Greedy Algorithms

Part Two

Announcements

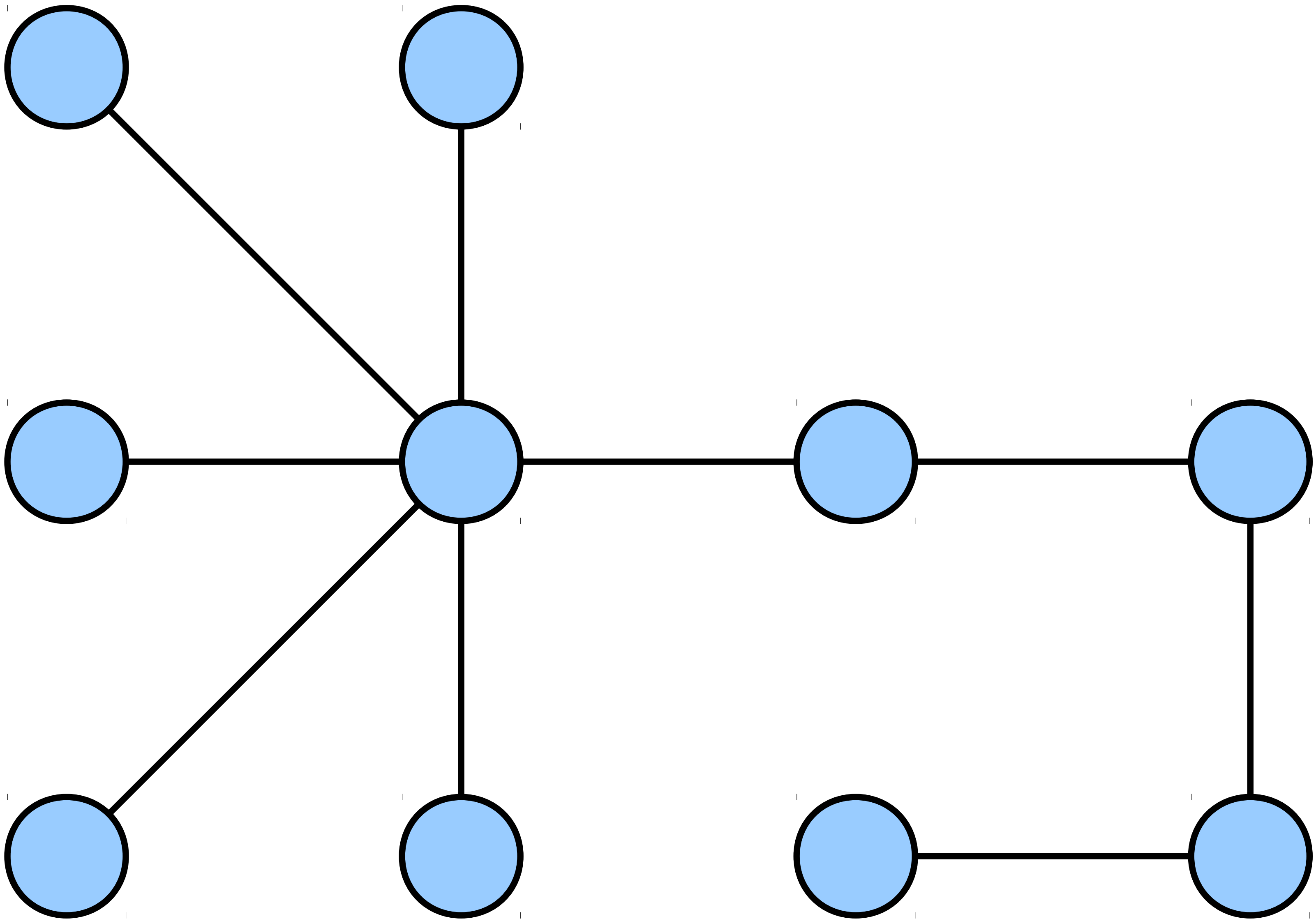
- Problem Set Three graded, will be returned at end of lecture.
- Problem Set Four due on Monday, or on Wednesday if you're using a late period.

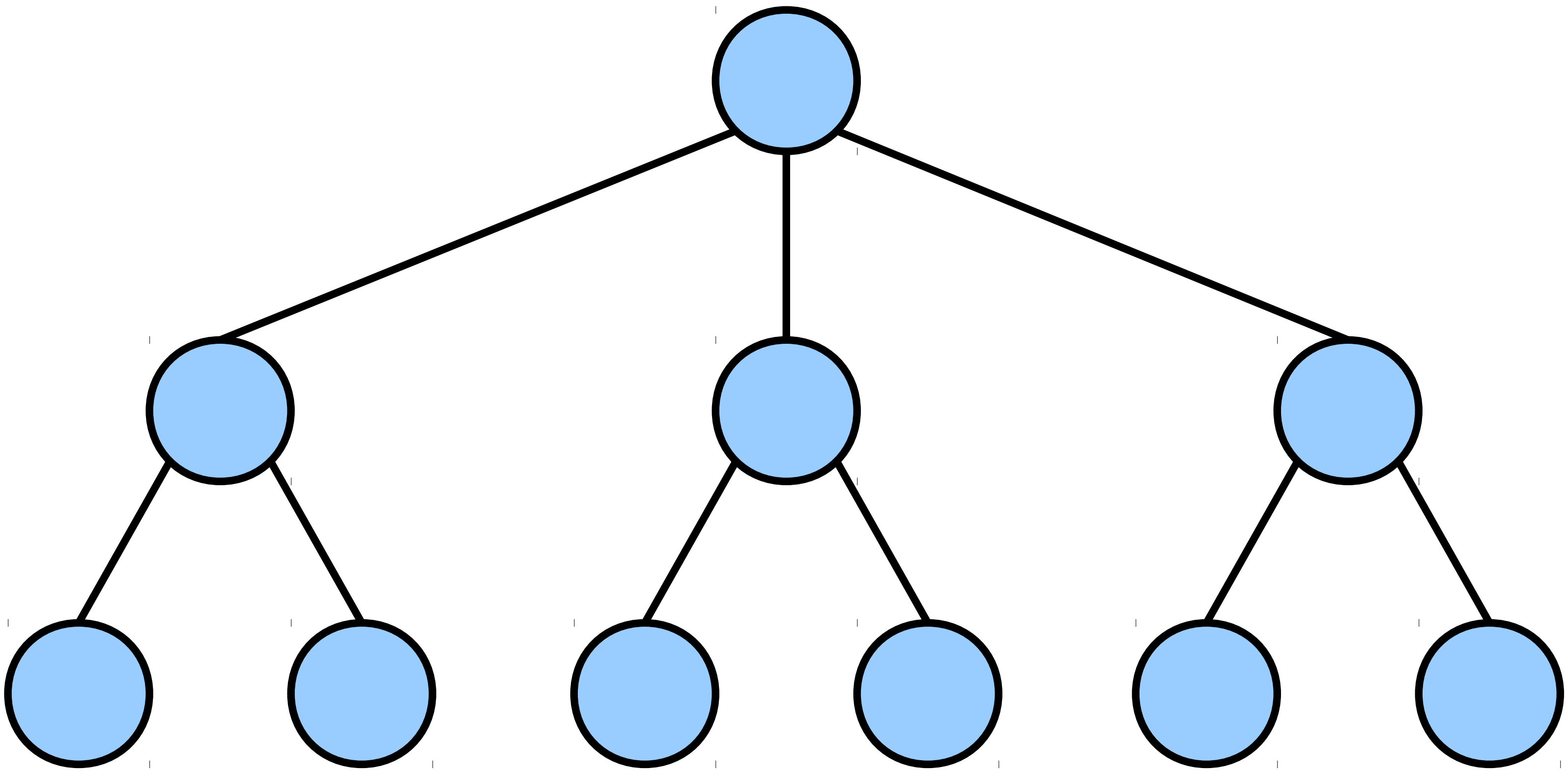
Outline for Today

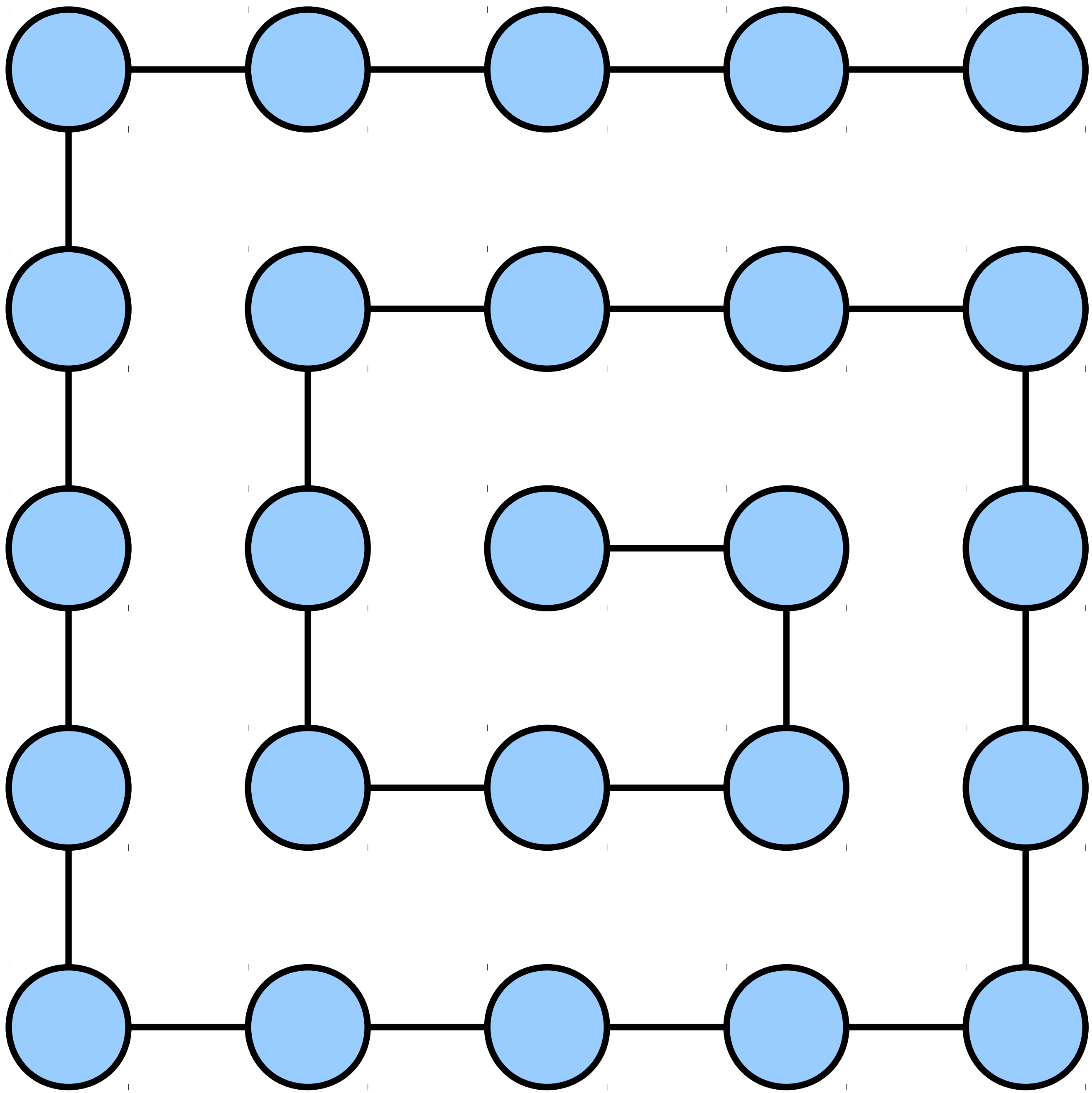
- **Minimum Spanning Trees**
  - What's the cheapest way to connect a graph?
- **Prim's Algorithm**
  - A simple and efficient algorithm for finding minimum spanning trees.
- **Exchange Arguments**
  - Another approach to proving greedy algorithms work correctly.

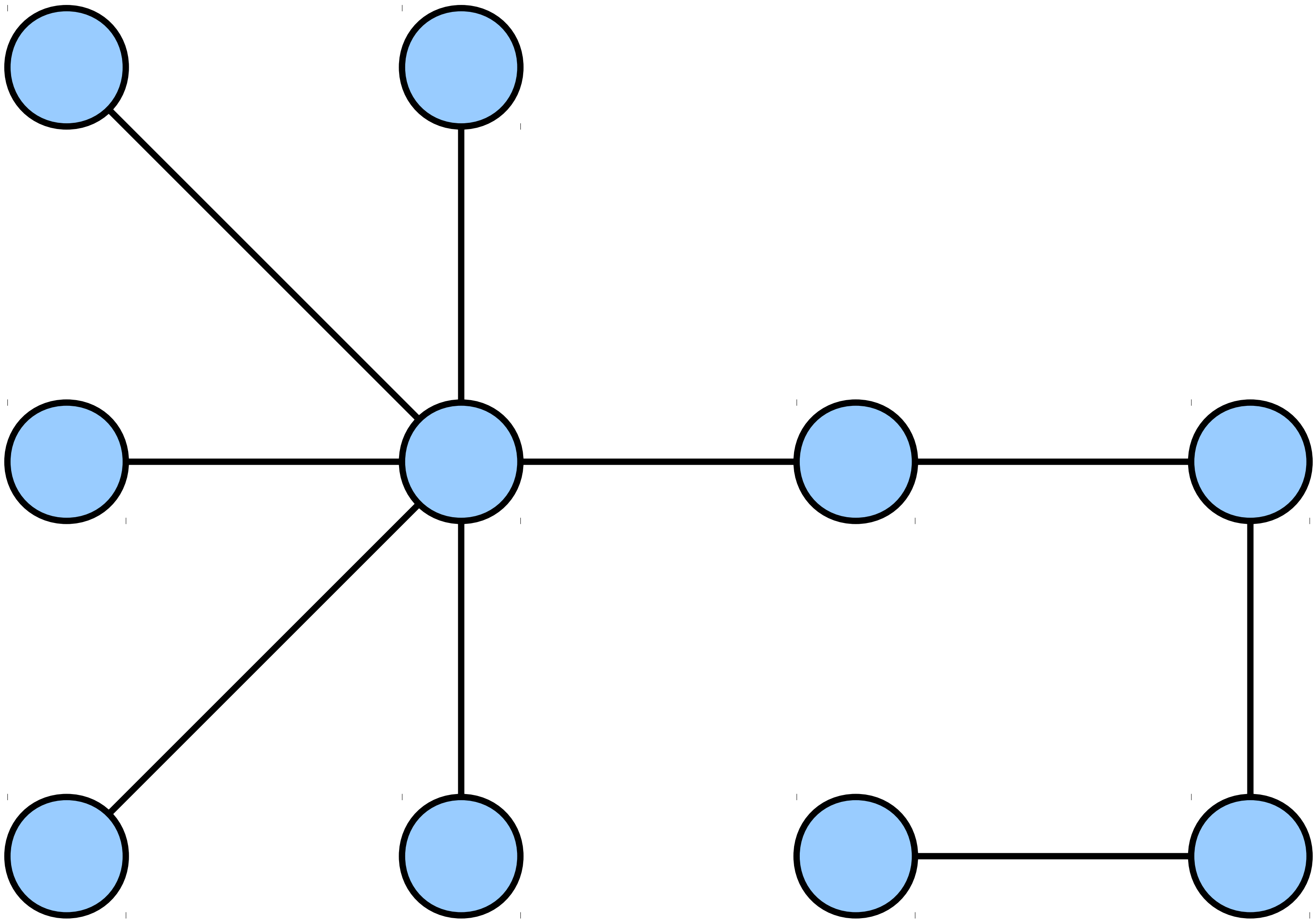
Trees

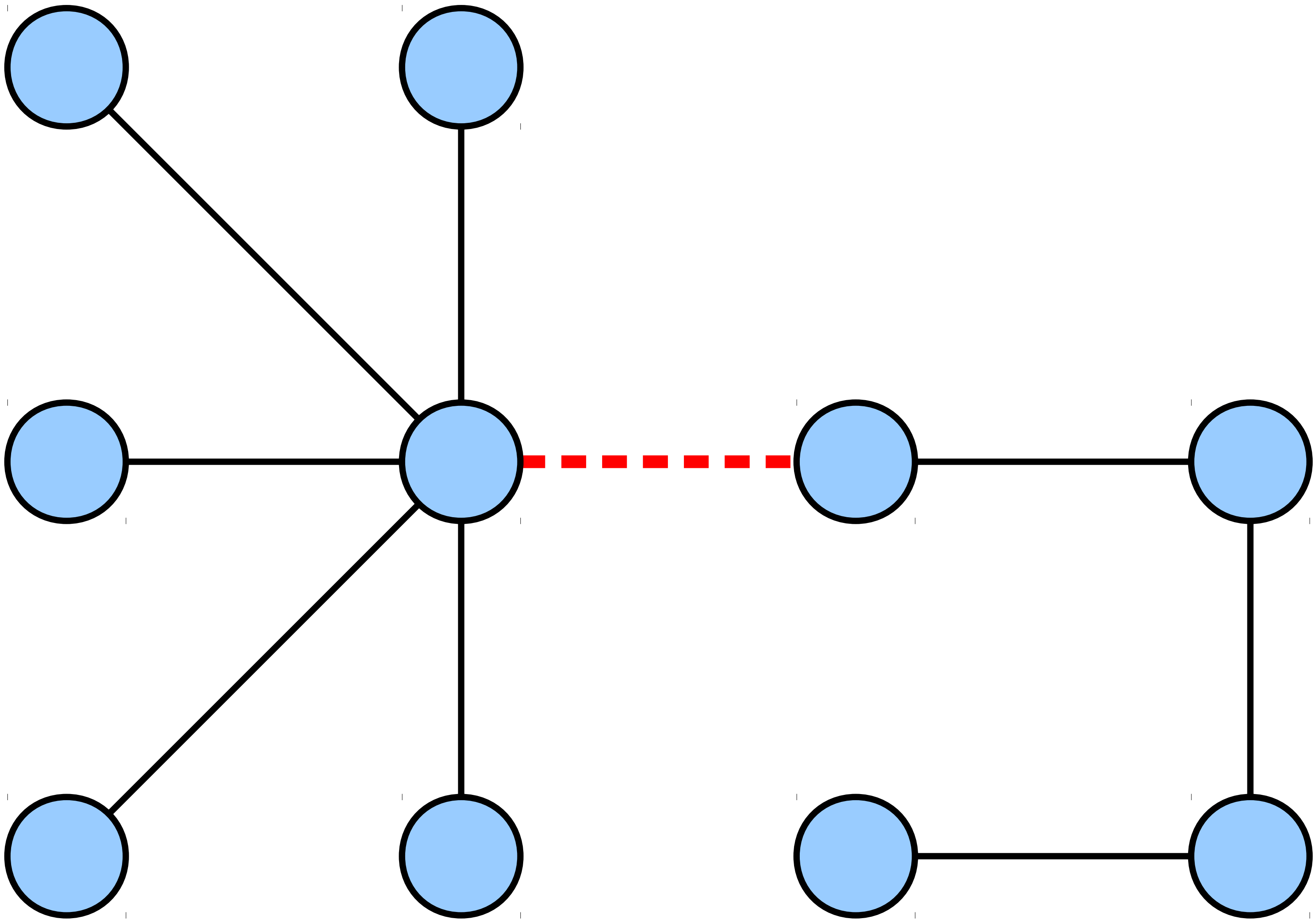
A **tree** is an undirected, acyclic, connected graph.

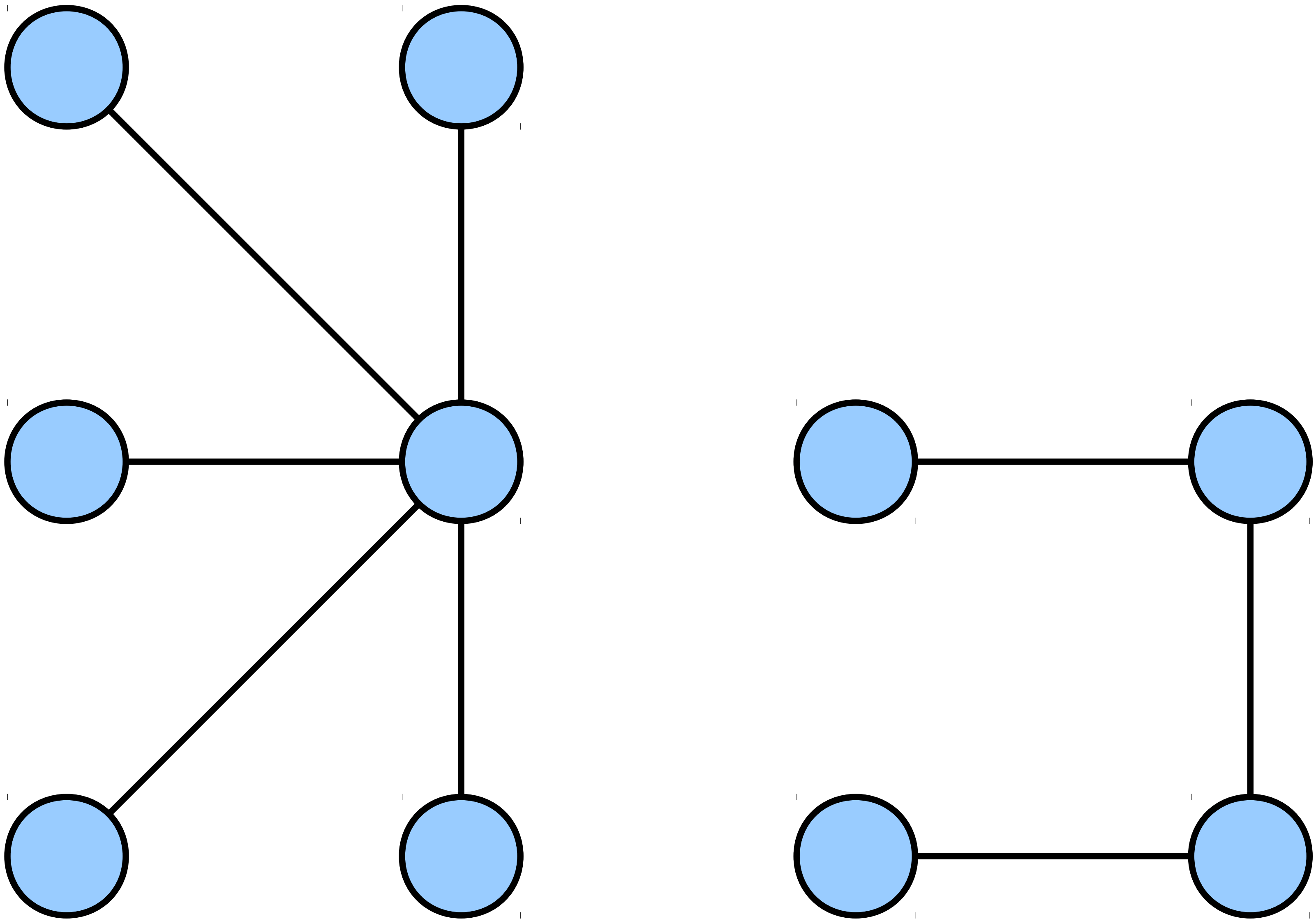


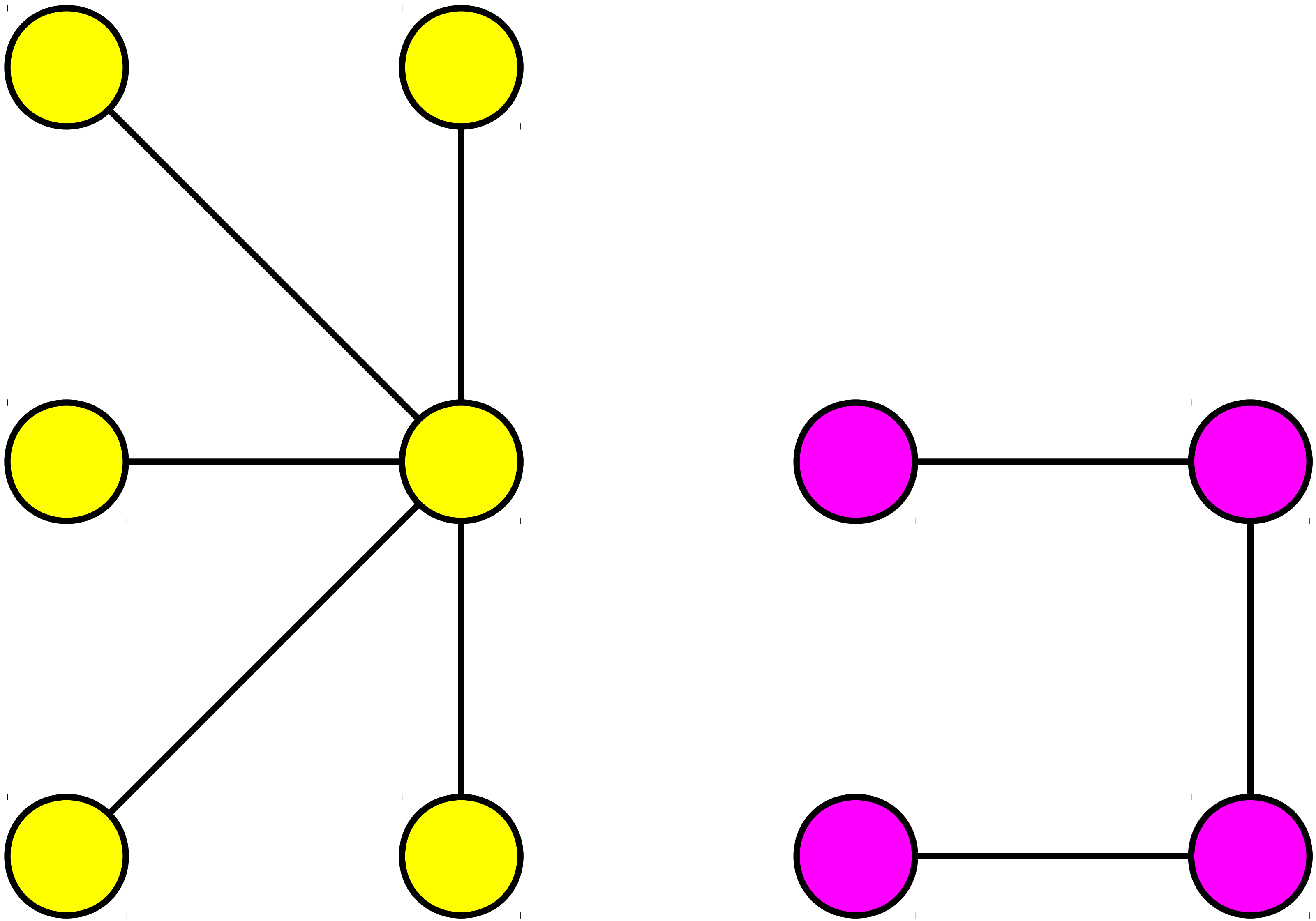


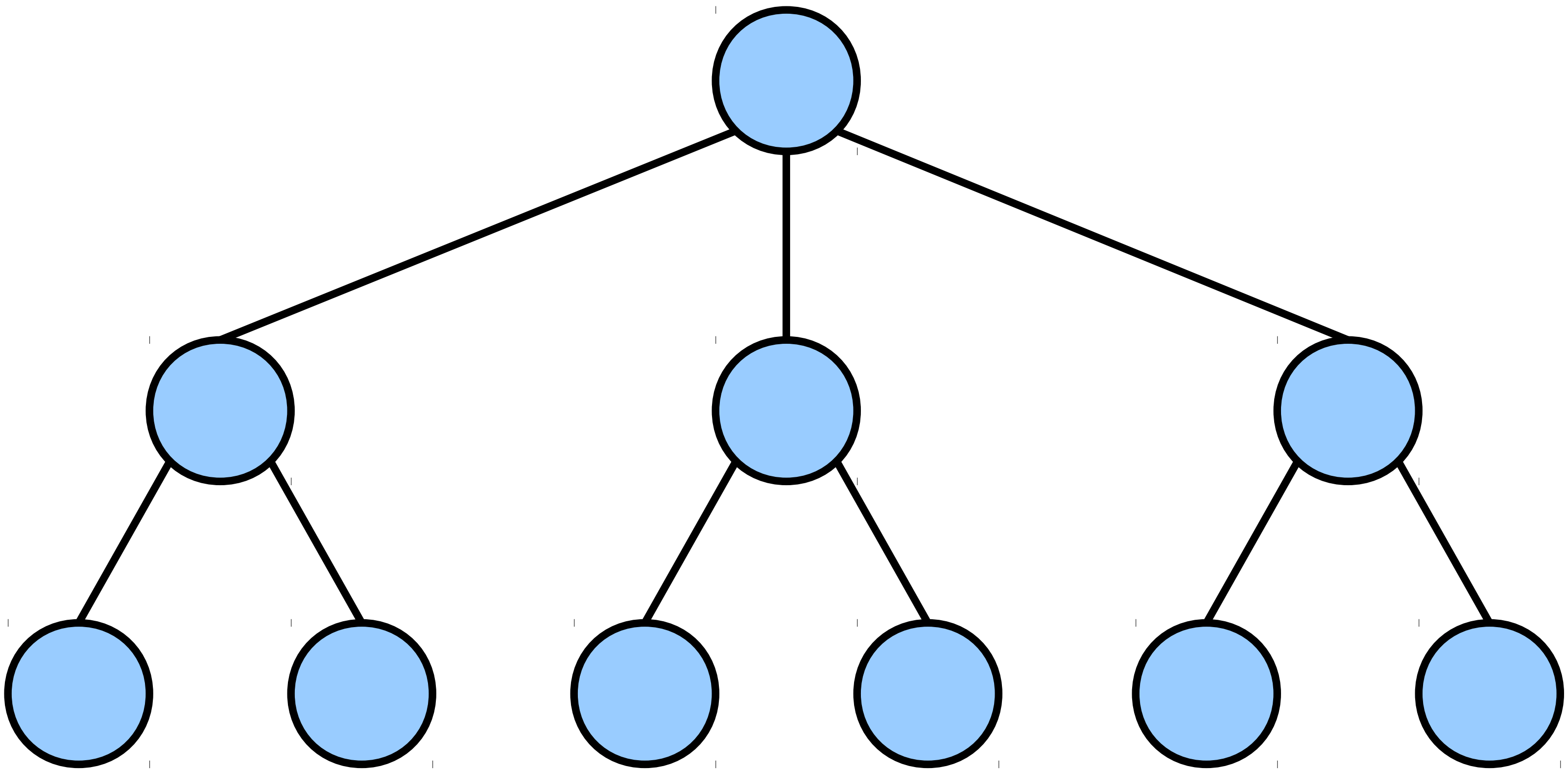


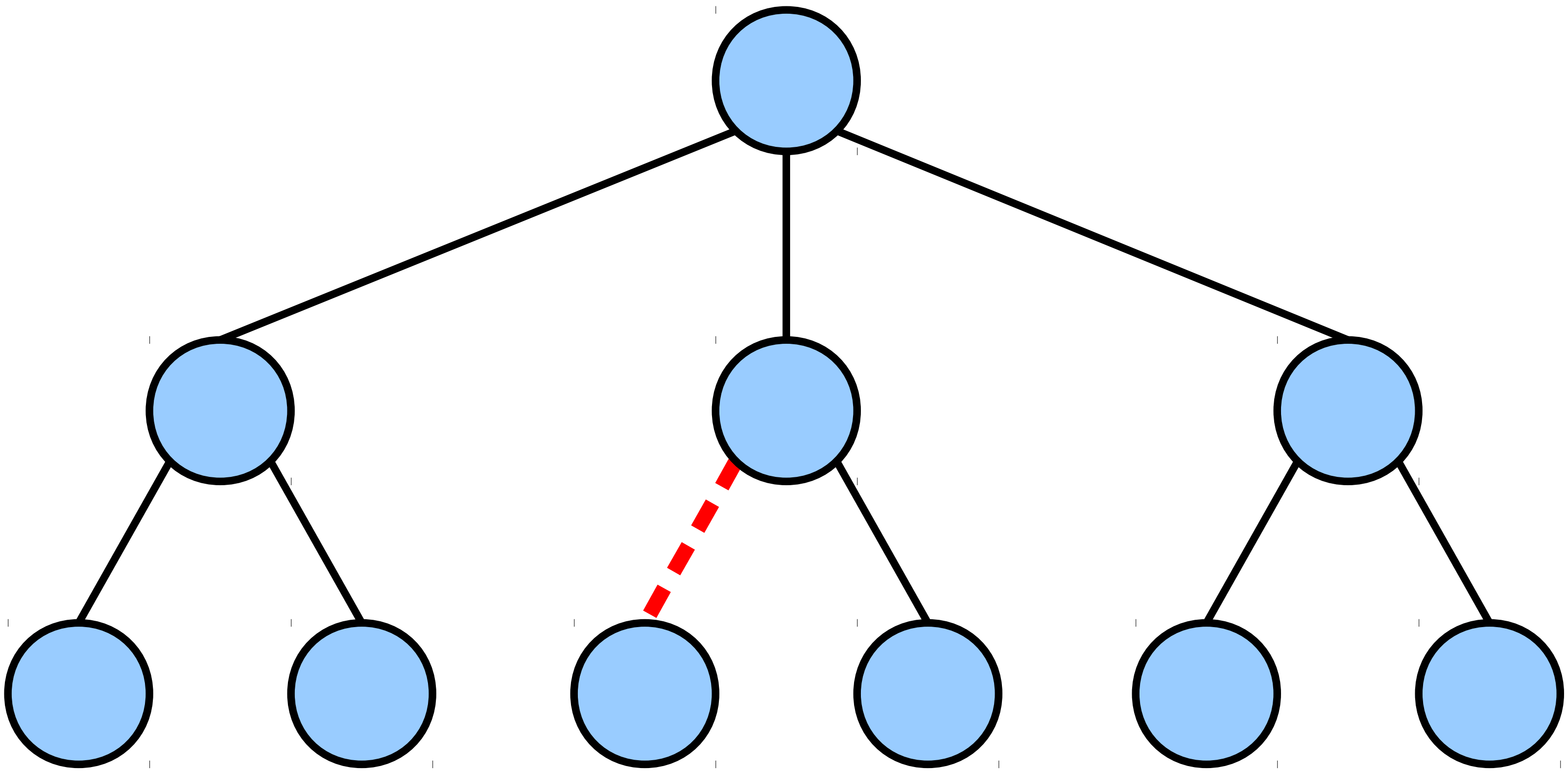


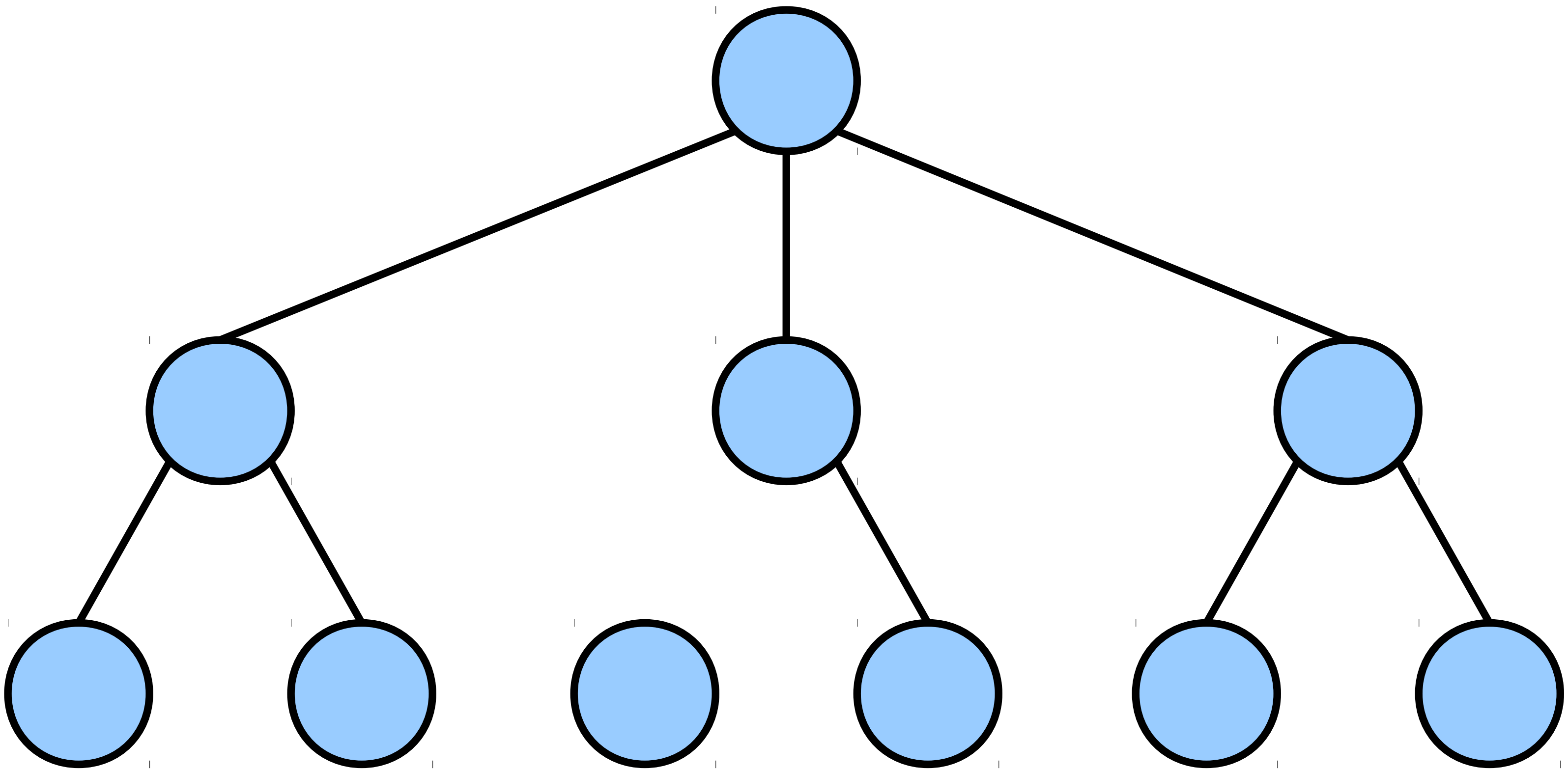


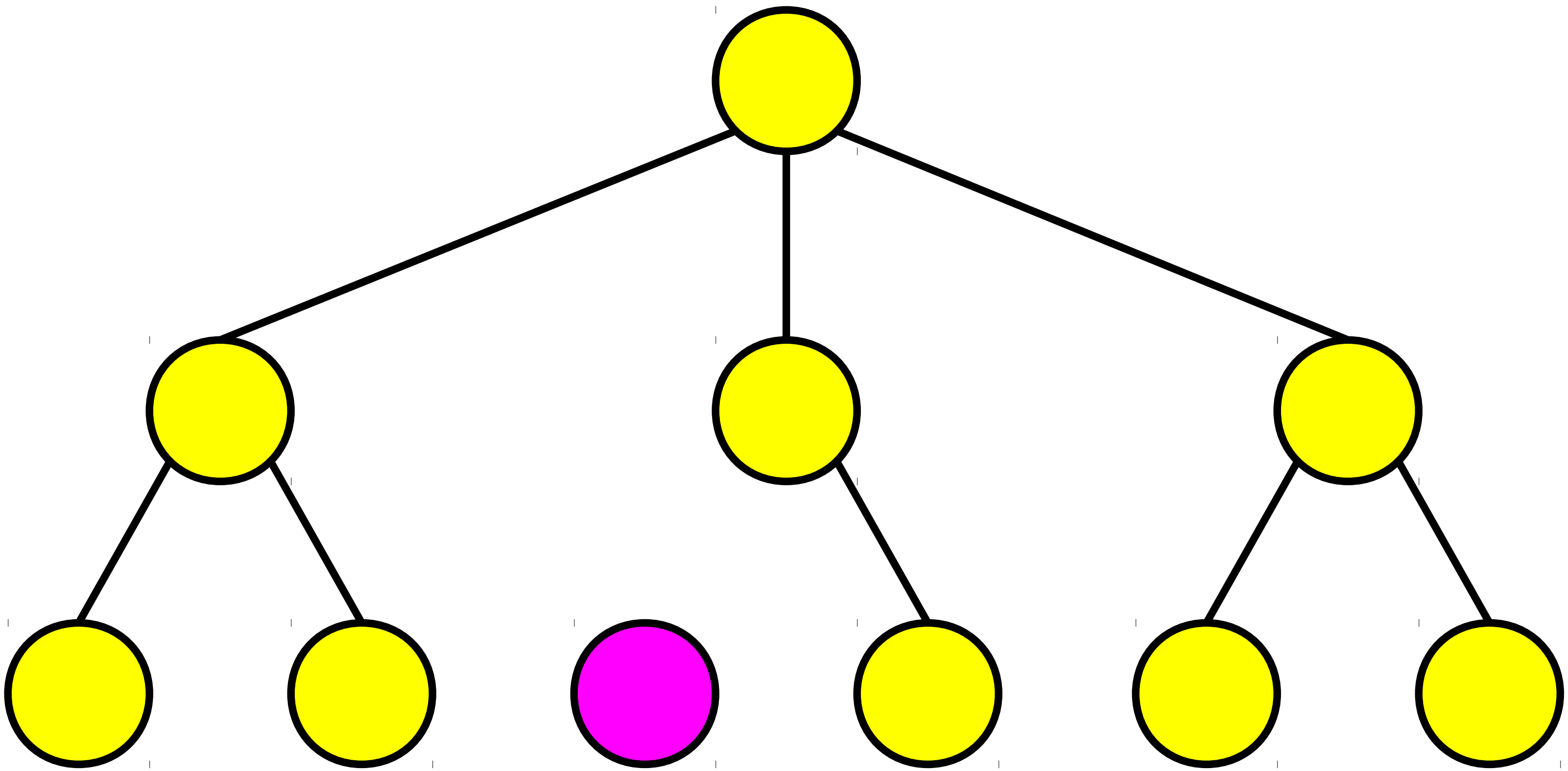


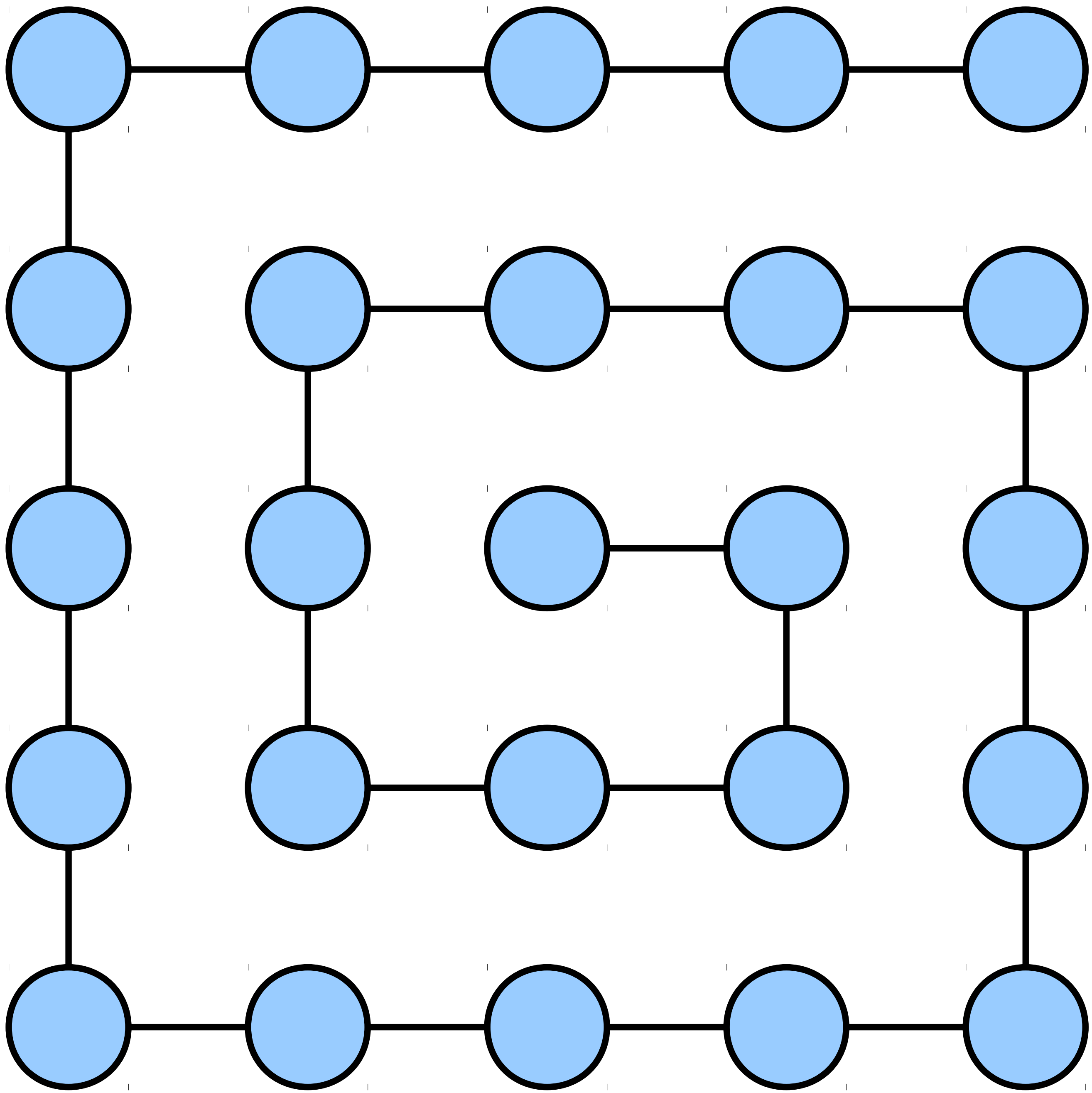


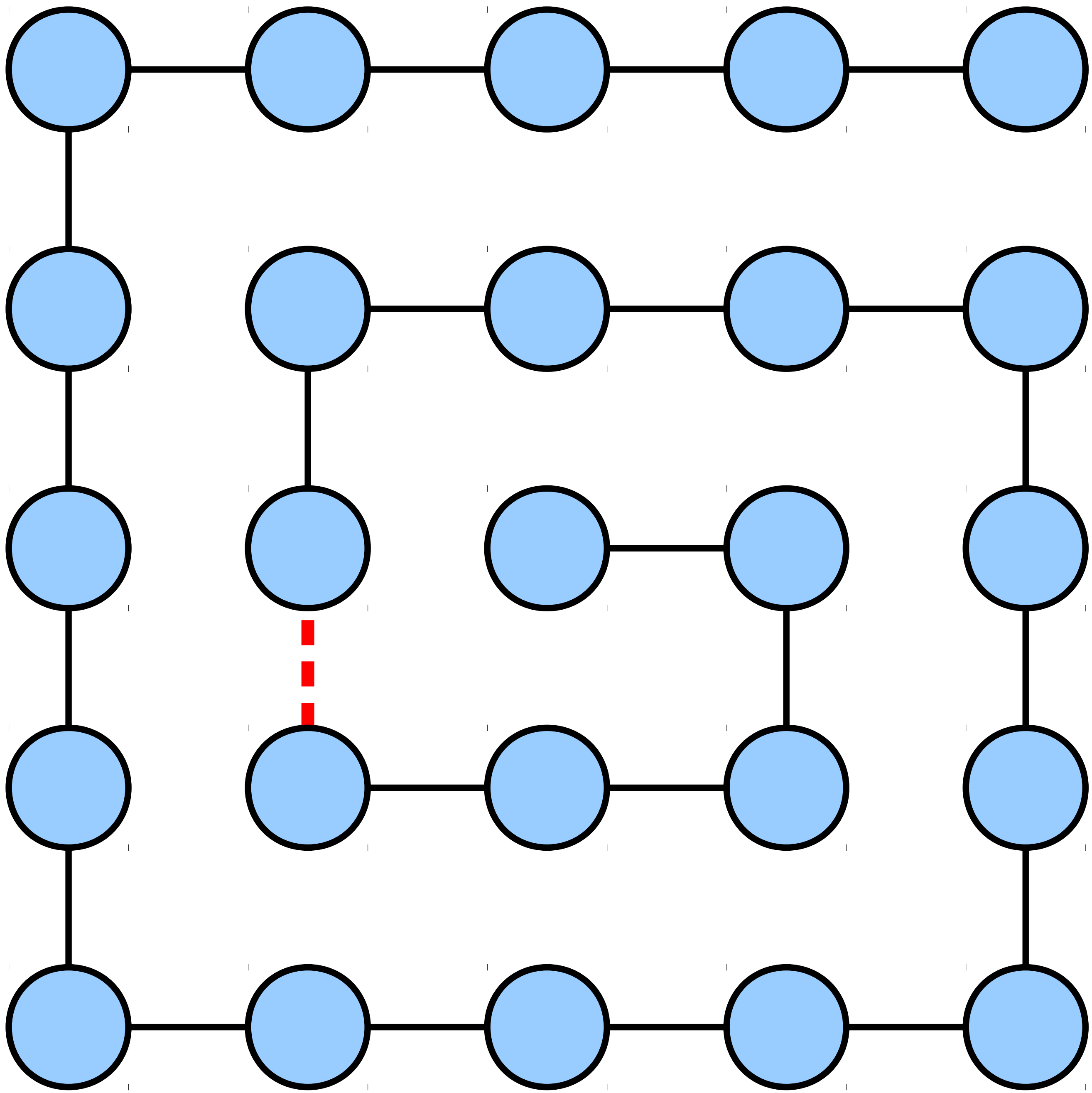


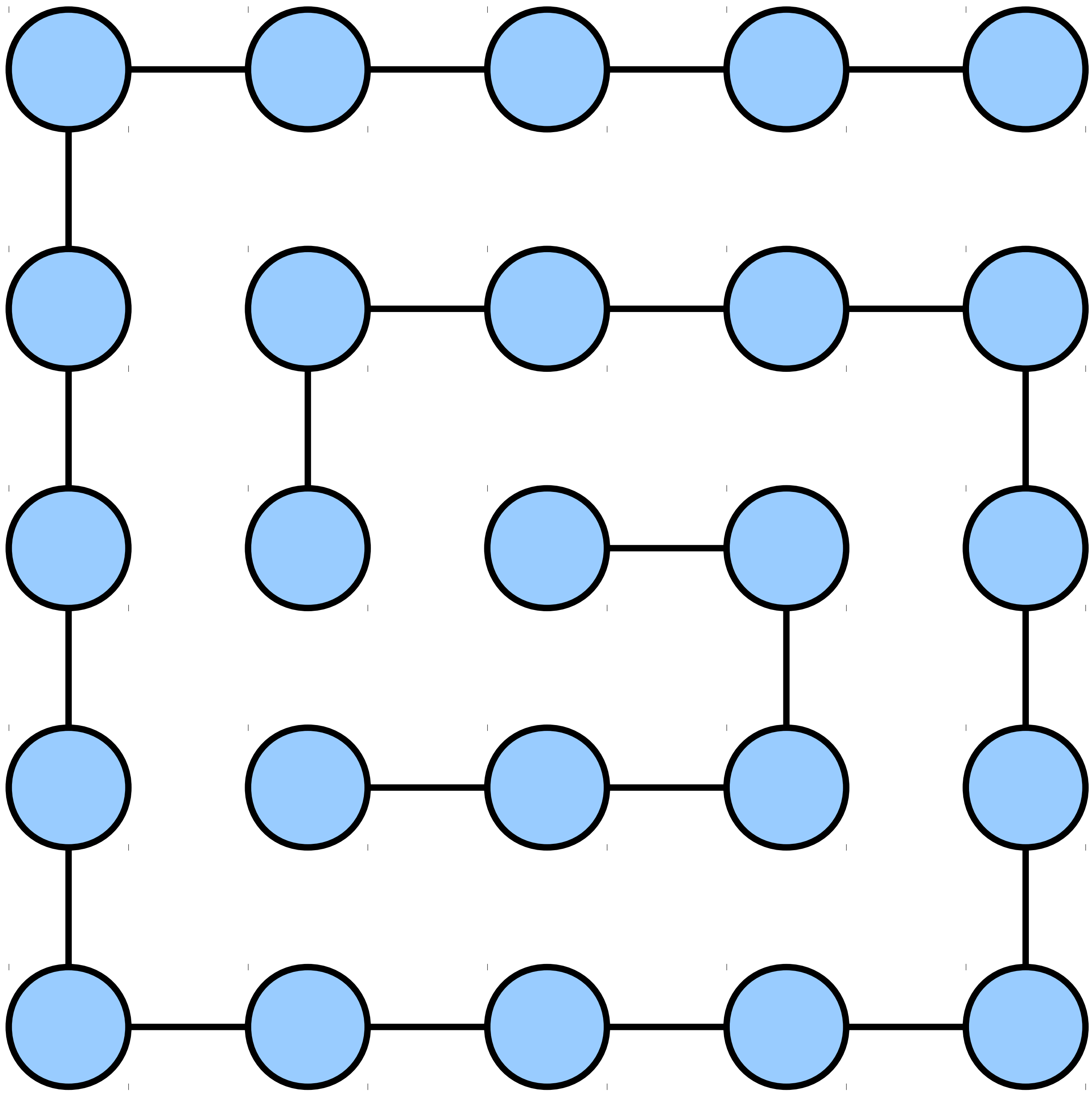


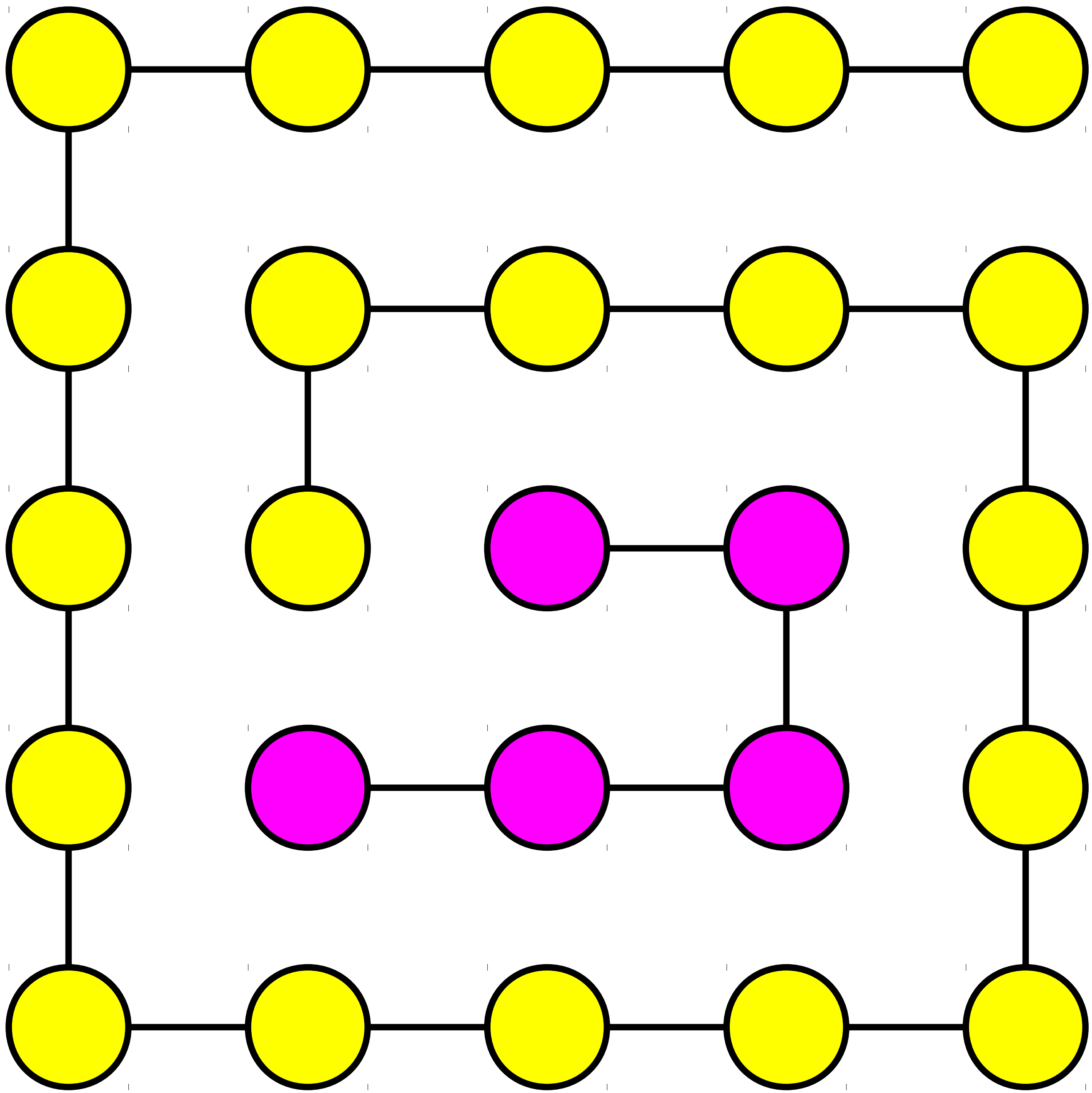






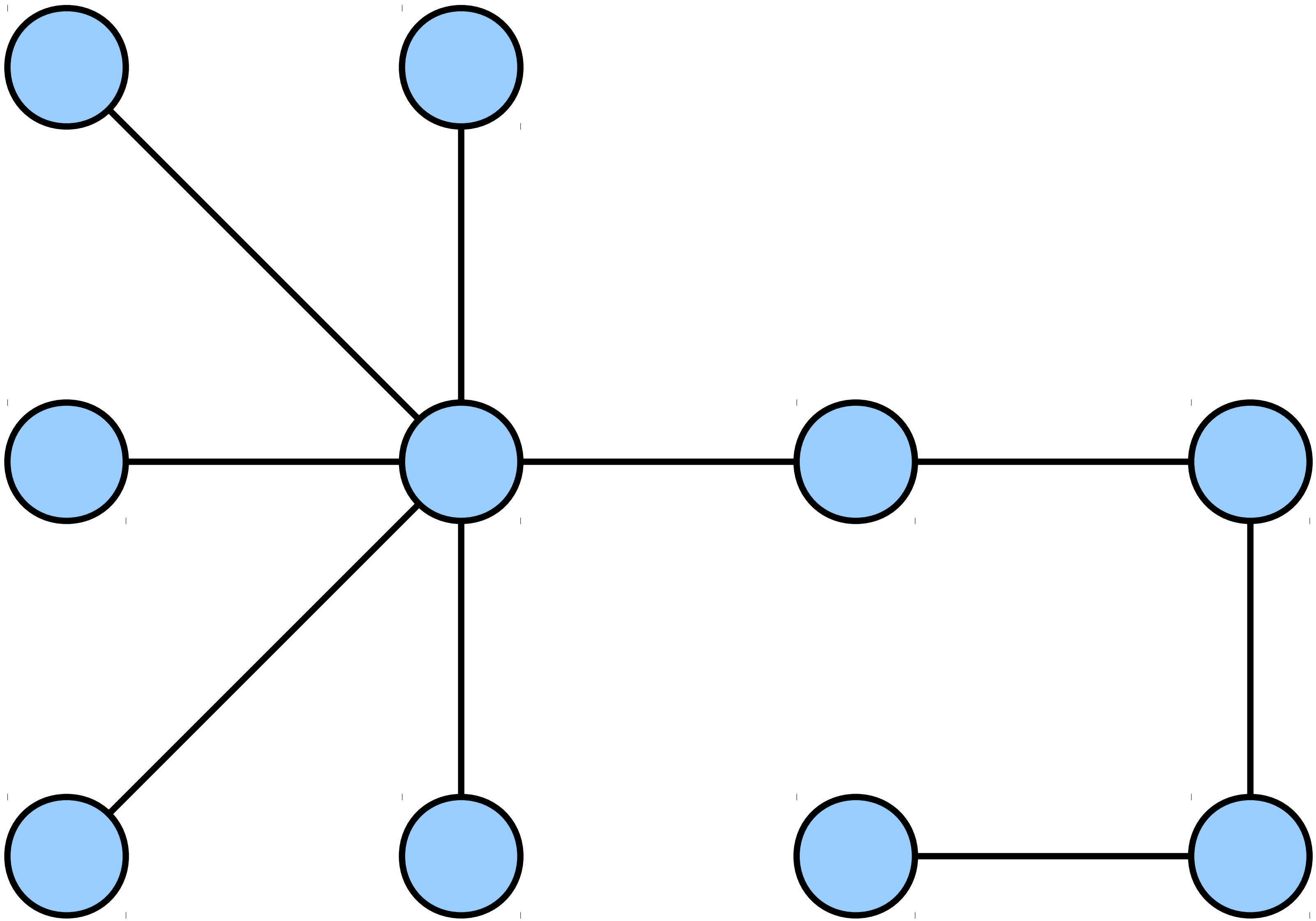


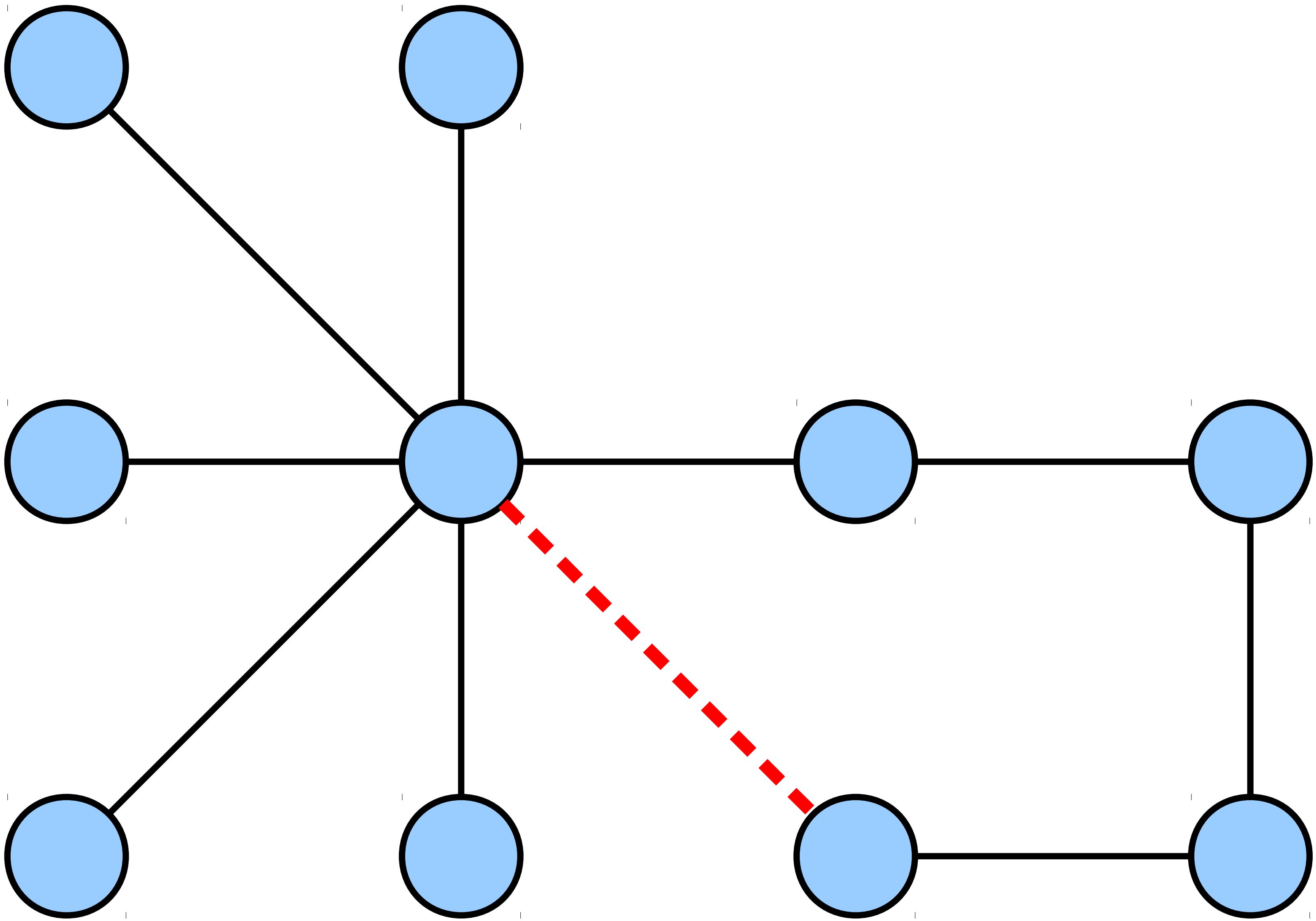


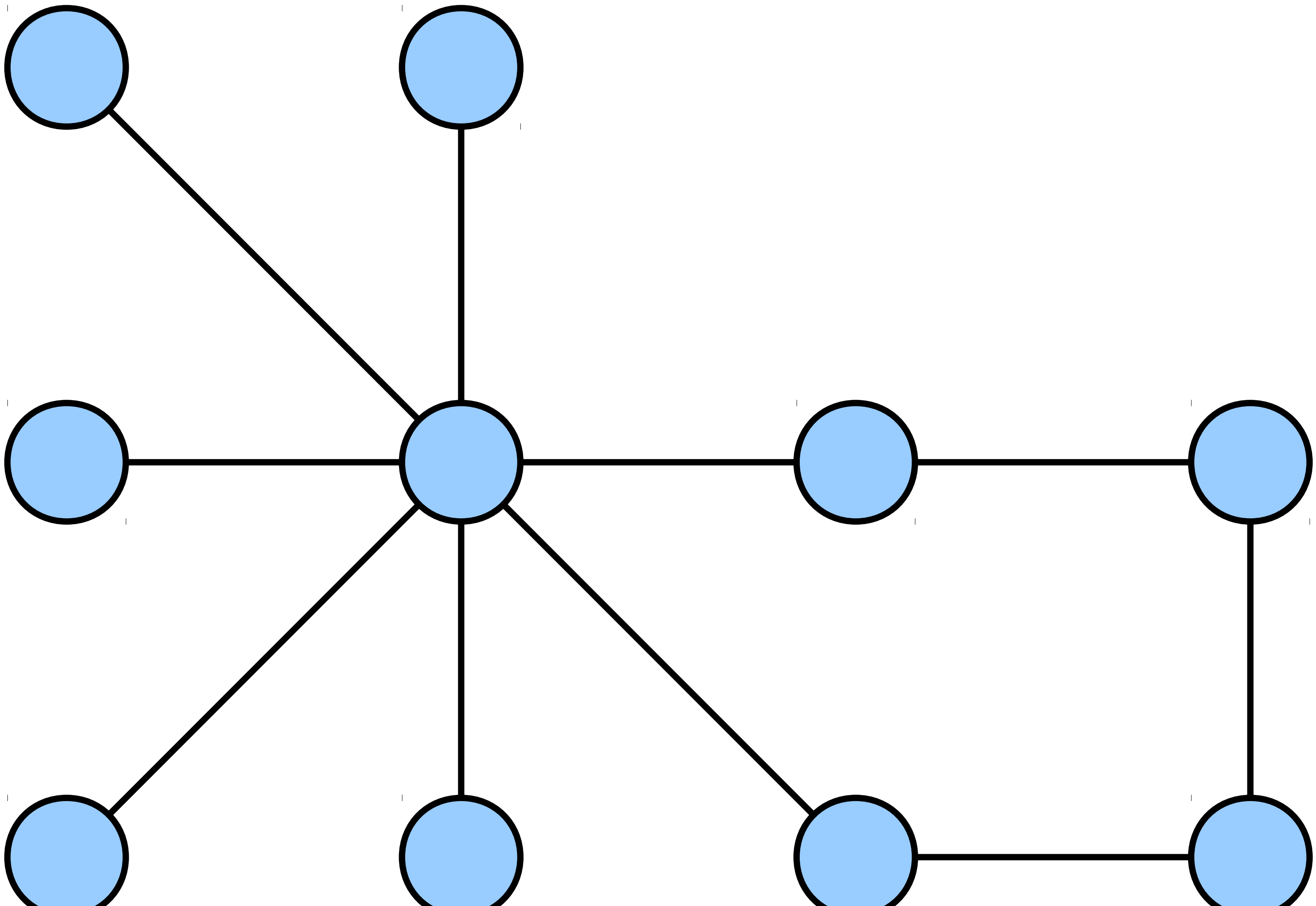


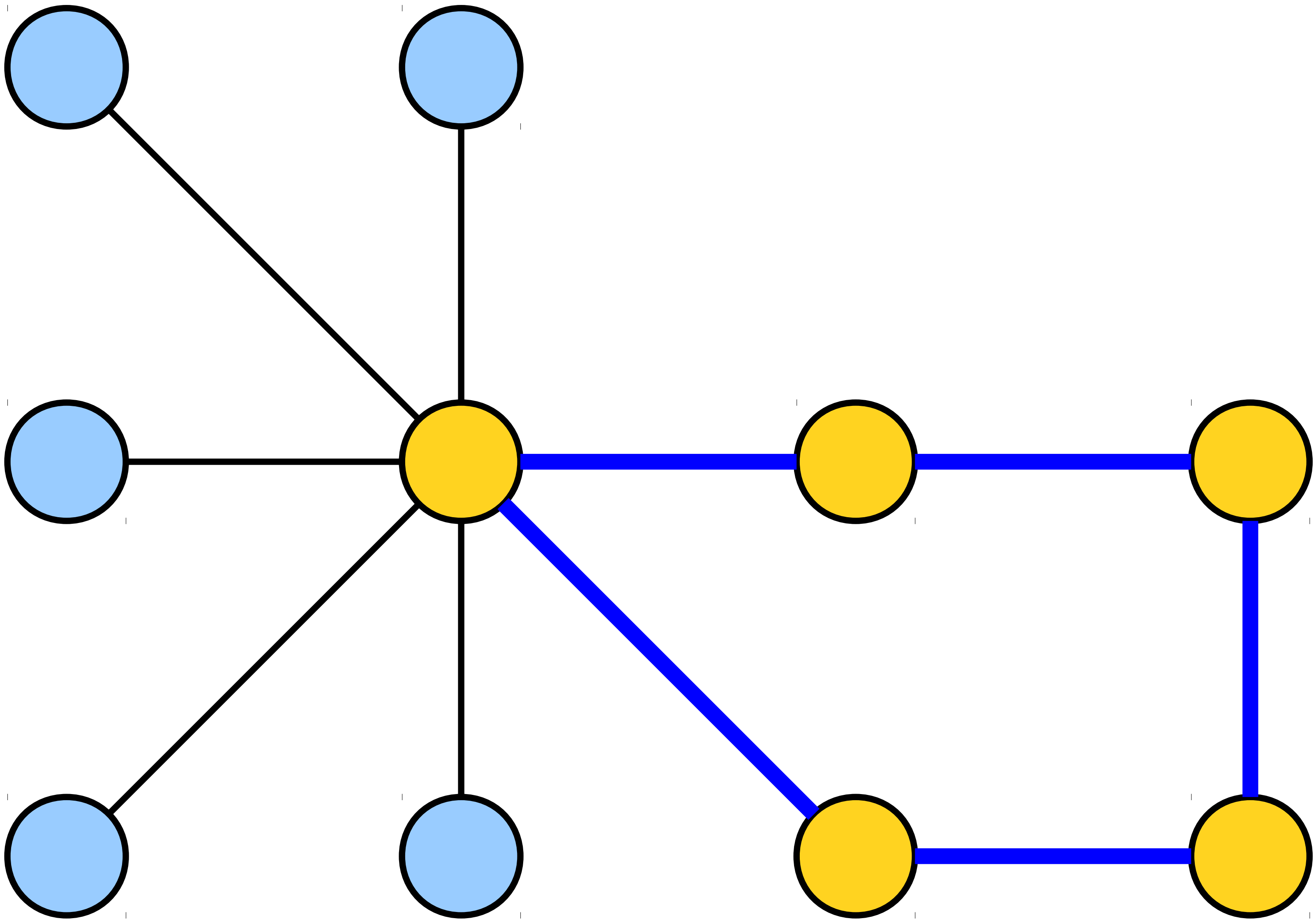
An undirected graph is called **minimally connected** iff it is connected and removing any edge disconnects it.

