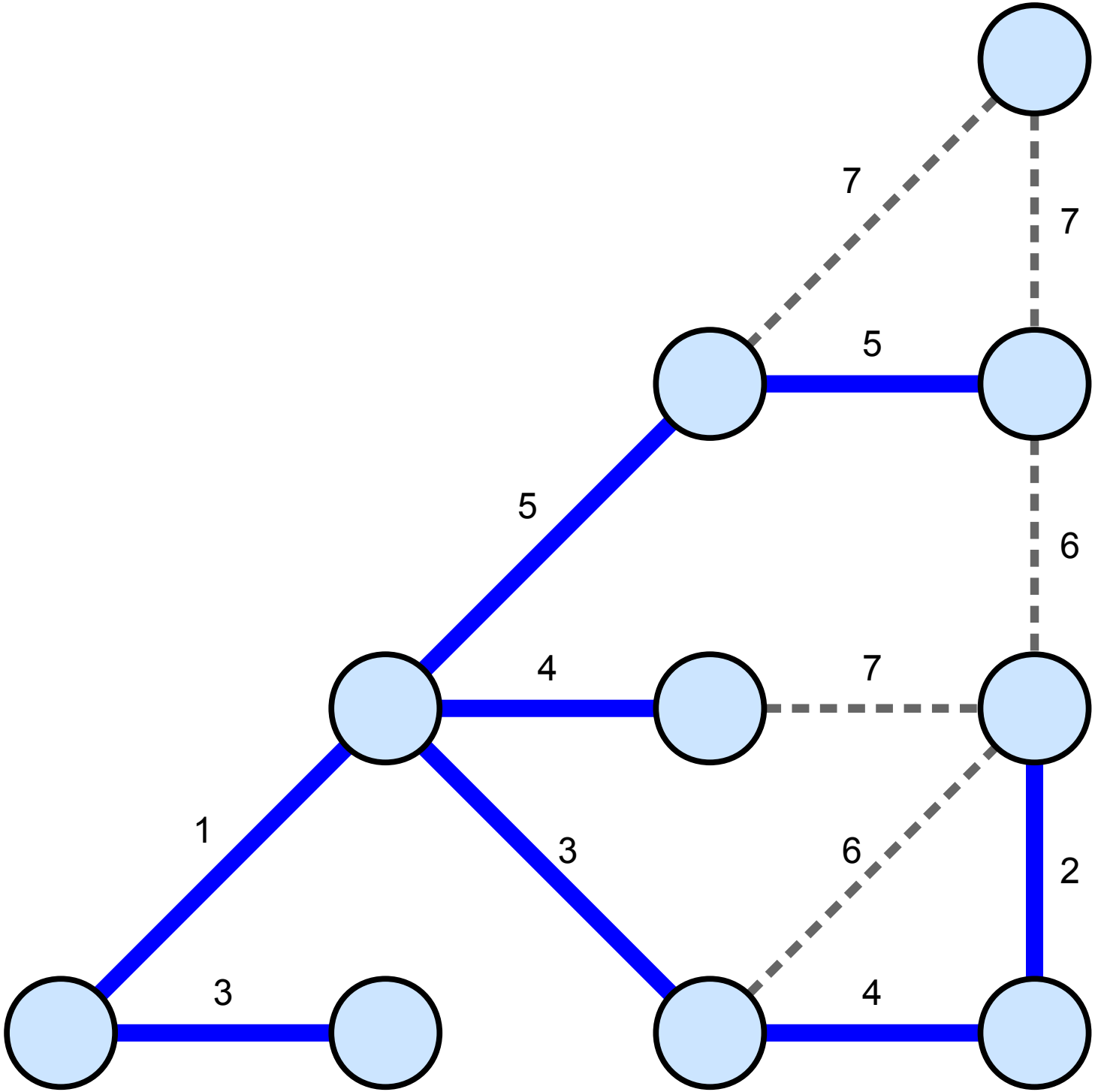

7
7
5
5
6
6
4
7
1
3
6
2
3
6
4

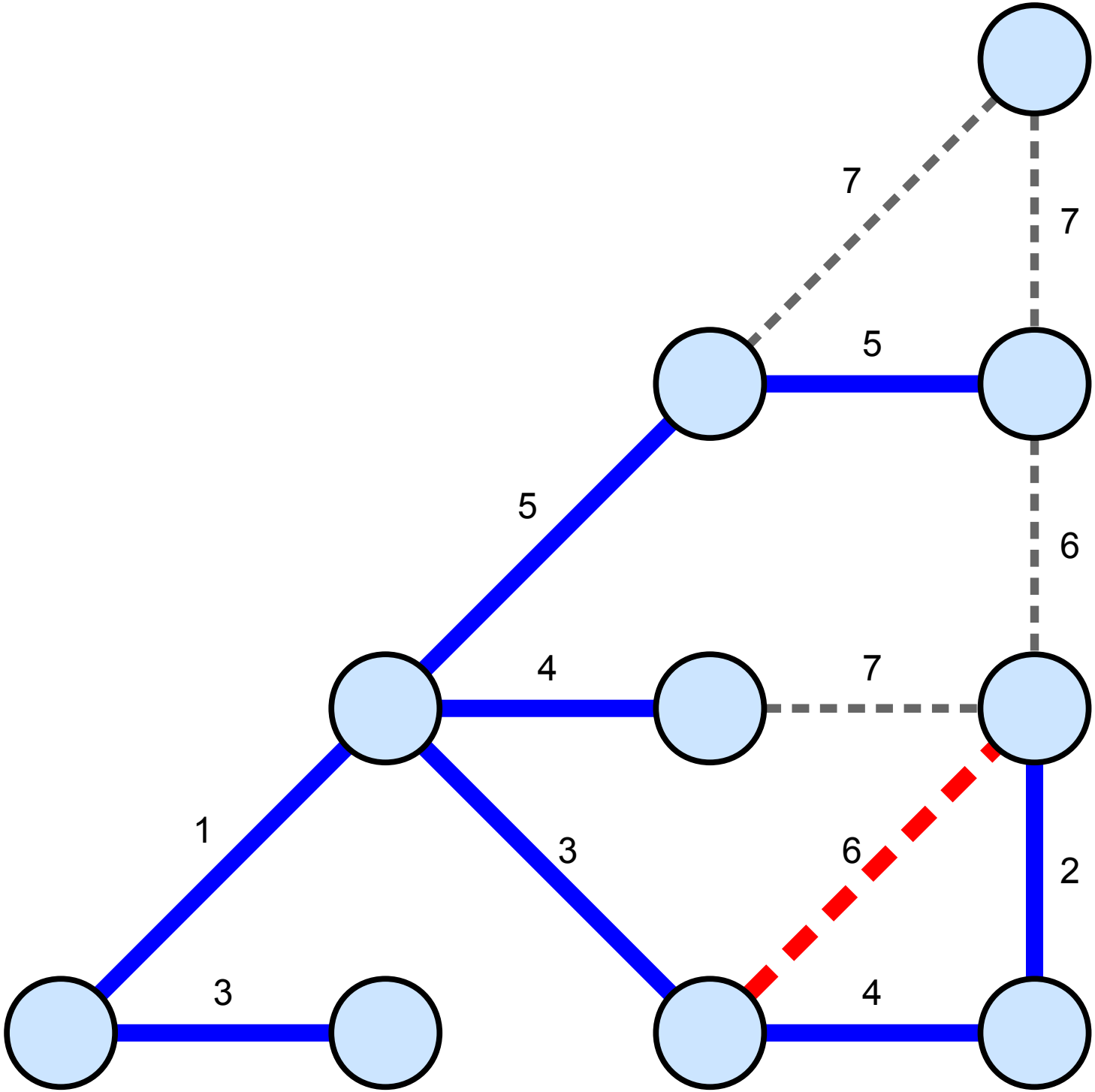

7
7
5
5
6
6
4
7
1
3
6
2
3
6
4

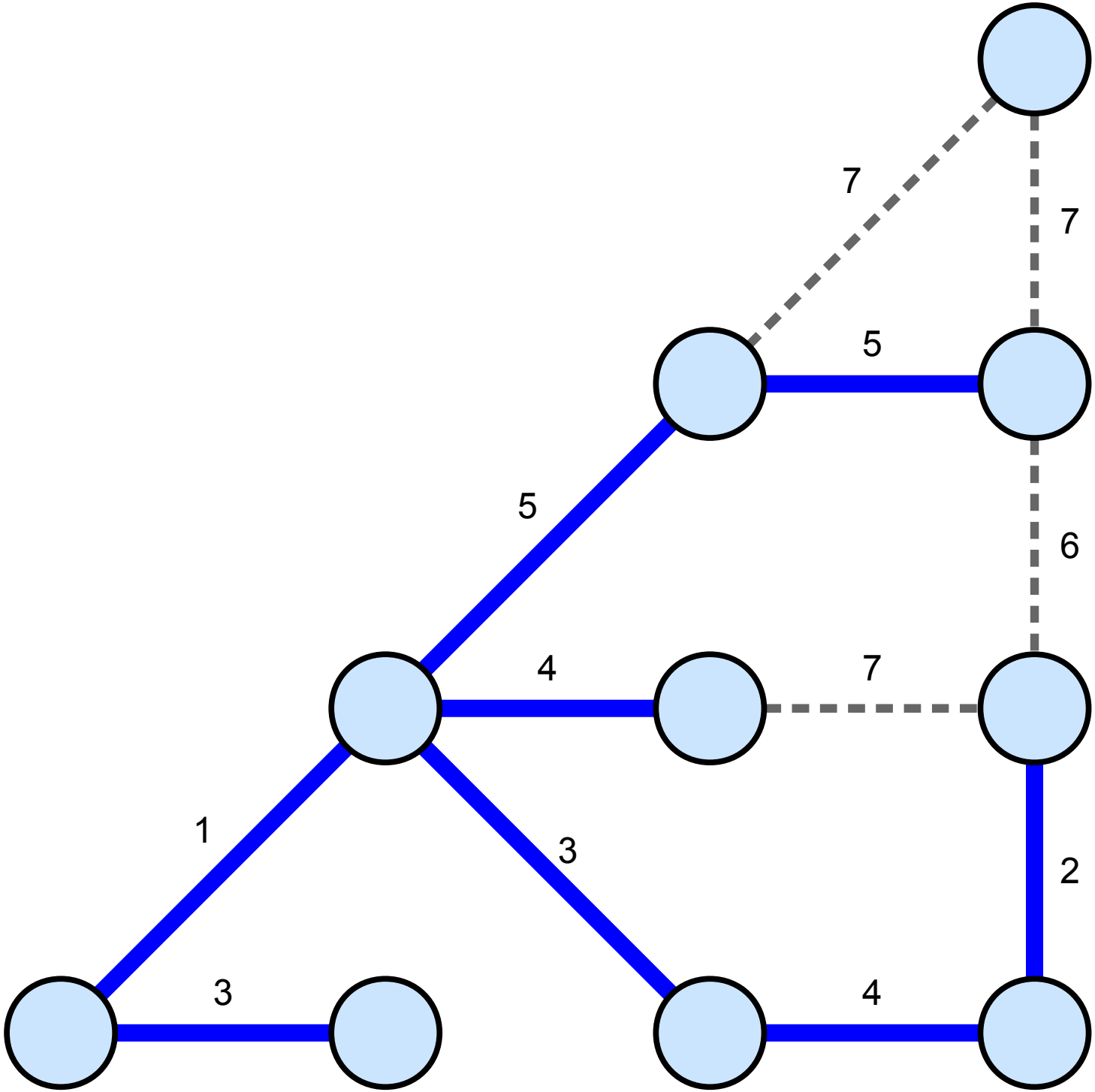

7
7
5
5
6
6
4
7
1
3
6
2
3
6
4

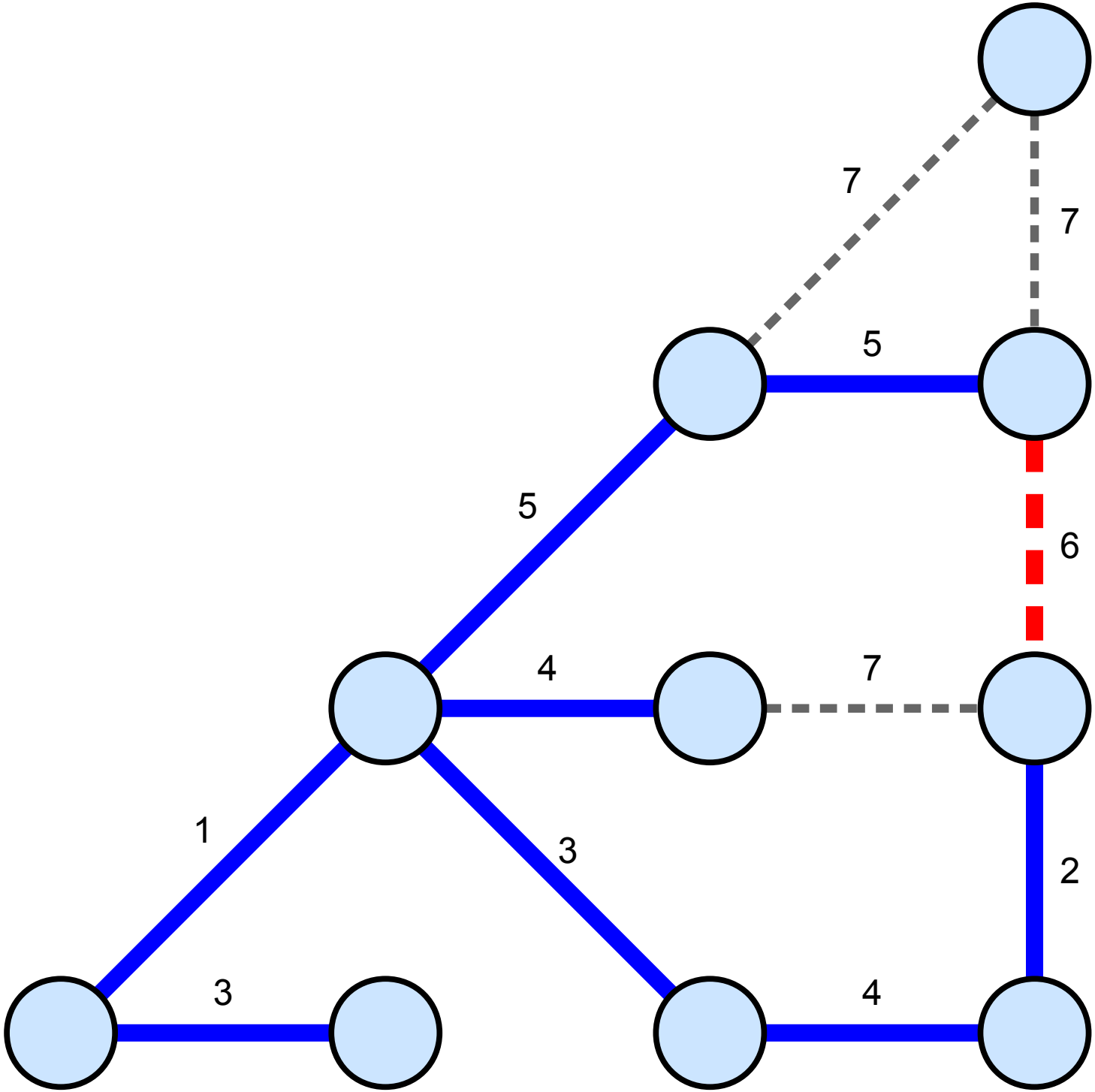

7
7
5
5
6
6
4
7
1
3
6
2
3
4

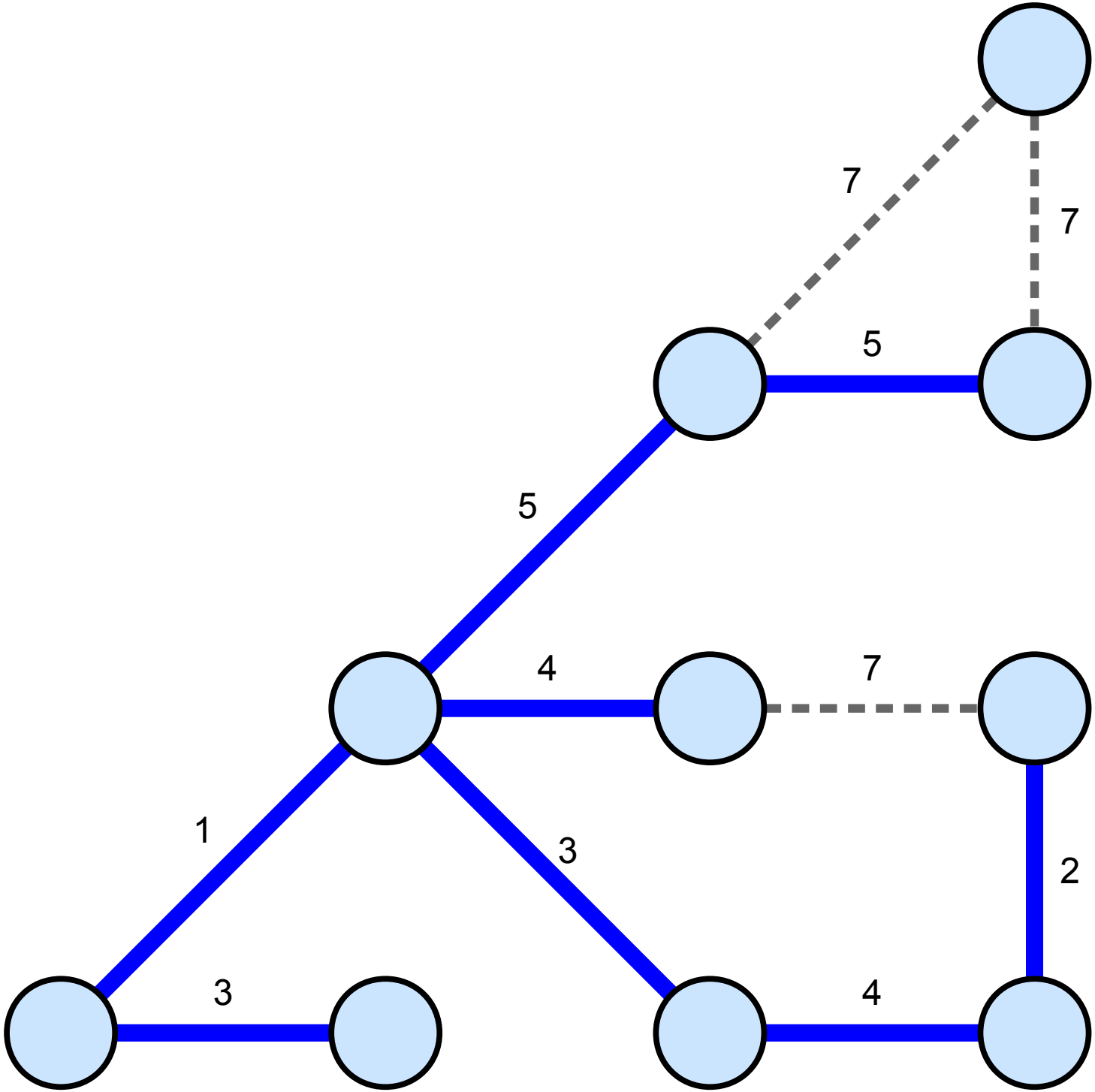

7
7
5
5
6
6
4
7
1
3
6
2
3
4

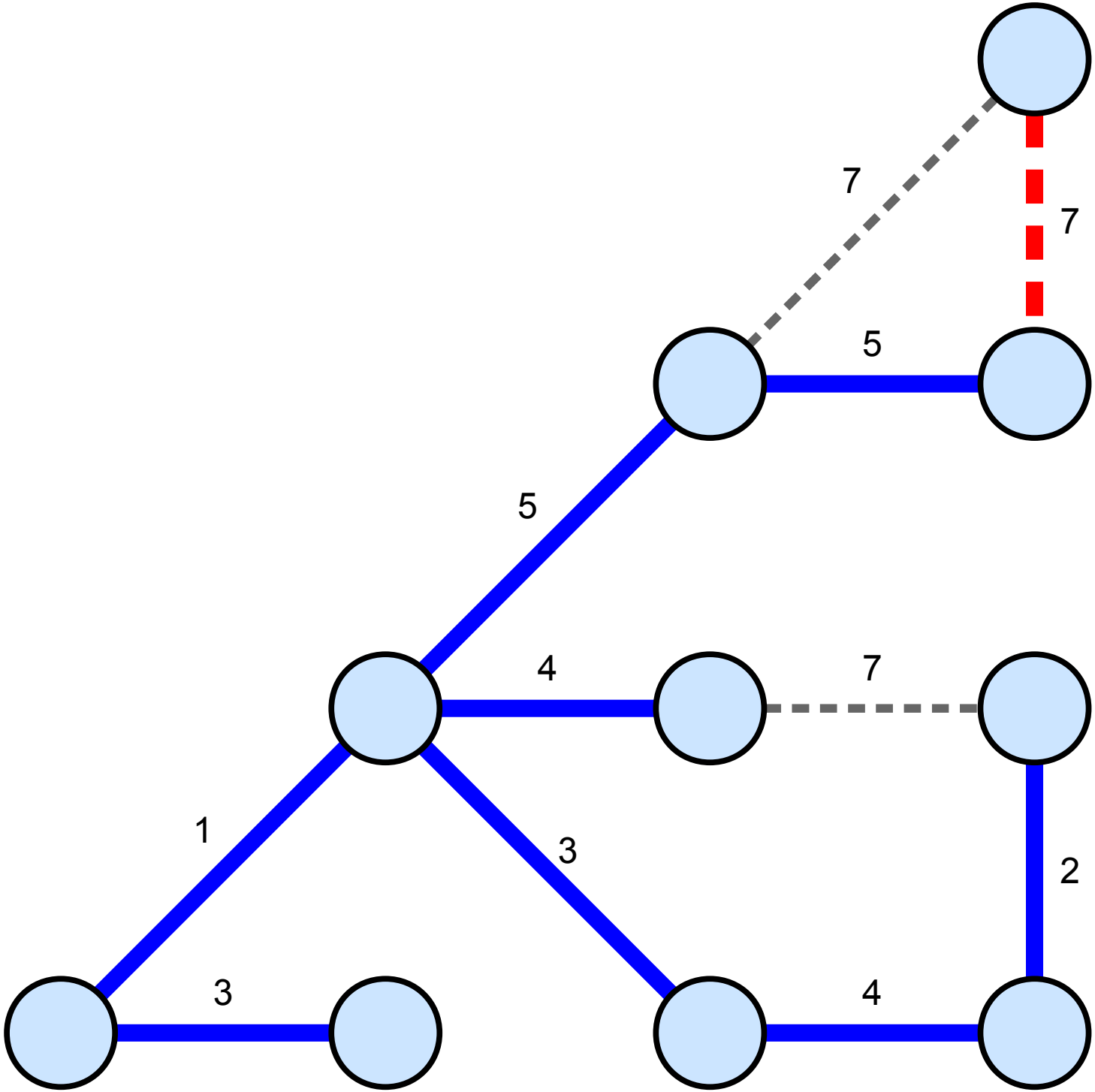

7
7
5
5
6
4
7
1
3
6
2
3
4

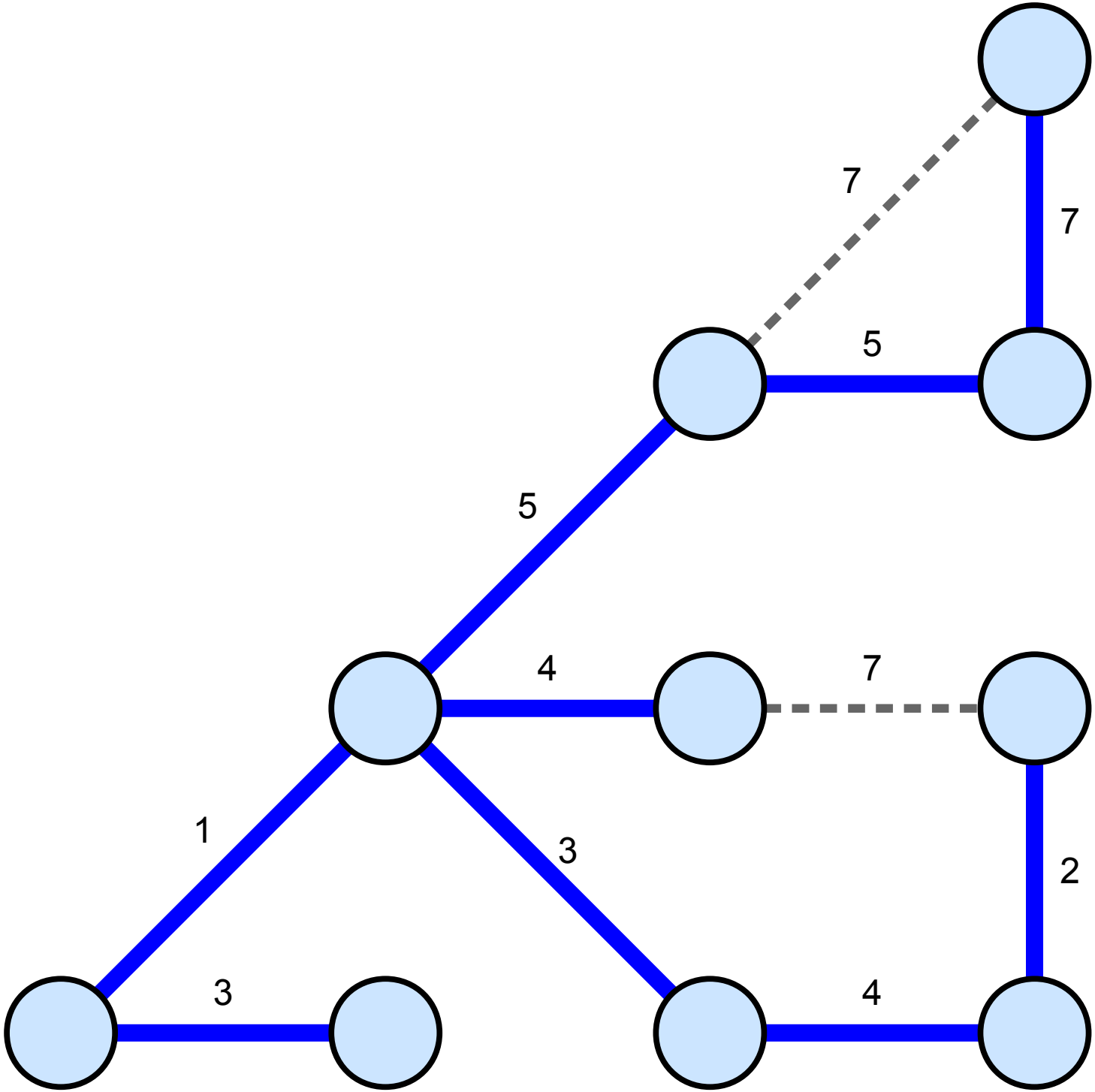

7
7
5
5
6
4
7
1
3
6
2
3
4

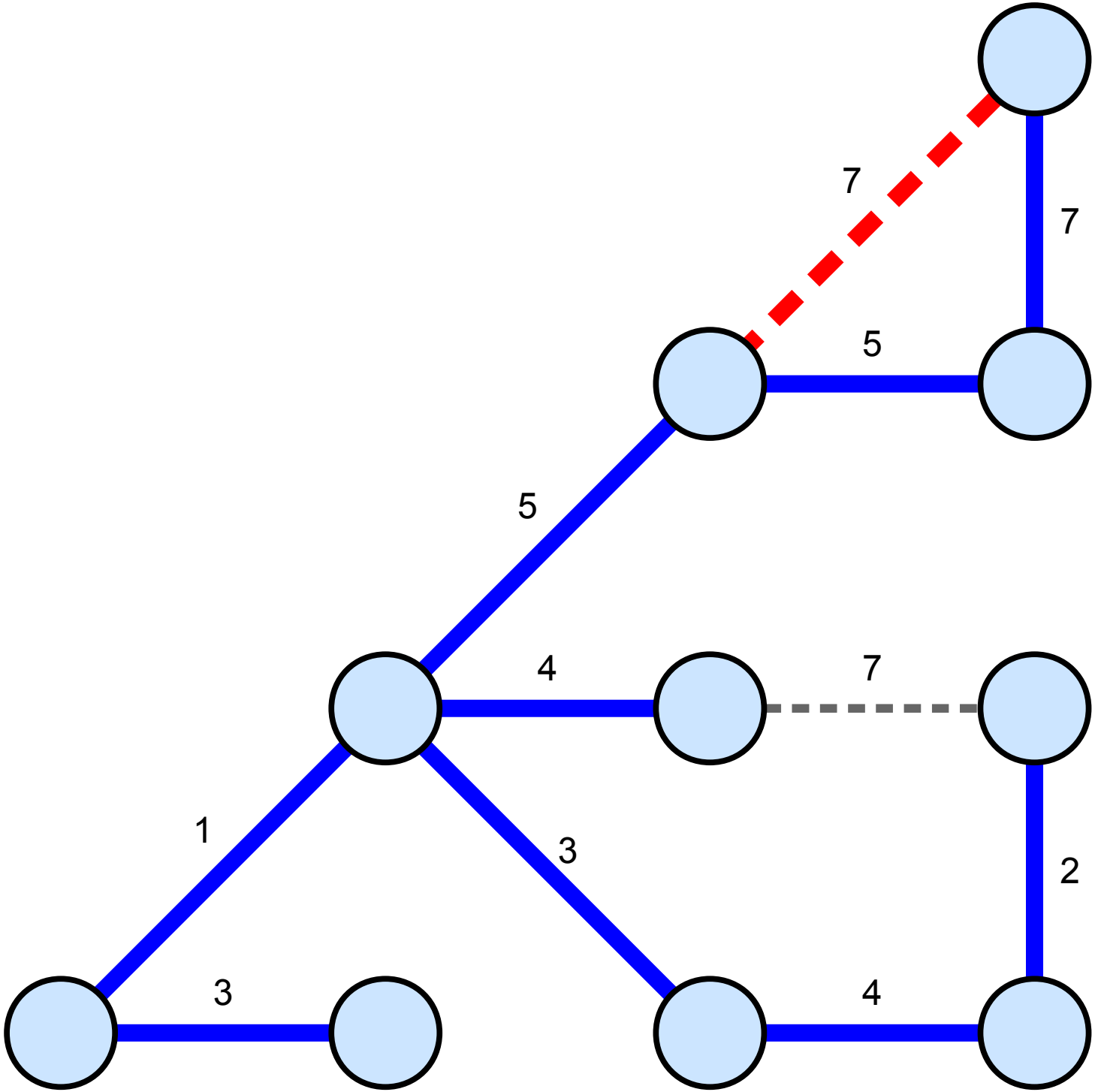

7
7
5
5
6
4
7
1
3
2
3
4

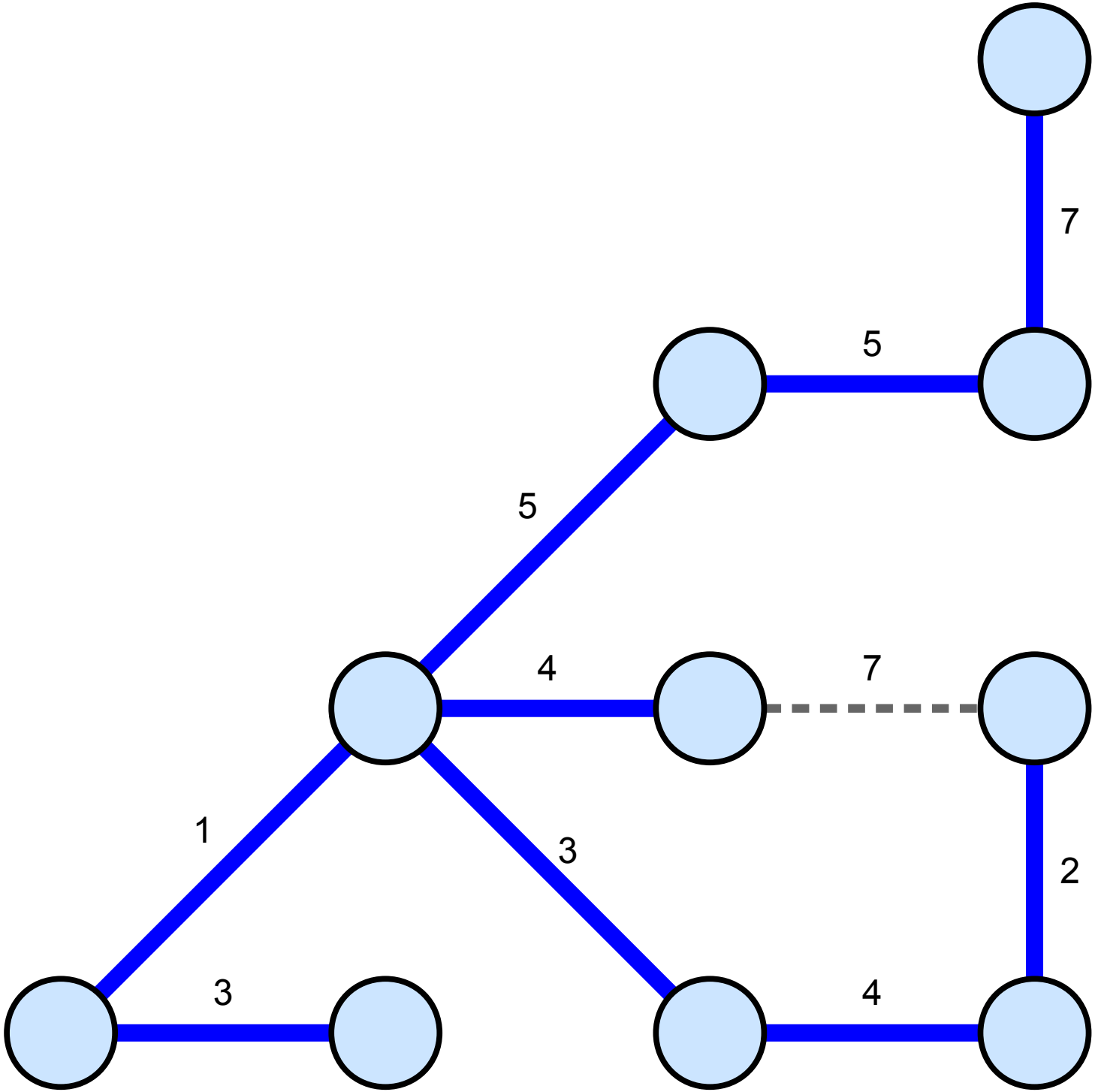

7
7
5
5
6
4
7
1
3
2
3
4

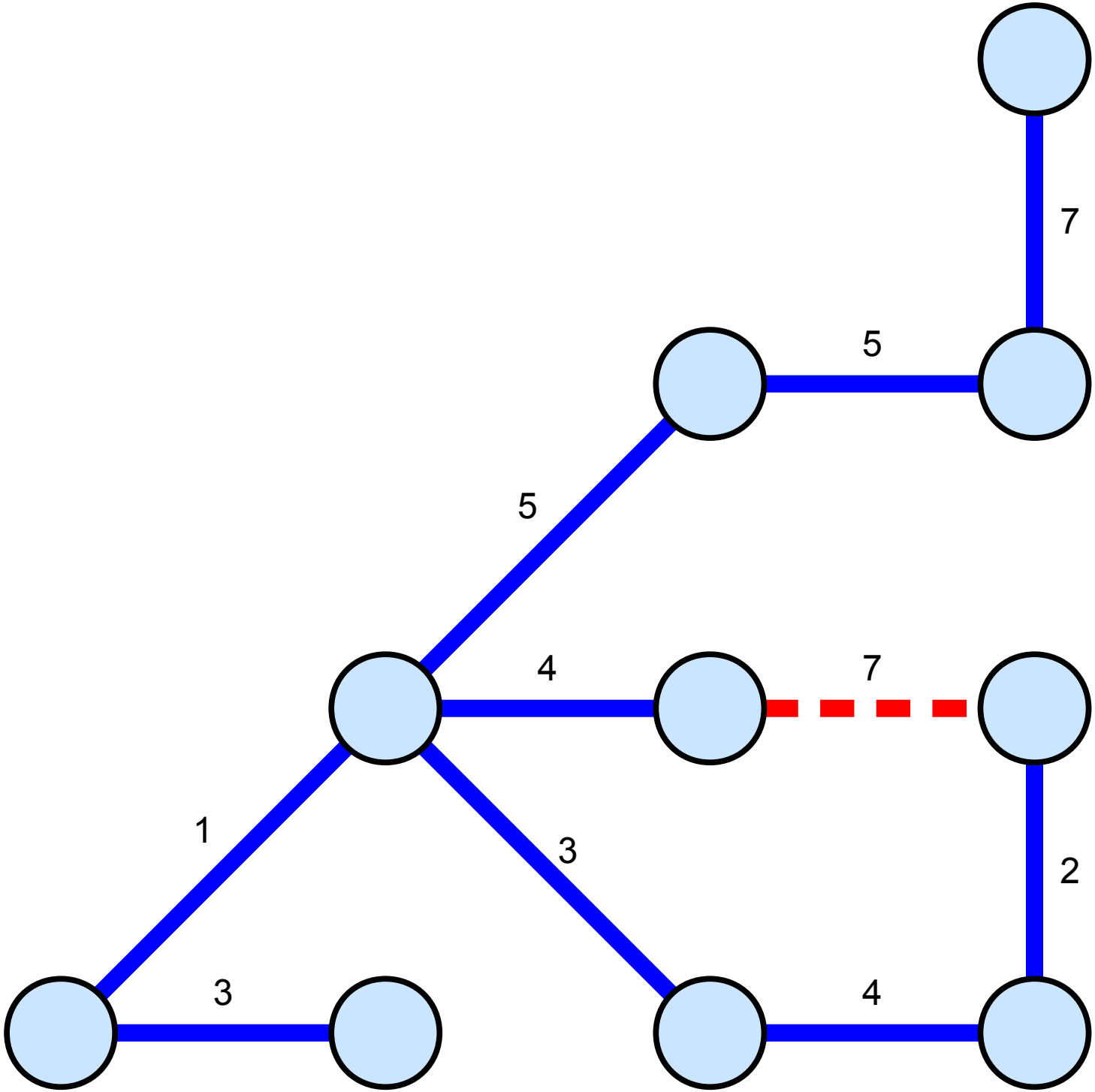

7
7
5
5
4
7
1
3
2
3
4

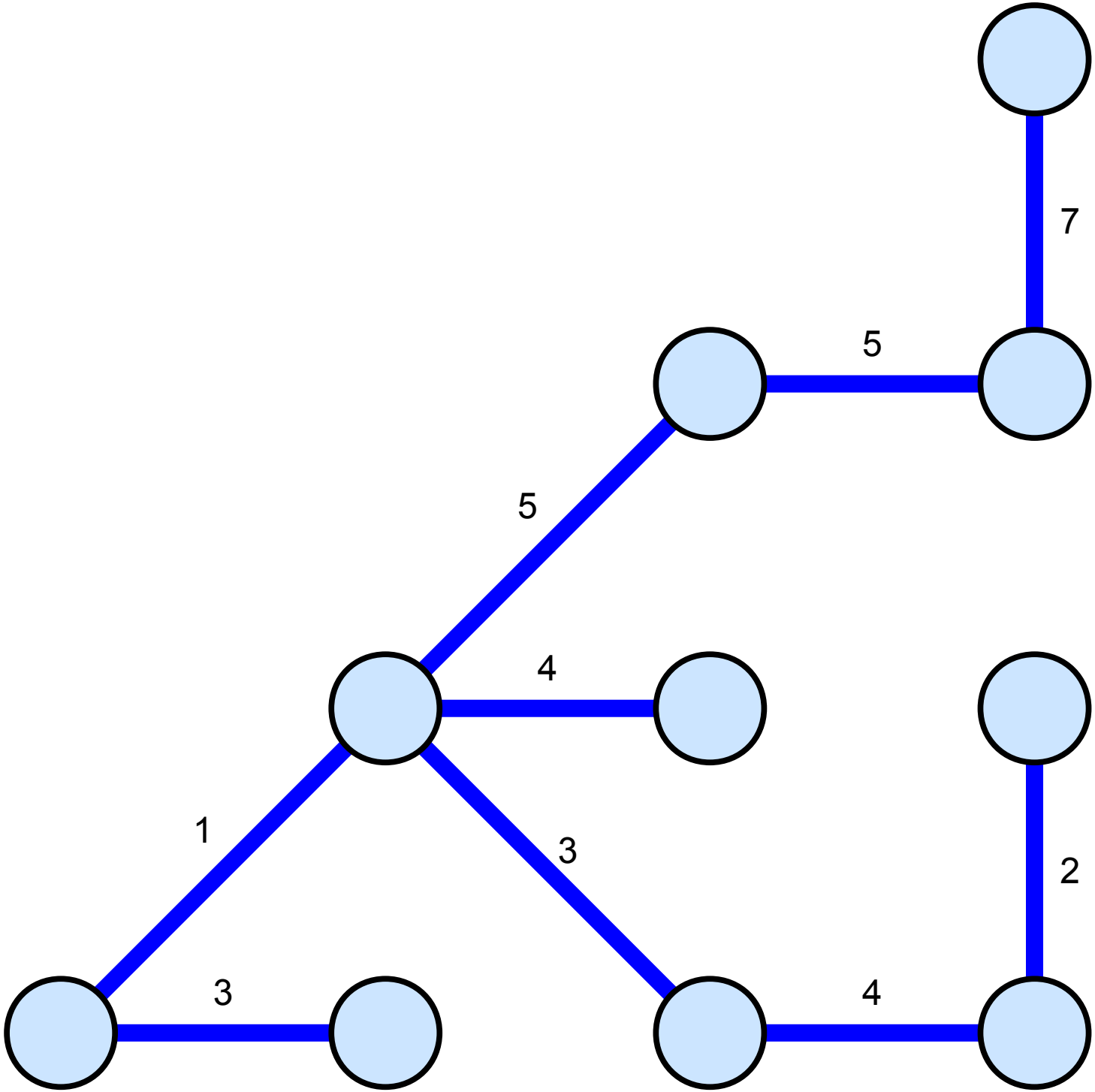

7
7
5
5
4
7
1
3
2
3
4

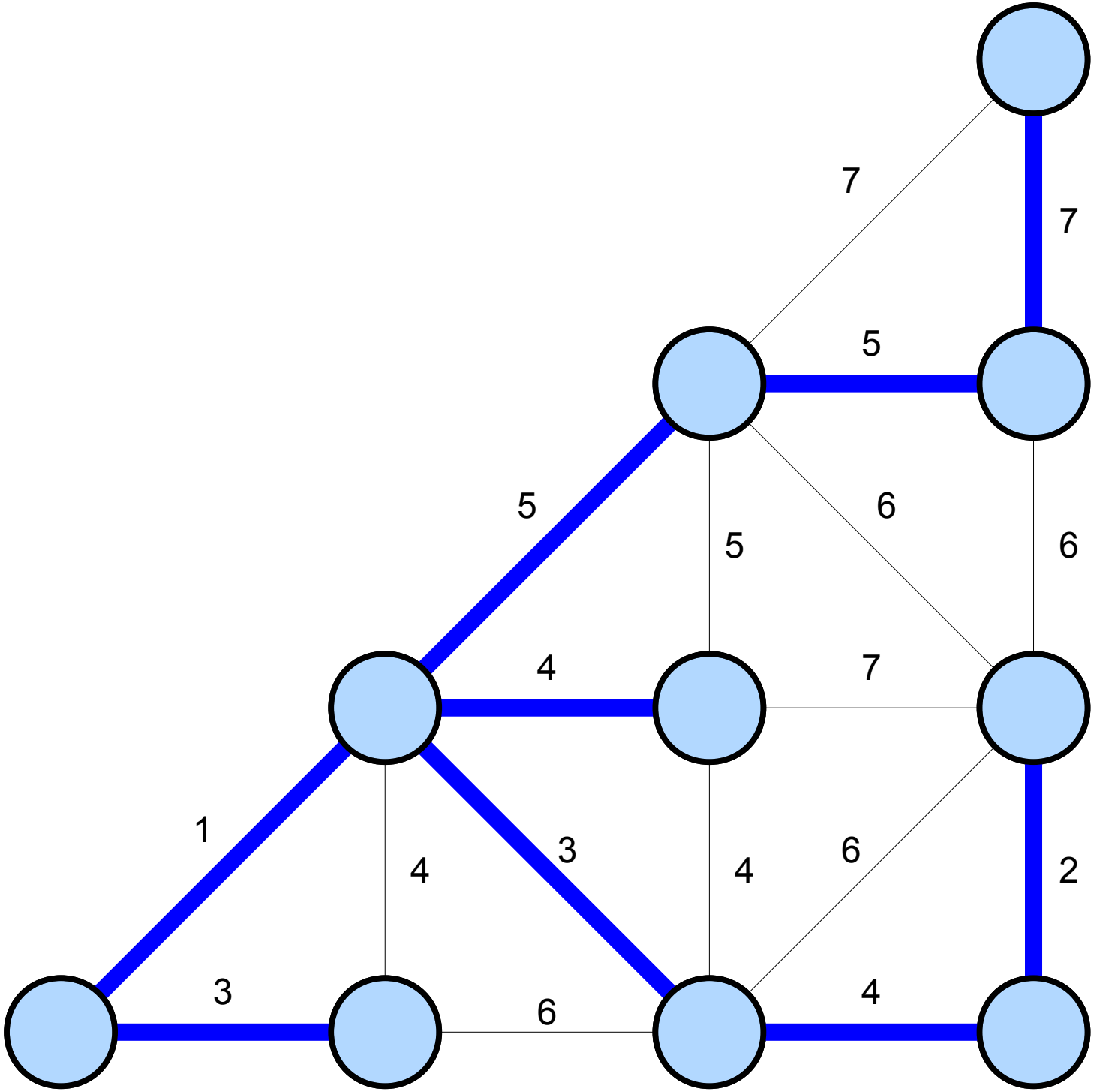

7
7
5
5
4
7
1
3
2
3
4


7
7
5
5
4
7
1
3
2
3
4


7
5
5
4
7
1
3
2
3
4


7
5
5
4
7
1
3
2
3
4

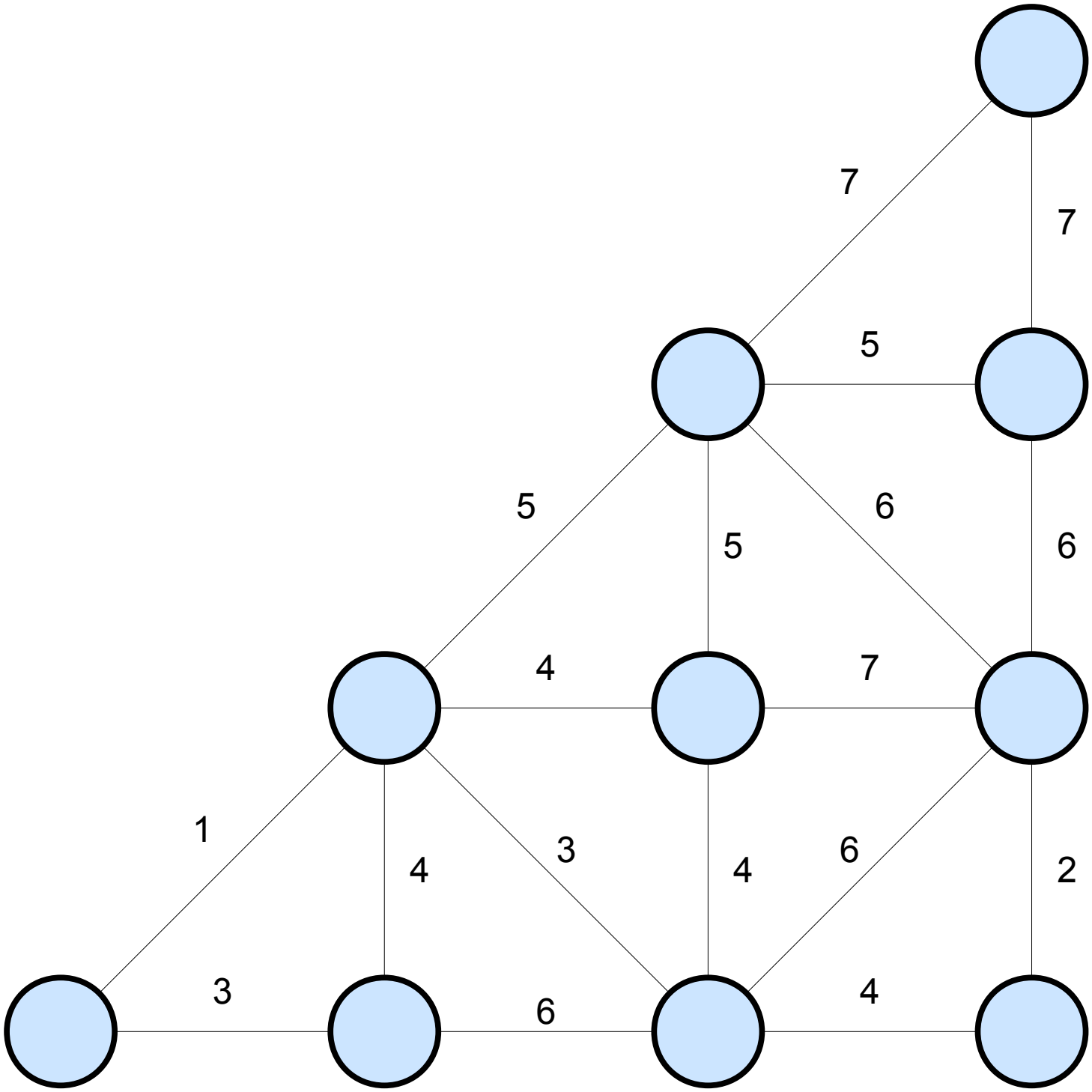

7
5
5
4
1
3
2
3
4

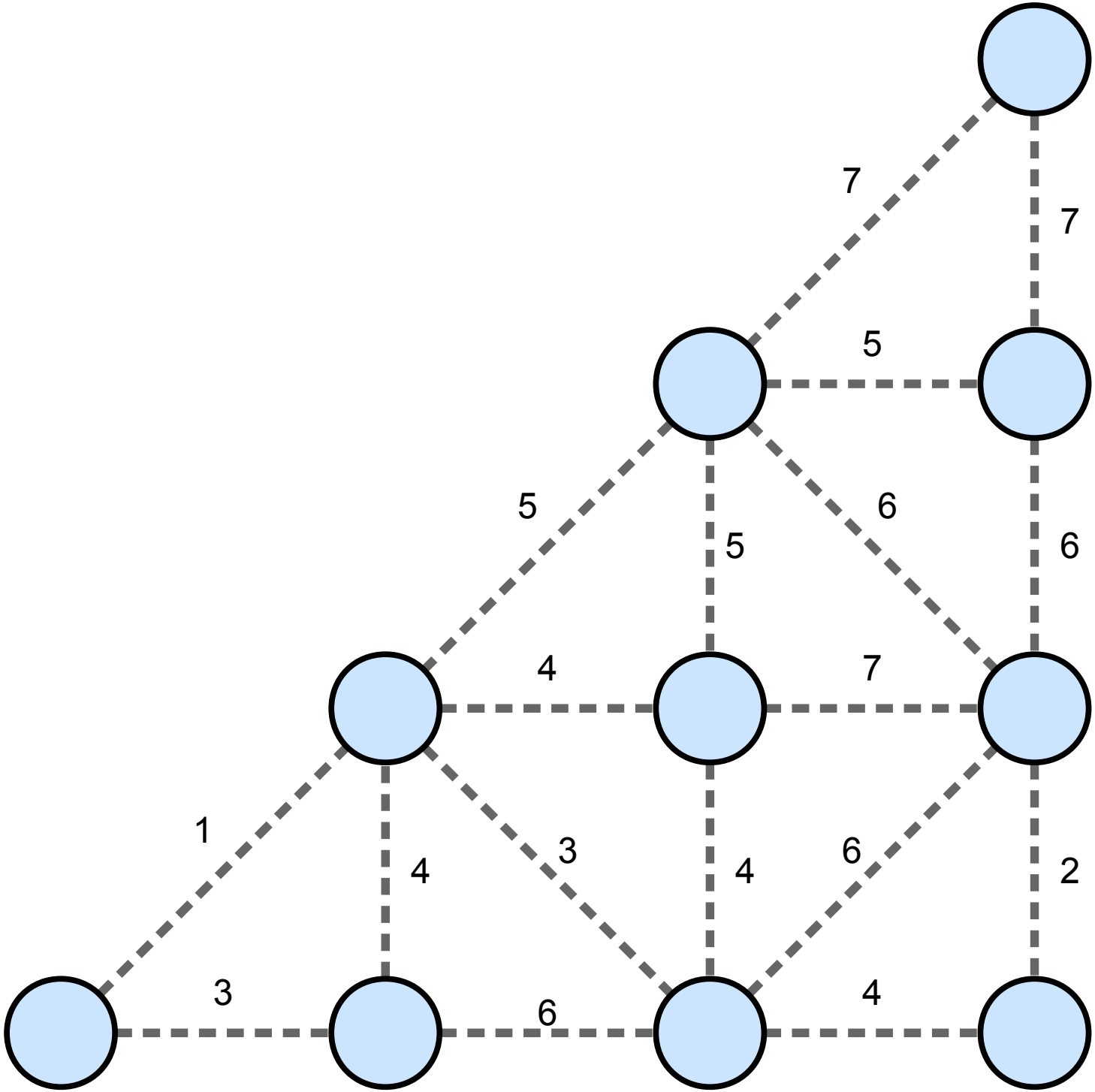

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

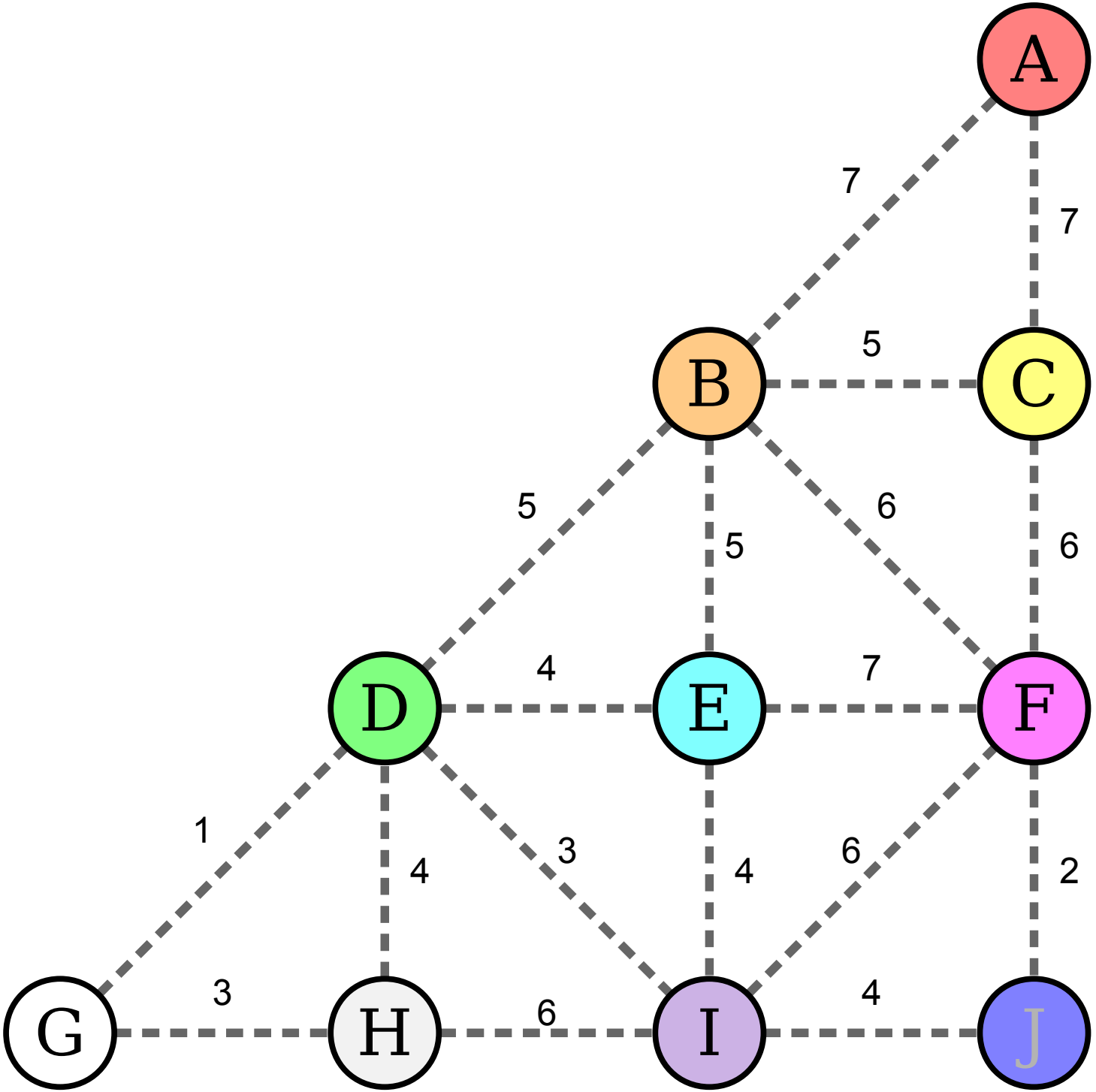
Very simple algorithm, tricky proof that it produces a Minimal Spanning Tree

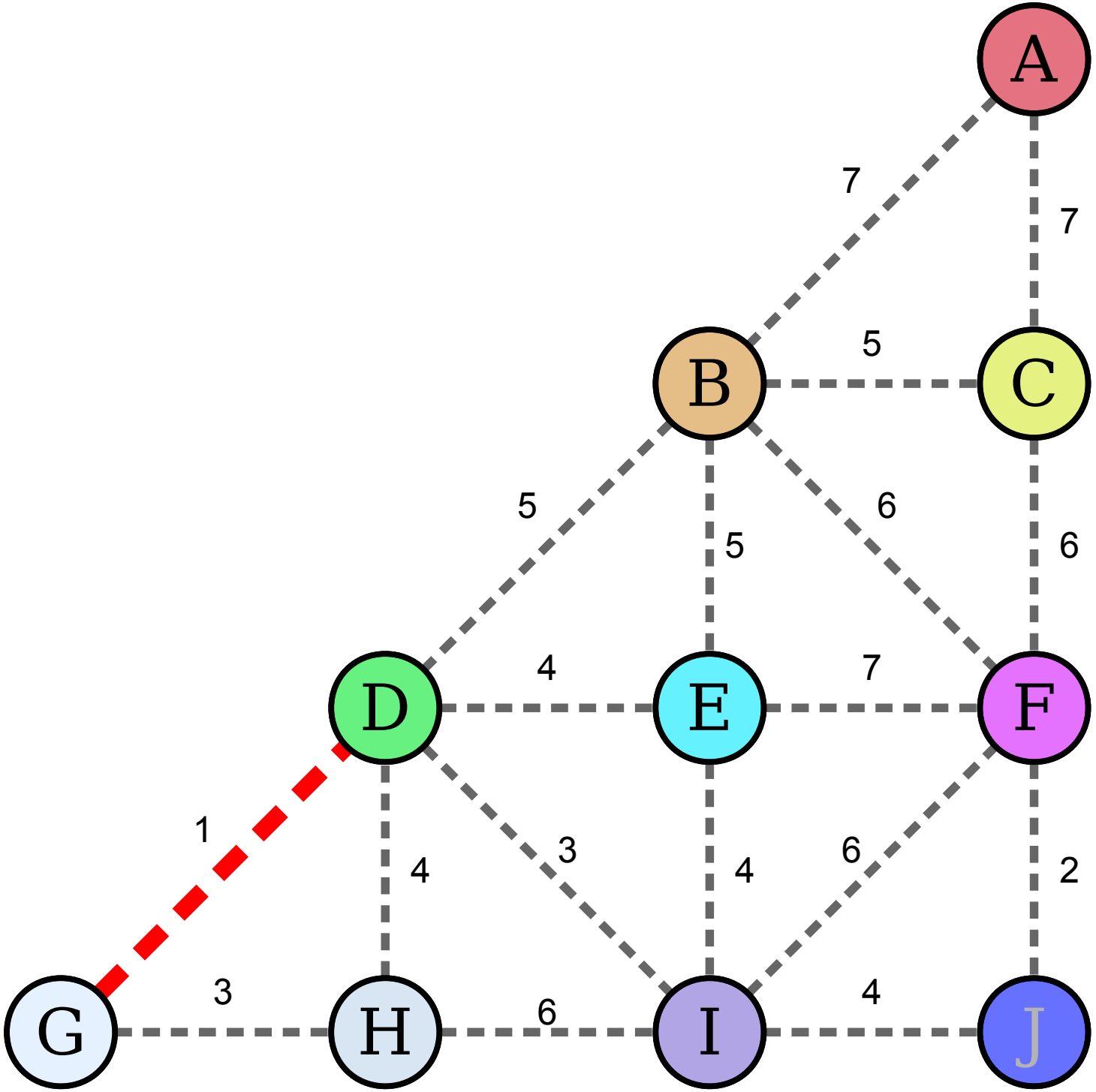
Maintaining Connectivity

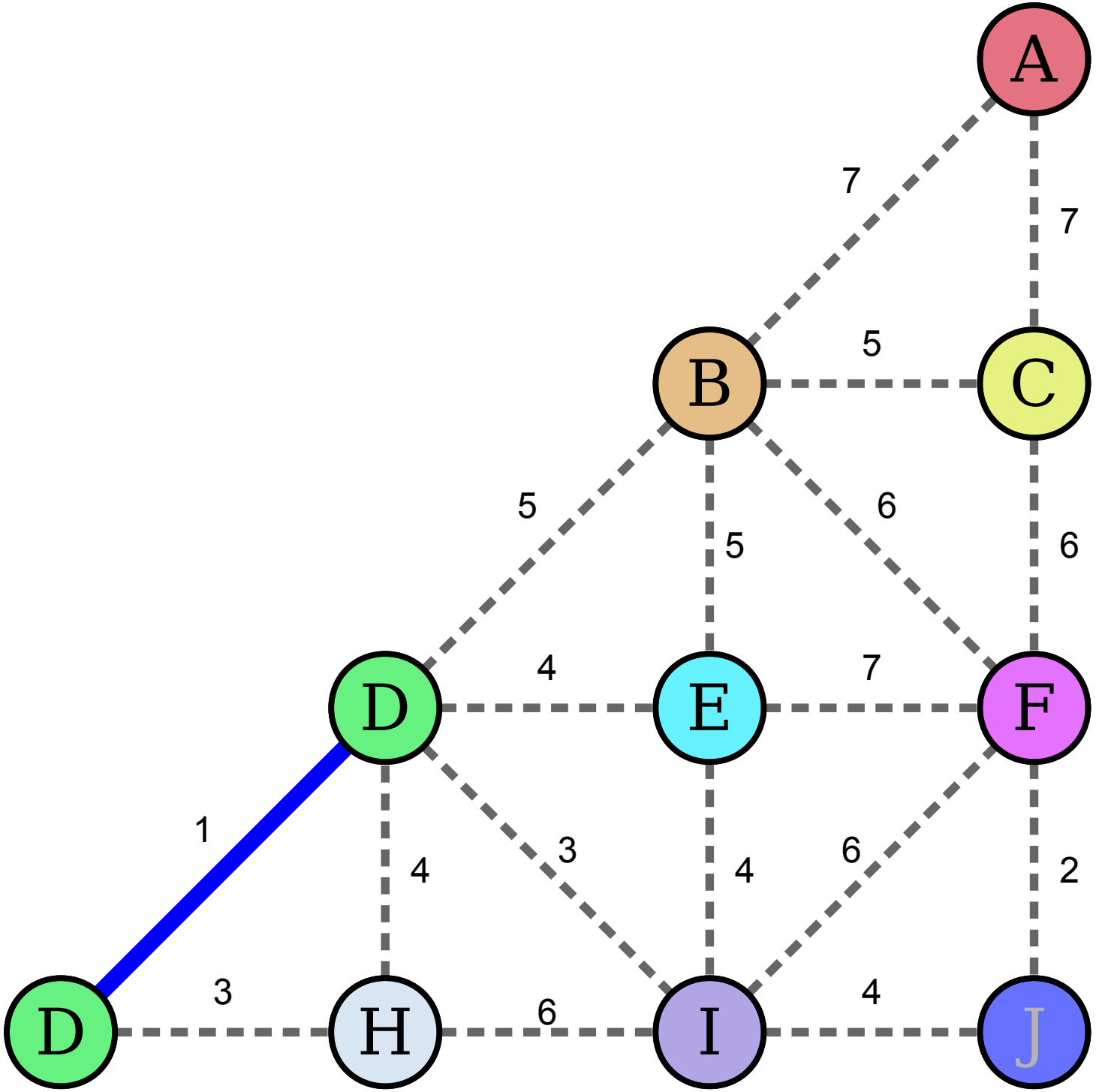
- The key step in Kruskal's algorithm is determining whether the two endpoints of an edge are already connected to one another.
- Typical approach: break the nodes apart into **clusters**.
  - Initially, each node is in its own cluster.
  - Whenever an edge is added, the clusters for the endpoints are merged together into a new cluster.