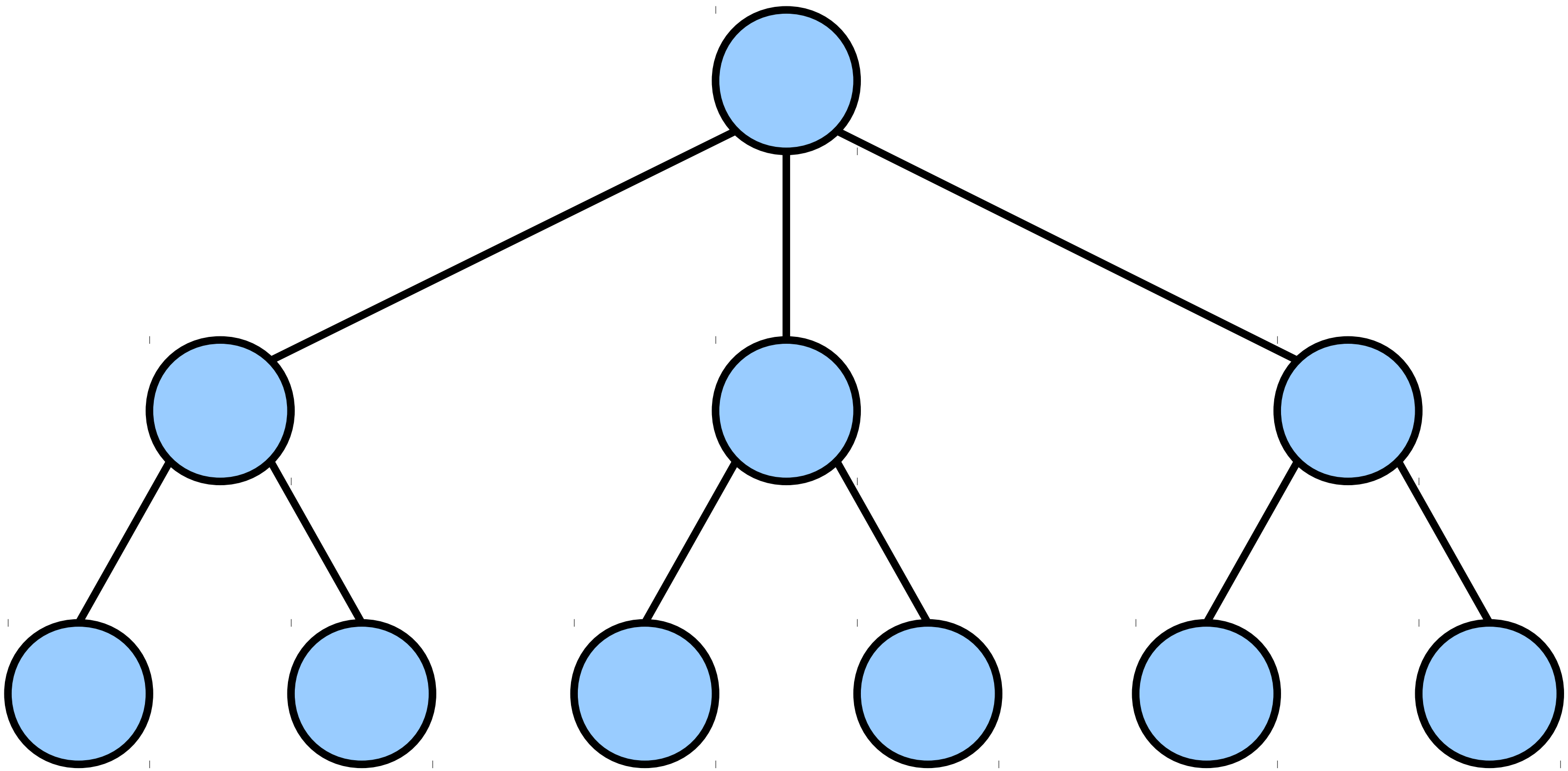
Theorem: An undirected graph is a tree iff it is minimally connected.

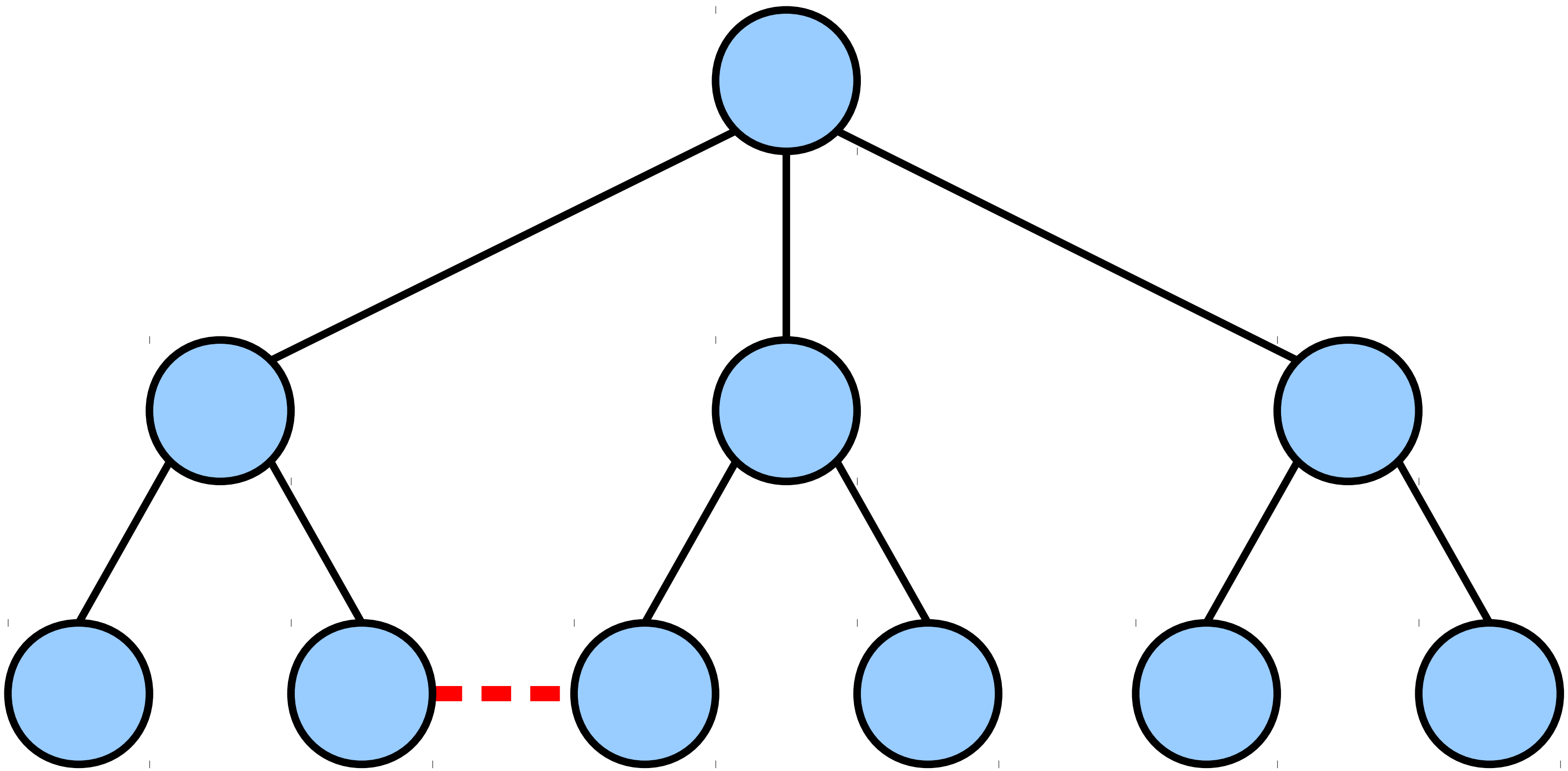


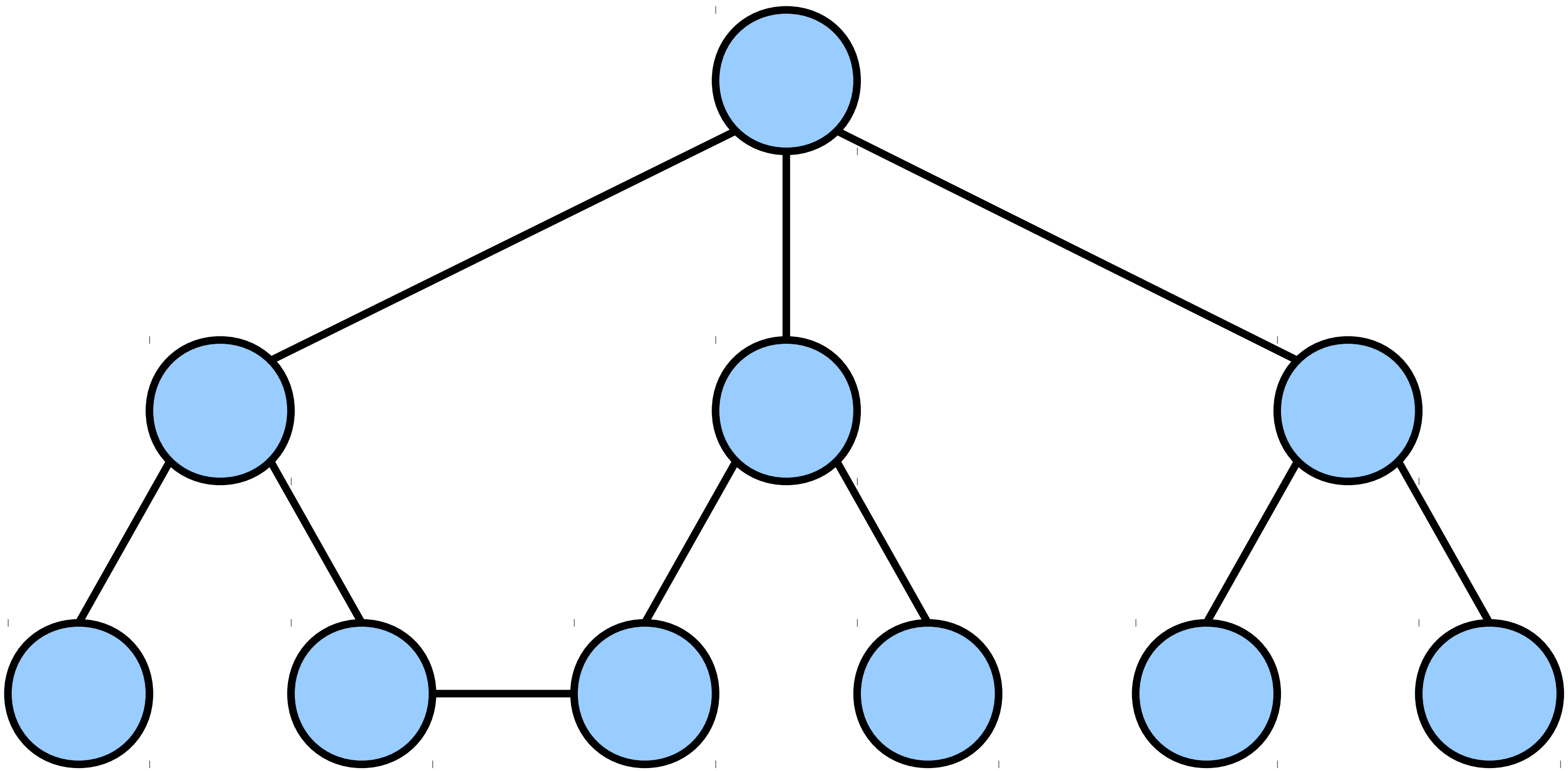


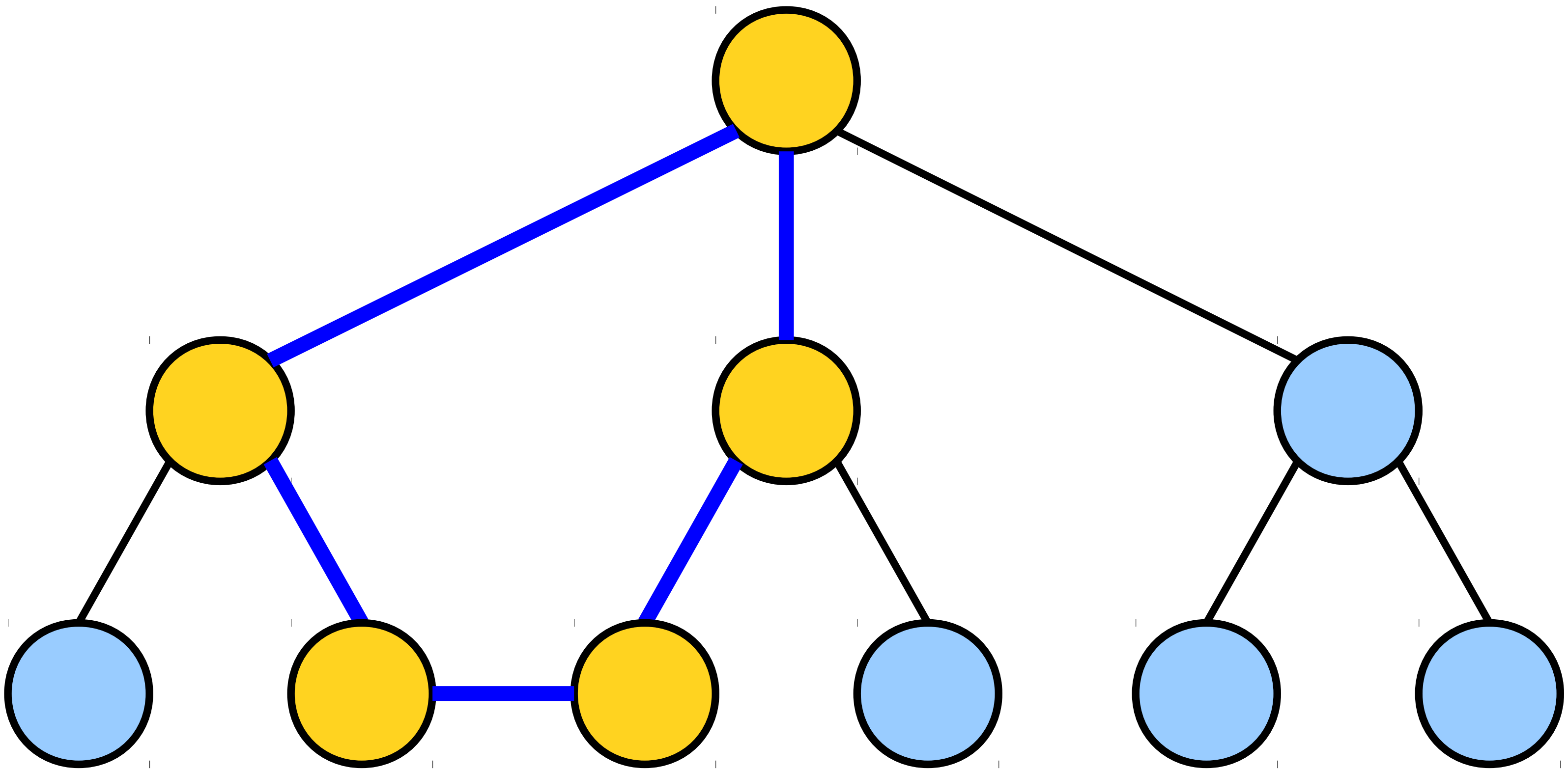


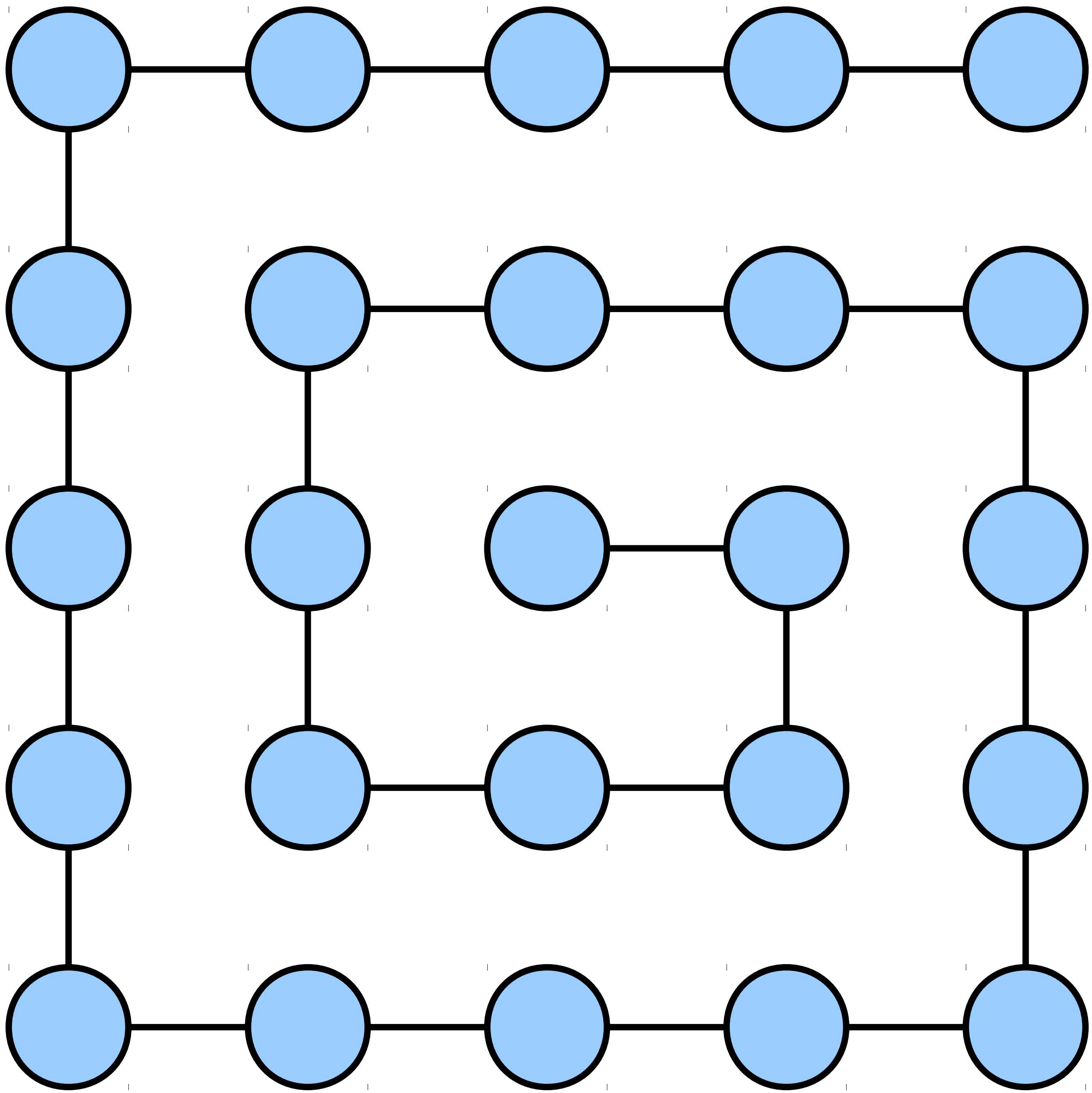


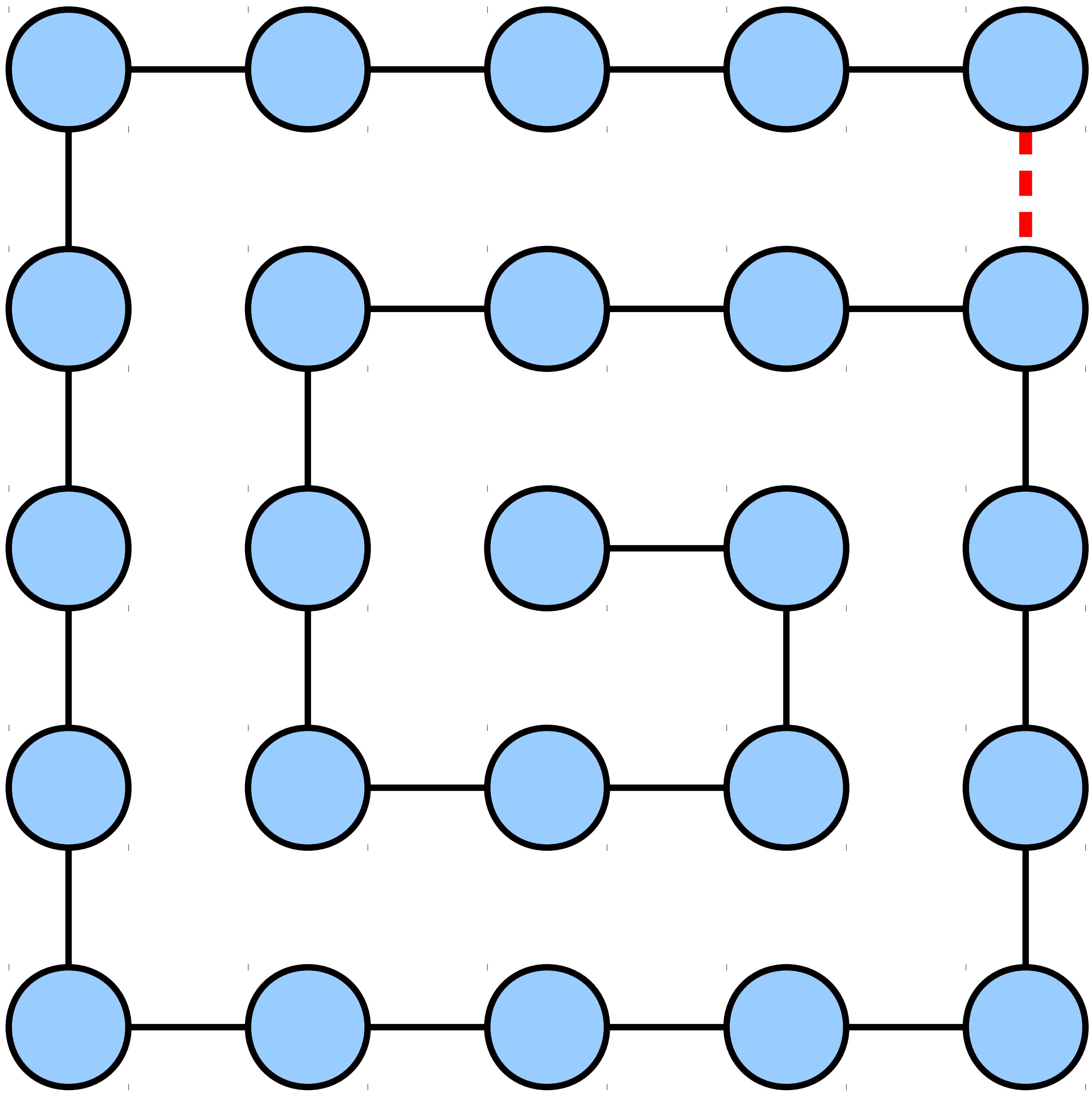


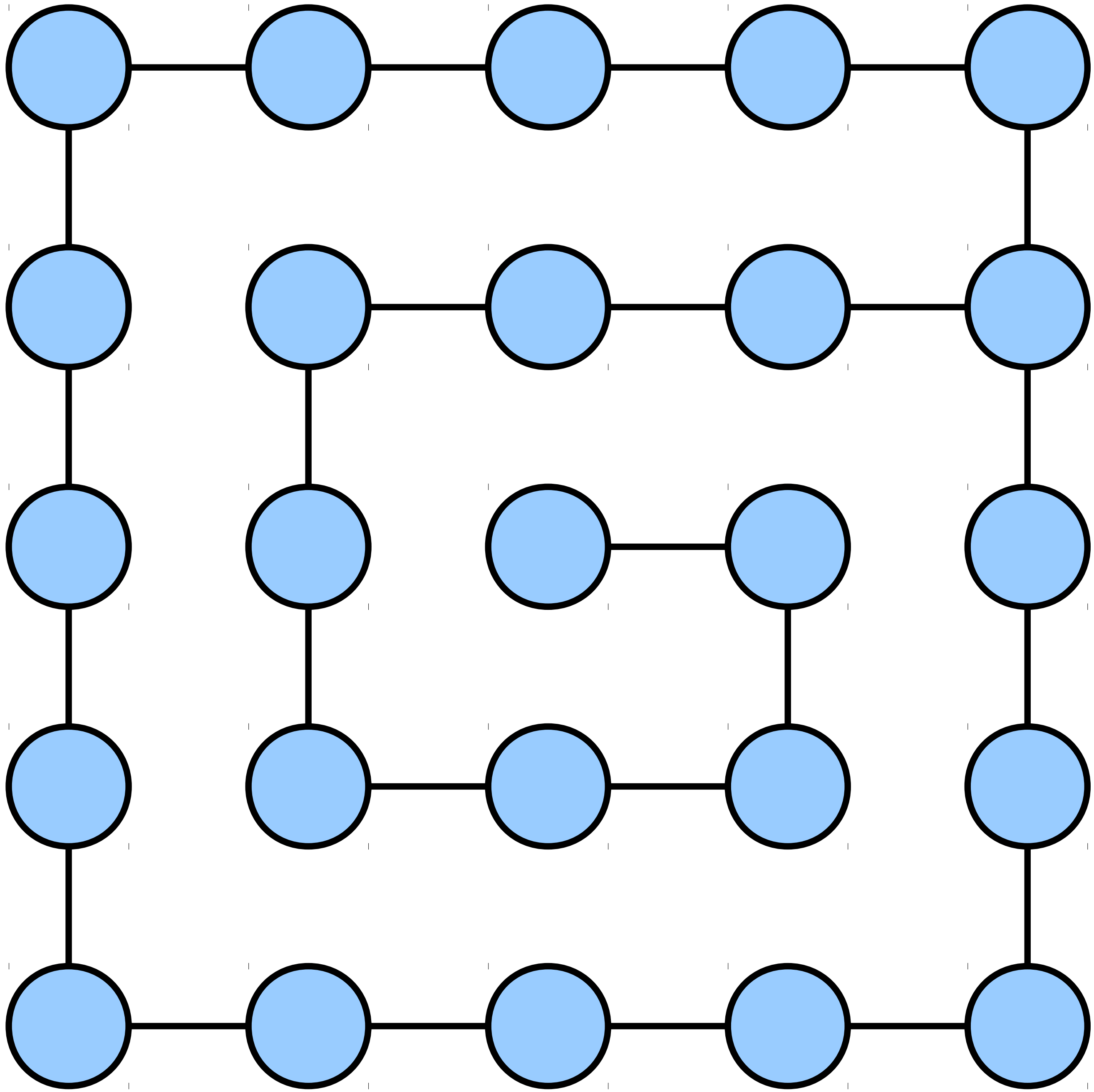


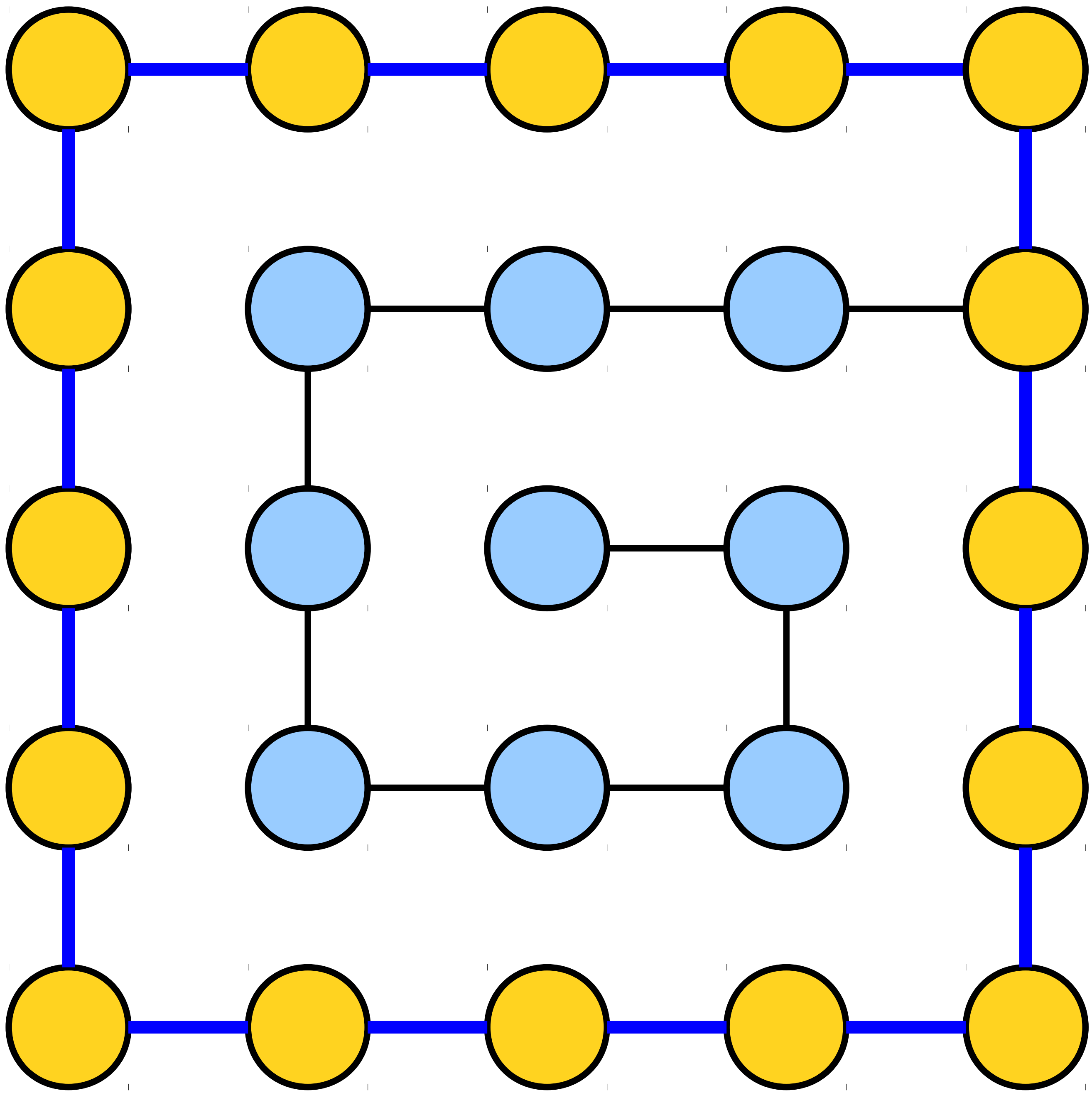






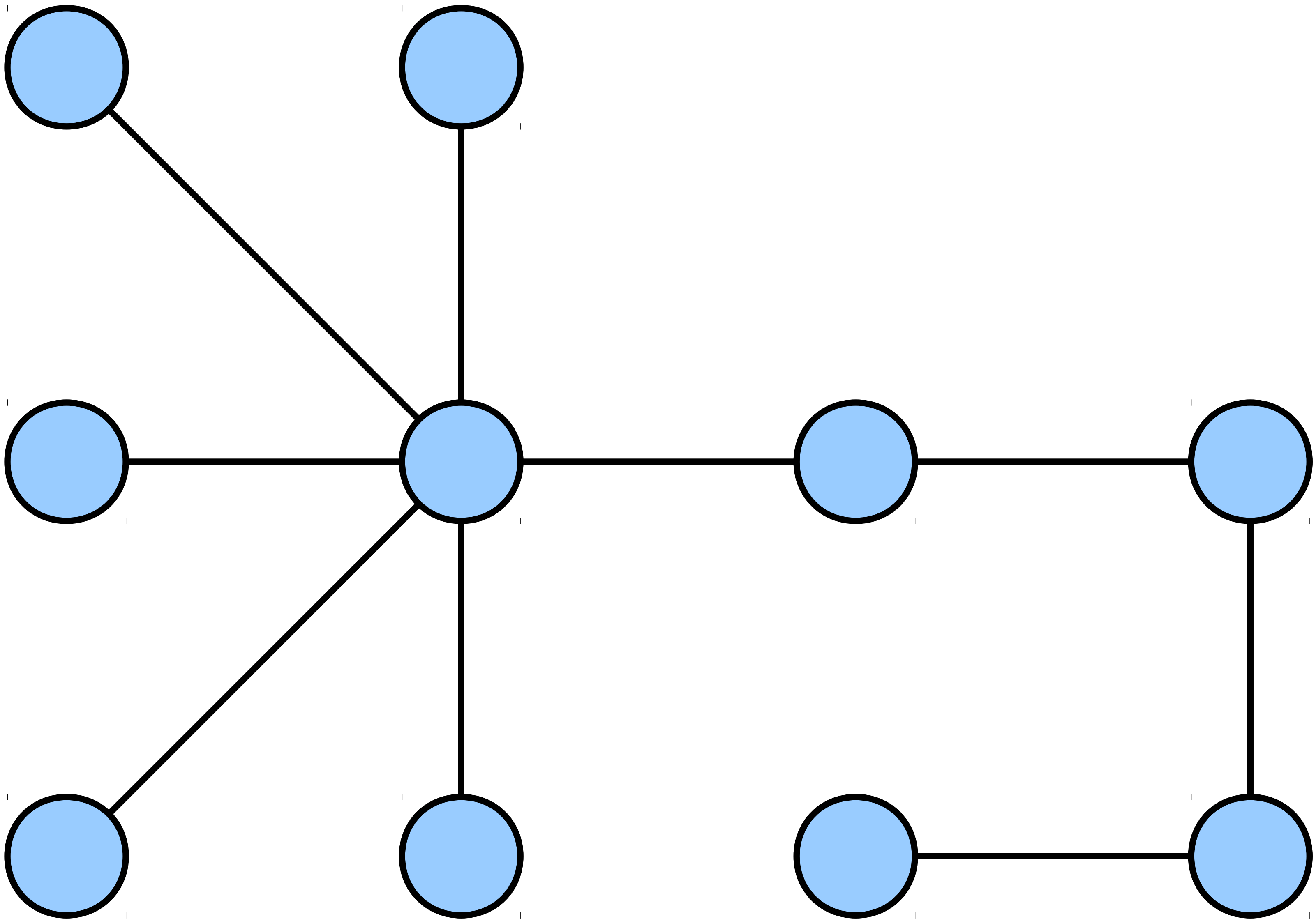


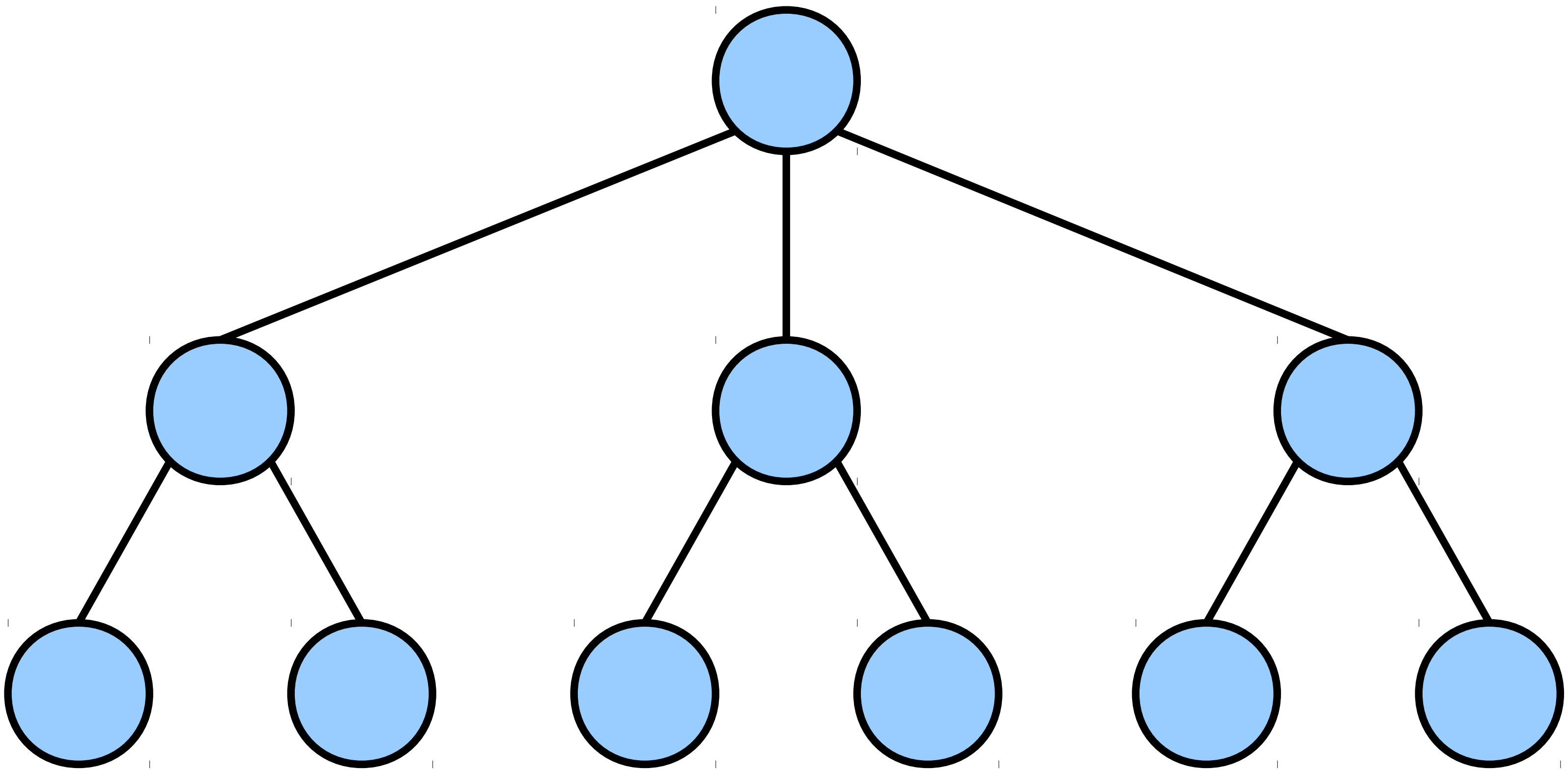


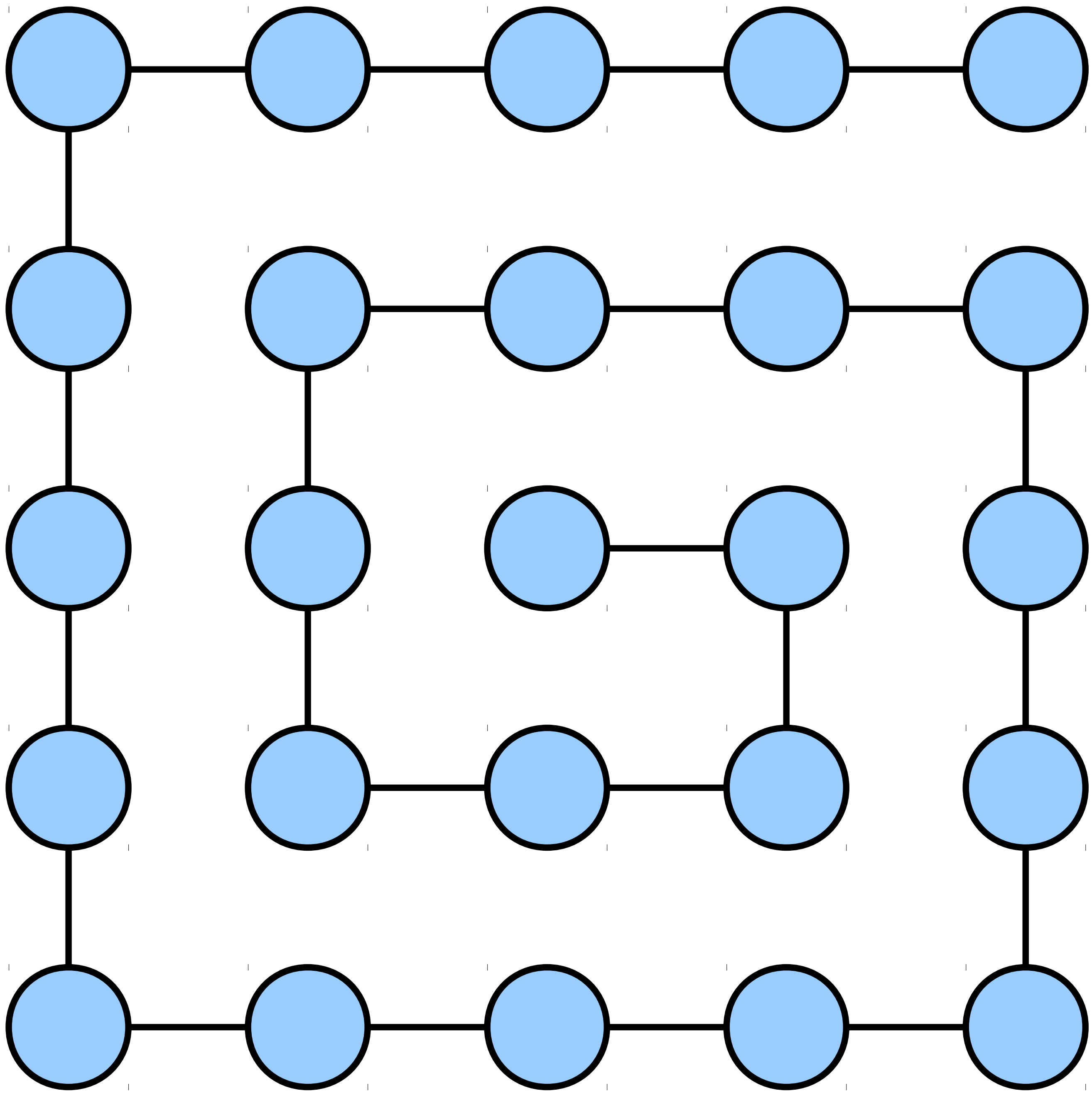


An undirected graph is called **maximally acyclic** iff adding any missing edge introduces a cycle.

Theorem: An undirected graph is a tree iff it is maximally acyclic.



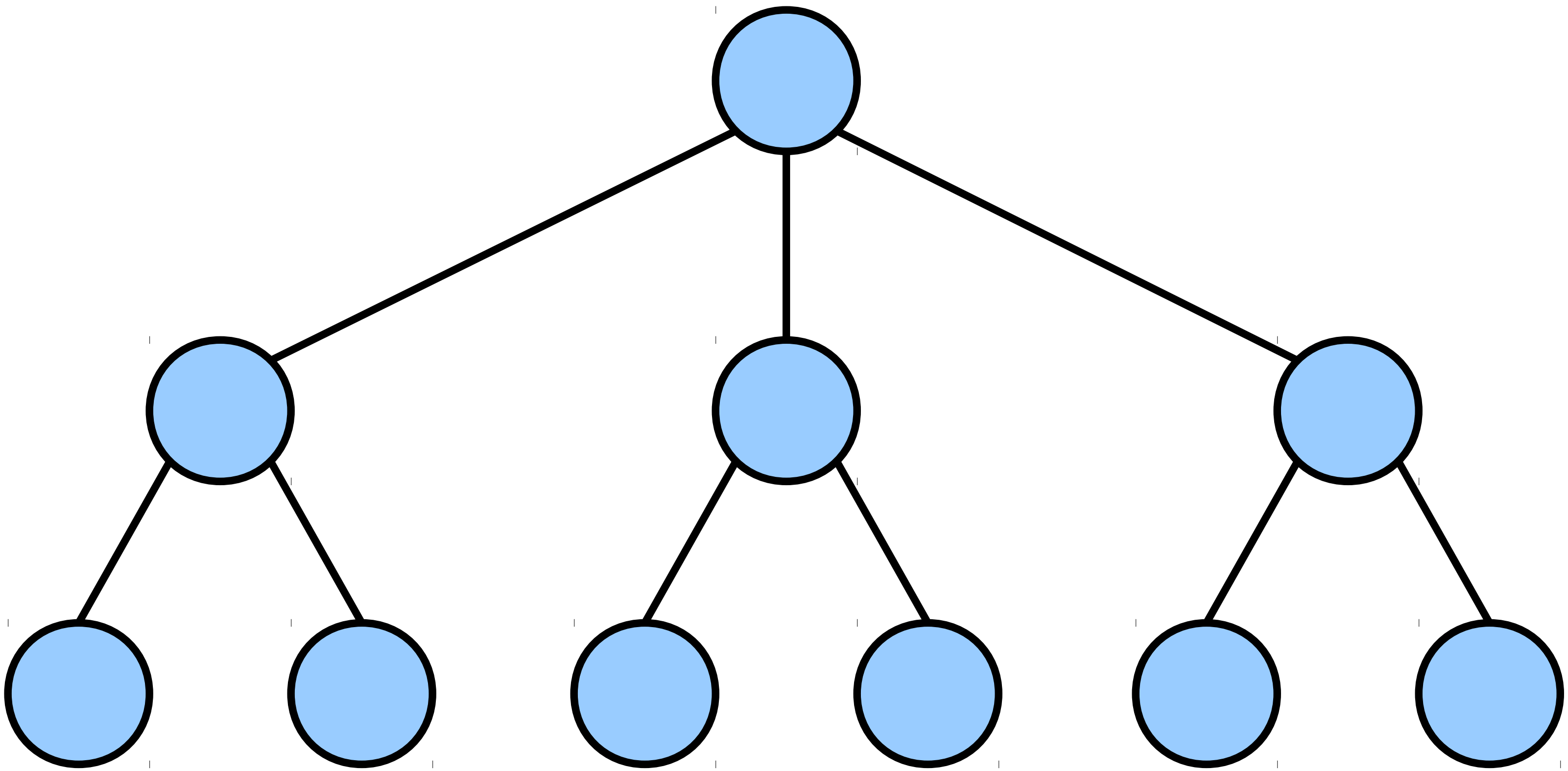


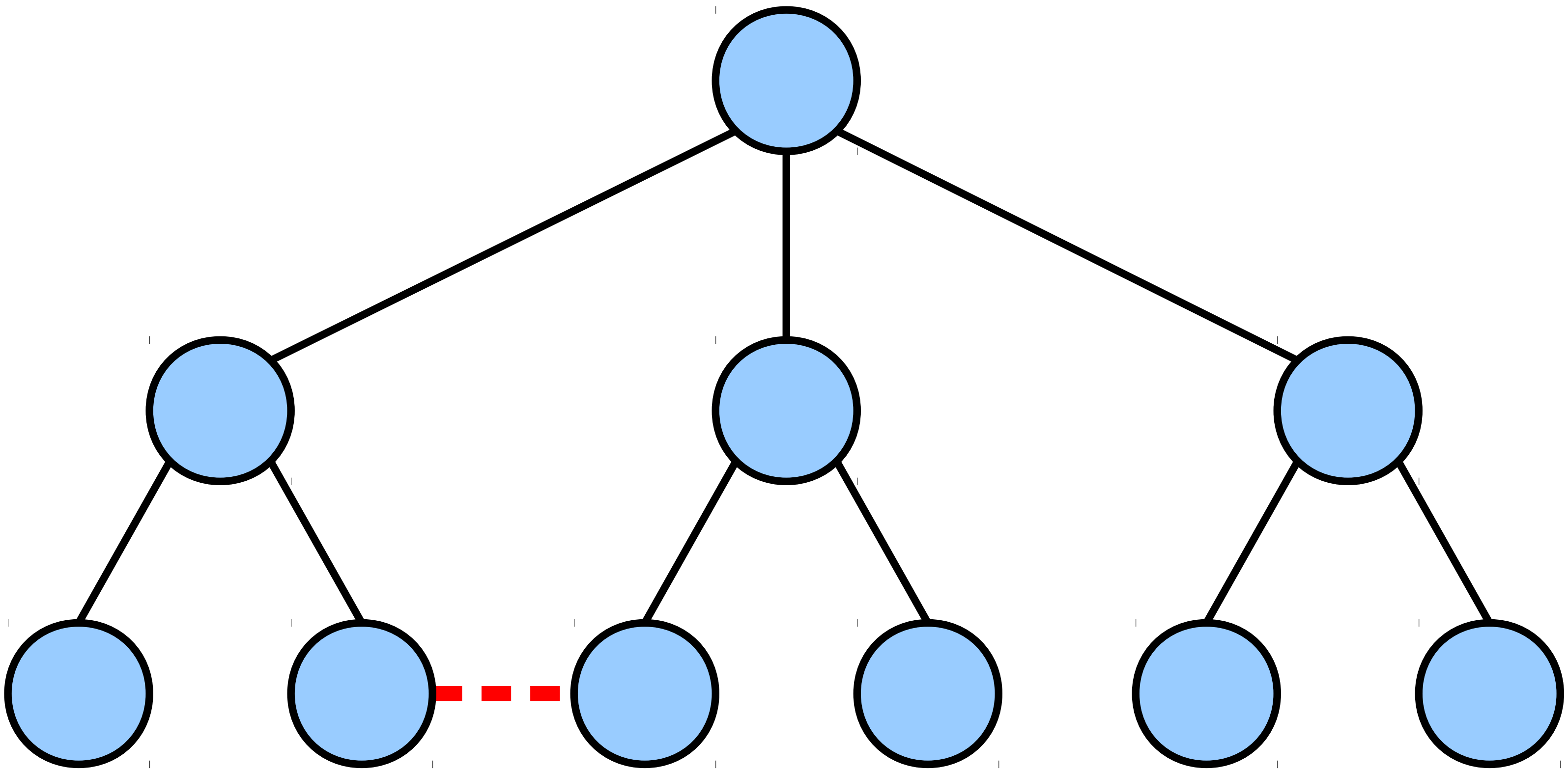


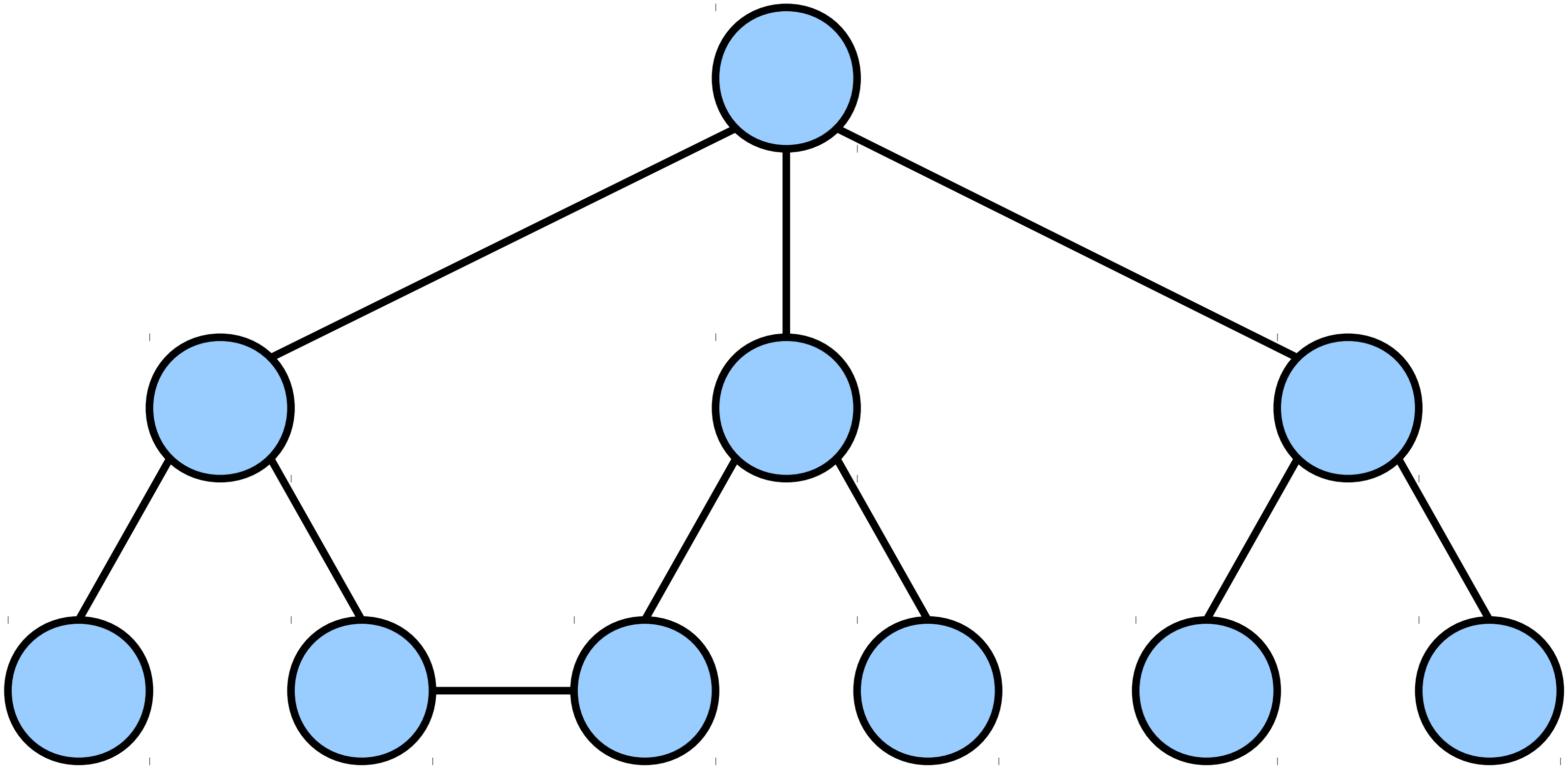
Theorem: An undirected graph is a tree iff it is connected and $|E| = |V| - 1$.

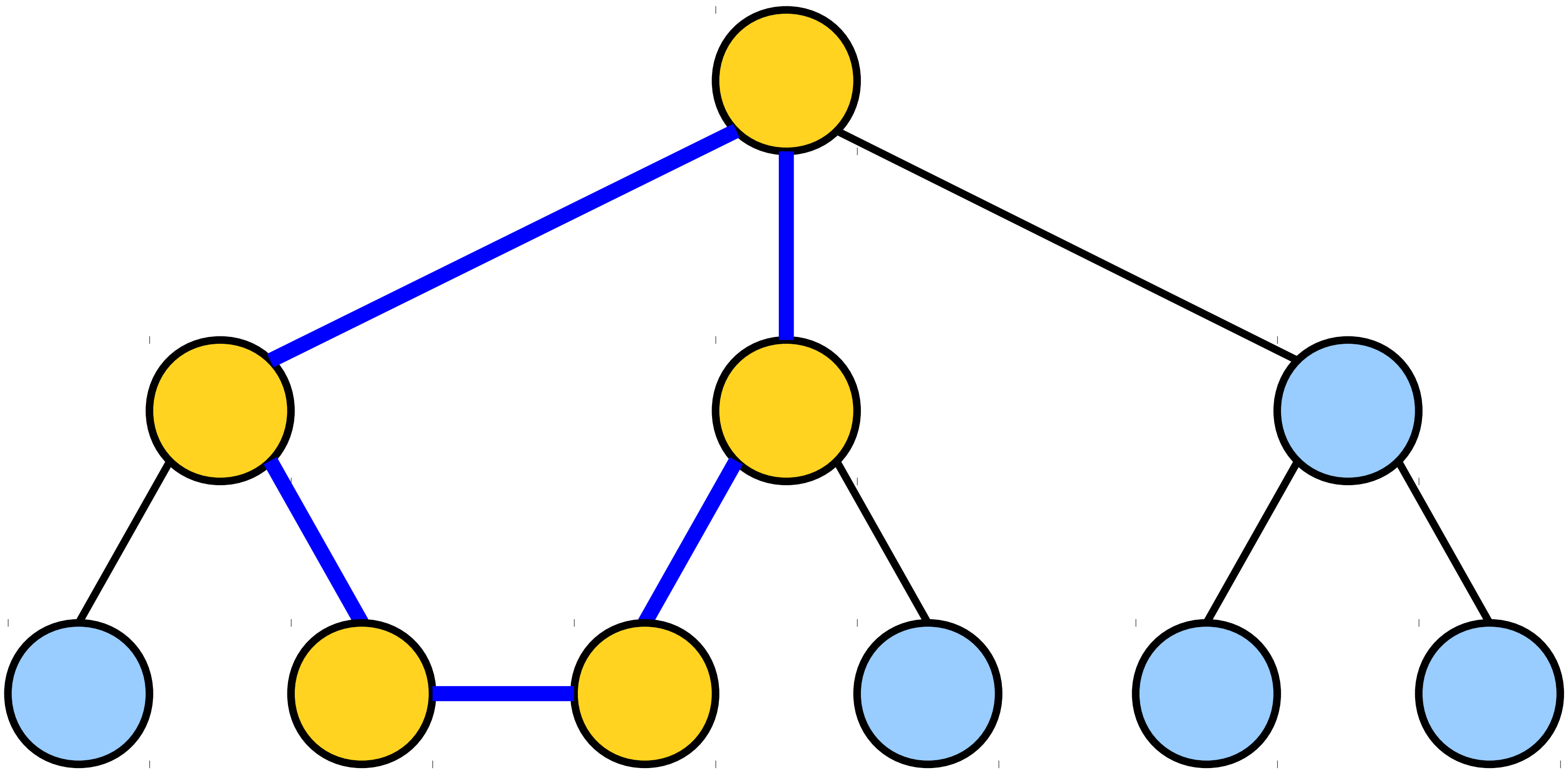
Trees

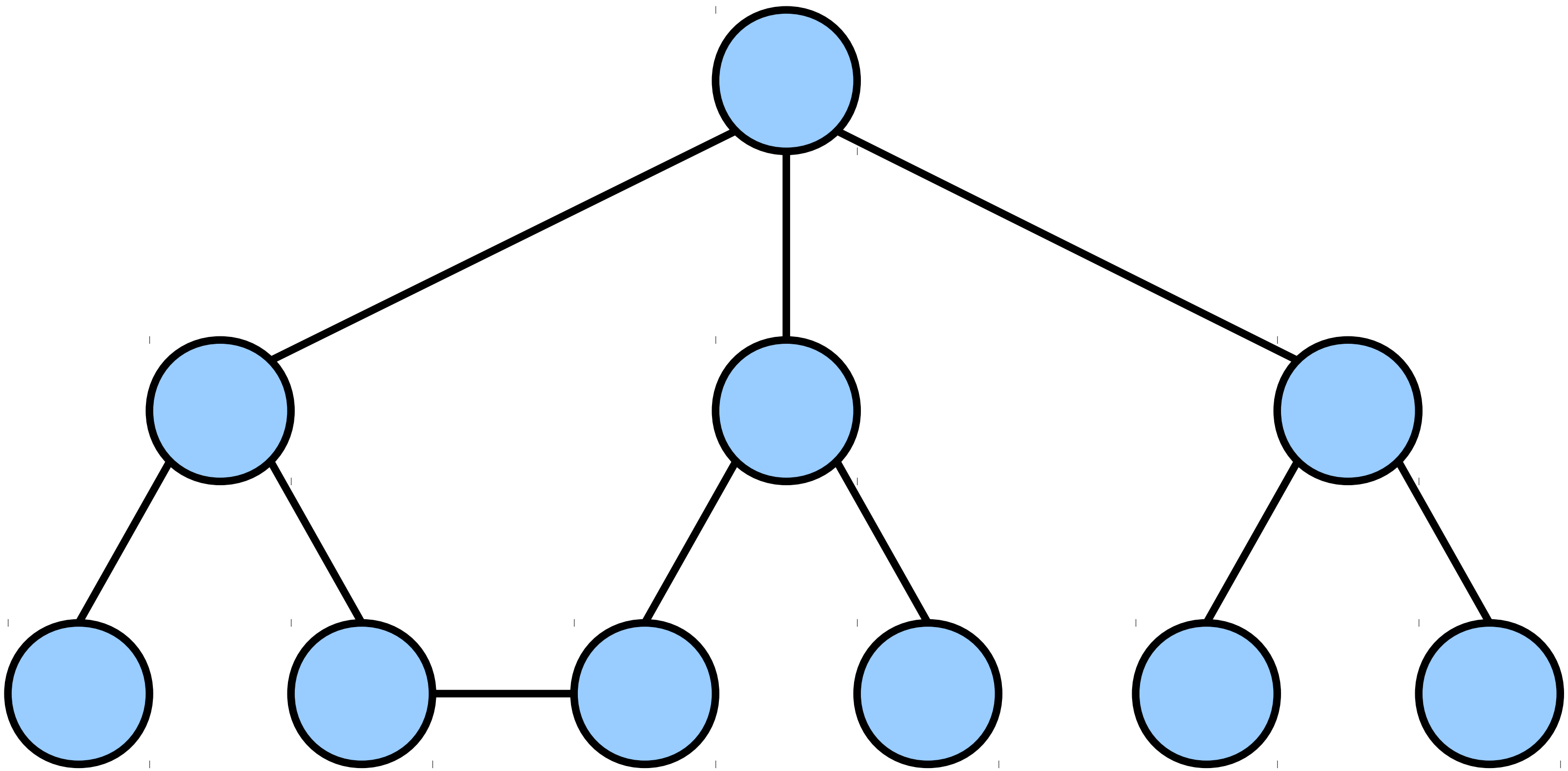
- A **tree** is an undirected graph $G = (V, E)$ that is connected and acyclic.
- All the following are equivalent:
  - G is a tree.
  - G is connected and acyclic.
  - G is **minimally connected** (removing any edge from G disconnects it.)
  - G is **maximally acyclic** (adding any edge creates a cycle)
  - G is connected and $|E| = |V| - 1$.