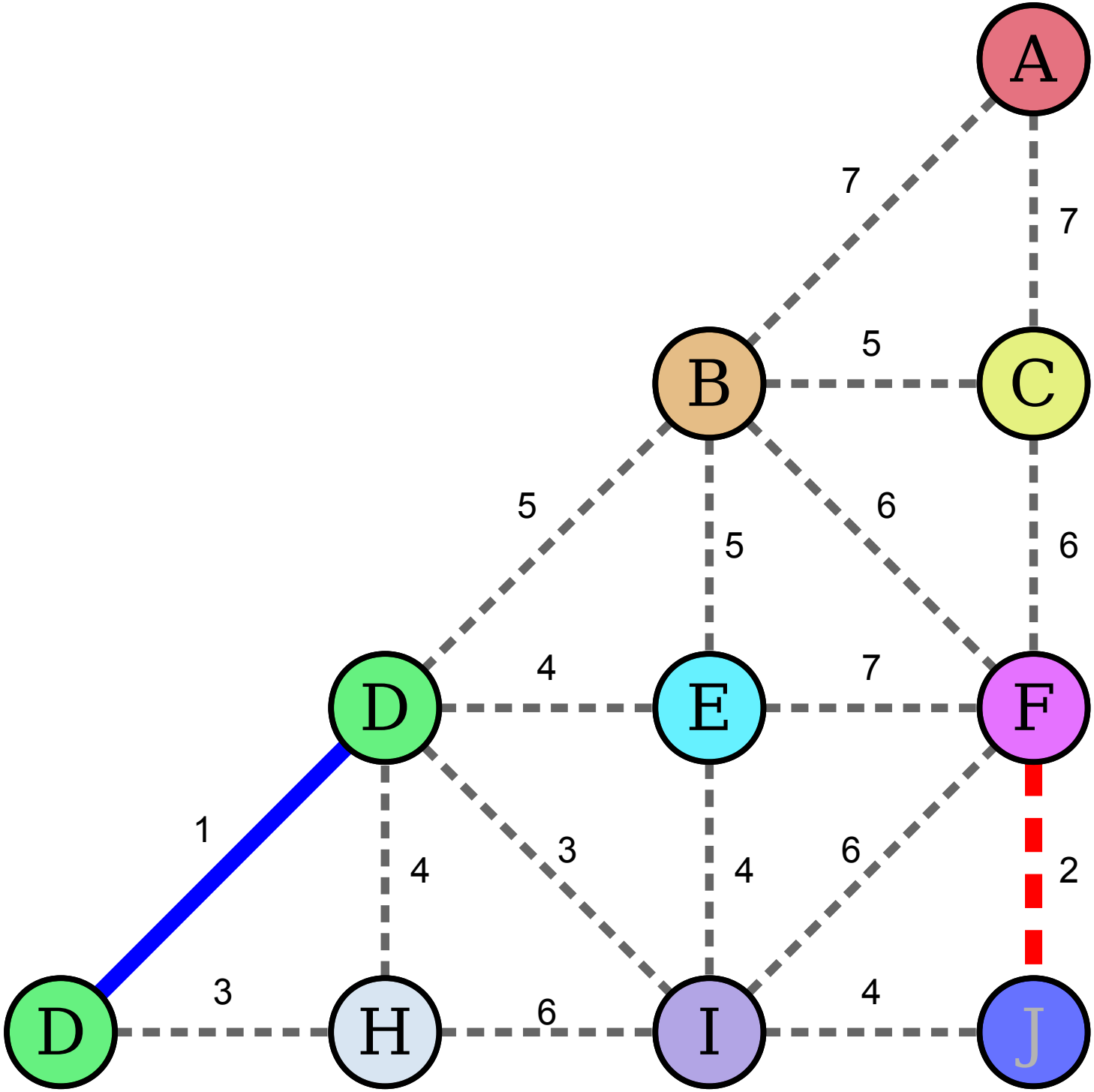

7
7
5
5
6
5
6
4
7
1
4
3
4
6
2
3
6
4

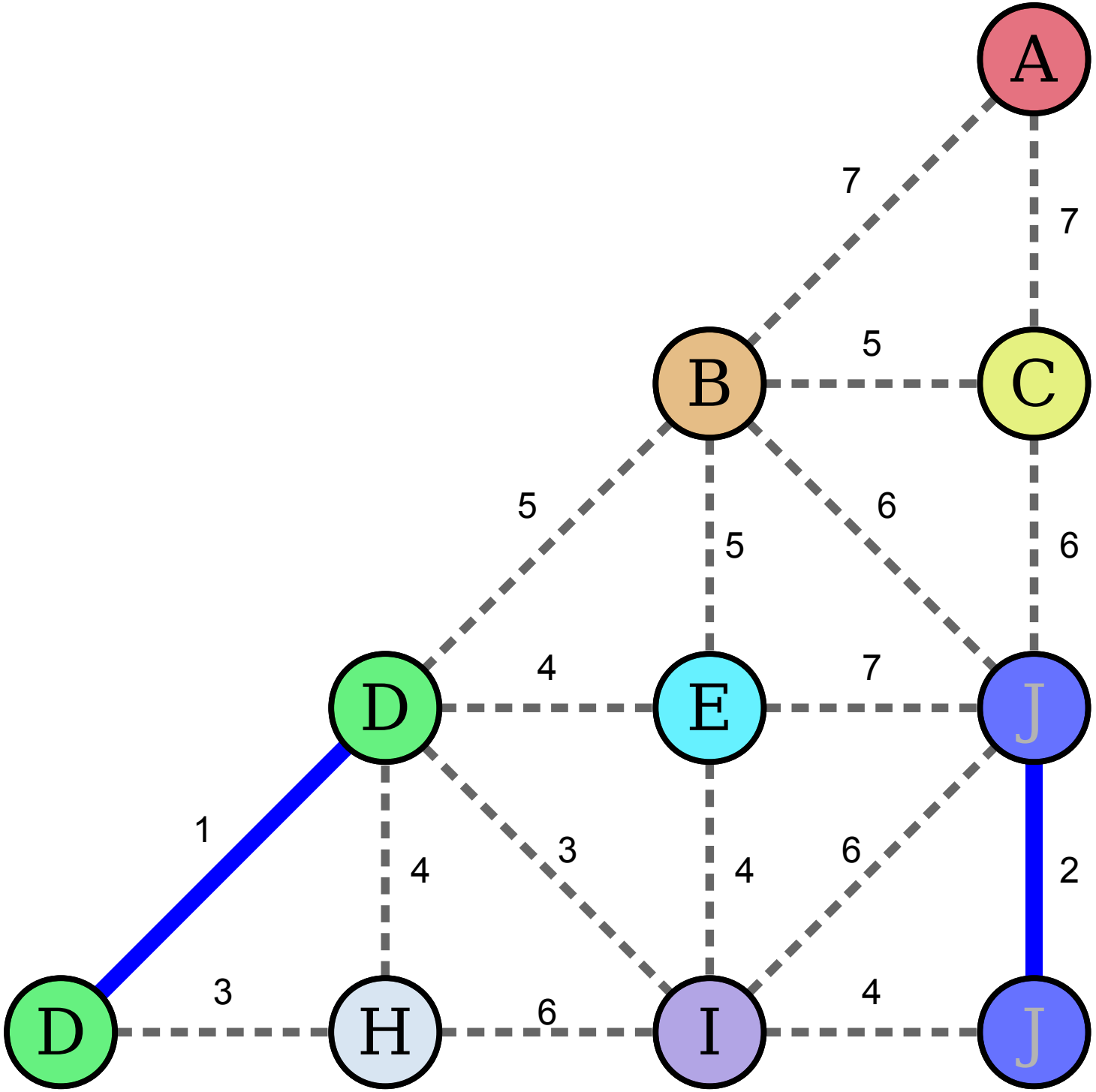

7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

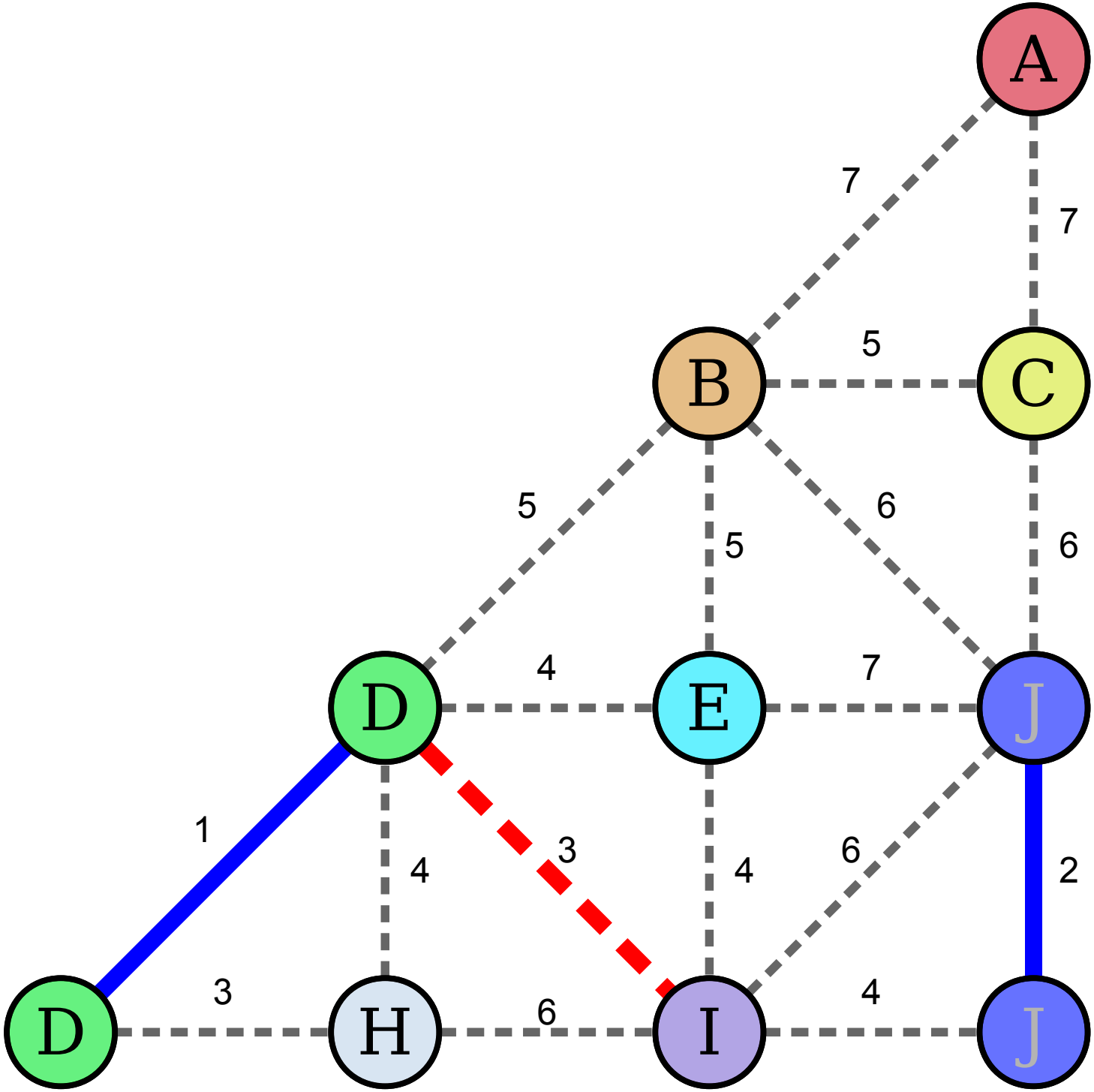

A
B
C
D
E
F
G
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

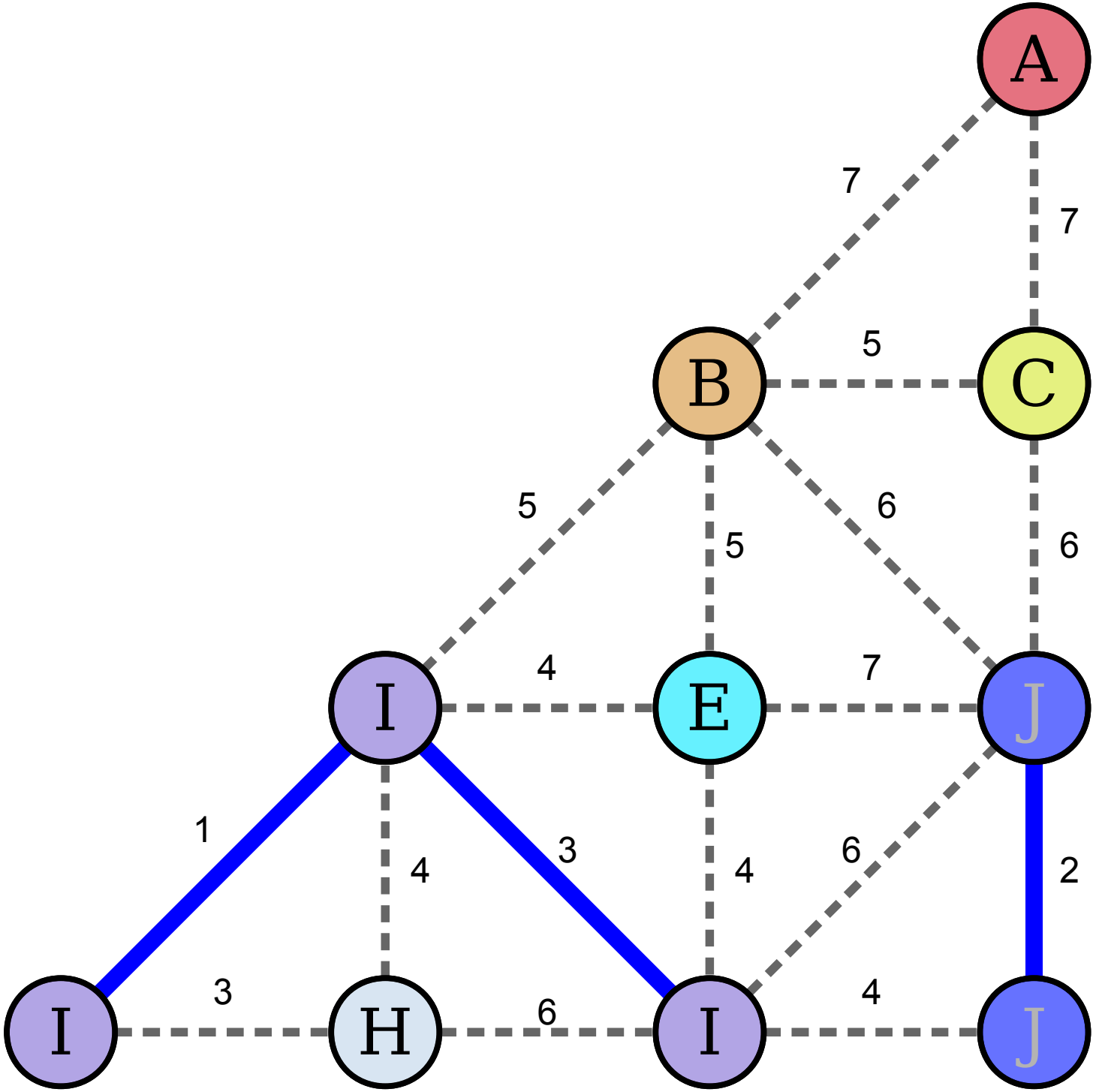

A
B
C
D
E
F
G
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

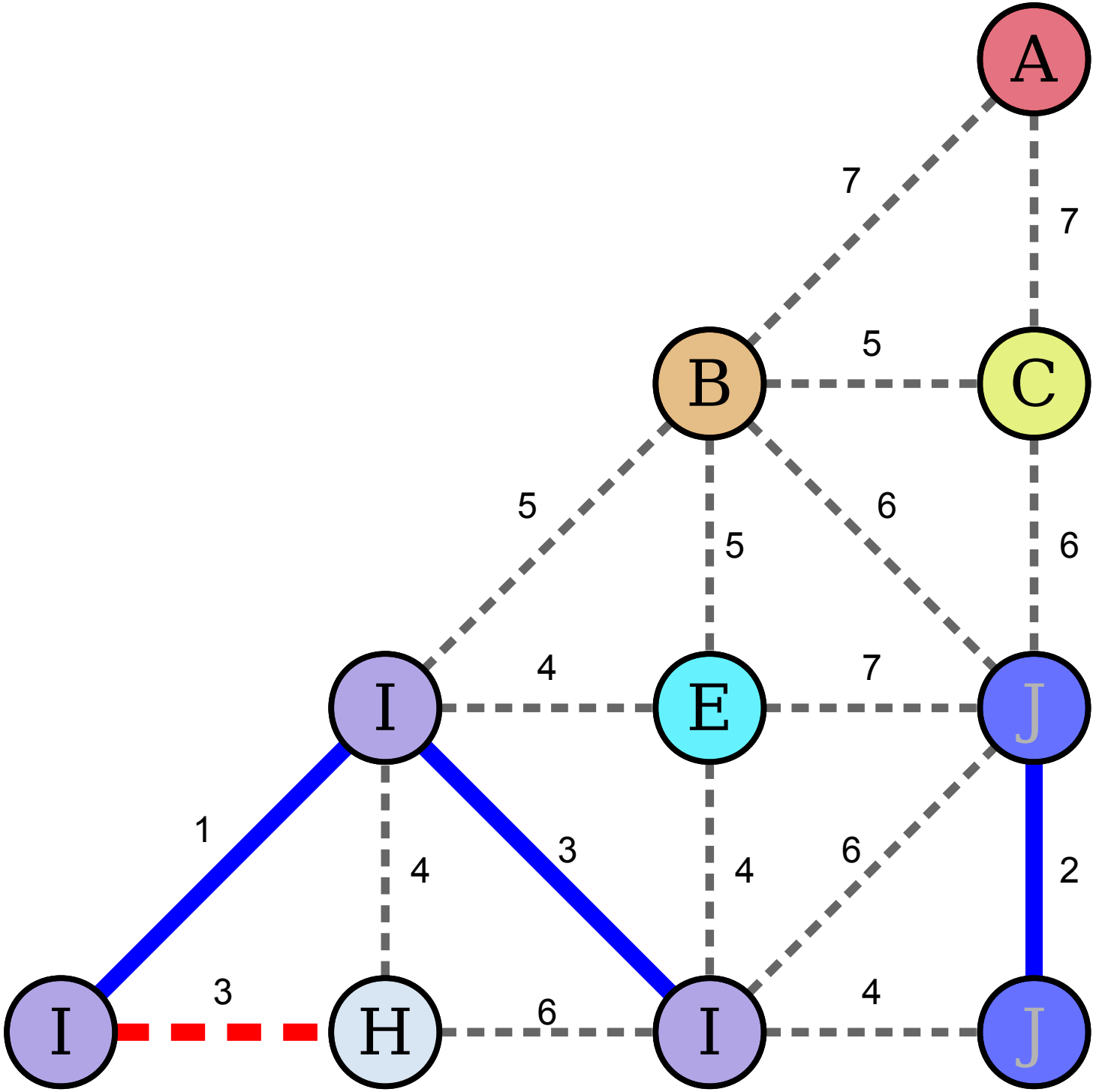

A
B
C
D
E
F
D
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

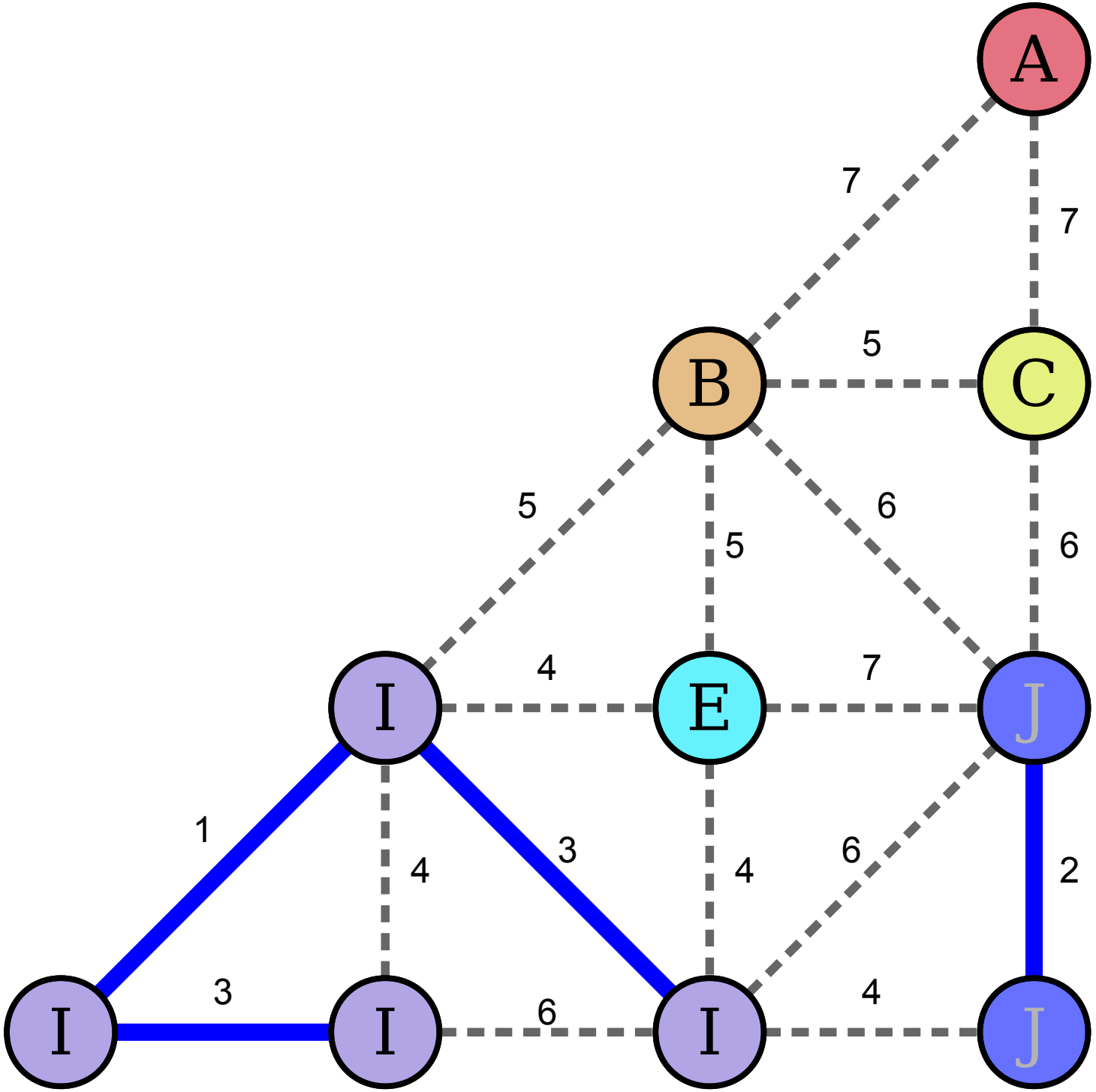

A
B
C
D
E
F
D
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

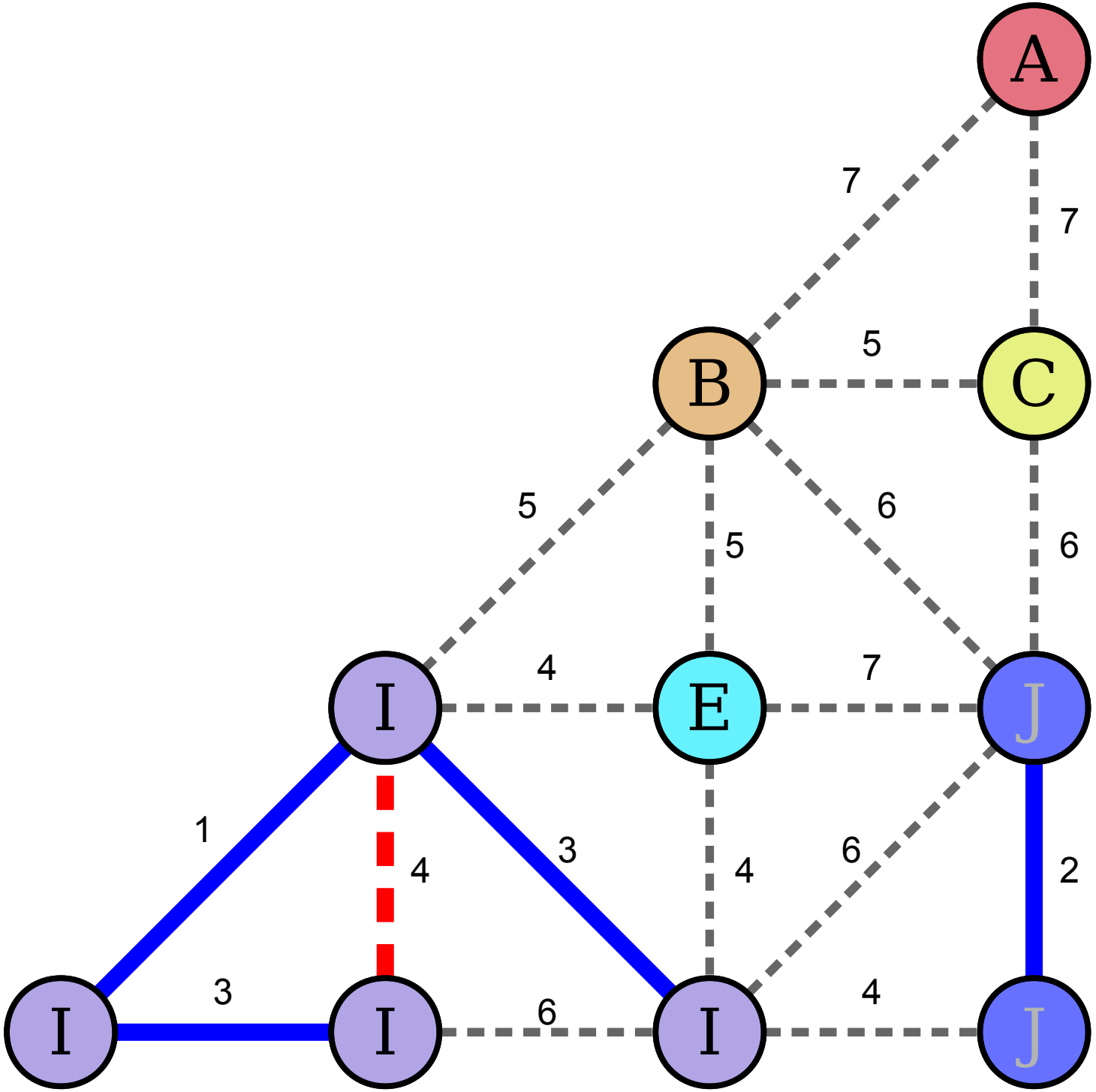

A
B
C
D
E
J
D
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

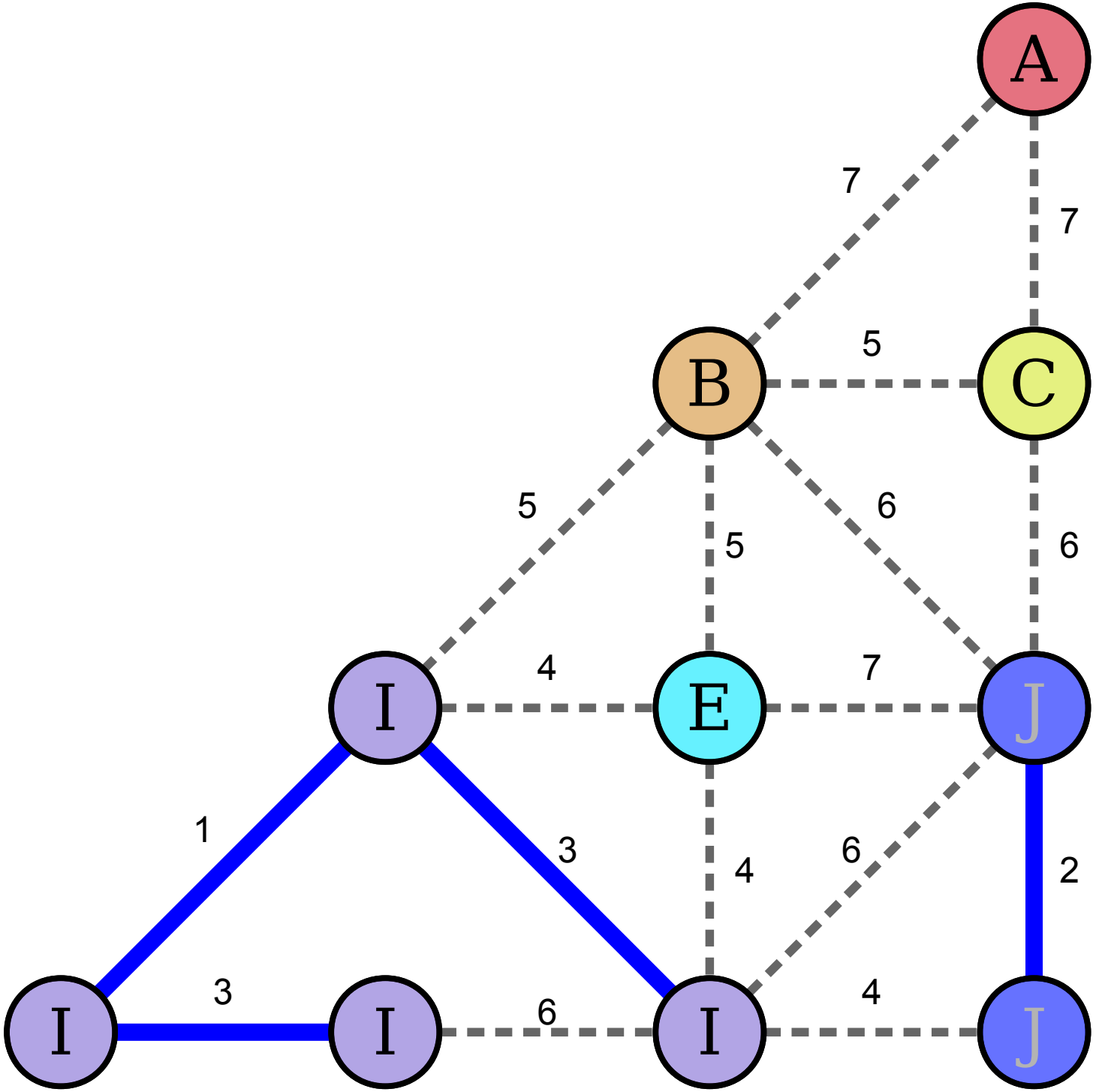

A
B
C
D
E
J
D
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

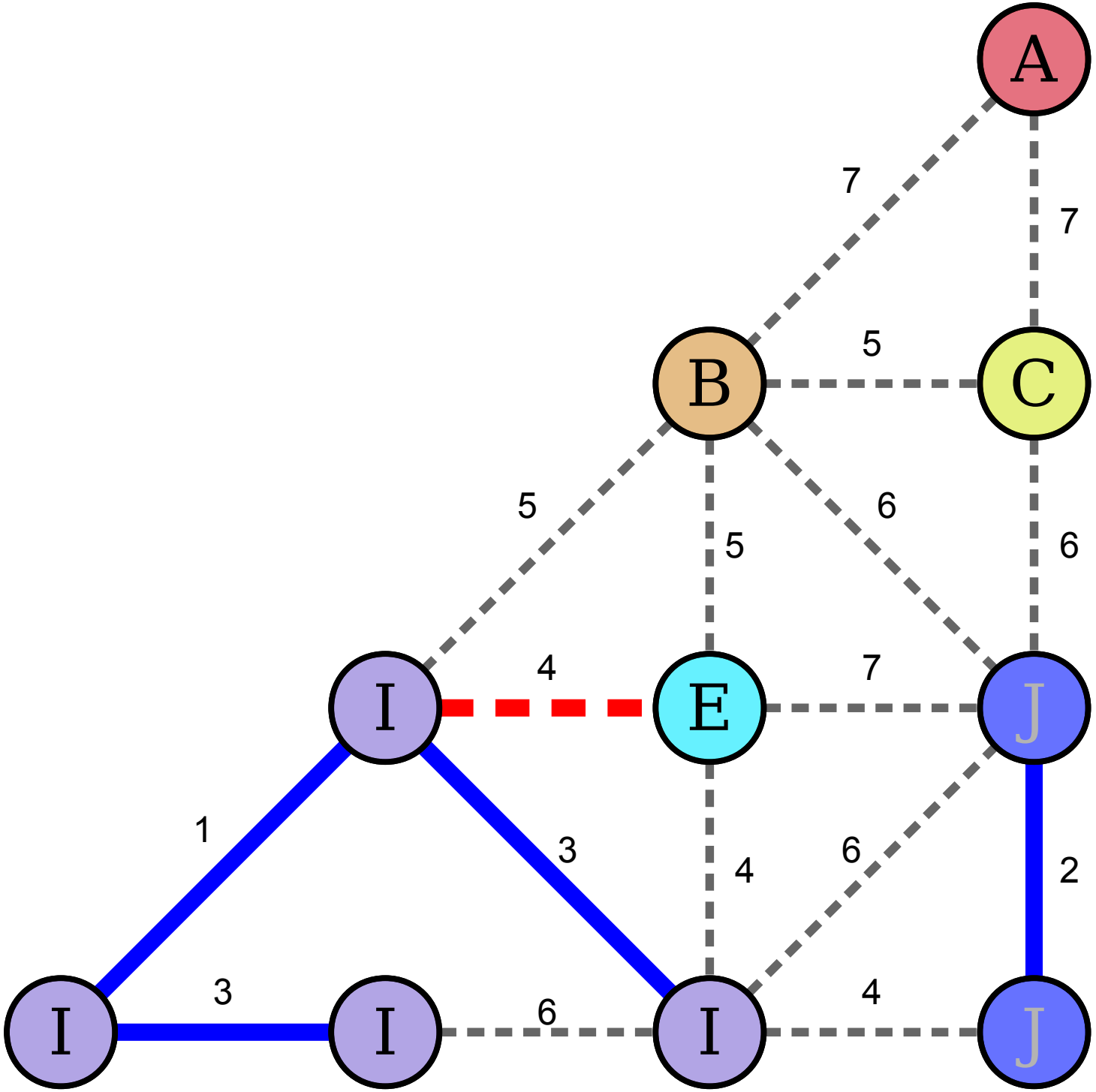

A
B
C
I
E
J
I
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

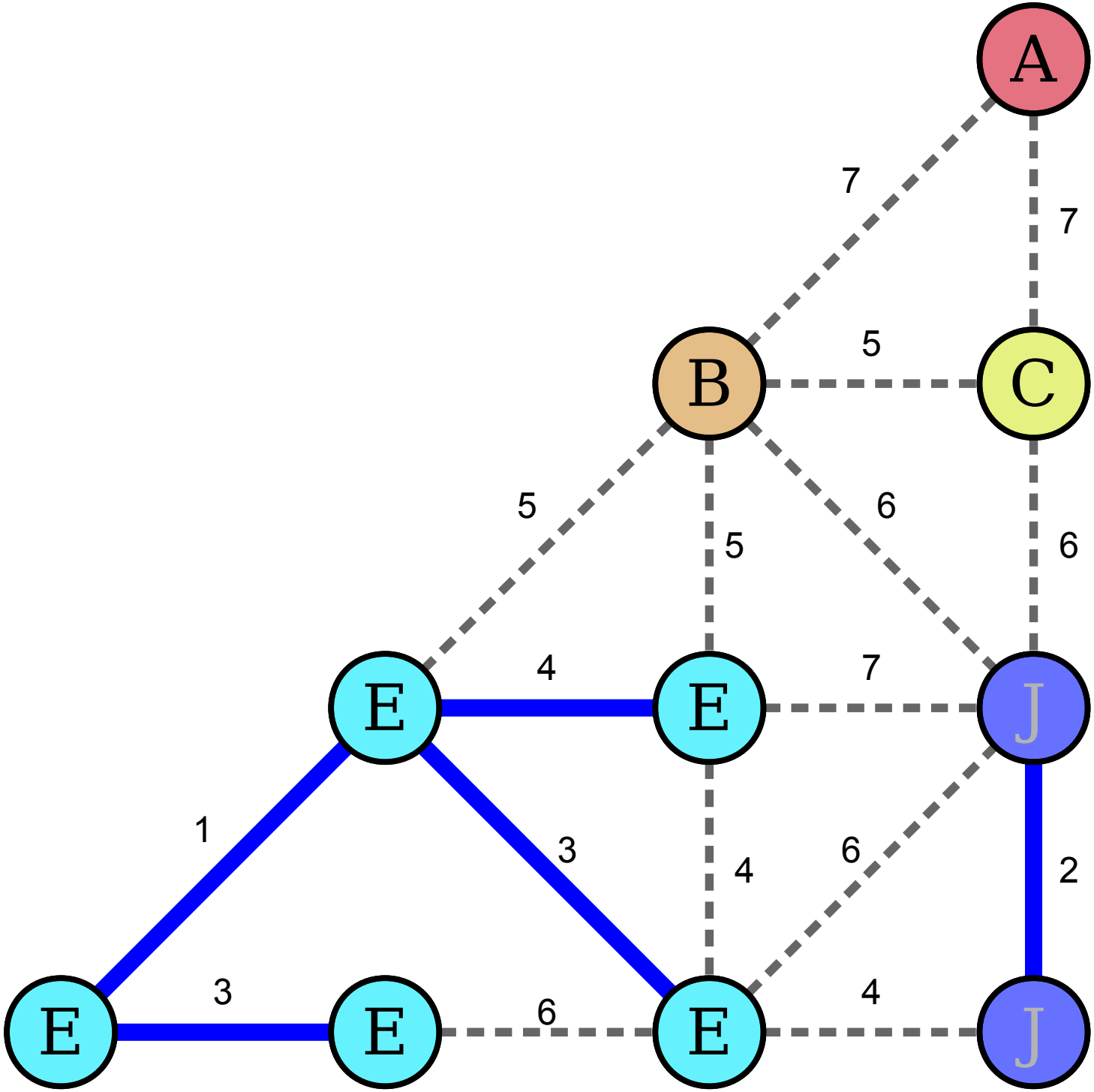

A
B
C
I
E
J
I
H
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

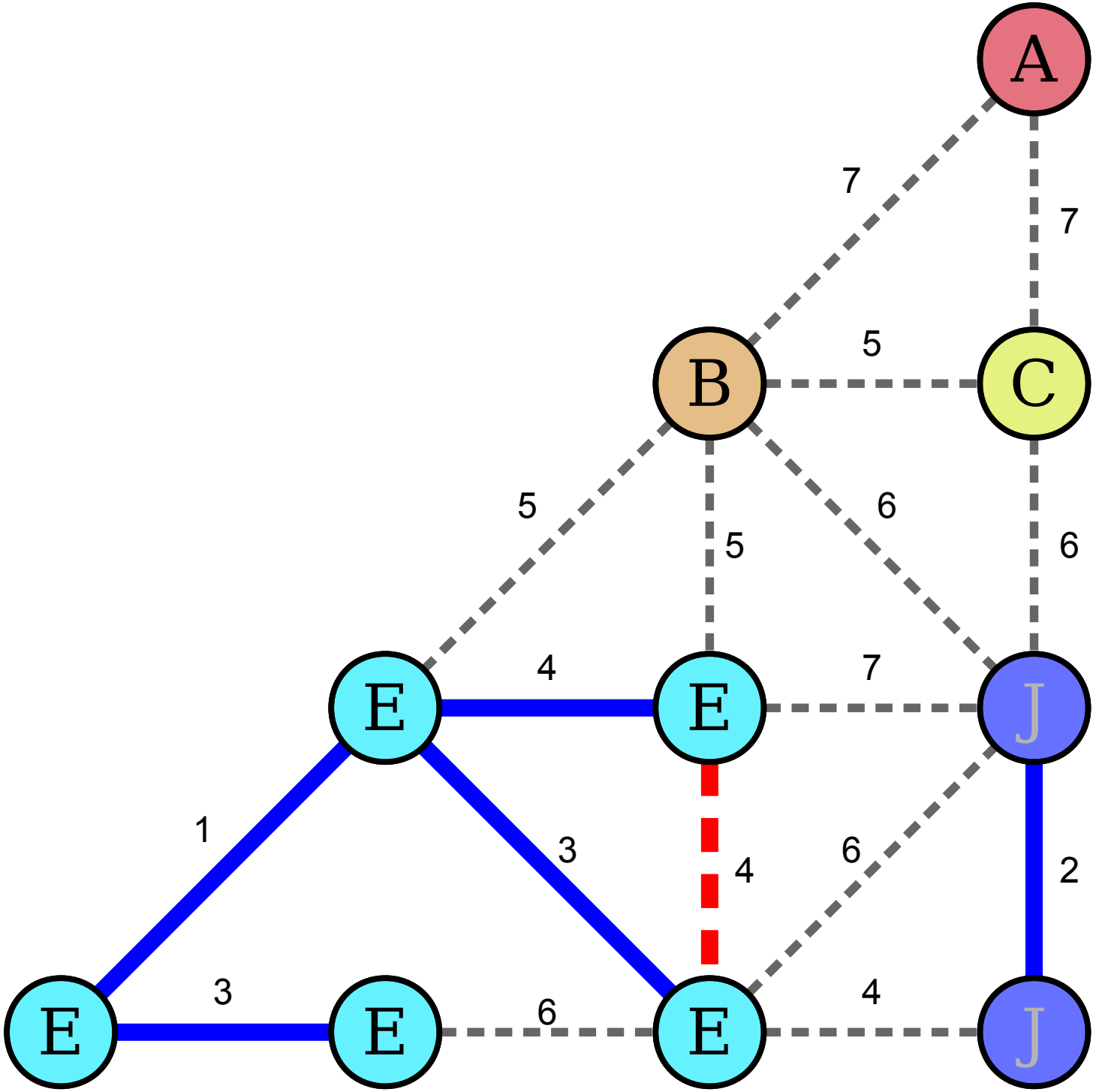

A
B
C
I
E
J
I
I
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

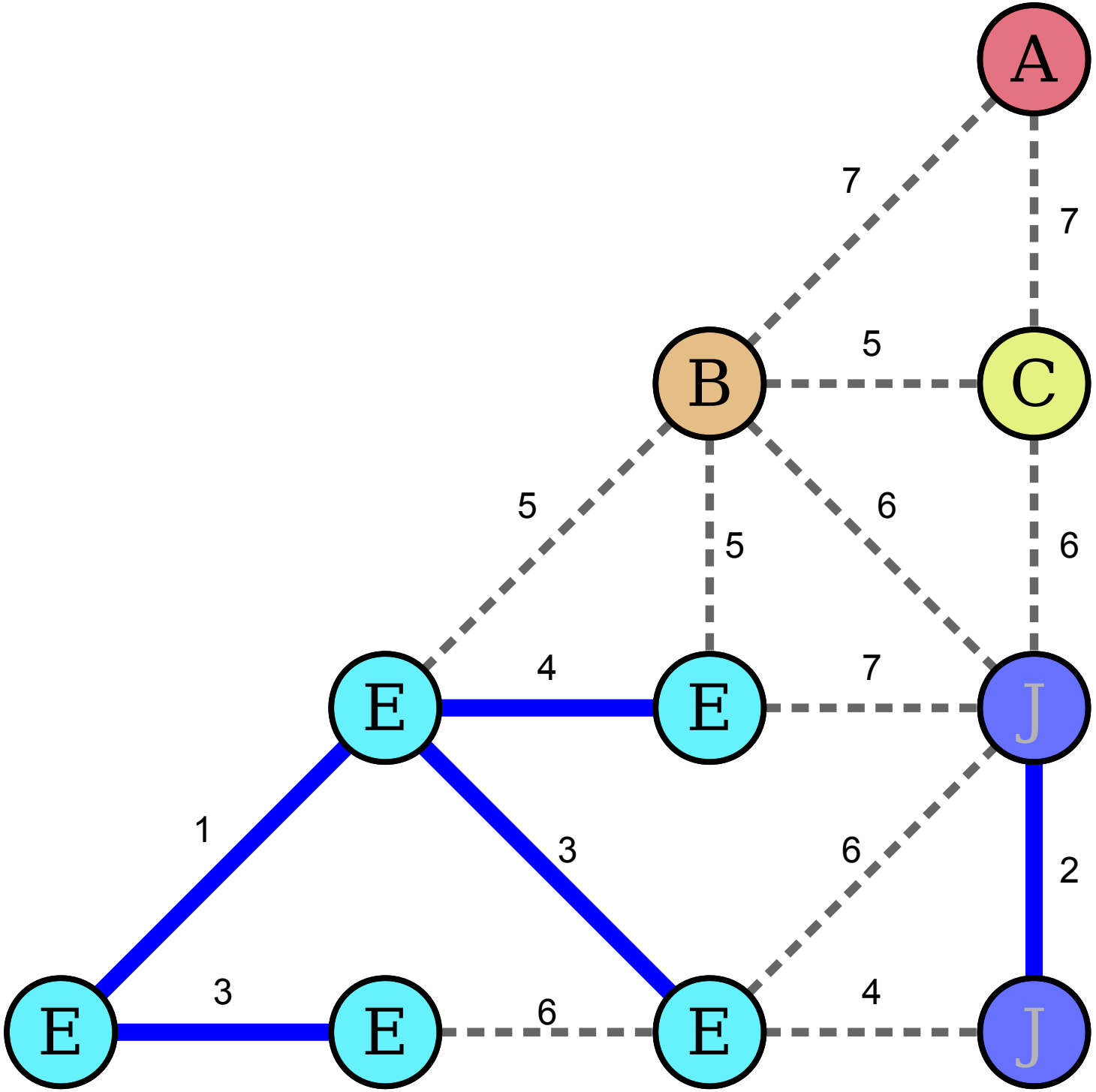

A
B
C
I
E
J
I
I
I
J
7
7
5
5
5
6
6
4
7
1
4
3
4
6
2
3
6
4

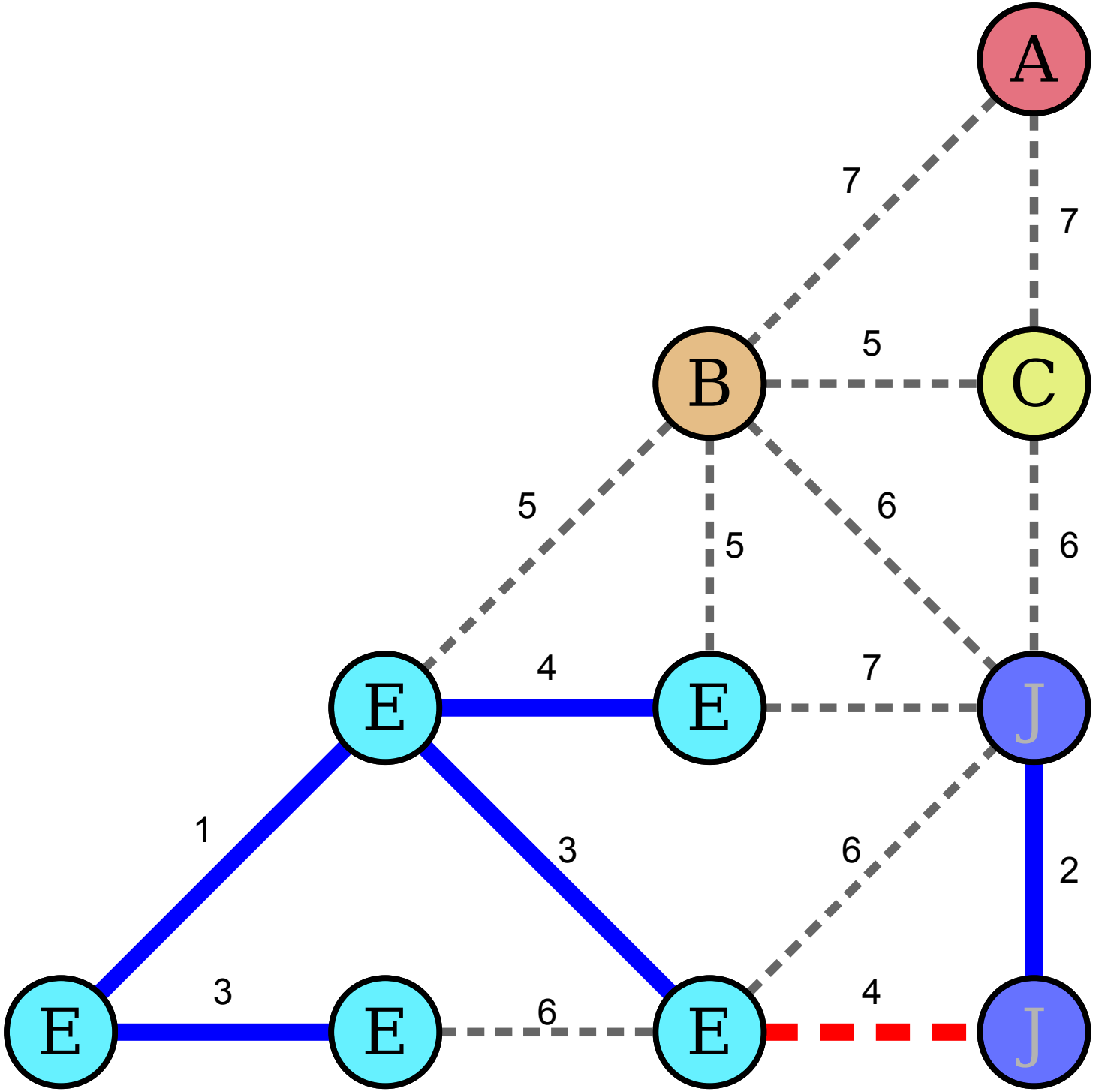

A
B
C
I
E
J
I
I
I
J
7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

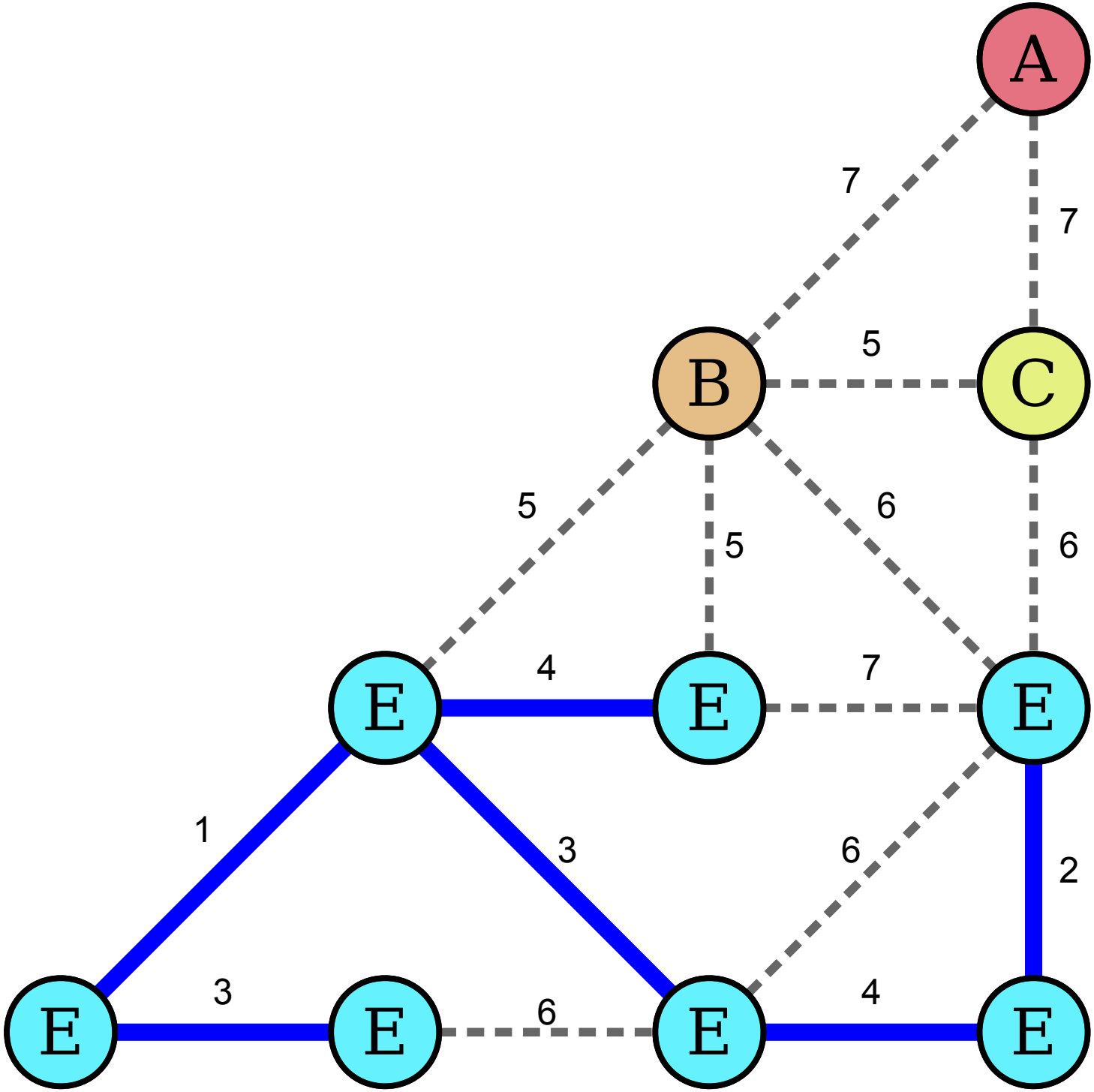

A
B
C
I
E
J
I
I
I
J
7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

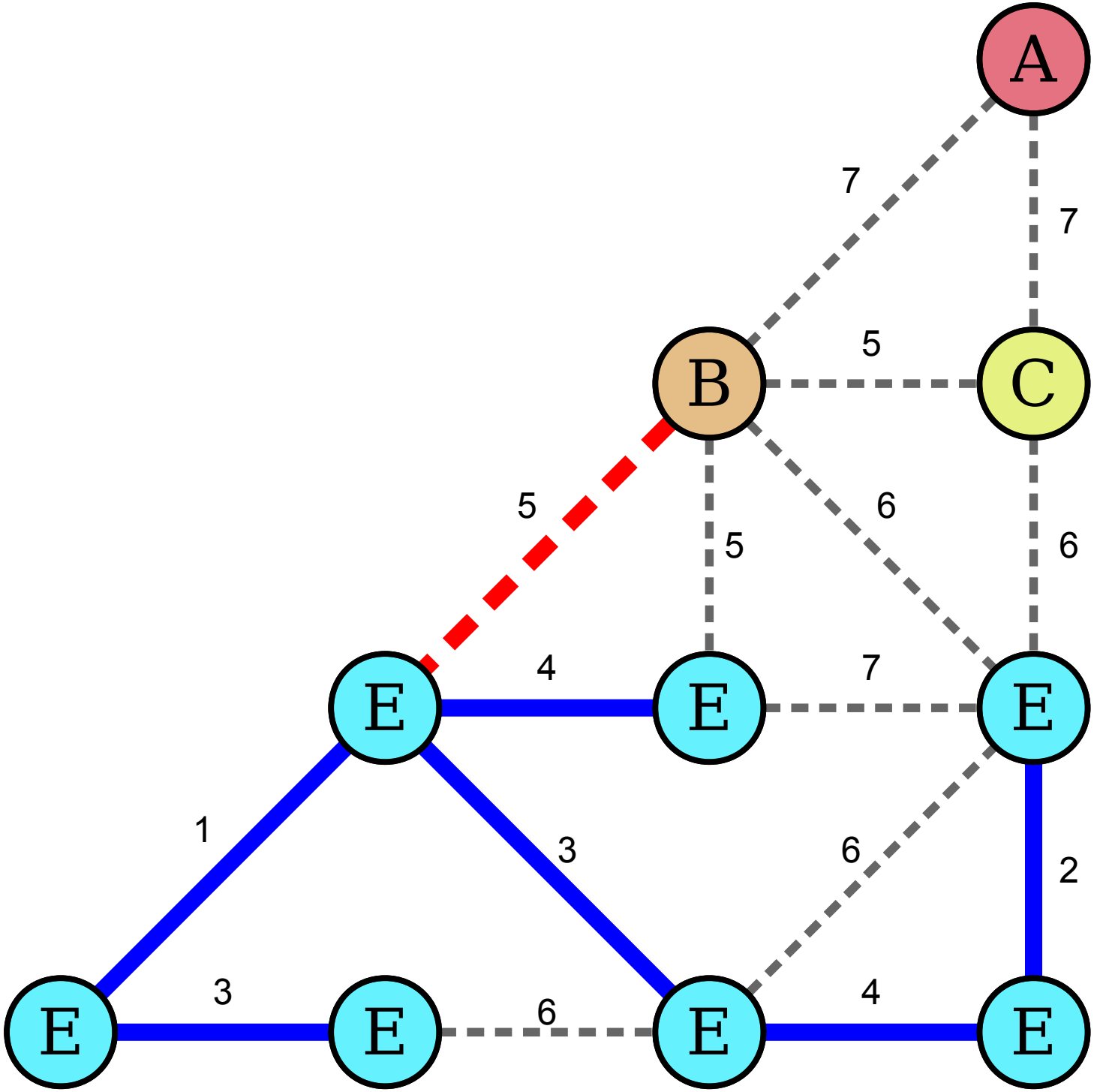

A
B
C
E
E
J
E
E
E
J
7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

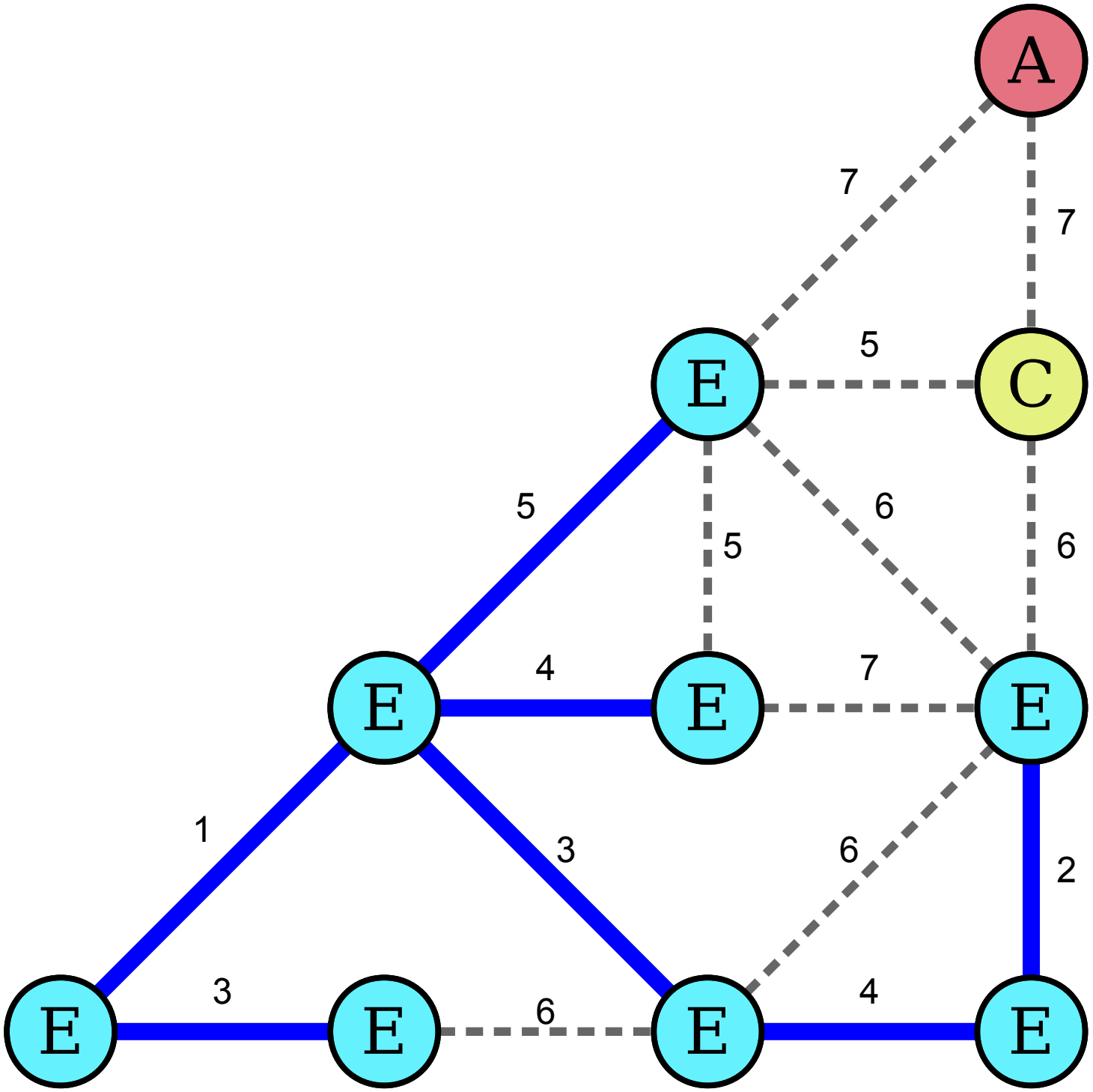

A
B
C
E
E
J
E
E
E
J
7
7
5
5
5
6
6
4
7
1
3
4
6
2
3
6
4

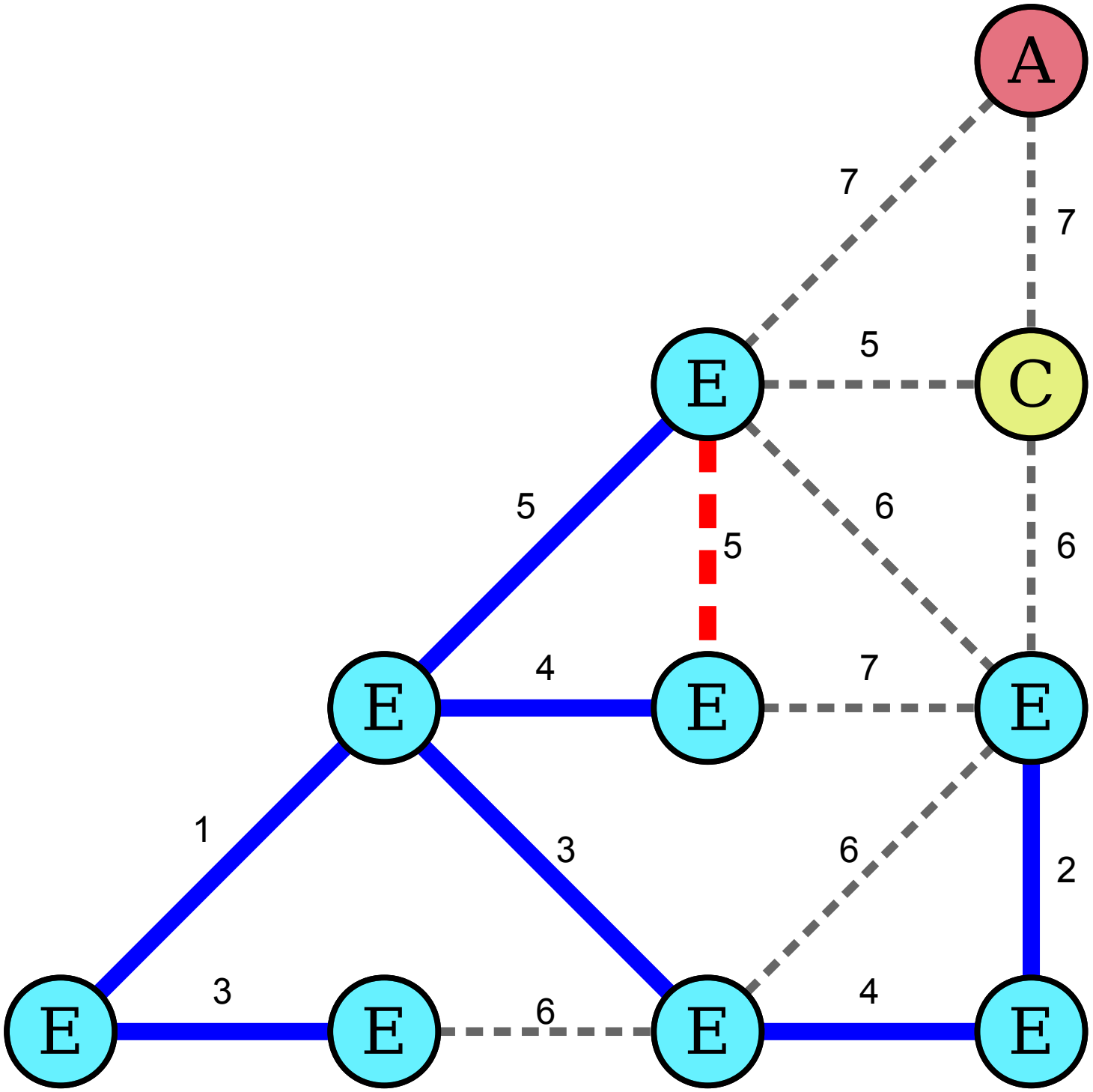

A
B
C
E
E
E
E
E
J
J
7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

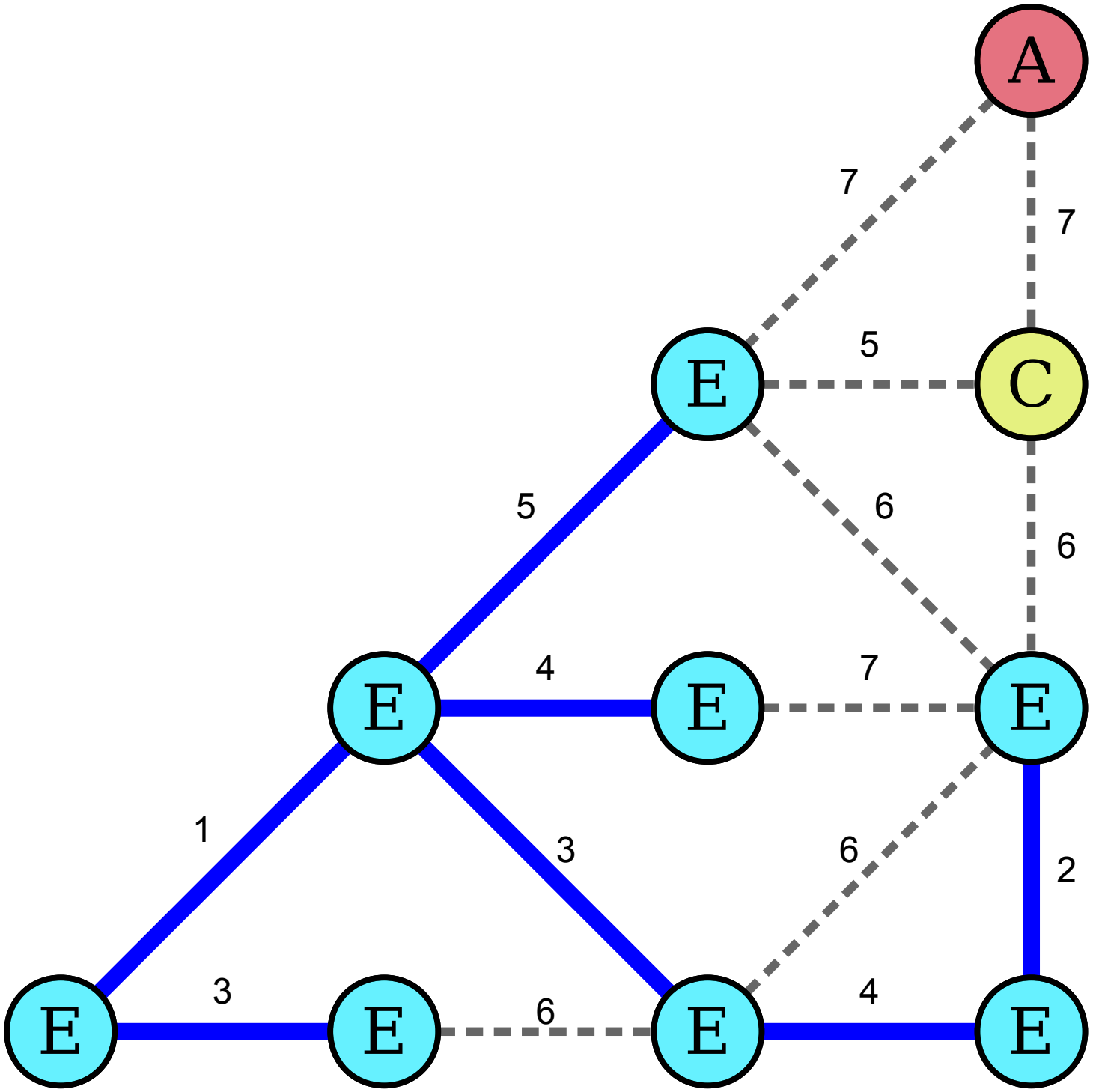

A
B
C
E
E
J
E
E
E
J
7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

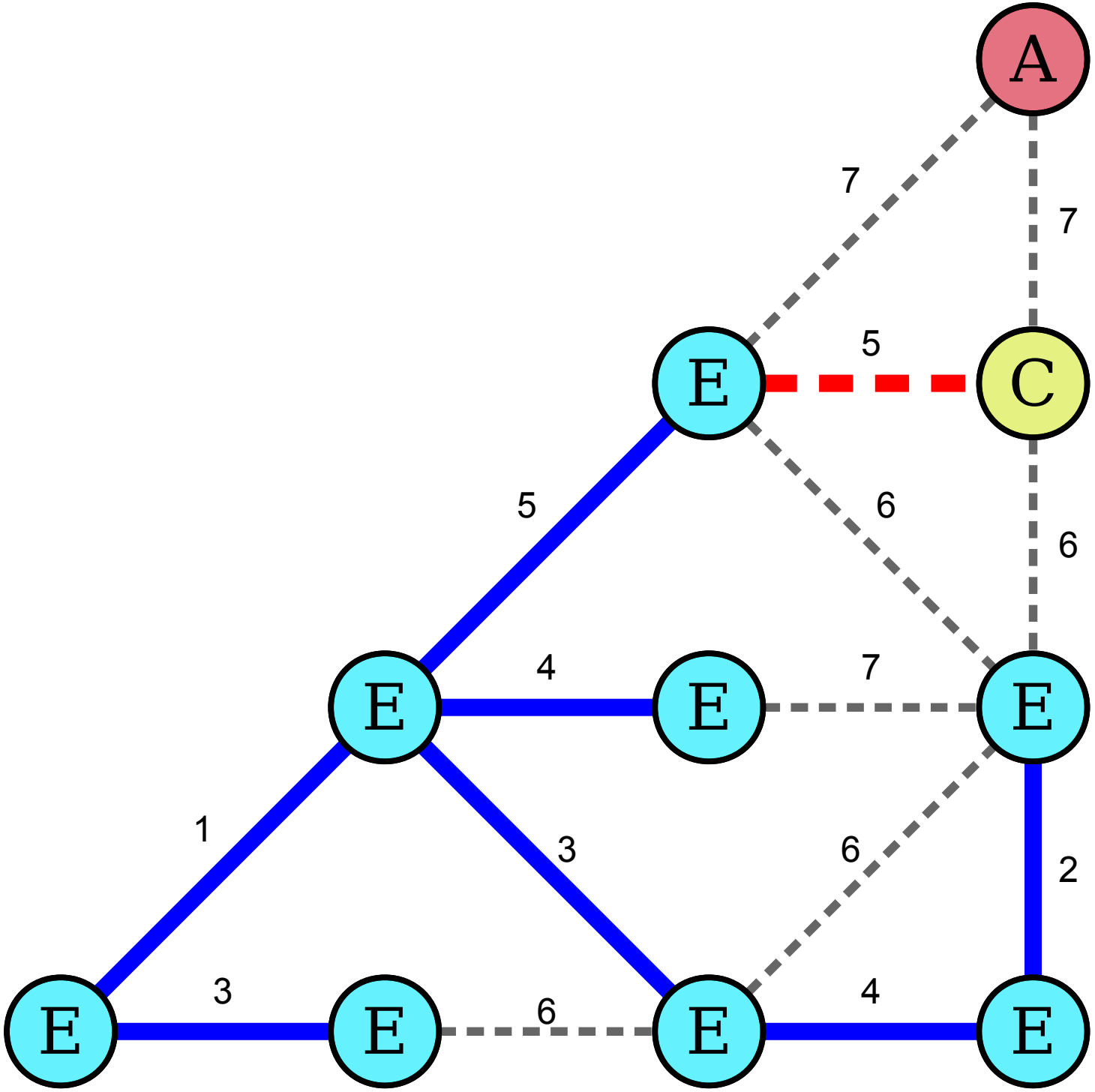

A
B
C
E
E
E
E
E
E
7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

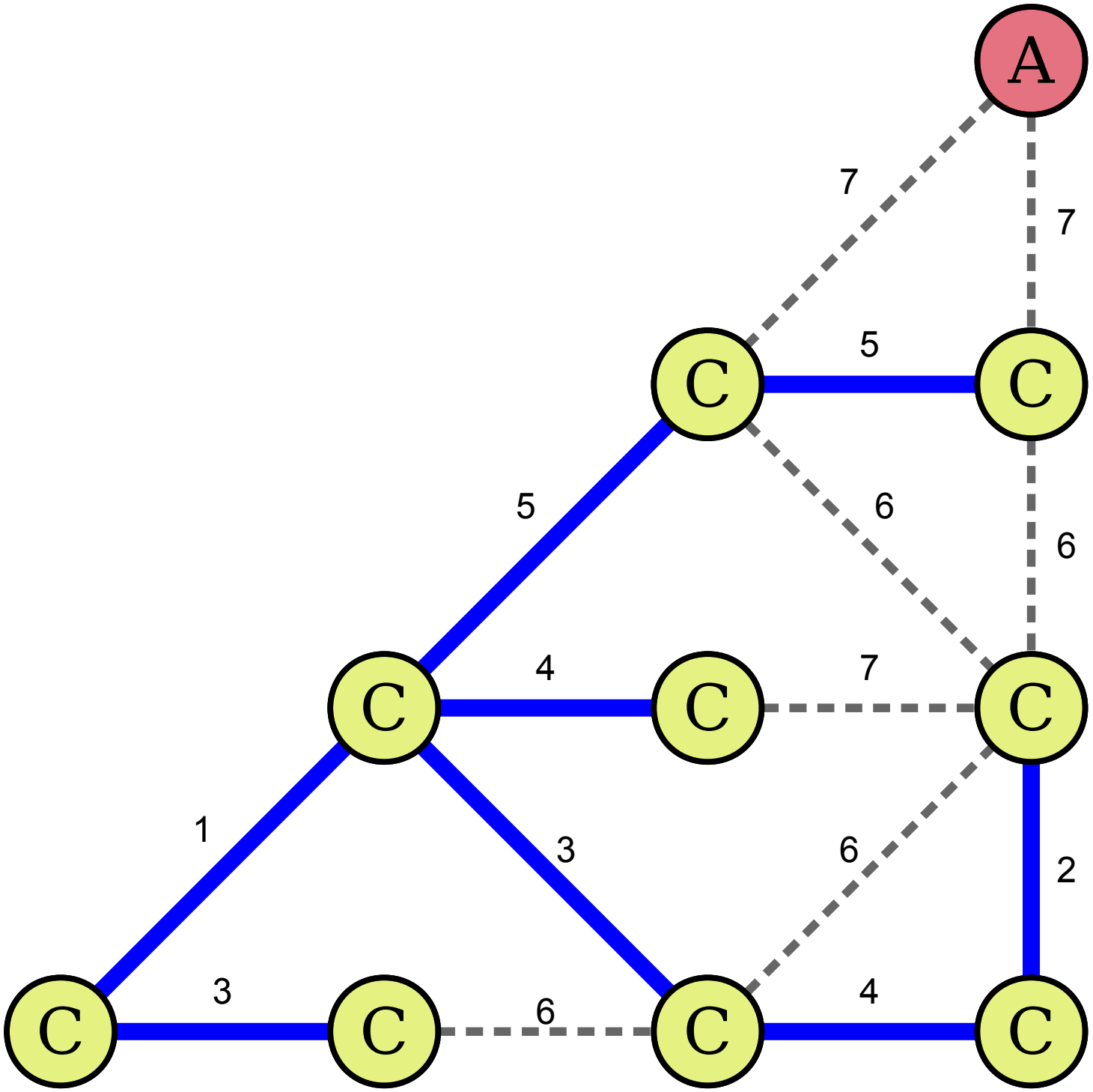

A
B
C
E
E
E
E
E
E
7
7
5
5
5
6
6
4
7
1
3
6
2
3
6
4

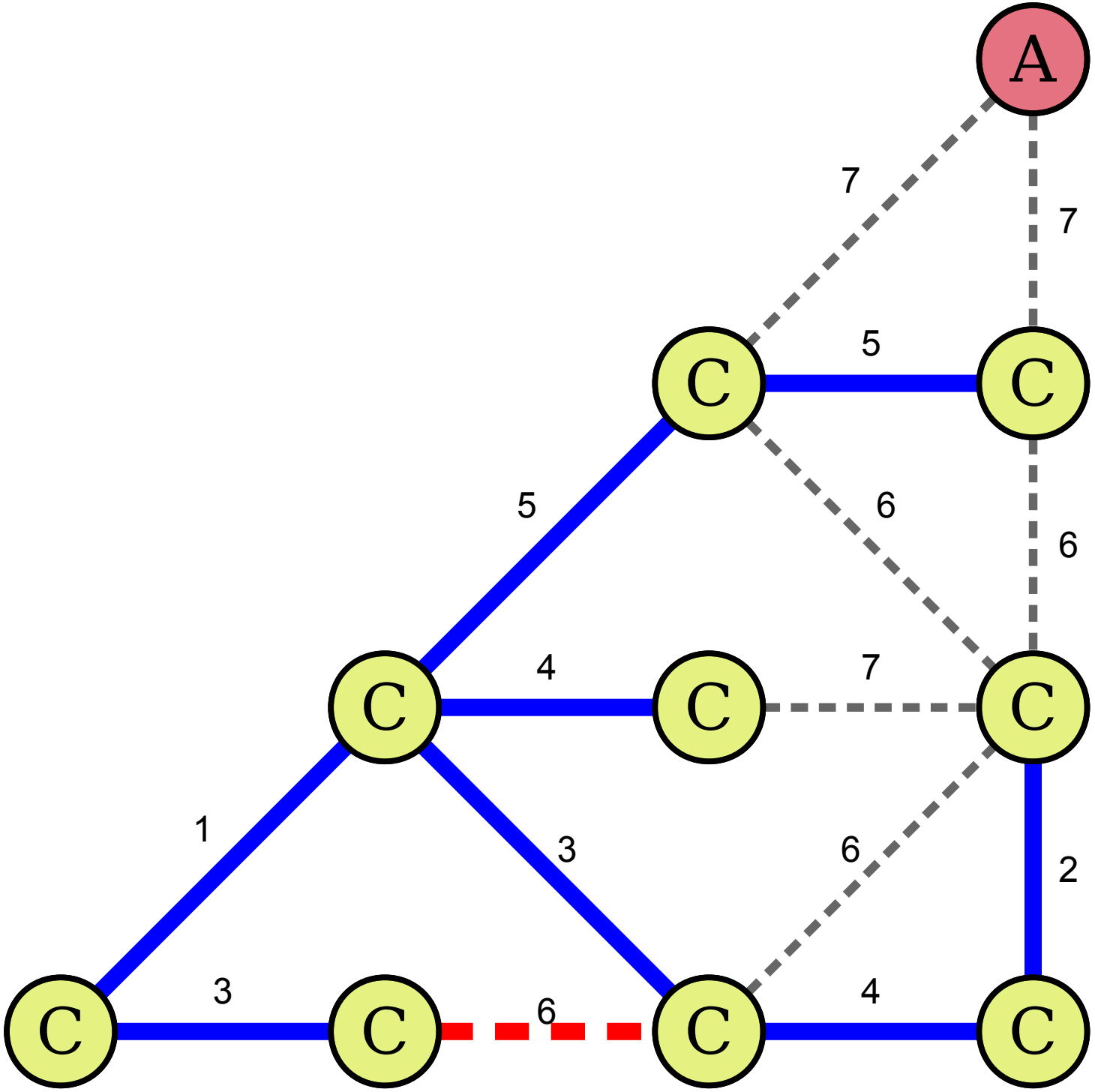

A
7
7
E
5
C
5
6
6
5
E
4
E
7
E
1
3
6
2
6
E
3
E
4
E
E

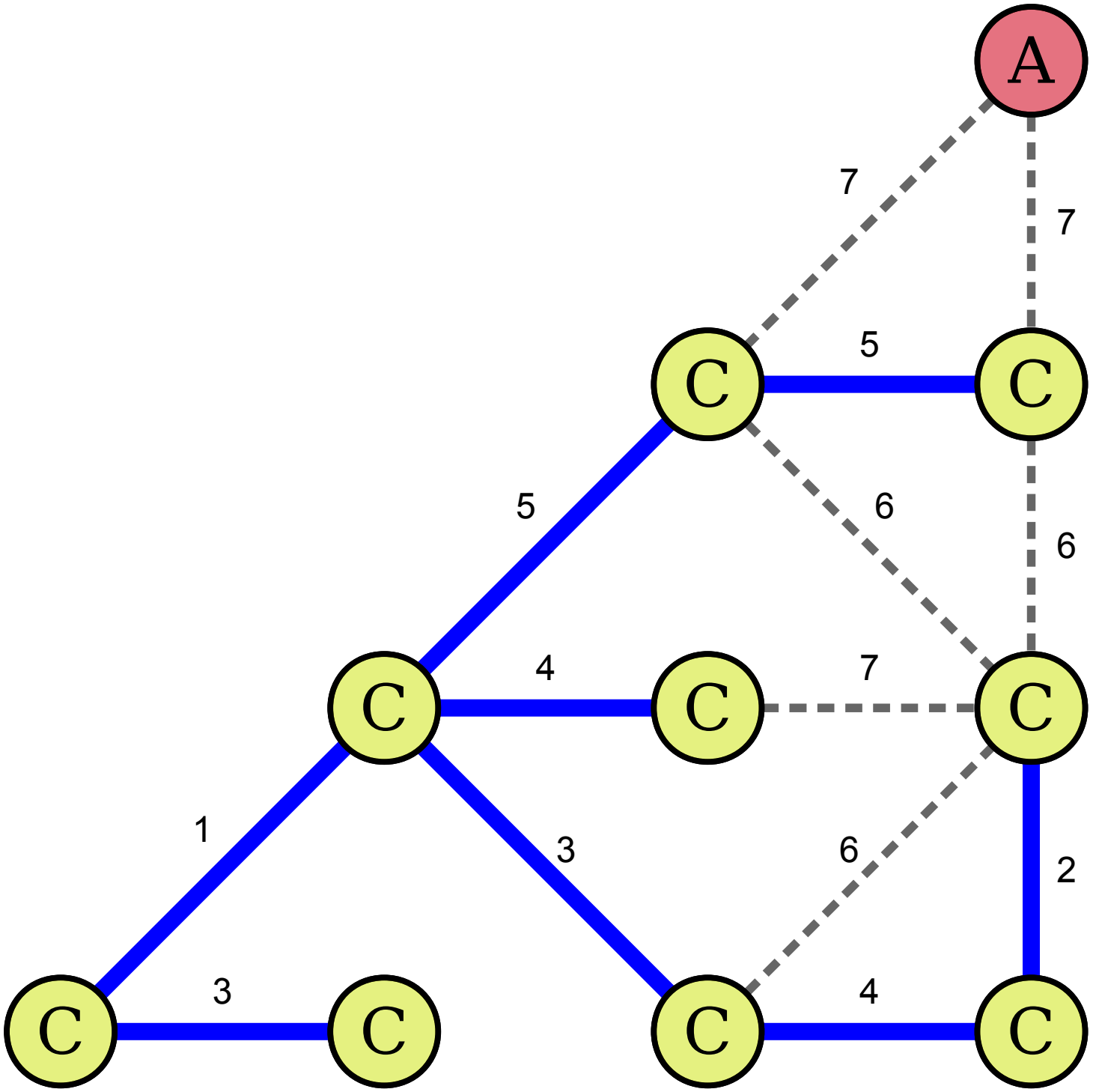

A
7
7
E
5
C
5
6
6
5
E
4
E
7
E
1
3
6
2
E
3
E
6
E
4
E

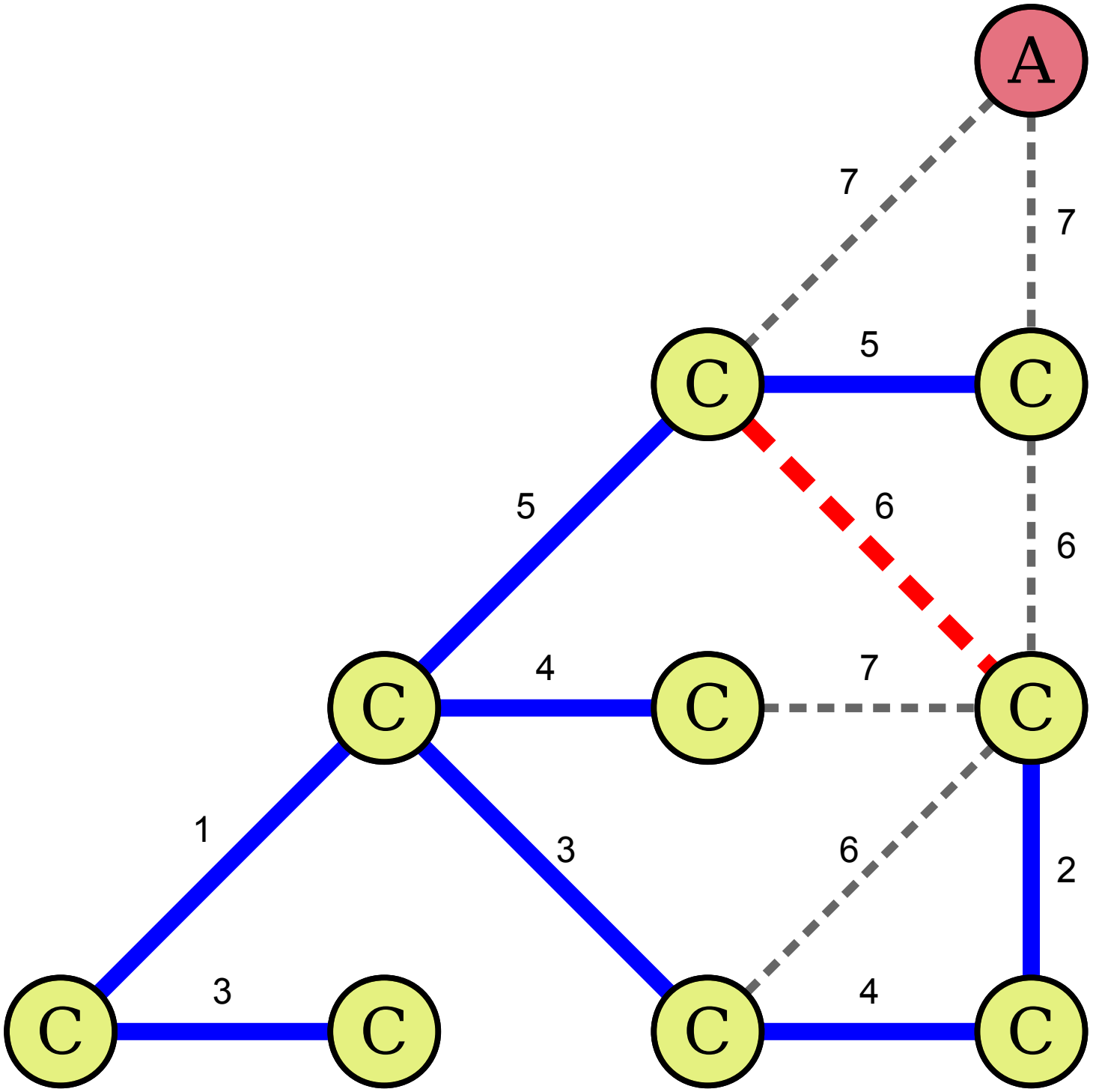

A
7
7
E
5
C
5
6
6
E
4
E
7
E
1
3
6
2
E
3
E
6
E
4
E

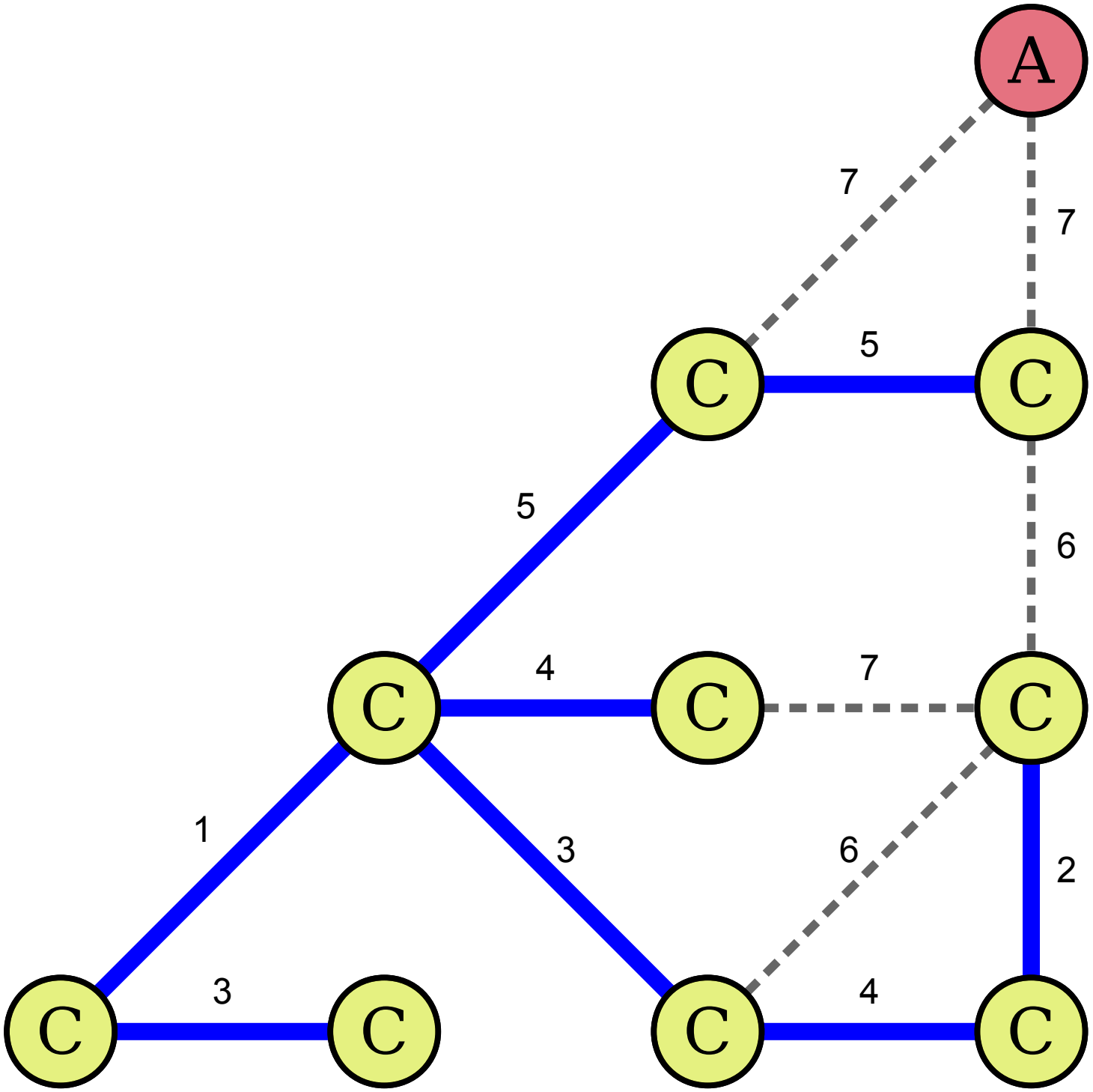

A
E
C
E
E
E
E
E
E
7
7
5
5
6
6
4
7
1
3
6
2
3
6
4