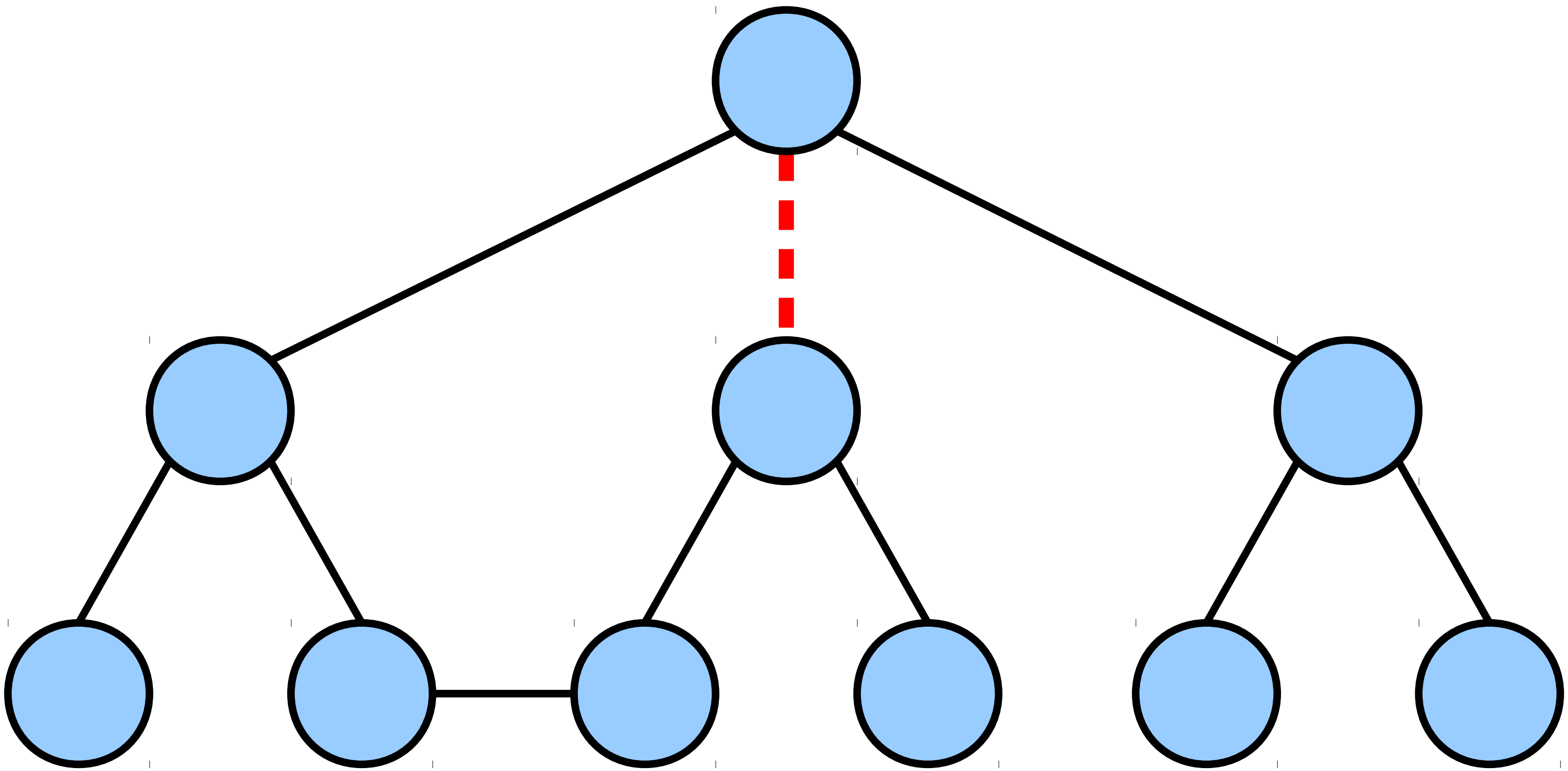


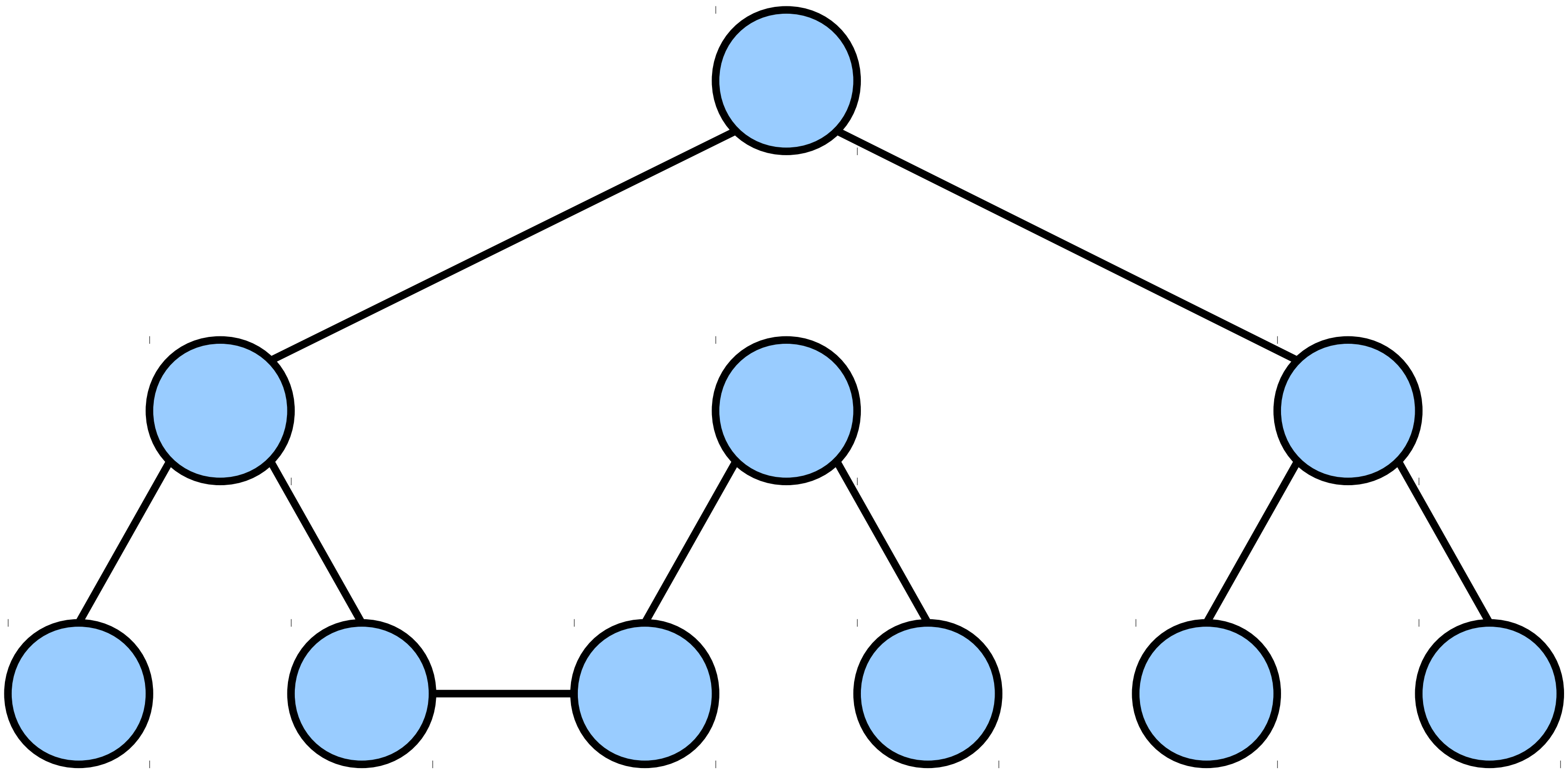


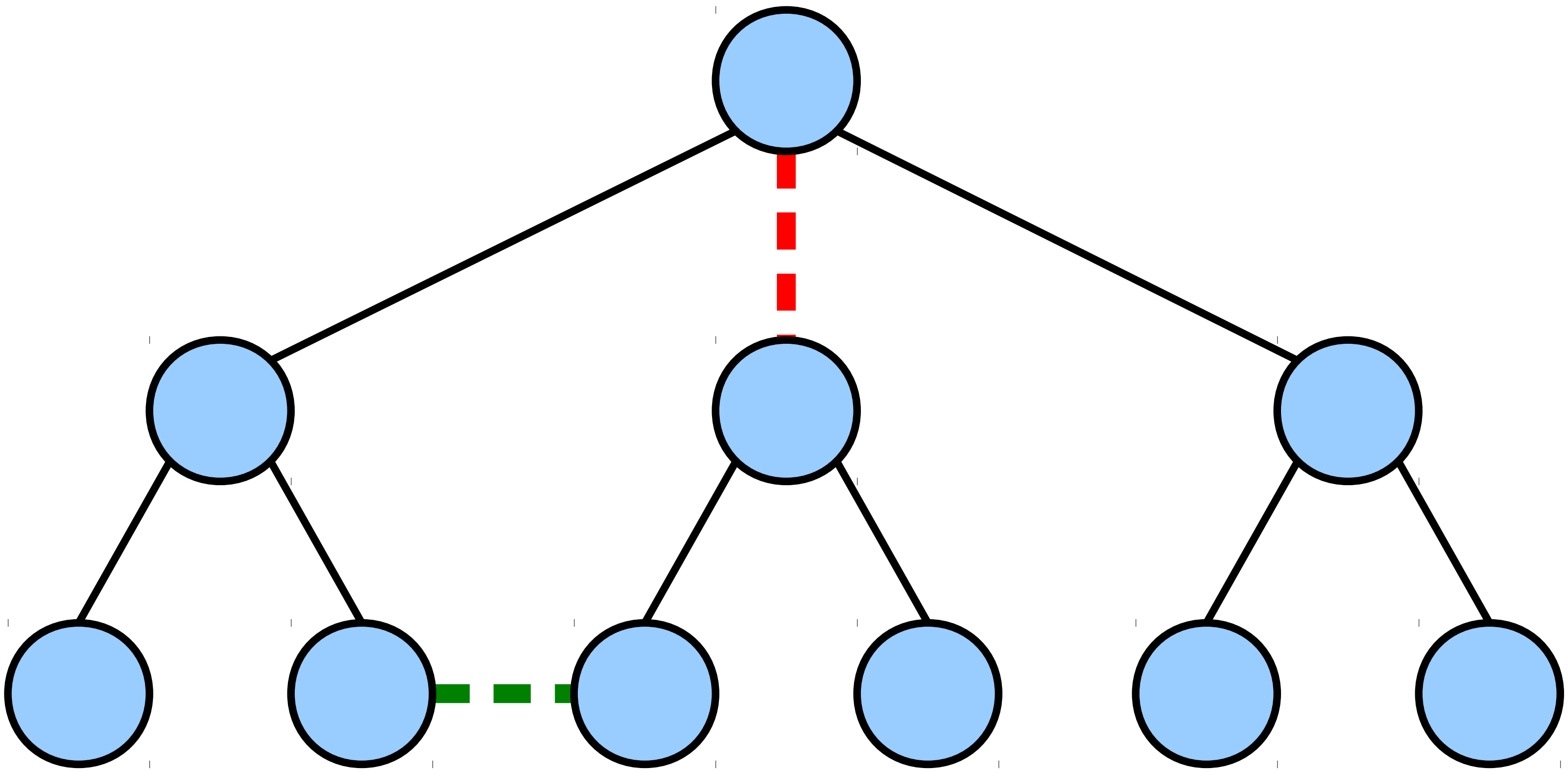










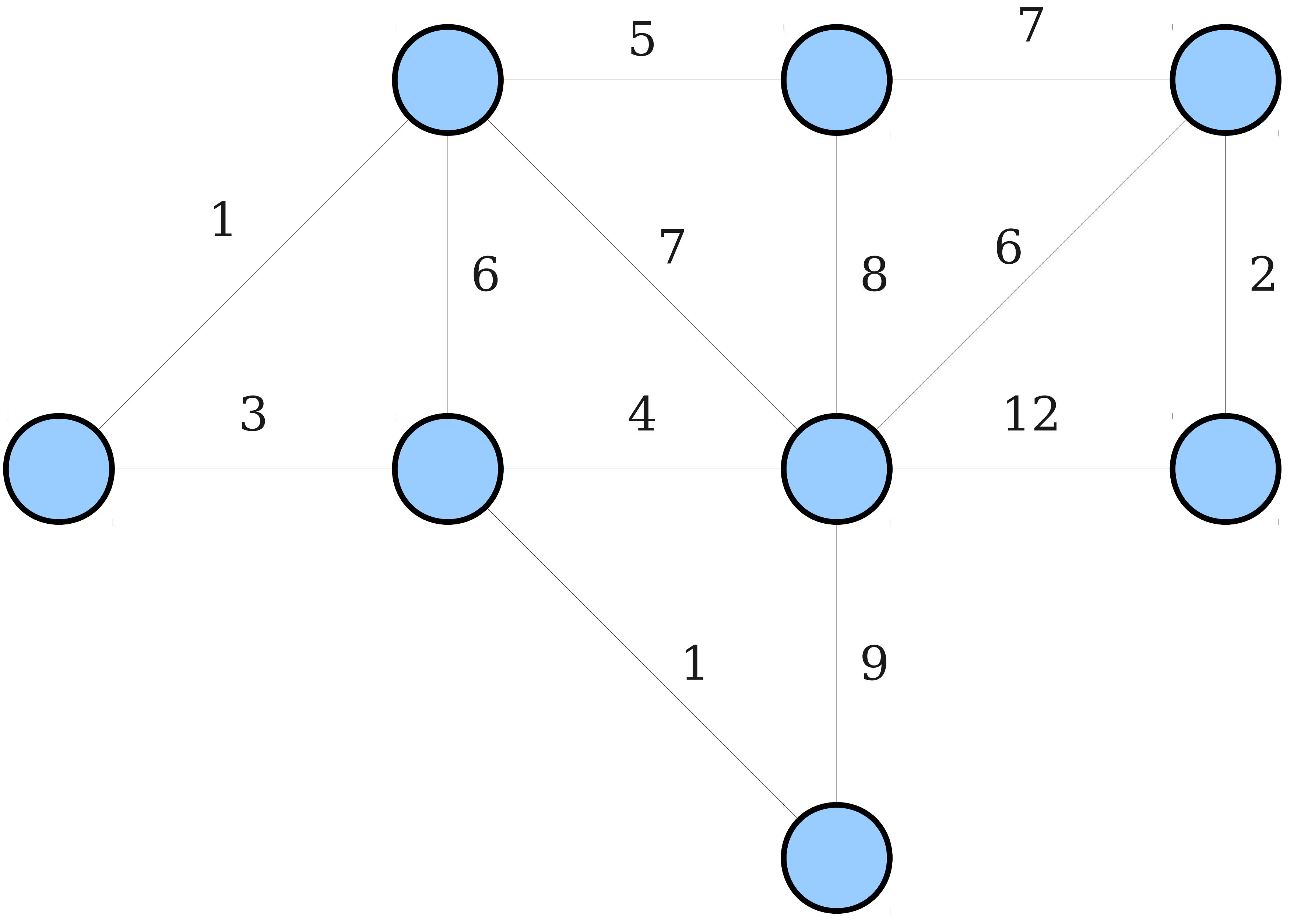


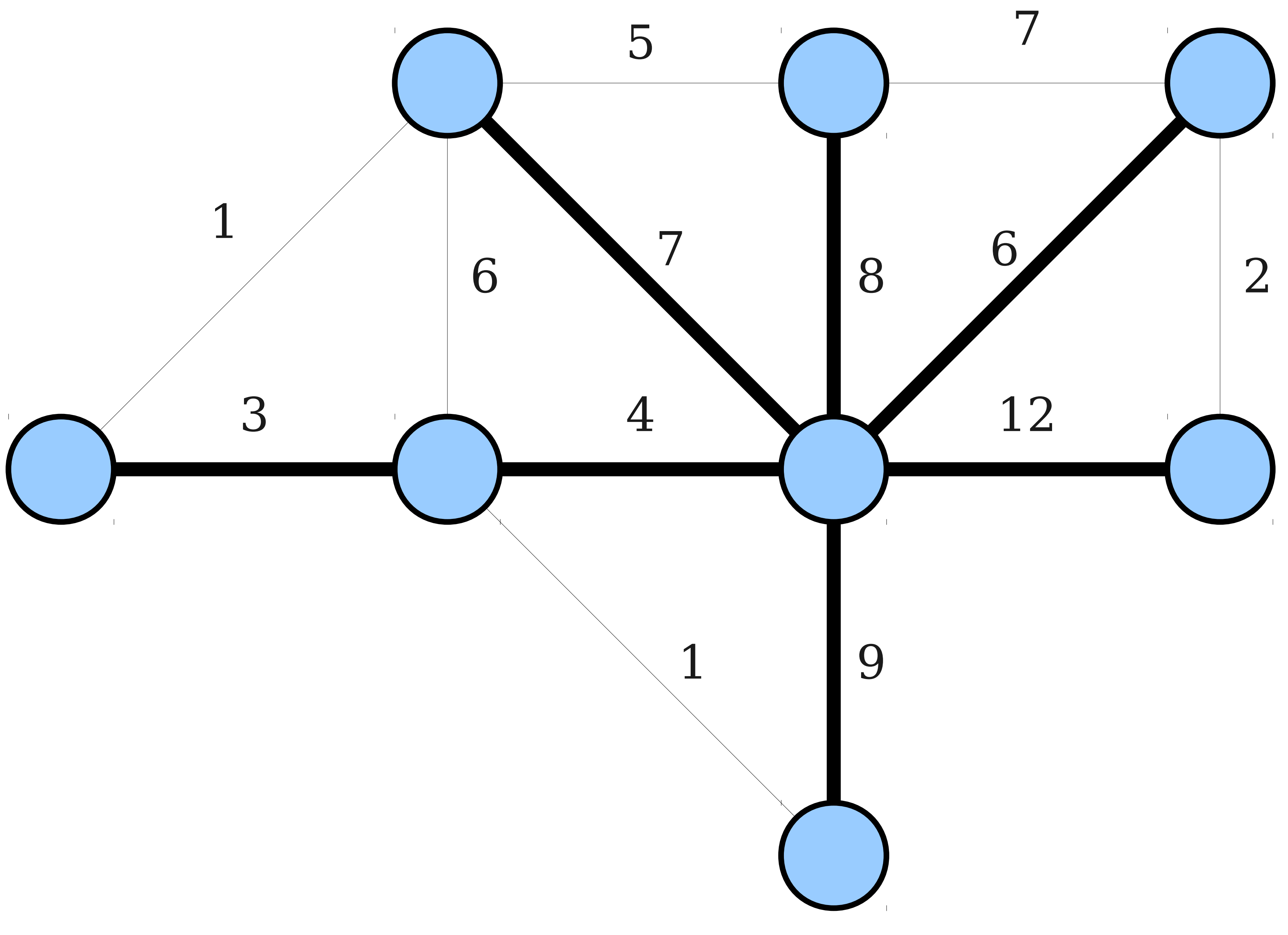
Theorem: Let T be a tree and $(u, v) \notin T$. The graph $T \cup \{(u, v)\}$ contains a cycle. For any edge (x, y) on the cycle, the graph $T' = T \cup \{(u, v)\} - \{(x, y)\}$ is a tree.

Proof: Since $(u, v) \notin T$ and $(x, y) \in T \cup \{(u, v)\}$, we know $|T'| = |T| + 1 - 1 = |T| = |V| - 1$. Therefore, we will show that T' is connected to conclude T' is a tree.

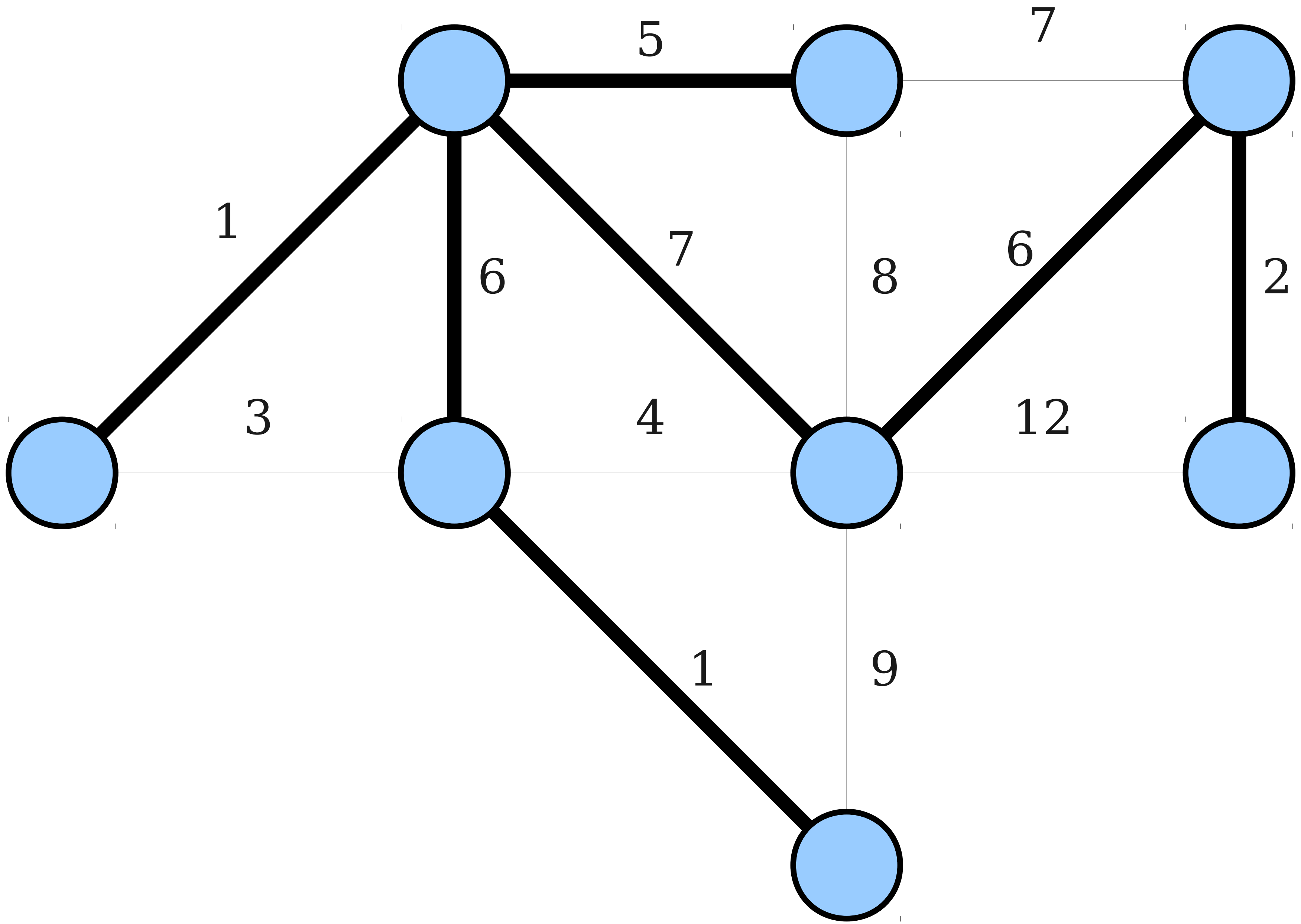
Consider any $s, t \in V$. Since T is connected, there is some path from s to t in T. If that path does not cross (x, y), or if $(x, y) = (u, v)$, then this path is also a path from s to t in T', so s and t are connected in T'. Otherwise, suppose the path from s to t crosses (x, y). Assume without loss of generality that the path starts at s, goes to x, crosses (x, y), then goes from y to t. Since (u, v) and (x, y) are part of the same cycle, we can modify the original path from s to t so that instead of crossing (x, y), it goes around the cycle from x to y. This new path is then a path from s to t in T', so s and t are connected in T'. Thus any arbitrary pair of nodes are connected in T', so T' is connected. ■

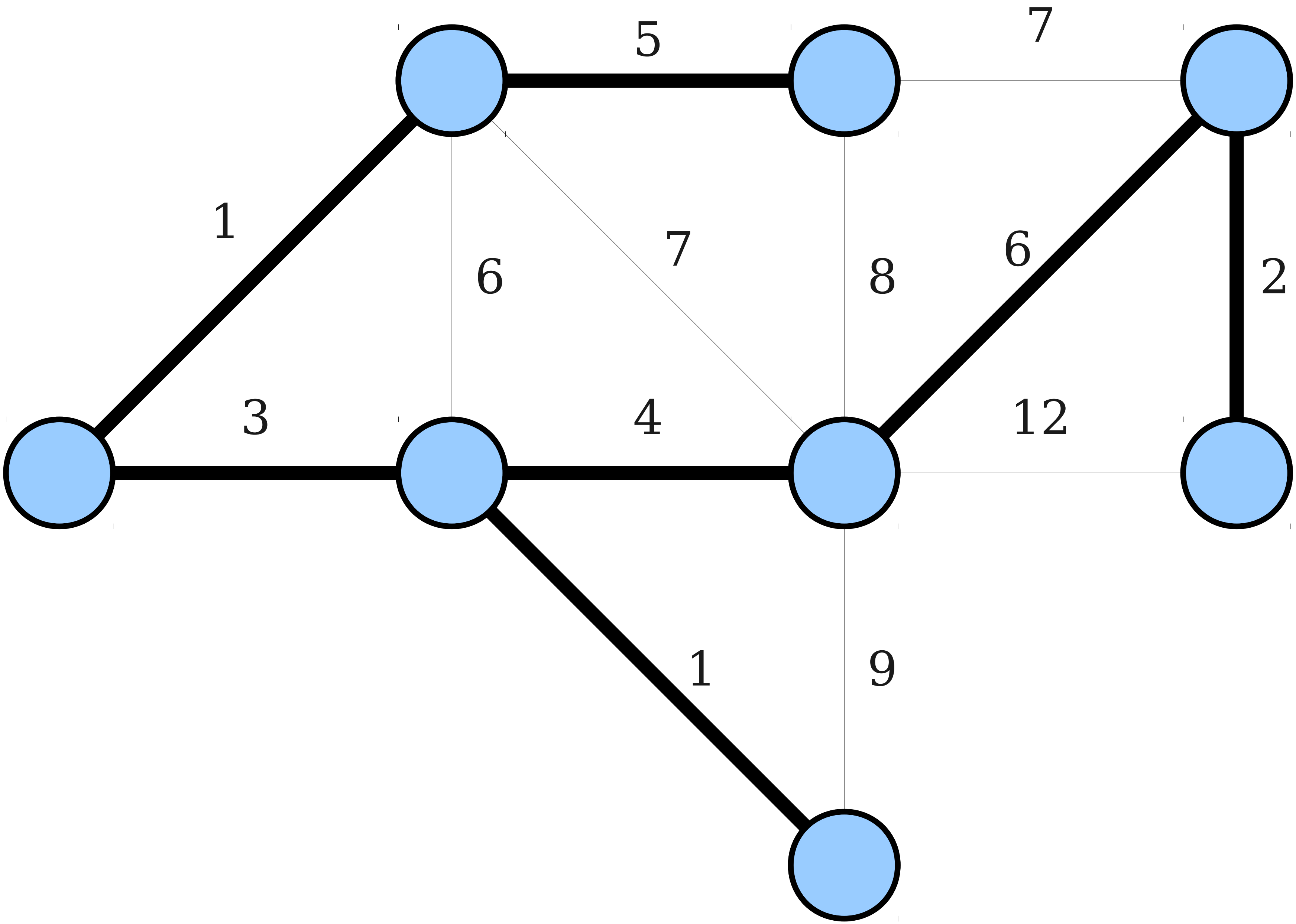
Minimum Spanning Trees


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



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


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


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

Spanning Trees

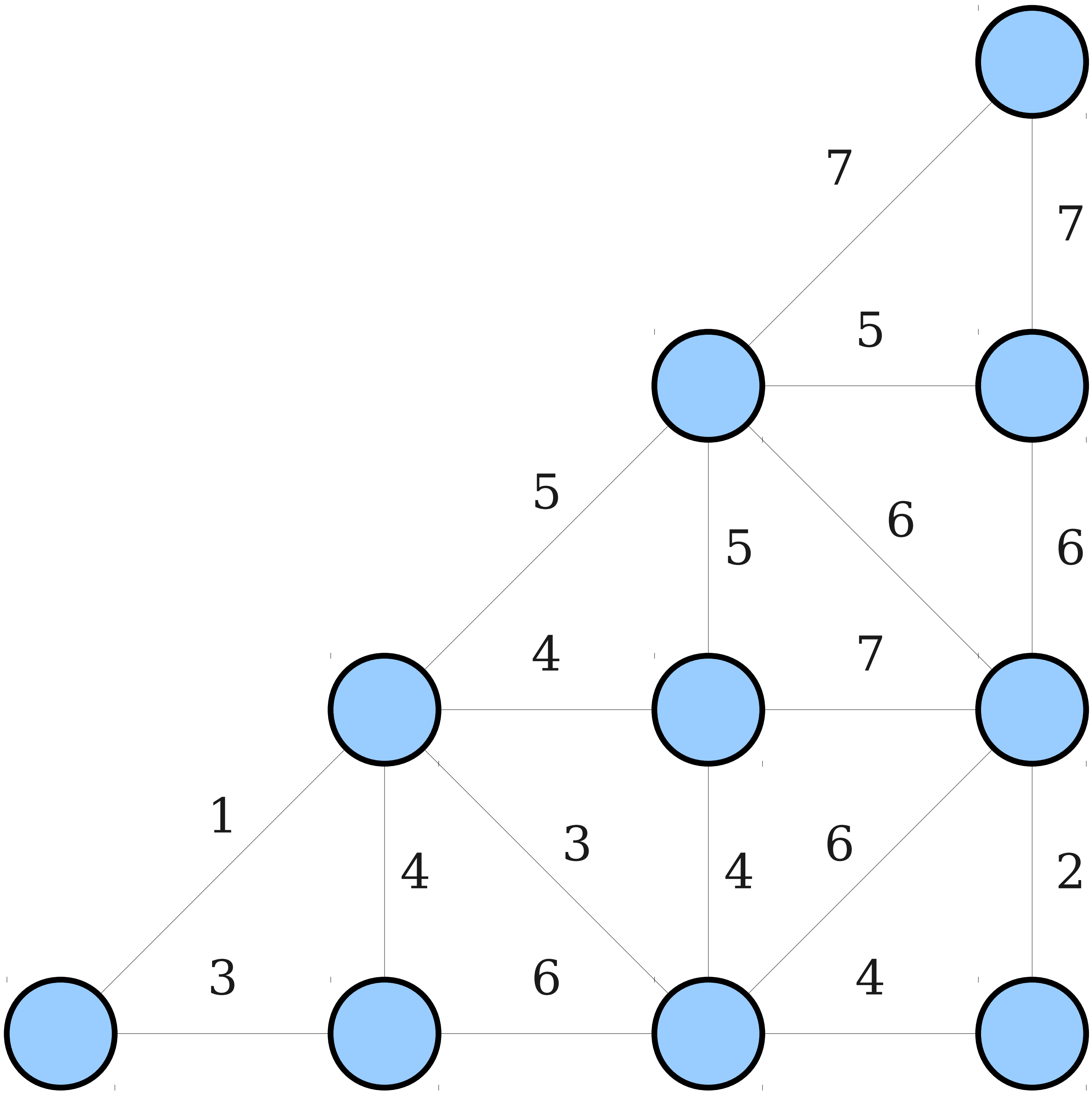
- Let $G = (V, E)$. A **spanning tree** (or **ST**) of G is a graph (V, T) such that (V, T) is a tree.
  - For notational simplicity: we'll identify a spanning tree with just the set of edges T.
- Suppose that each edge $(u, v) \in E$ is assigned a **cost** $c(u, v)$.
- The **cost of a tree** T, denoted $c(T)$, is the sum of the costs of the edges in T:

$$c(T) = \sum_{(u,v) \in T} c(u,v)$$

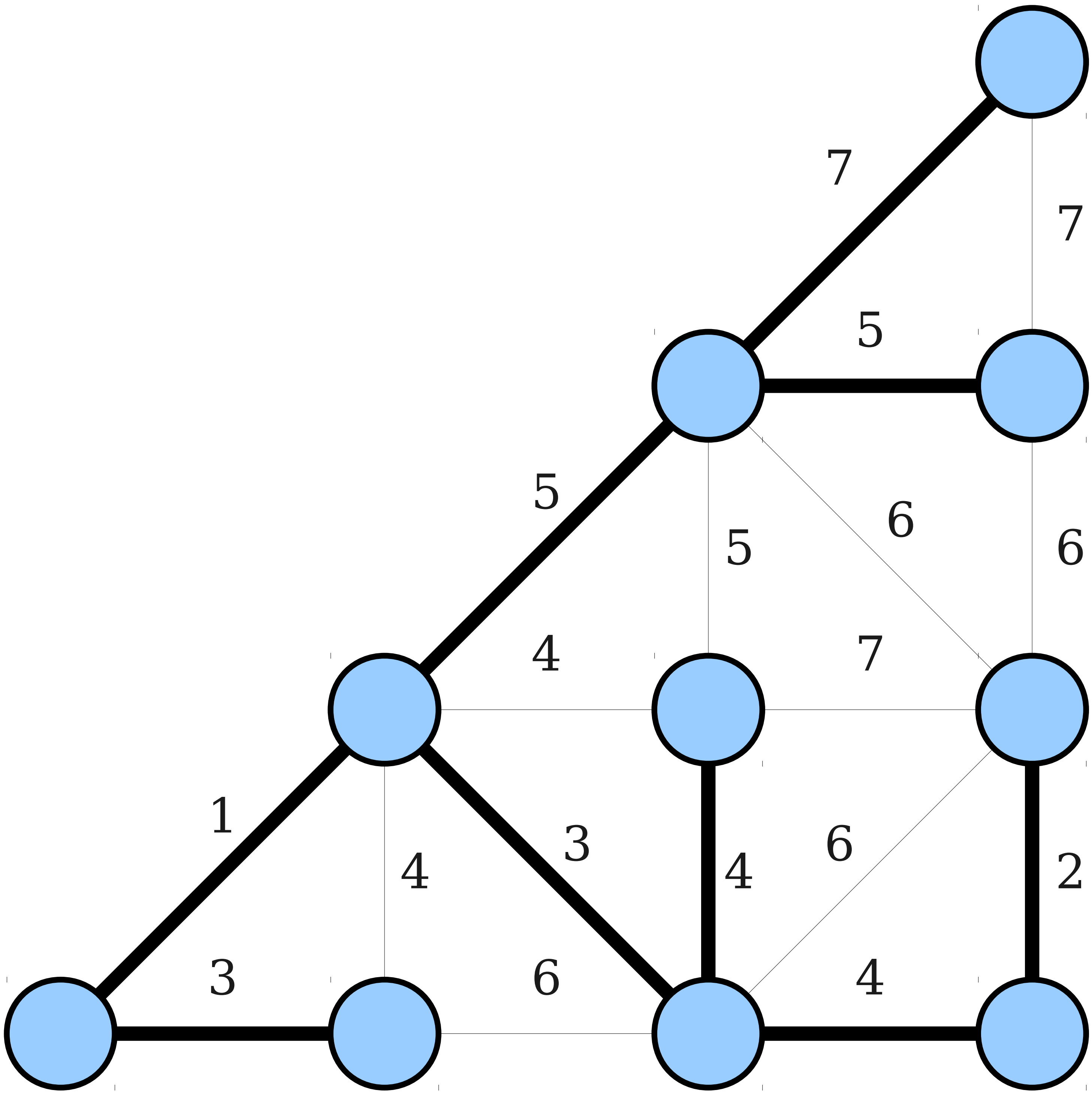
- A **minimum spanning tree** (or **MST**) of G is a spanning tree T^* of G with minimum cost.

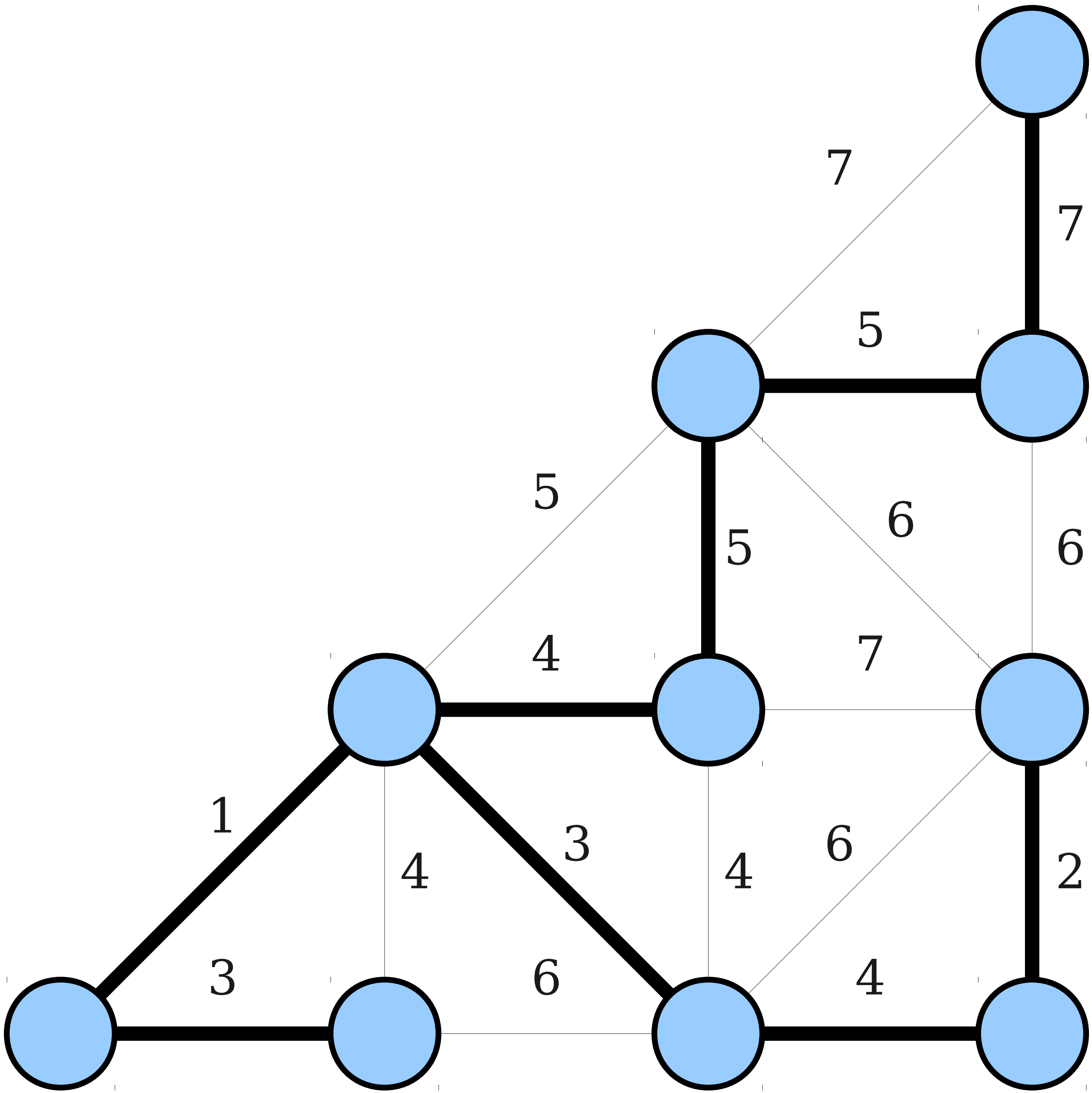
Minimum Spanning Trees

- There are *many* greedy algorithms for finding MSTs:
  - Borůvka's algorithm (1926)
  - Kruskal's algorithm (1956)
  - Prim's algorithm (1930, rediscovered 1957)
- We will explore Kruskal's algorithm and Prim's algorithm in this course.
- *Lots* of research into this problem: parallel implementions, optimal serial implementations, implementations harnessing bitwise operations, etc...

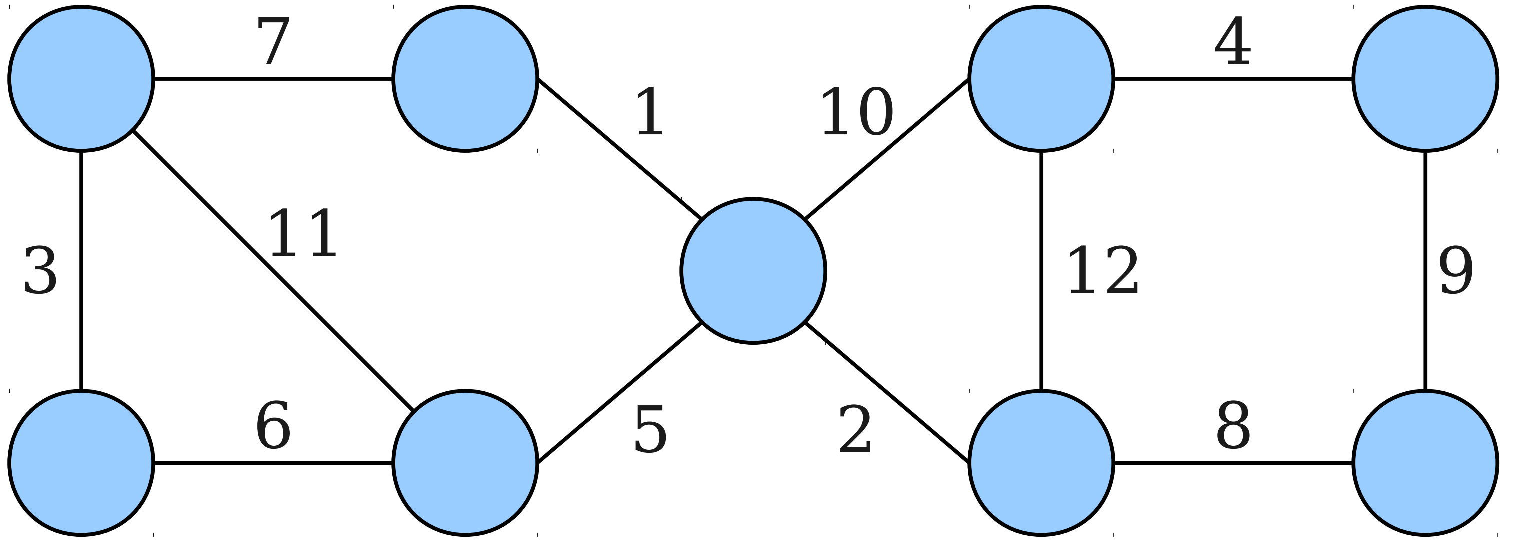


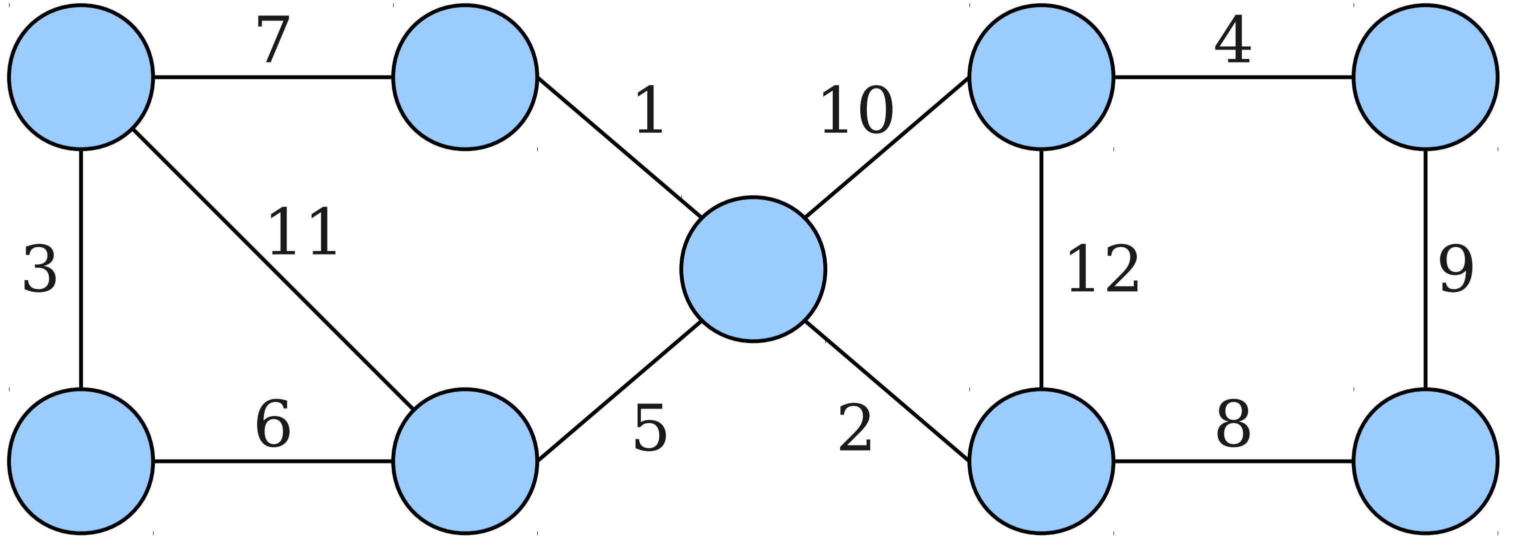
7
7
5
5
5
6
6
4
7
1
3
4
6
4
2
3
6
4

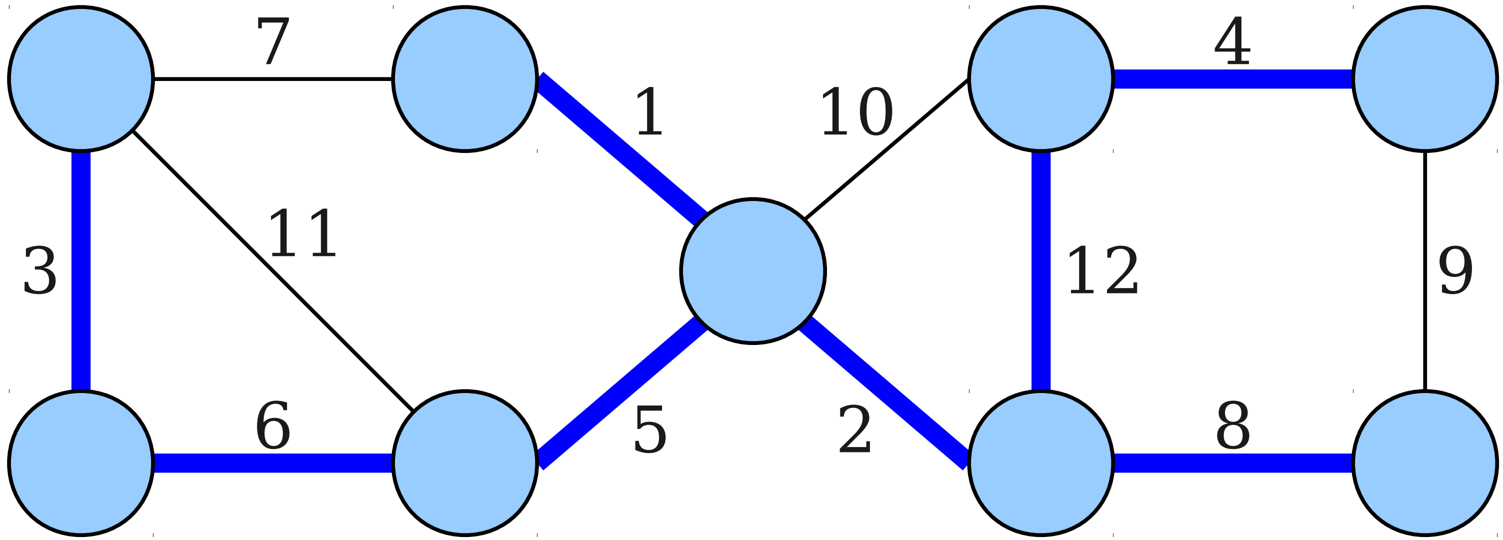

7
7
5
5
6
5
6
4
7
1
3
4
6
4
2
3
6
4

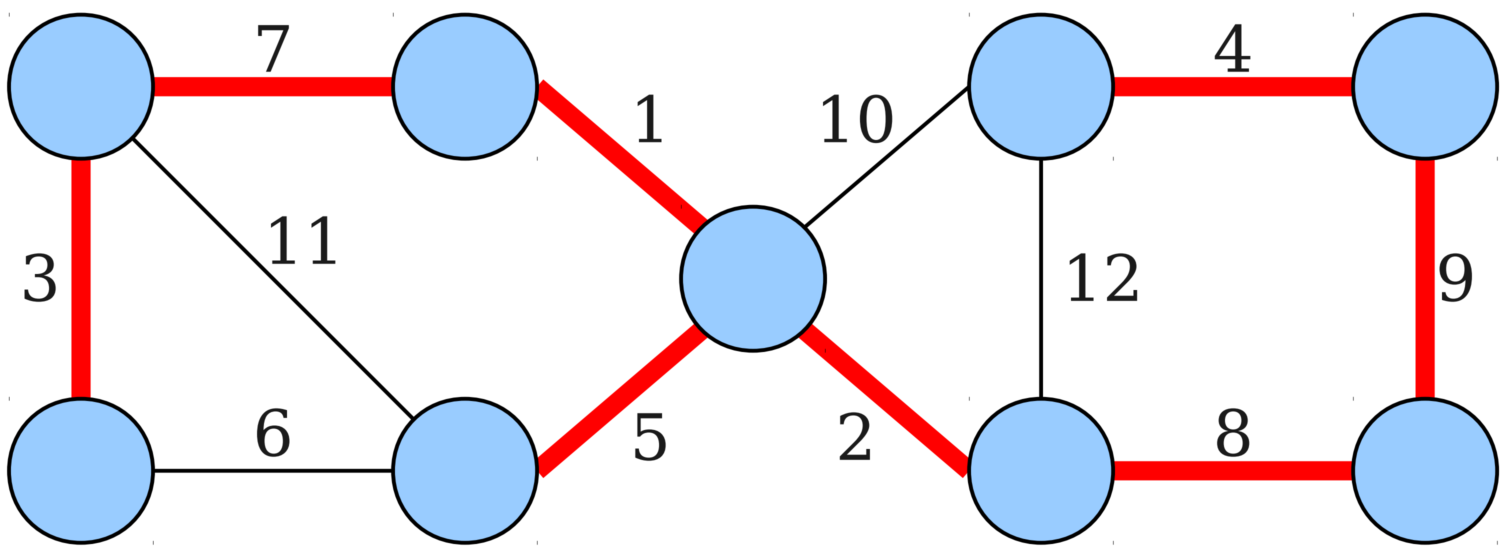


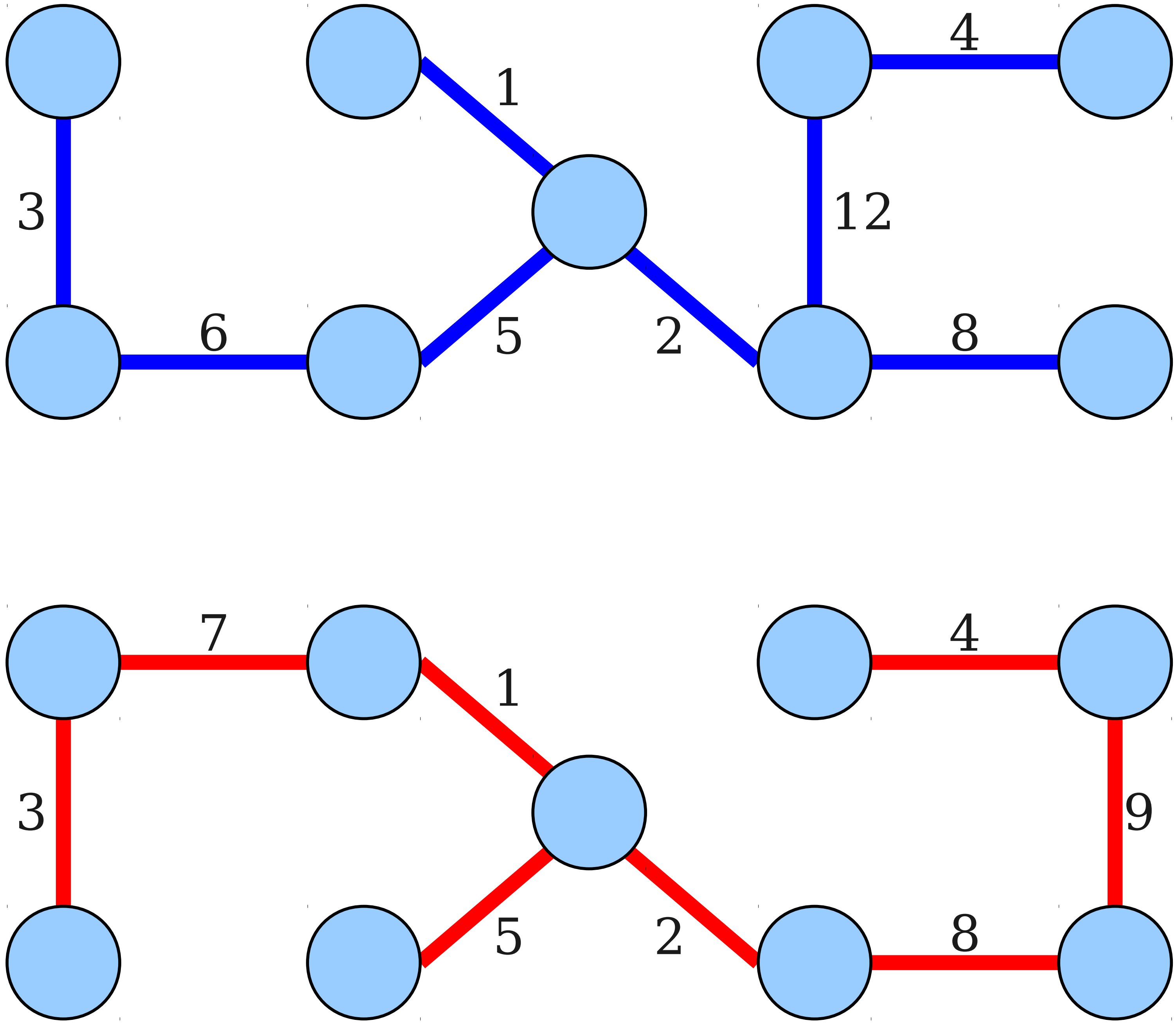
7
7
5
5
6
5
6
4
7
1
3
4
6
4
2
3
6
4

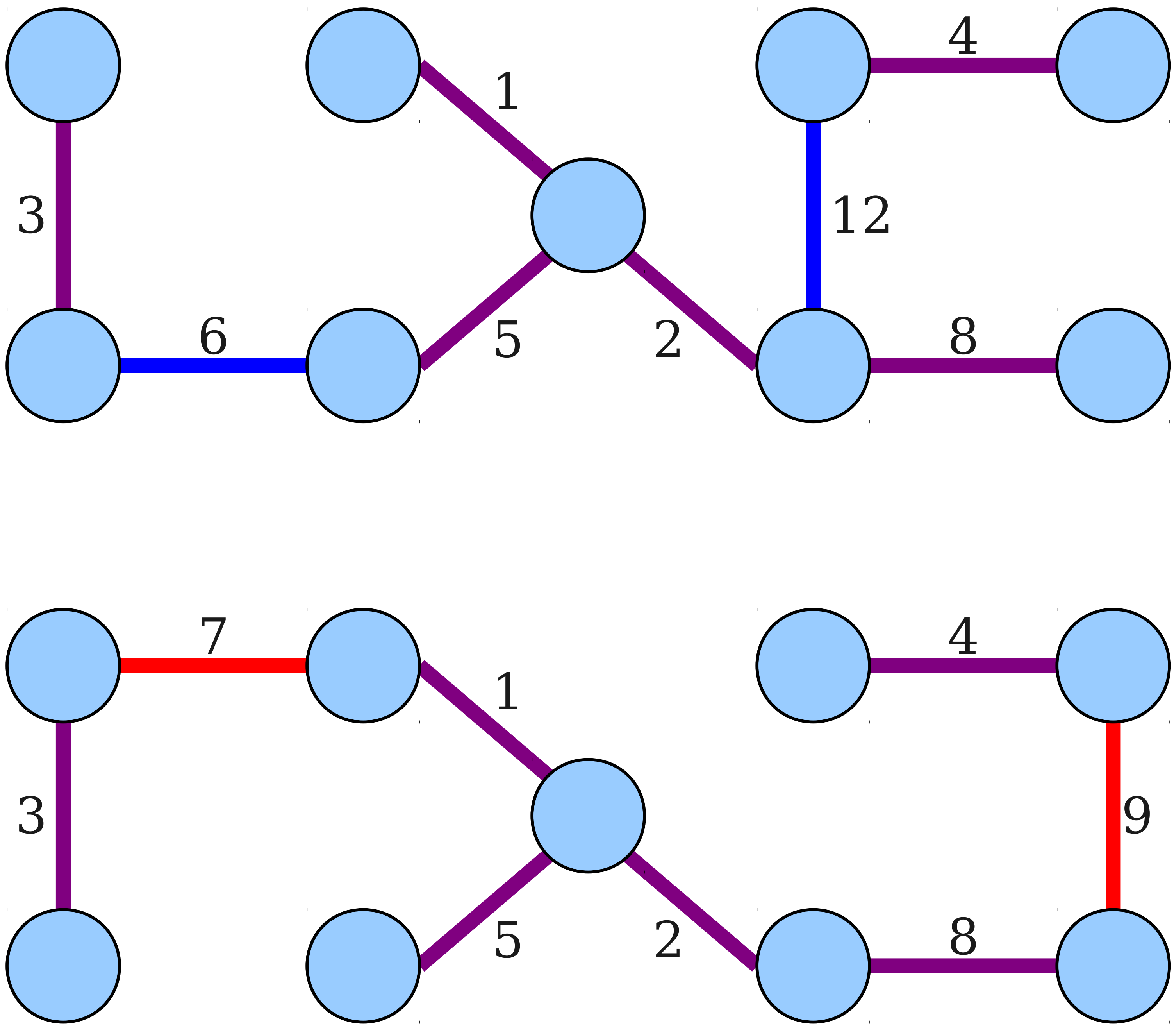

7
1
10
4
11
3
12
9
6
5
2
8


7
1
10
4
11
3
12
9
6
5
2
8

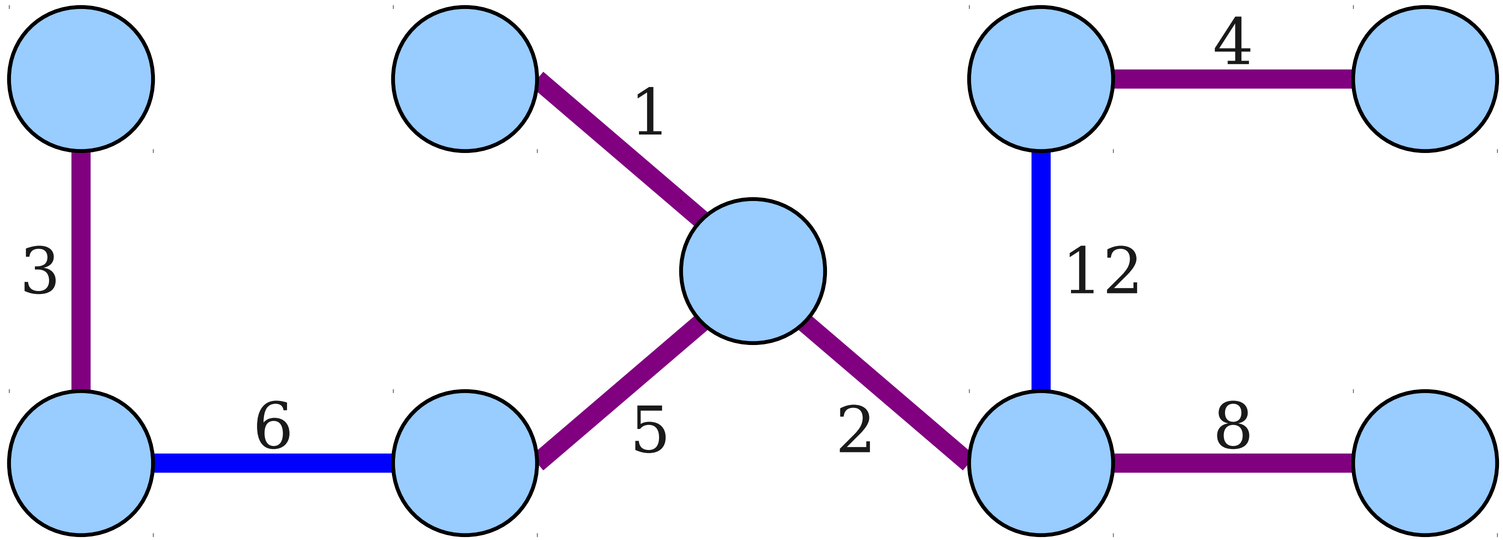

7
1
10
4
3
11
12
9
6
5
2
8


7
1
10
4
3
11
12
9
6
5
2
8

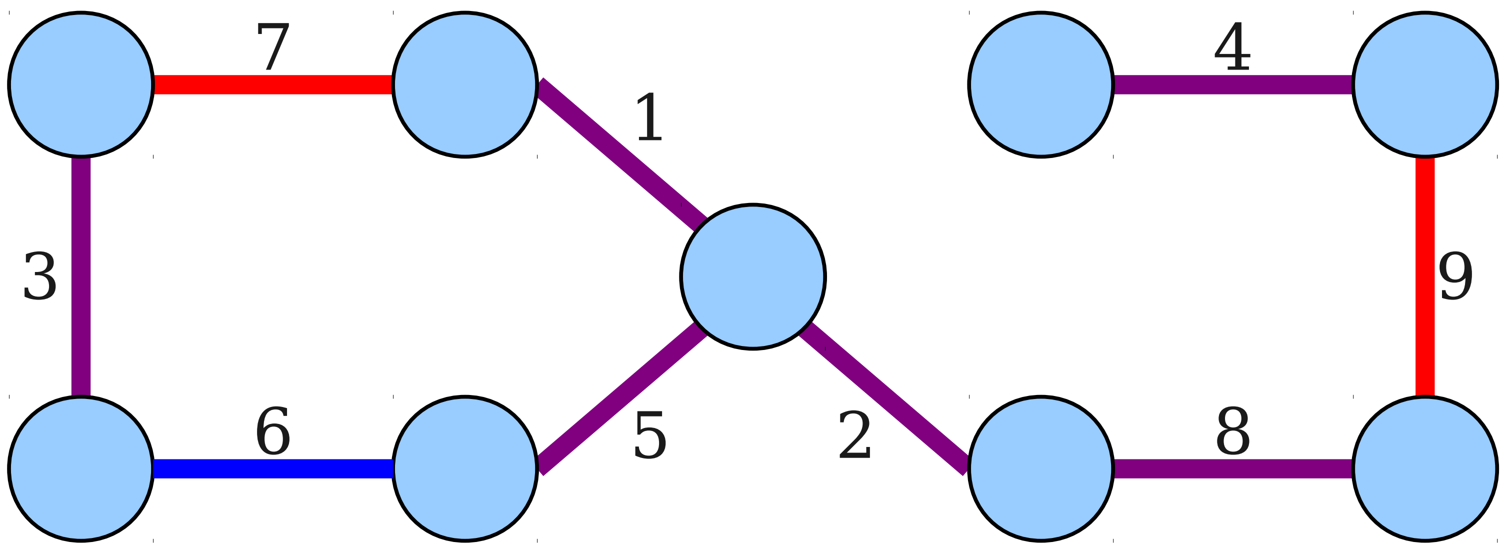

1
4
3
12
6
5
2
8
7
1
4
3
9
5
2
8



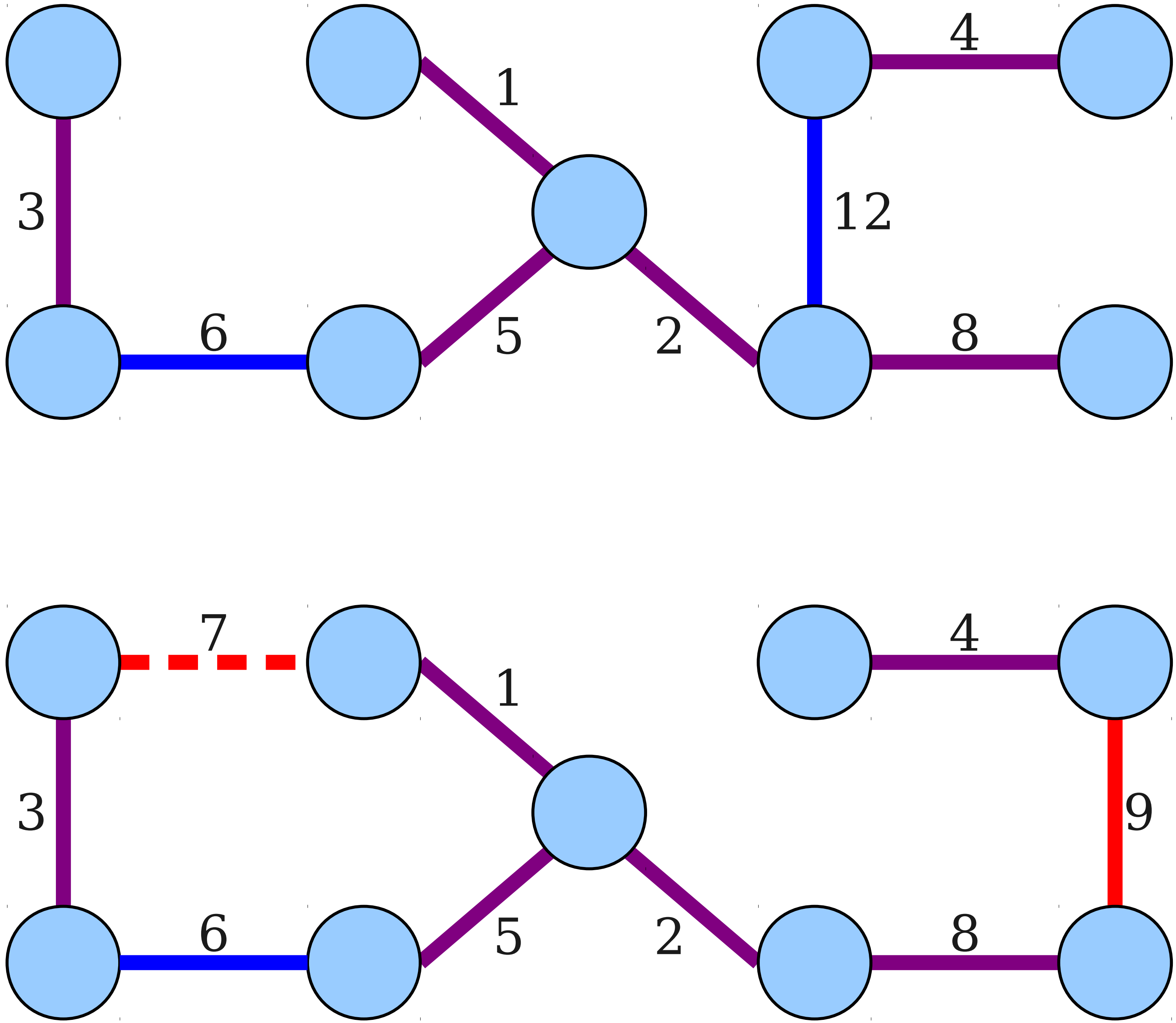
1
4
3
12
6
5
2
8
7
1
4
3
9
5
2
8



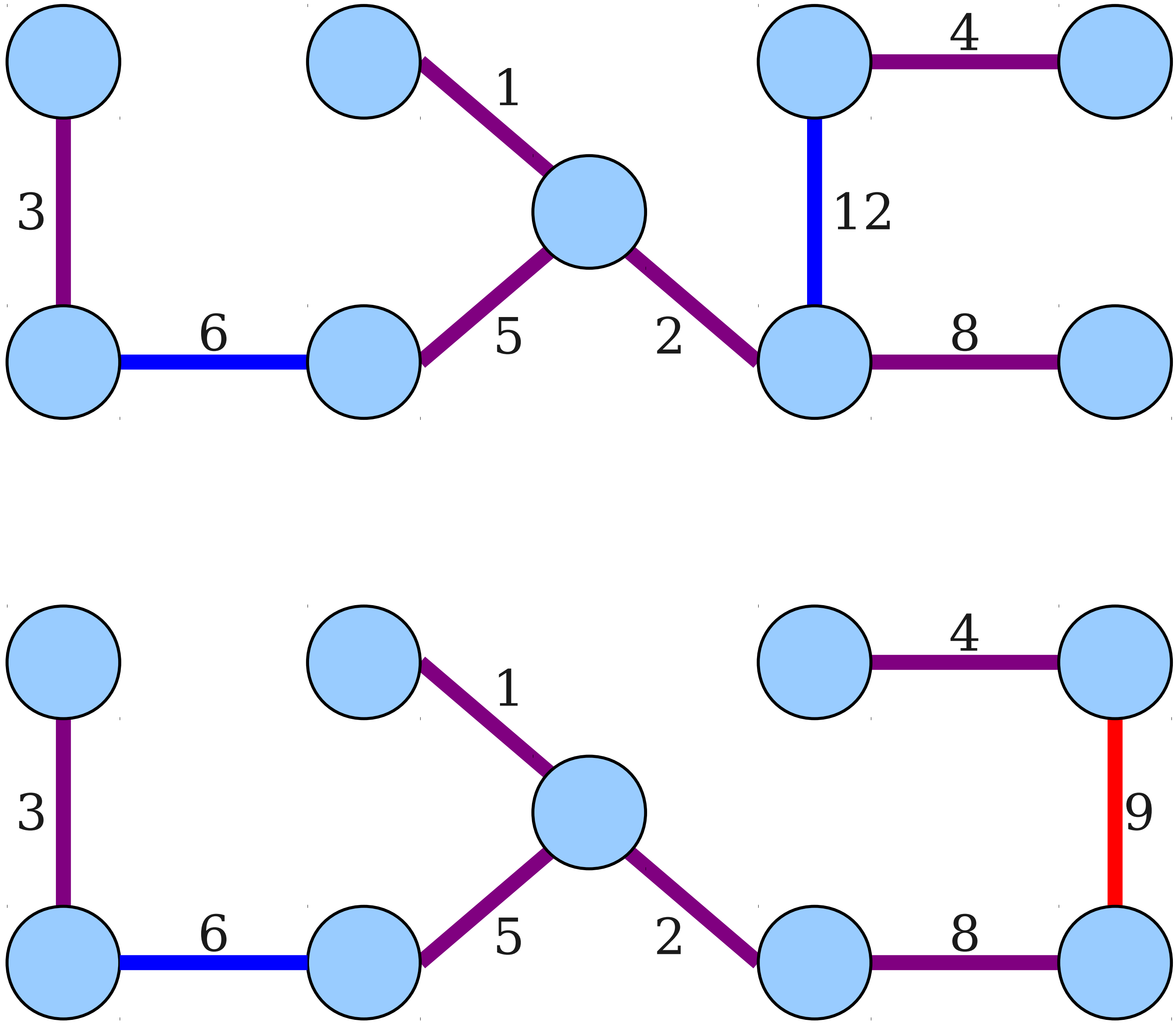
1
4
3
12
6
5
2
8

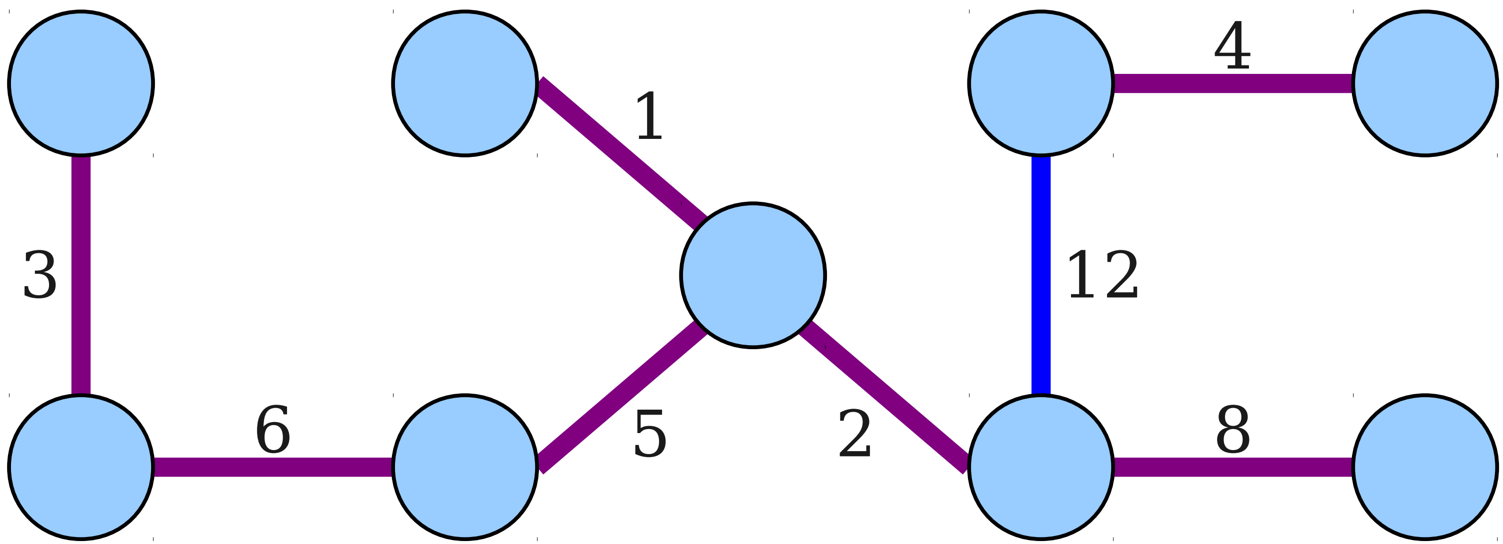


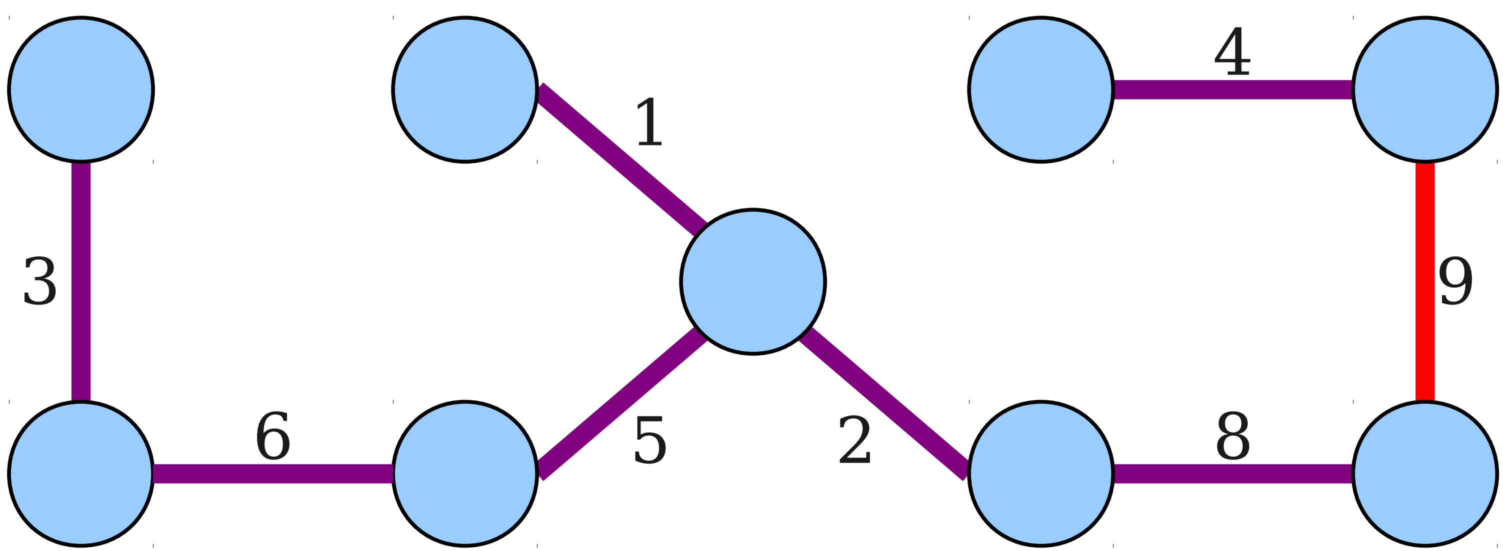
7
1
4
3
9
6
5
2
8

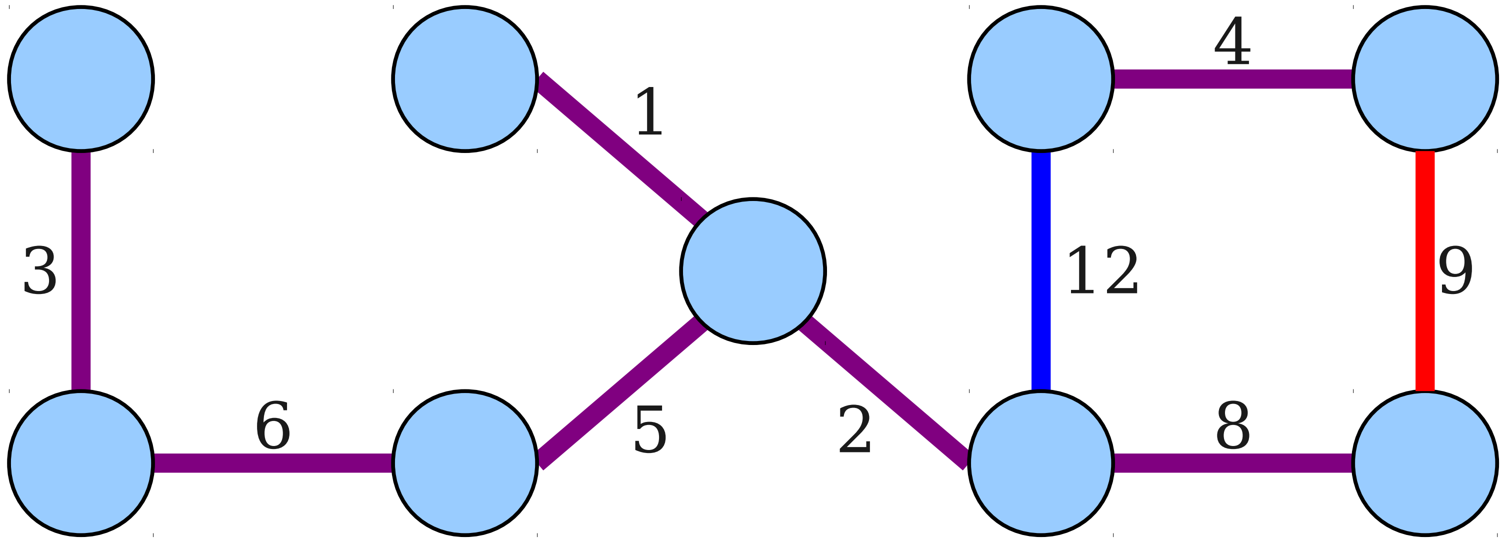


1
4
3
12
6
5
2
8
7
1
4
3
9
6
5
2
8

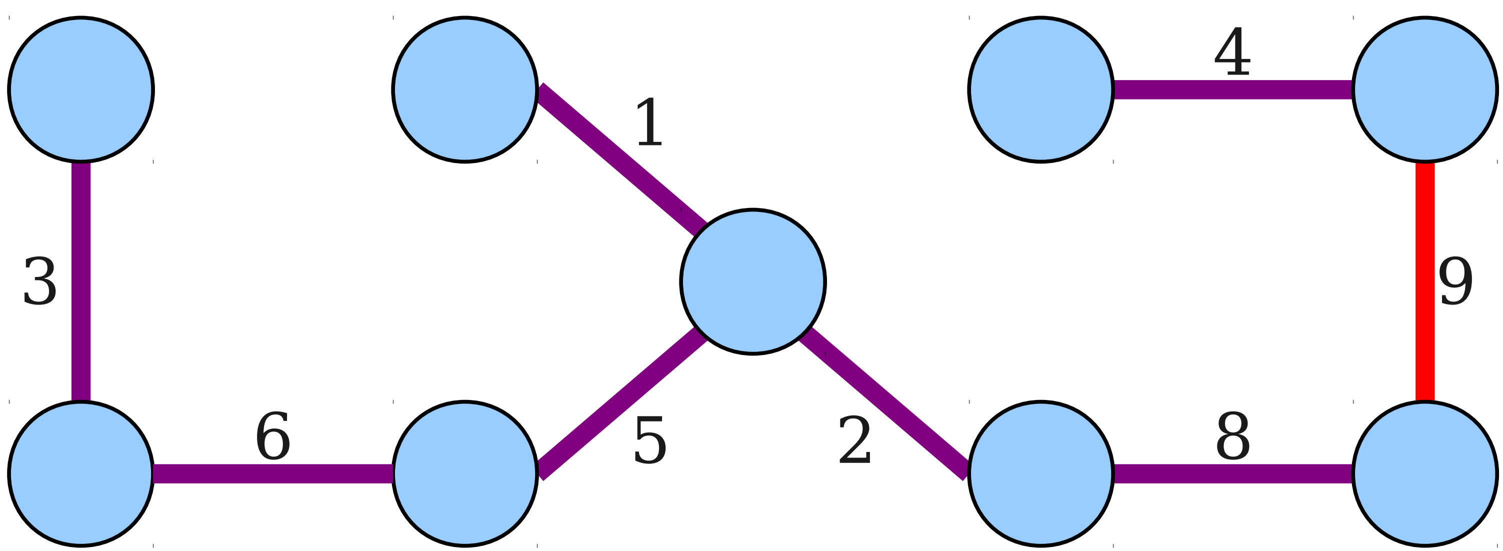

1
4
3
12
6
5
2
8
1
4
3
9
6
5
2
8


1
4
3
12
6
5
2
8

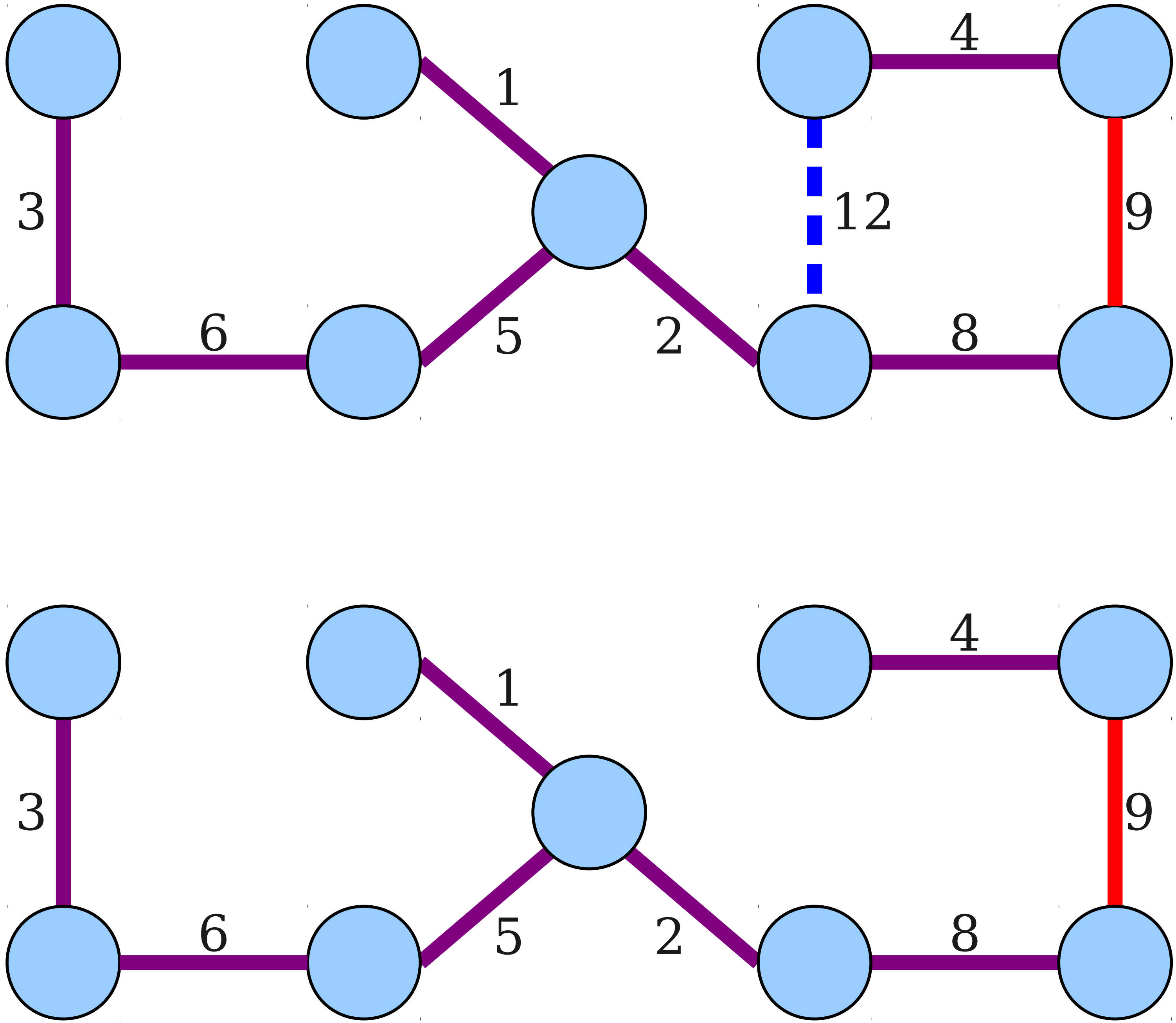

1
4
3
9
6
5
2
8



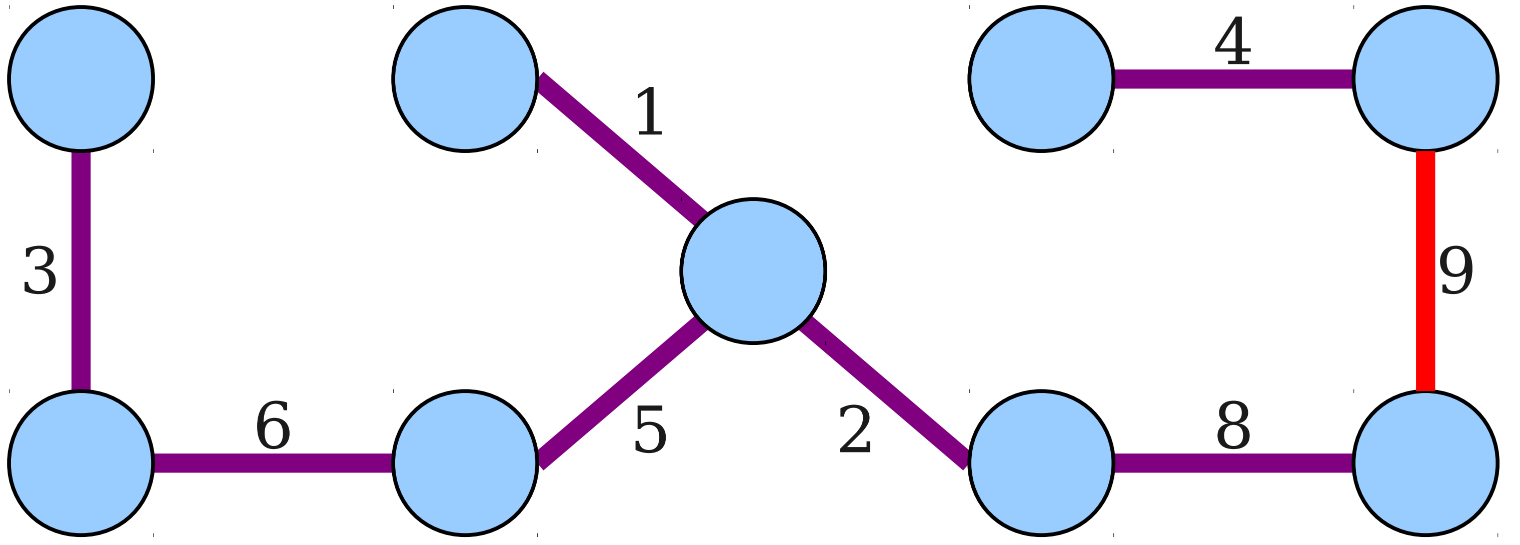
4
1
3
12
9
6
5
2
8

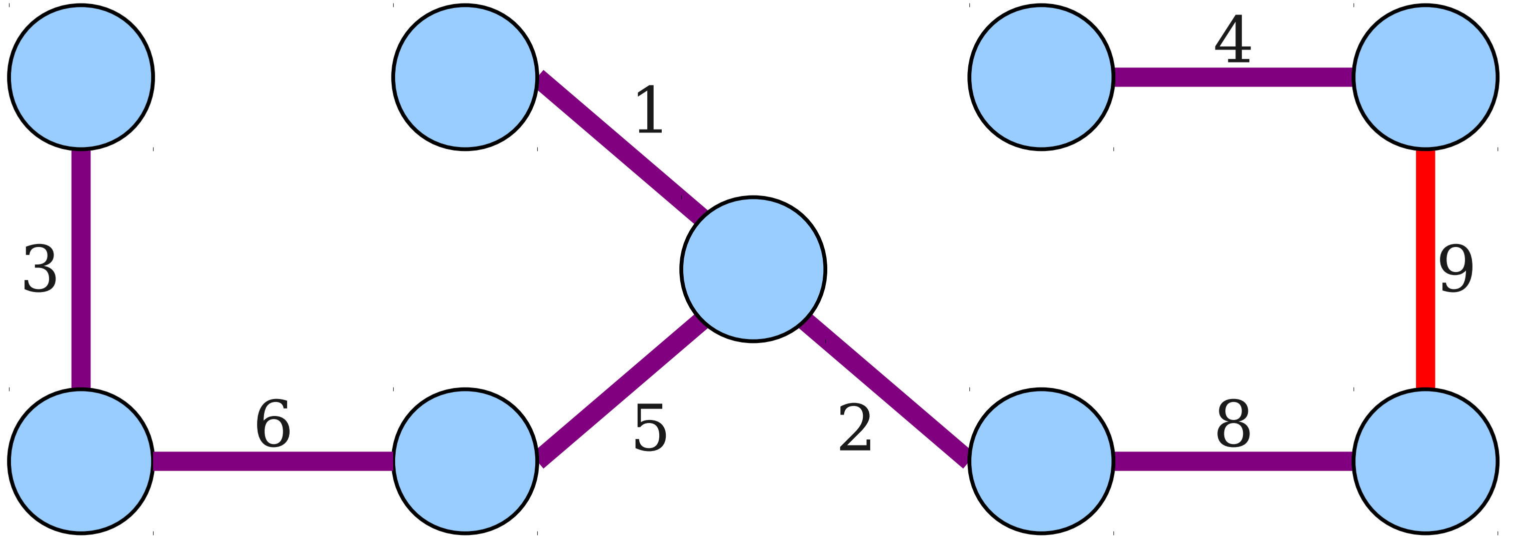


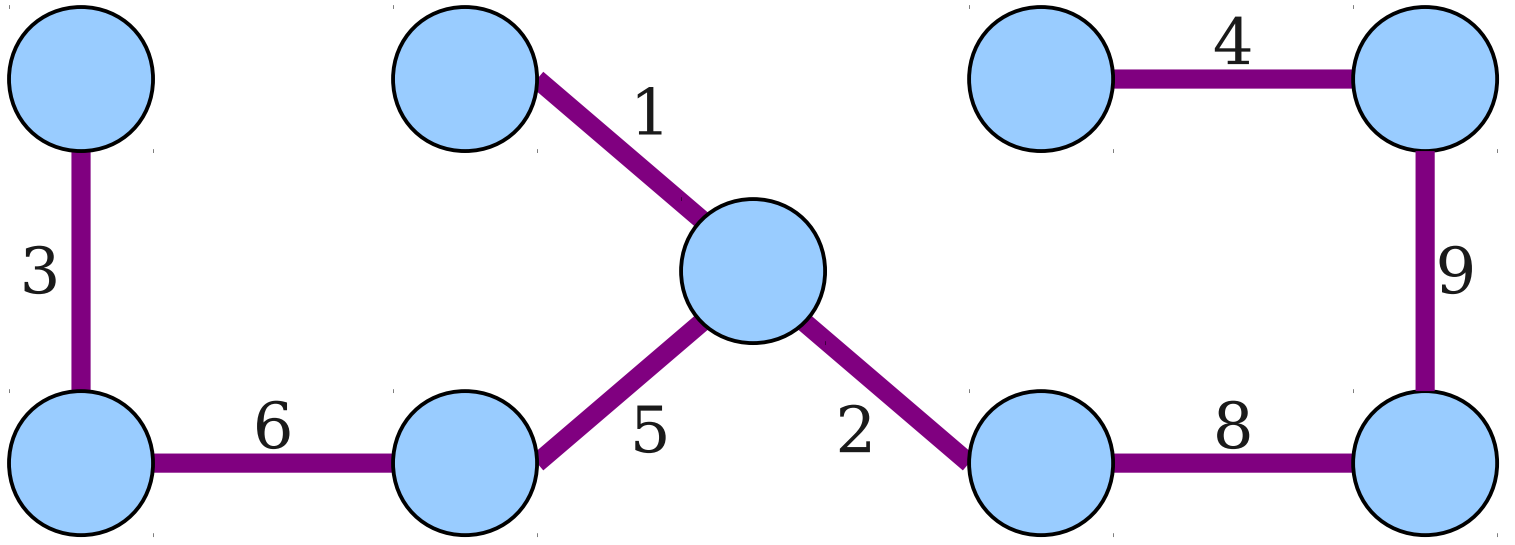
4
1
3
9
6
5
2
8



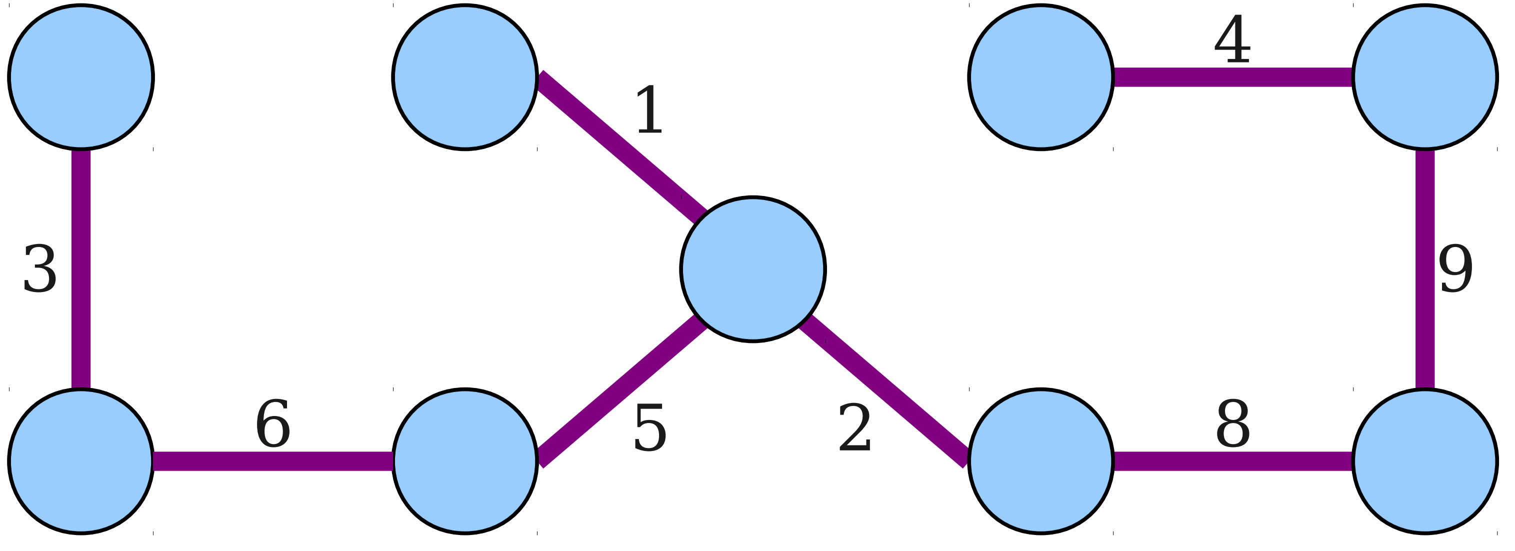
1
4
3
12
9
6
5
2
8
1
4
3
9
6
5
2
8


1
4
3
9
6
5
2
8


1
4
3
9
6
5
2
8



1
4
3
9
6
5
2
8



1
4
3
9
6
5
2
8

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v).

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v).

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1. The most expensive edge in $T_2 - T_1$ costs more than $c(u, v)$; otherwise (u, v) would not be the cheapest edge in $T_1 \Delta T_2$.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1. The most expensive edge in $T_2 - T_1$ costs more than $c(u, v)$; otherwise (u, v) would not be the cheapest edge in $T_1 \Delta T_2$. Thus the highest-cost edge in the cycle has cost at least $c(u, v)$.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1. The most expensive edge in $T_2 - T_1$ costs more than $c(u, v)$; otherwise (u, v) would not be the cheapest edge in $T_1 \Delta T_2$. Thus the highest-cost edge in the cycle has cost at least $c(u, v)$.

As proven earlier, $T' = T_2 \cup \{(u, v)\} - \{(x, y)\}$ is a spanning tree of G.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1. The most expensive edge in $T_2 - T_1$ costs more than $c(u, v)$; otherwise (u, v) would not be the cheapest edge in $T_1 \Delta T_2$. Thus the highest-cost edge in the cycle has cost at least $c(u, v)$.

As proven earlier, $T' = T_2 \cup \{(u, v)\} - \{(x, y)\}$ is a spanning tree of G. But $c(T') = c(T_2) + c(u, v) - c(x, y) < c(T_2)$, which contradicts that T_2 is an MST.

Theorem: Let G be a connected, weighted graph. If all edge weights in G are distinct, G has exactly one MST.

Proof: Since G is connected, it has at least one MST. We will show G has at most one MST by contradiction. Assume T_1 and T_2 are distinct MSTs of G. Since $|T_1| = |T_2|$, the set $T_1 \Delta T_2$ is nonempty, so it contains a least-cost edge (u, v). Assume without loss of generality that $(u, v) \in T_1$.

Consider $T_2 \cup \{(u, v)\}$. Since T_2 is a tree, this graph has a cycle C involving (u, v). Let (x, y) be the edge in C with the highest total cost. We claim $c(x, y) > c(u, v)$. To see this, note that every edge in C other than (u, v) belongs either to $T_2 \cap T_1$ or to $T_2 - T_1$. Some edge in the cycle must belong to $T_2 - T_1$, or otherwise (u, v) closes a cycle in T_1. The most expensive edge in $T_2 - T_1$ costs more than $c(u, v)$; otherwise (u, v) would not be the cheapest edge in $T_1 \Delta T_2$. Thus the highest-cost edge in the cycle has cost at least $c(u, v)$.

As proven earlier, $T' = T_2 \cup \{(u, v)\} - \{(x, y)\}$ is a spanning tree of G. But $c(T') = c(T_2) + c(u, v) - c(x, y) < c(T_2)$, which contradicts that T_2 is an MST. Thus our assumption was wrong and there is at most one MST in G. ■

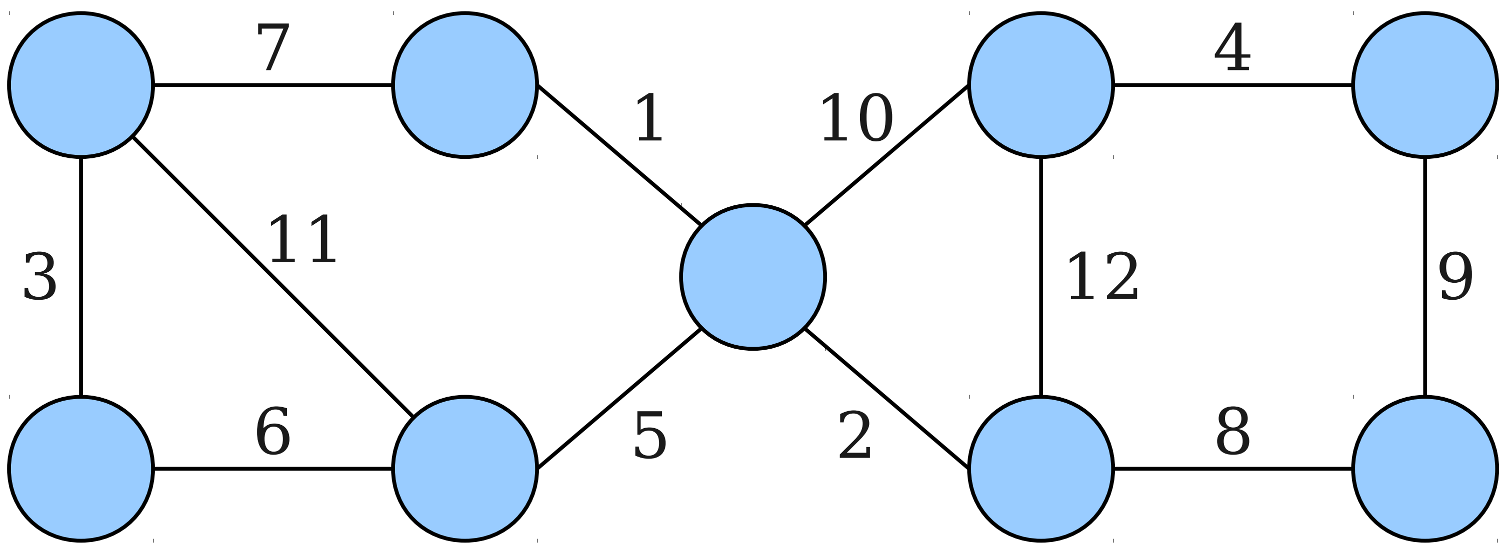
The Cycle Property

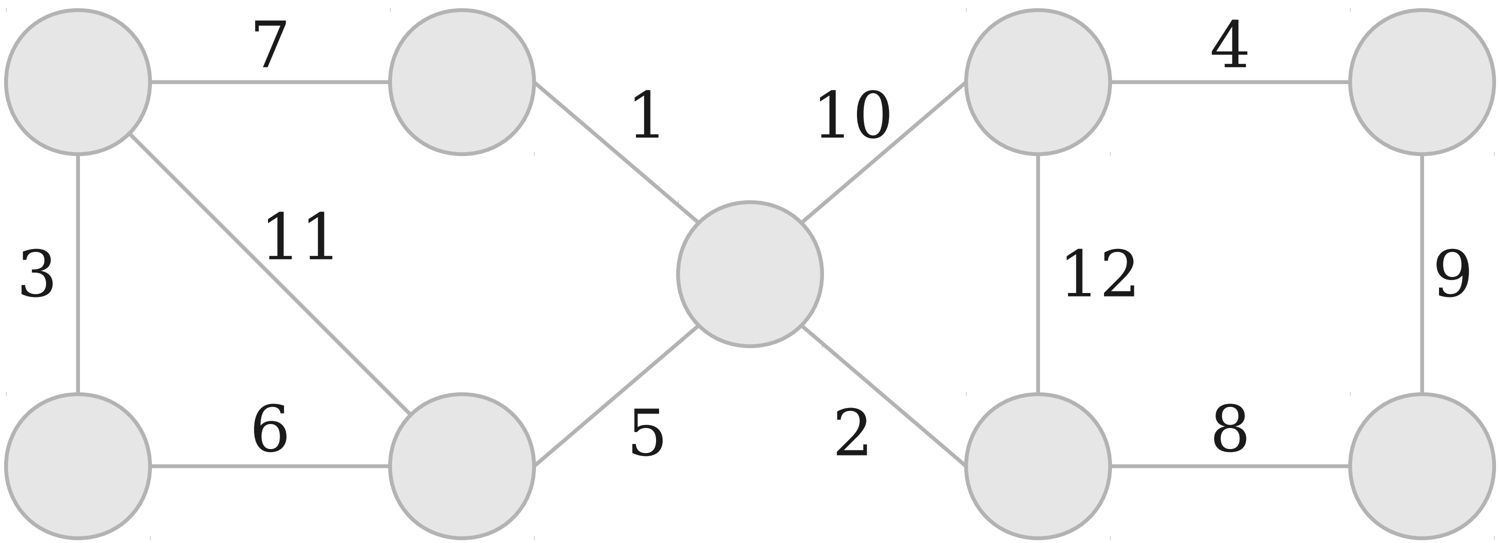
- This previous proof relies on a property of MSTs called the *cycle property*.

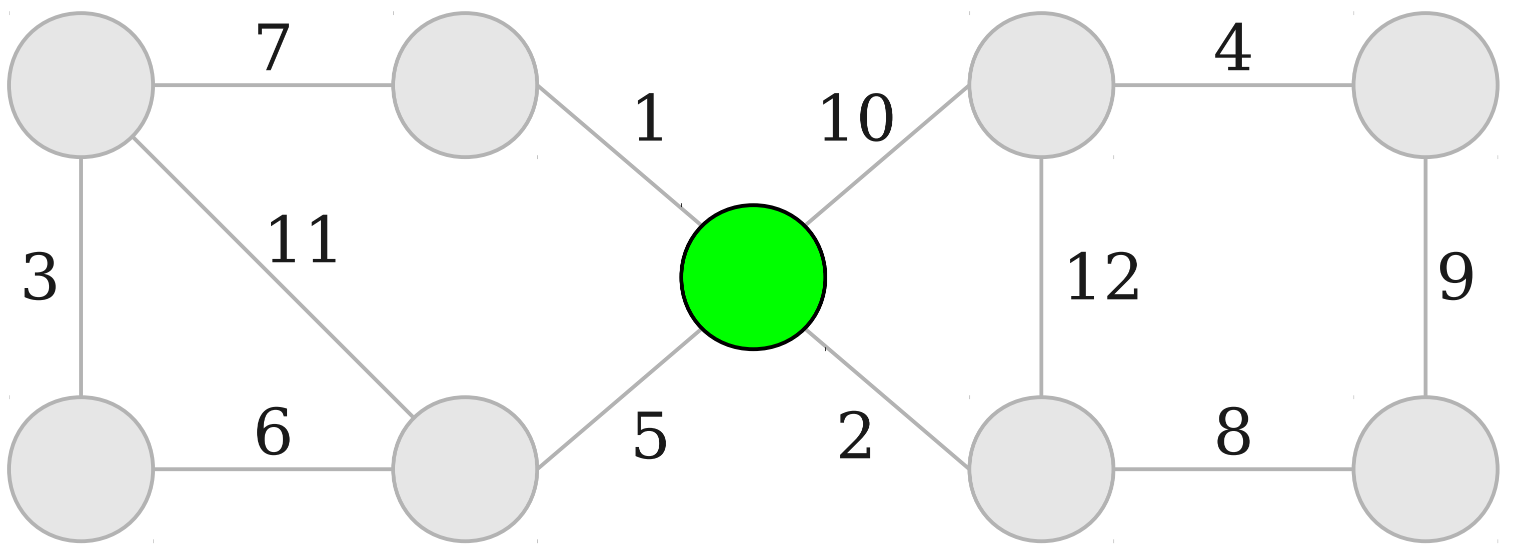
  Theorem (Cycle Property): If (x, y) is an edge in G and is the heaviest edge on some cycle C, then (x, y) does not belong to any MST of G.

- Proof along the lines of what we just saw: if it did belong to some MST, adding the cheapest edge on that cycle and removing (x, y) leaves a lower-cost spanning tree.

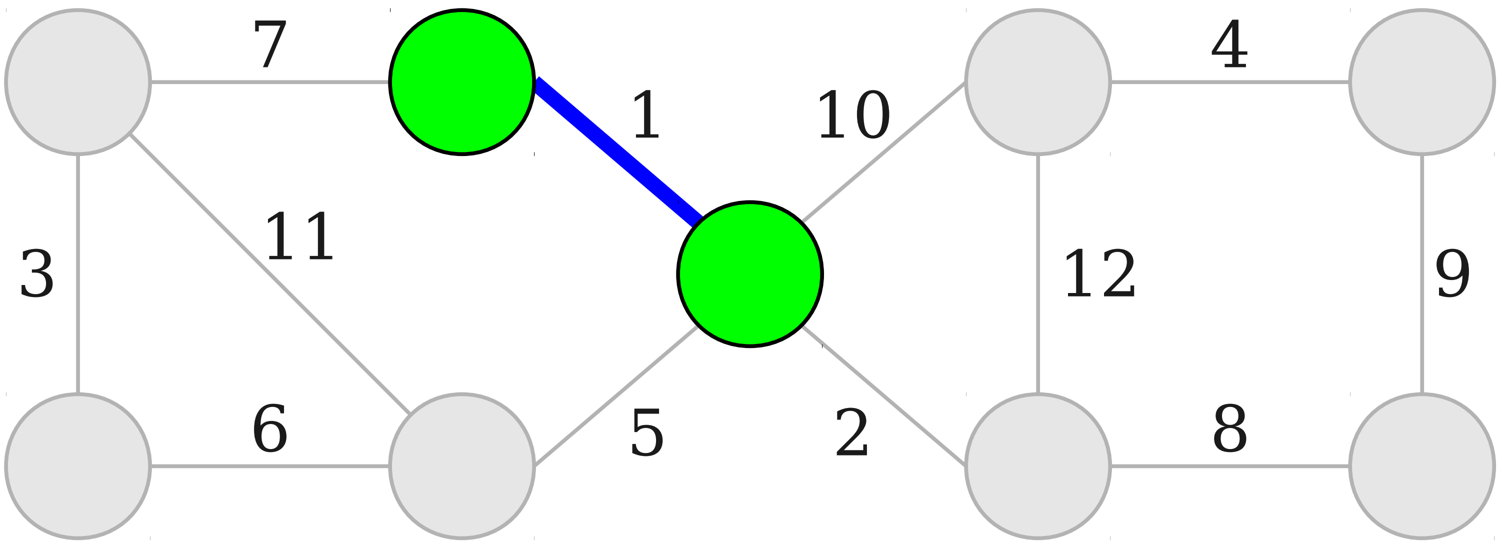
Finding MSTs: **Prim's Algorithm**

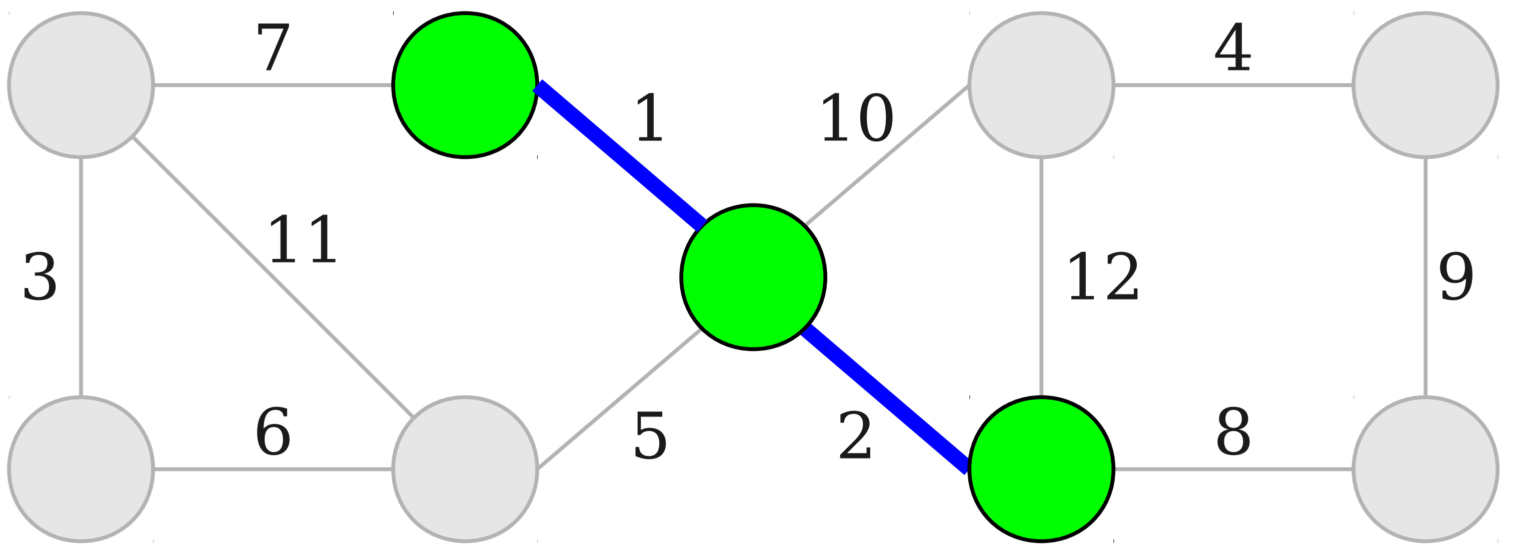

7
1
10
4
11
3
12
9
6
5
2
8


7
1
10
4
3
11
12
9
6
5
2
8

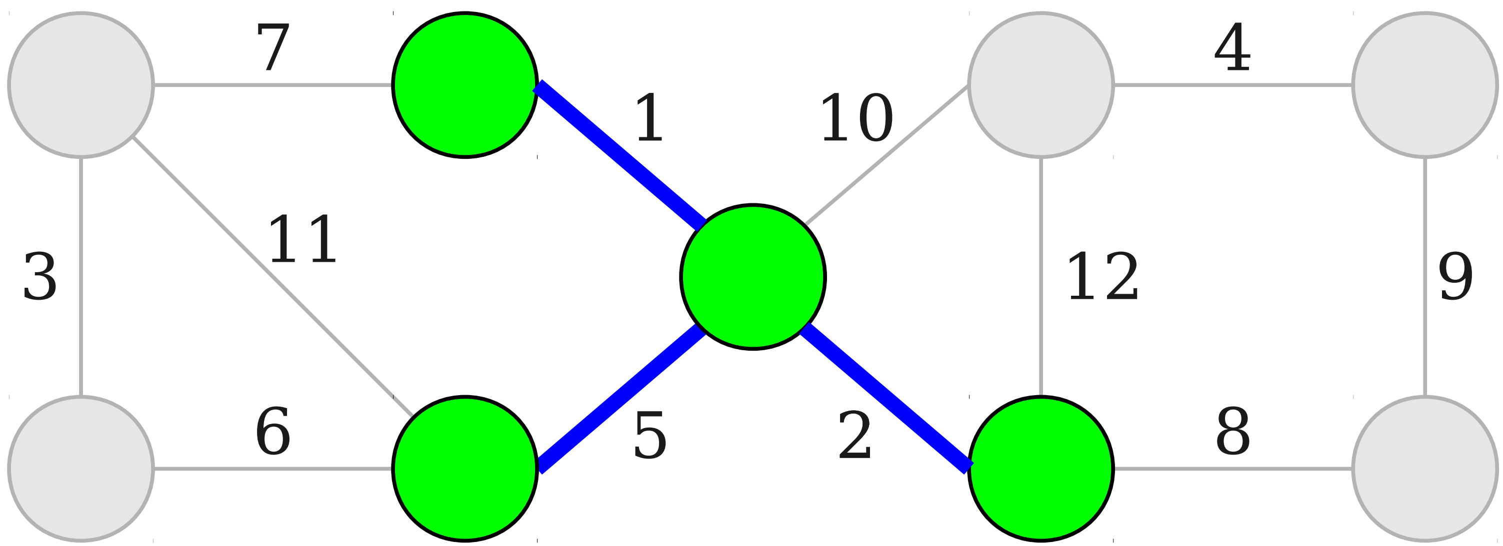


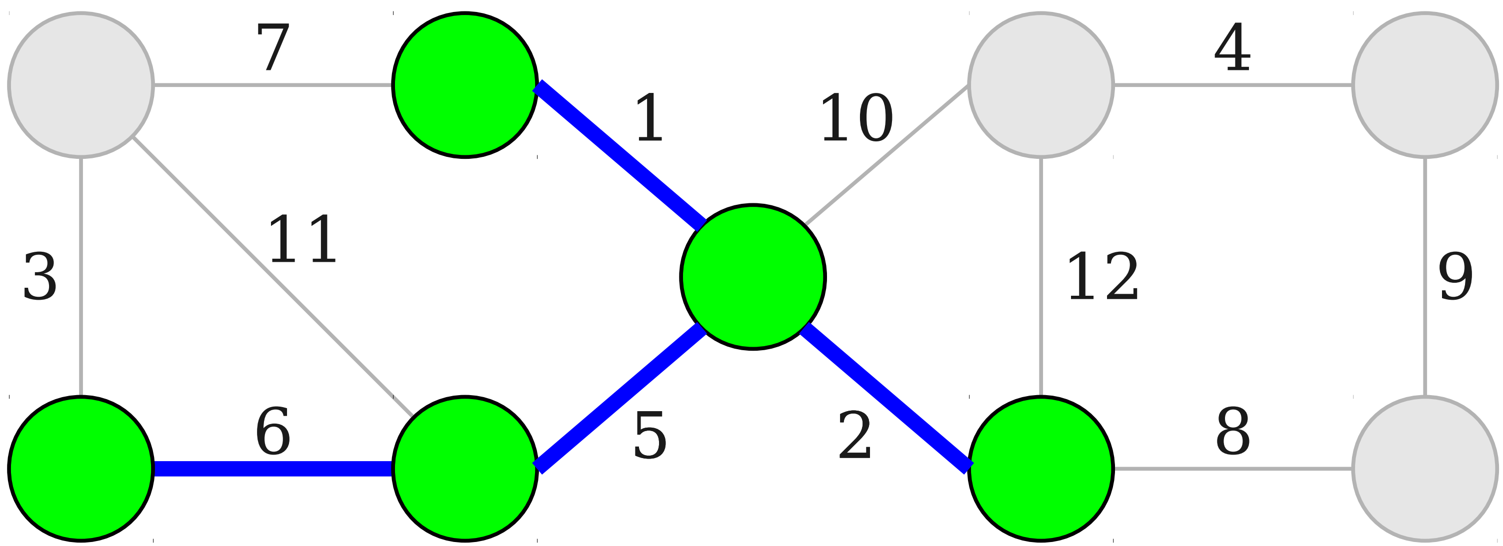
7
1
10
4
3
11
12
9
6
5
2
8

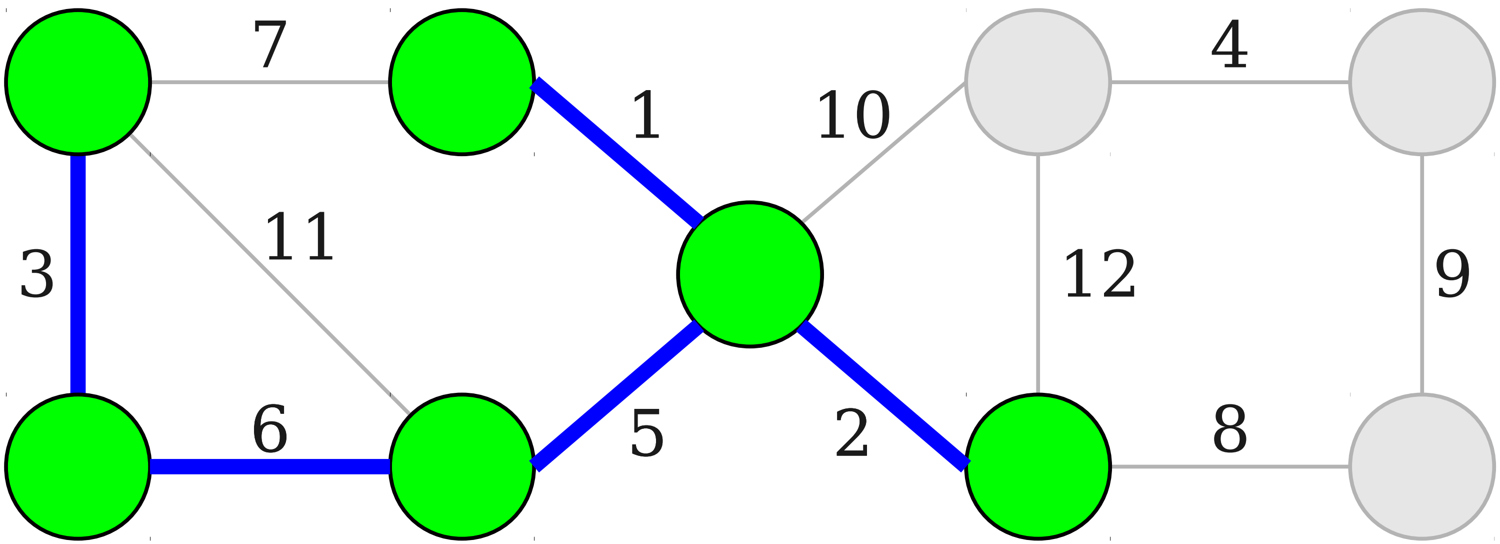

7
1
10
4
3
11
12
9
6
5
2
8

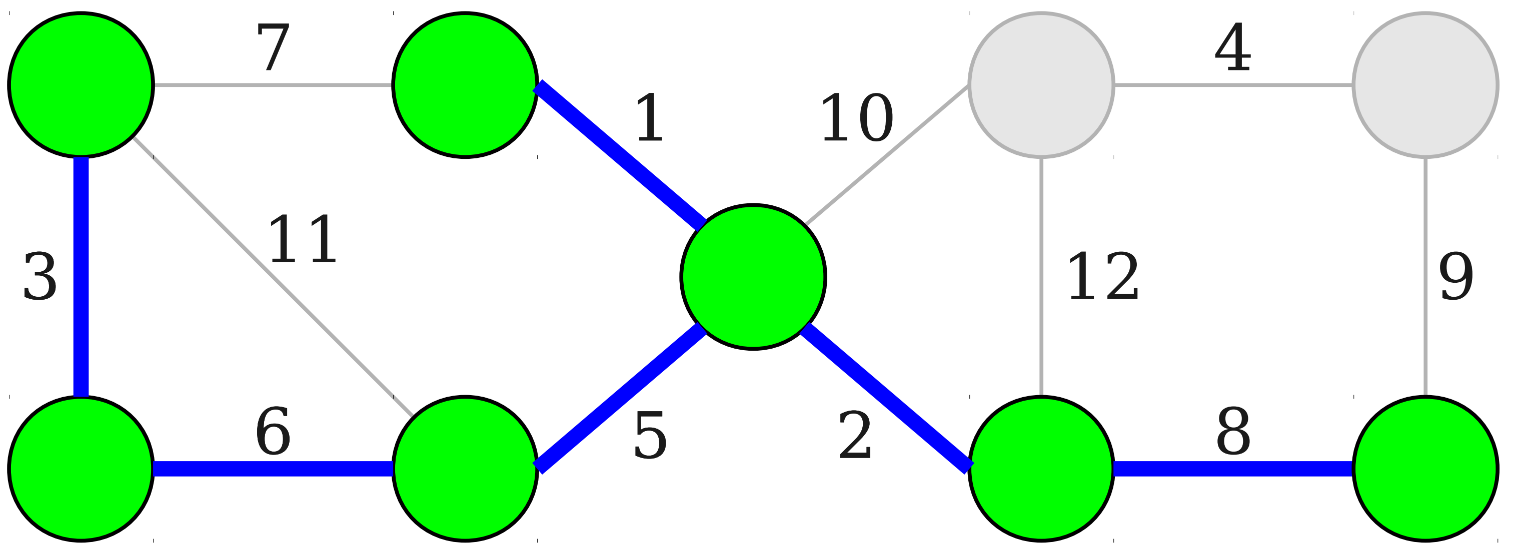


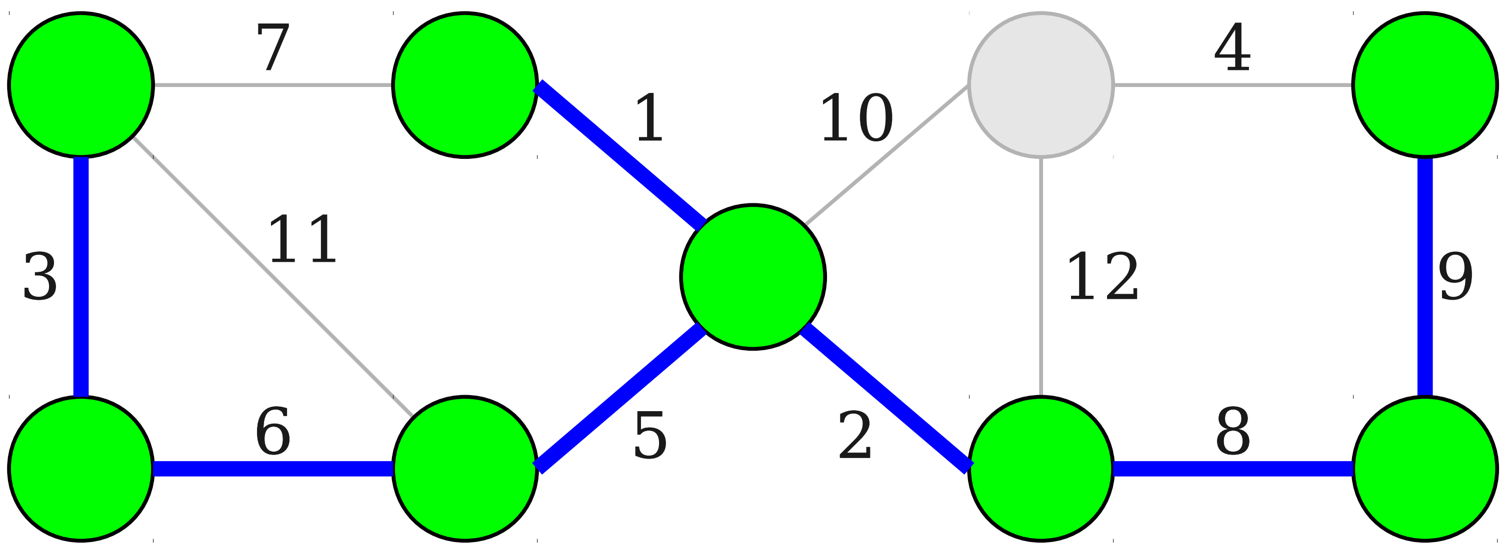
7
1
10
4
3
11
12
9
6
5
2
8

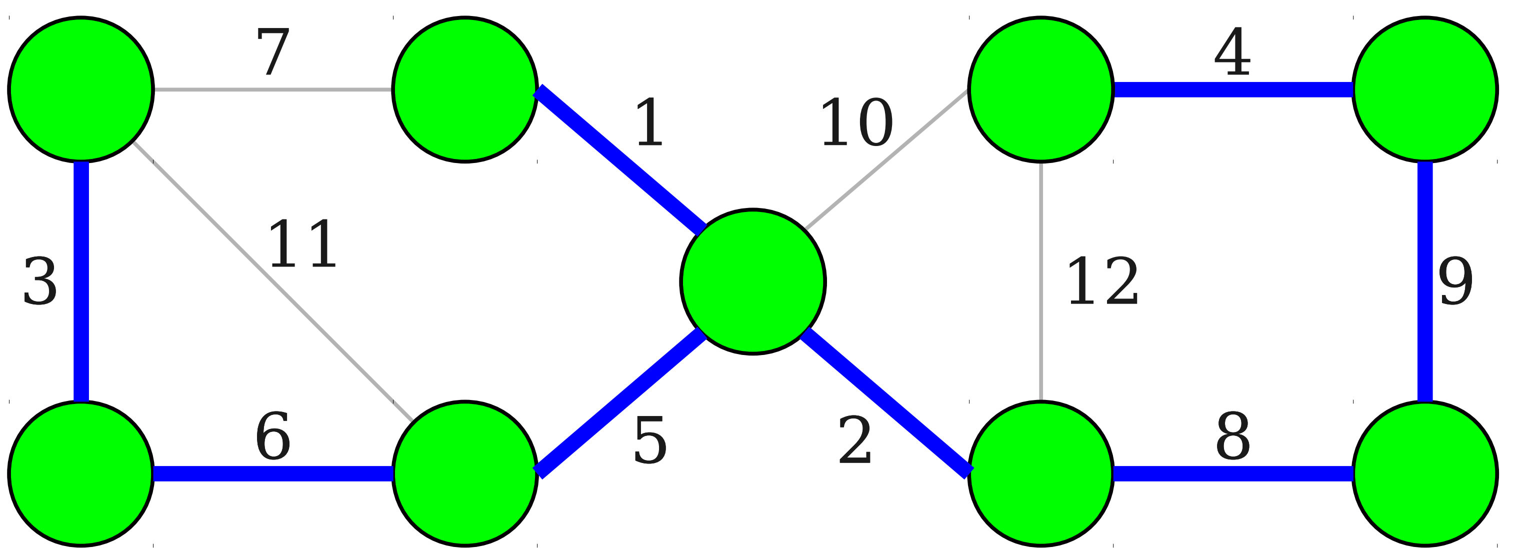

7
1
10
4
3
11
12
9
6
5
2
8

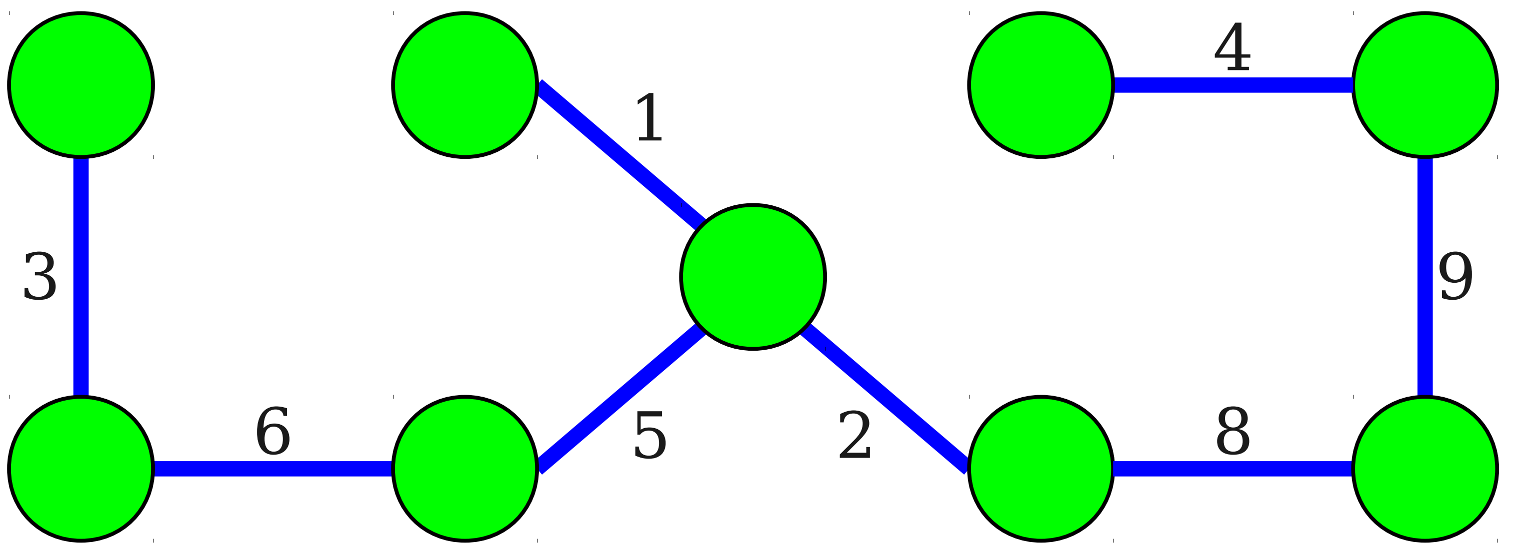

7
1
10
4
3
11
12
9
6
5
2
8


7
1
10
4
11
3
12
9
6
5
2
8


7
4
1
10
11
3
12
9
6
5
2
8


7
1
10
4
11
3
12
9
6
5
2
8


7
1
10
4
11
3
12
9
6
5
2
8


4
1
3
9
6
5
2
8

Prim's Algorithm

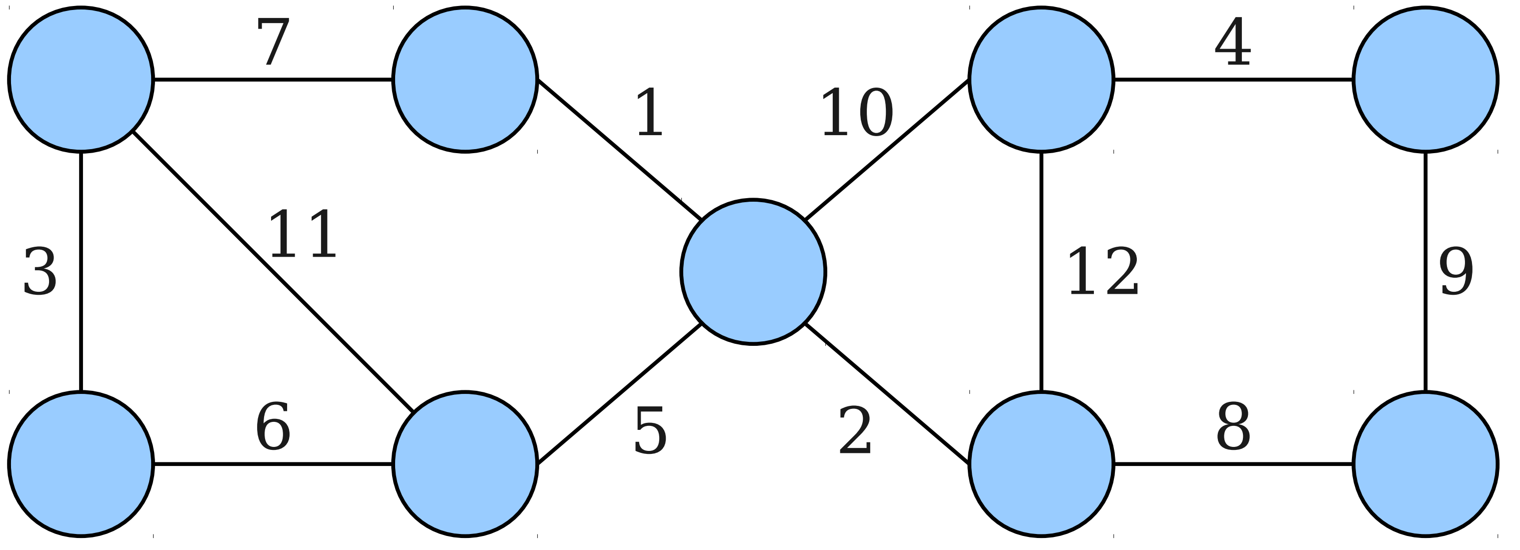
- **Prim's Algorithm** is the following:
  - Choose some $v \in V$ and let $S = \{v\}$.
  - Let $T = \emptyset$.
  - While $S \neq V$:
    - Choose a least-cost edge e with one endpoint in S and one endpoint in $V - S$.
    - Add e to T.
    - Add both endpoints of e to S.
- (Quick history: This was originally invented by Czech mathematician Vojtěch Jarník in 1930.)

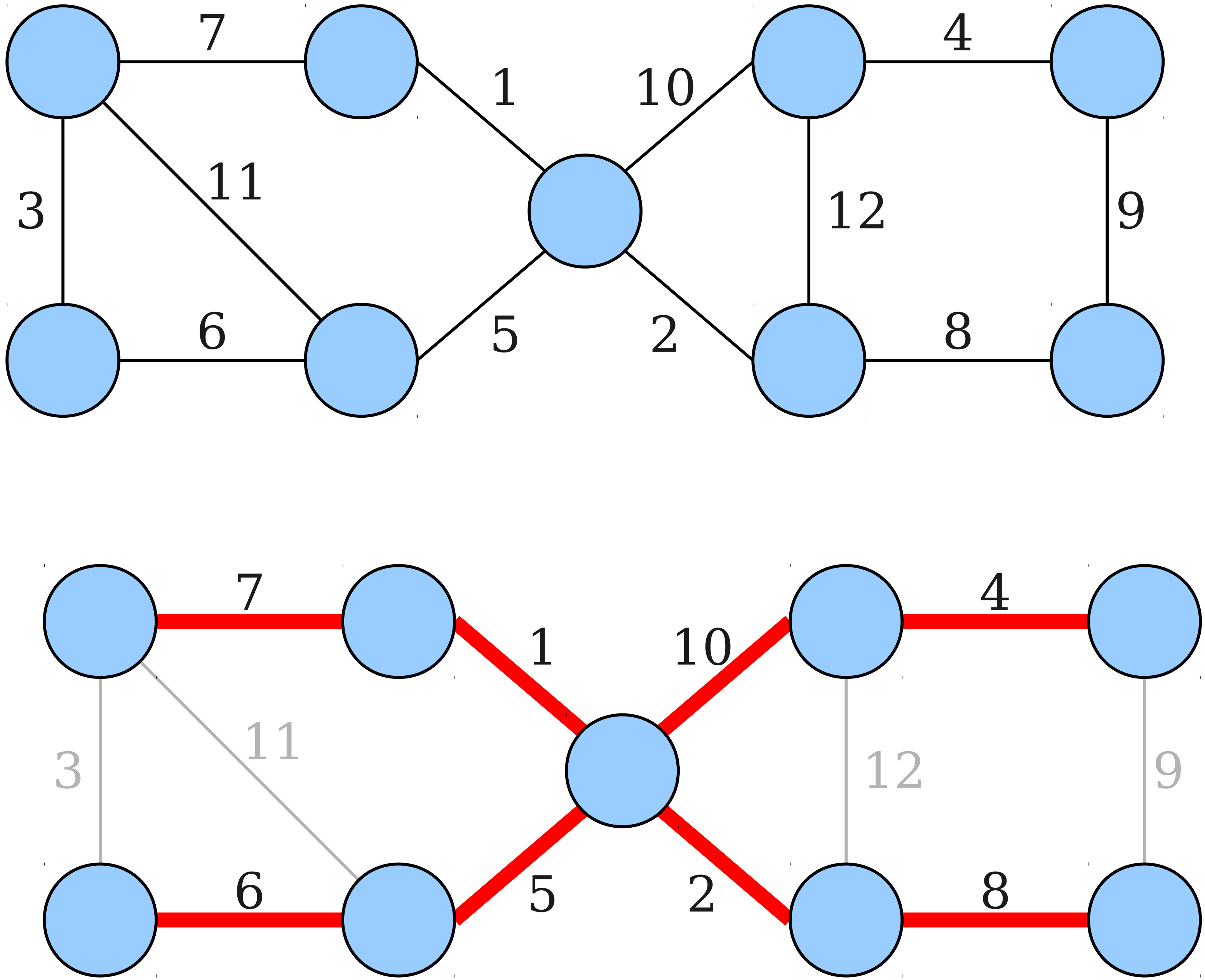
Proving Legality

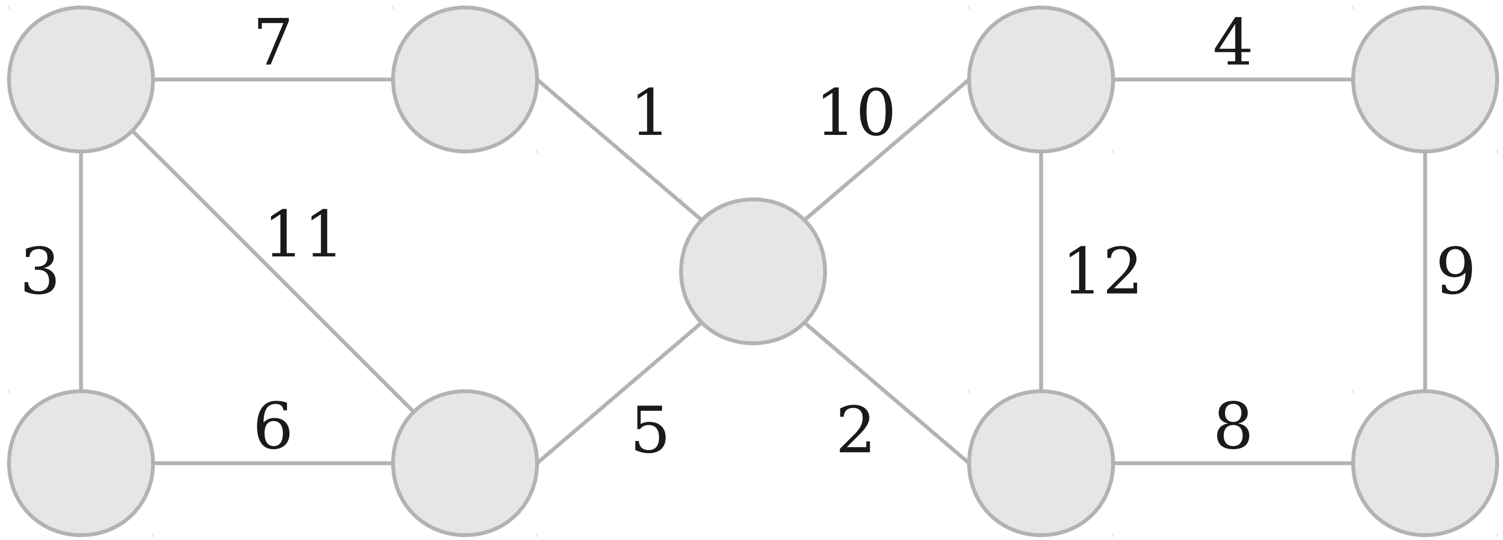
- ***Claim:*** Prim's algorithm produces a spanning tree of G.
- ***Proof idea:*** Show by induction that T forms a spanning tree of the nodes in S. Conclude that since eventually $S = V$, that T is a spanning tree for G.