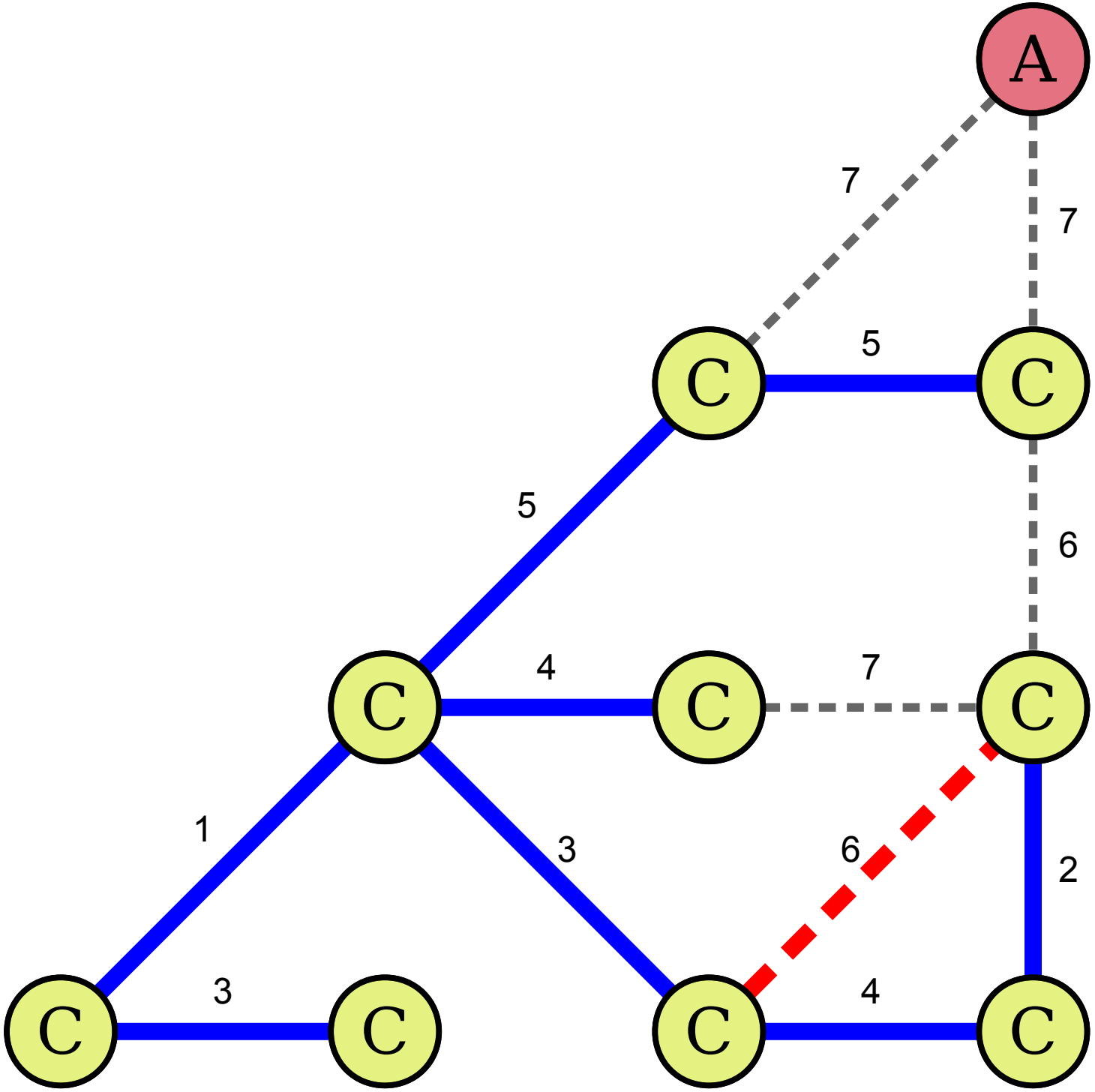

A
7
7
C
5
C
5
6
6
C
4
C
7
C
1
3
6
2
C
3
C
6
C
4
C

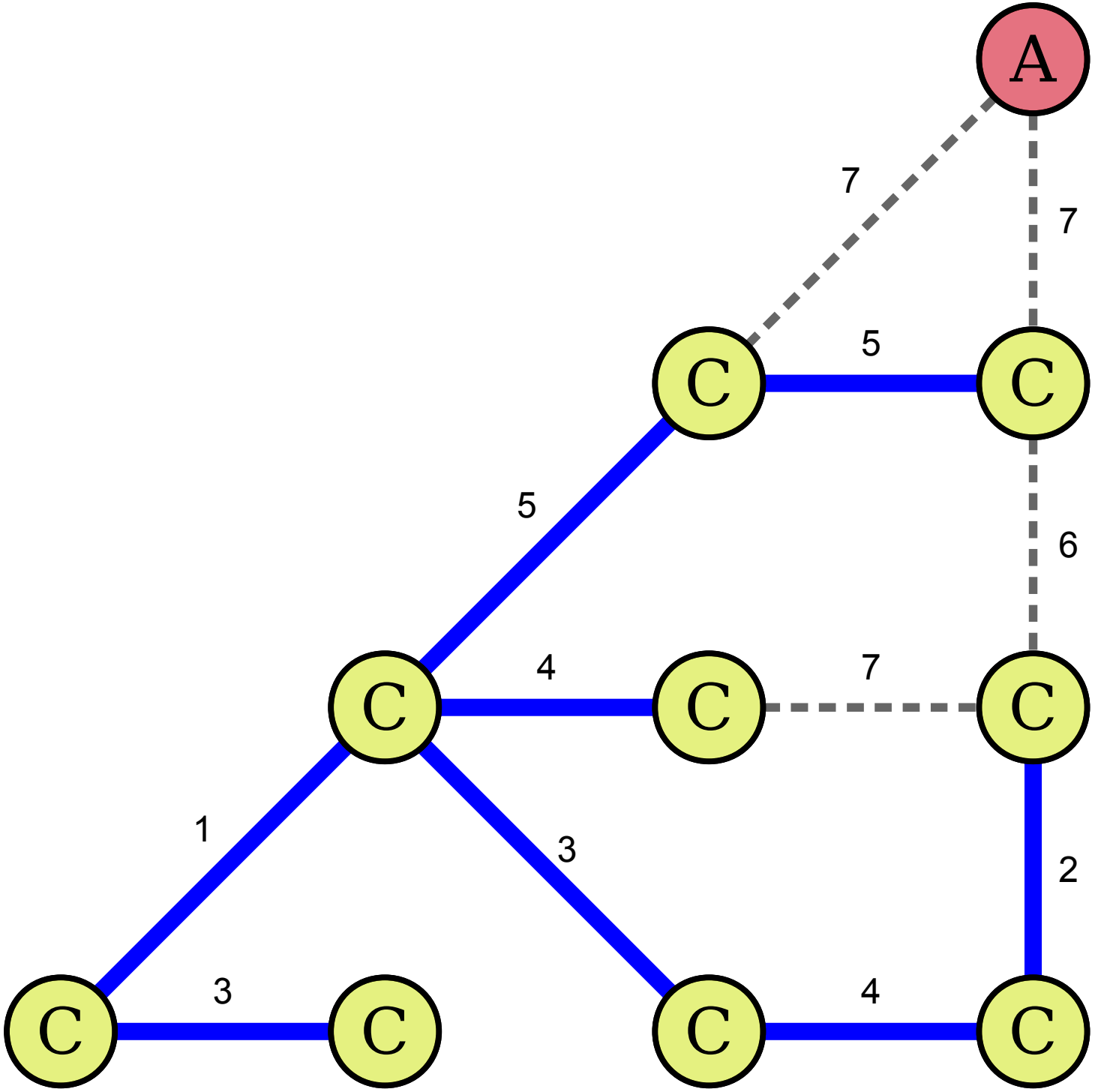

A
7
7
C
5
C
5
6
6
C
4
C
7
C
1
3
6
2
C
3
C
6
C
4
C

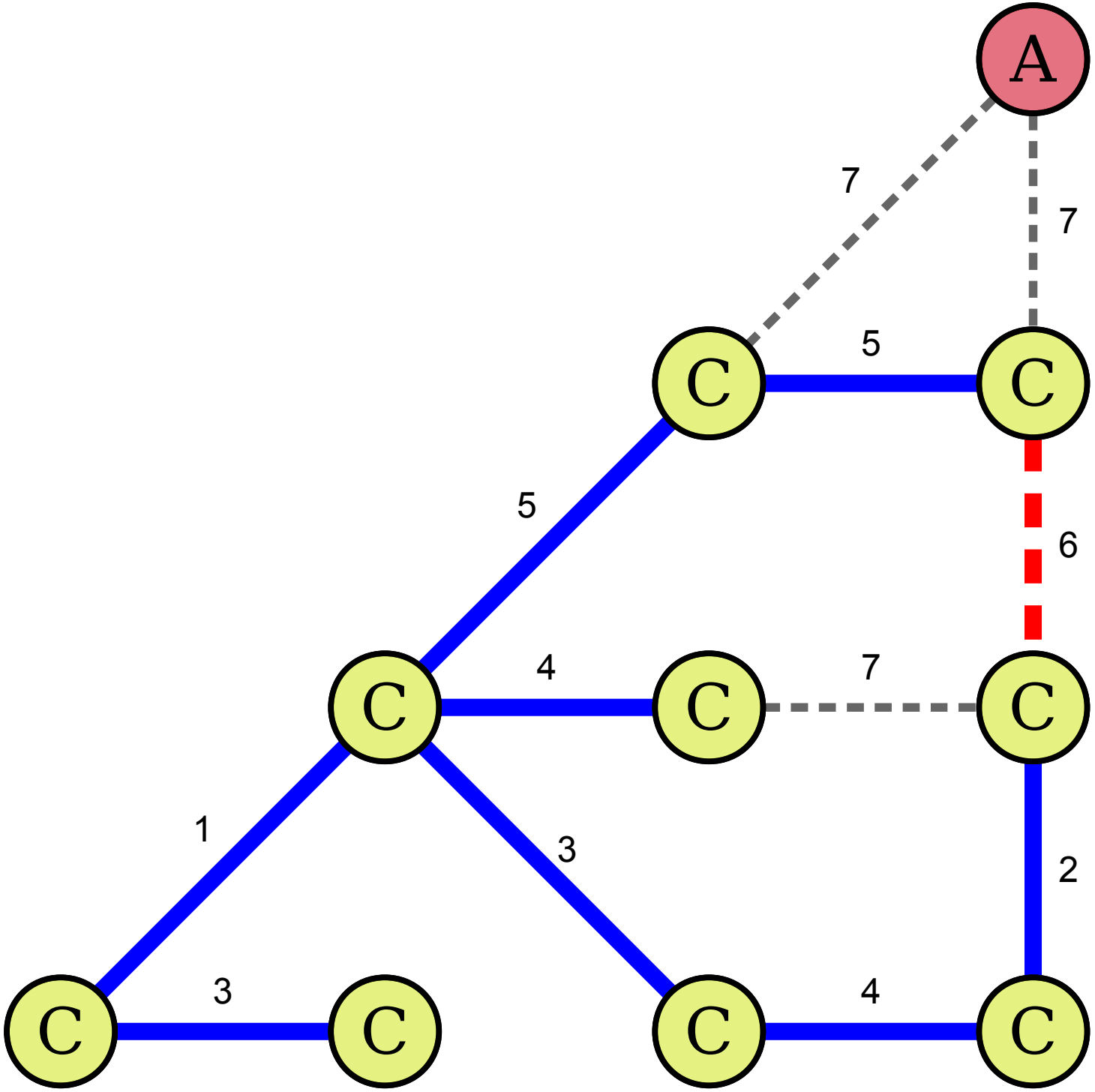

A
7
7
C
5
C
5
6
6
C
4
C
7
C
1
3
6
2
C
3
C
C
4
C

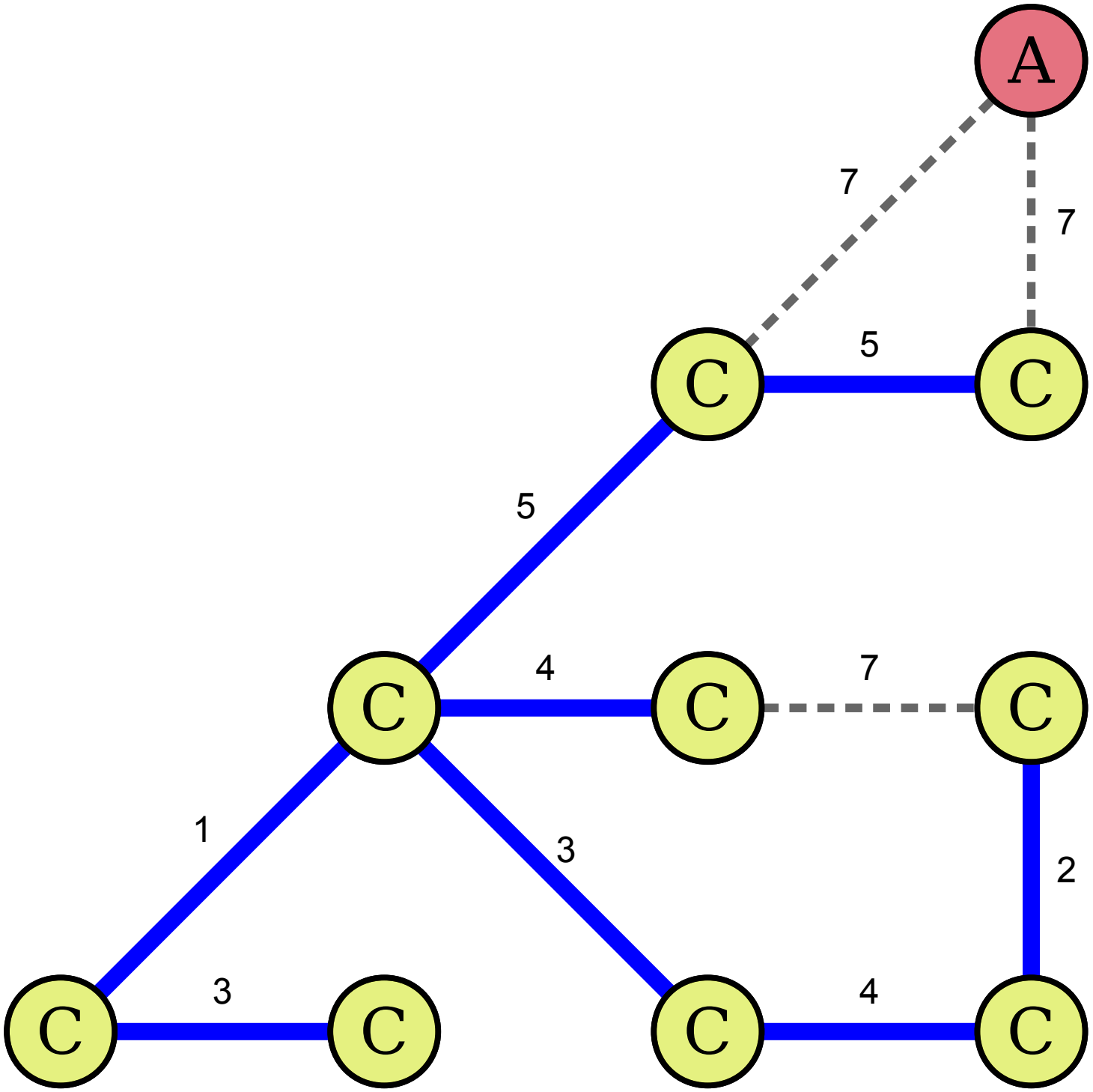

A
7
7
C
5
C
5
6
6
C
4
C
7
C
1
3
6
2
C
3
C
C
4
C

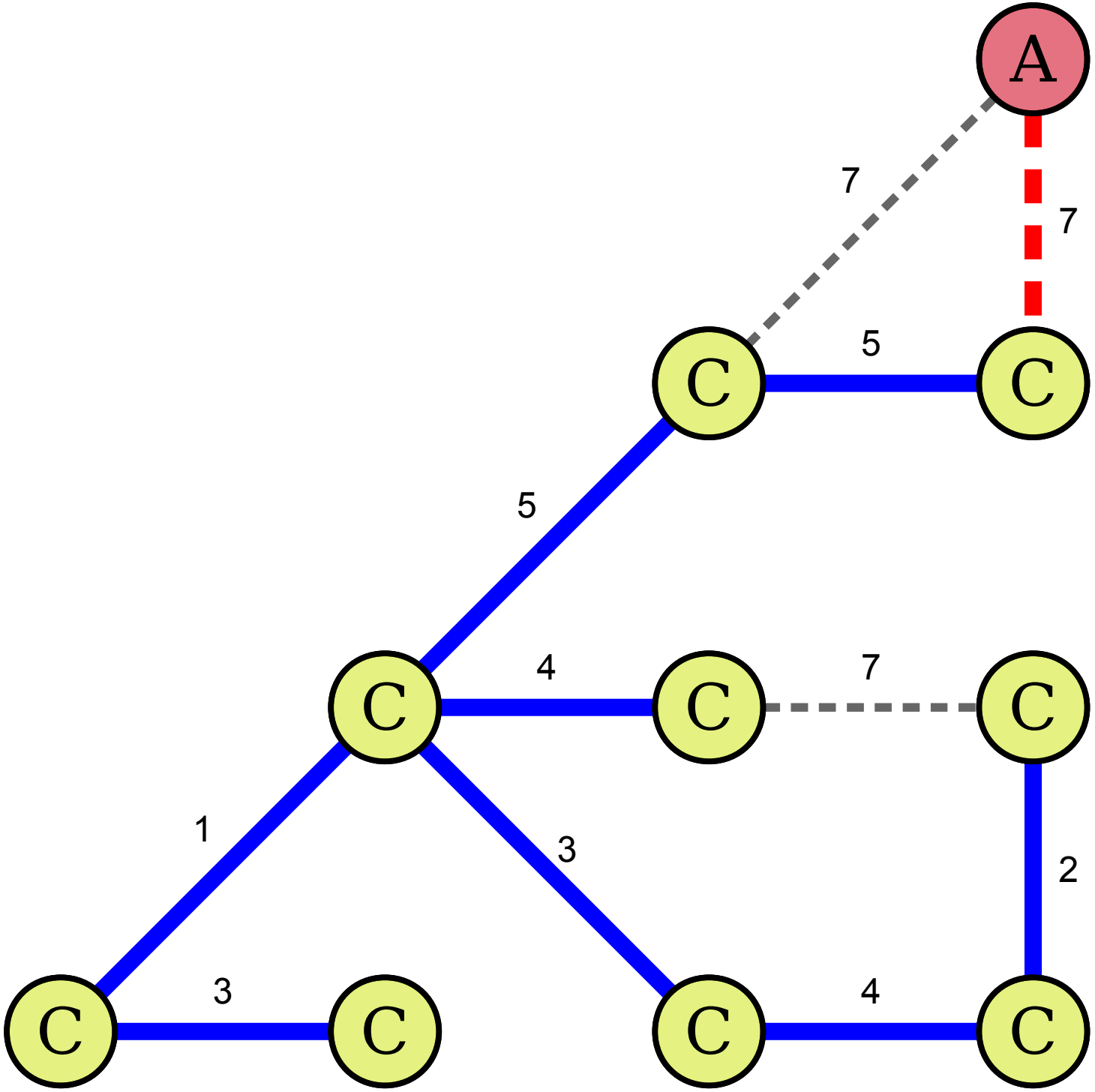

A
7
7
C
5
C
5
6
C
4
C
7
C
1
3
6
2
C
3
C
C
4
C

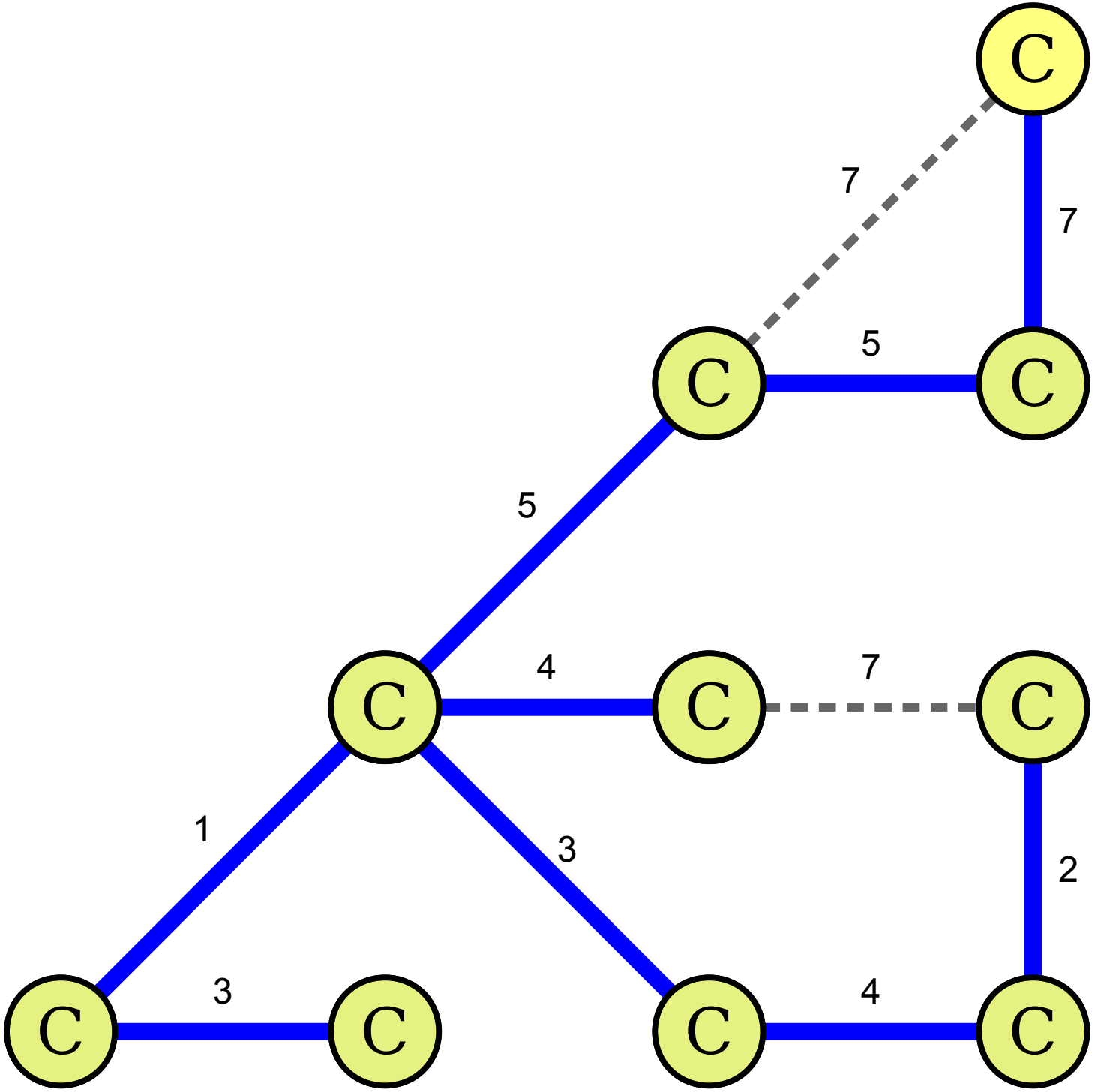

A
7
7
C
5
C
5
6
C
4
C
7
C
1
3
6
2
C
3
C
C
4
C

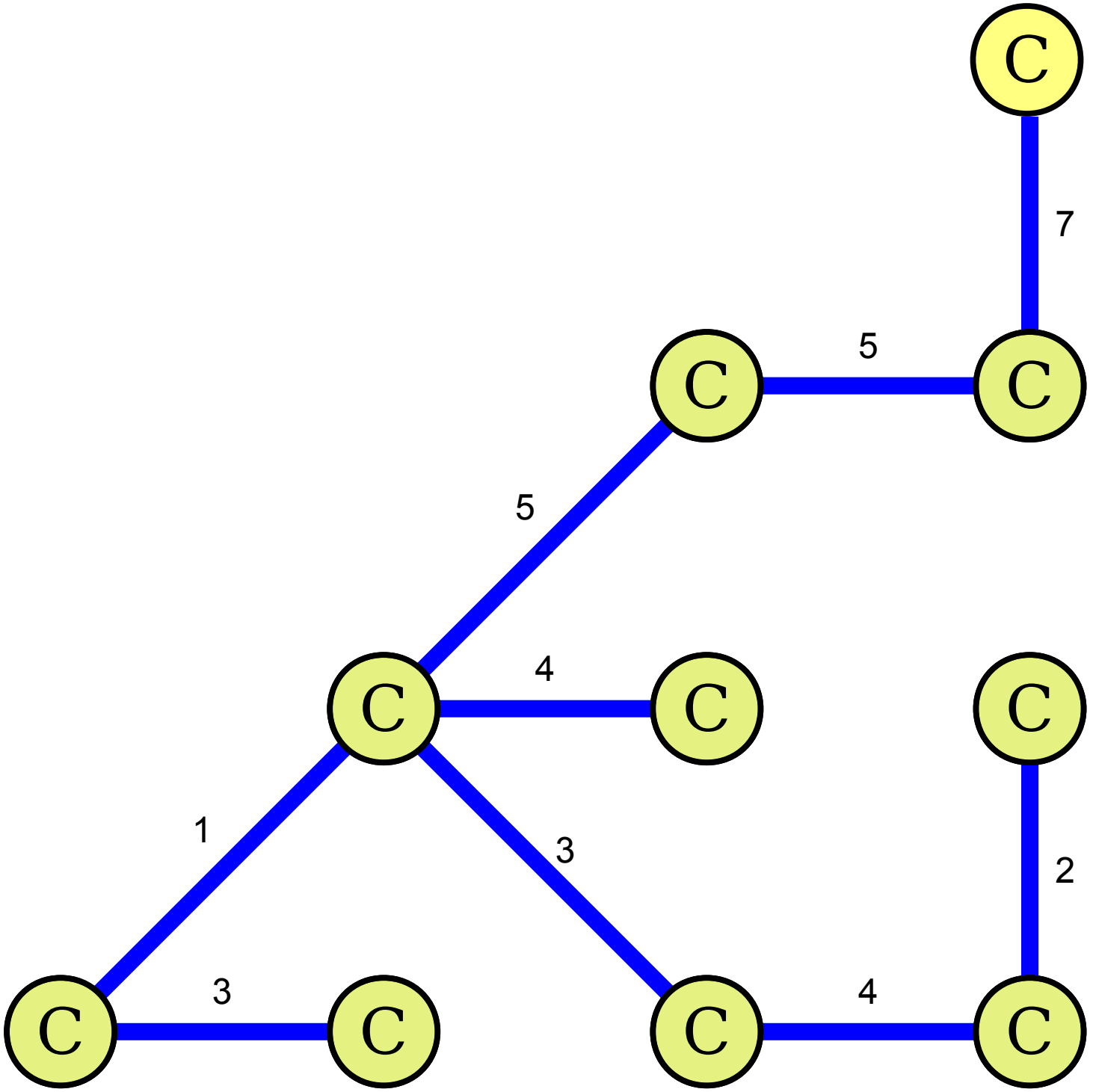

A
7
7
C
5
C
5
6
C
4
C
7
C
1
3
2
C
3
C
C
4
C


A
7
7
C
5
C
5
6
C
4
C
7
C
1
3
2
C
3
C
C
4
C


A
7
7
C
5
C
5
C
4
C
7
C
1
3
2
C
3
C
C
4
C


A
7
7
C
5
C
5
C
4
C
7
C
1
3
2
C
3
C
C
4
C


C
C
C
C
C
C
C
C
C
7
7
5
5
4
7
1
3
2
3
4


C
C
C
C
C
C
C
C
C
7
5
5
4
1
3
2
3
4

Kruskal's with Clusters

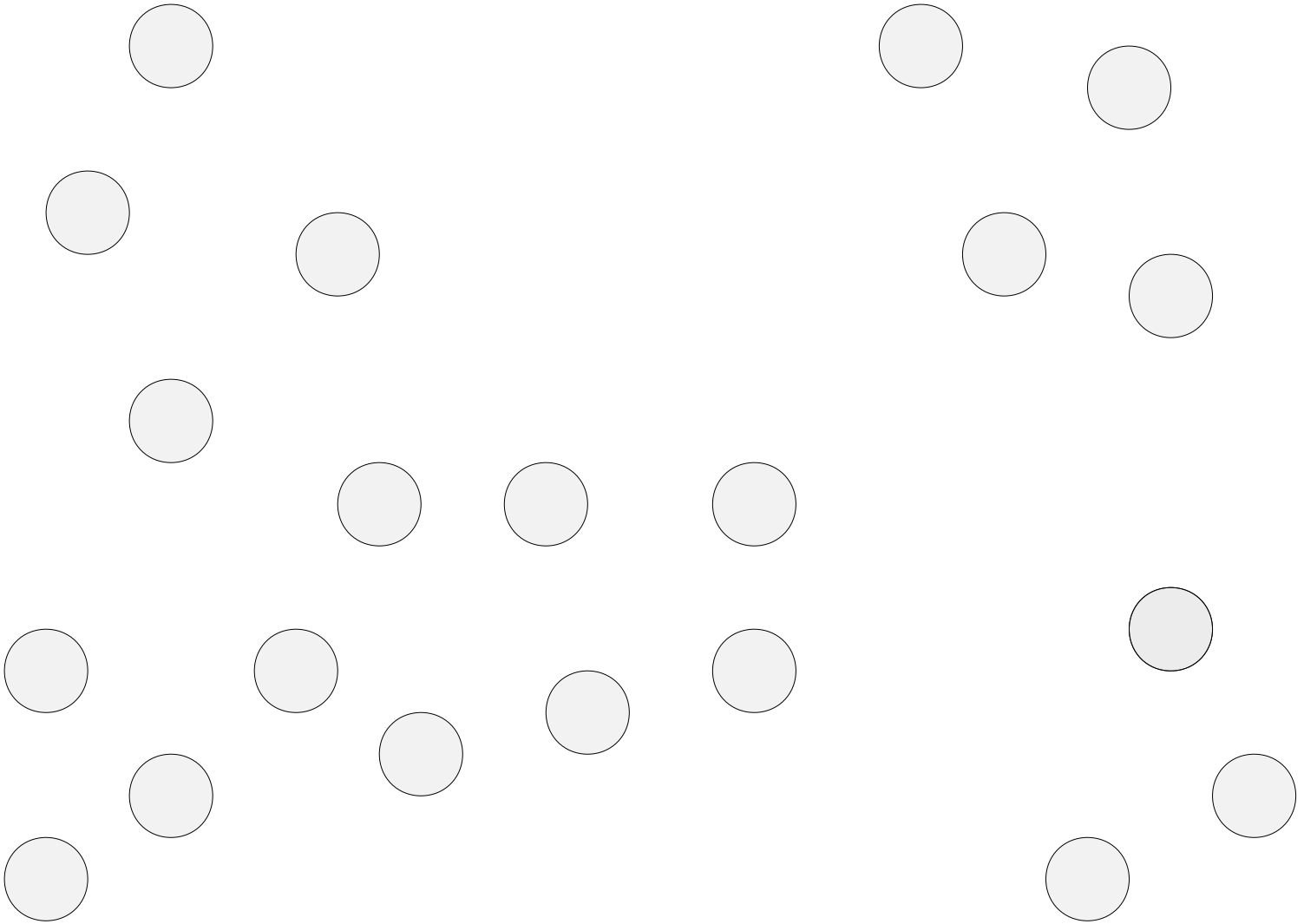
- Place every node into its own cluster.
- Place all edges into a priority queue.
- While there are two or more clusters remaining:
  - Dequeue an edge from the priority queue.
  - If its endpoints are not in the same cluster:
    - Merge the clusters containing the endpoints.
    - Add the edge to the resulting spanning tree.
- Return the resulting spanning tree.