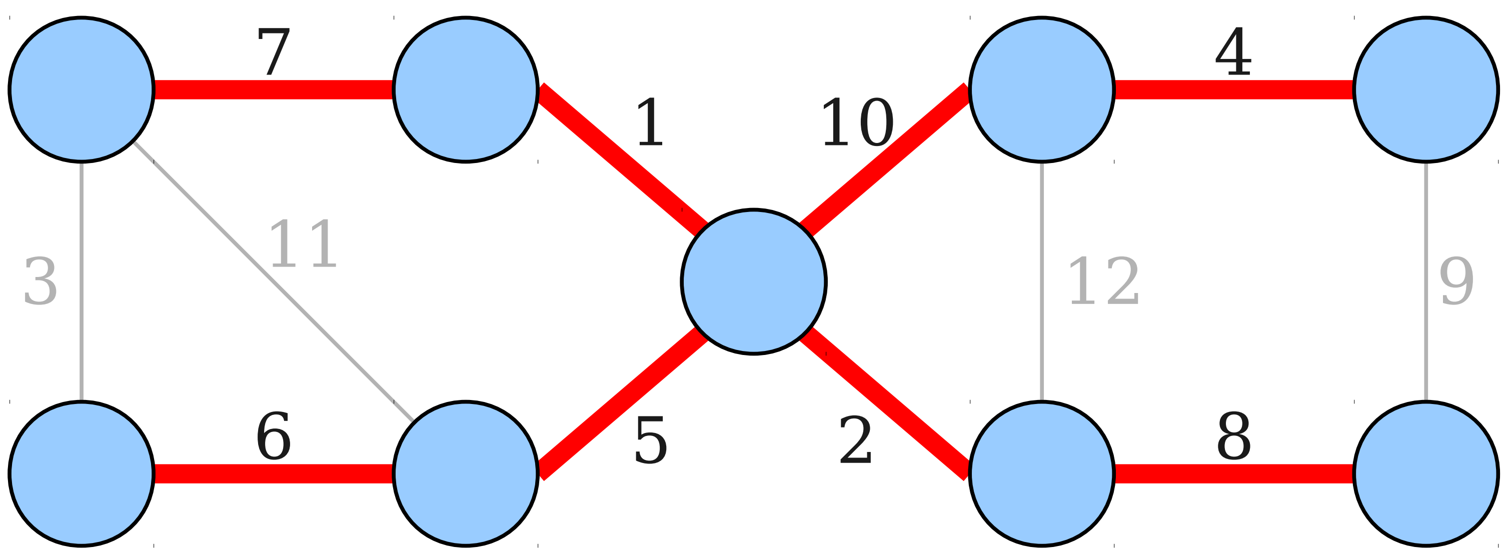
Proving Optimality

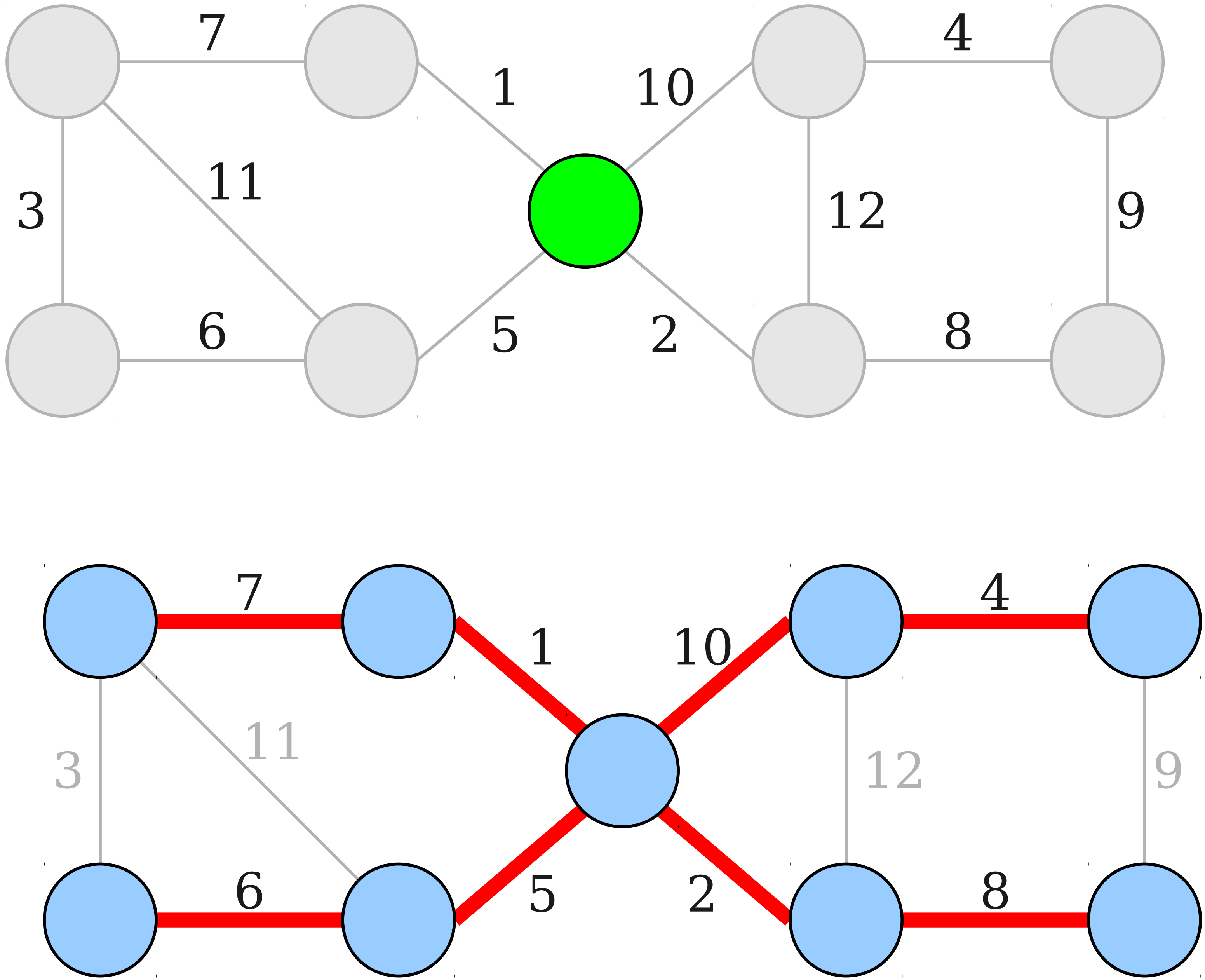
- To show that Prim's algorithm produces an MST, we will work in two steps:
  - First, as a warmup, show that Prim's algorithm produces an MST as long as all edge costs are distinct.
  - Then, for the full proof, show that Prim's algorithm produces an MST even if there are multiple edges with the same cost.
- In doing so, we will see the ***exchange argument*** as another method for proving a greedy algorithm is optimal.

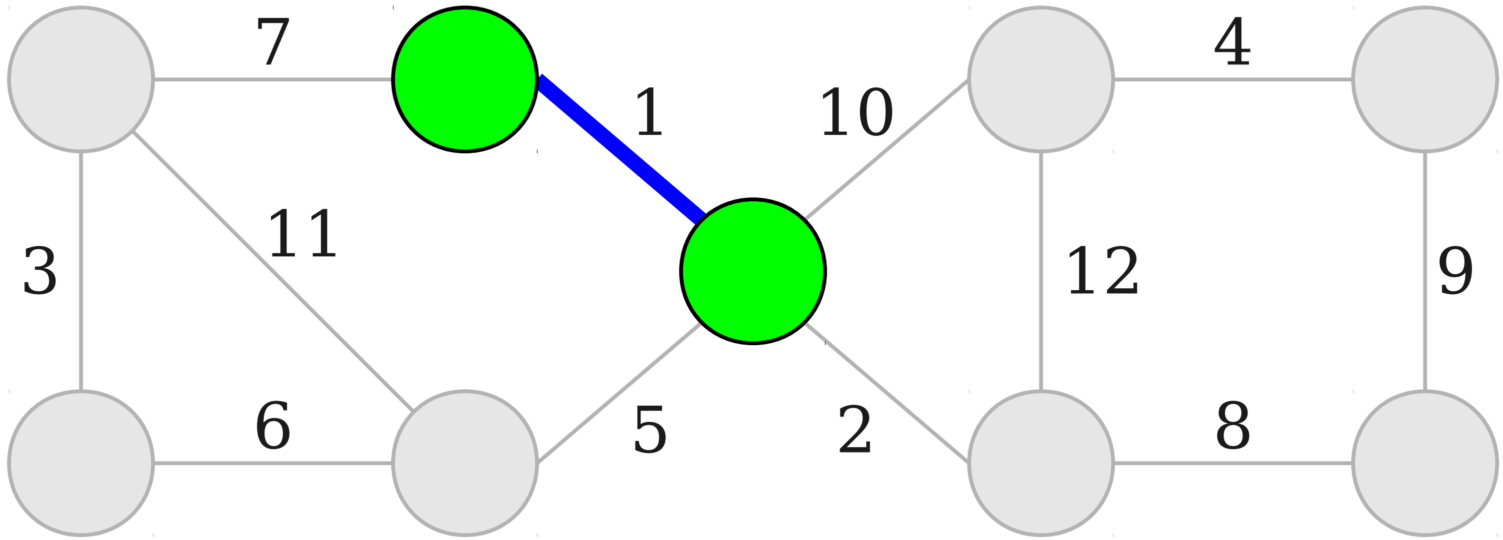

7
1
10
4
11
3
12
9
6
5
2
8

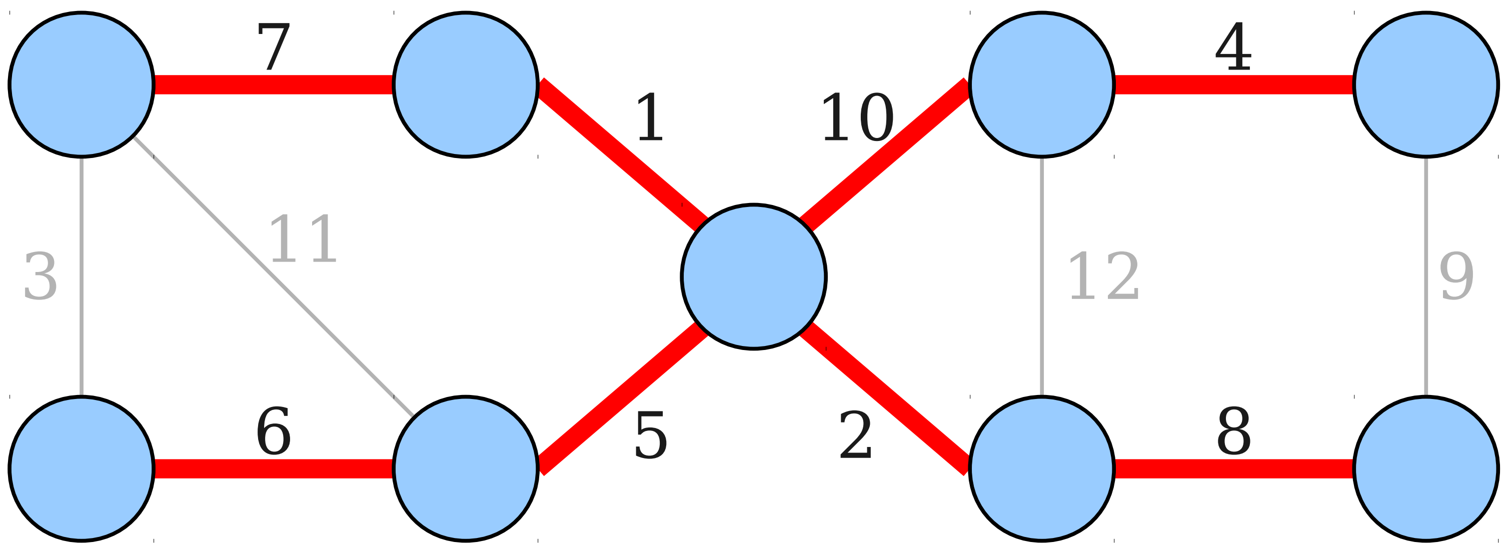

7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8

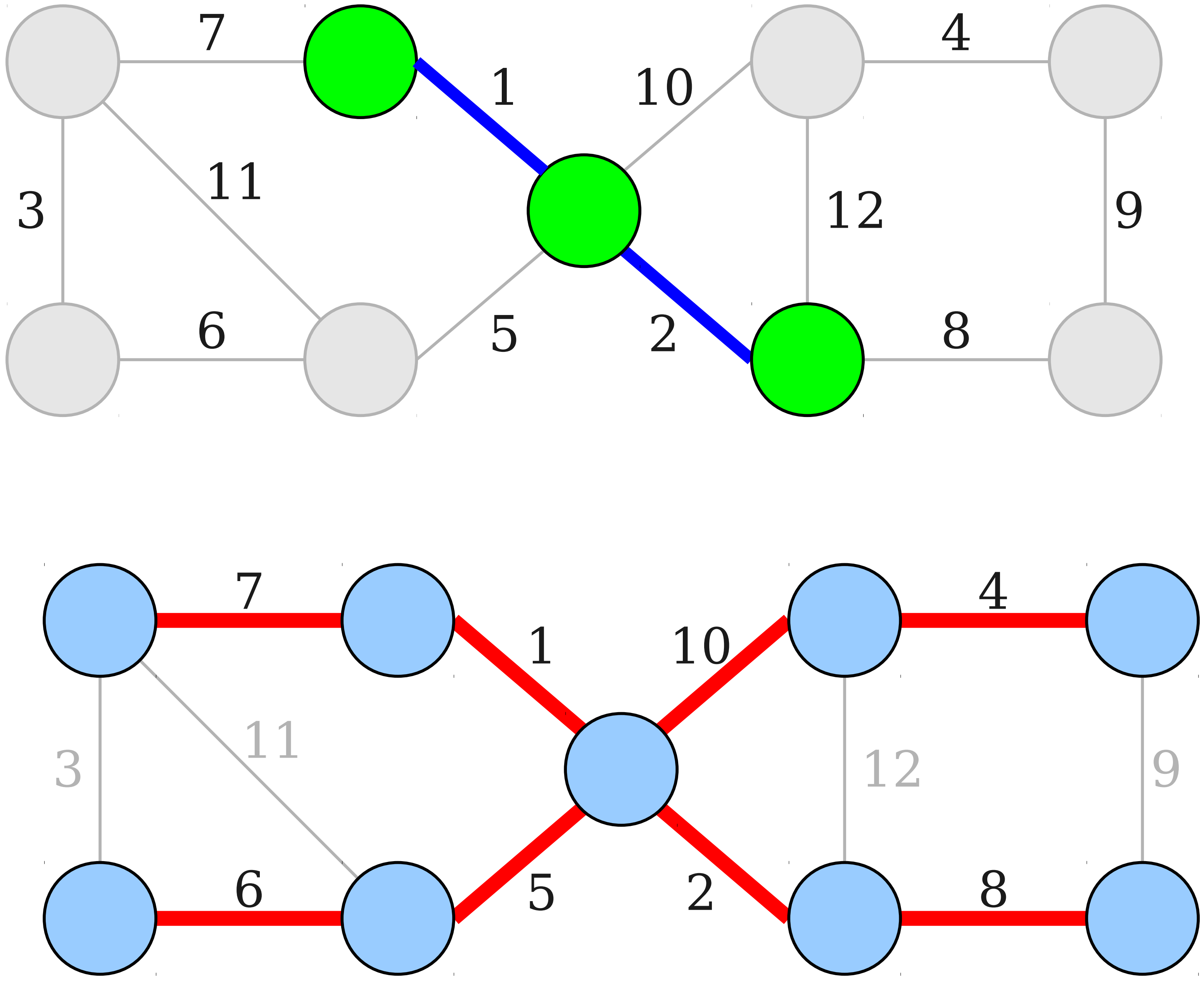

7
1
10
4
3
11
12
9
6
5
2
8

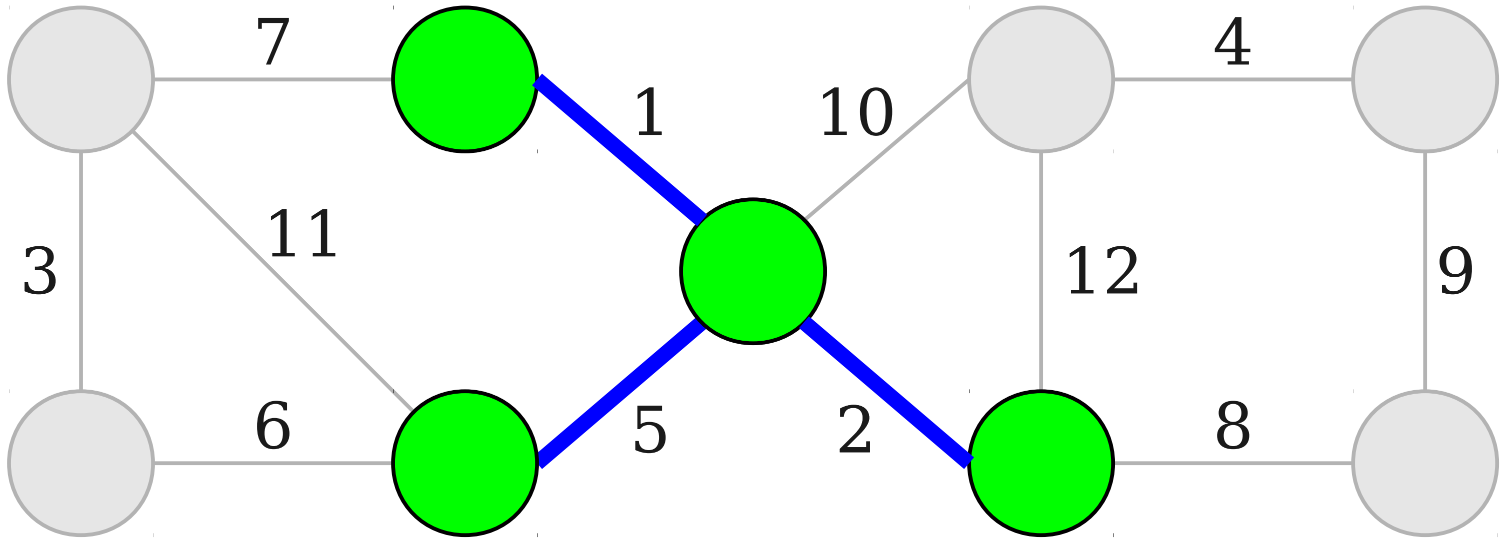

7
1
10
4
3
11
12
9
6
5
2
8

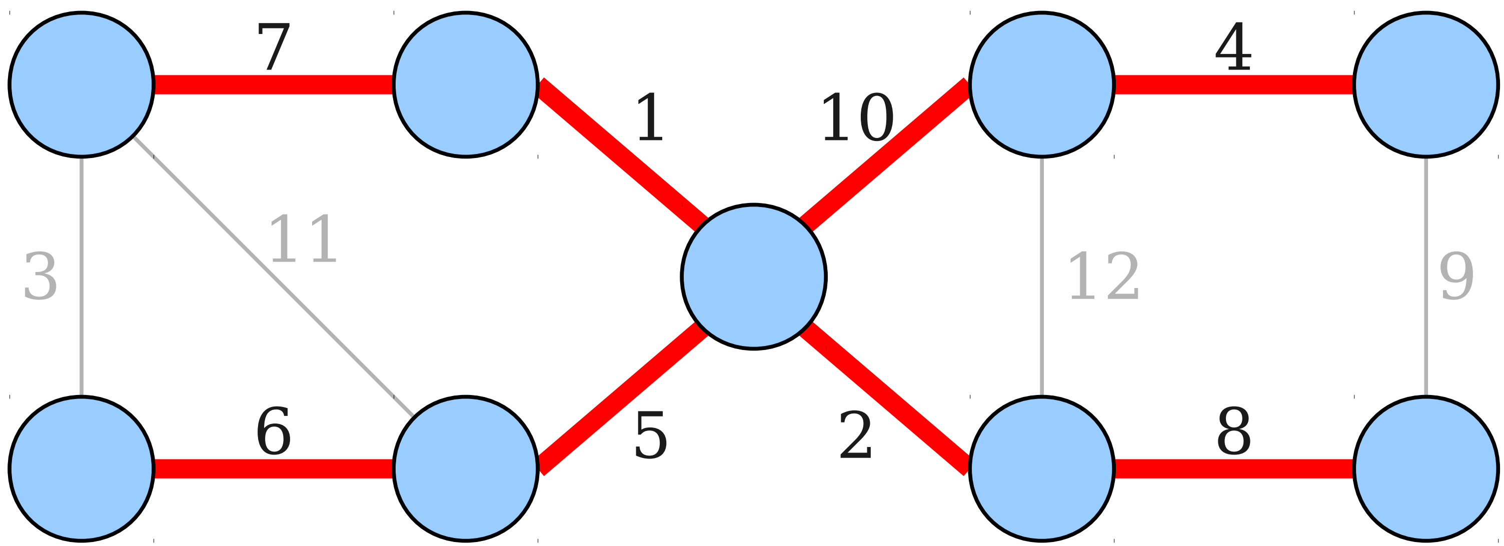

7
4
1
10
11
3
12
9
6
5
2
8
7
4
1
10
11
3
12
9
6
5
2
8

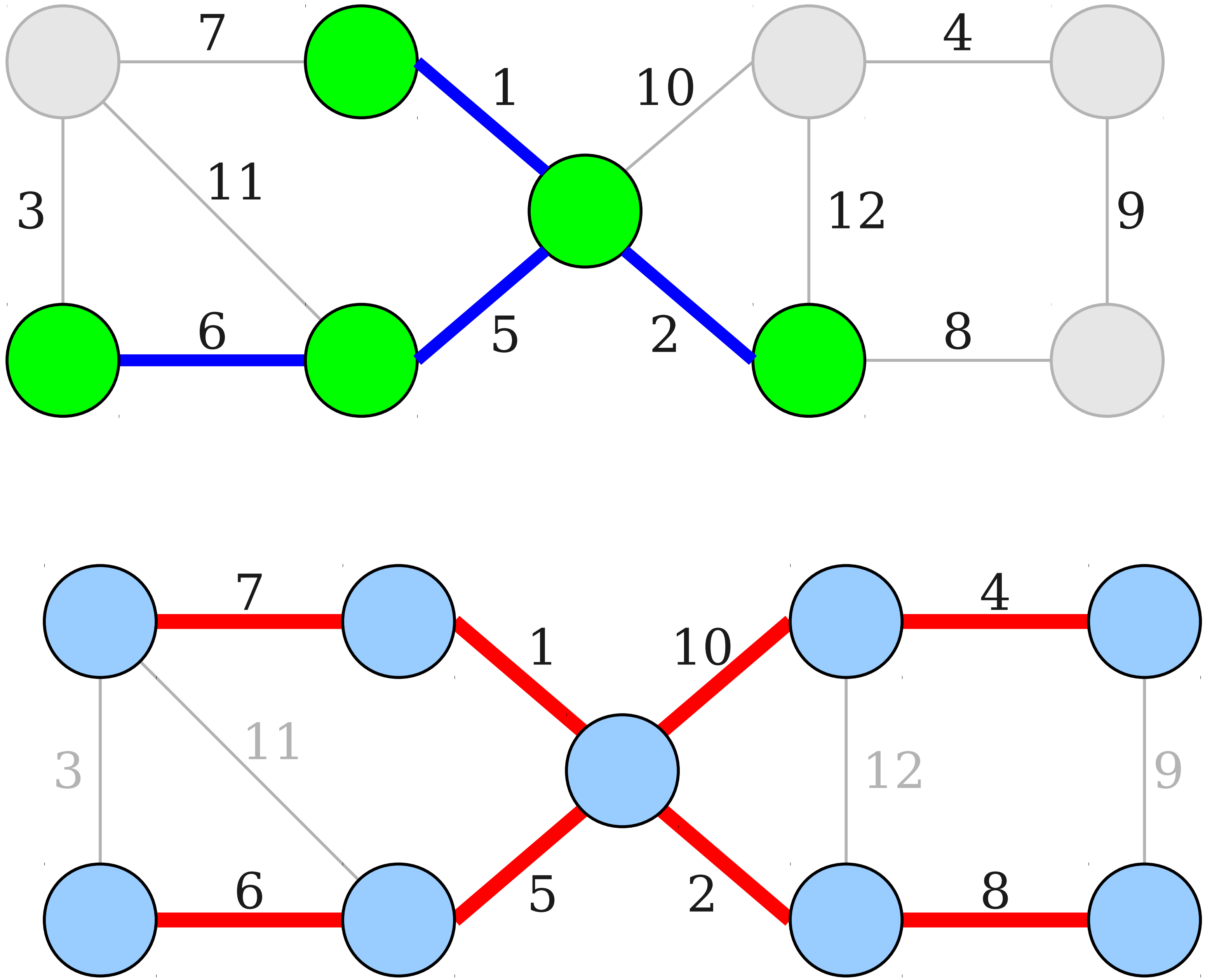

7
1
10
4
3
11
12
9
6
5
2
8

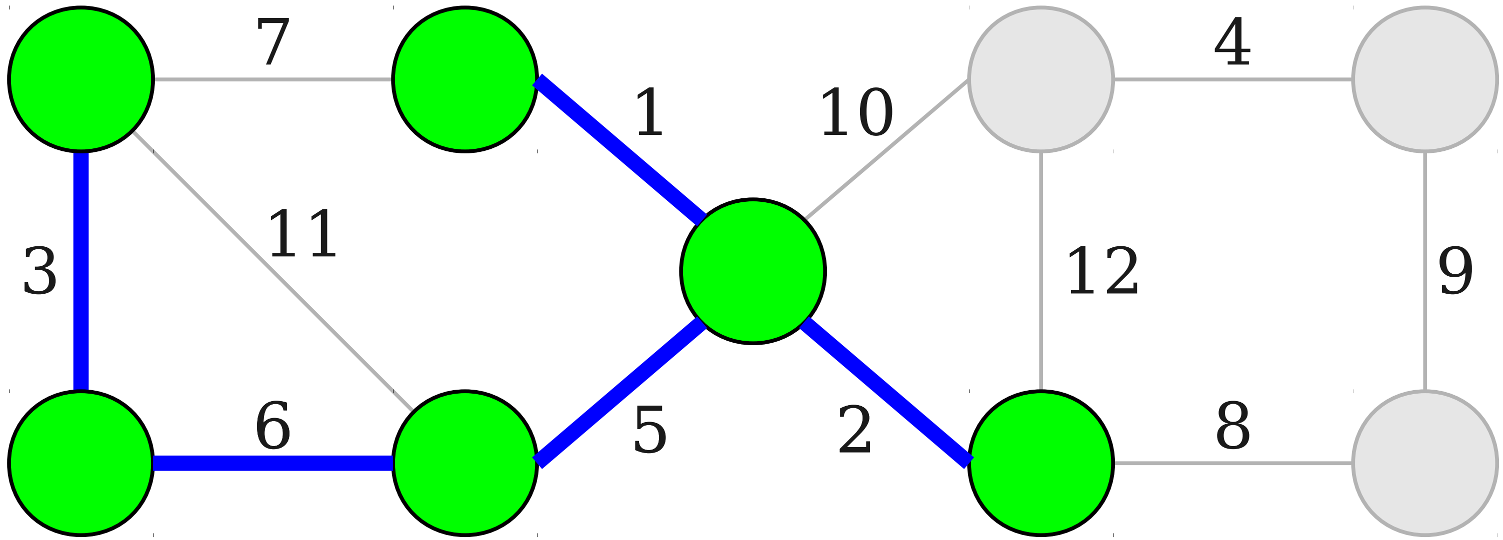

7
1
10
4
3
11
12
9
6
5
2
8

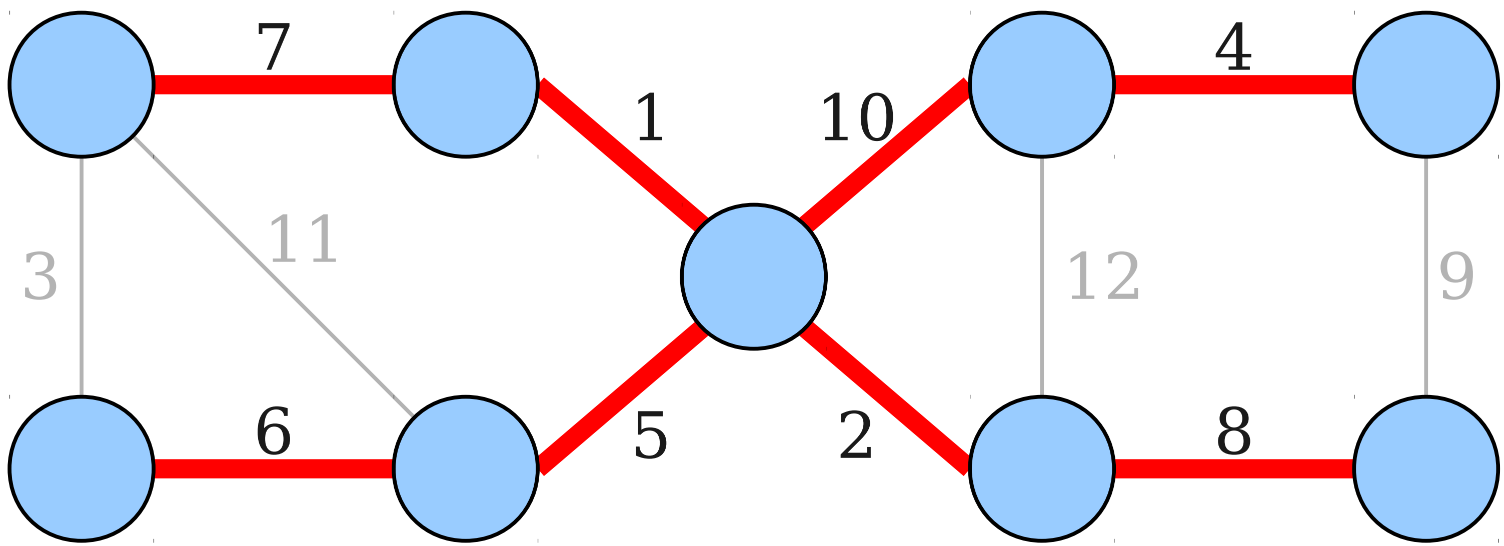

7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

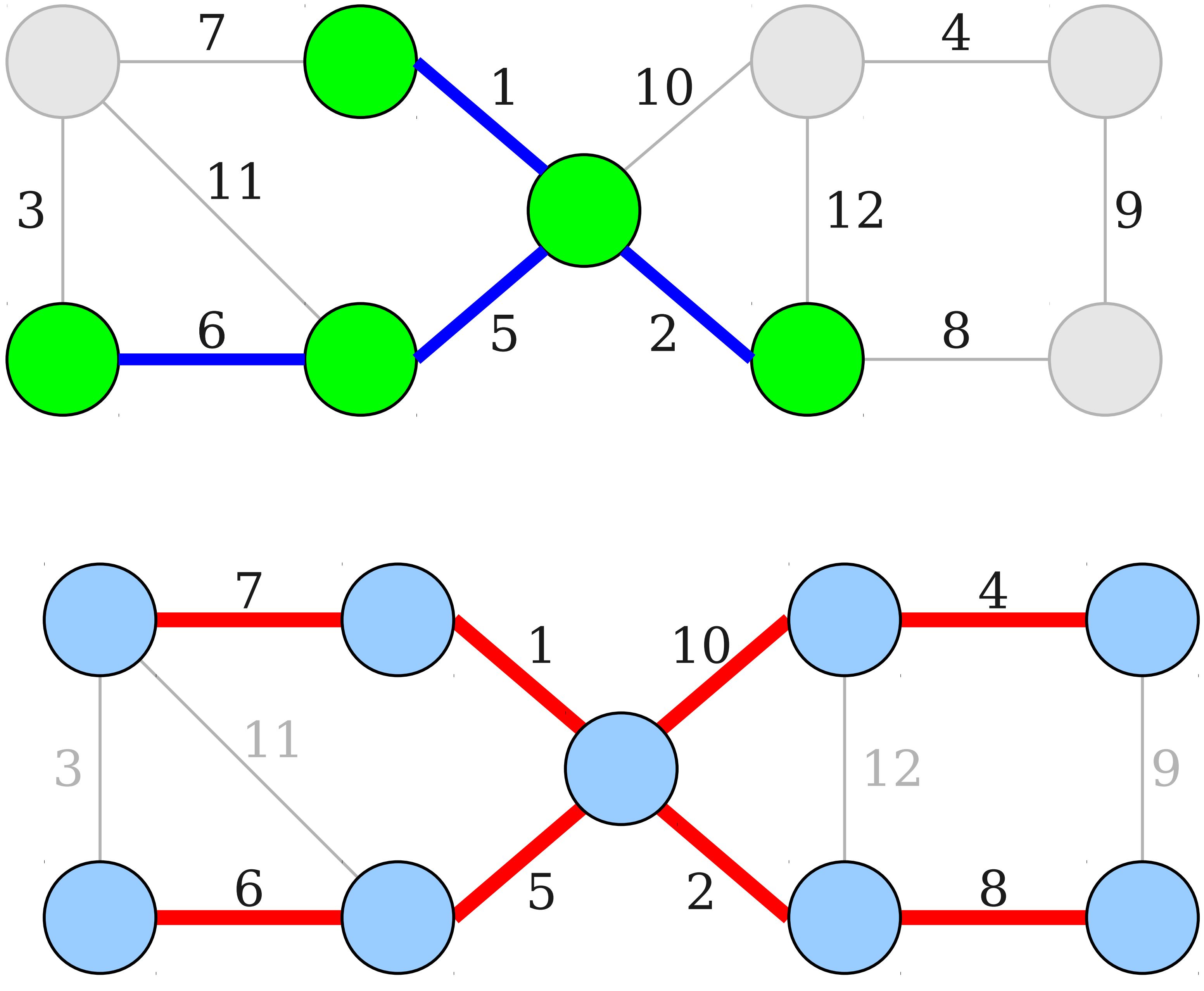

7
1
10
4
11
3
12
9
6
5
2
8


7
1
10
4
11
3
12
9
6
5
2
8

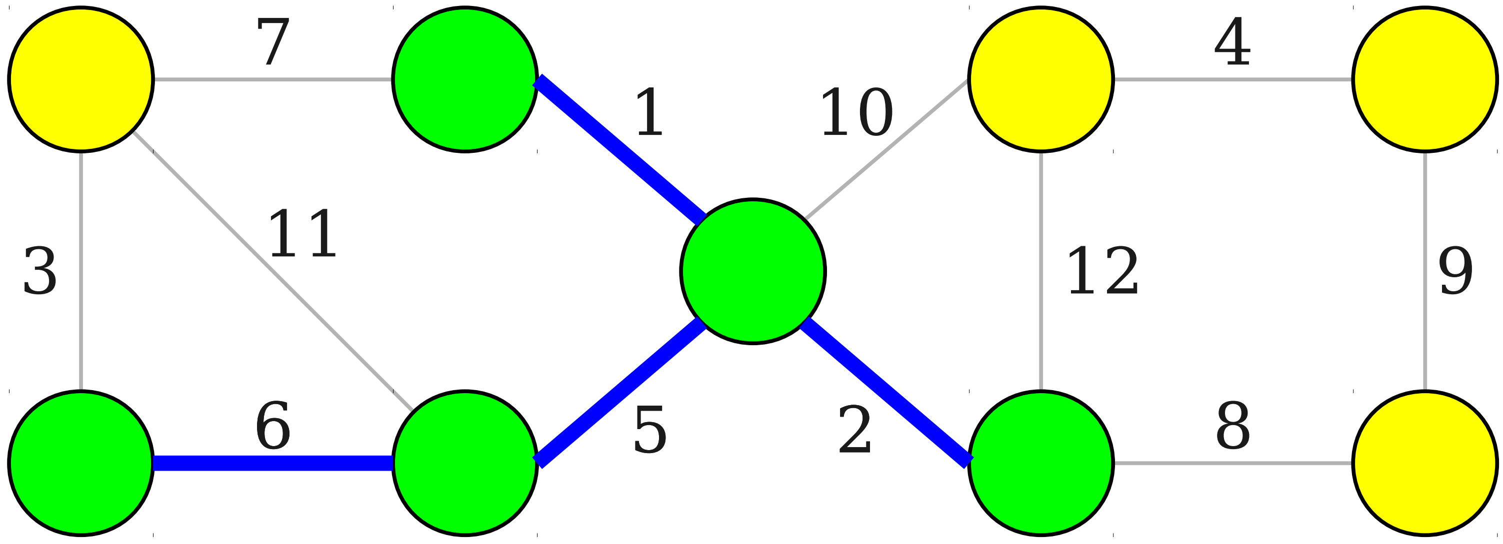

7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

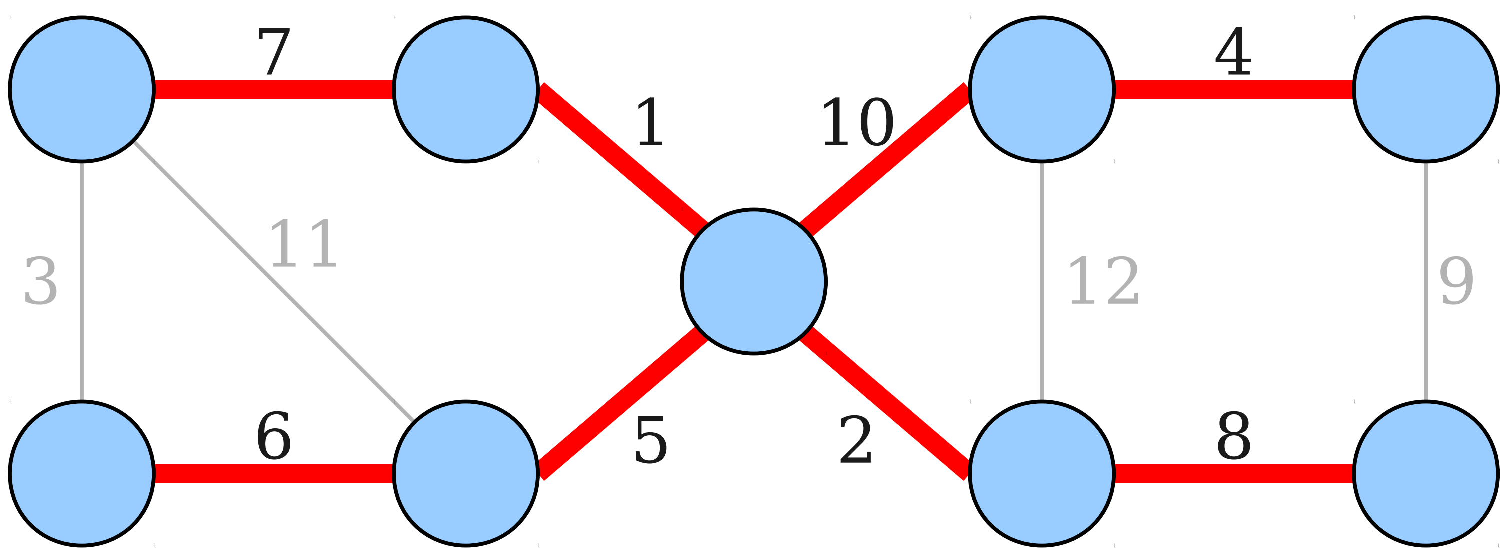

7
1
10
4
3
11
12
9
6
5
2
8

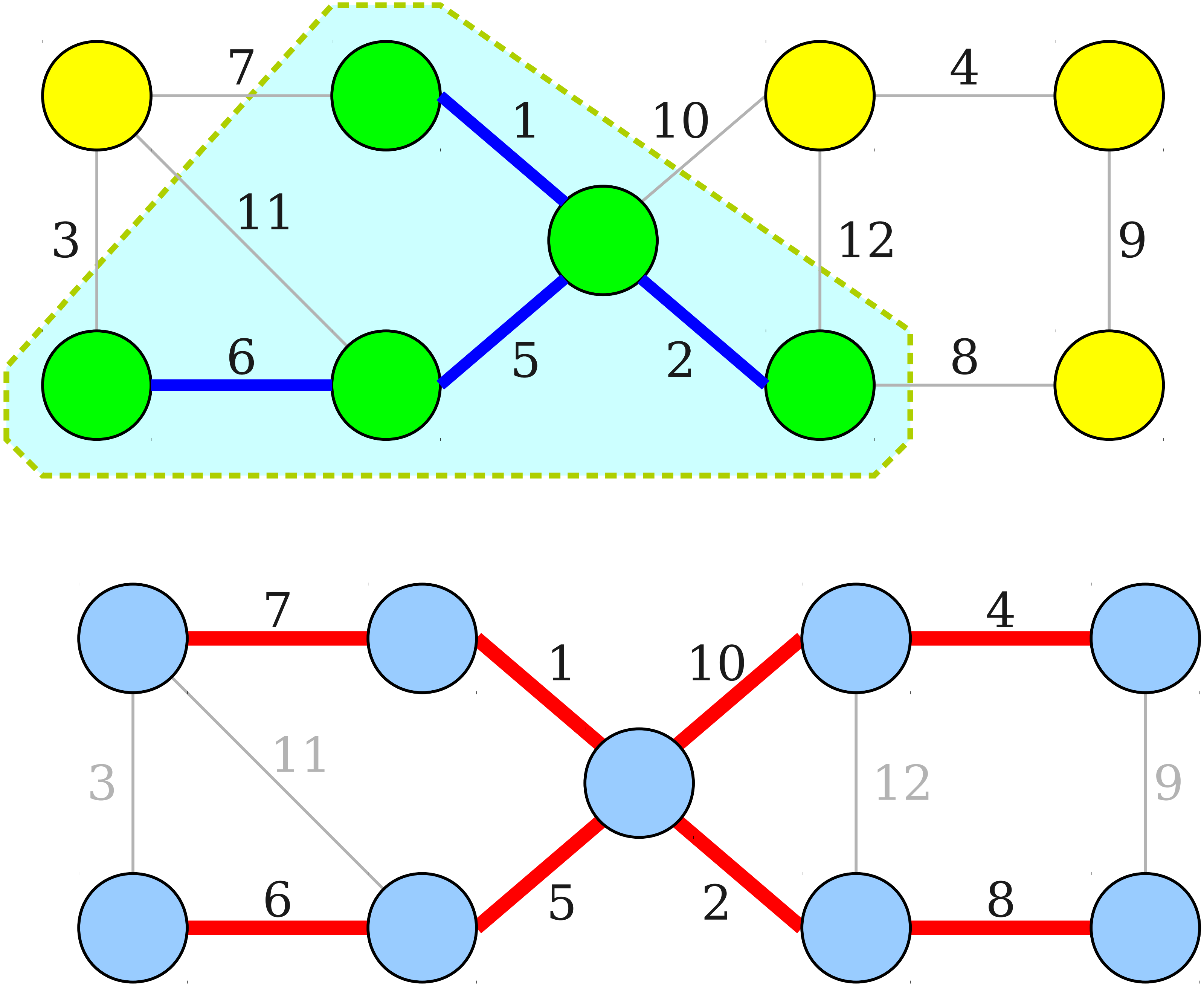

7
1
10
4
3
11
12
9
6
5
2
8



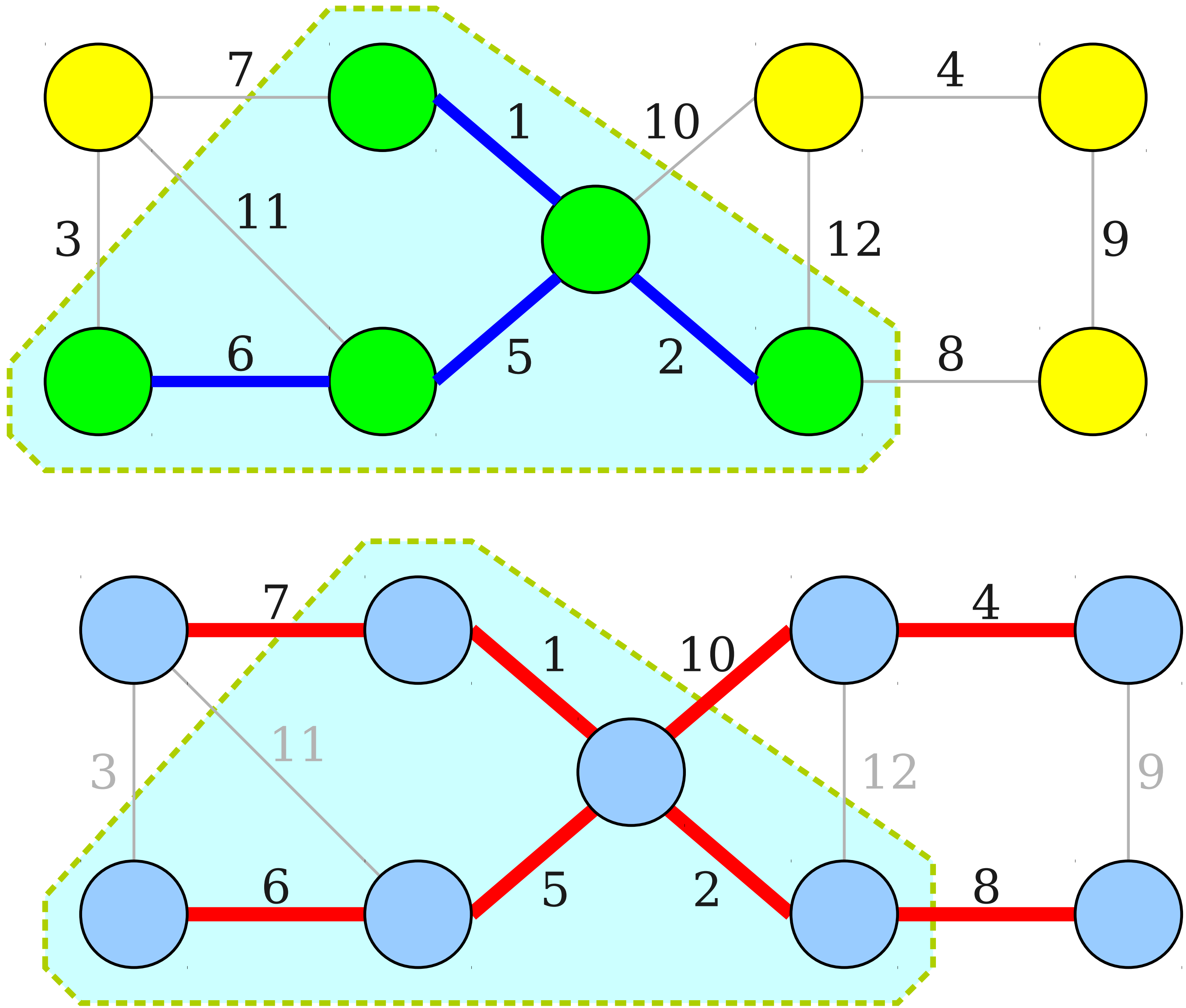
7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

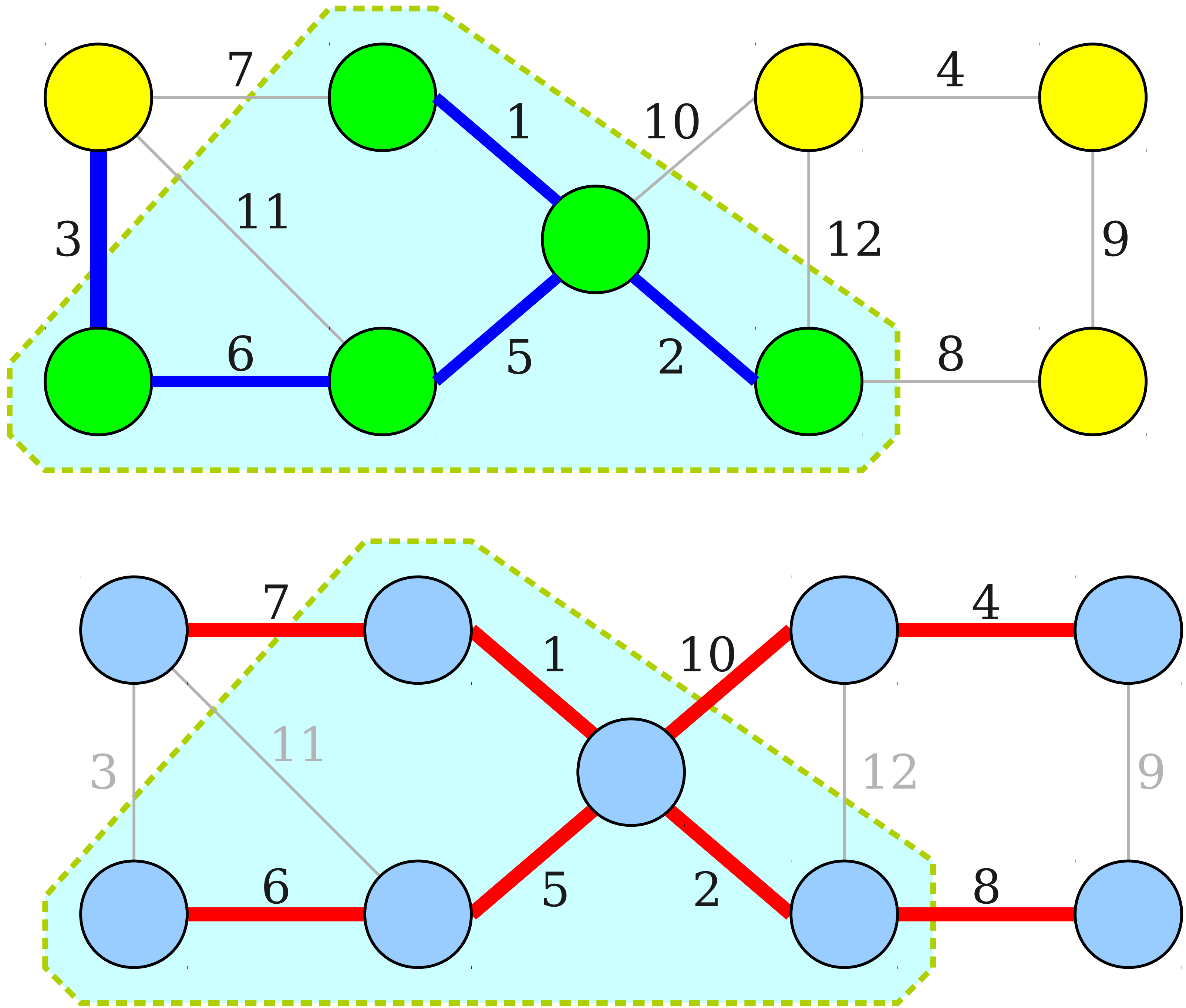

7
1
10
4
3
11
12
9
6
5
2
8

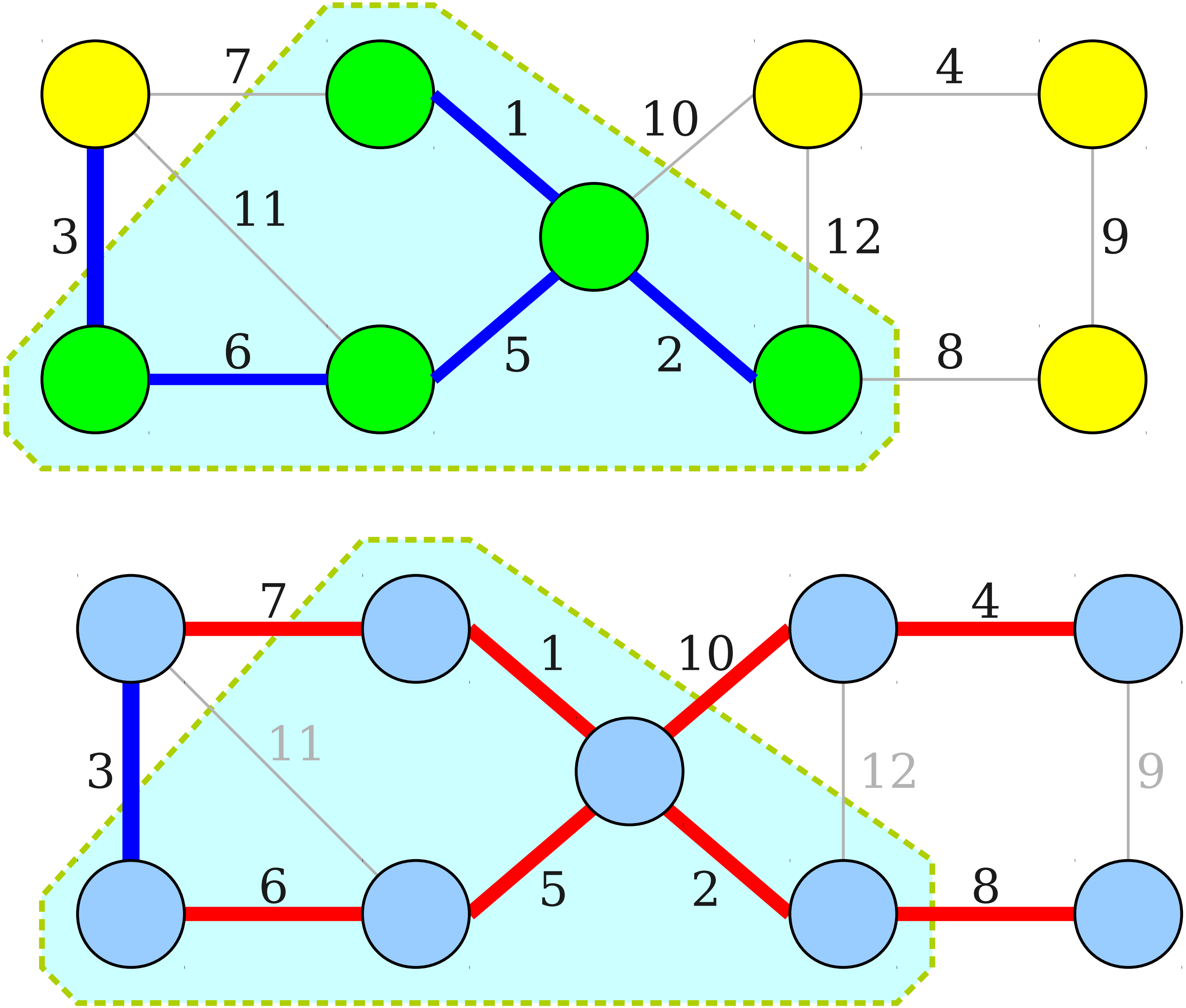

7
1
10
4
3
11
12
9
6
5
2
8



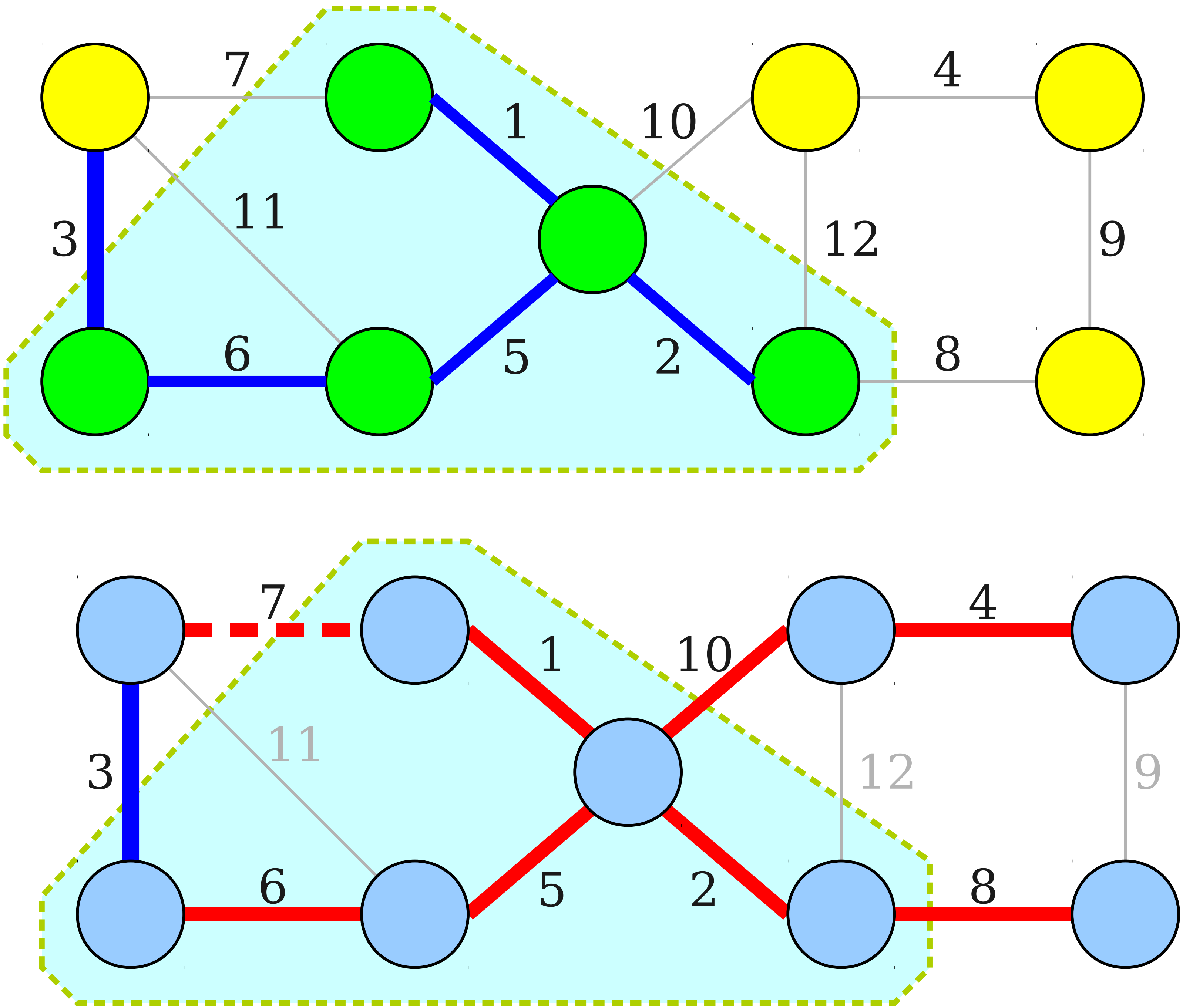
7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8

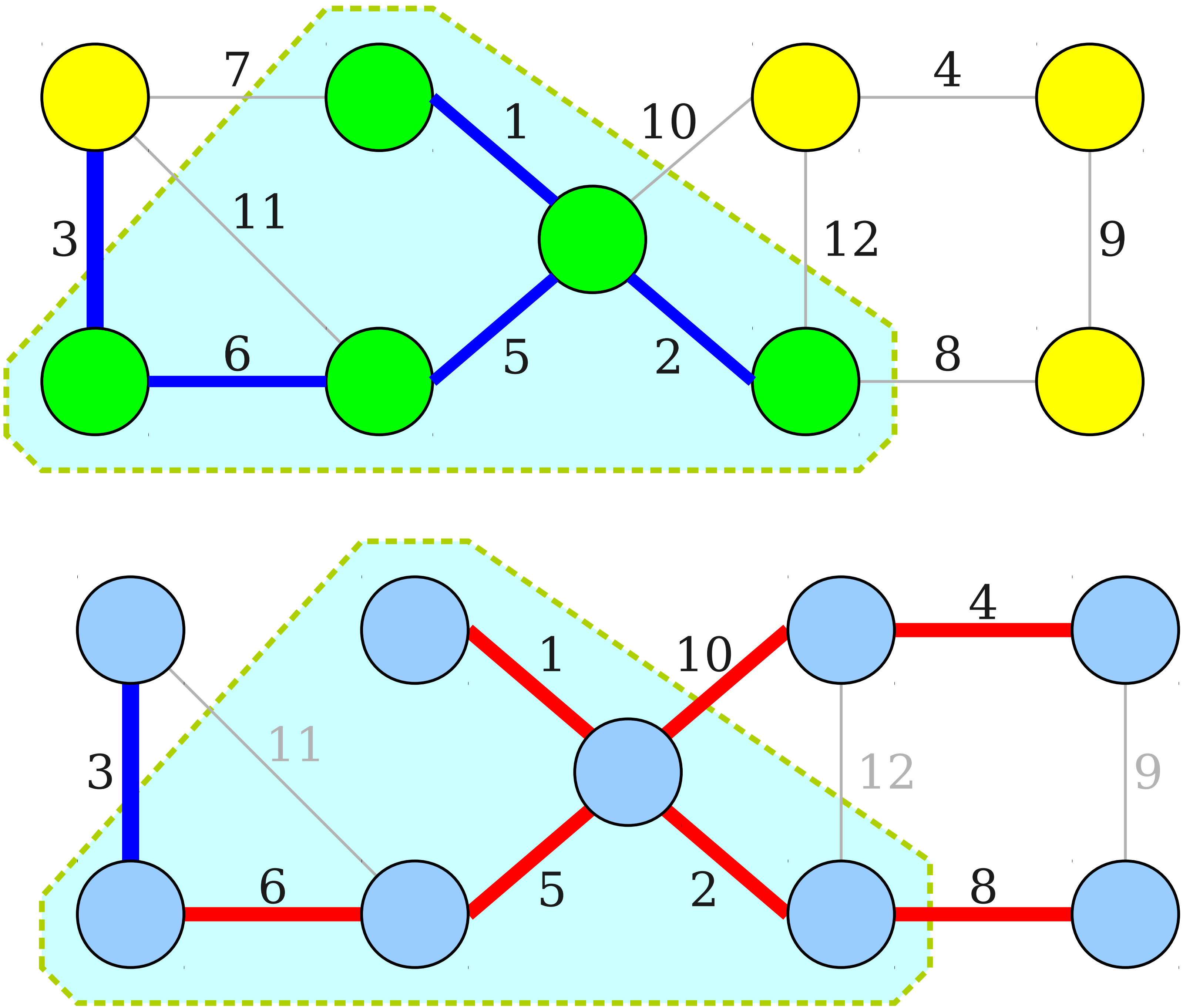

7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8

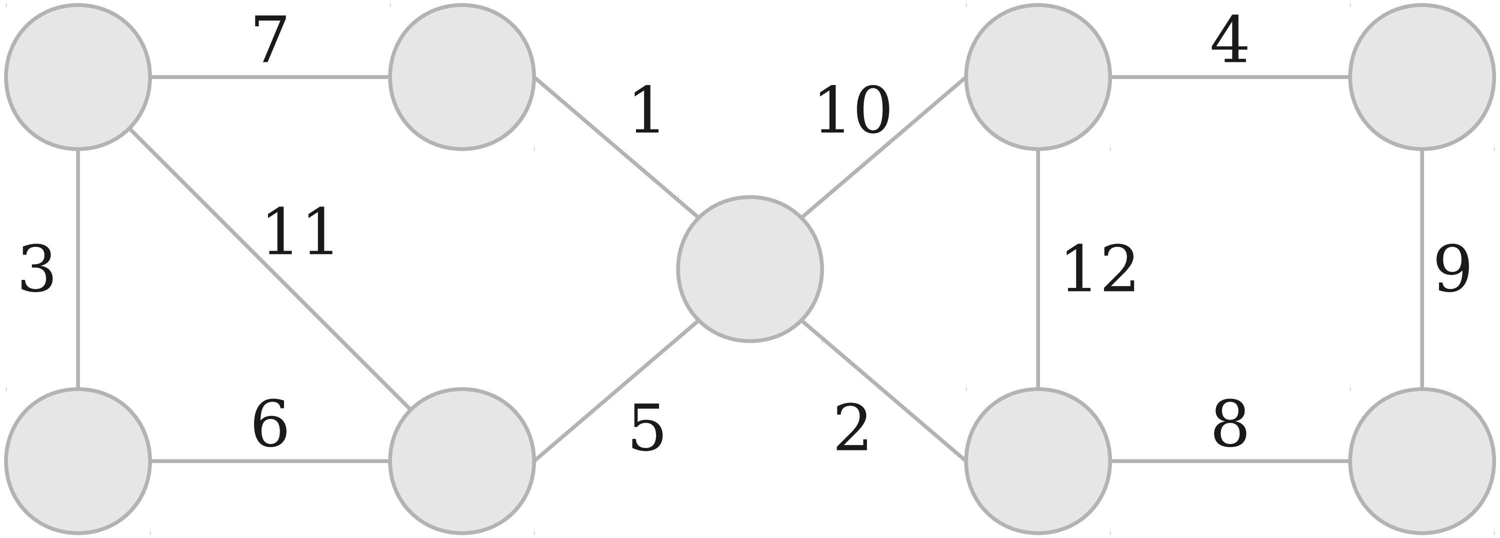

7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8

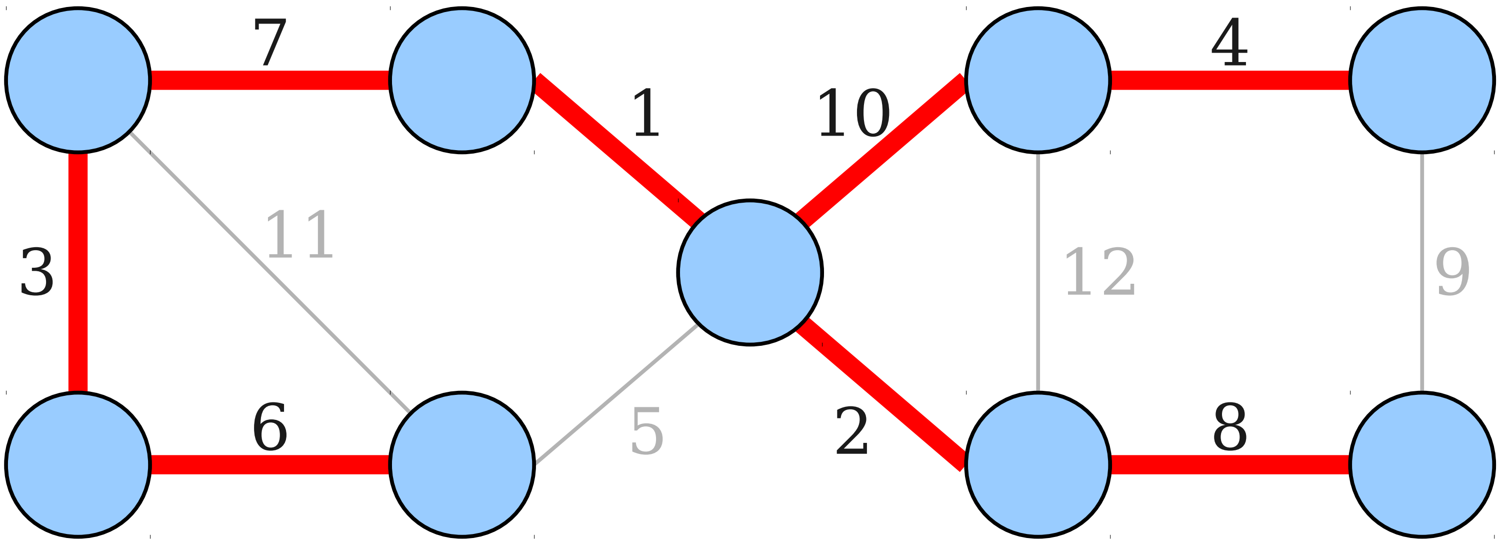


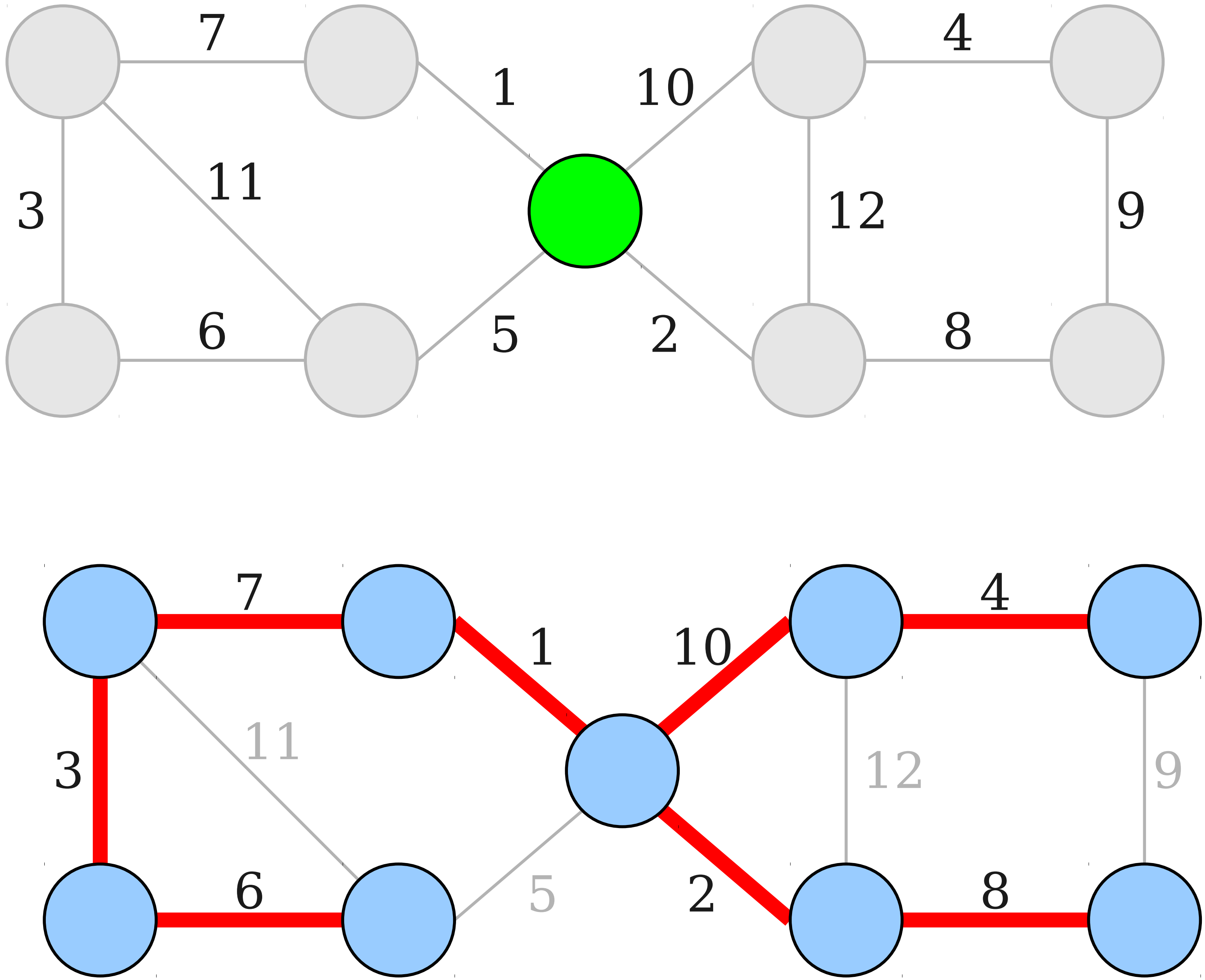
7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

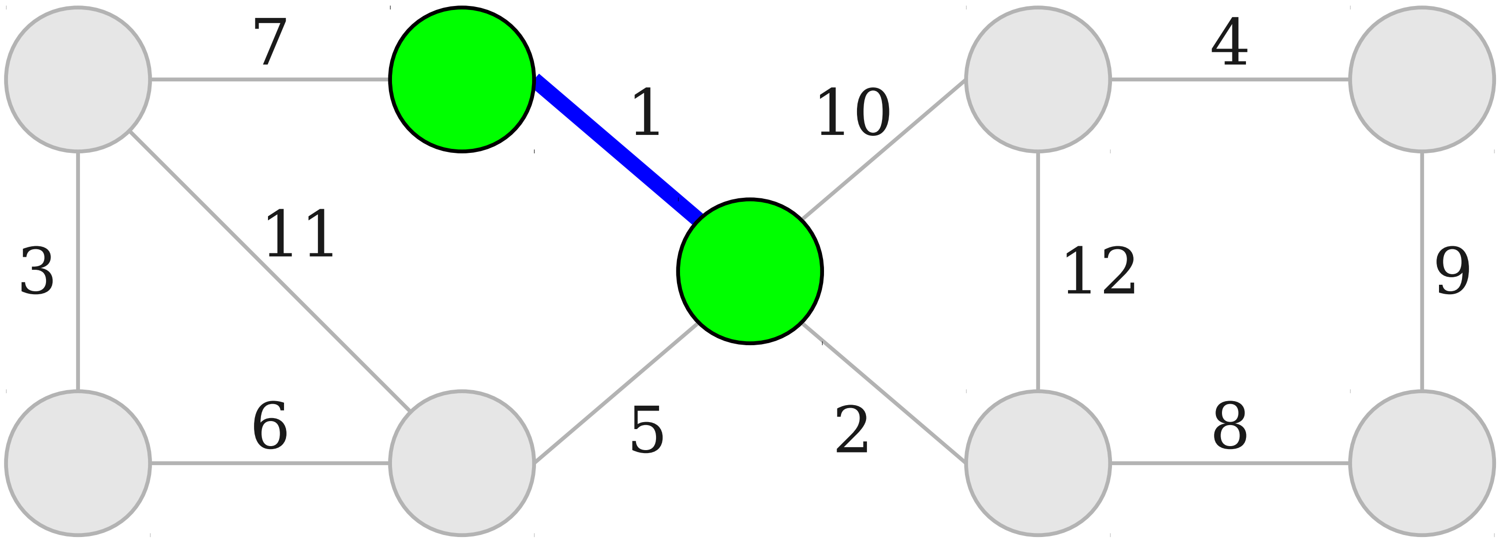

7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

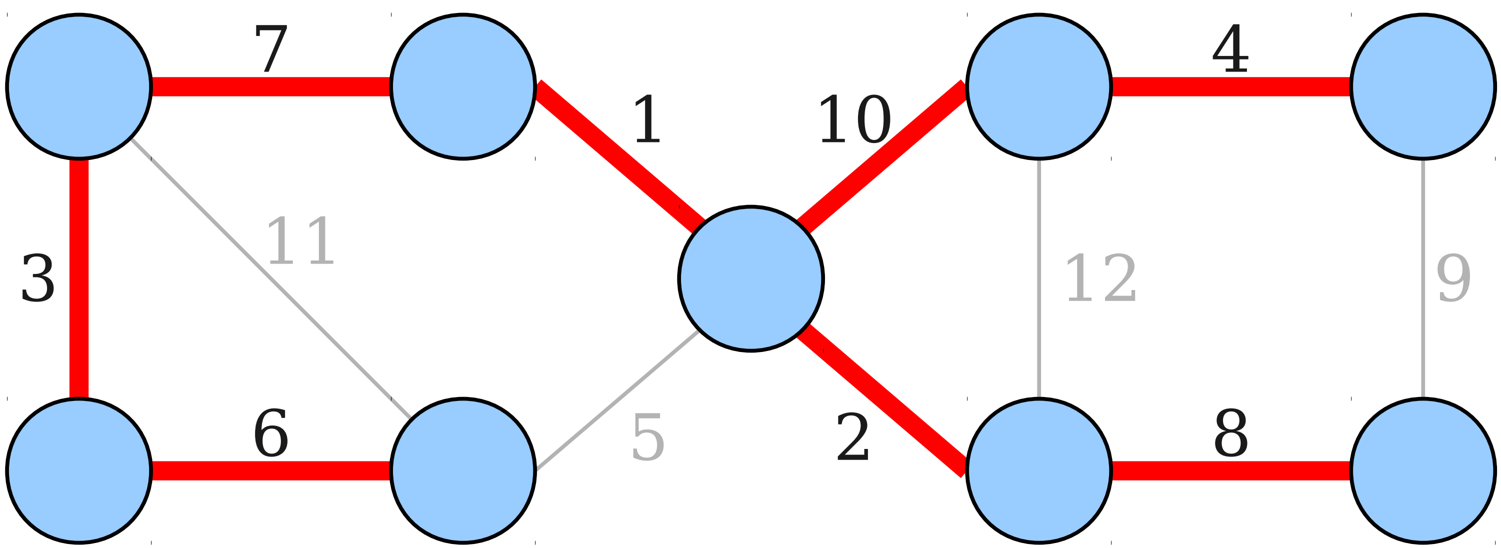

7
1
10
4
11
3
12
9
6
5
2
8
4
1
10
11
3
12
9
6
5
2
8

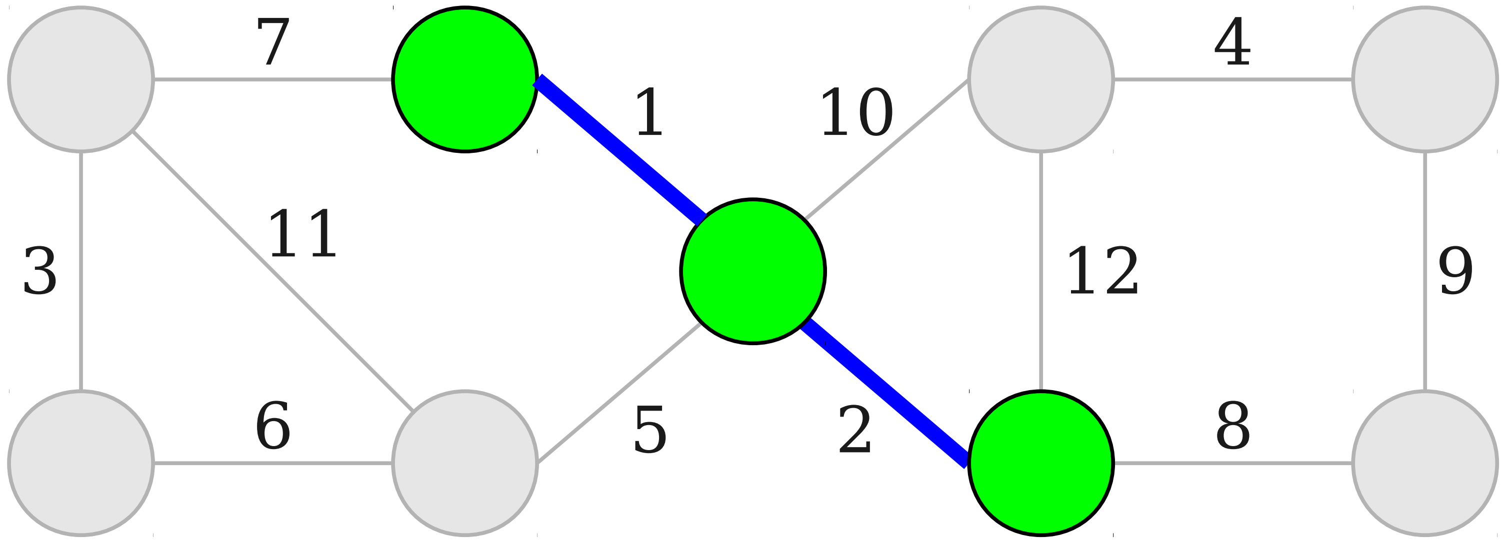

7
1
10
4
11
3
12
9
6
5
2
8

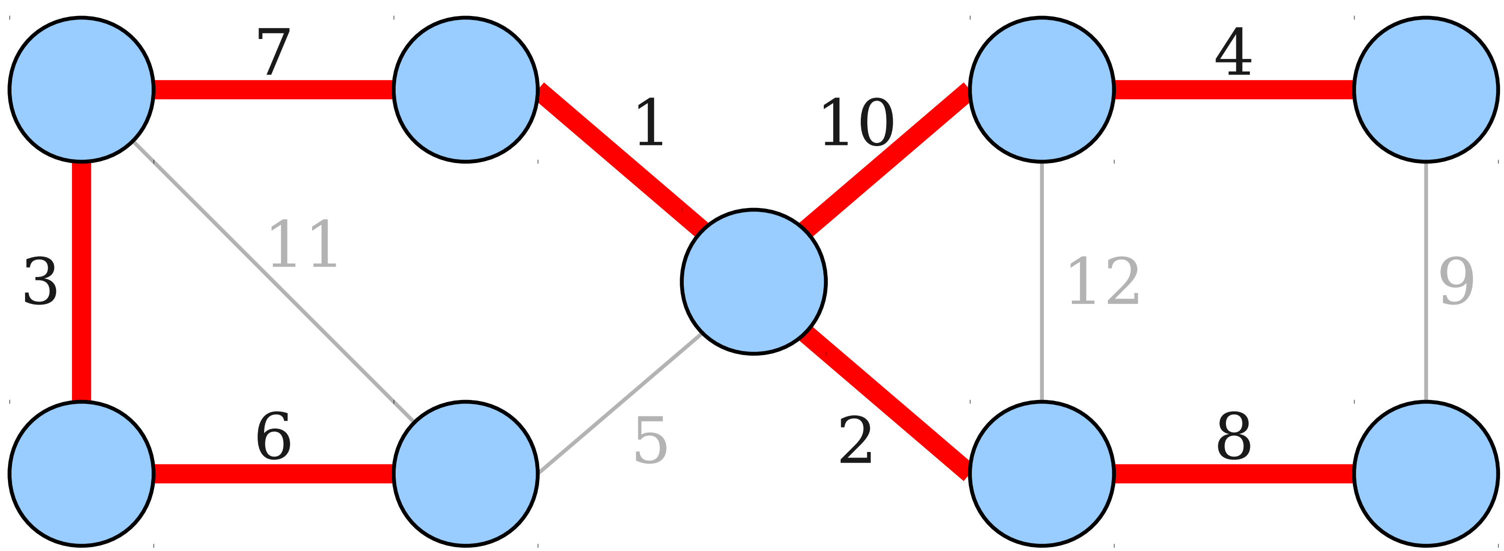

7
1
10
4
11
3
12
9
6
5
2
8

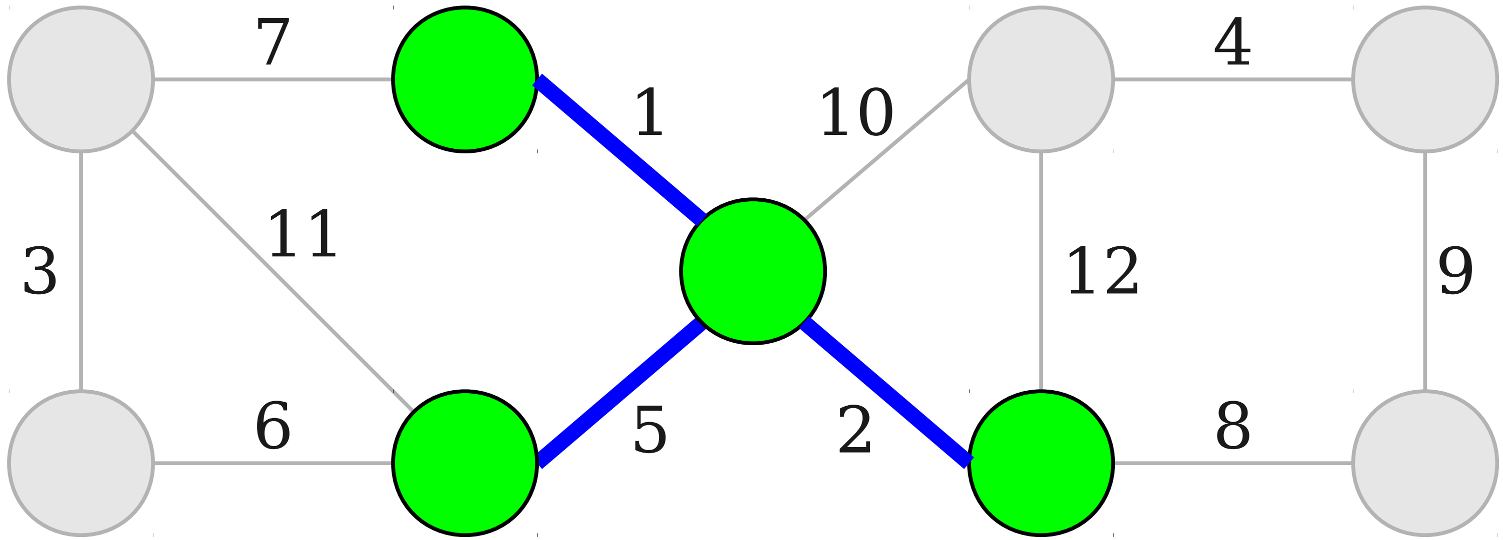

7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8

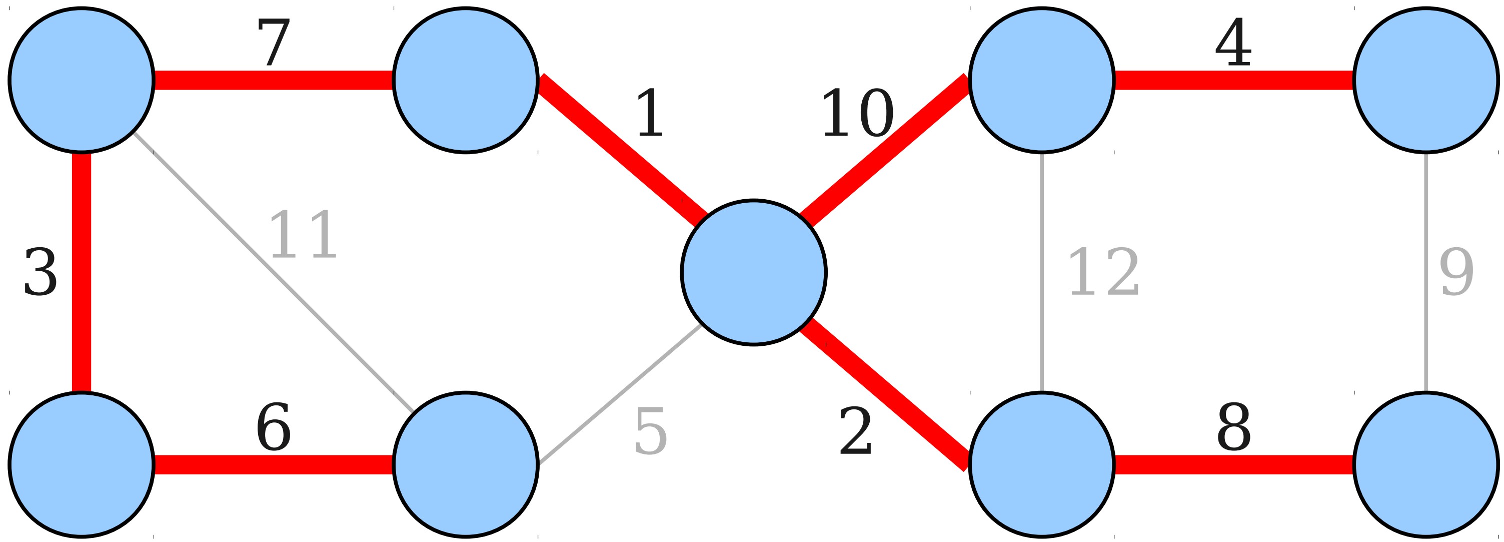

7
1
10
4
3
11
12
9
6
5
2
8

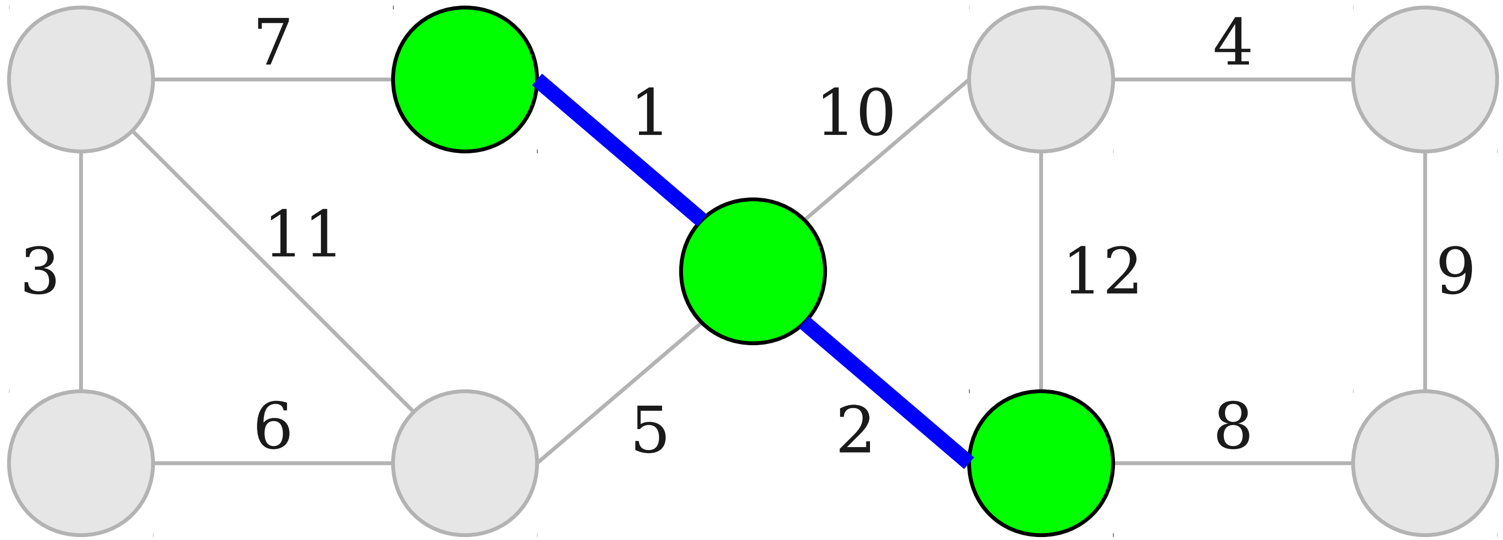

7
1
10
4
3
11
12
9
6
5
2
8

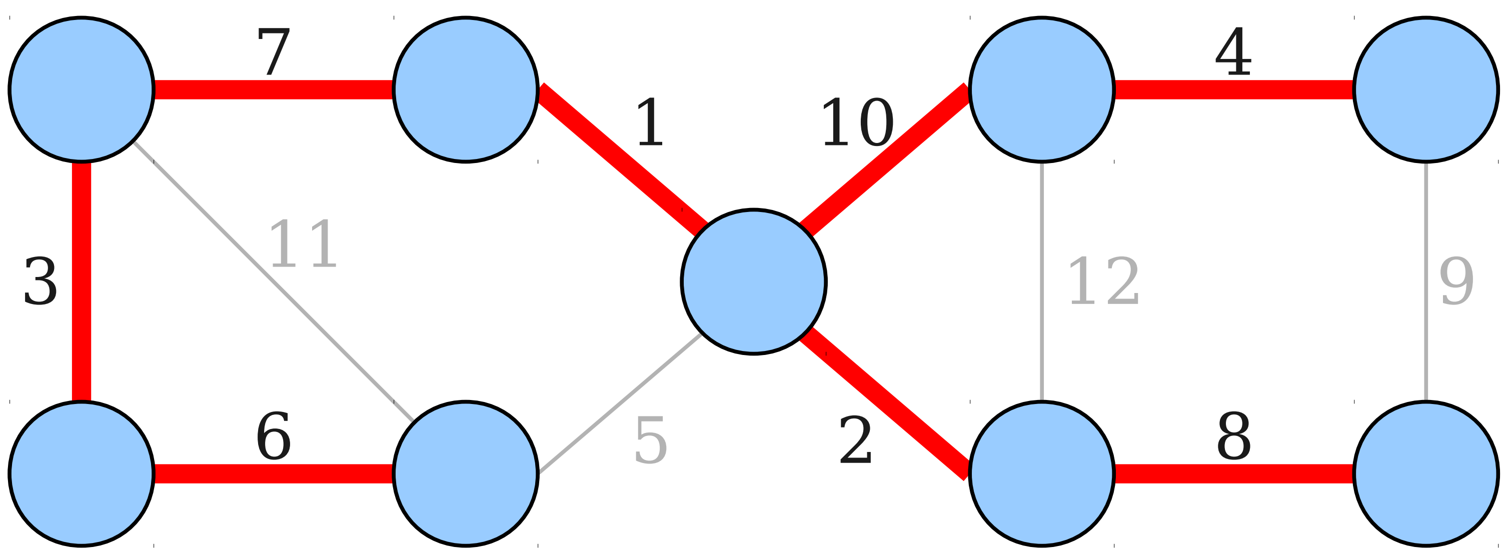

7
1
10
4
3
11
12
9
6
5
2
8

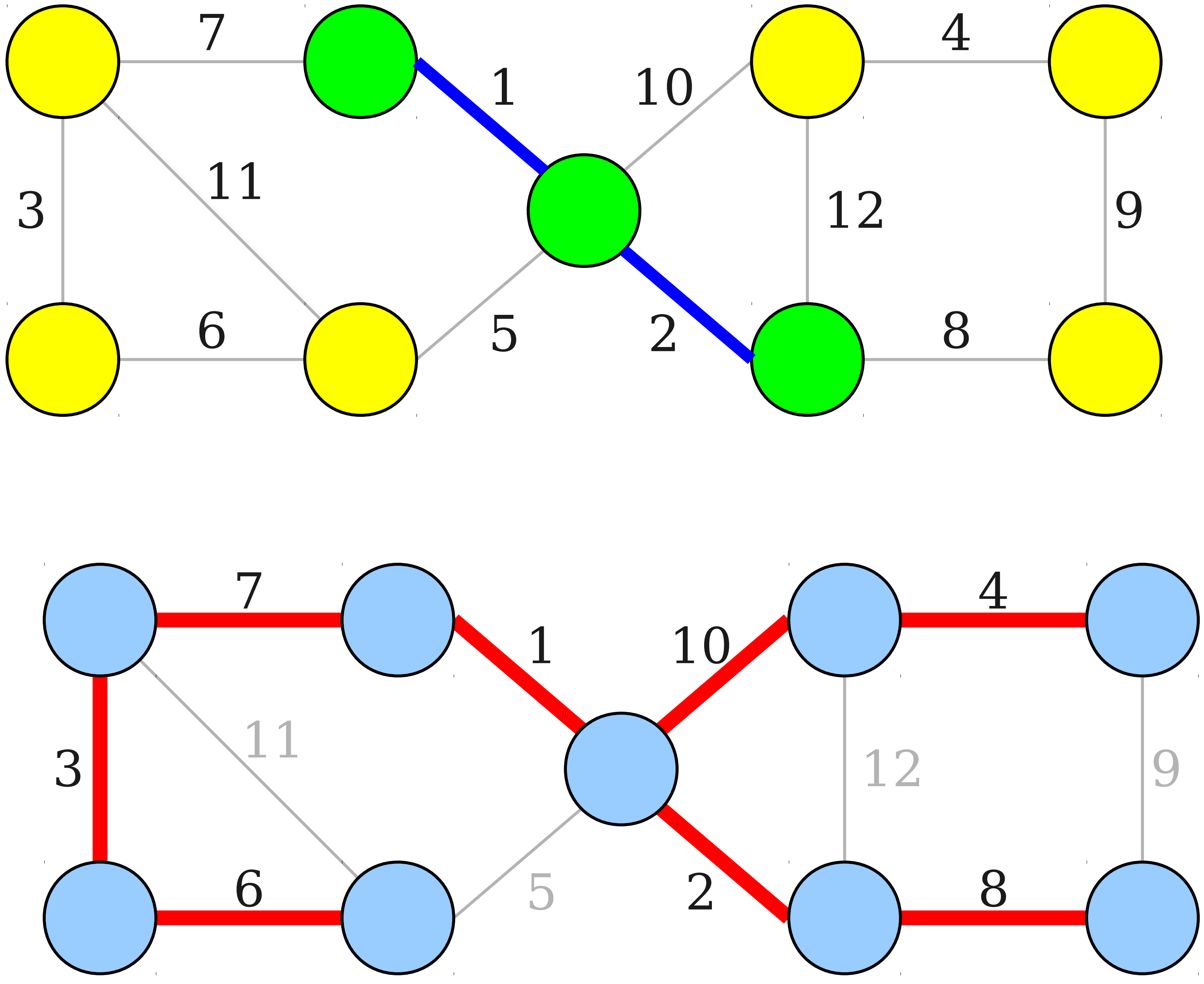

7
1
10
4
3
11
12
9
6
5
2
8

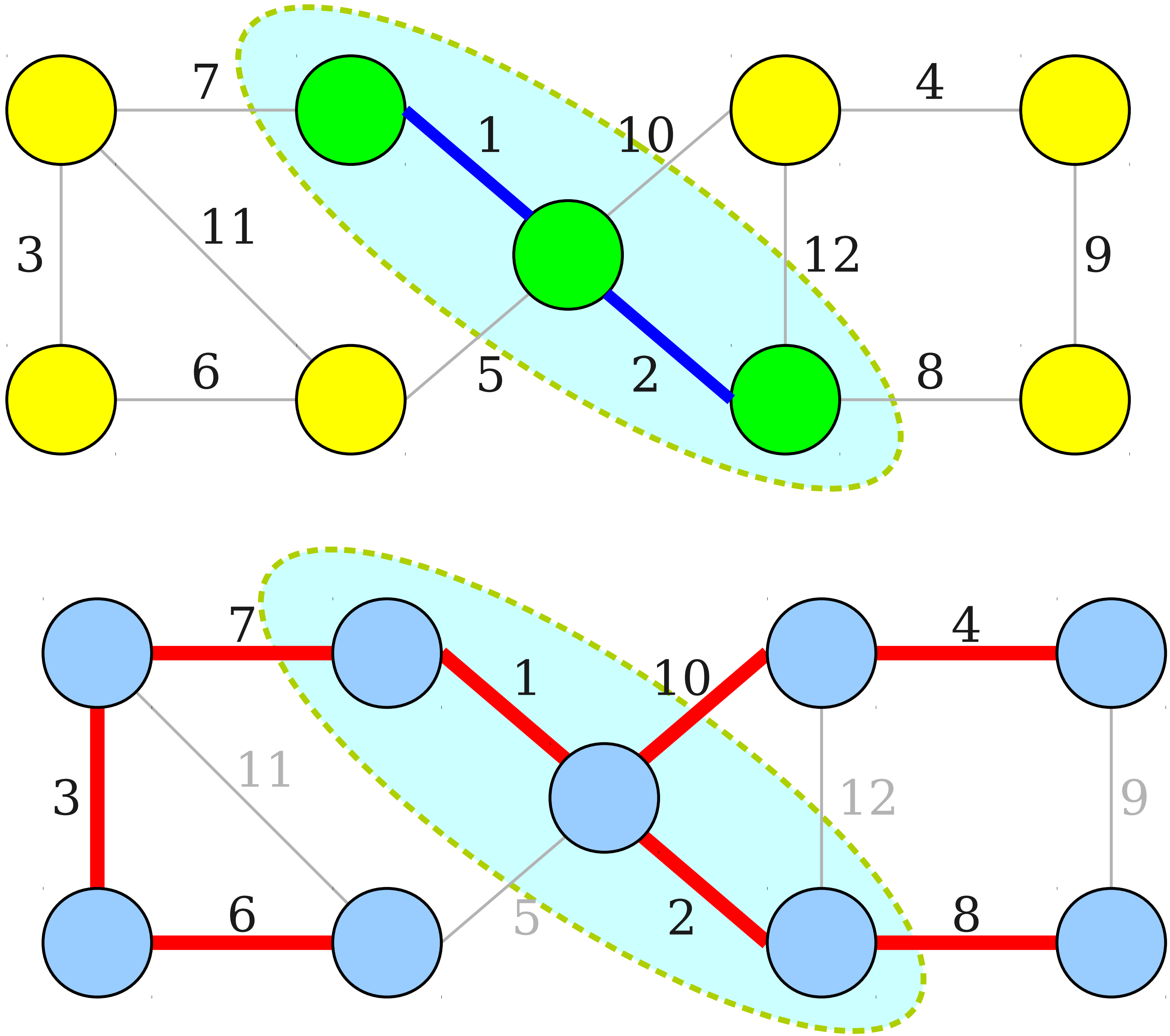

7
1
10
4
11
3
12
9
6
5
2
8

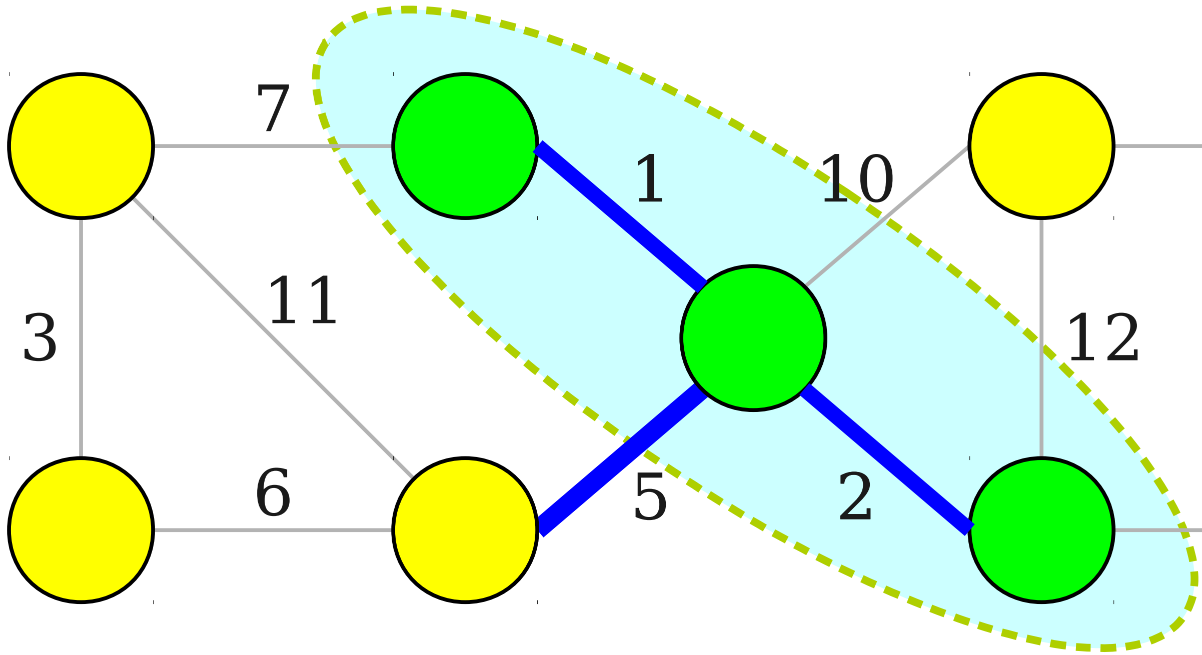

7
1
10
4
11
3
12
9
6
5
2
8

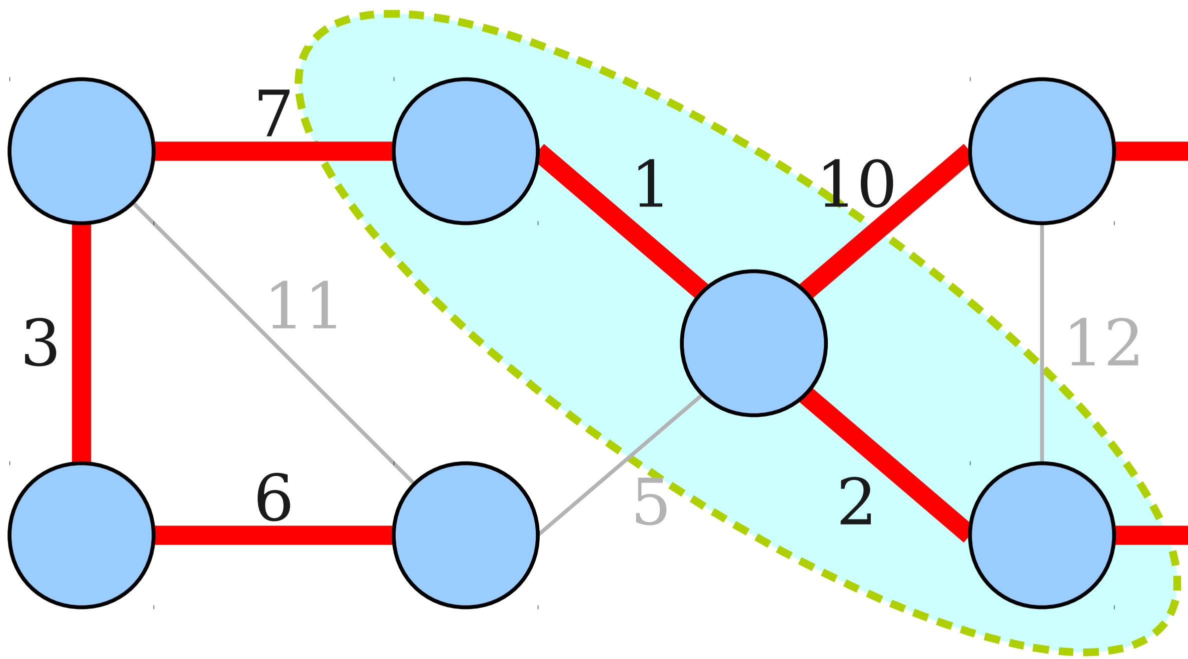

7
1
10
4
3
11
12
9
6
5
2
8

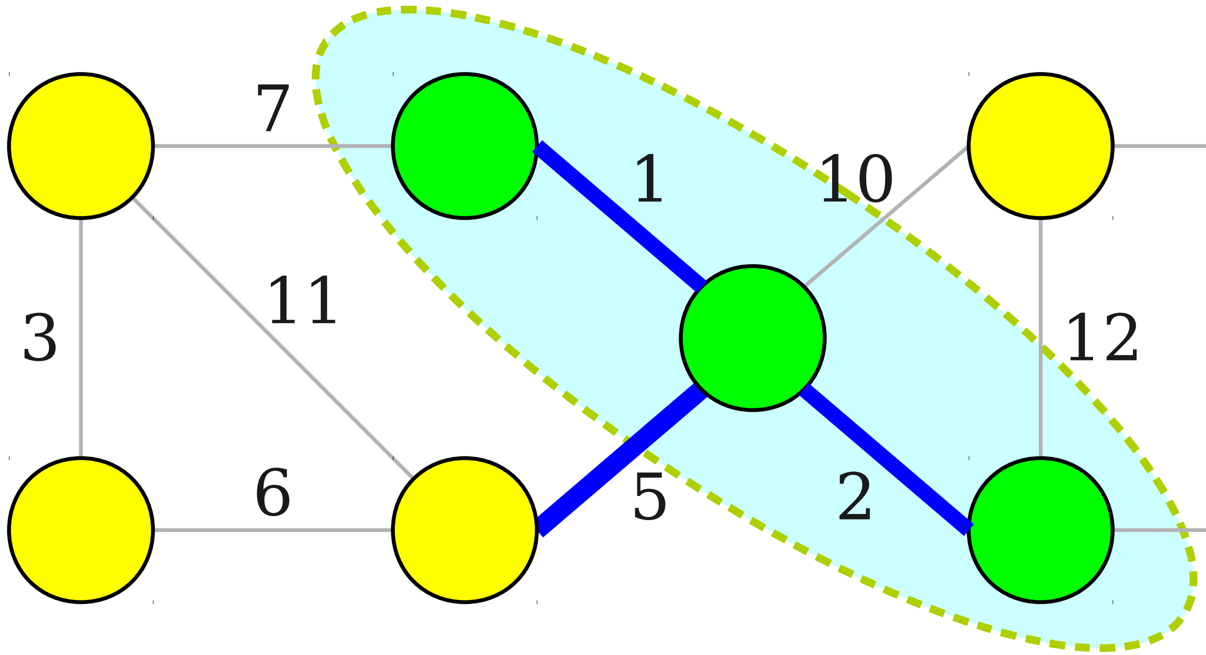

7
1
10
4
3
11
12
9
6
5
2
8

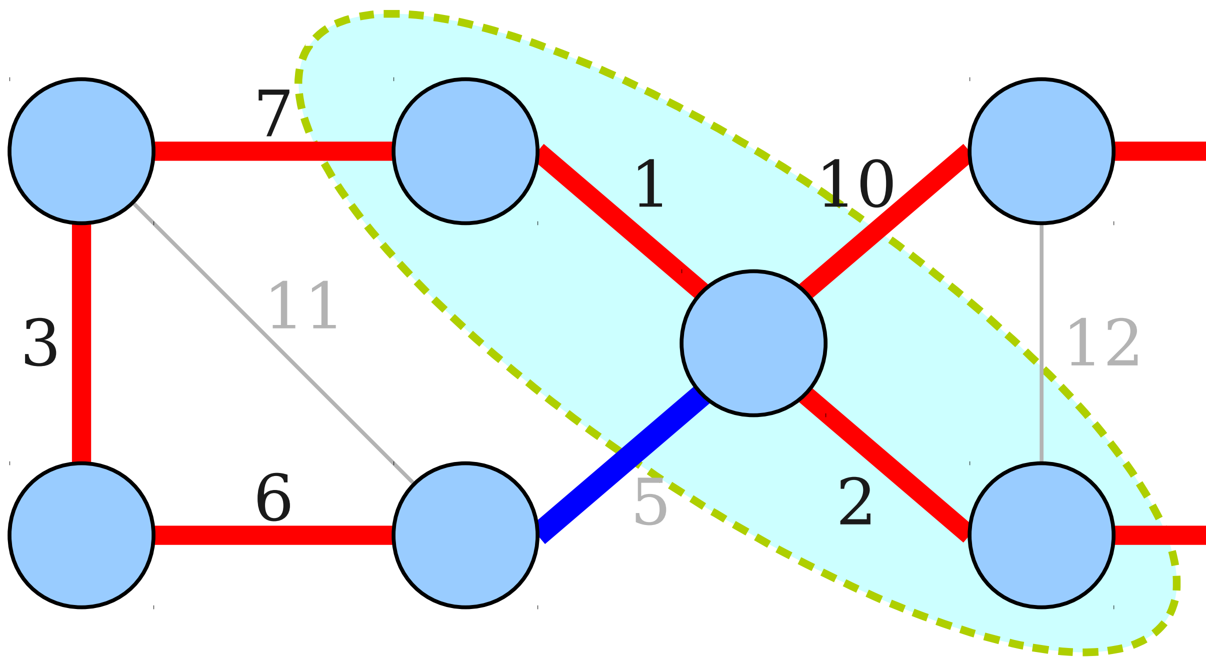

7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8

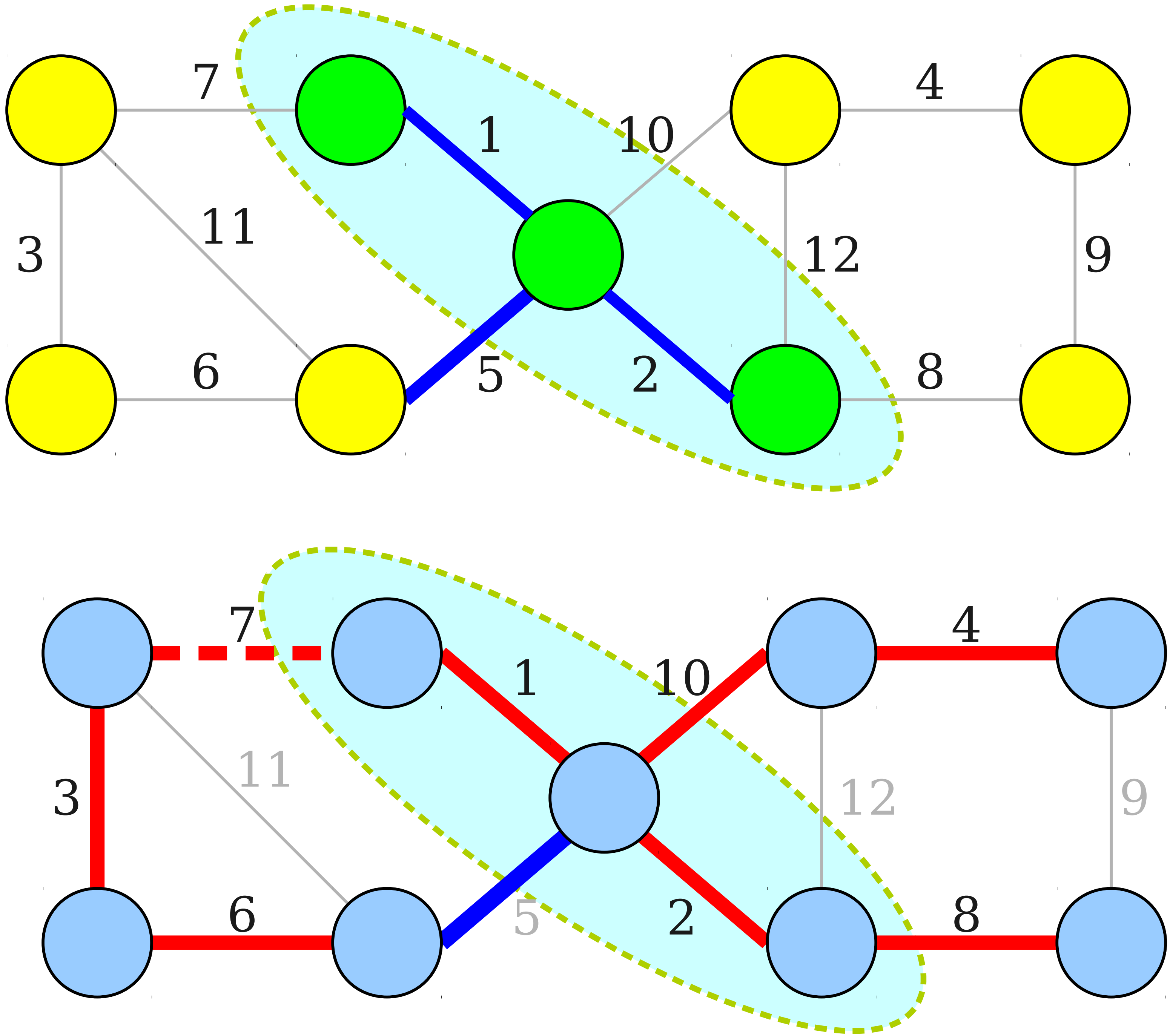

7
1
10
4
11
3
12
9
6
5
2
8
7
1
10
4
11
3
12
9
6
5
2
8


7
1
10
4
3
11
12
9
6
5
2
8

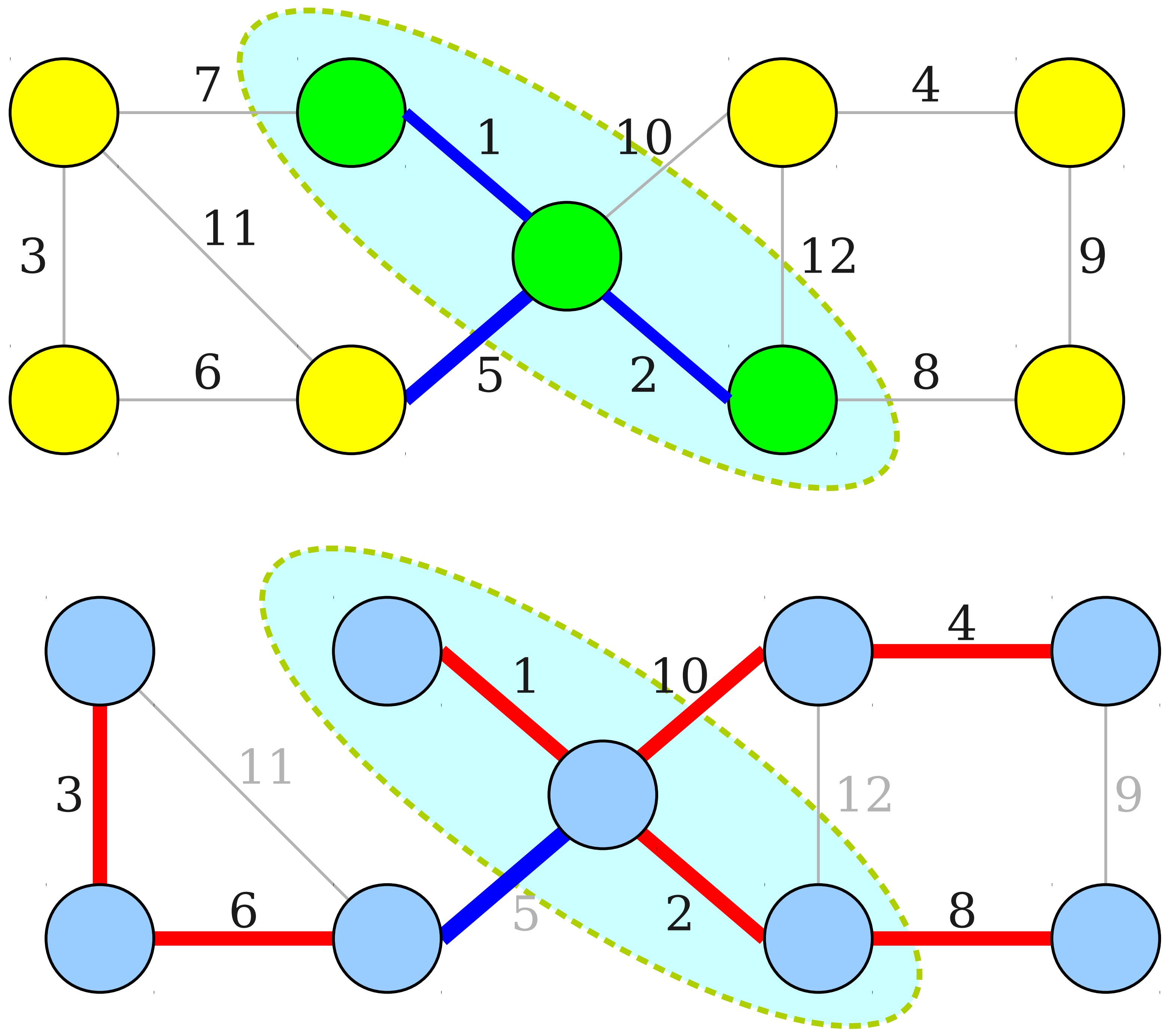

7
1
10
4
3
11
12
9
6
5
2
8


7
4
1
10
11
3
12
9
6
5
2
8


7
4
1
10
11
3
12
9
6
5
2
8



7
1
10
4
3
11
12
9
6
5
2
8
7
1
10
4
3
11
12
9
6
5
2
8



7
1
10
4
3
11
12
9
6
5
2
8
1
10
4
3
11
12
9
6
5
2
8

The Intuition

- By construction, every edge added in Prim's algorithm is the cheapest edge crossing some cut $(S, V - S)$.
- Any tree other than the one produced by Prim's algorithm has to exclude some edge that was included by Prim's algorithm.
- Adding that edge closes a cycle that crosses the cut.
- Deleting an edge in the cycle that crosses the cut strictly lowers the cost of the tree.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is the least-cost edge crossing $(S, V - S)$, we have $c(u, v) < c(x, y)$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is the least-cost edge crossing $(S, V - S)$, we have $c(u, v) < c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \varnothing$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is the least-cost edge crossing $(S, V - S)$, we have $c(u, v) < c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is the least-cost edge crossing $(S, V - S)$, we have $c(u, v) < c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. However, $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) < c(T^*)$, contradicting that T^* is an MST.

Theorem: If G is a connected, weighted graph with distinct edge weights, Prim's algorithm correctly finds an MST.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be the MST of G. We will prove $T = T^*$ by contradiction. Assume $T \neq T^*$. Therefore, $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

When (u, v) was added to T, it was the least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is the least-cost edge crossing $(S, V - S)$, we have $c(u, v) < c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. However, $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) < c(T^*)$, contradicting that T^* is an MST.

We have reached a contradiction, so our assumption must have been wrong. Thus $T = T^*$, so T is an MST. ■

Exchange Arguments

- This proof of optimality for Prim's algorithm uses an argument called an ***exchange argument***.
- General structure is as follows *
    - Assume the greedy algorithm does not produce the optimal solution, so the greedy and optimal solutions are different.
    - Show how to *exchange* some part of the optimal solution with some part of the greedy solution in a way that improves the optimal solution.
    - Reach a contradiction and conclude the greedy and optimal solutions must be the same.
- *(* This assumes there is a **unique** optimal solution; we'll generalize this shortly.)*

The Cut Property

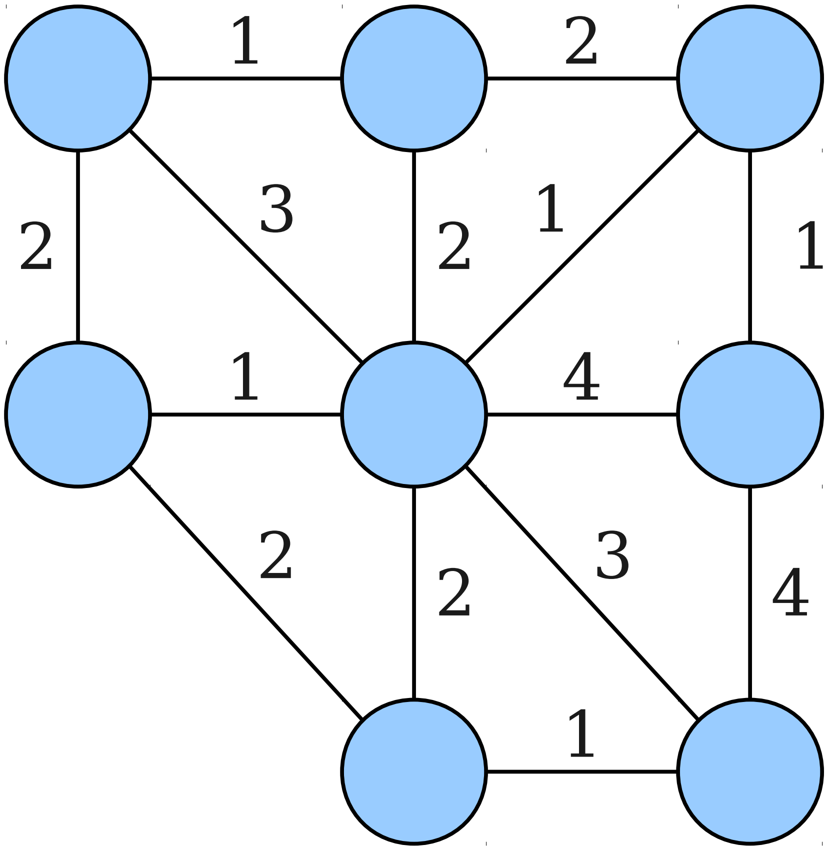
- The previous correctness proof relies on a property of MSTs called the *cut property*:

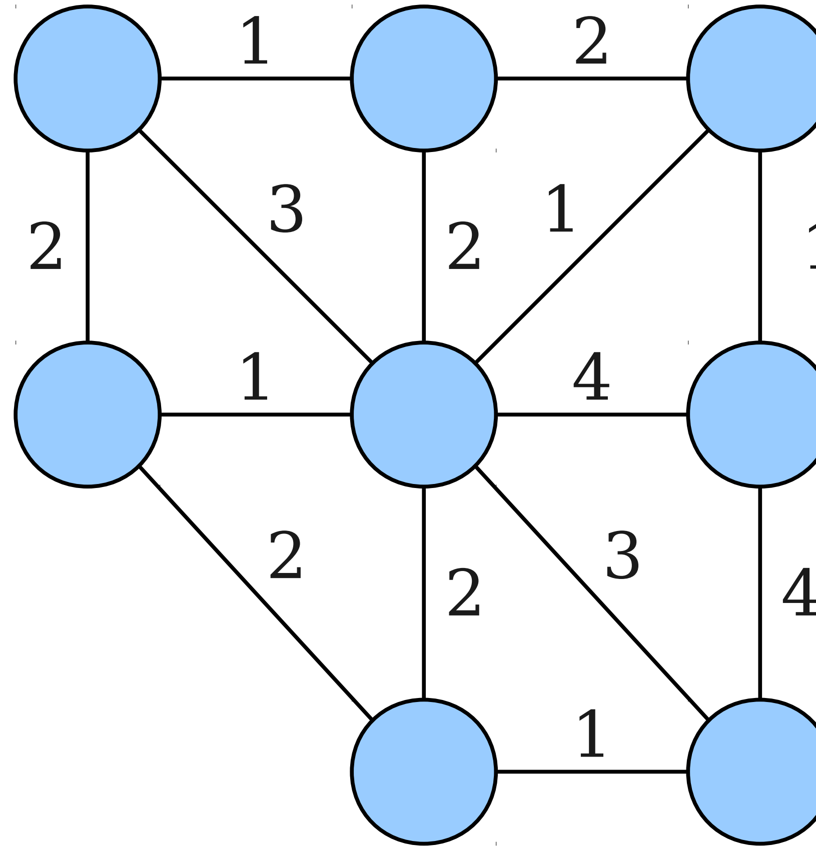
  Theorem (Cut Property): Let $(S, V - S)$ be a nontrivial cut in G (i.e. $S \neq \emptyset$ and $S \neq V$). If (u, v) is the lowest-cost edge crossing $(S, V - S)$, then (u, v) is in every MST of G.

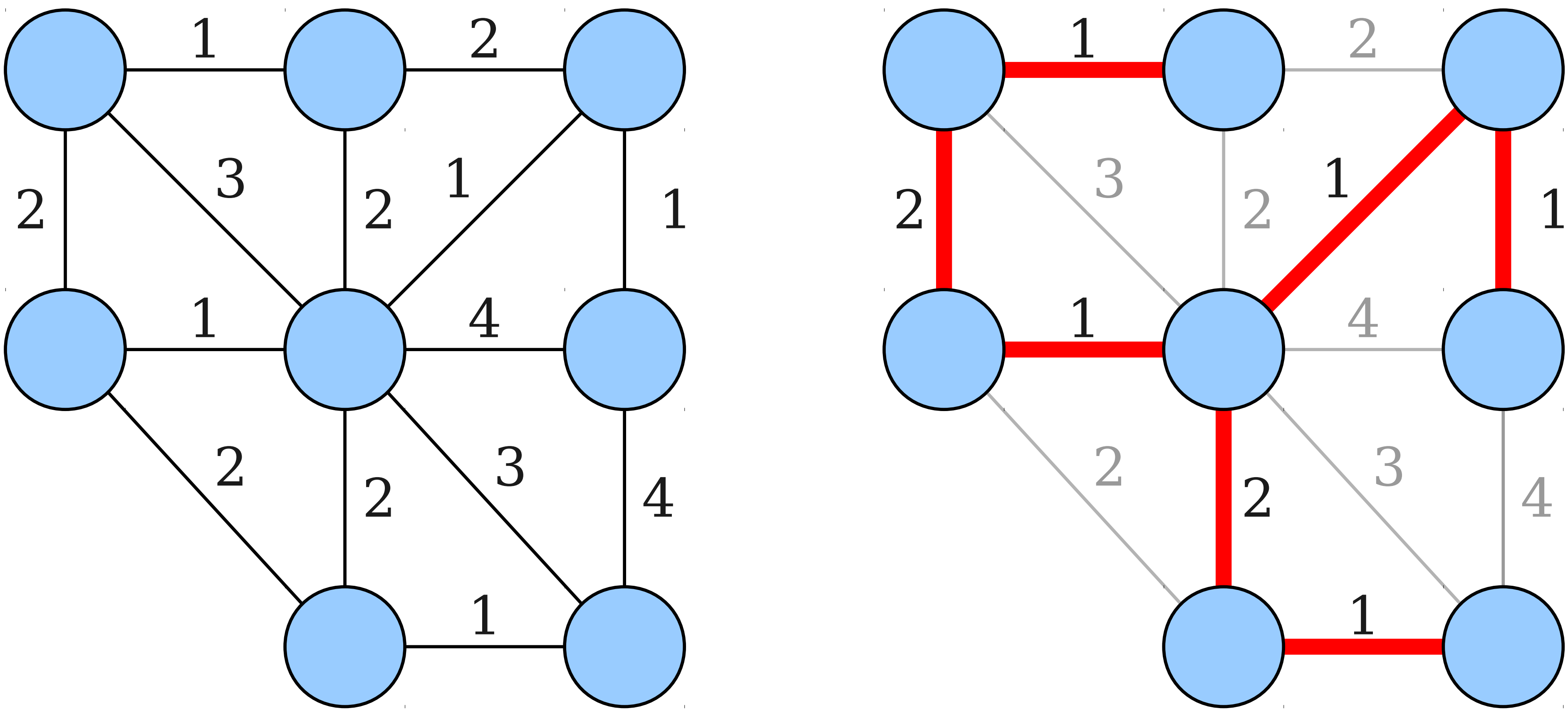
- Proof uses an exchange argument: swap out the lowest-cost edge crossing the cut for some other edge crossing the cut.

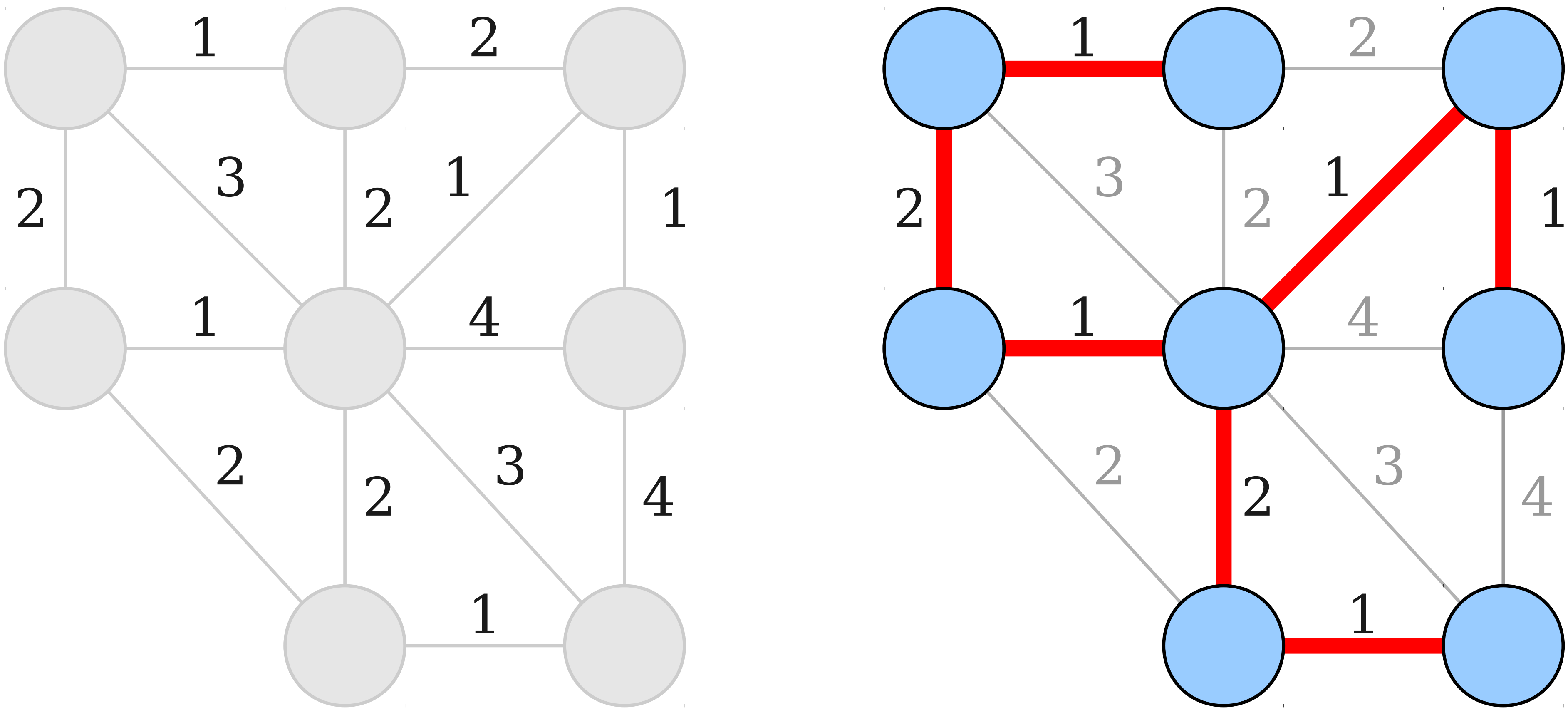
One Problem

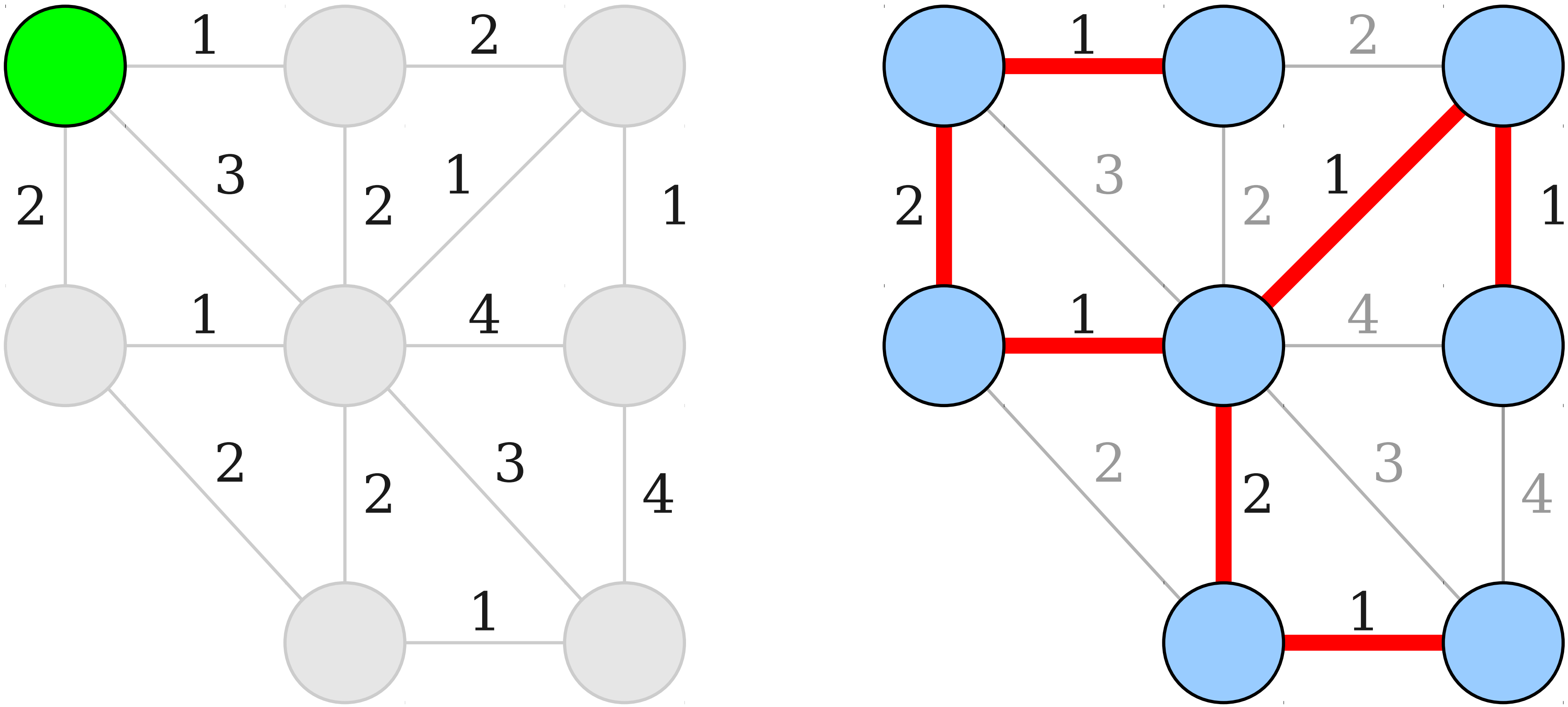
- This proof of correctness relies on edge weights being distinct in two ways:
  - Assumes there is a **unique** MST in the graph.
  - Assumes swapping one edge crossing the cut for another **strictly** improves the cost of an alleged MST.
- Neither of these are true if weights can be duplicated.
- How do we account for this?