Kruskal's with Clusters

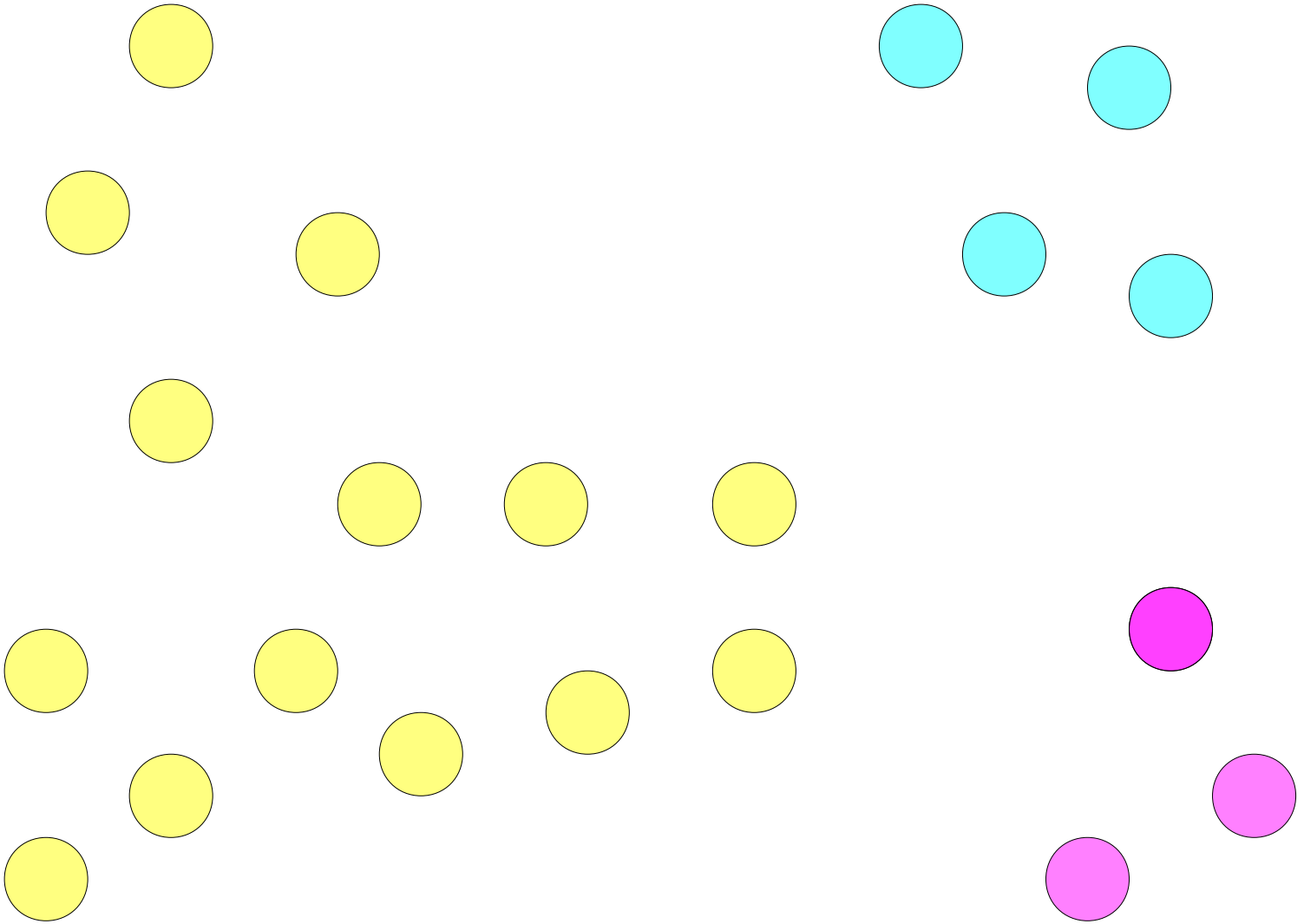
- Specialized data structures exist for maintaining the clusters in Kruskal's algorithm.
- One such structure: **disjoint-set forest**.
  - Not particularly complicated.
  - Check Wikipedia for details.
  - Easy extra credit on the last assignment (details in a bit.)

Applications of Kruskal's Algorithm

Data Clustering



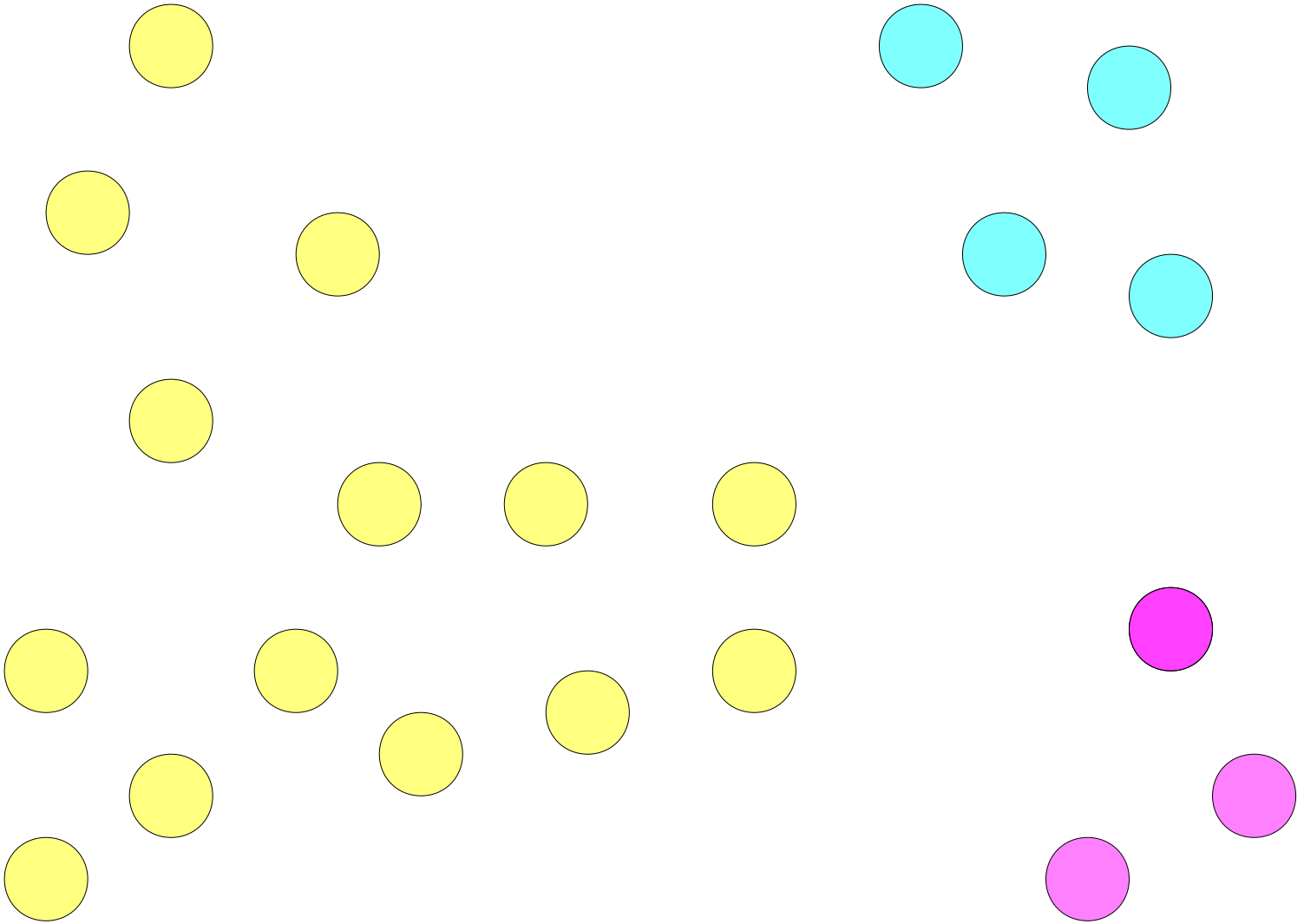
Data Clustering



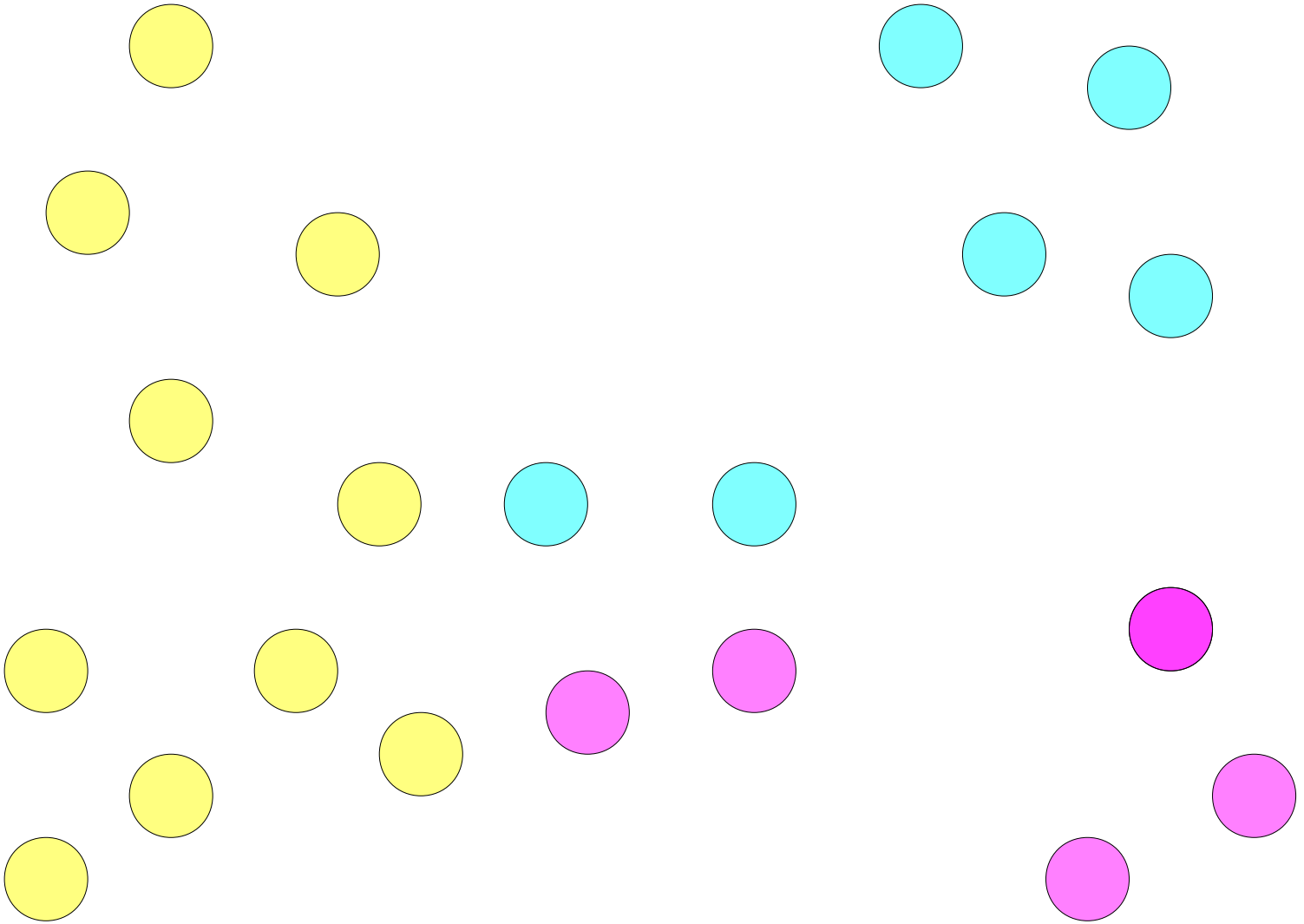
Data Clustering

- Given a set of points, break those points apart into clusters.
- Immensely useful across all disciplines:
  - Cluster individuals by phenotype to try to determine what genes influence which traits.
  - Cluster images by pixel color to identify objects in pictures.
  - Cluster essays by various features to see how students learn to write.

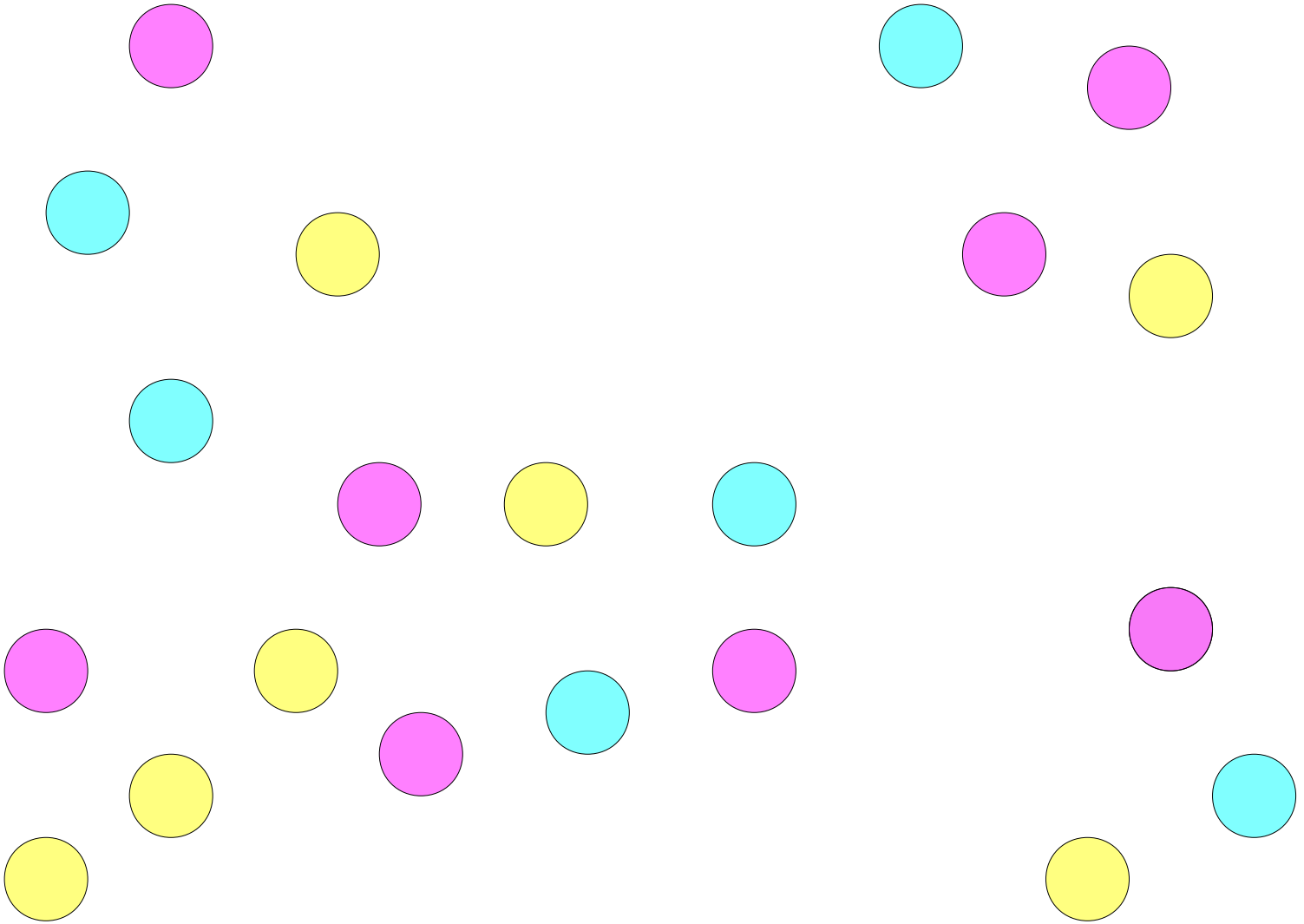
Data Clustering



Data Clustering



Data Clustering

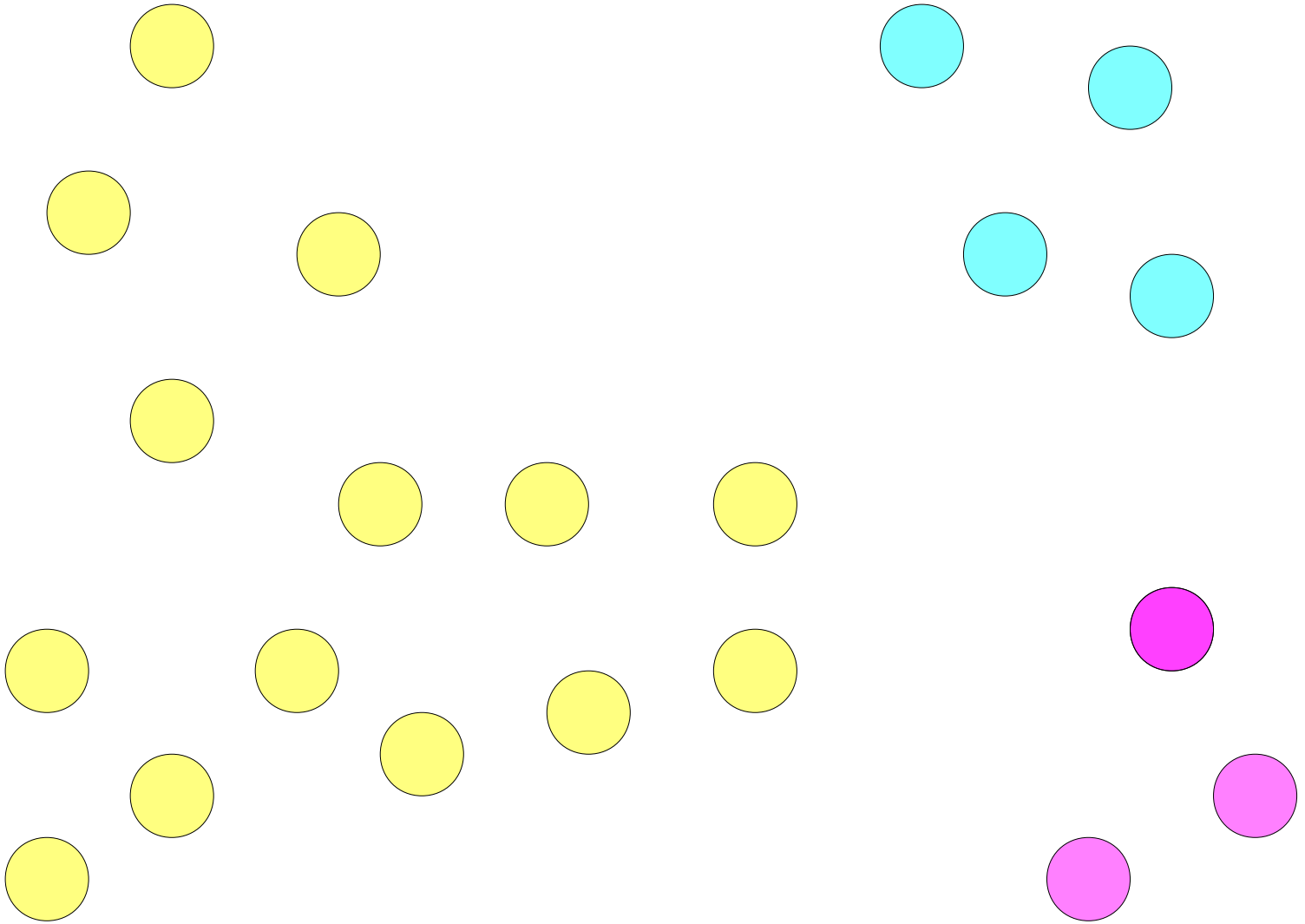


What makes a clustering “good?”

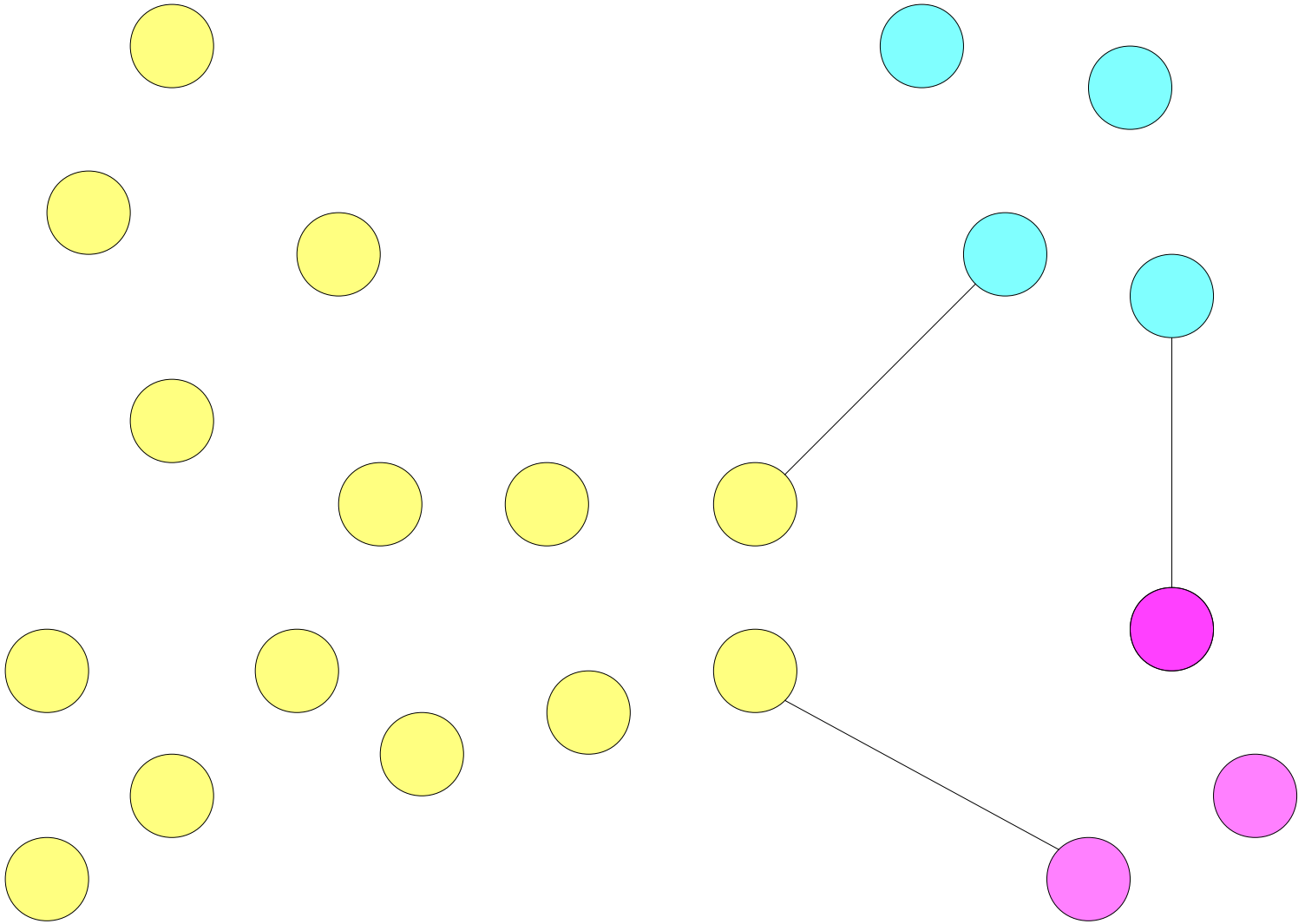
Maximum-Separation Clustering

- **Maximum-separation clustering** tries to find a clustering that maximizes the separation between different clusters.
- Specifically: Maximize the minimum distance between any two points of different clusters.
- Very good on many data sets, though not always ideal.

Maximum-Separation Clustering



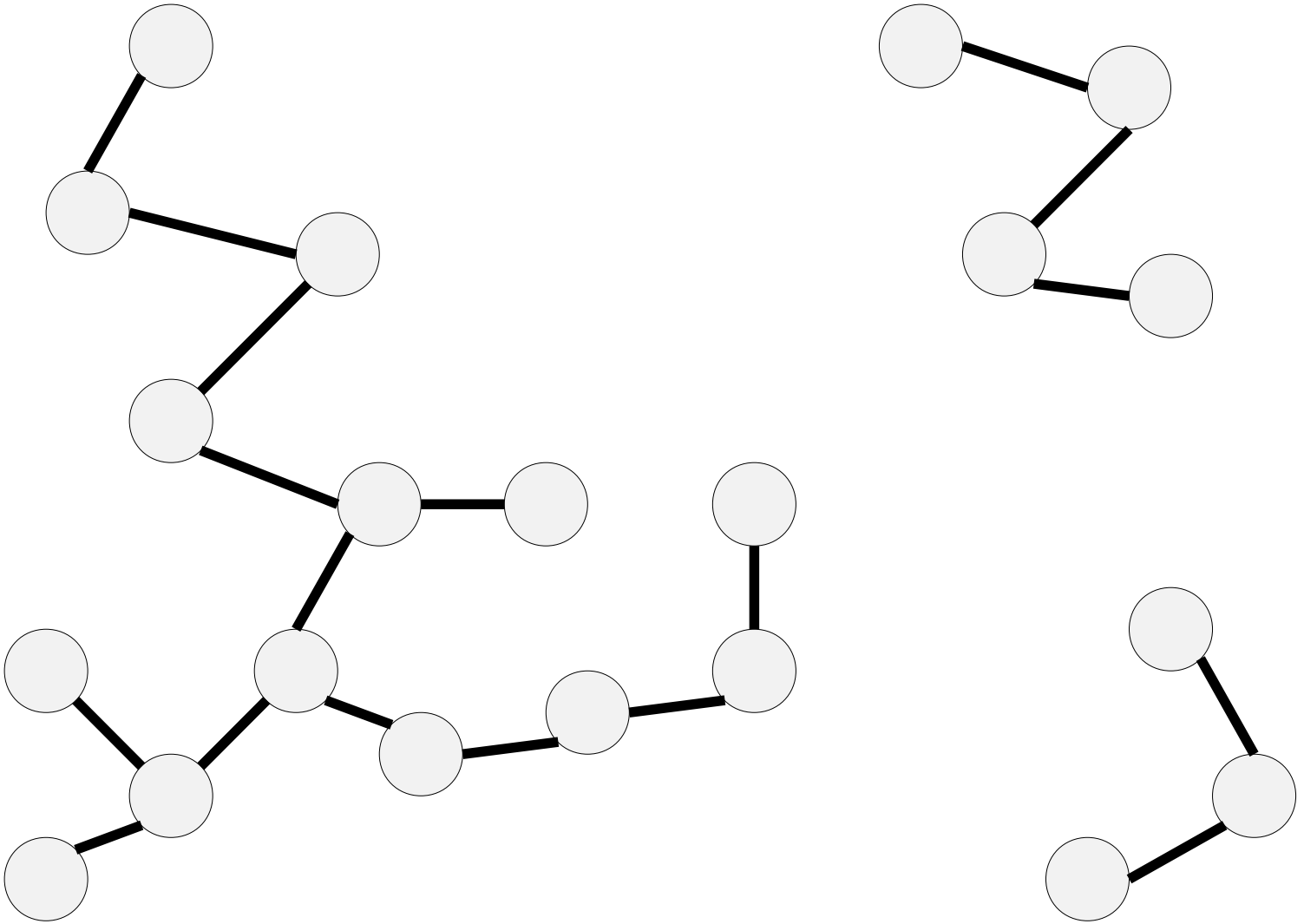
Maximum-Separation Clustering



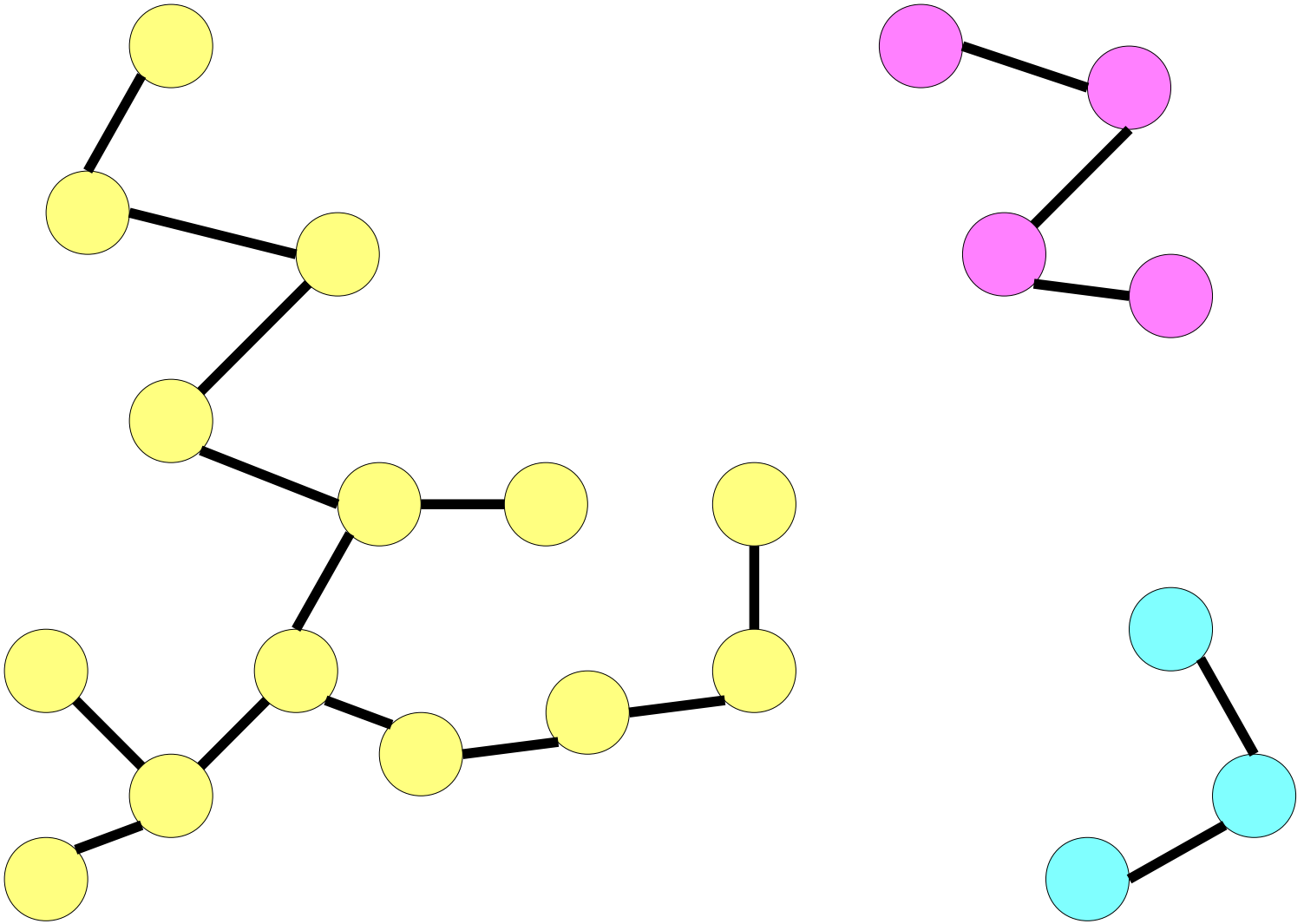
Maximum-Separation Clustering

- It is extremely easy to adopt Kruskal's algorithm to produce a maximum-separation set of clusters.
  - Suppose you want k clusters.
  - Given the data set, add an edge from each node to each other node whose length depends on their similarity.
  - Run Kruskal's algorithm until only k clusters remain.
  - The pieces of the graph that have been linked together are k maximally-separated clusters.