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

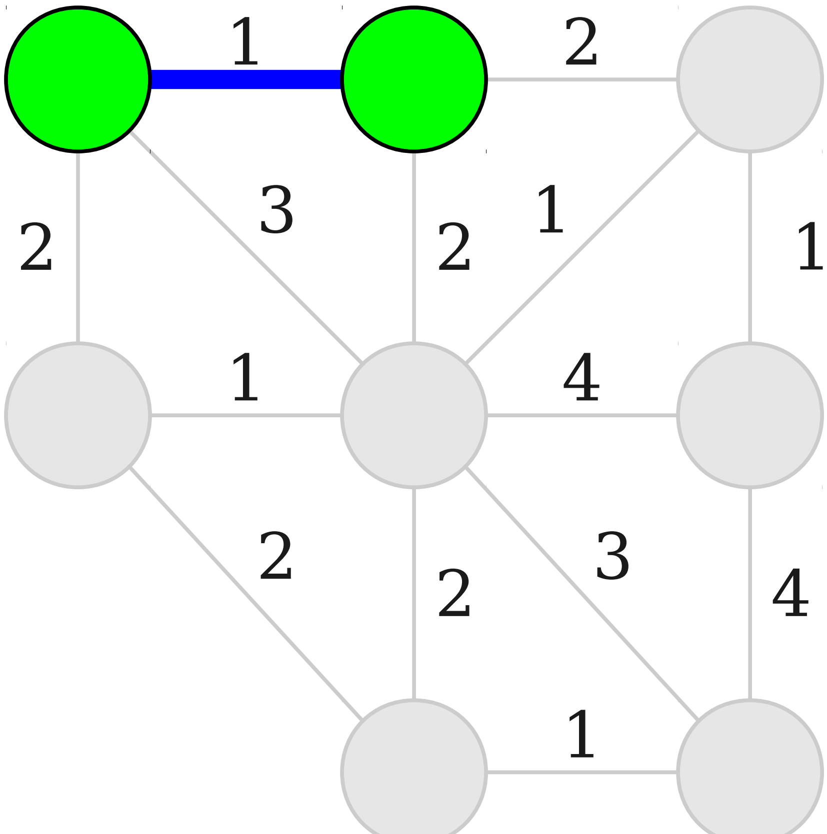

1
2
3
1
2
2
1
1
4
2
3
2
4
1

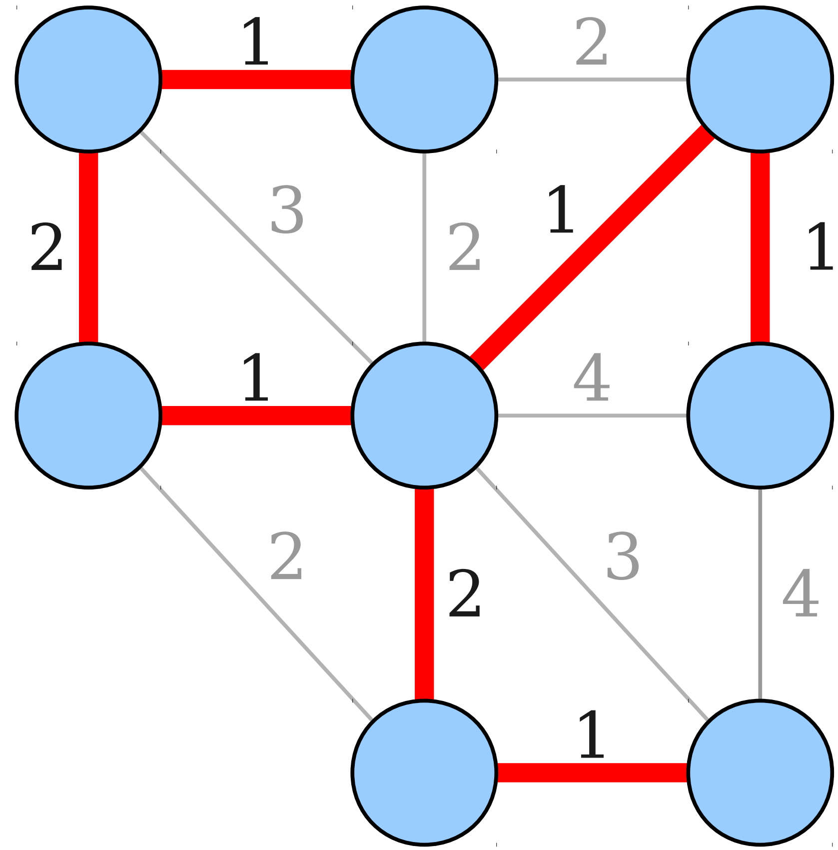

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

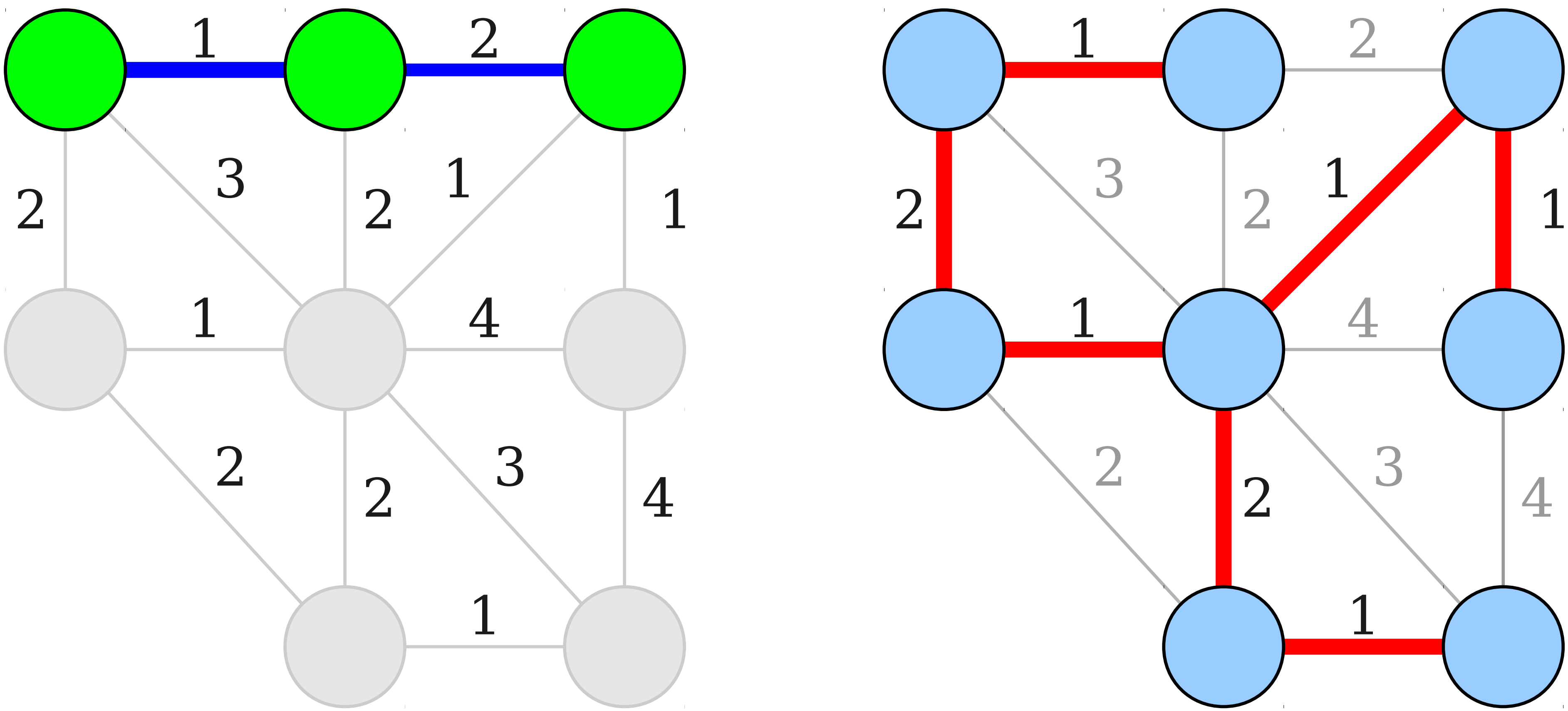

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1



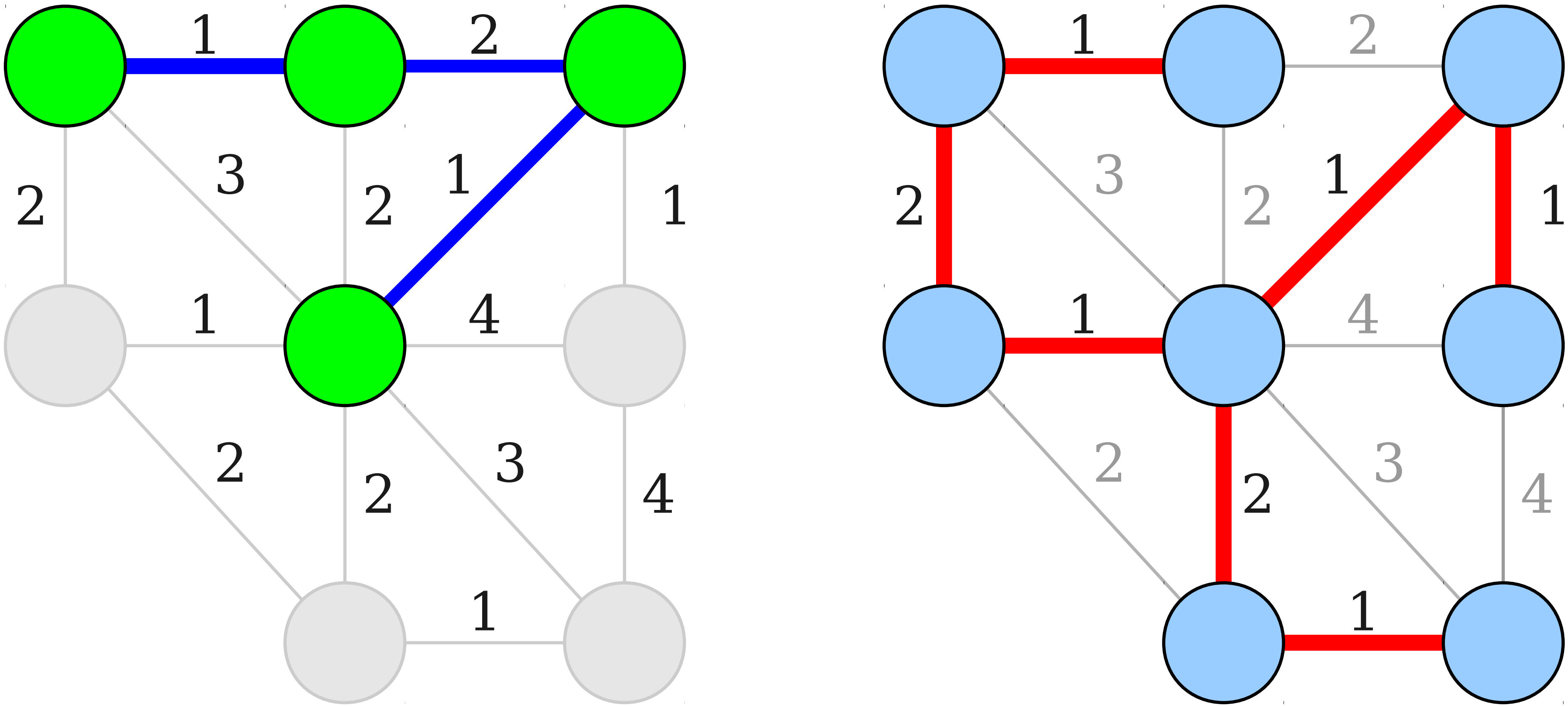
1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

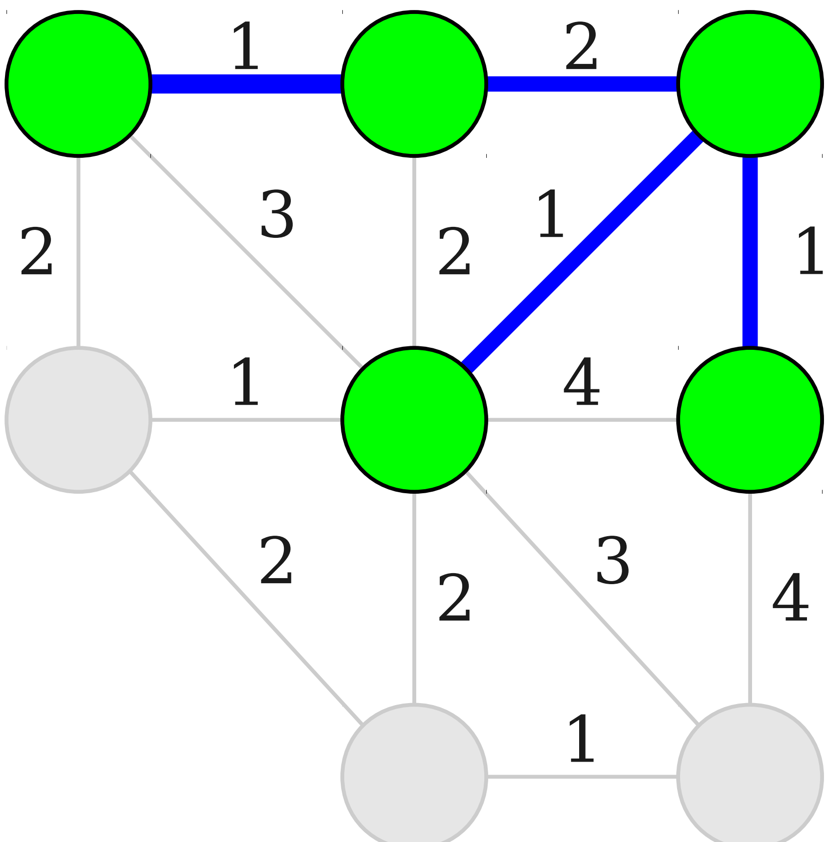

1
2
2
3
2
1
1
1
4
2
2
3
4
1

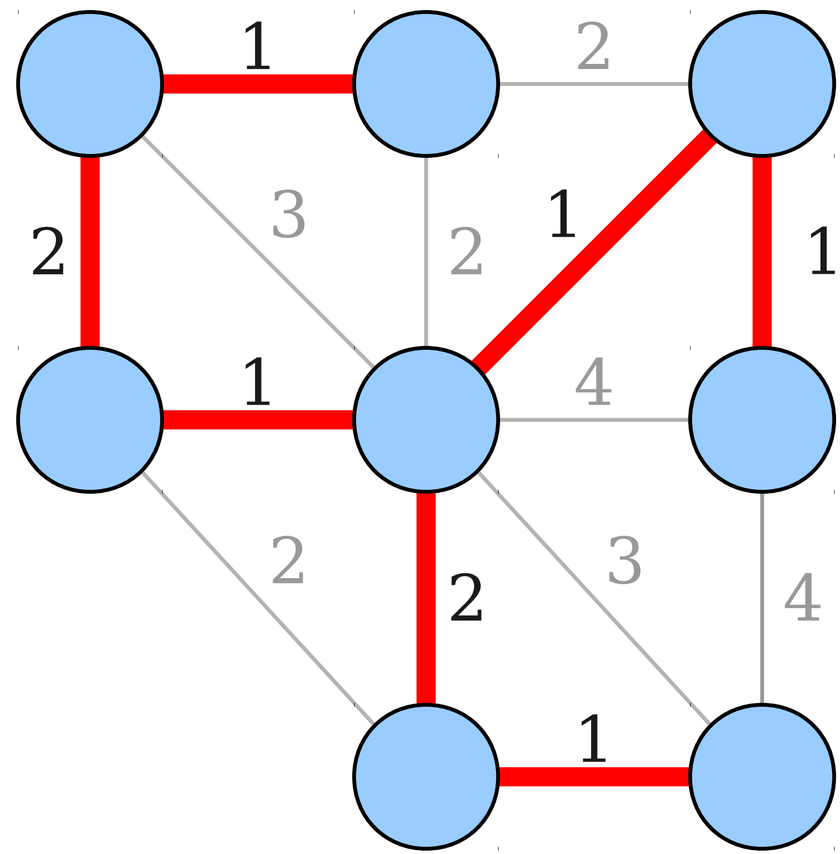

1
2
2
3
2
1
1
1
4
2
2
3
4
1

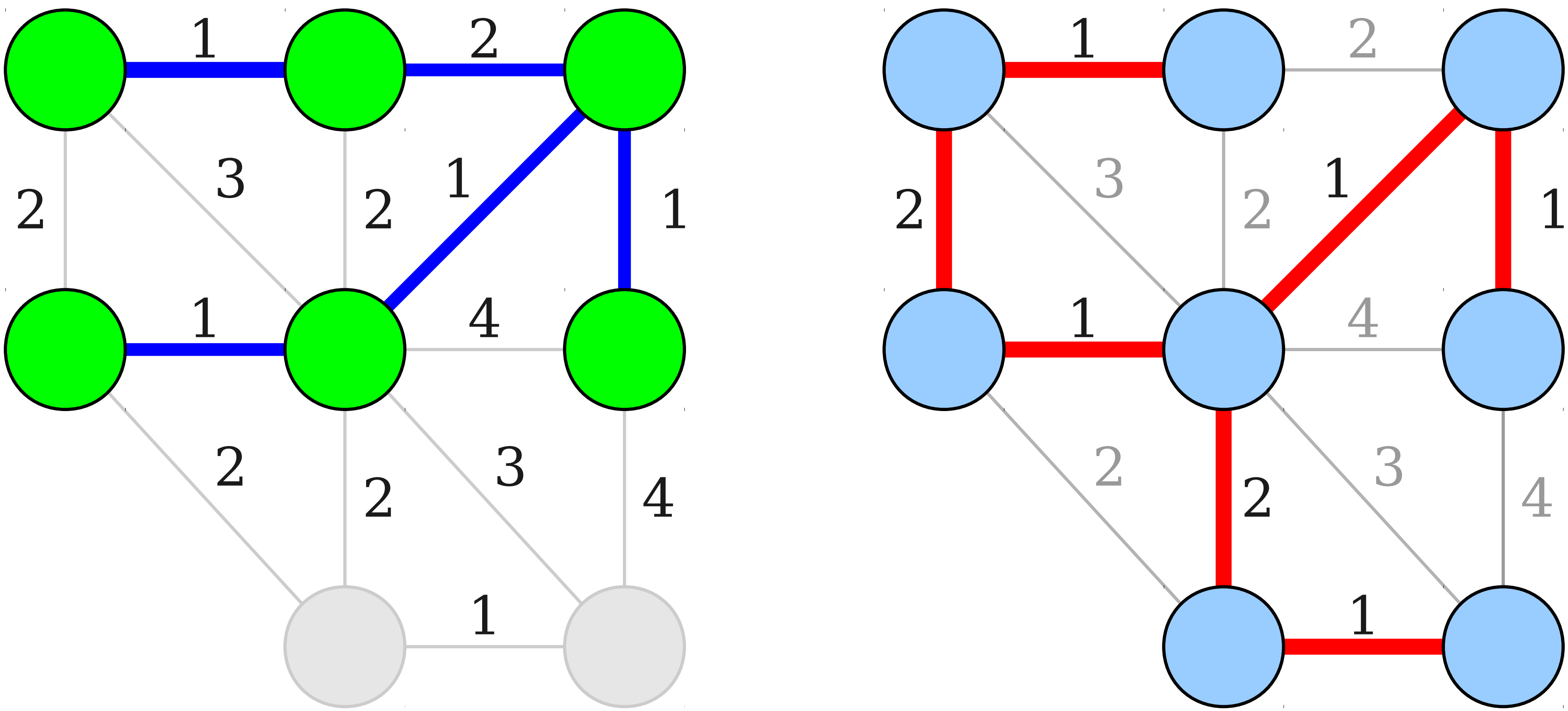


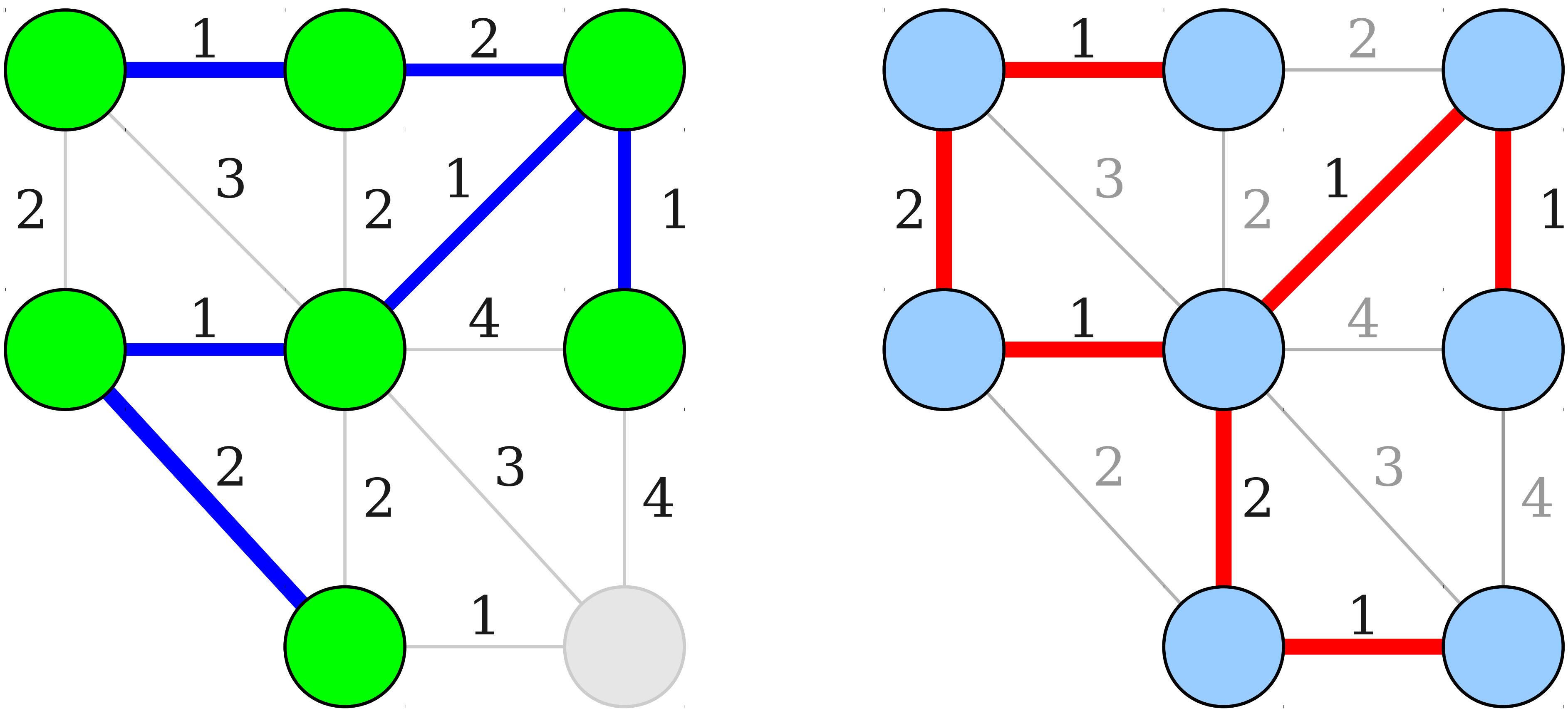
1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

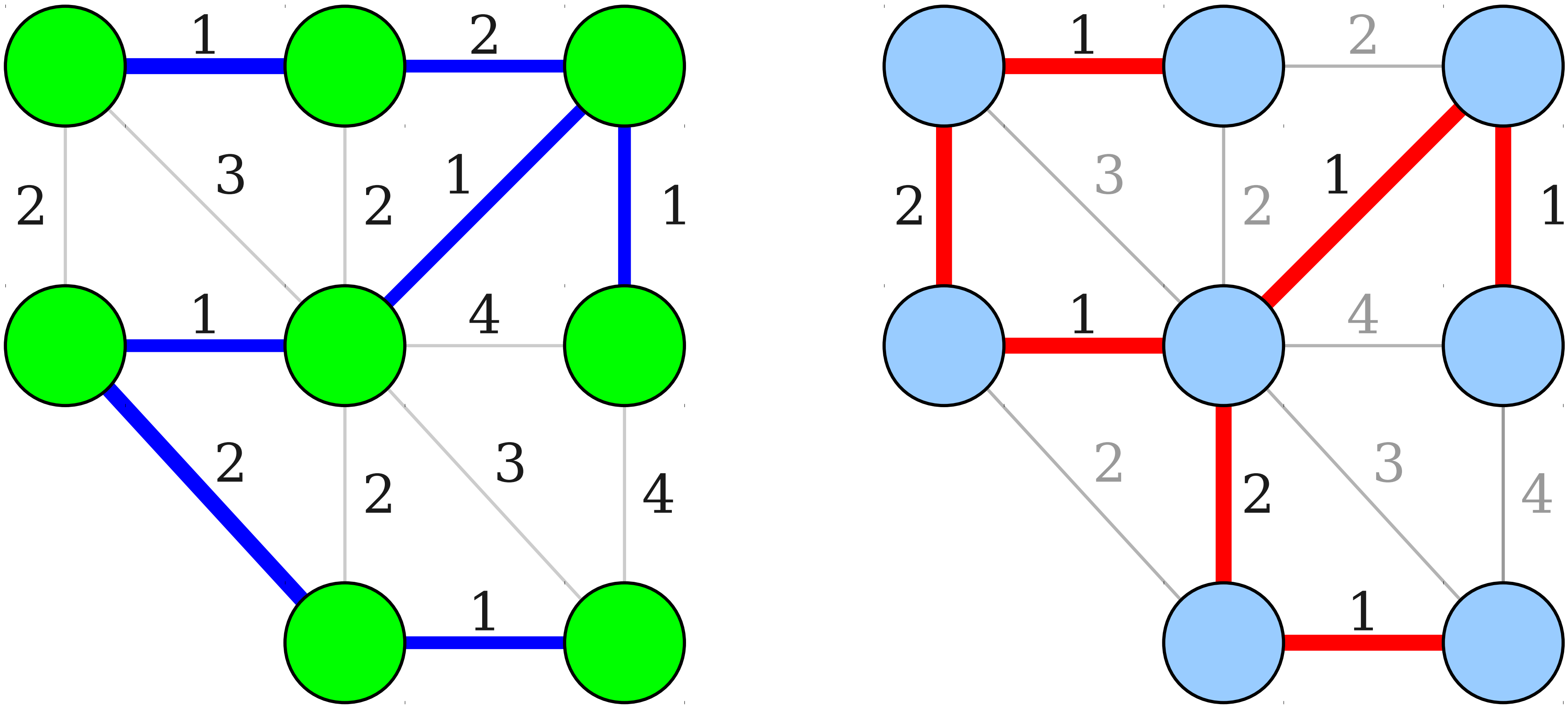

1
2
3
2
1
2
1
1
4
2
2
3
4
1
1
2
3
2
1
2
1
1
4
2
2
3
4
1


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

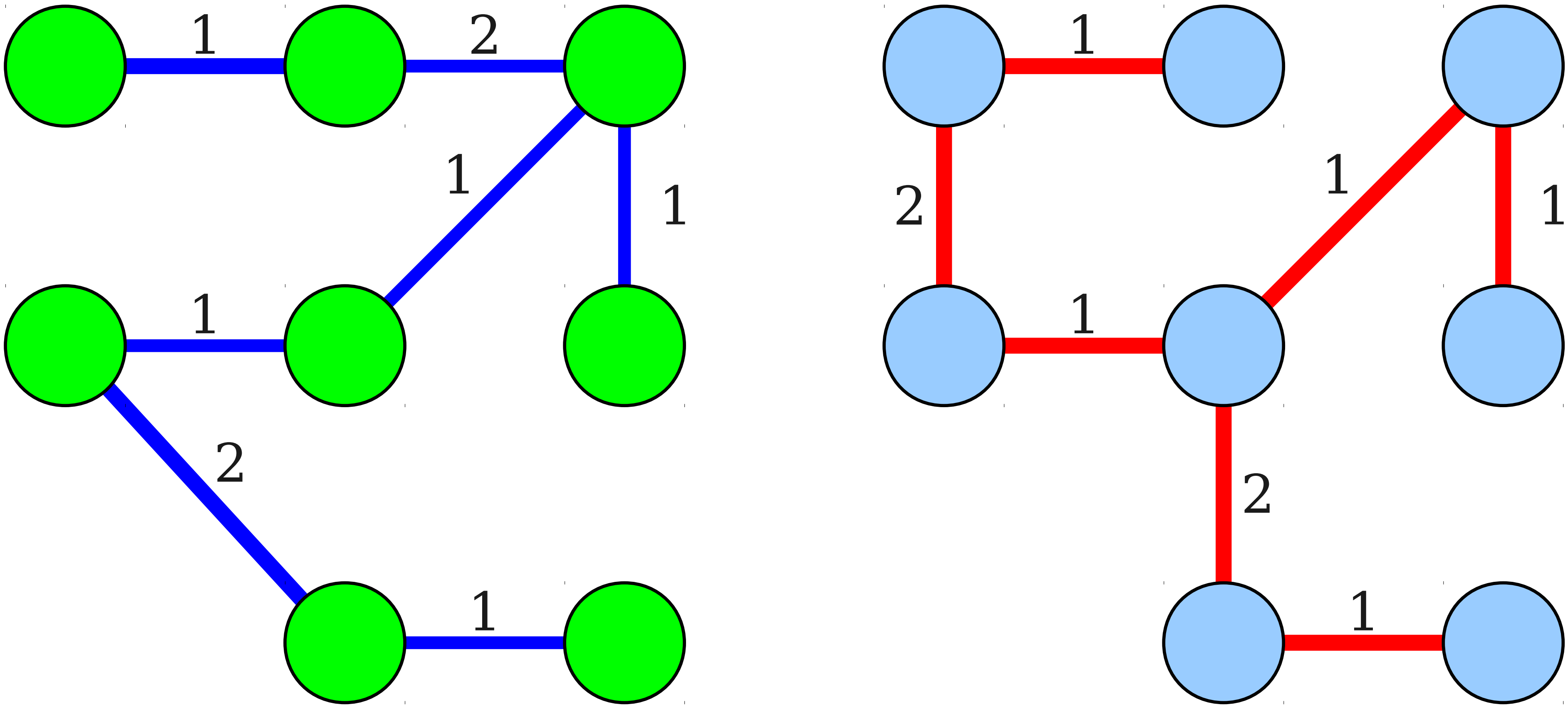

1
2
3
2
2
1
1
1
4
2
3
2
4
1

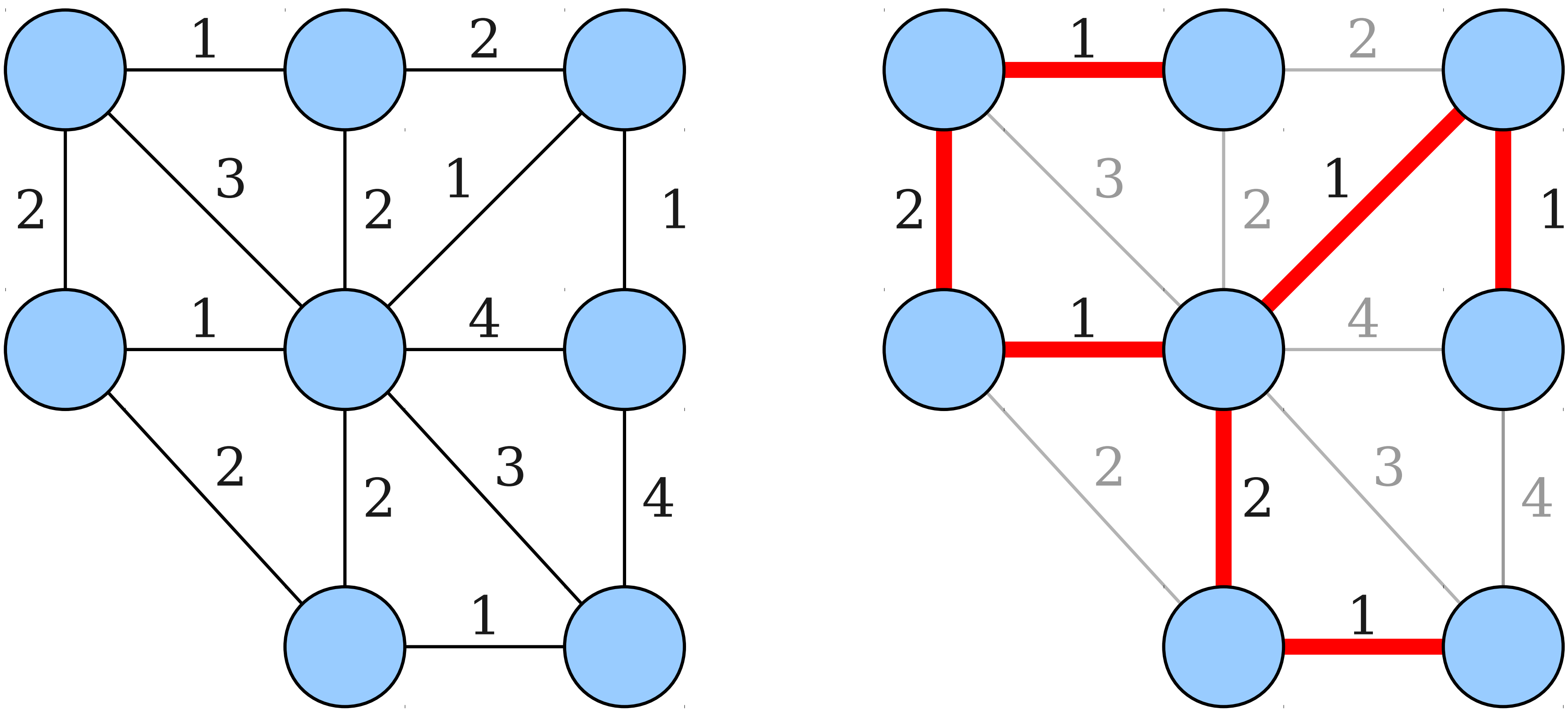

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1


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

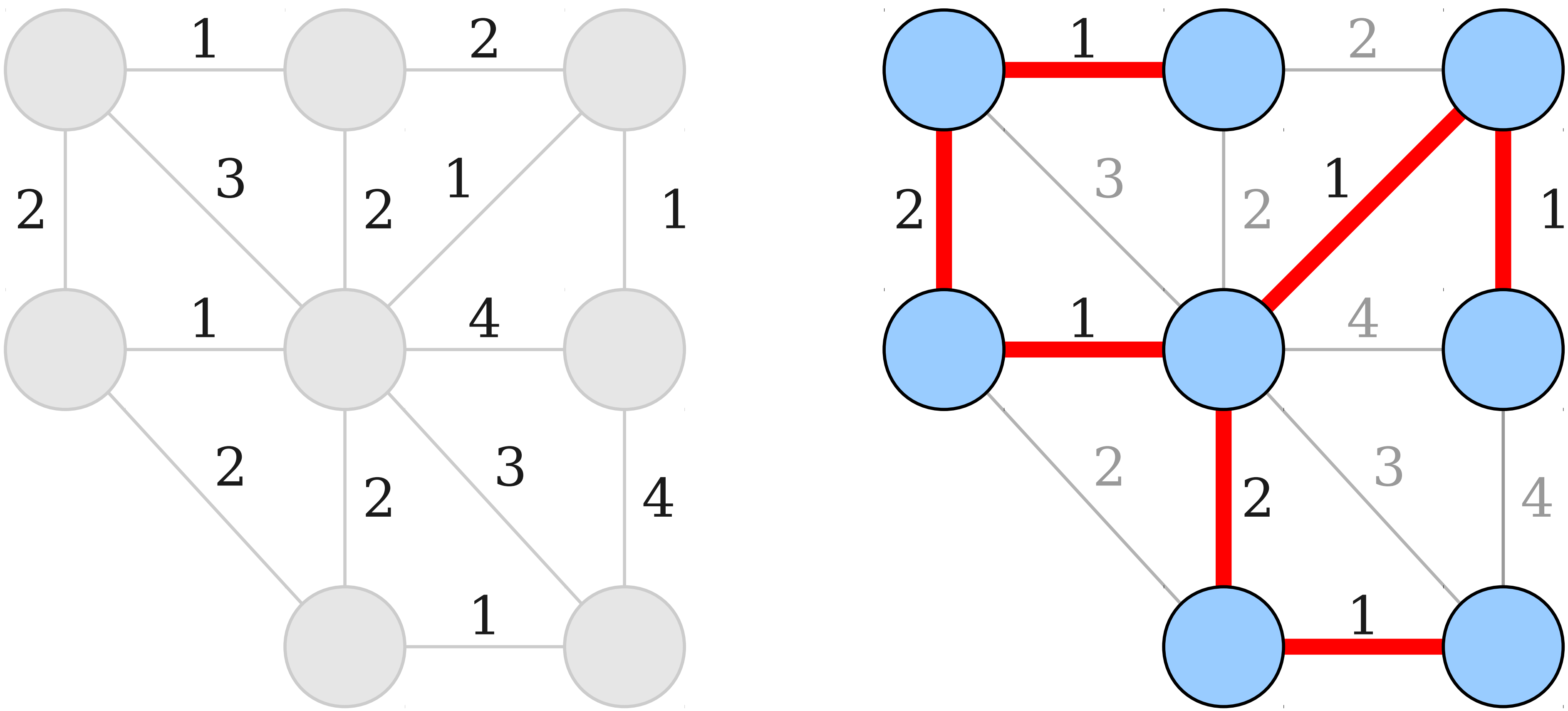


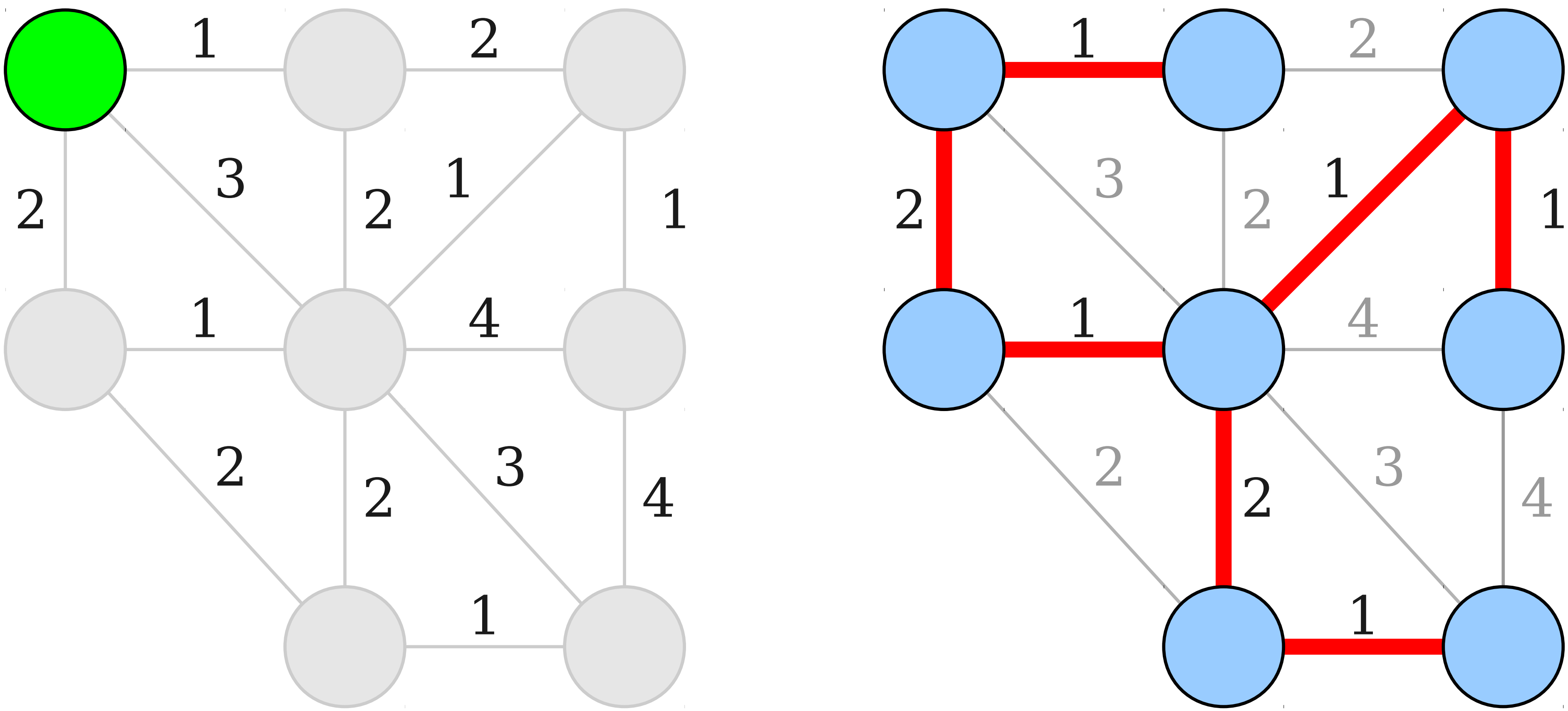
1
2
3
2
1
2
1
1
4
2
3
2
4
1
1
2
3
2
1
2
1
1
4
2
3
2
4
1

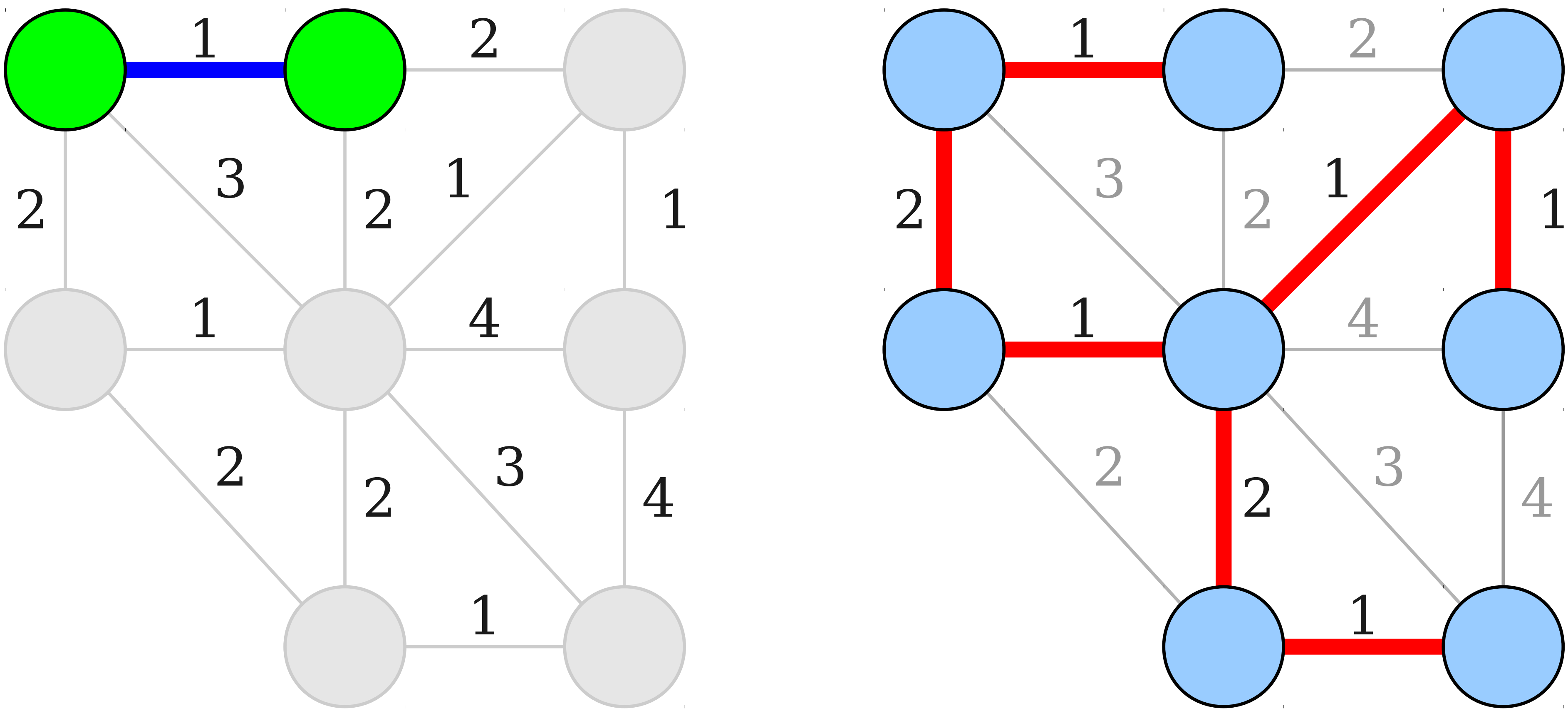

1
2
1
1
1
2
1
1
2
1
1
1
2
1



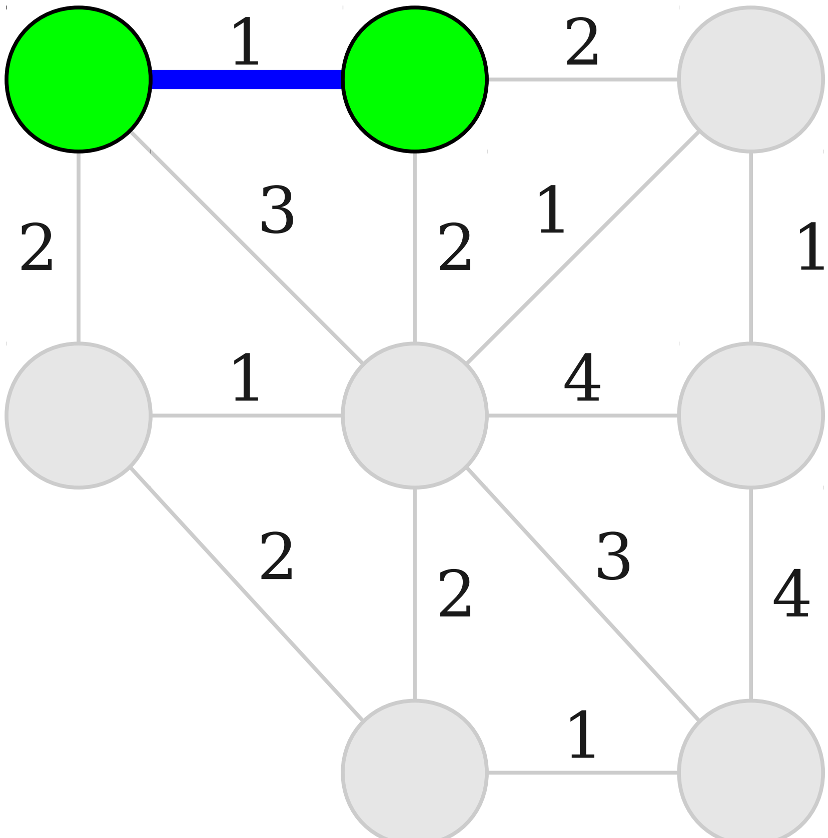
1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

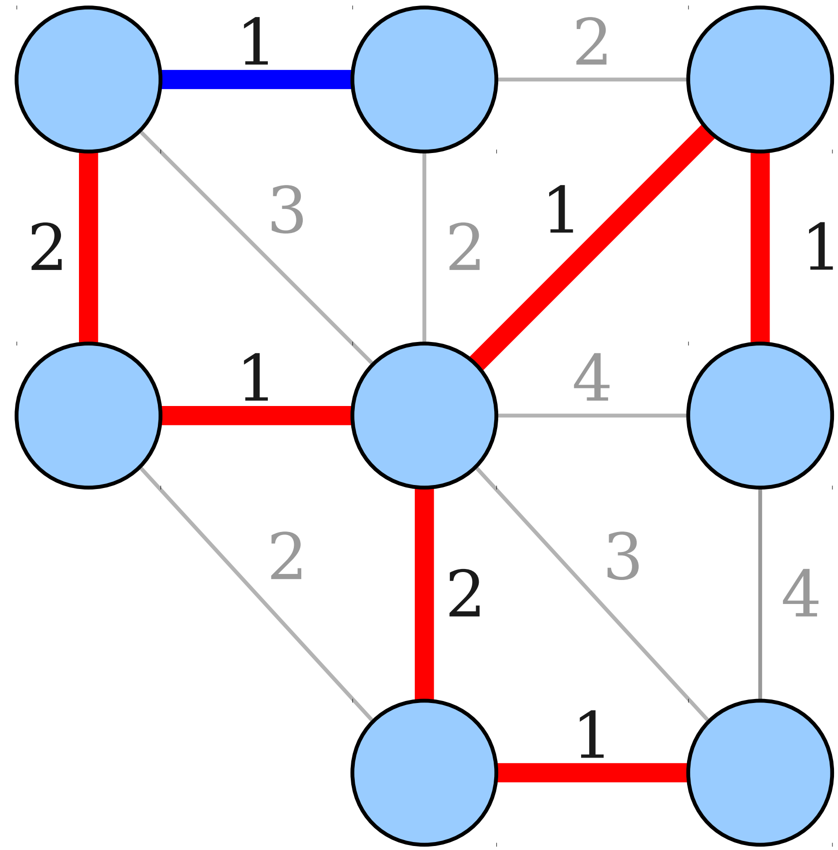

1
2
3
2
1
2
1
1
4
2
2
3
4
1
1
2
3
2
1
2
1
1
4
2
2
3
4
1

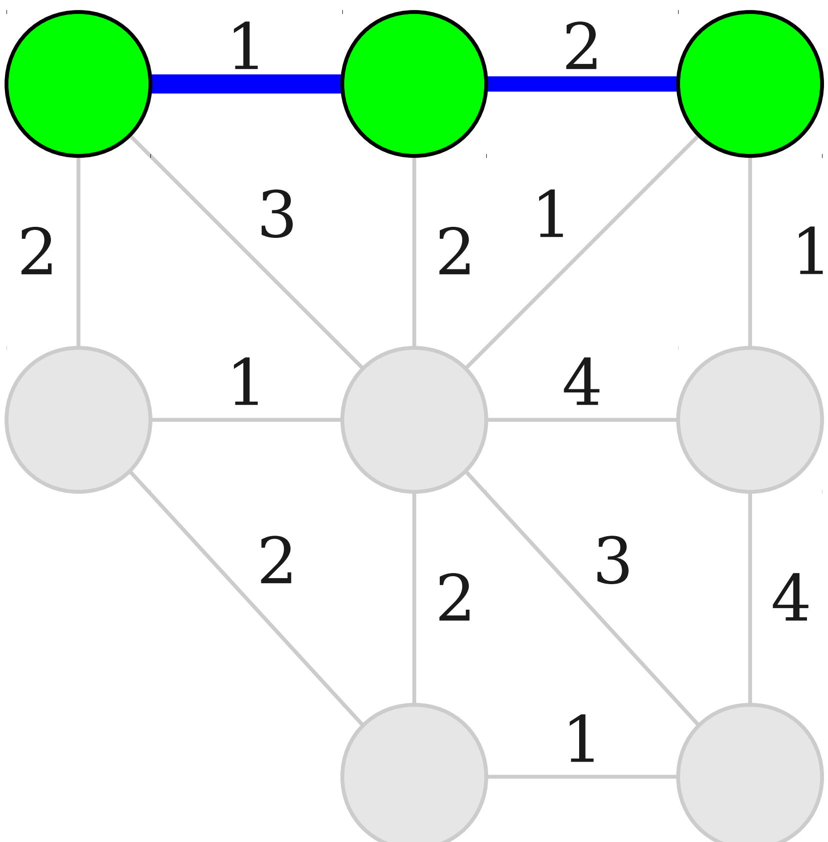
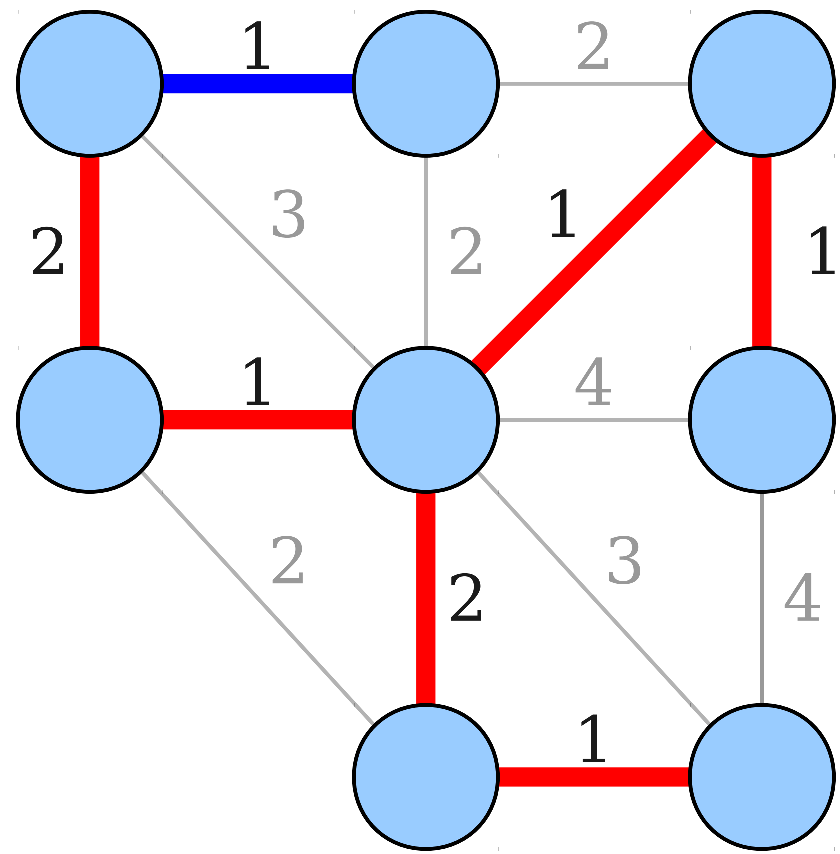

1
2
2
3
2
1
1
1
4
2
3
2
4
1
1
2
2
3
2
1
1
1
4
2
3
2
4
1

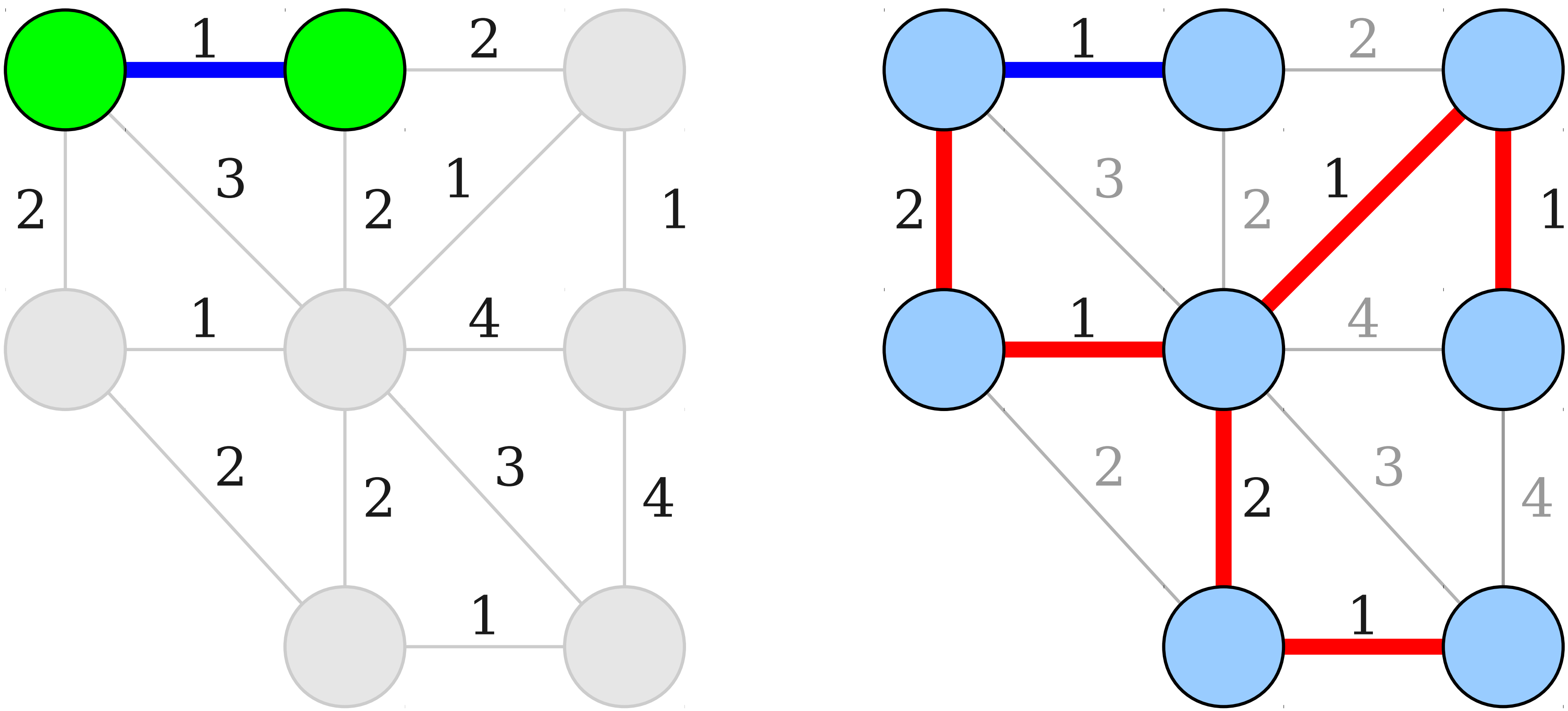


1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

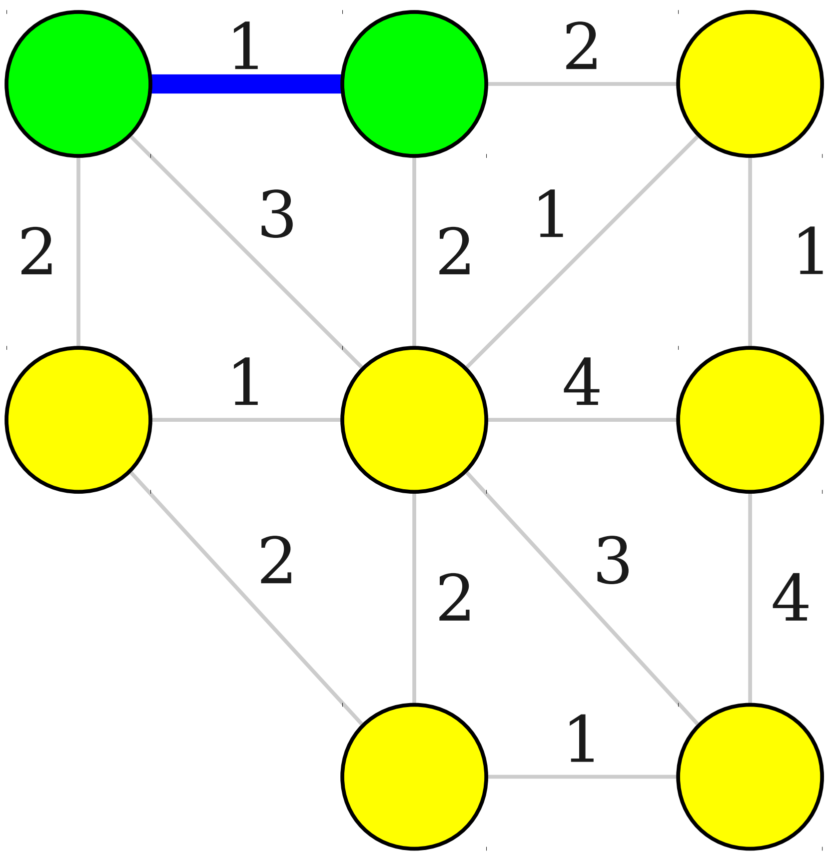

1
2
3
2
1
2
1
1
4
2
2
3
4
1

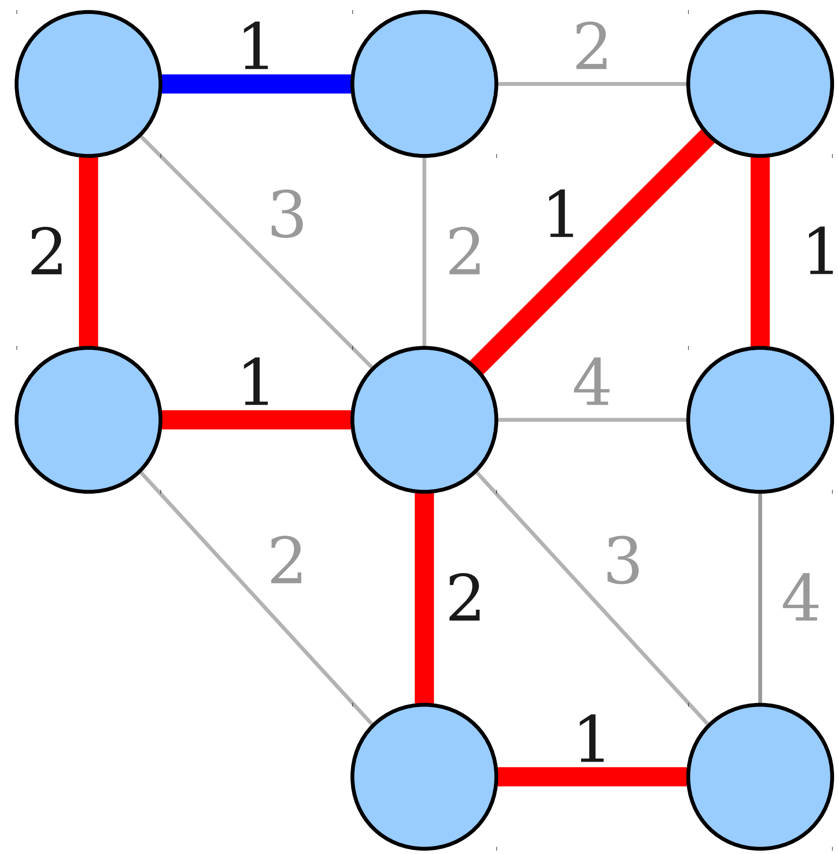

1
2
3
2
1
2
1
1
4
2
2
3
4
1

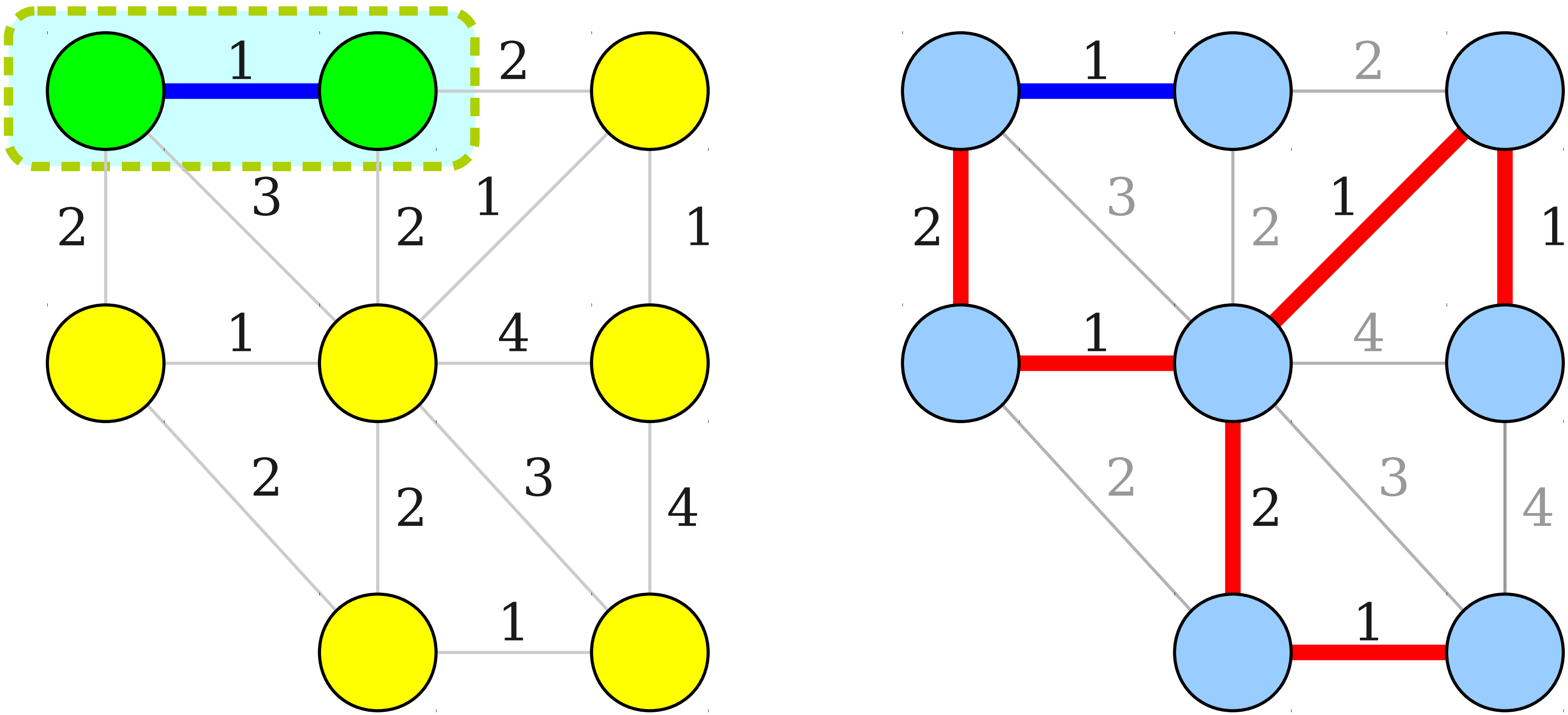

1
2
3
1
2
2
1
1
4
2
3
2
4
1

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

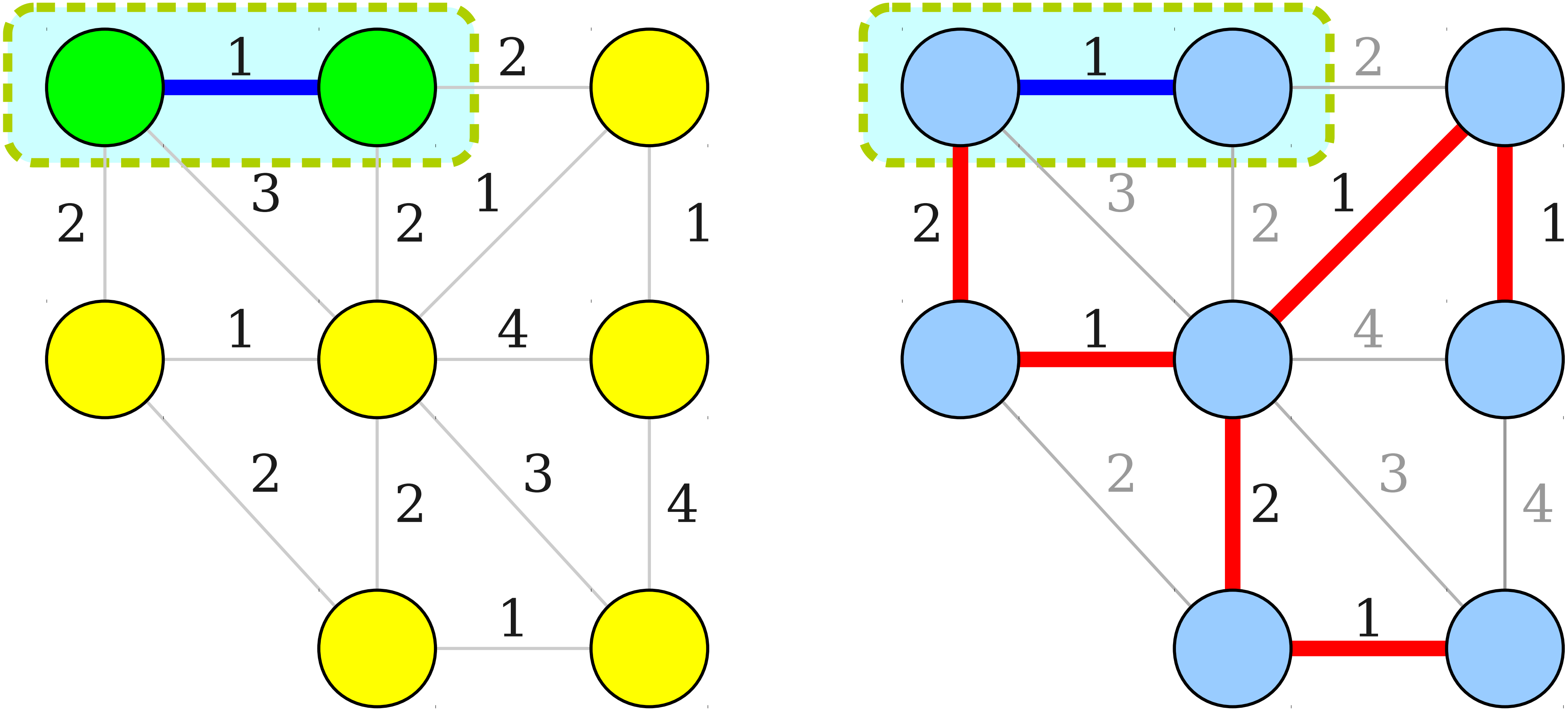


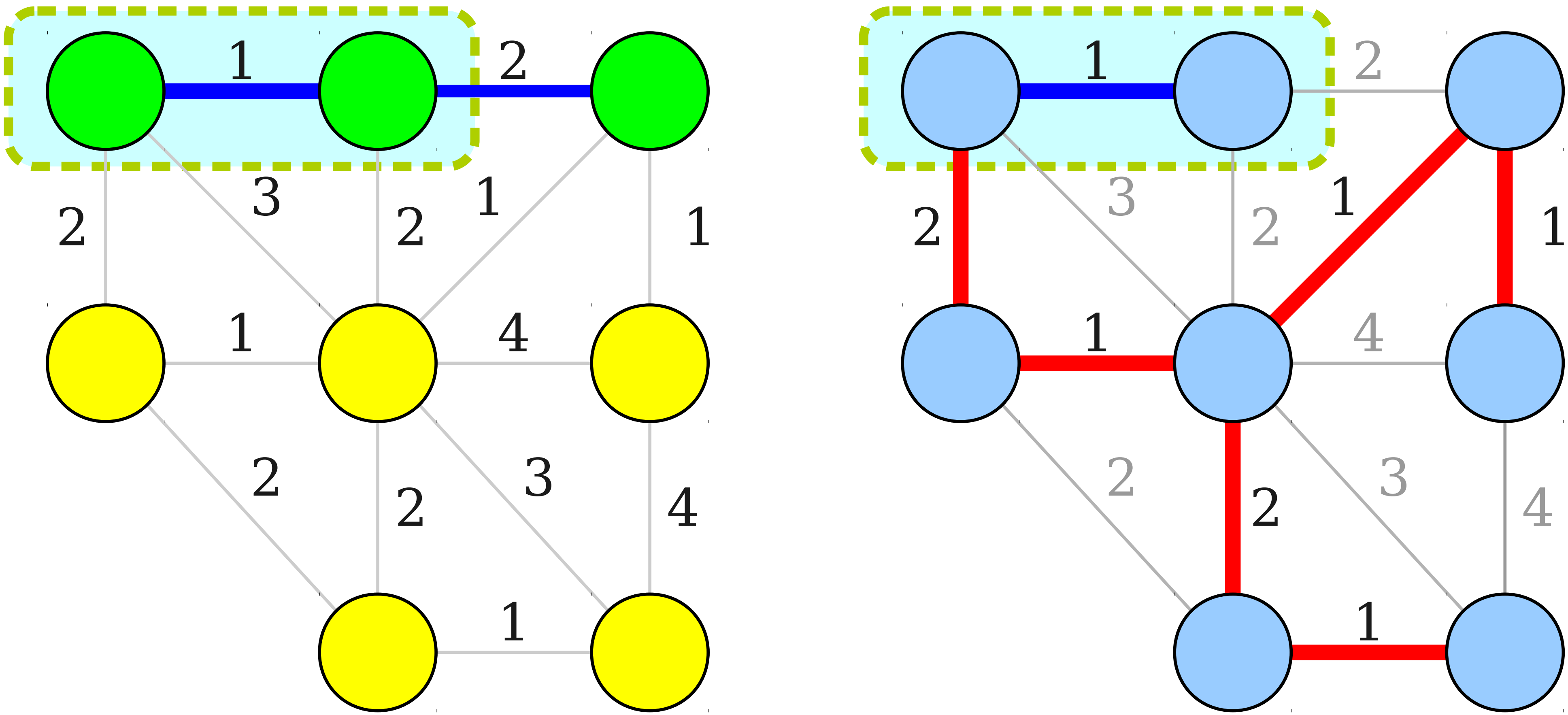
1
2
3
1
2
2
1
1
4
2
3
2
4
1
1
2
3
1
2
2
1
1
4
2
3
2
4
1