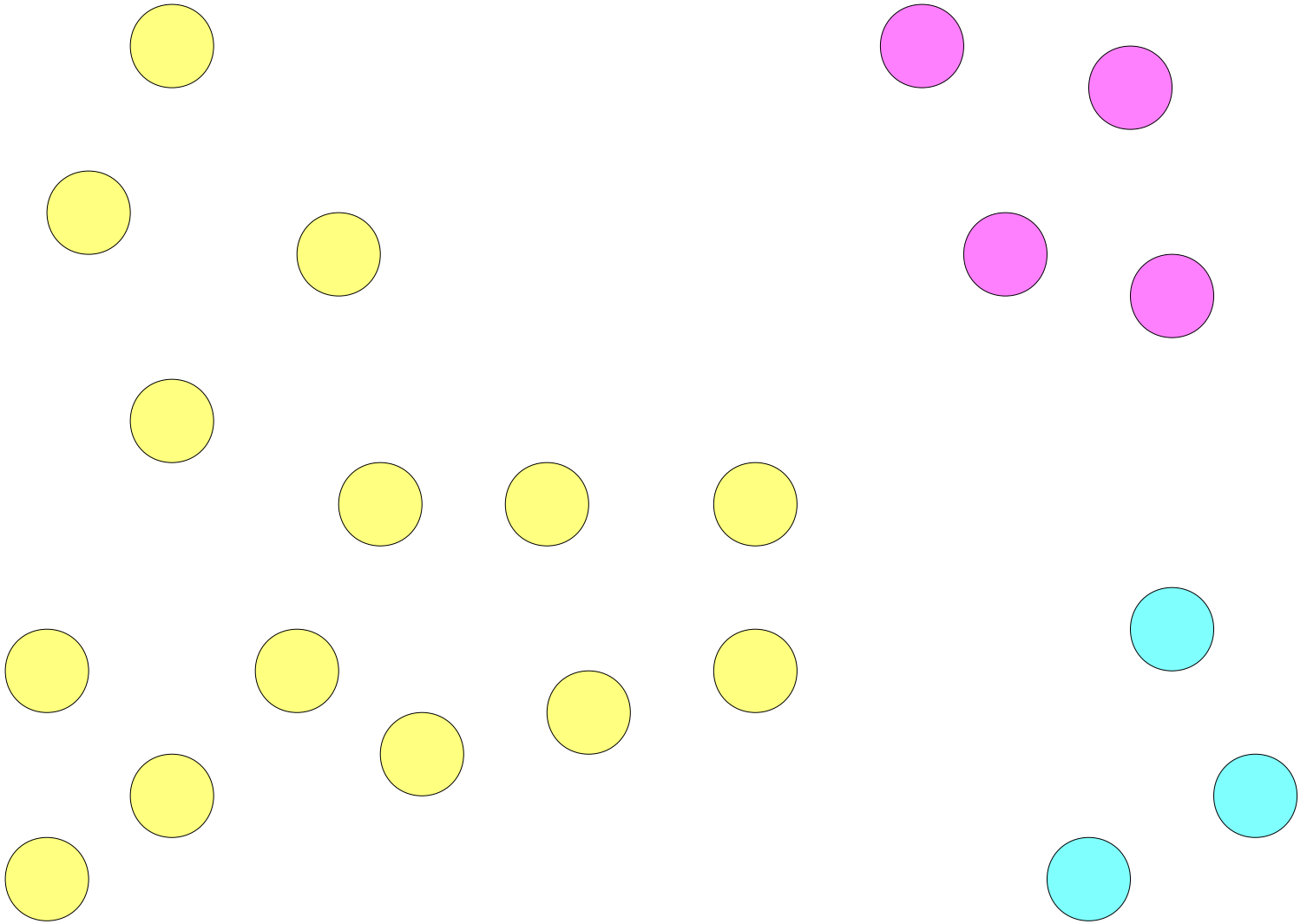
Maximum-Separation Clustering



Maximum-Separation Clustering

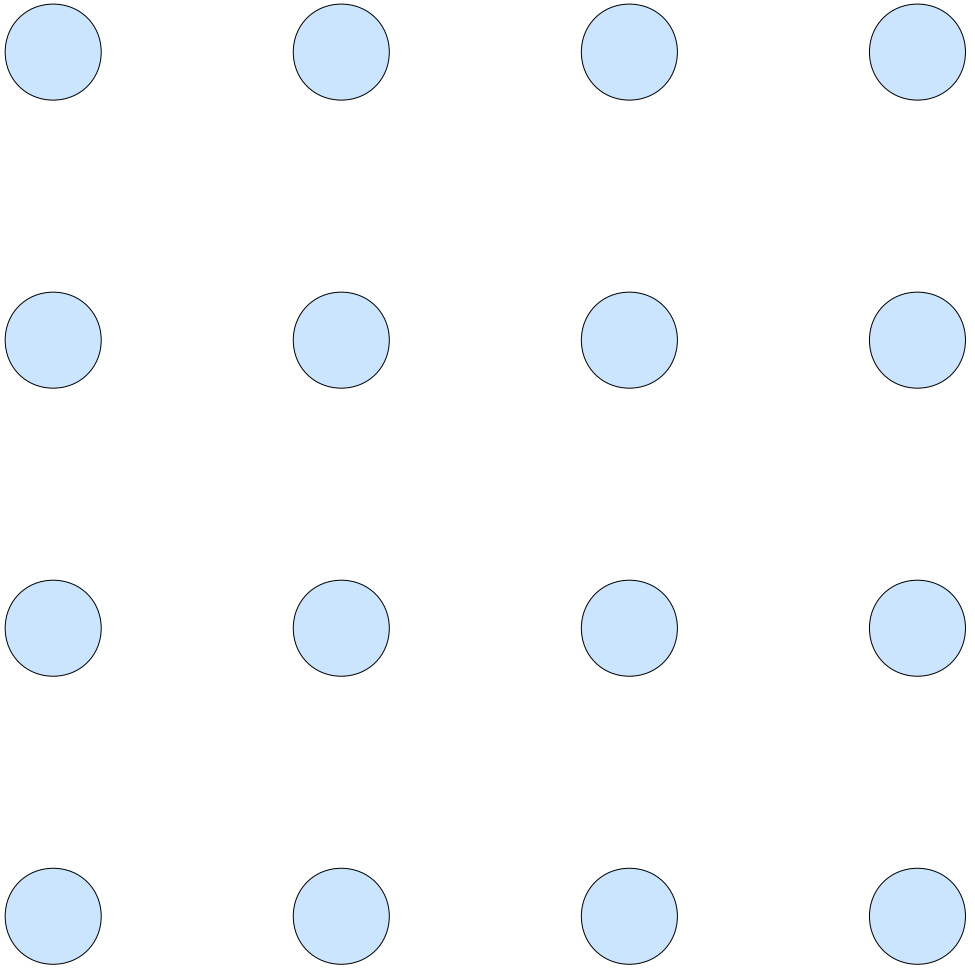


Maximum-Separation Clustering

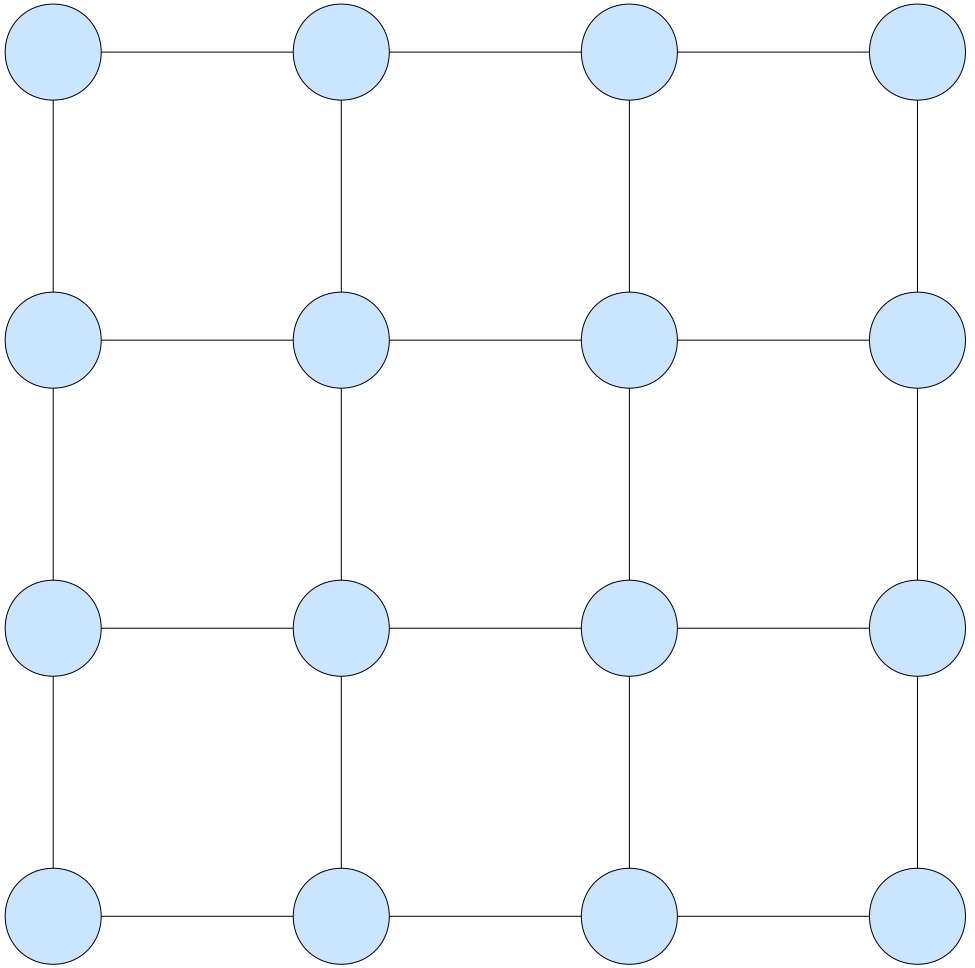


Mazes with Kruskal's Algorithm

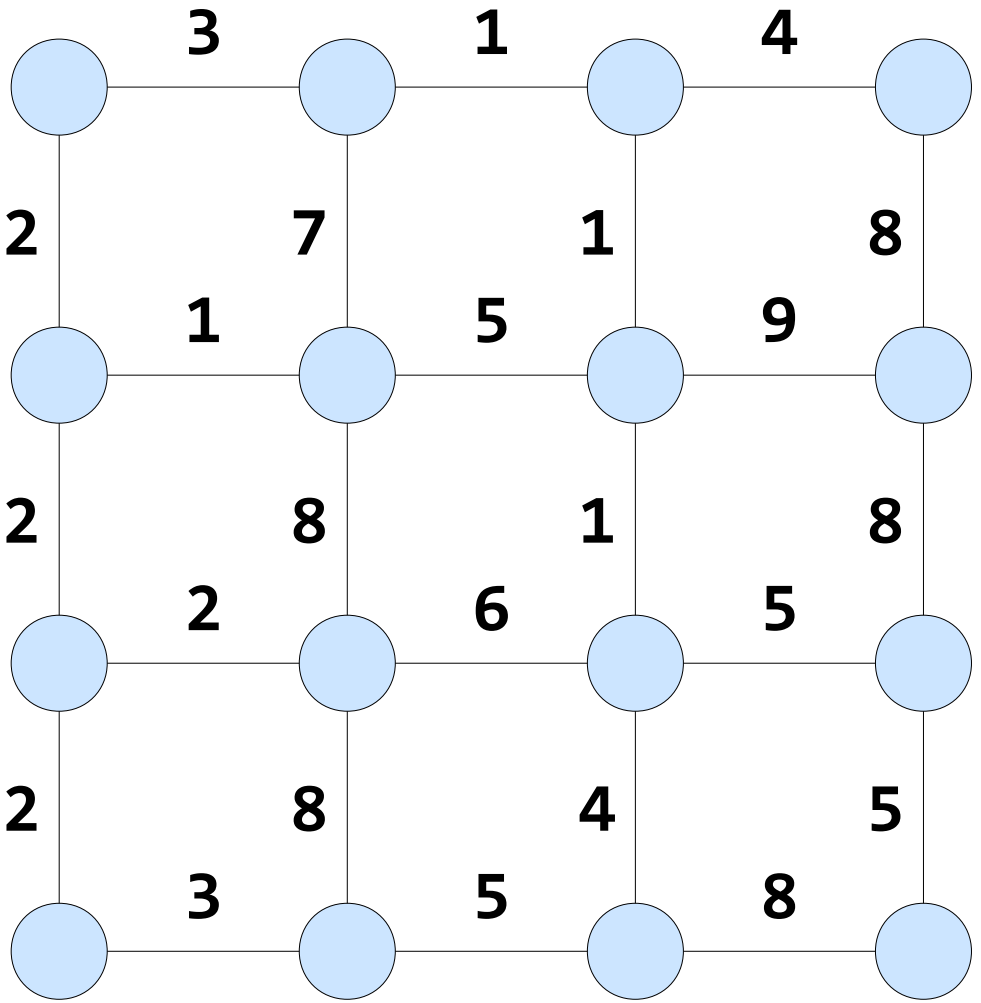
Mazes with Kruskal's Algorithm



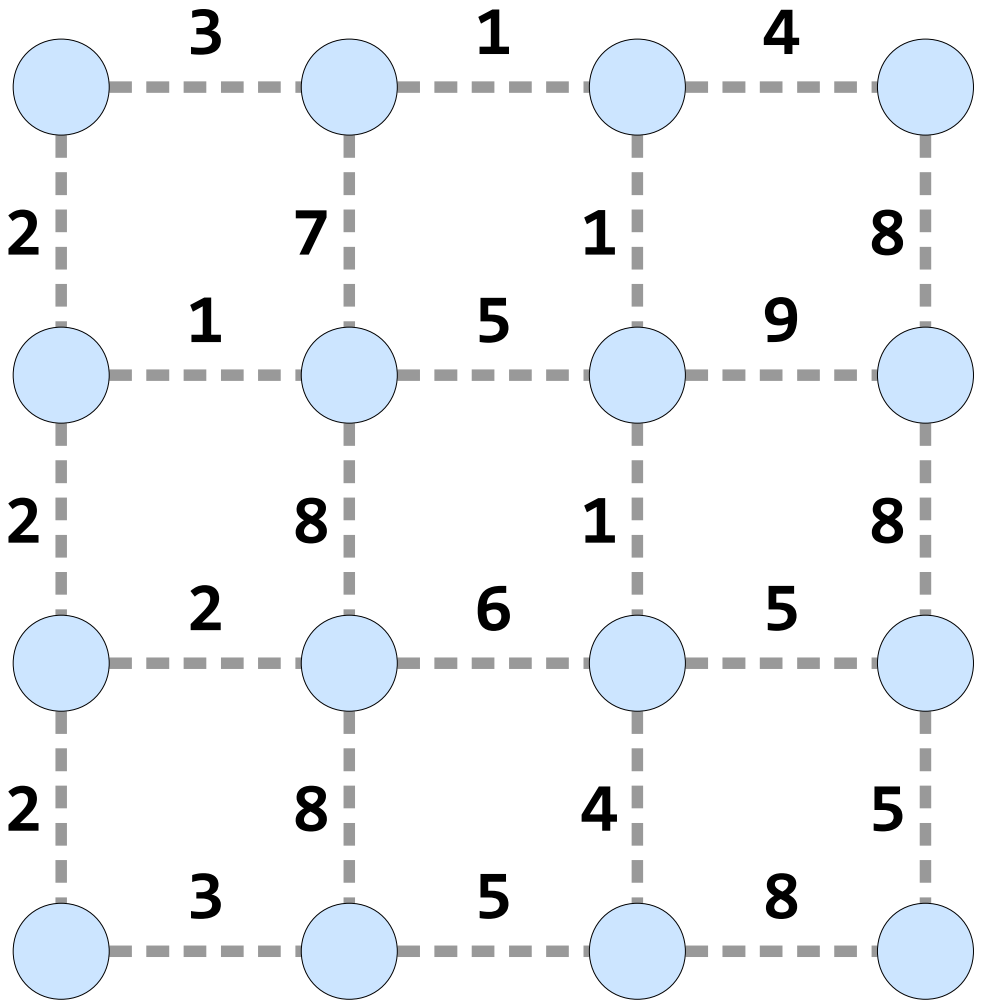
Mazes with Kruskal's Algorithm



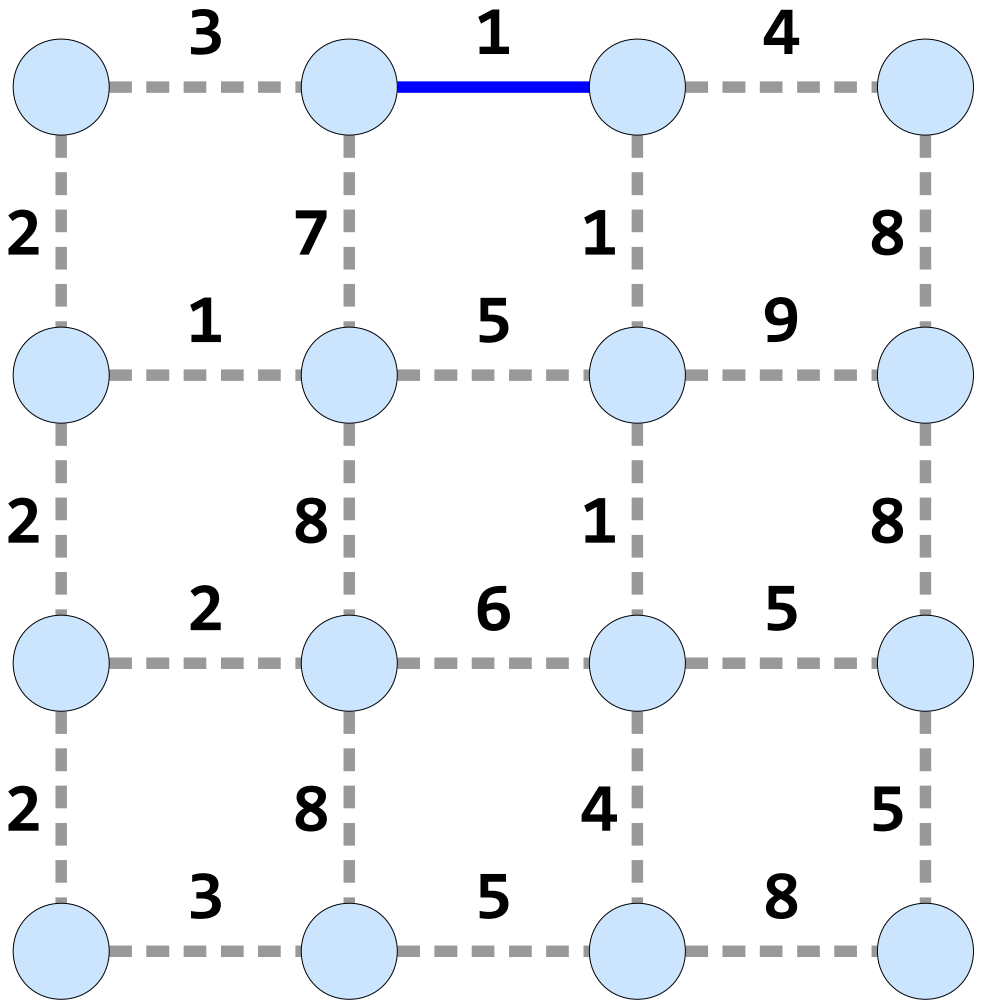
Mazes with Kruskal's Algorithm



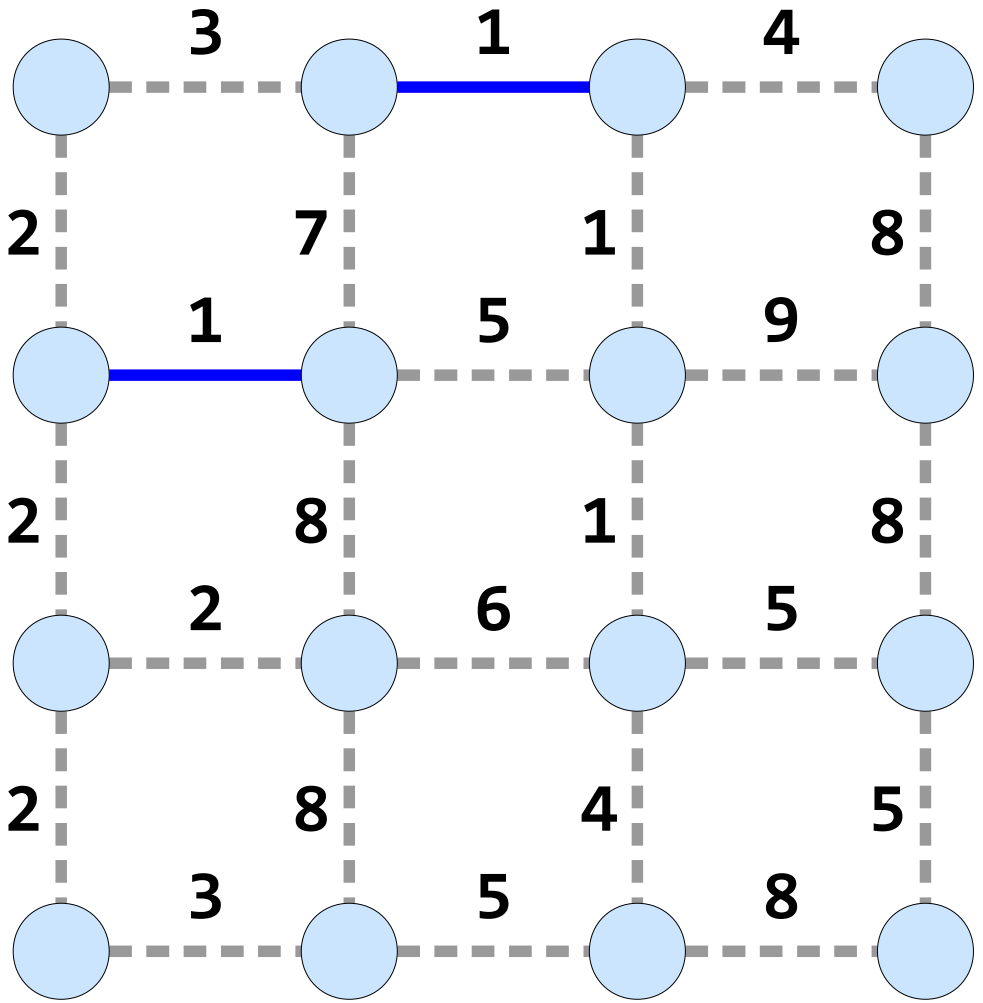
Mazes with Kruskal's Algorithm



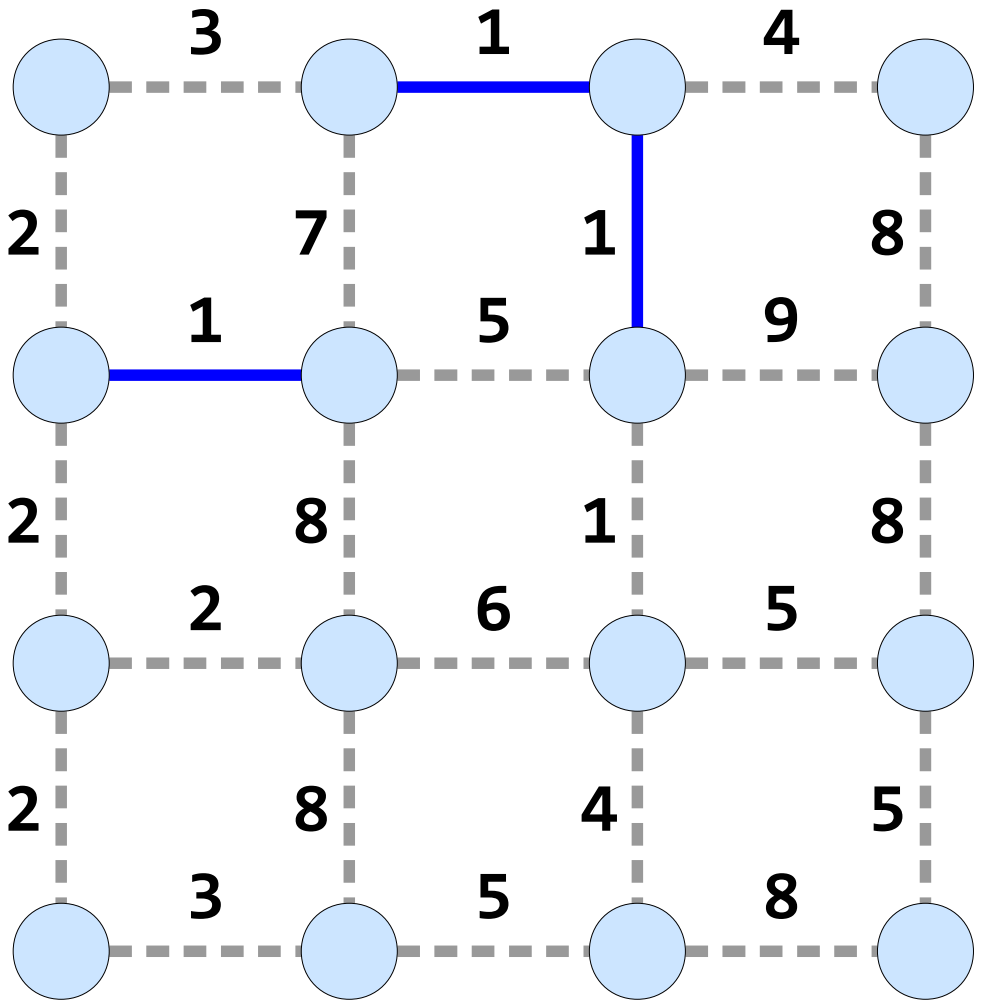
Mazes with Kruskal's Algorithm



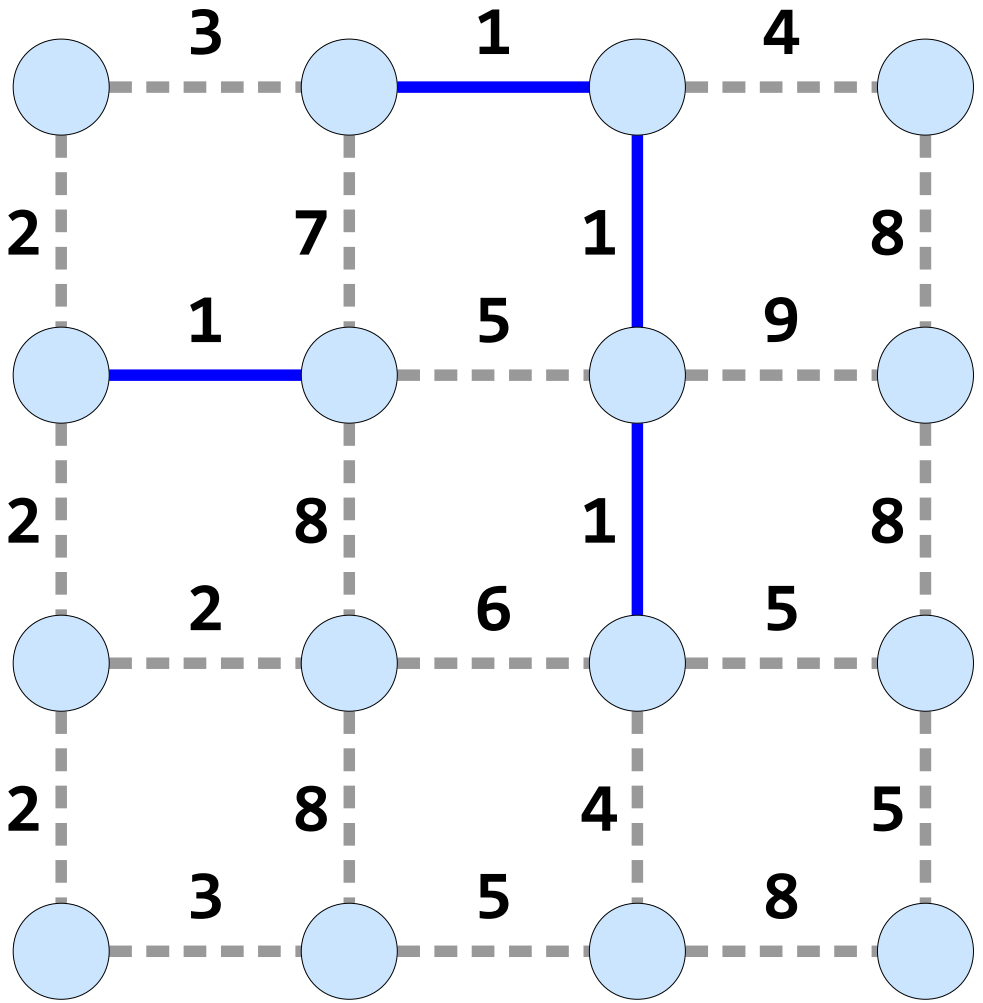
Mazes with Kruskal's Algorithm



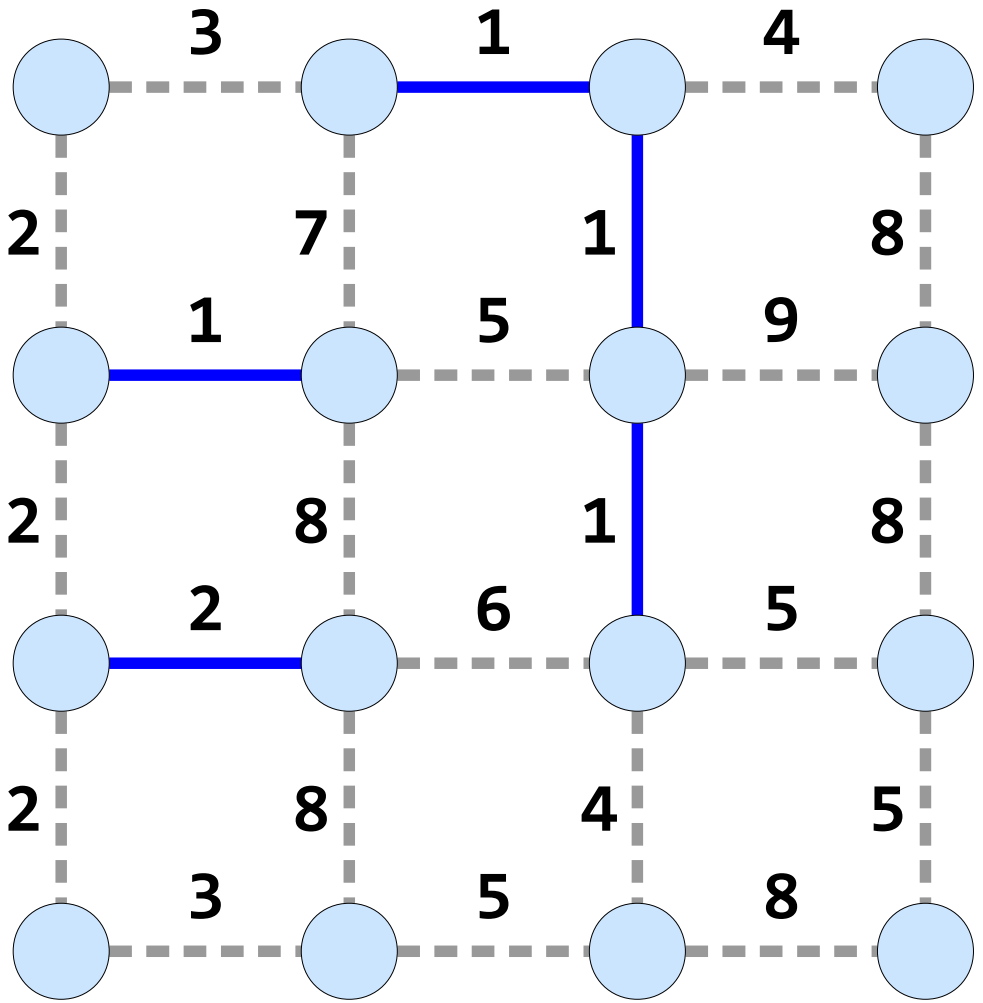
Mazes with Kruskal's Algorithm



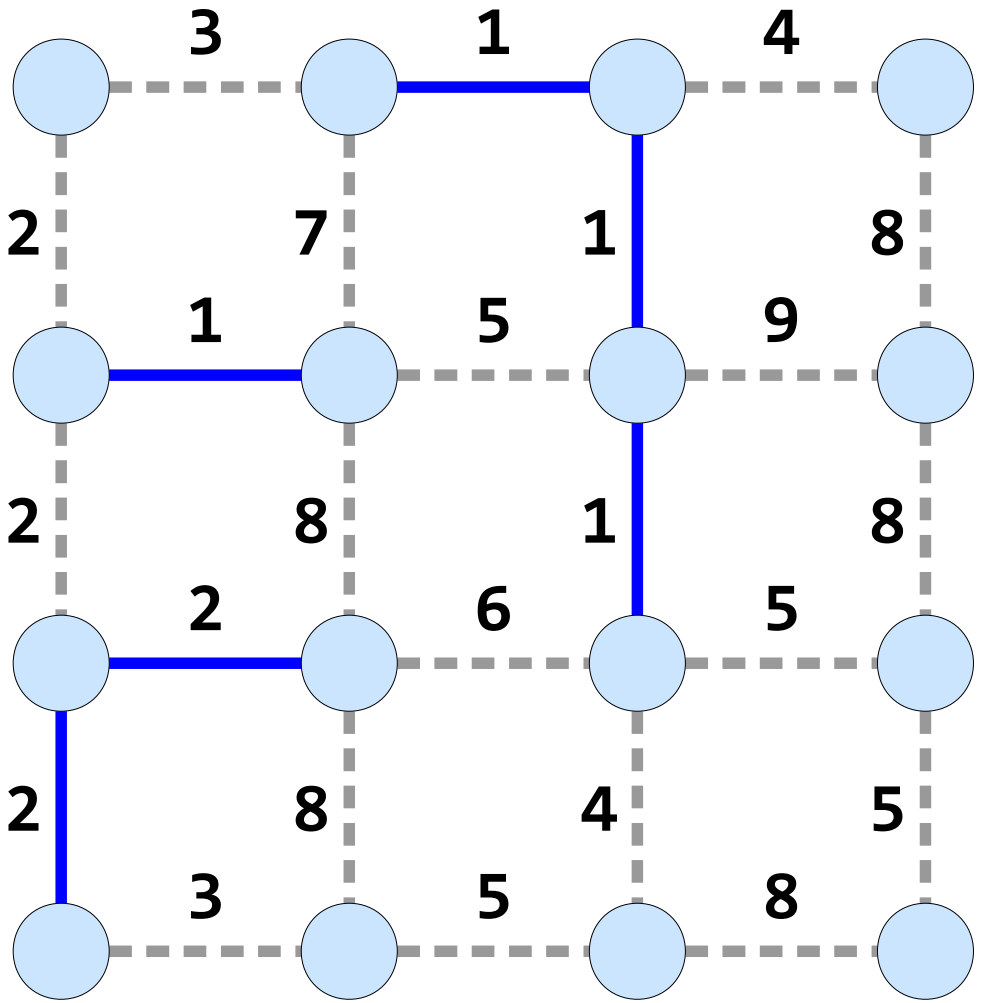
Mazes with Kruskal's Algorithm



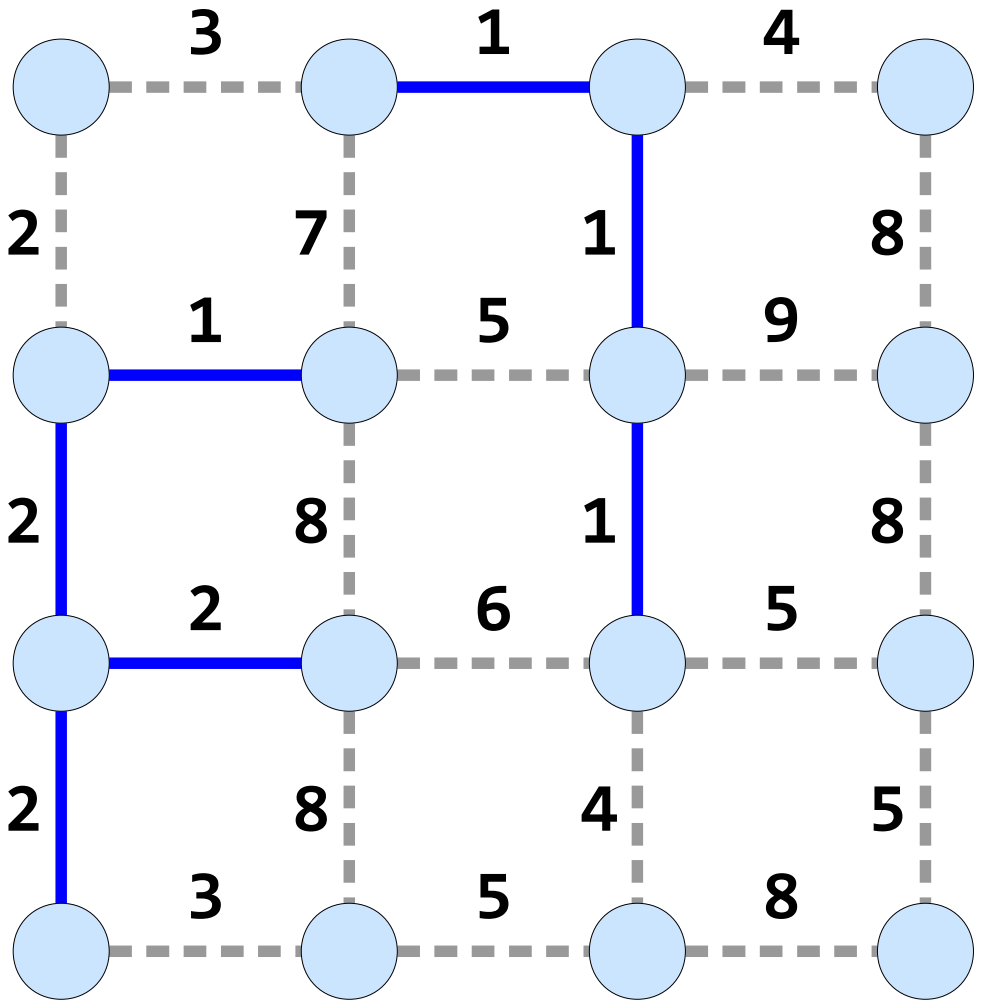
Mazes with Kruskal's Algorithm



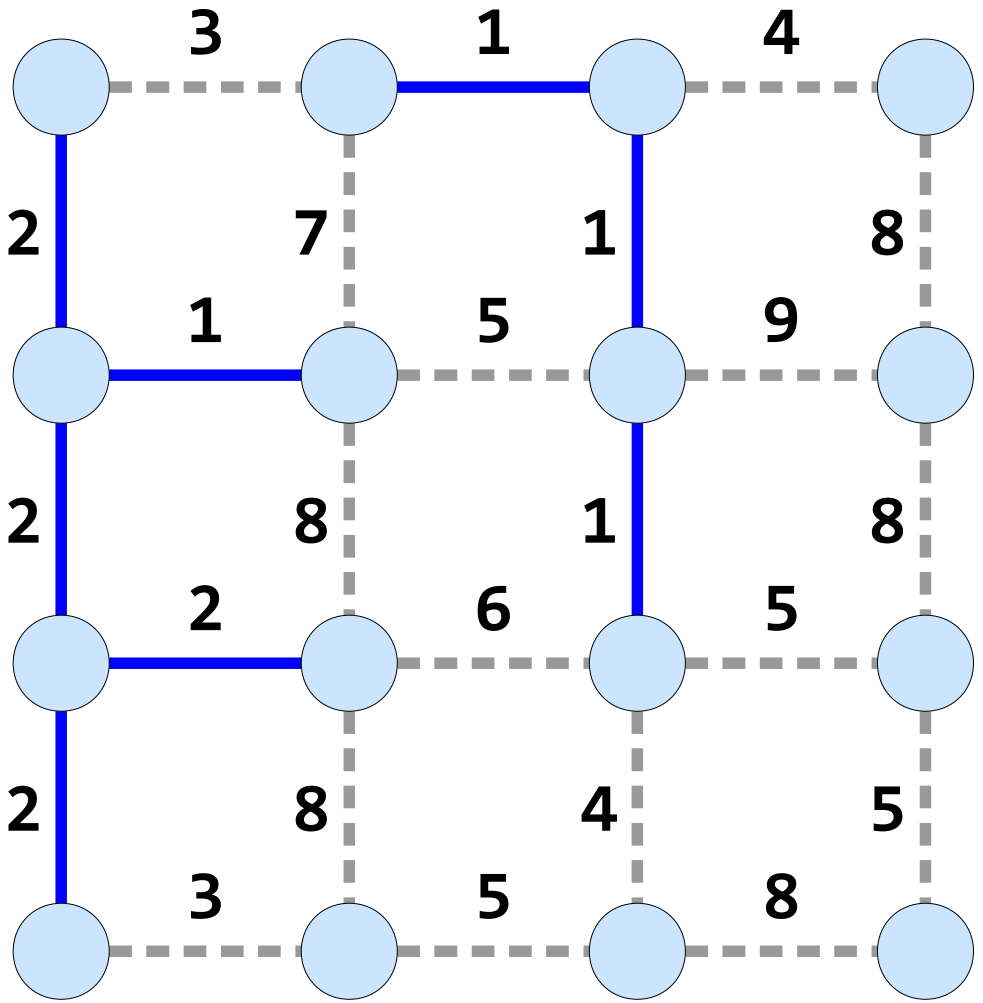
Mazes with Kruskal's Algorithm



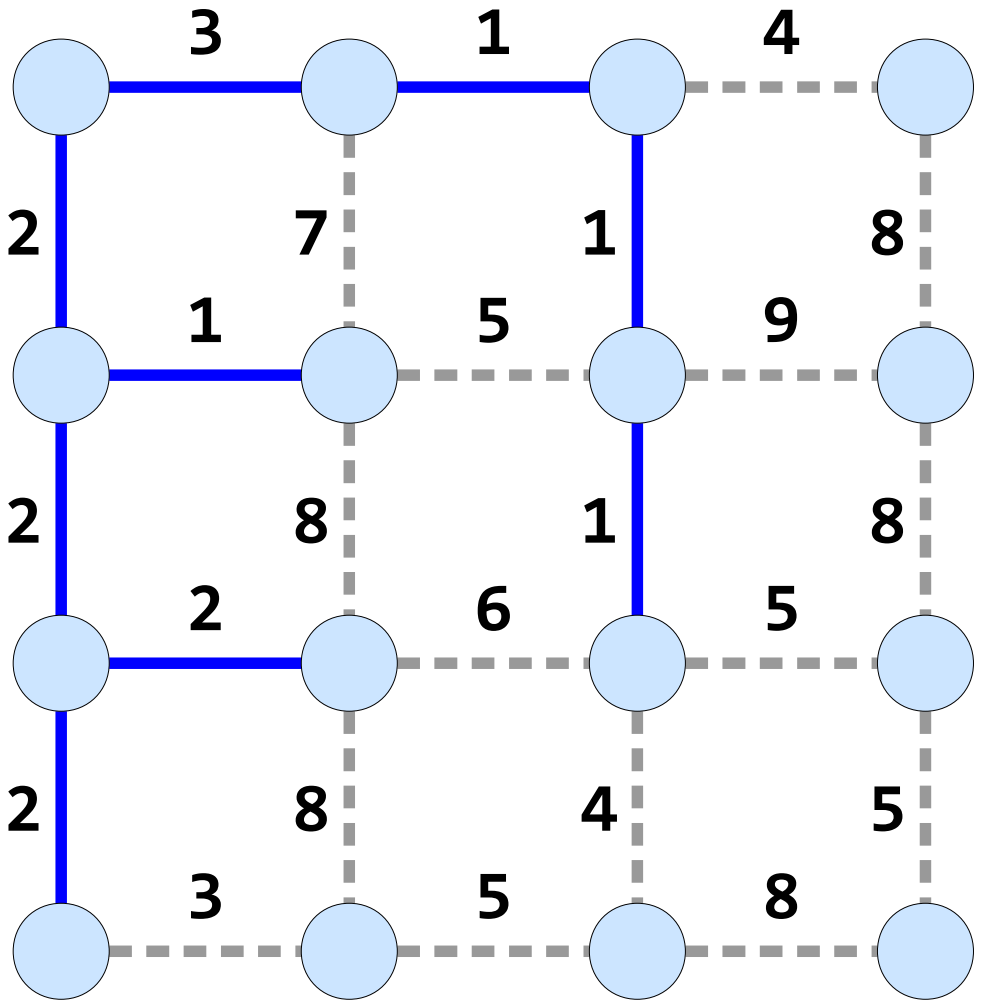
Mazes with Kruskal's Algorithm



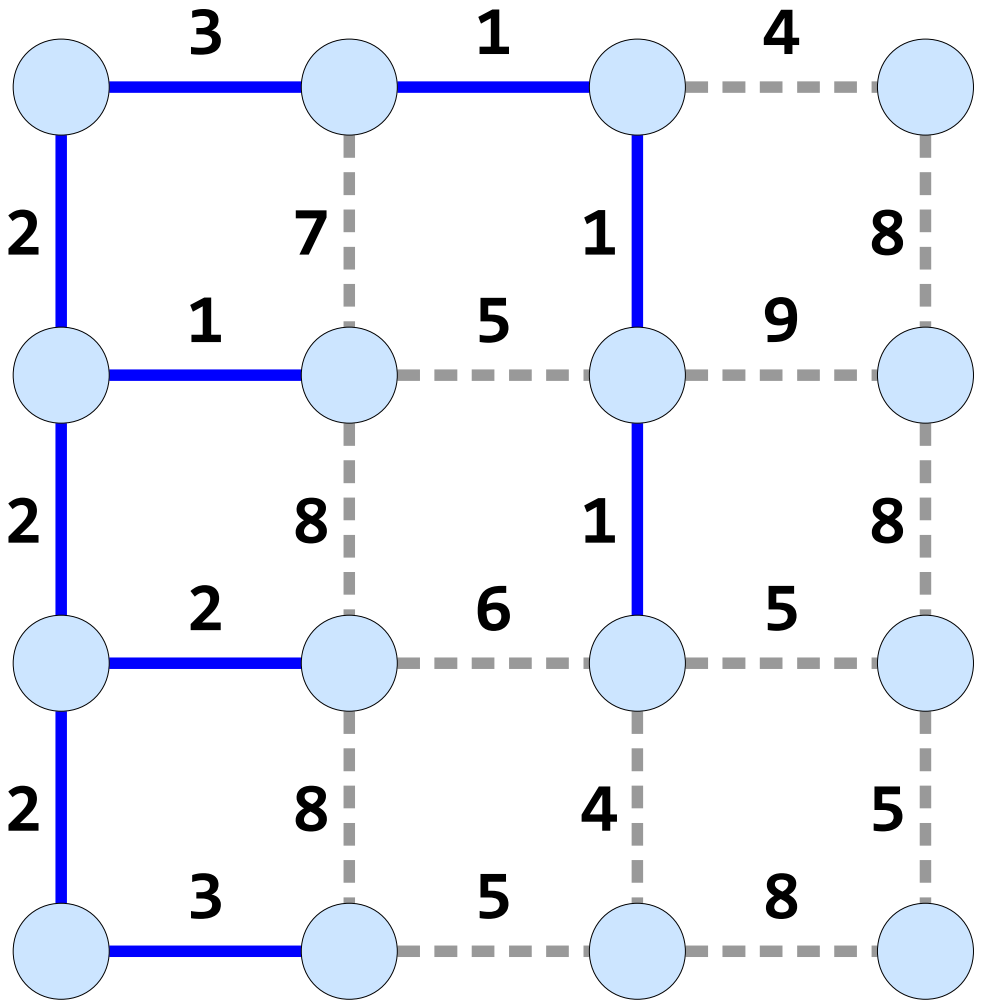
Mazes with Kruskal's Algorithm



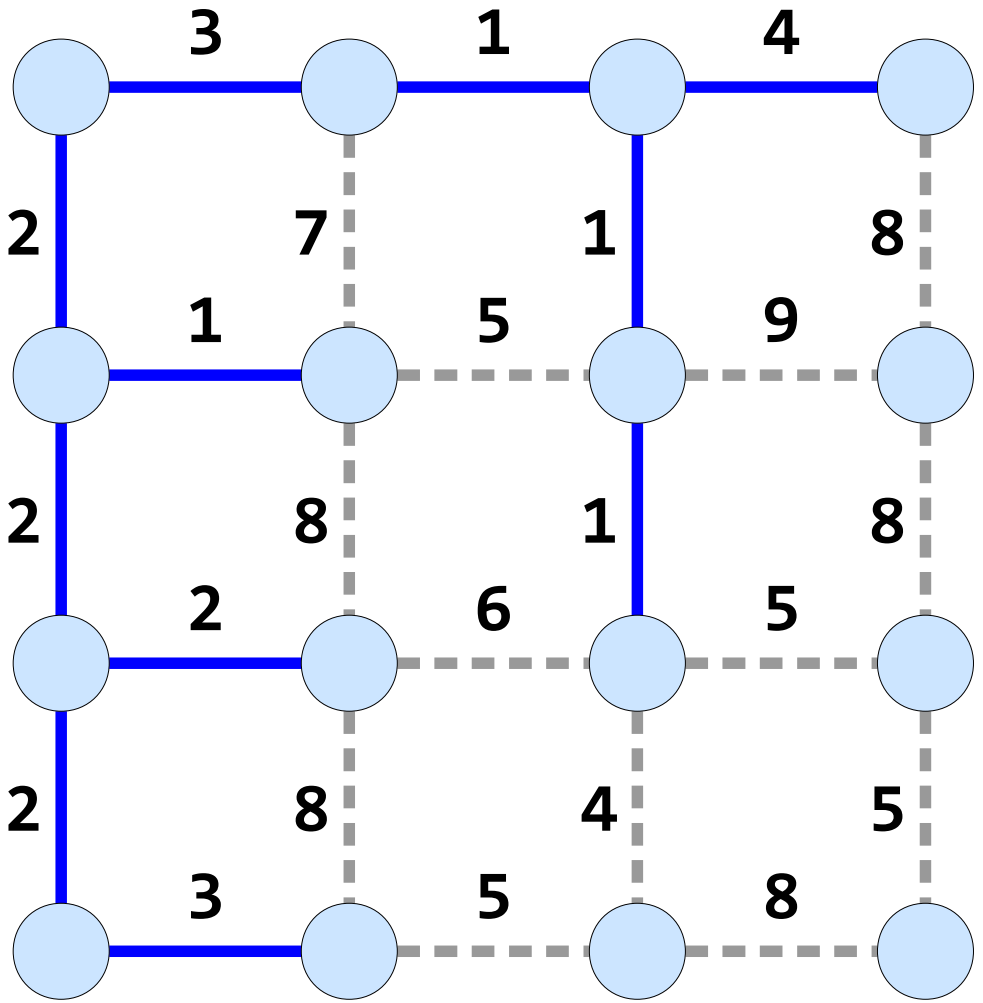
Mazes with Kruskal's Algorithm



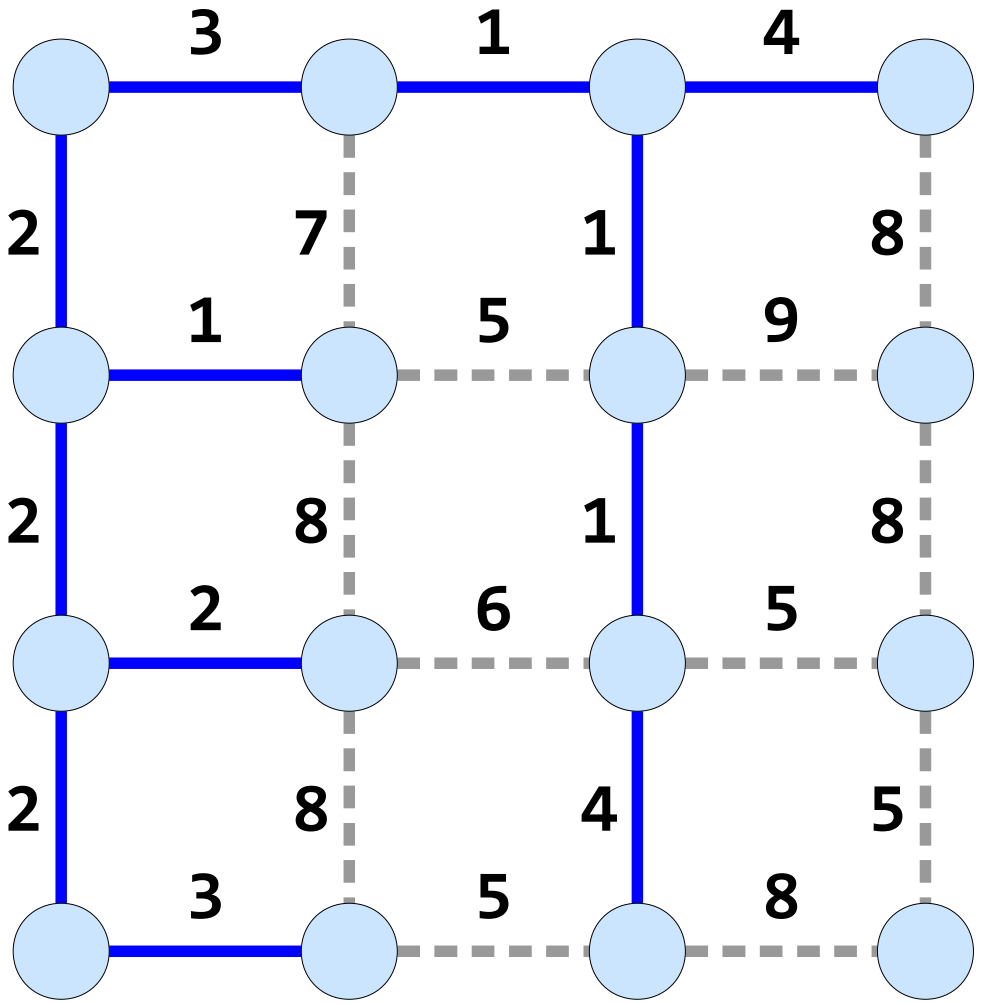
Mazes with Kruskal's Algorithm



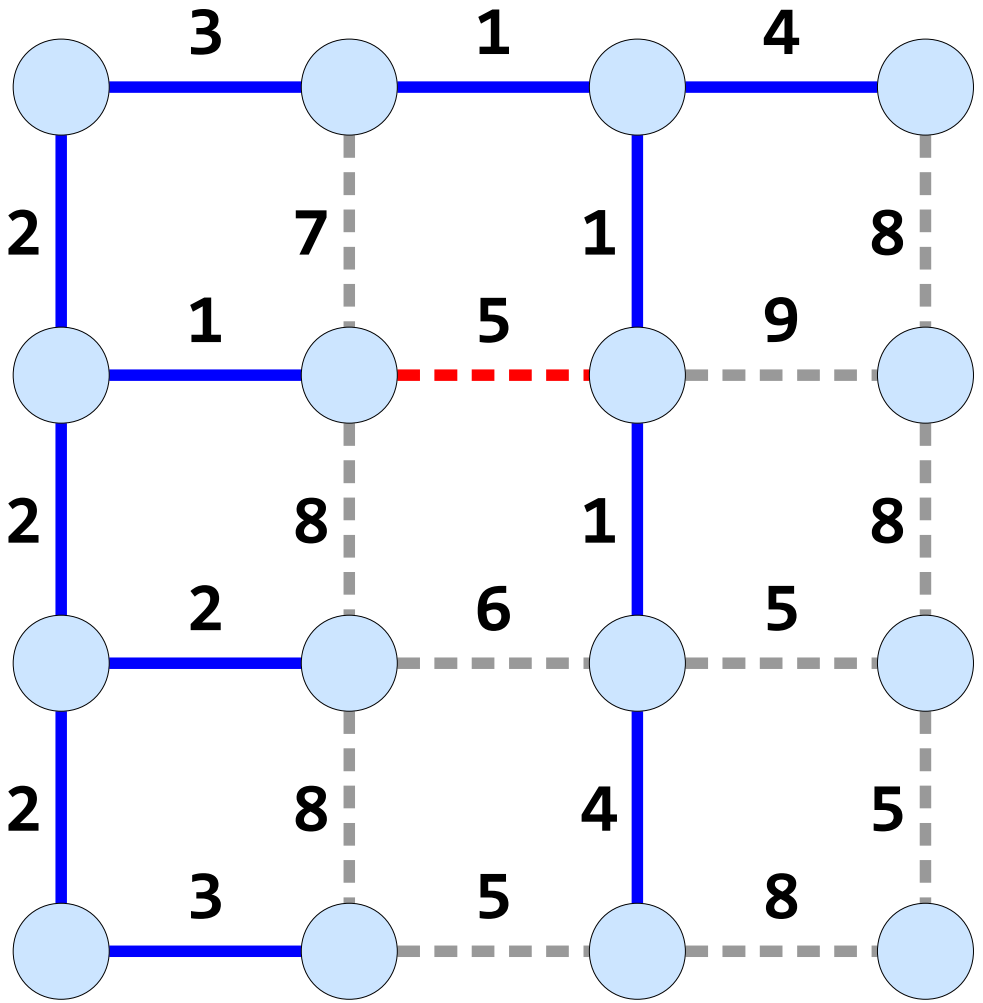
Mazes with Kruskal's Algorithm



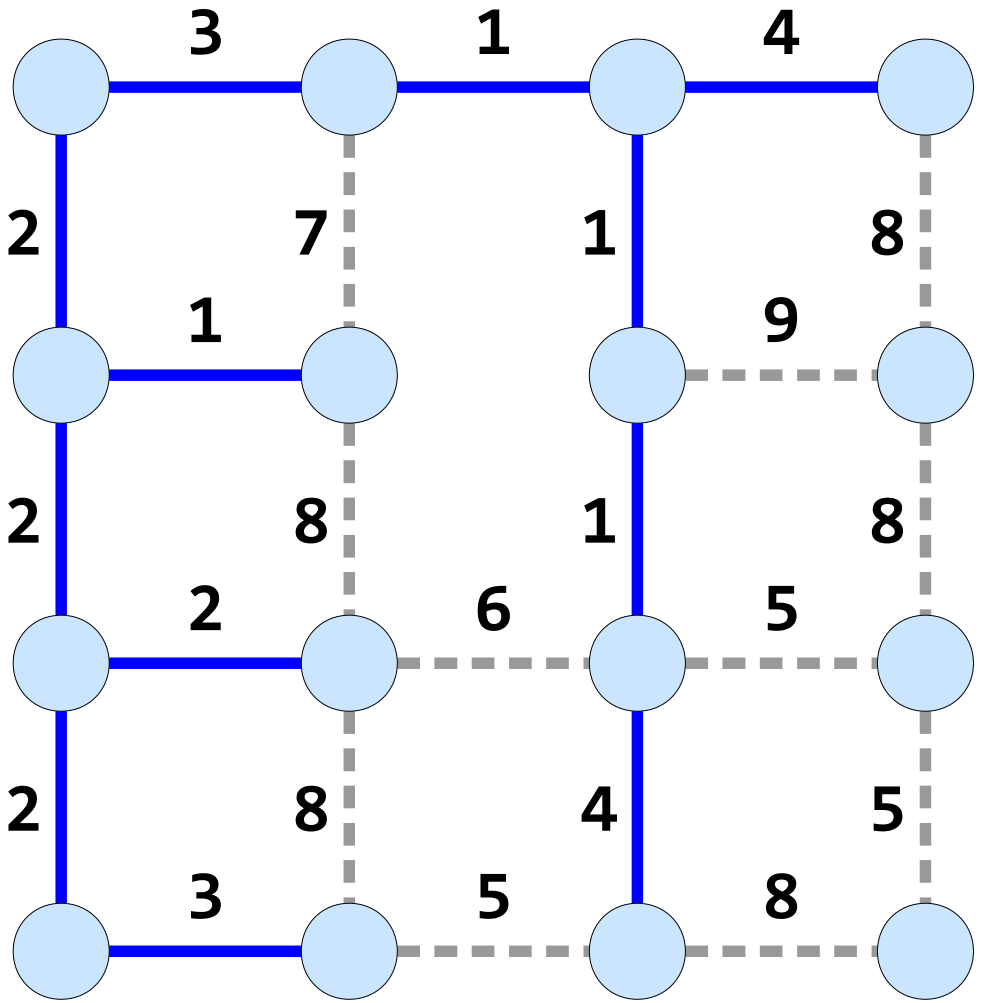
Mazes with Kruskal's Algorithm



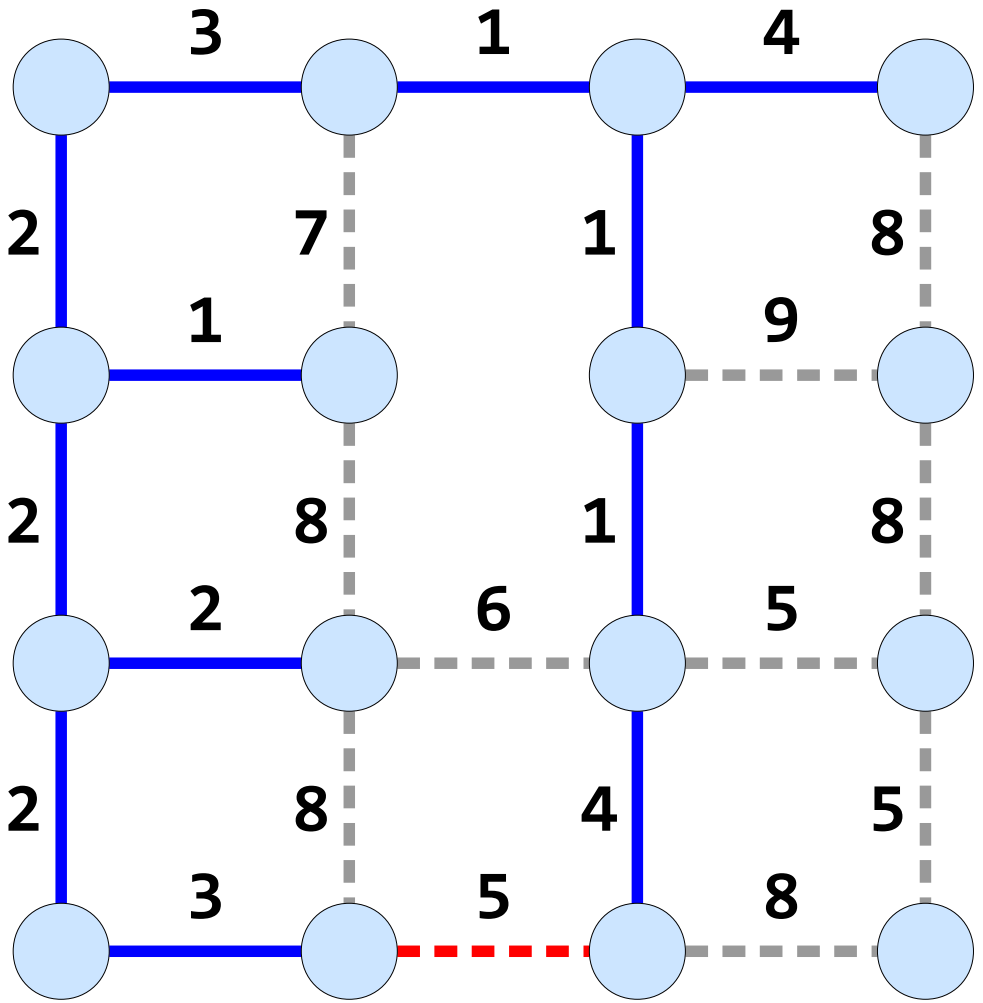
Mazes with Kruskal's Algorithm



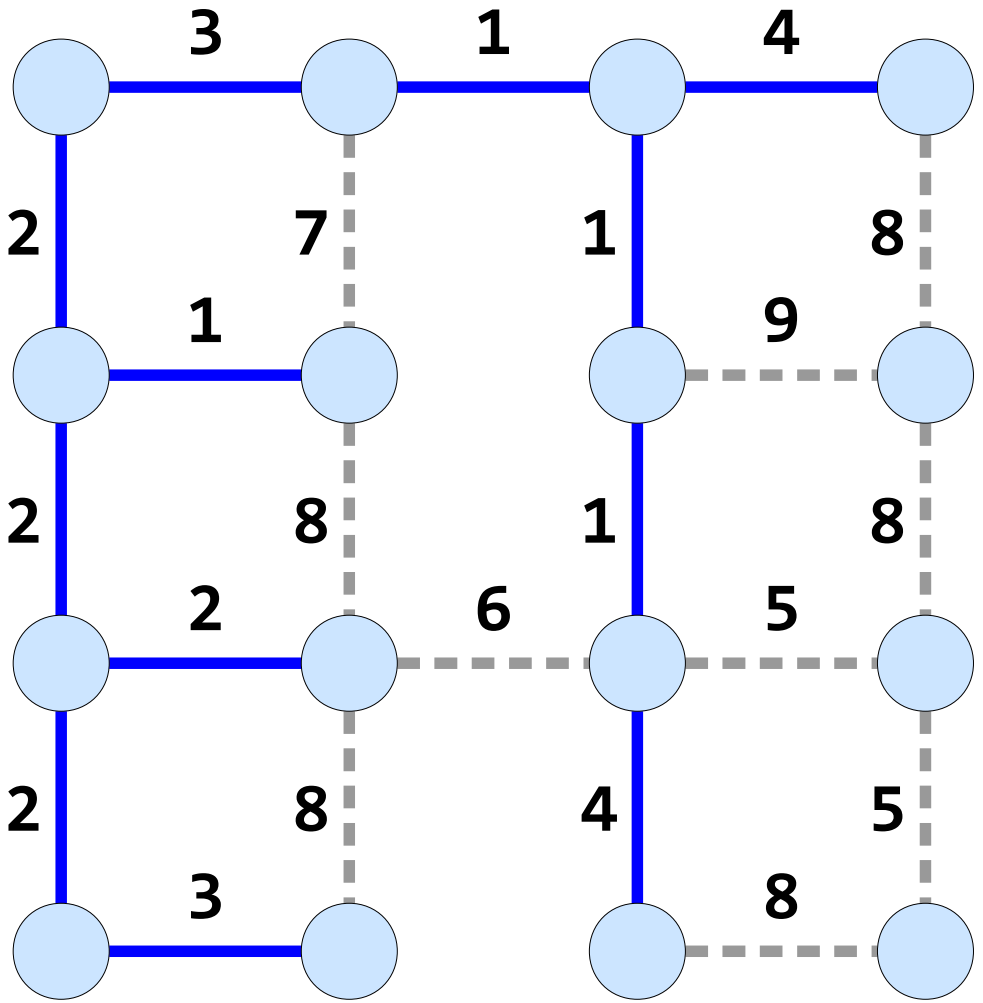
Mazes with Kruskal's Algorithm



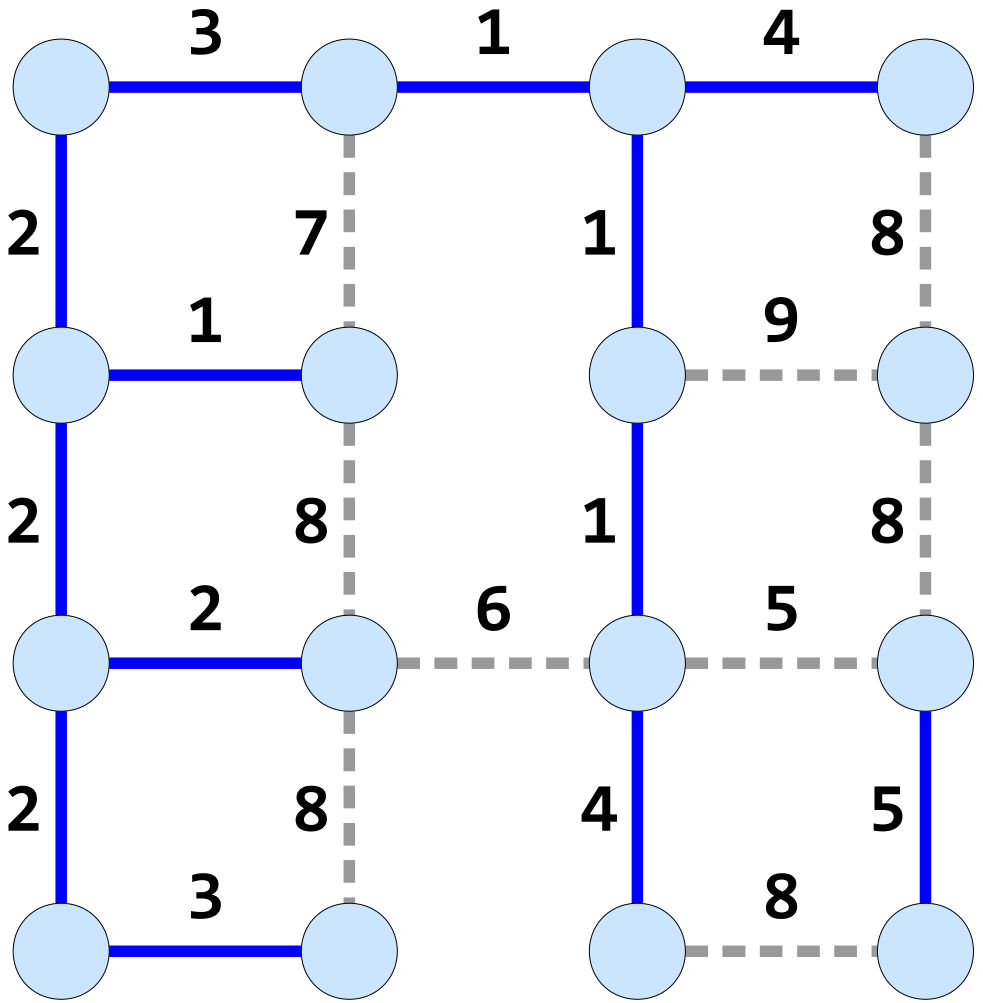
Mazes with Kruskal's Algorithm



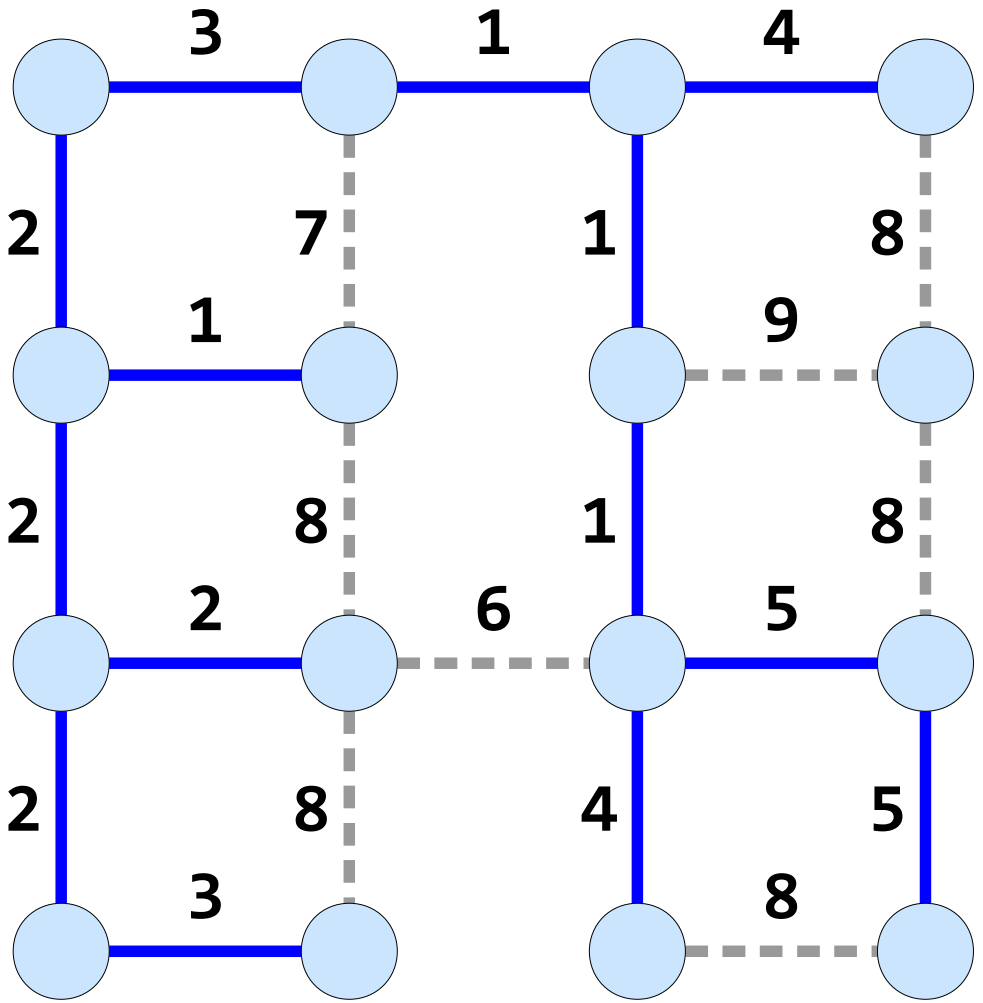
Mazes with Kruskal's Algorithm



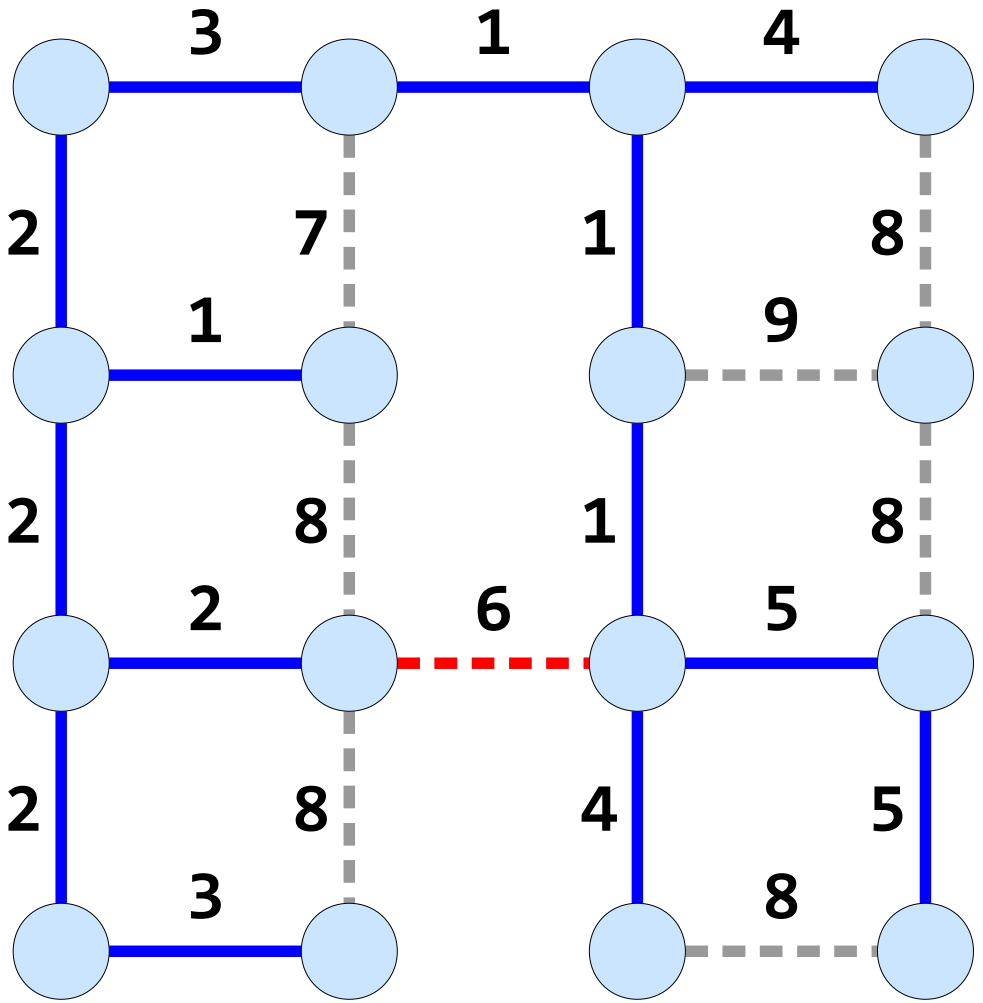
Mazes with Kruskal's Algorithm



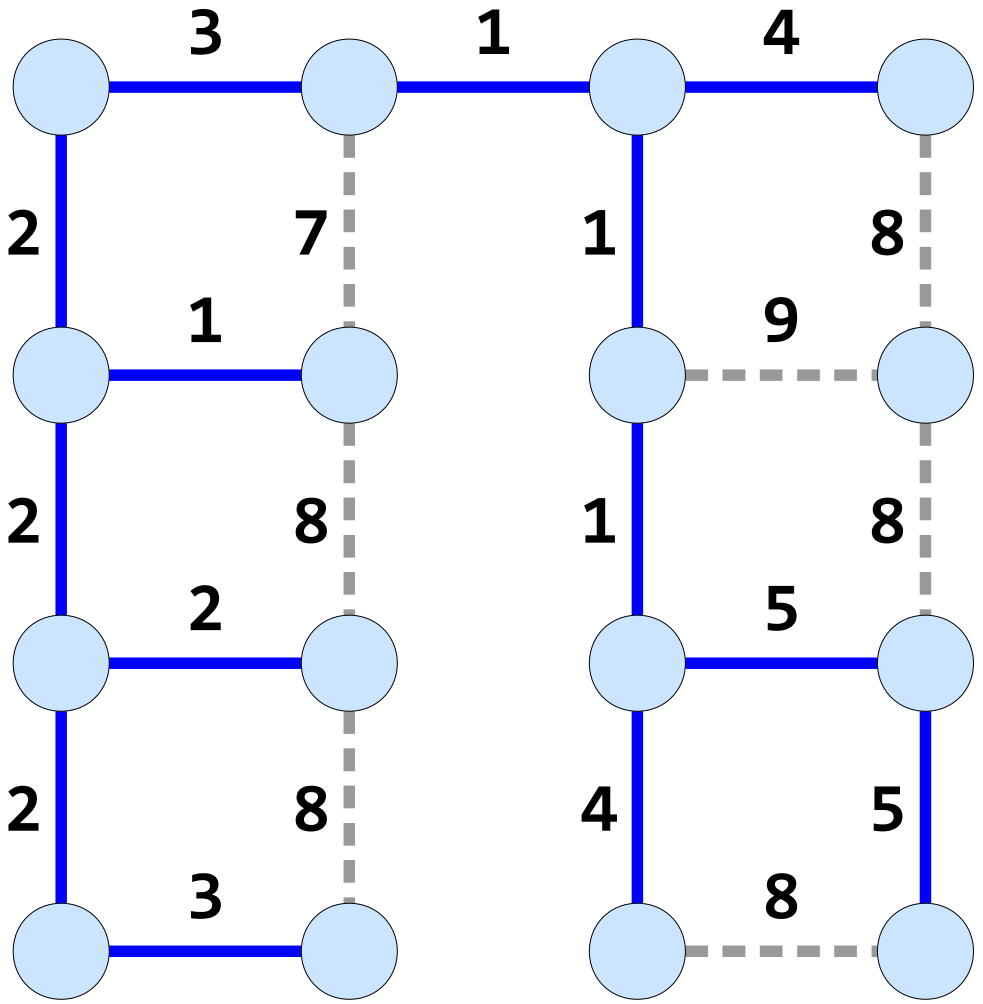
Mazes with Kruskal's Algorithm



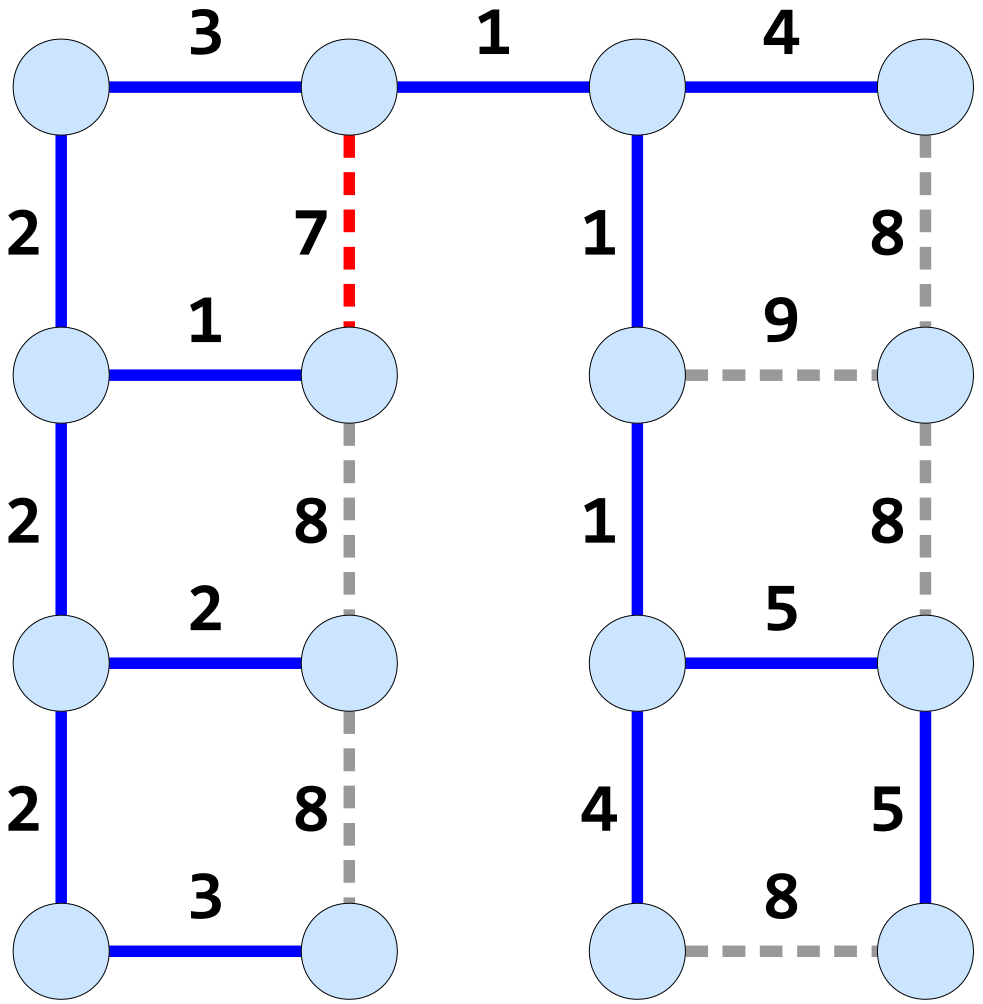
Mazes with Kruskal's Algorithm



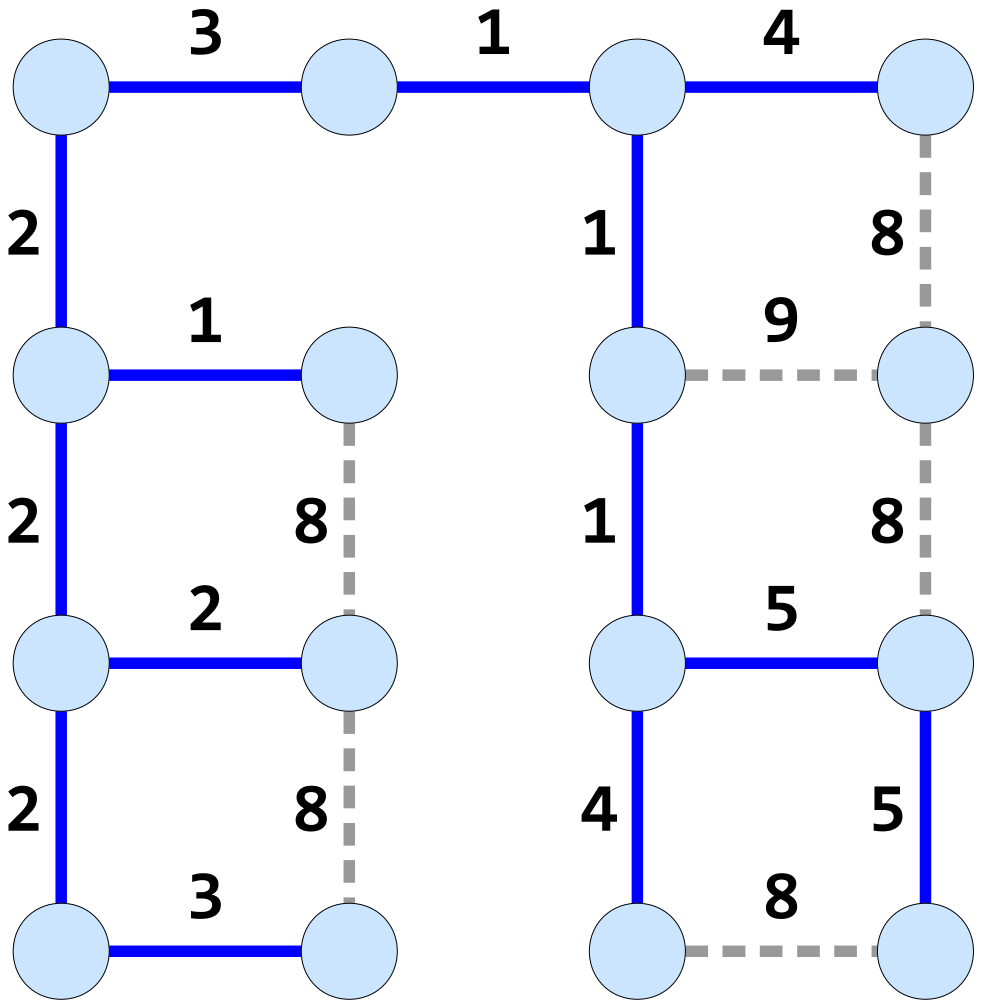
Mazes with Kruskal's Algorithm



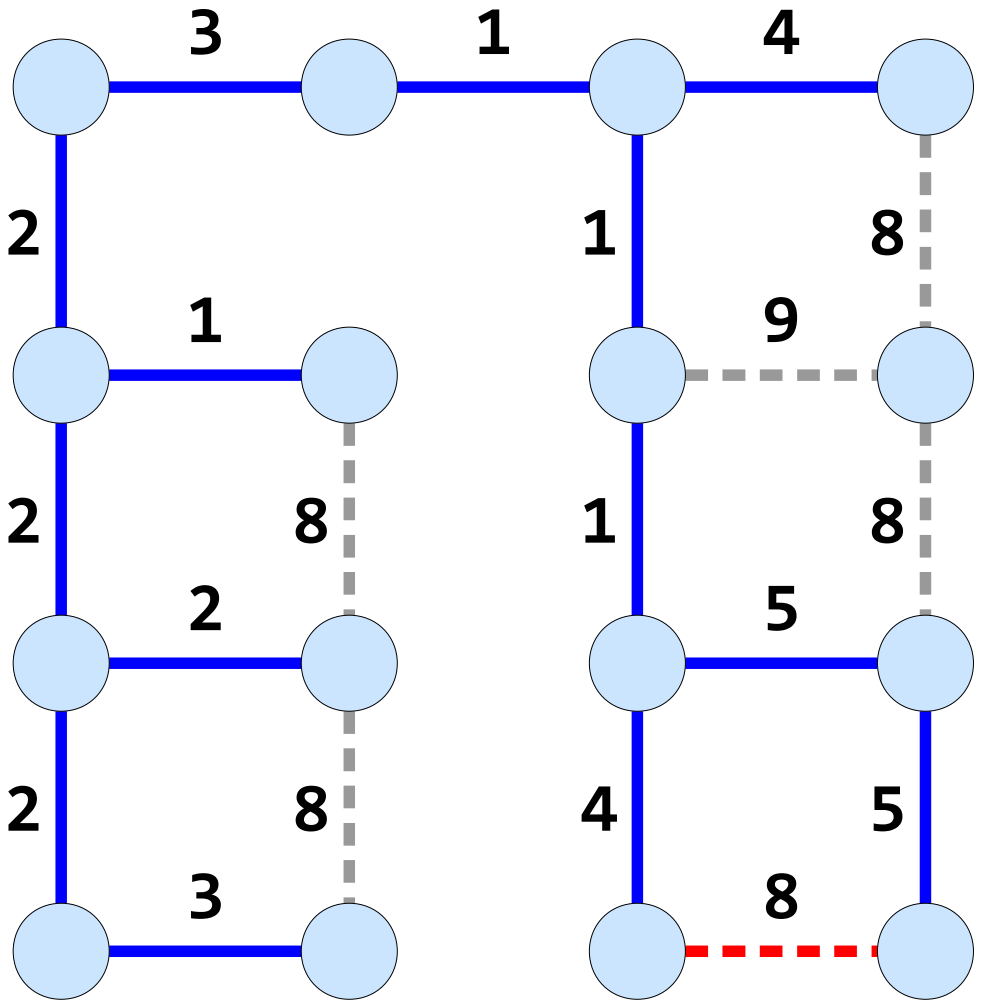
Mazes with Kruskal's Algorithm



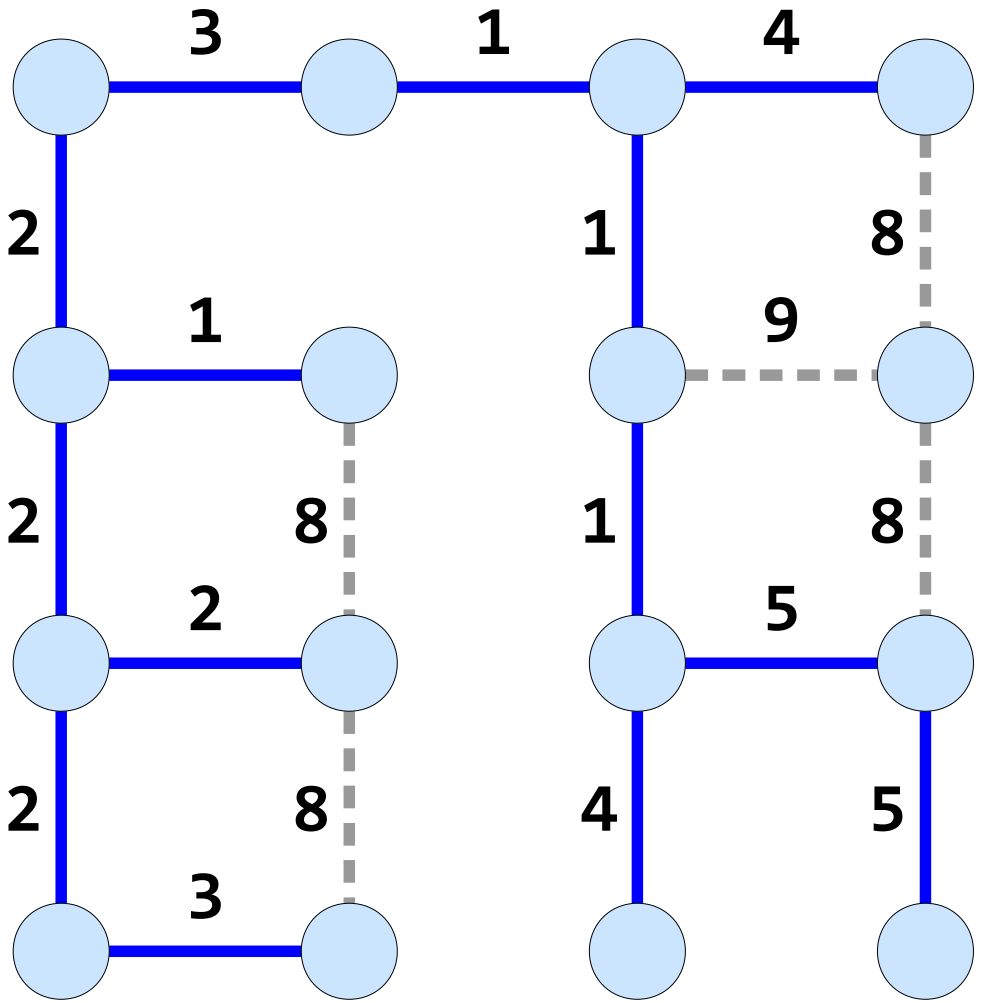
Mazes with Kruskal's Algorithm



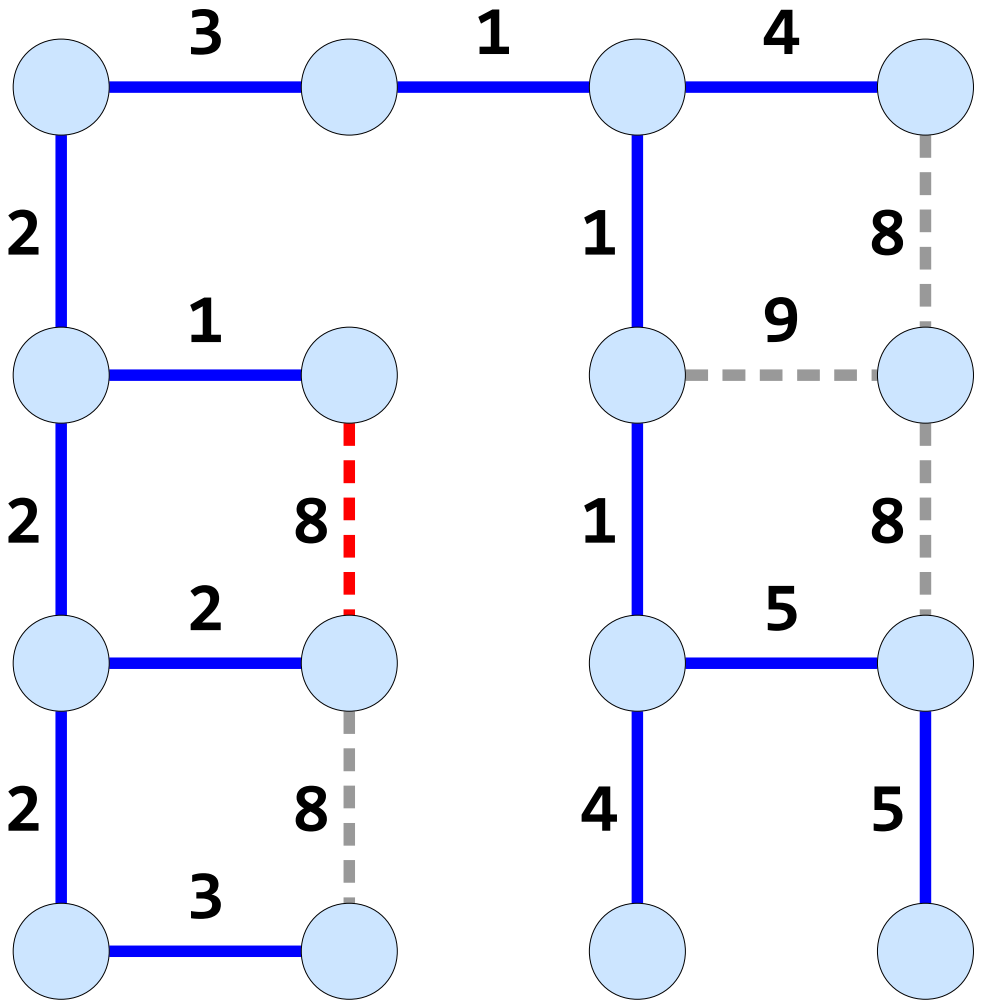
Mazes with Kruskal's Algorithm



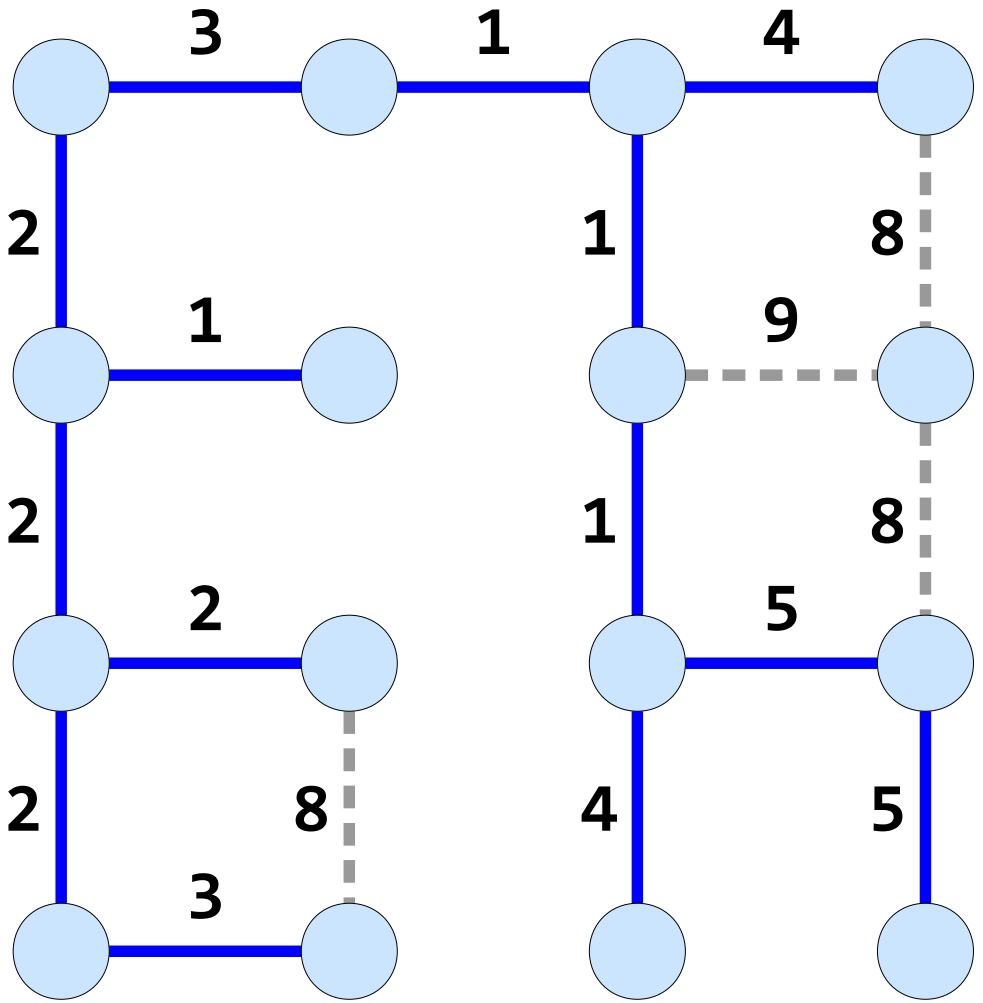
Mazes with Kruskal's Algorithm



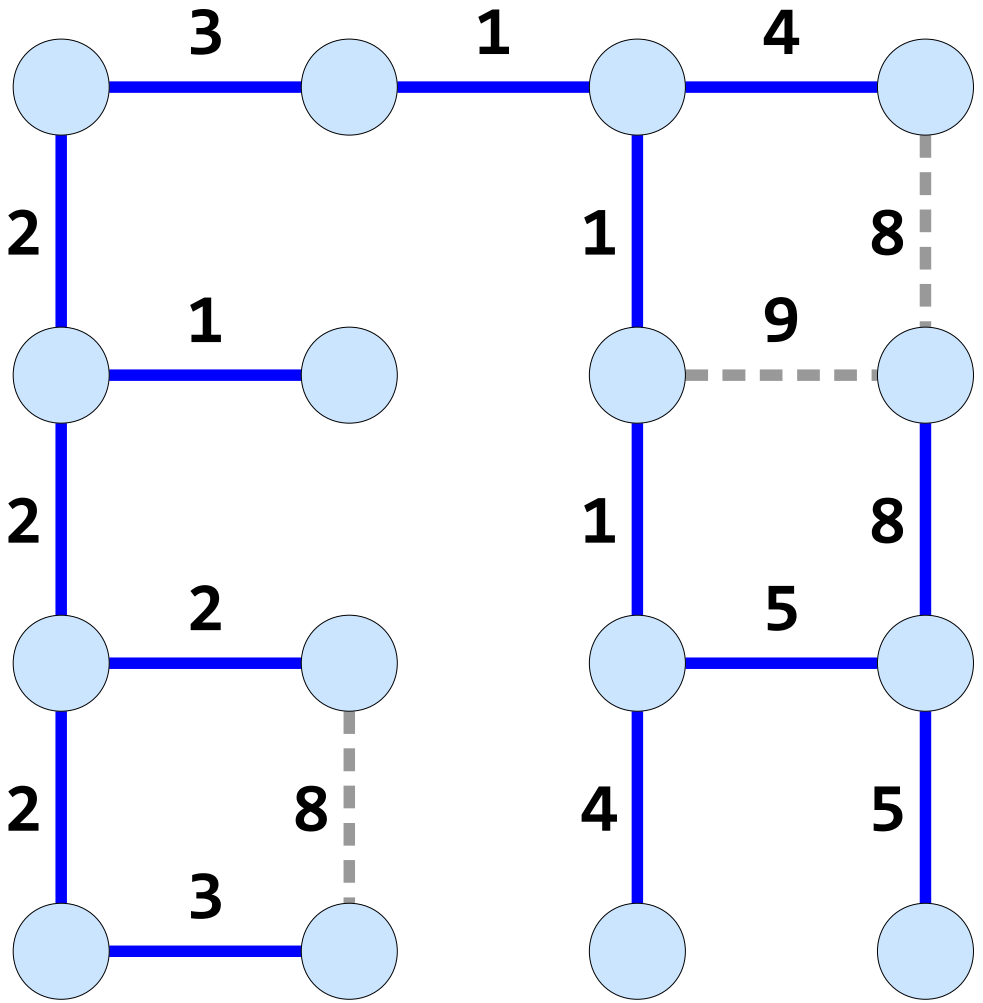
Mazes with Kruskal's Algorithm



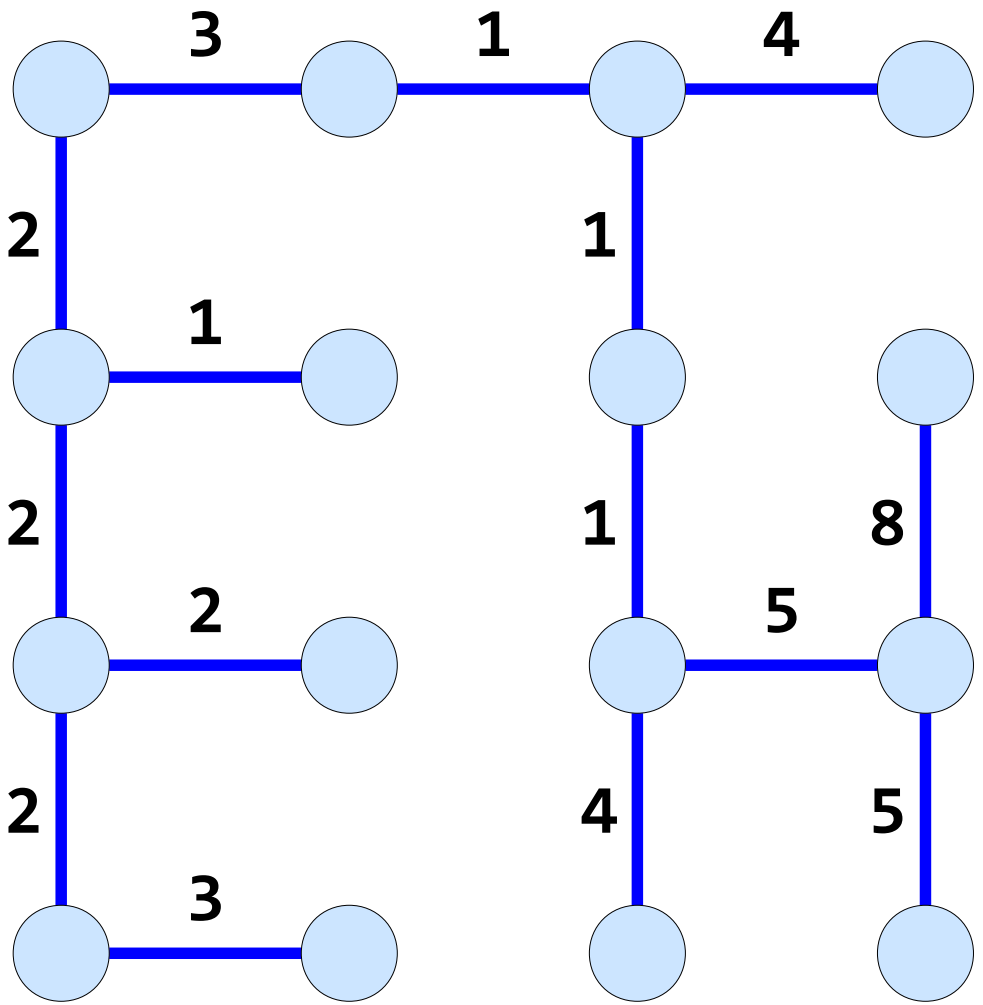
Mazes with Kruskal's Algorithm



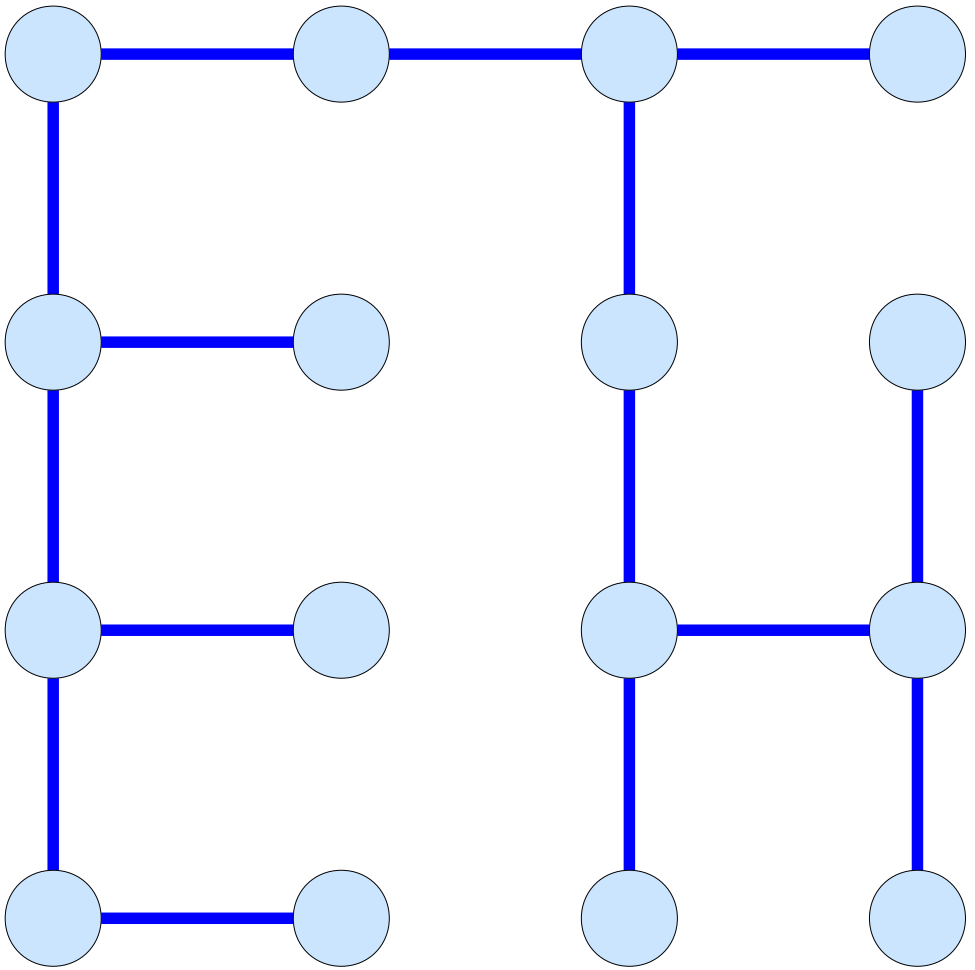
Mazes with Kruskal's Algorithm



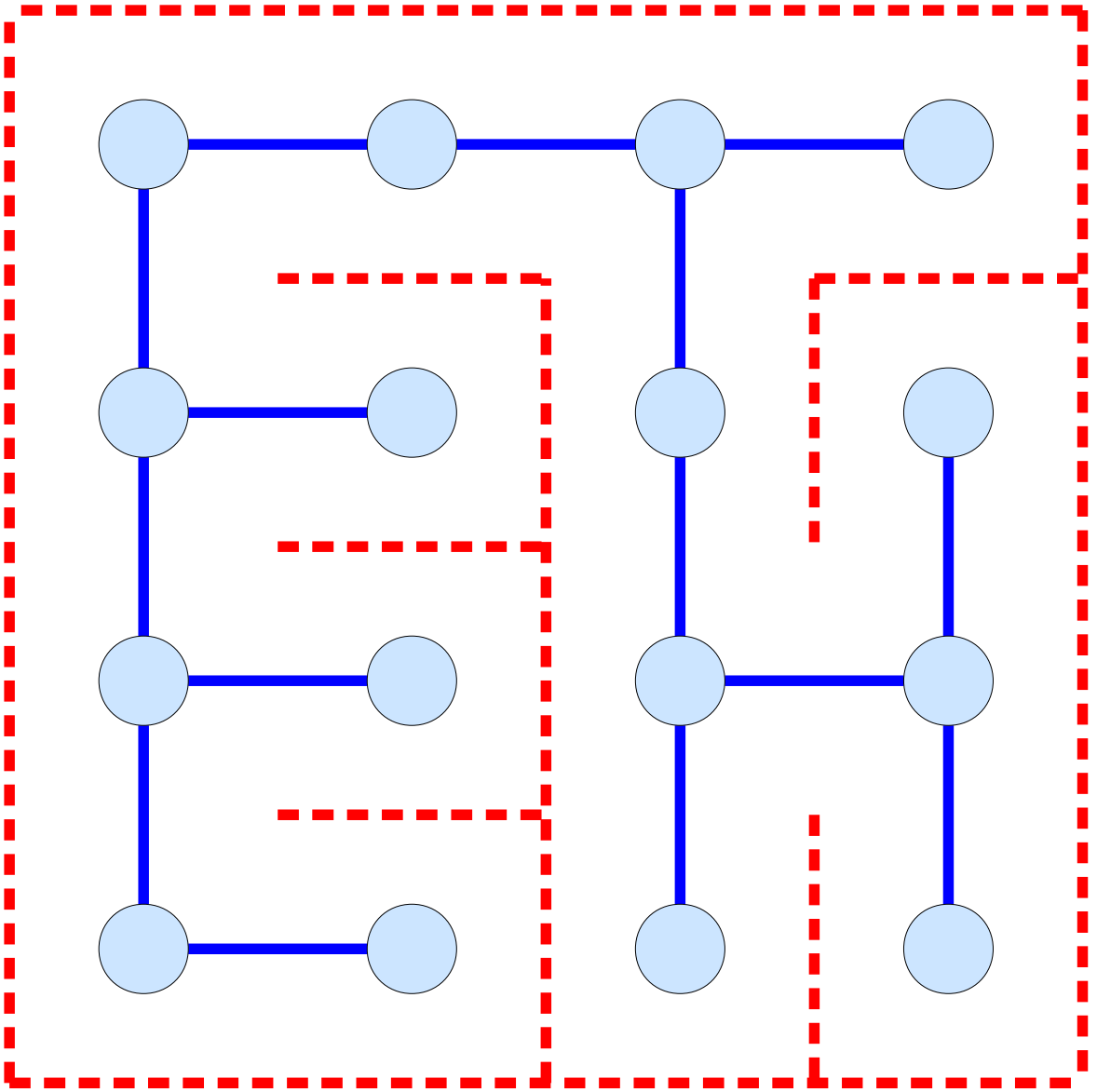
Mazes with Kruskal's Algorithm



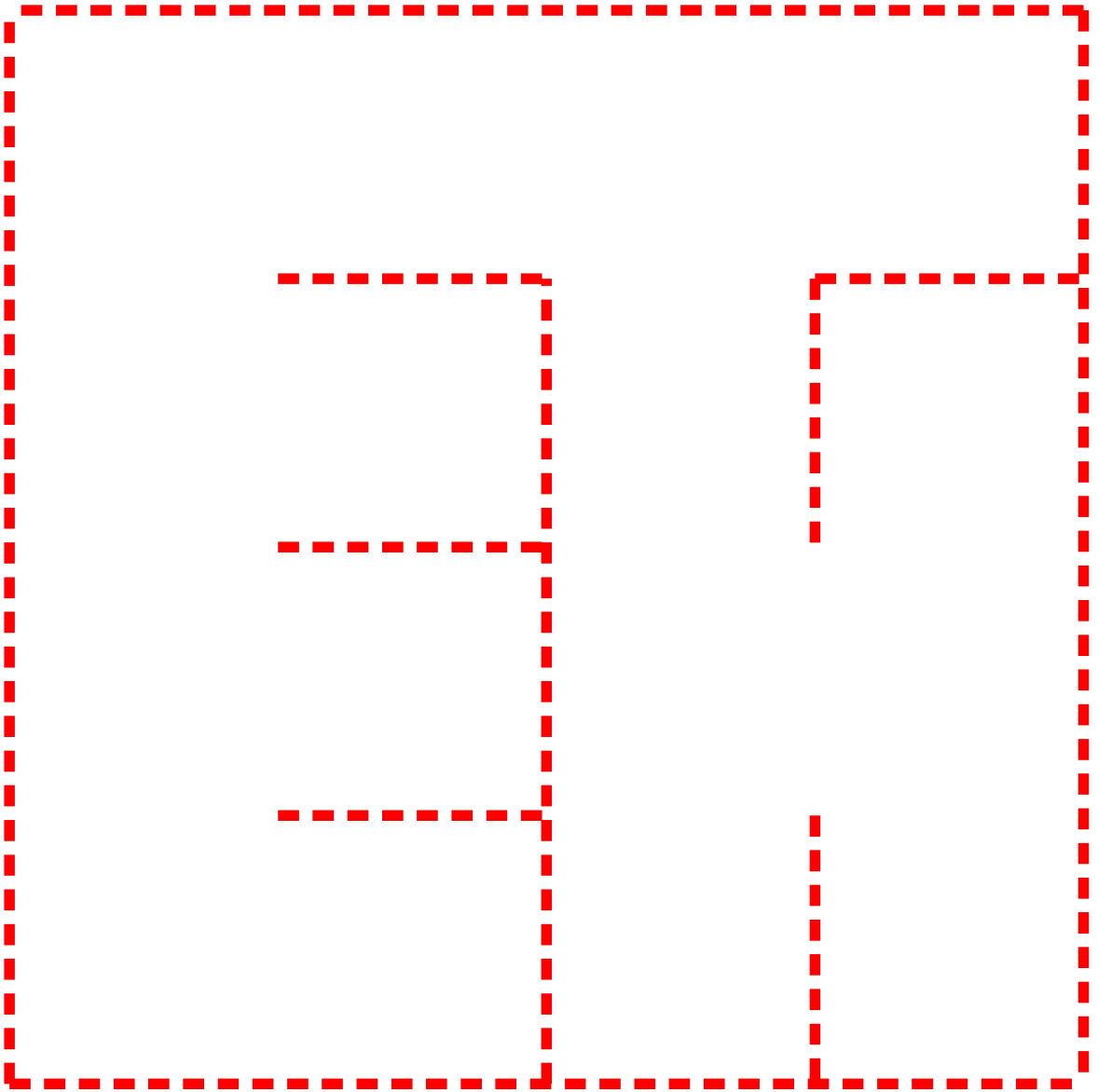
Mazes with Kruskal's Algorithm



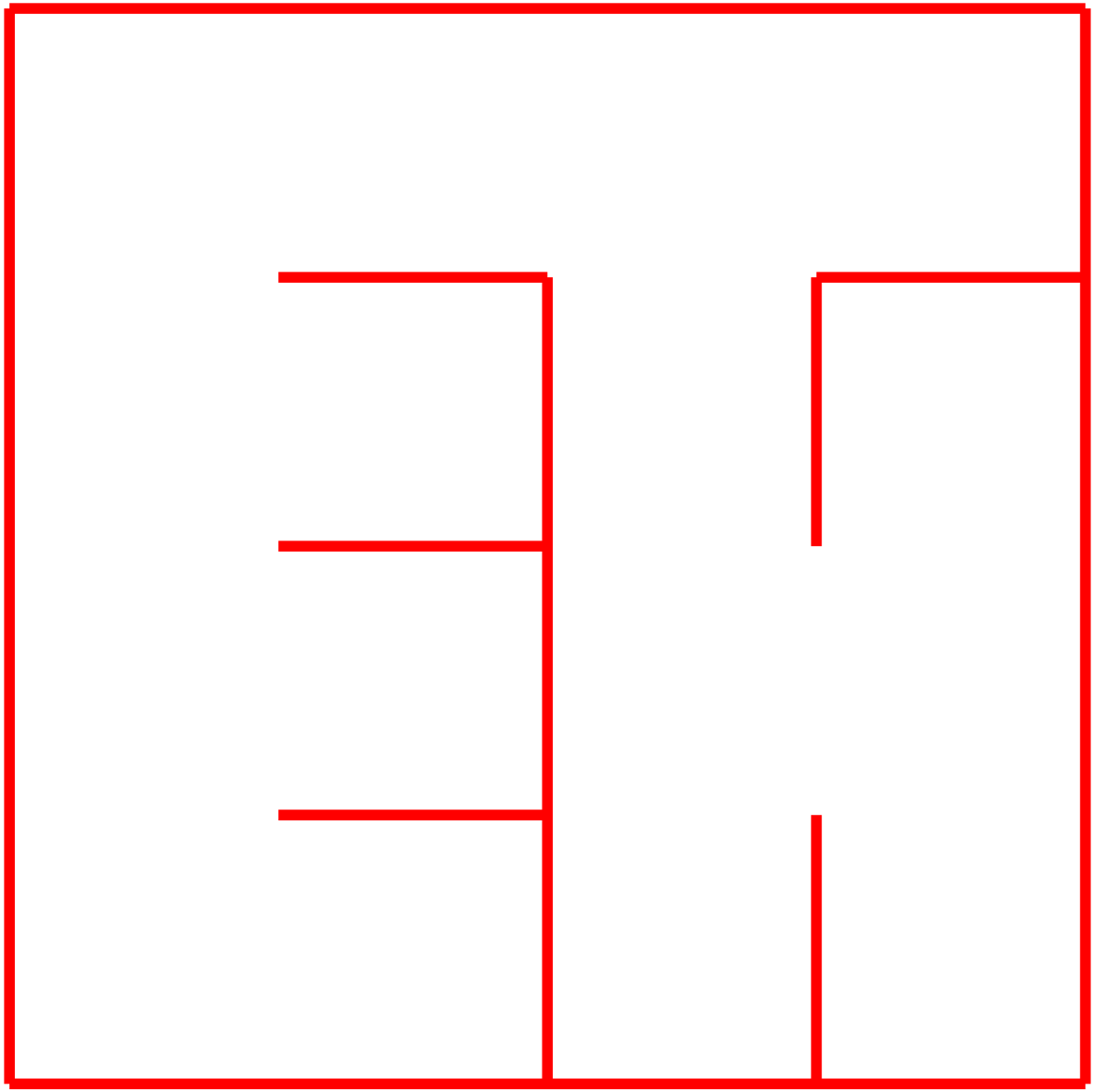
Mazes with Kruskal's Algorithm



Mazes with Kruskal's Algorithm



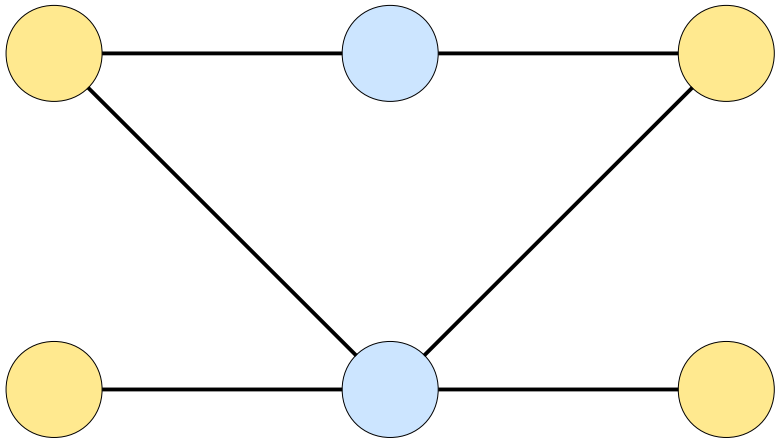
Mazes with Kruskal's Algorithm

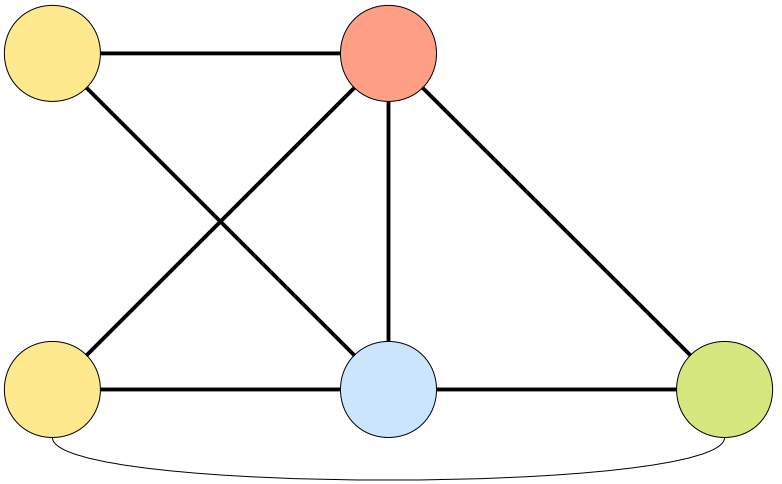


Other Cool Graph Problems

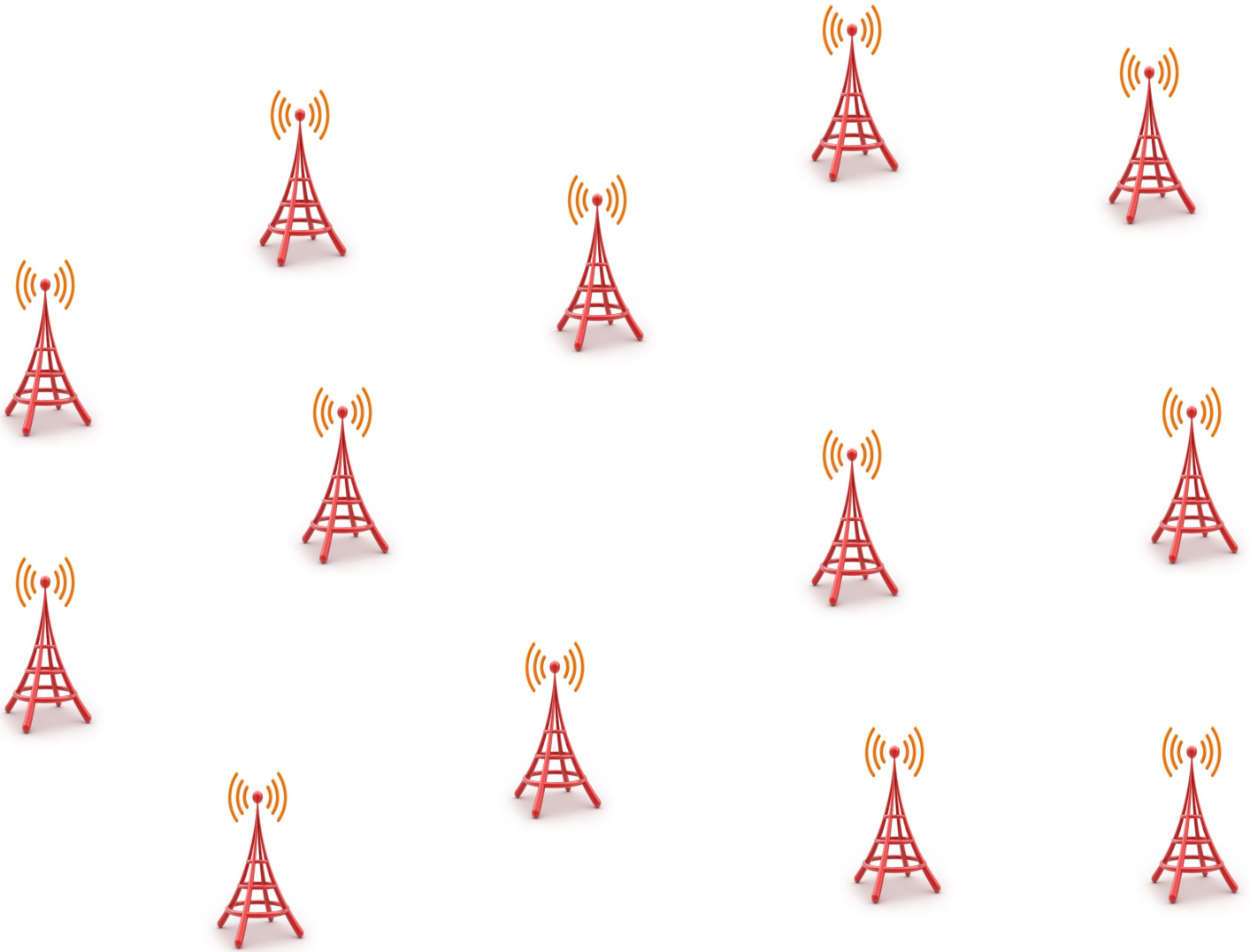
Graph Coloring

- Given a graph G, assign **colors** to the nodes so that no edge has endpoints of the same color.
- The **chromatic number** of a graph is the fewest number of colors needed to color it.

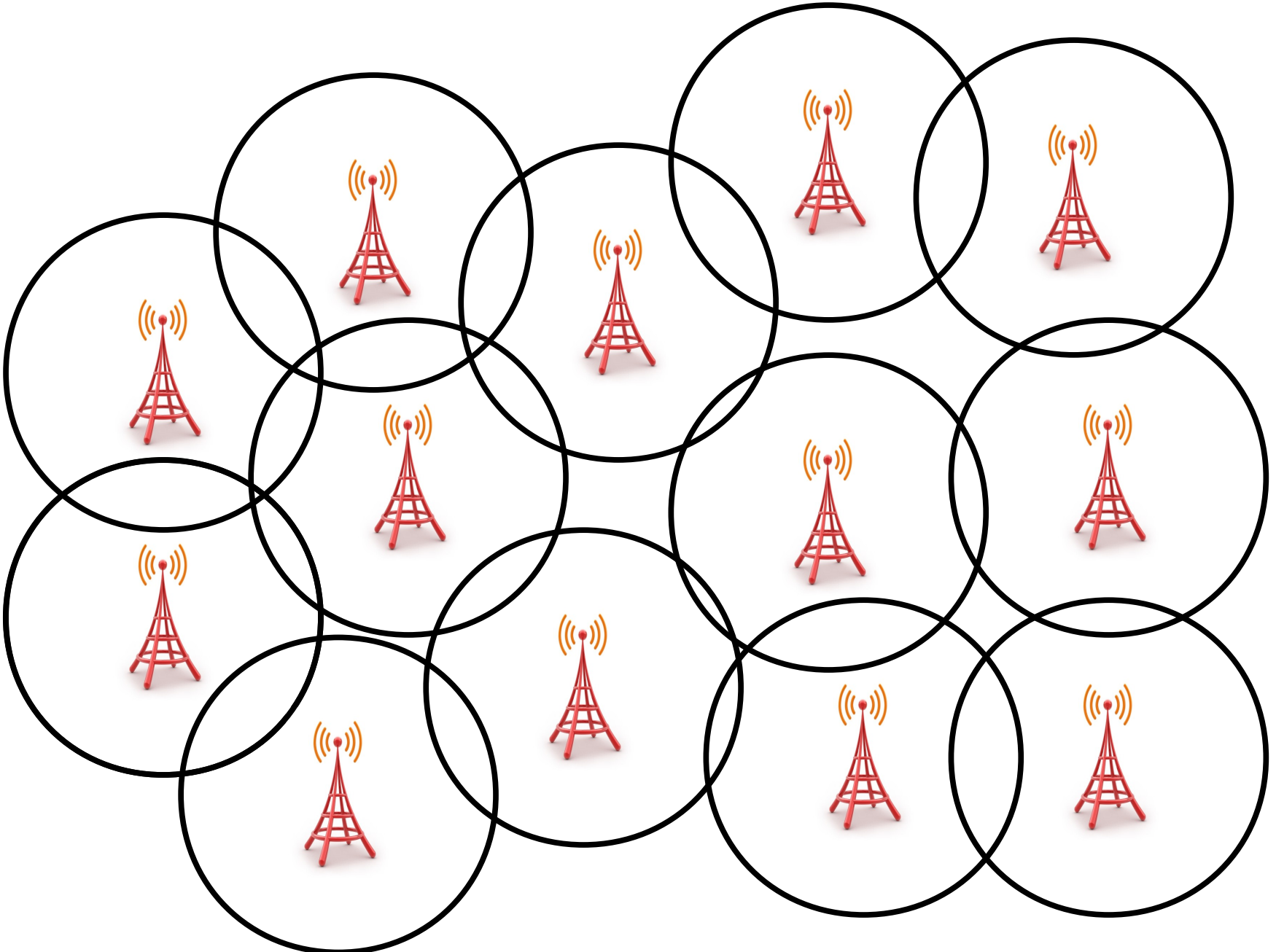




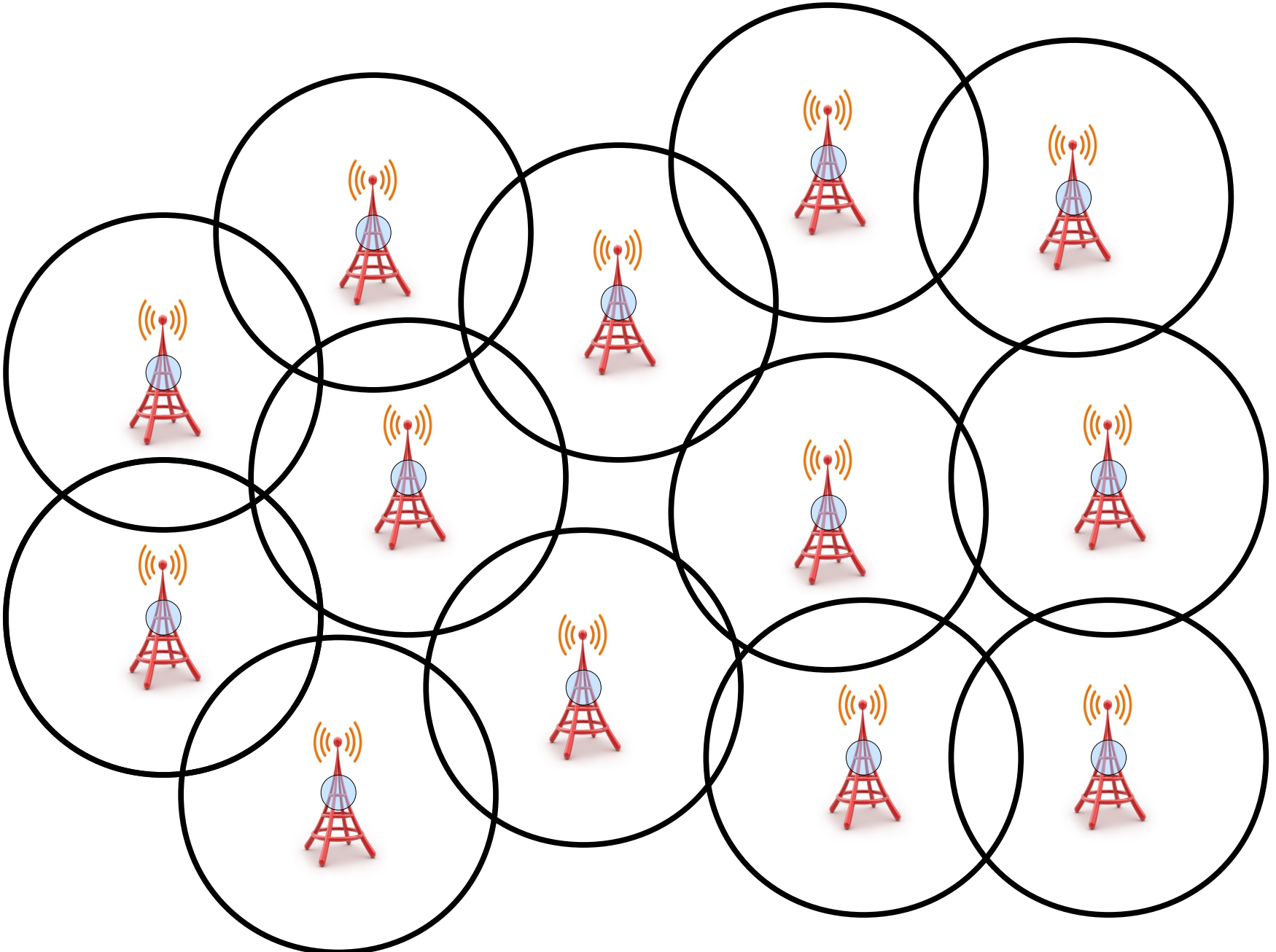
Graph Coloring



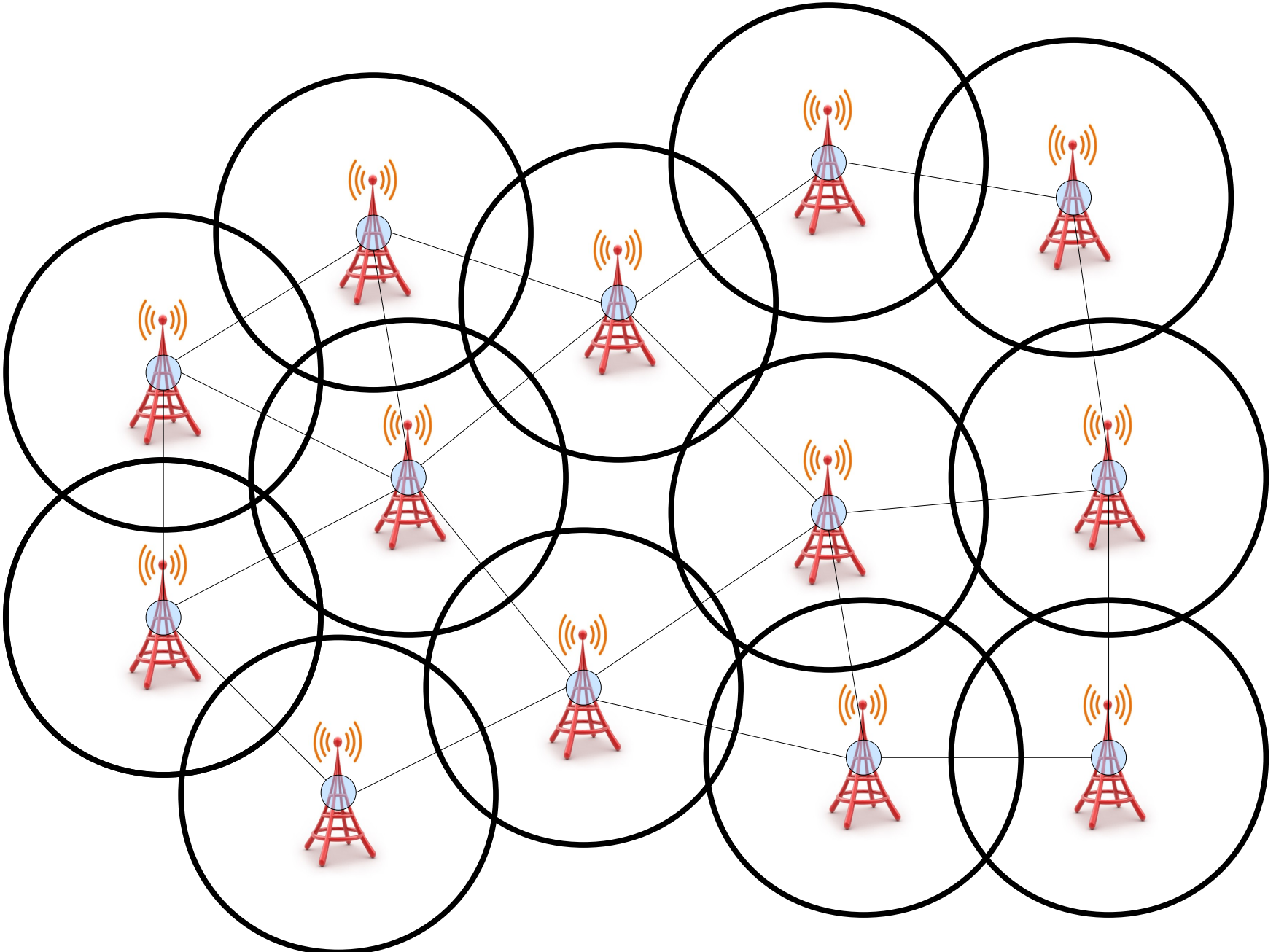
Graph Coloring is Useful



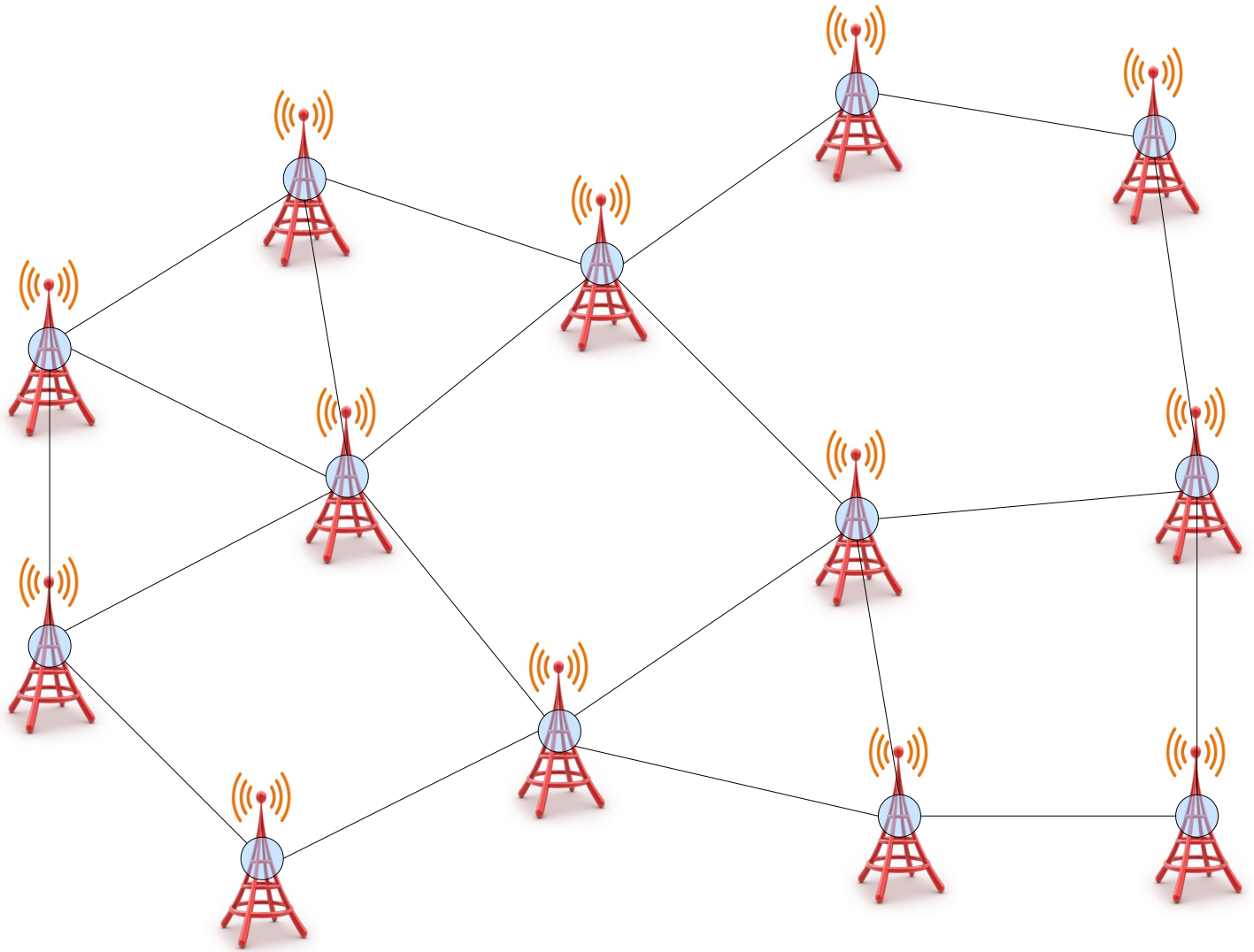
Graph Coloring is Useful



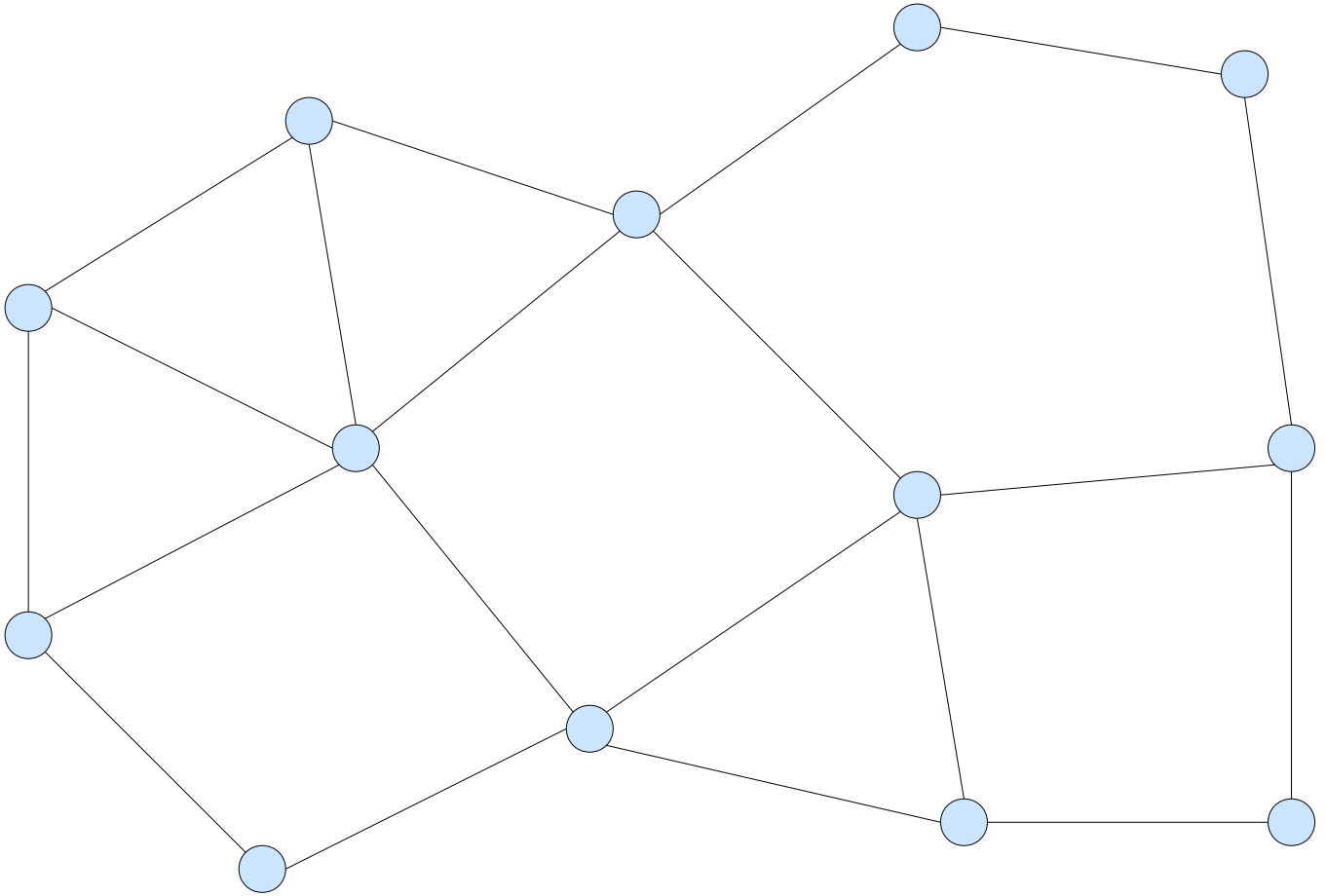
Graph Coloring is Useful



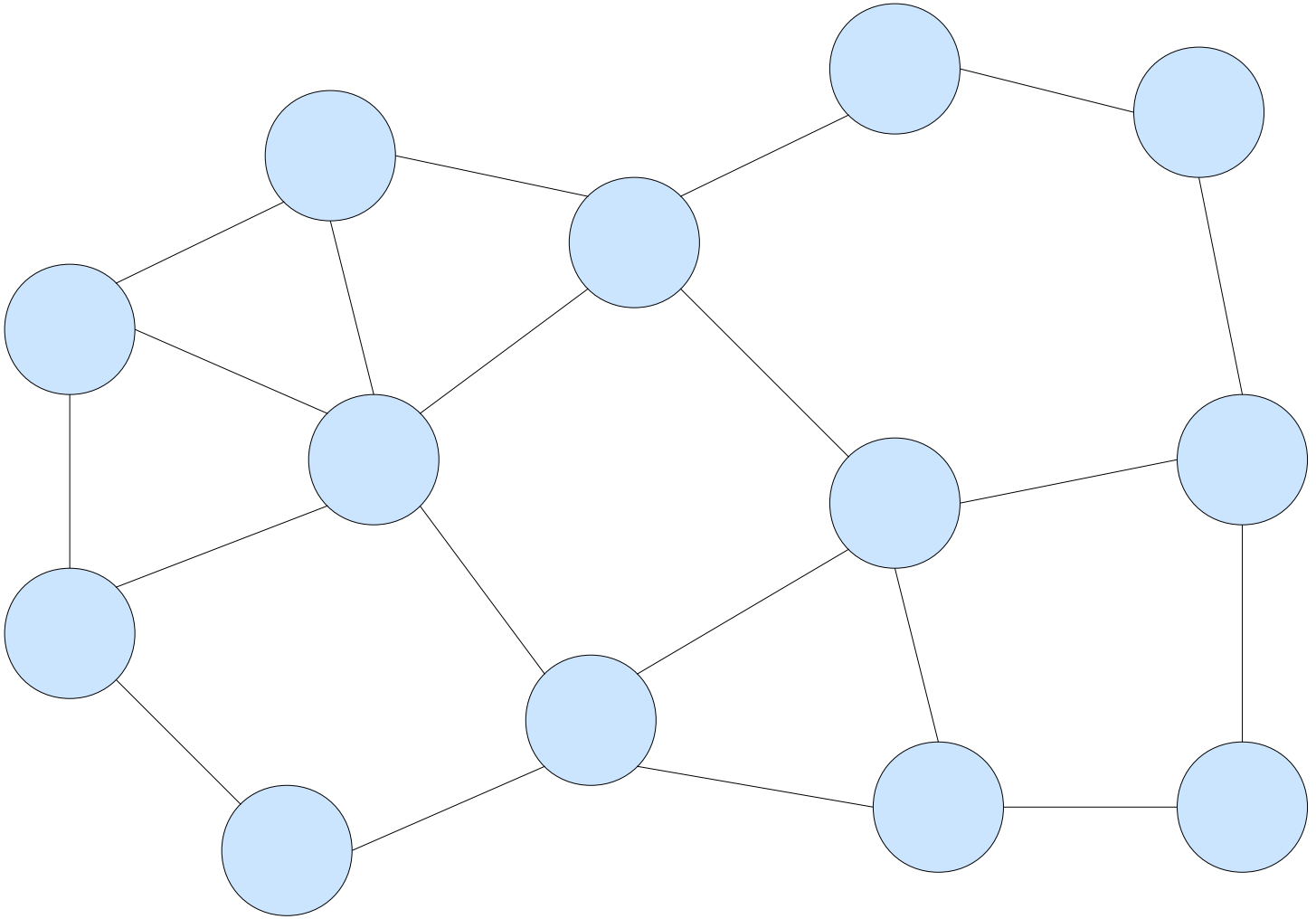
Graph Coloring is Useful



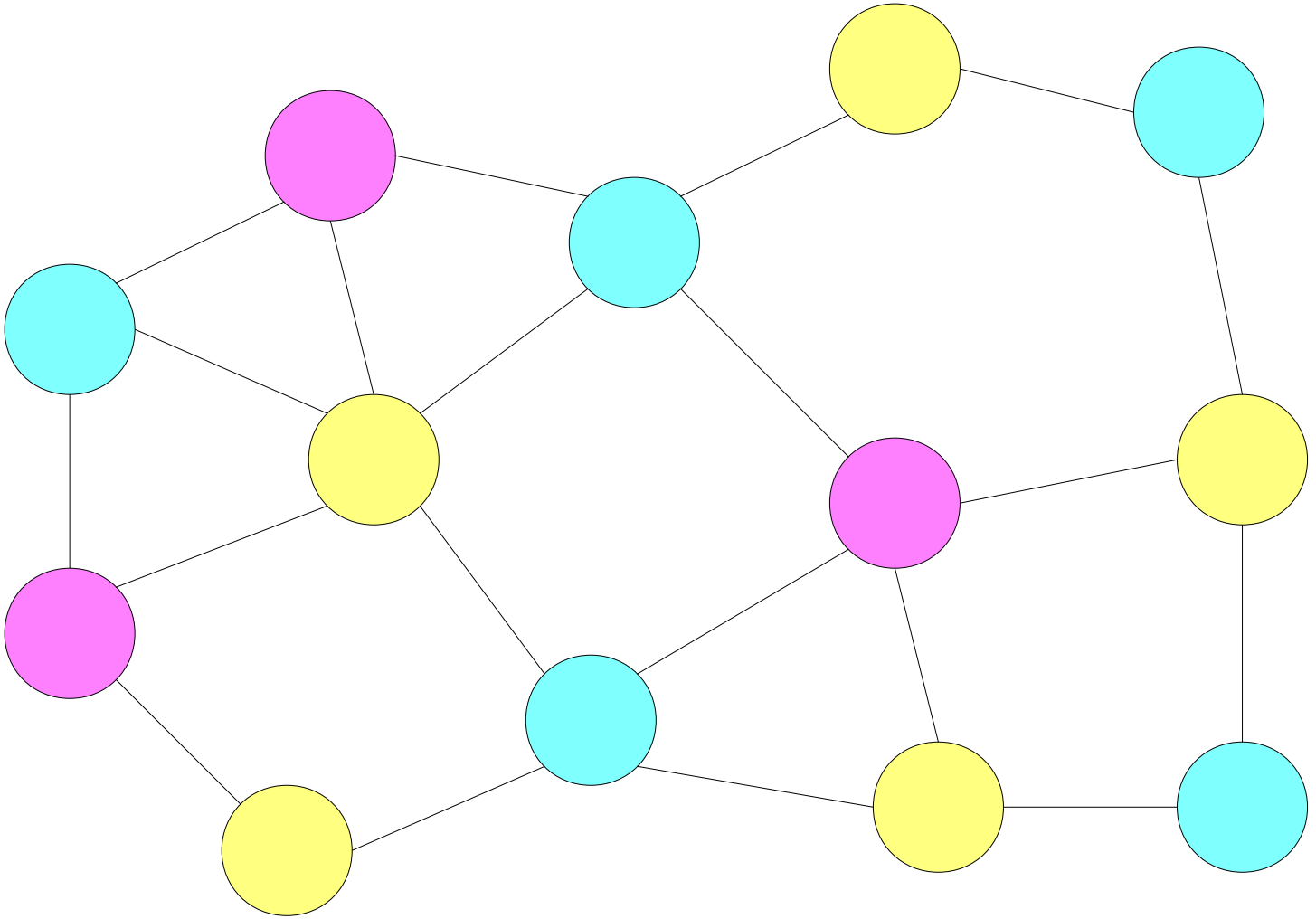
Graph Coloring is Useful



Graph Coloring is Useful



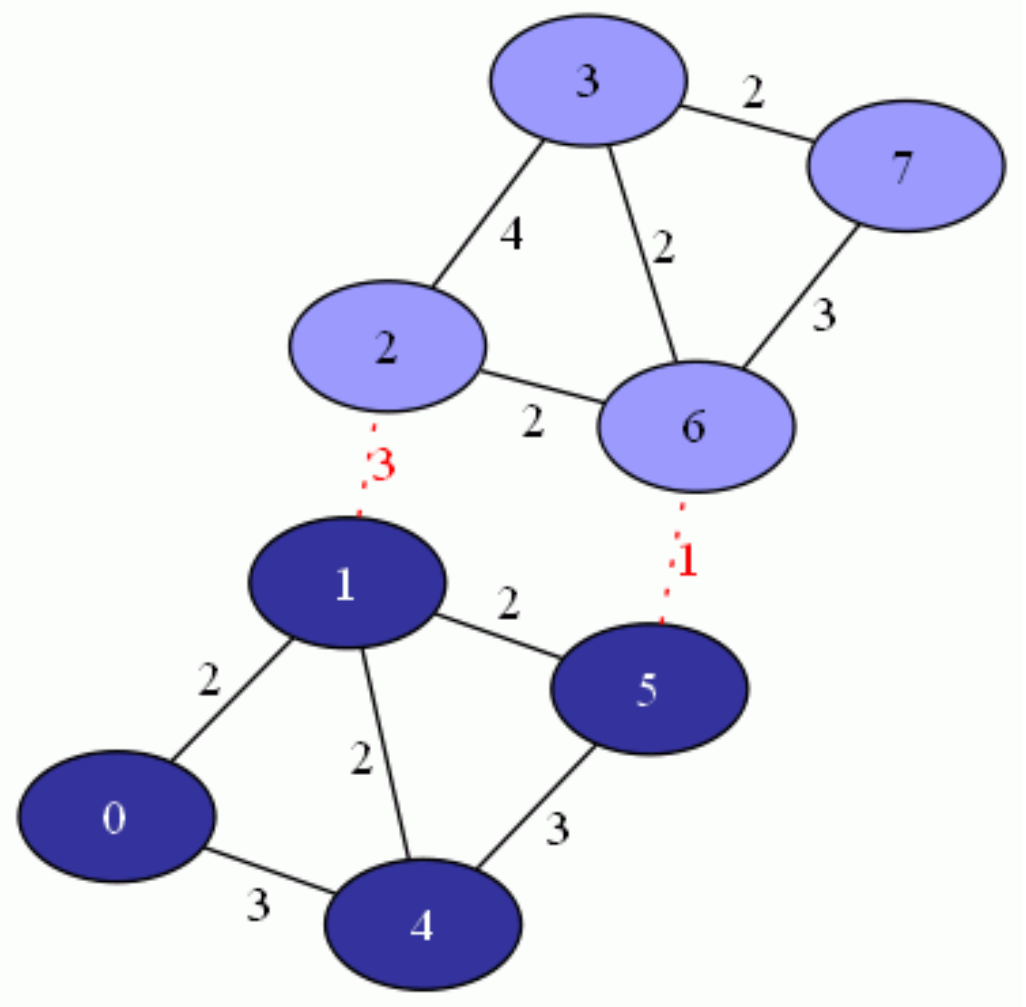
Graph Coloring is Useful



Graph Coloring is **Hard**.

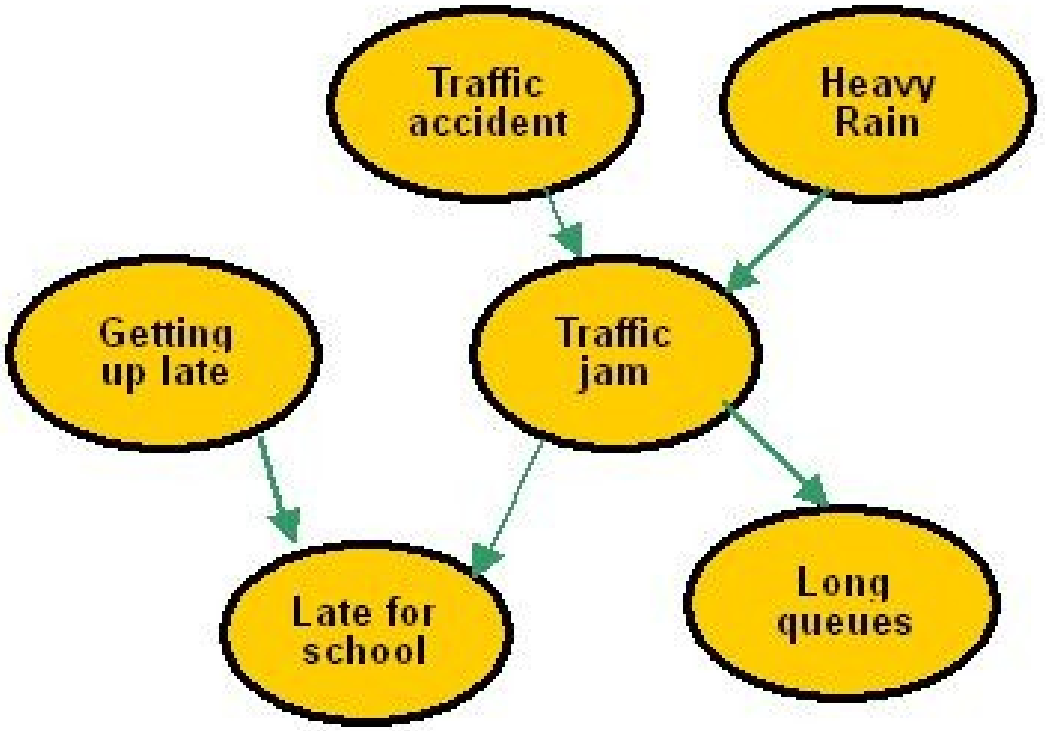
- No efficient algorithms are known for determining whether a graph can be colored with k colors for any $k > 2$.
- Want \$1,000,000? **Find a polynomial-time algorithm** or **prove that none exists**.

Minimum Cut



http://www.boost.org/doc/libs/1_53_0_beta1/libs/graph/doc/stoer_wagner_imgs/stoer_wagner-example-min-cut.gif

Probabilistic Graphical Models



http://philosophy.hku.hk/think/sci/causal-net-1.jpg

Summary of Graphs

- Graphs are an enormously flexible framework for encoding relationships between structures.
- MANY uses for graphs
- Want to learn more? Take **CS161**!