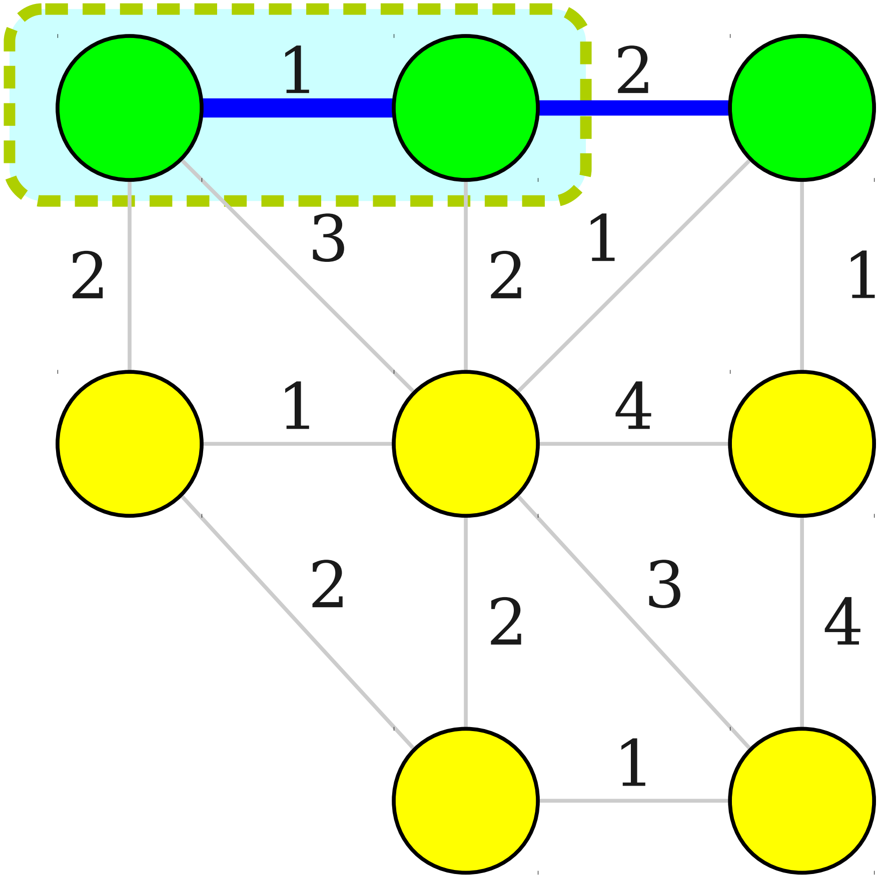

1
2
2
3
2
1
1
1
4
2
2
3
4
1


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

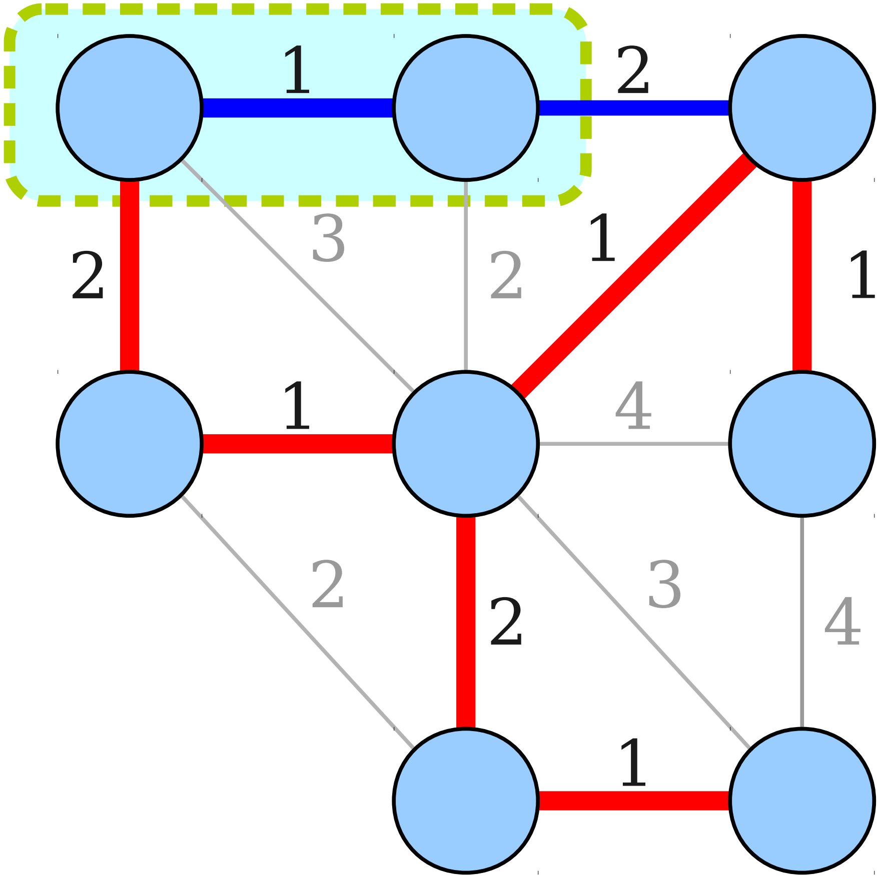

1
2
3
2
1
2
1
1
4
2
2
3
4
1
1
2
3
2
1
2
1
1
4
2
2
3
4
1


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

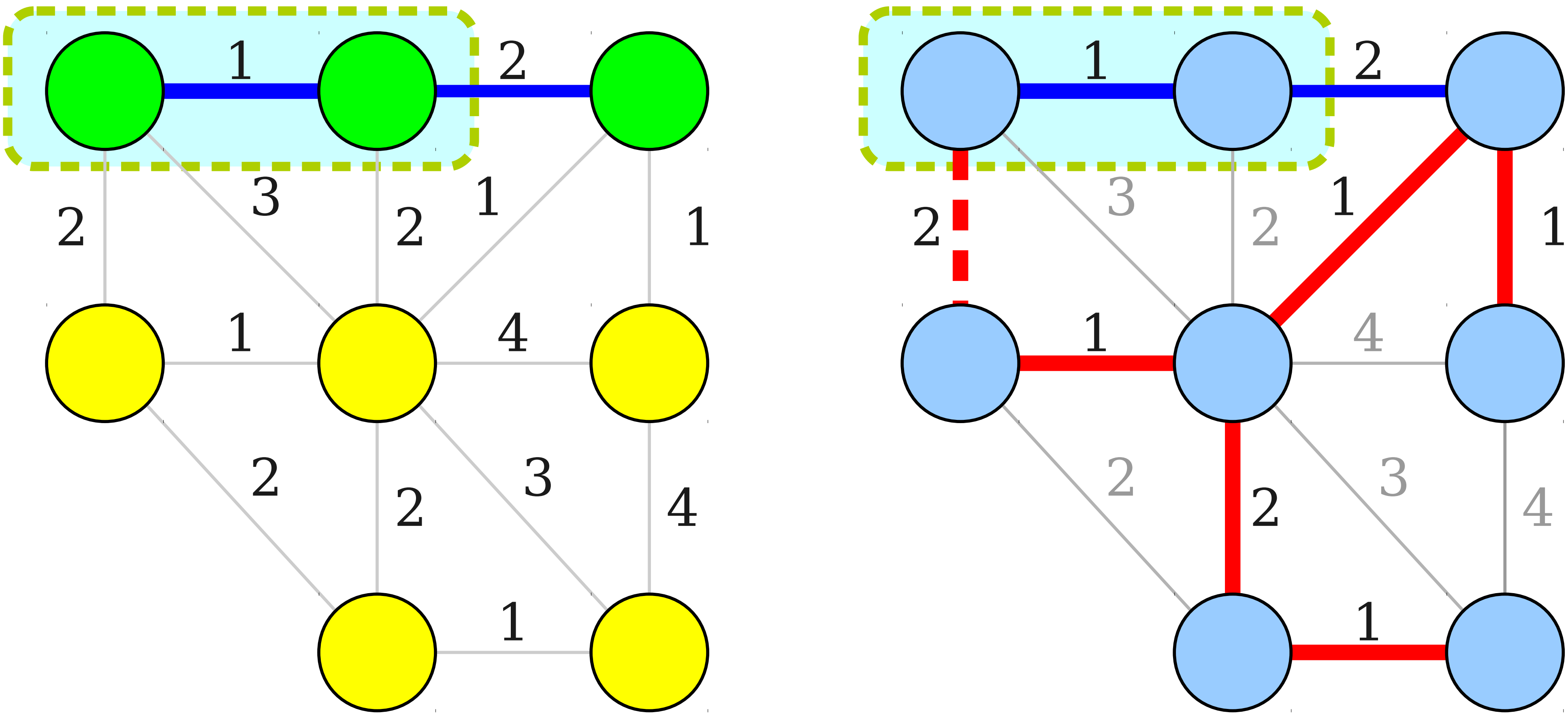

1
2
3
2
1
1
2
1
4
2
2
3
4
1
1
2
3
2
1
1
2
1
4
2
2
3
4
1



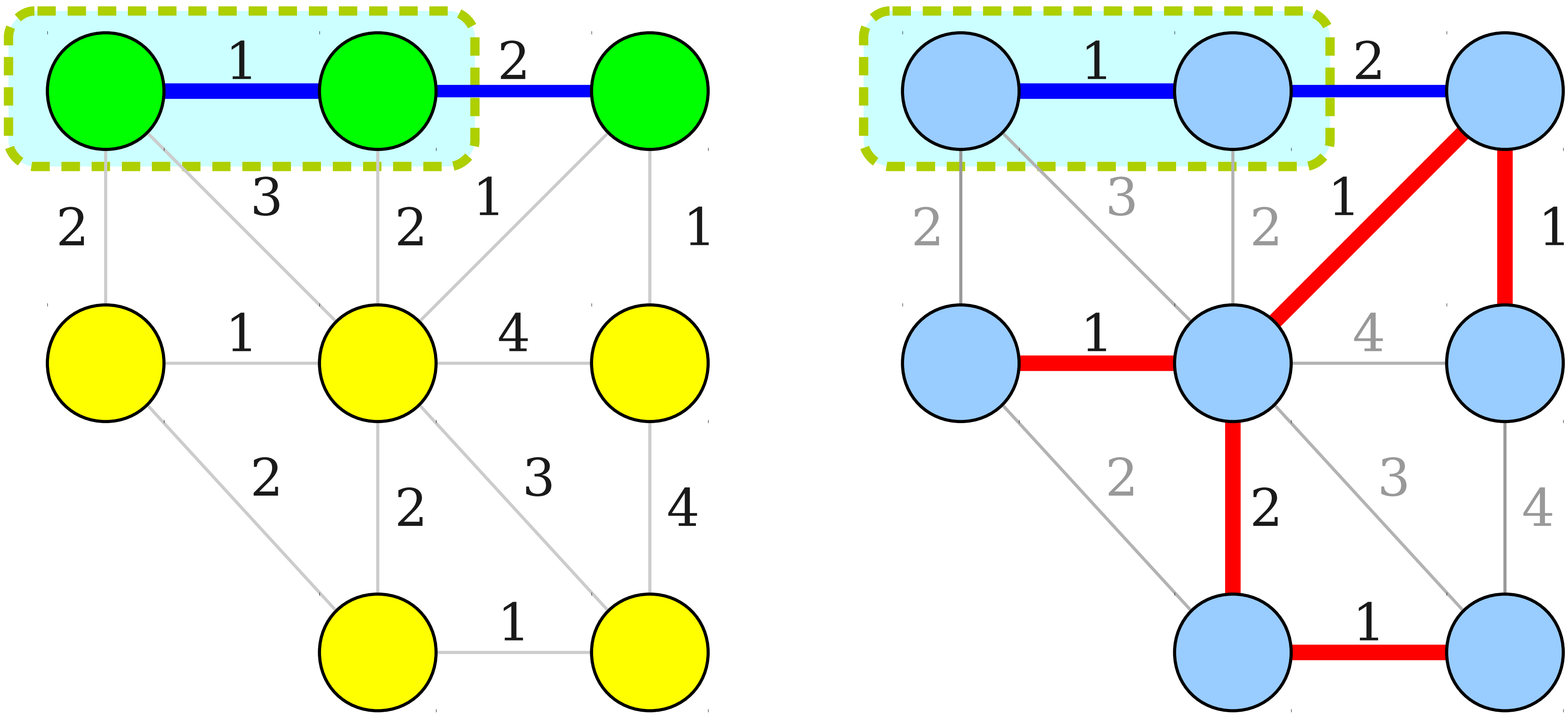
1
2
3
2
1
2
1
1
4
2
2
3
4
1

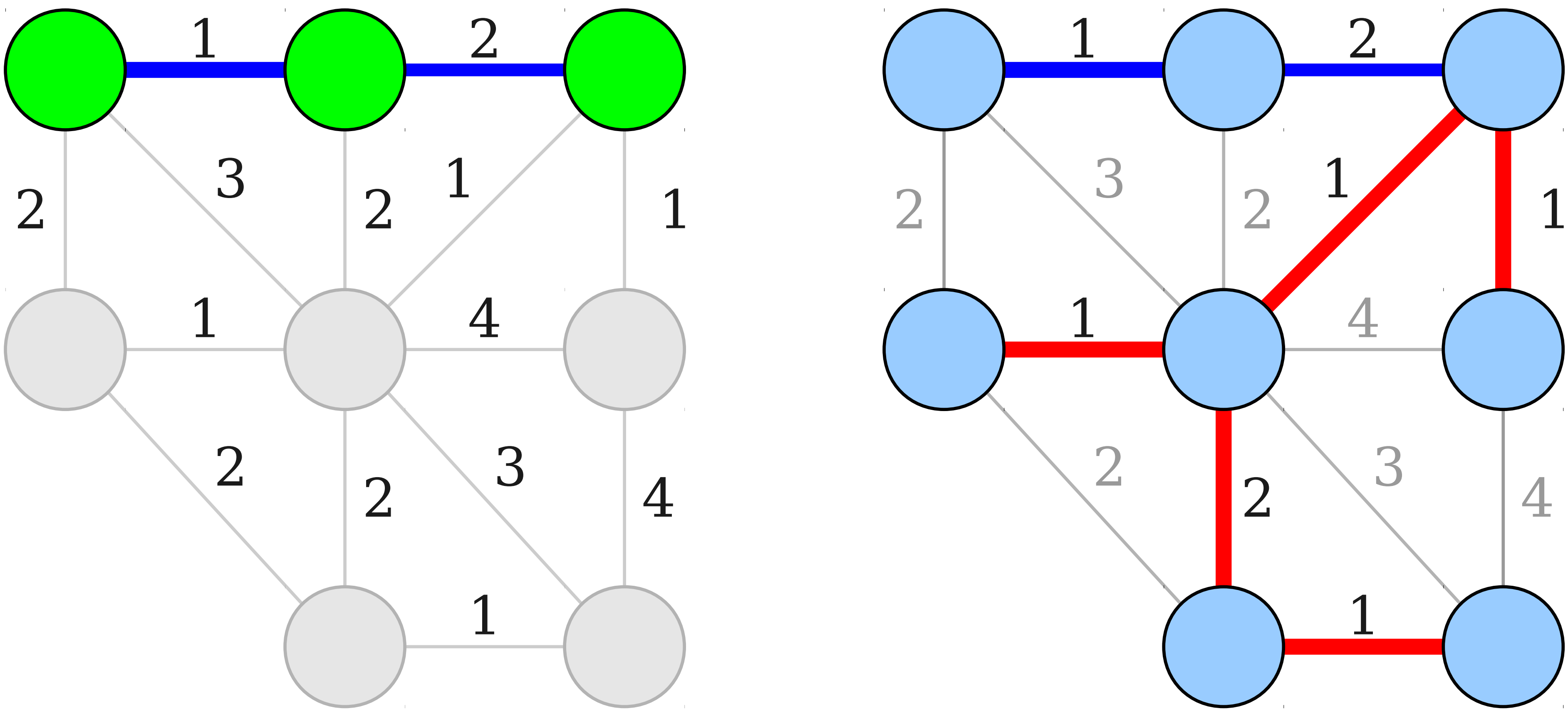


1
2
2
3
2
1
1
1
4
2
2
3
4
1

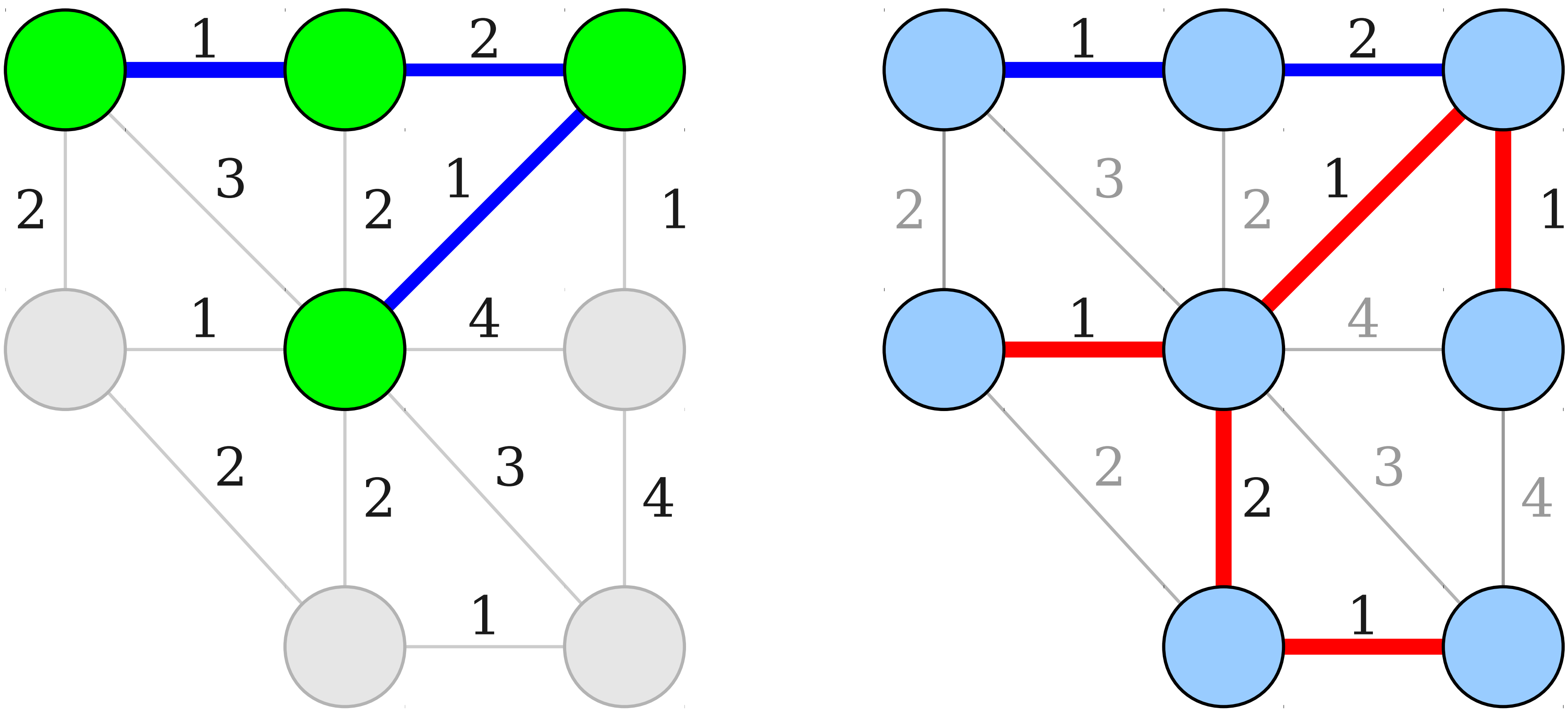


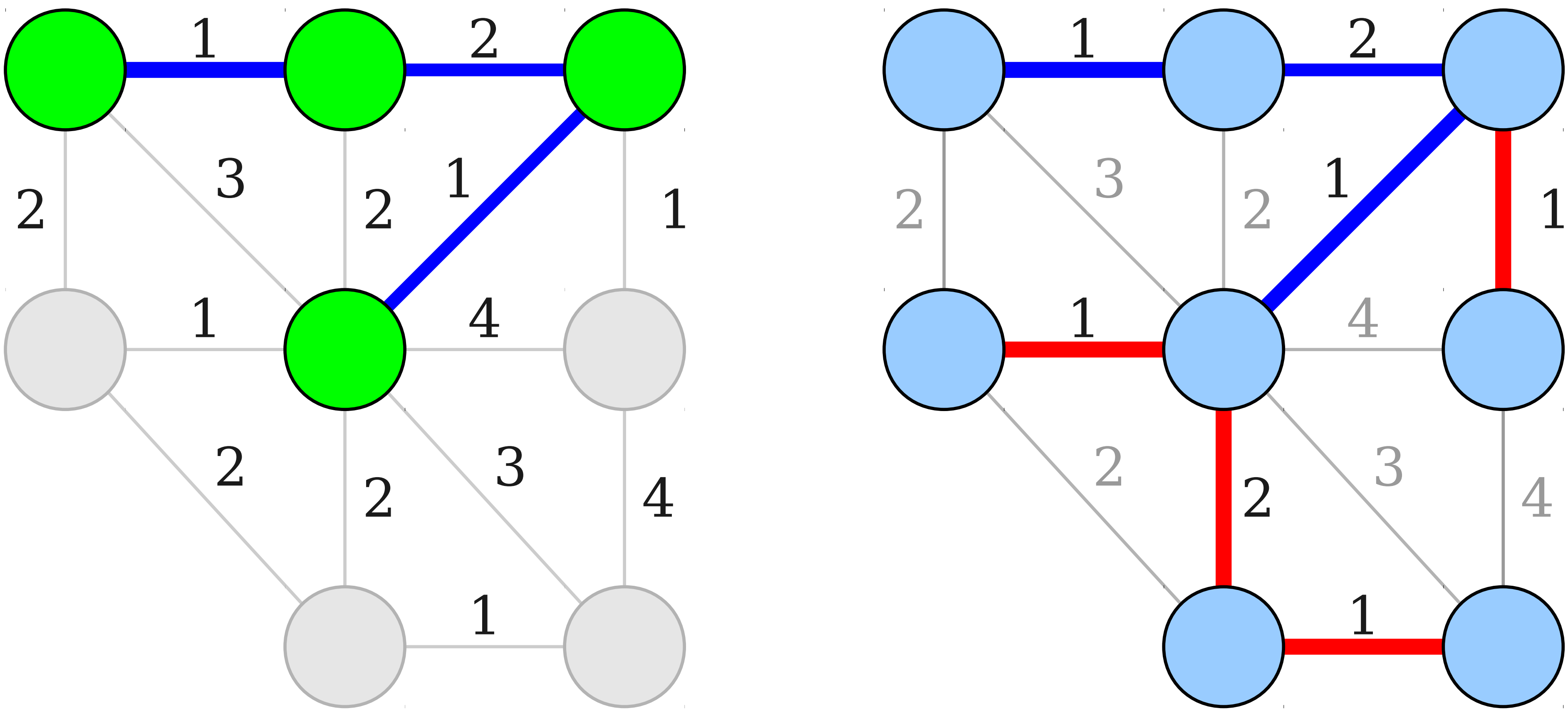
1
2
3
2
1
2
1
1
4
2
3
2
4
1
1
2
3
2
1
2
1
1
4
2
3
2
4
1


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

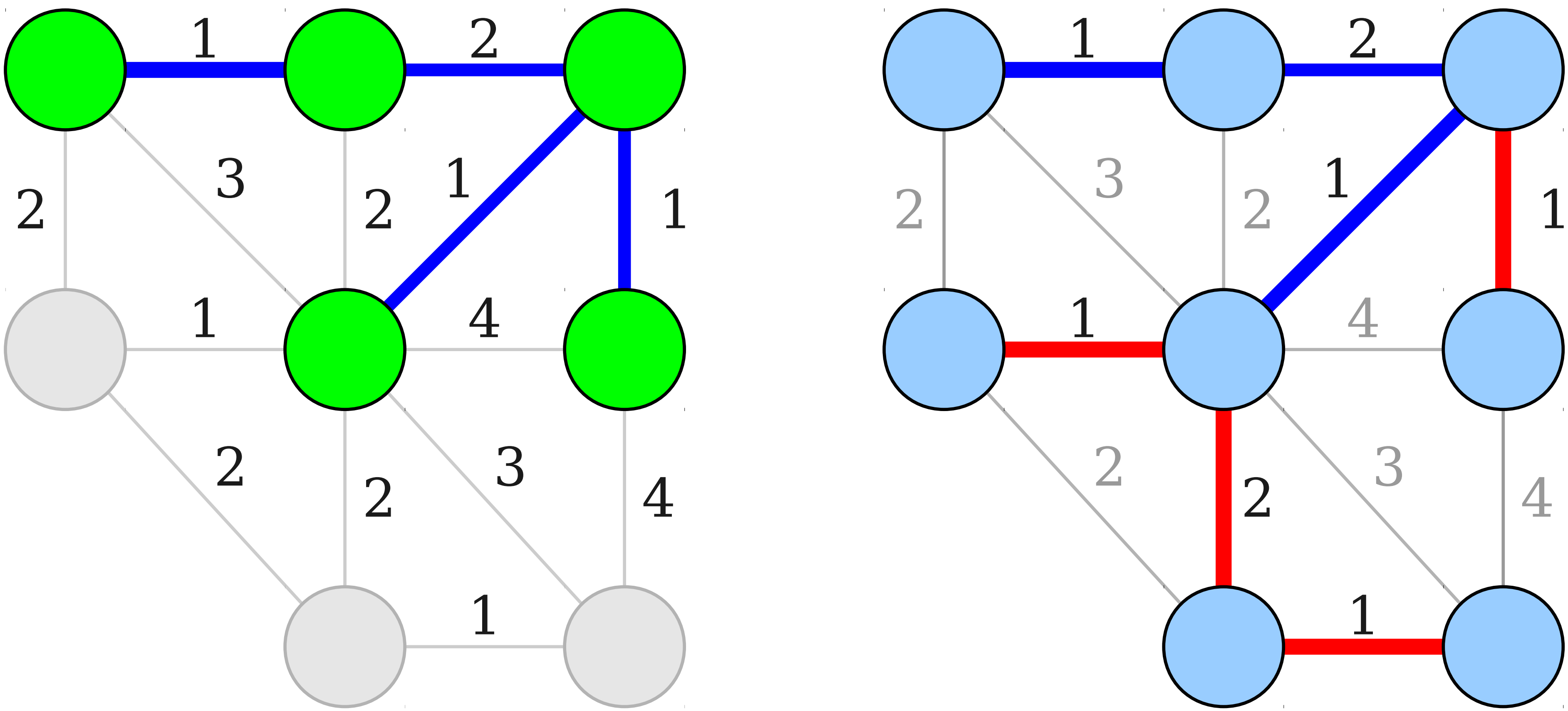


1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

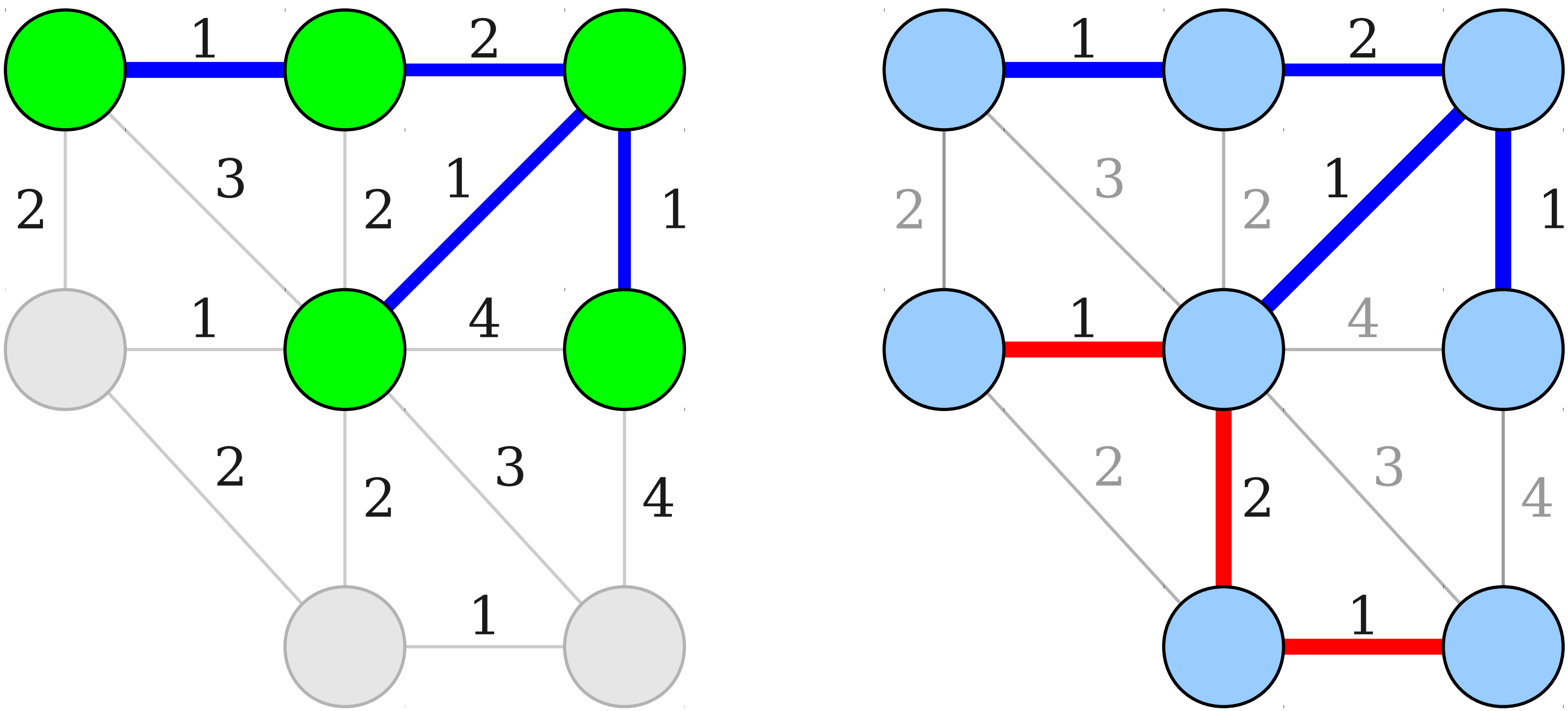

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

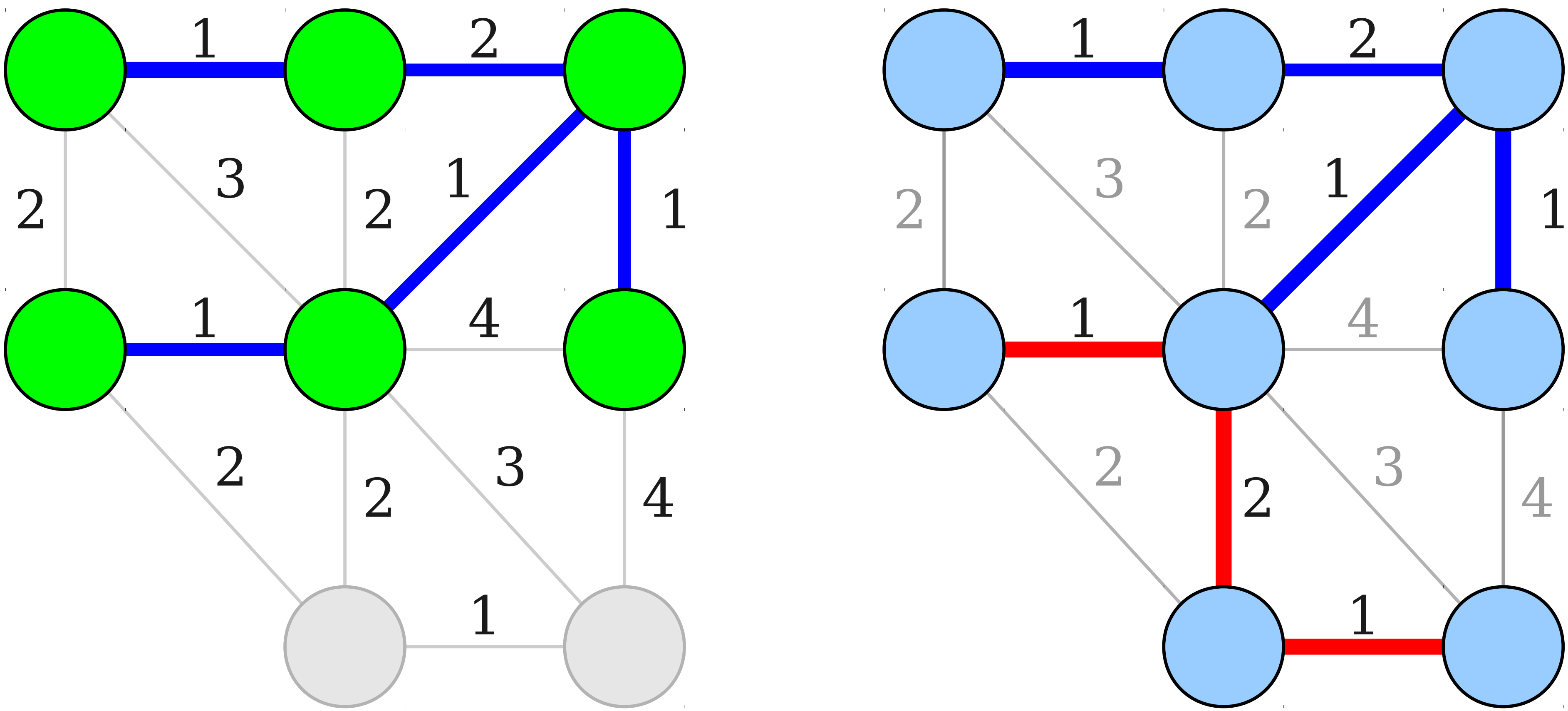


1
2
3
2
1
2
1
1
4
2
2
3
4
1
1
2
3
2
1
2
1
1
4
2
2
3
4
1

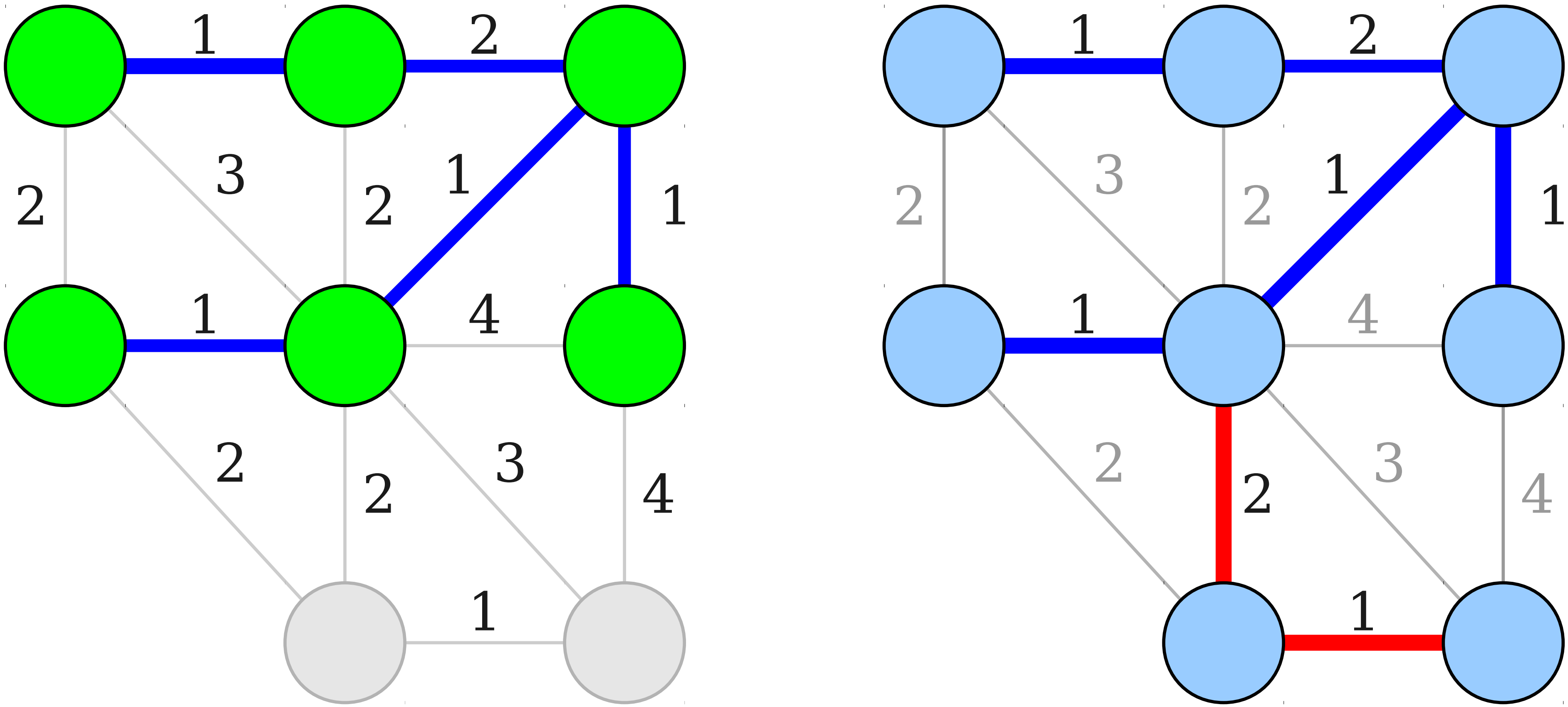


1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

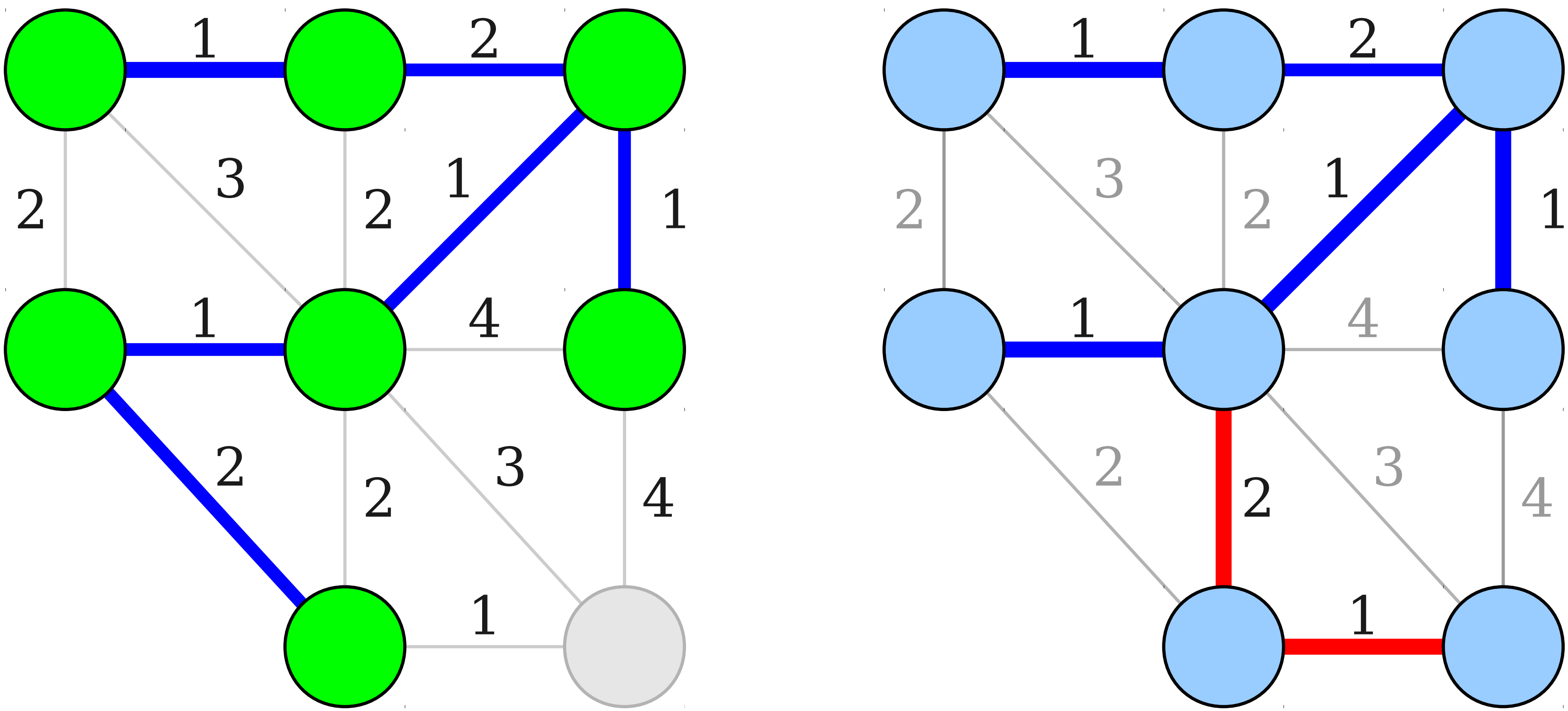

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

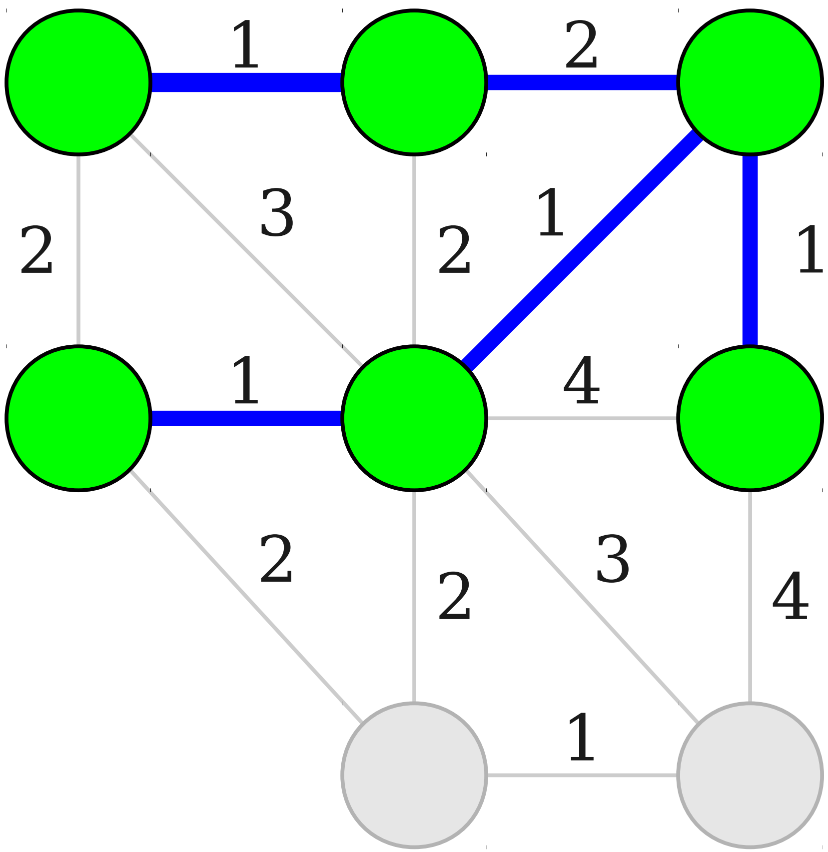


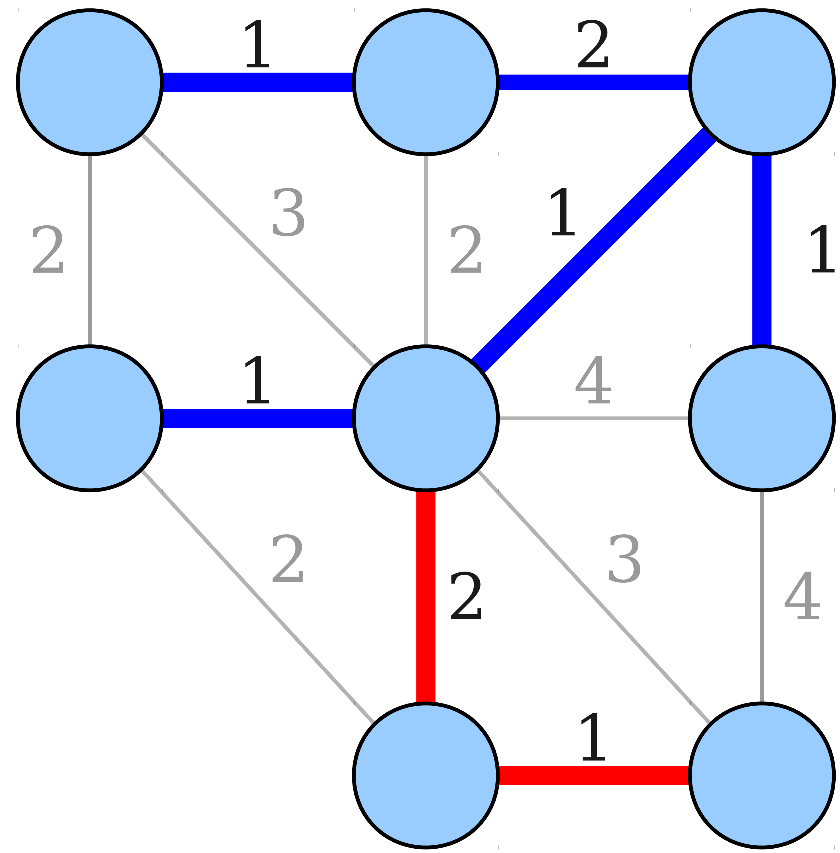
1
2
3
2
1
2
1
1
4
2
2
3
4
1
1
2
3
2
1
2
1
1
4
2
2
3
4
1

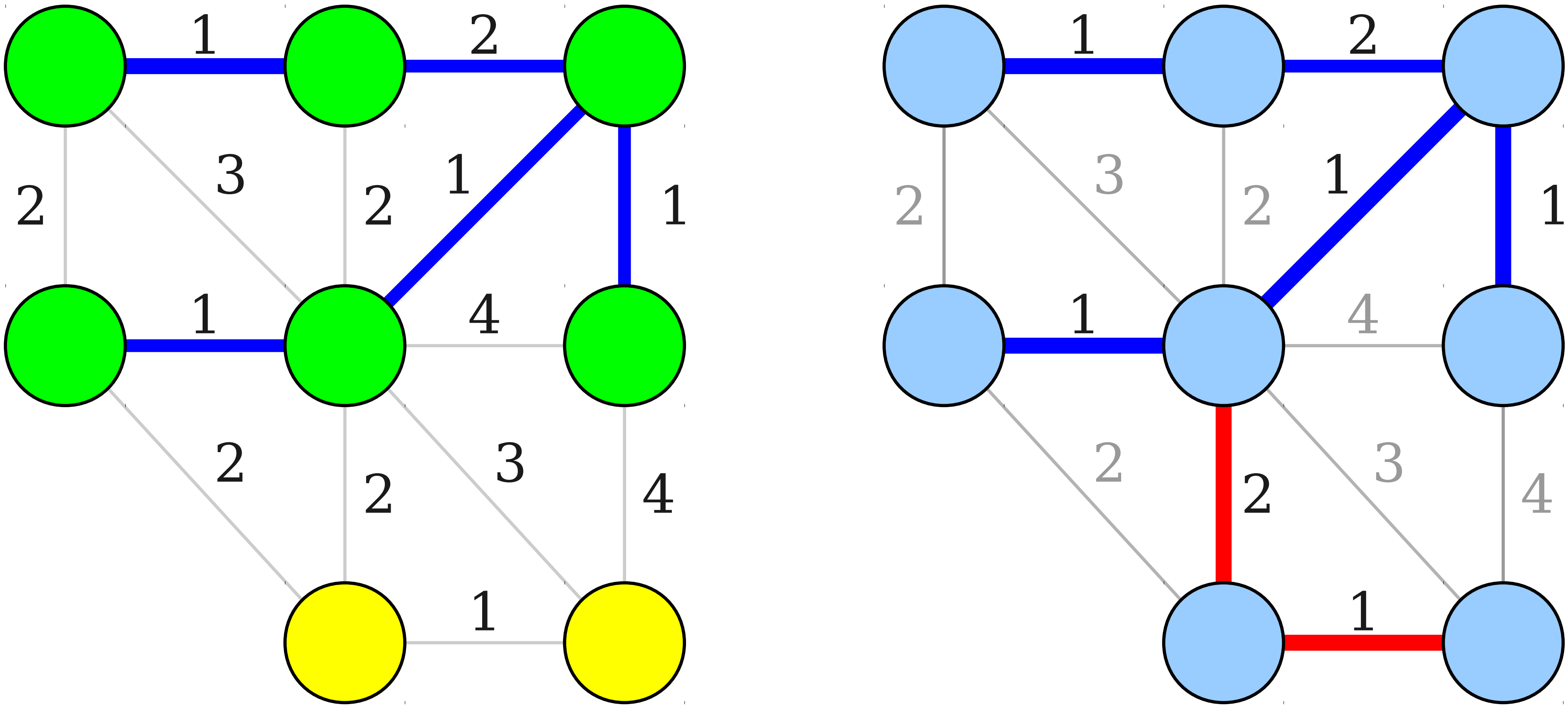


1
2
3
2
2
1
1
1
4
2
2
3
4
1
1
2
3
2
2
1
1
1
4
2
2
3
4
1

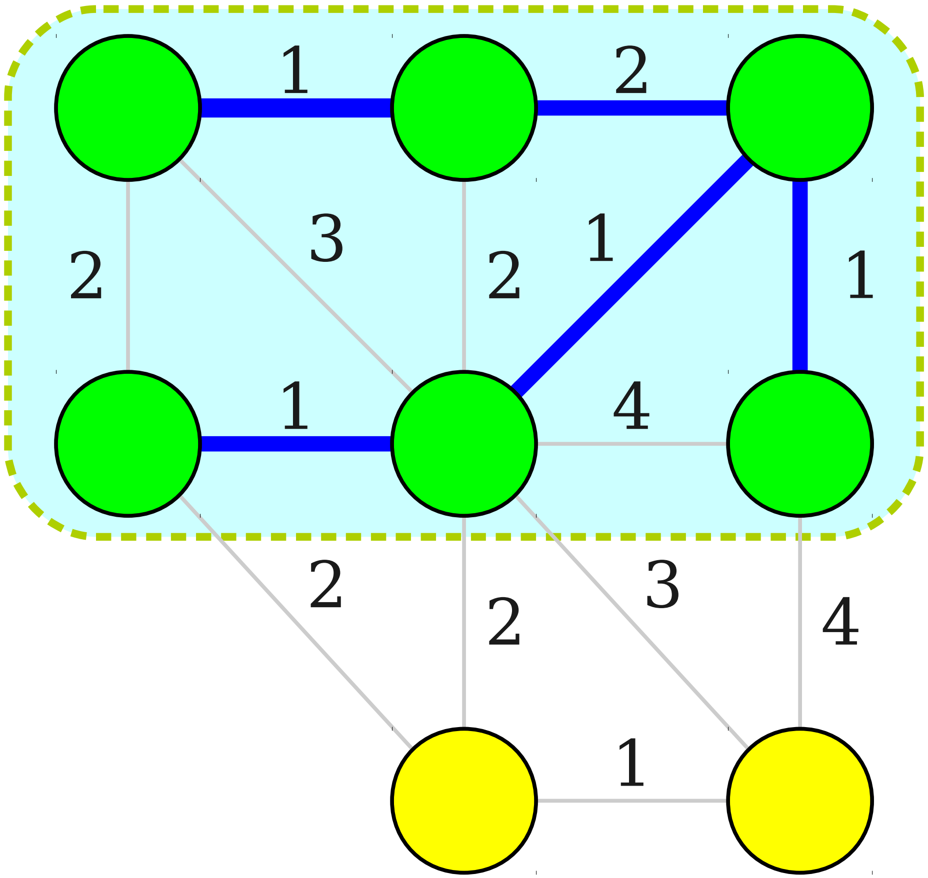

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
1
2
3
2
1
1
1
4
2
2
3
4
1

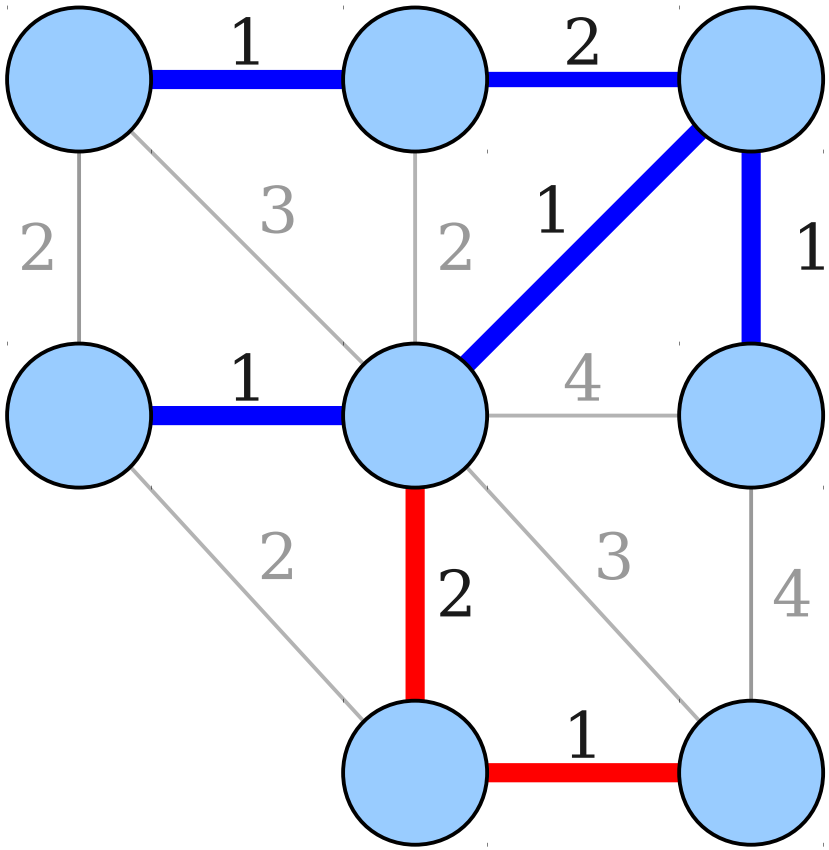

1
2
3
1
2
2
1
1
4
2
3
2
4
1

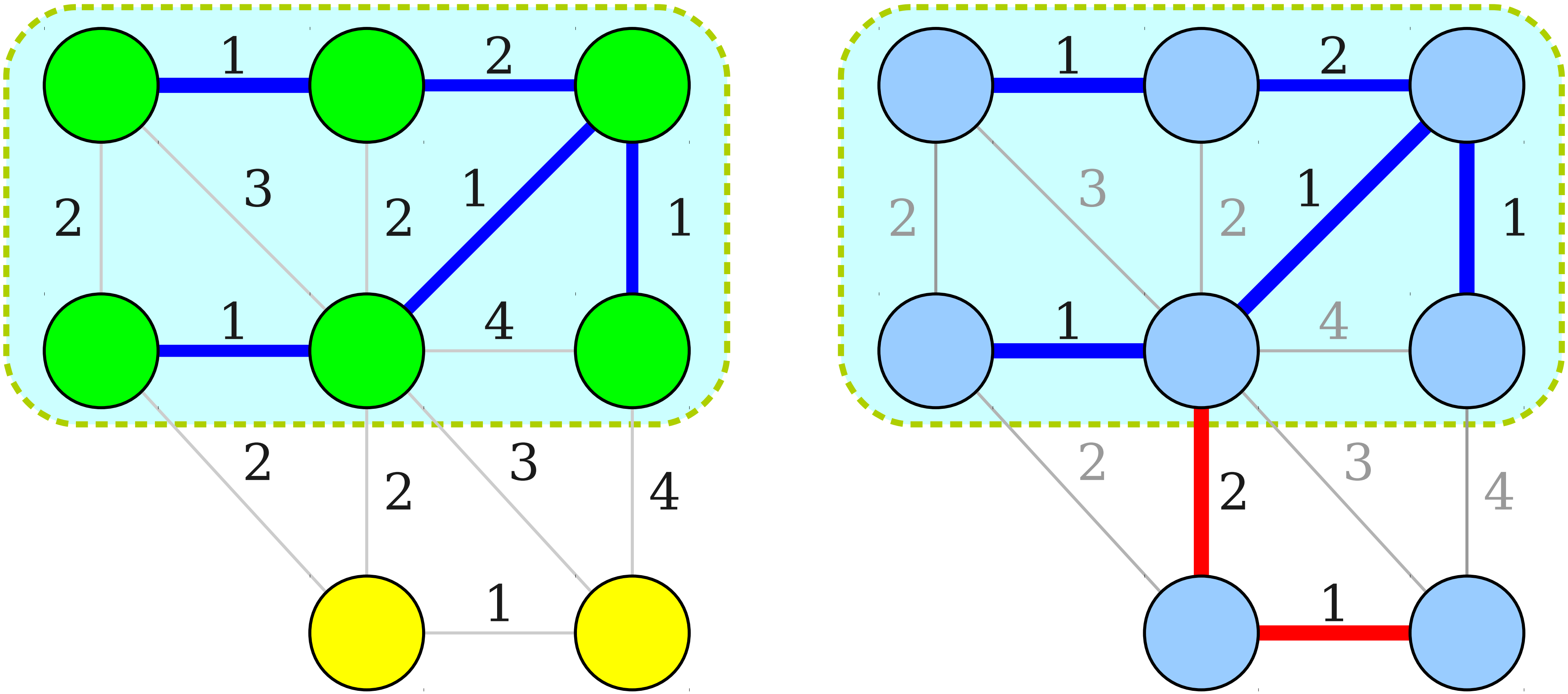

1
2
3
1
2
2
1
1
4
2
3
2
4
1



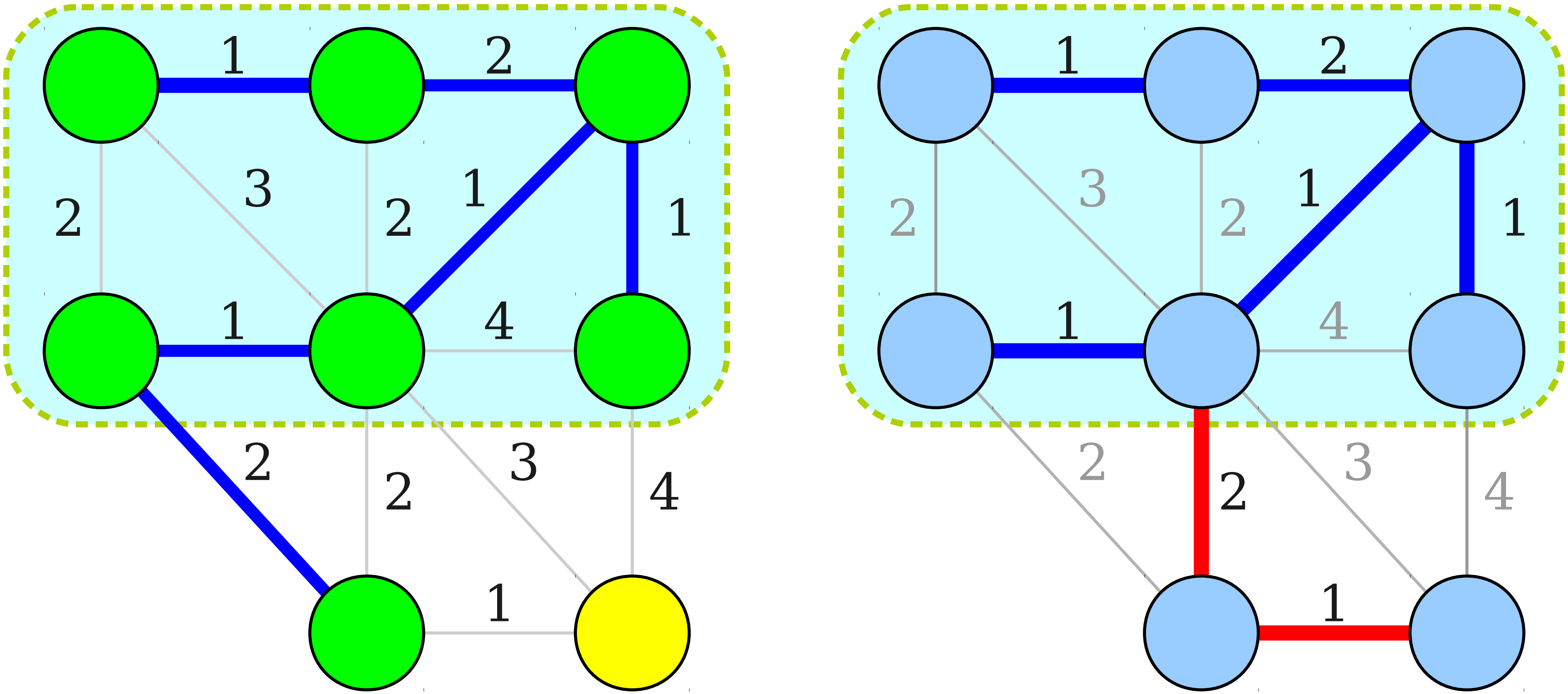
1
2
3
2
1
1
2
1
4
2
3
2
4
1
1
2
3
2
1
1
2
1
4
2
3
2
4
1

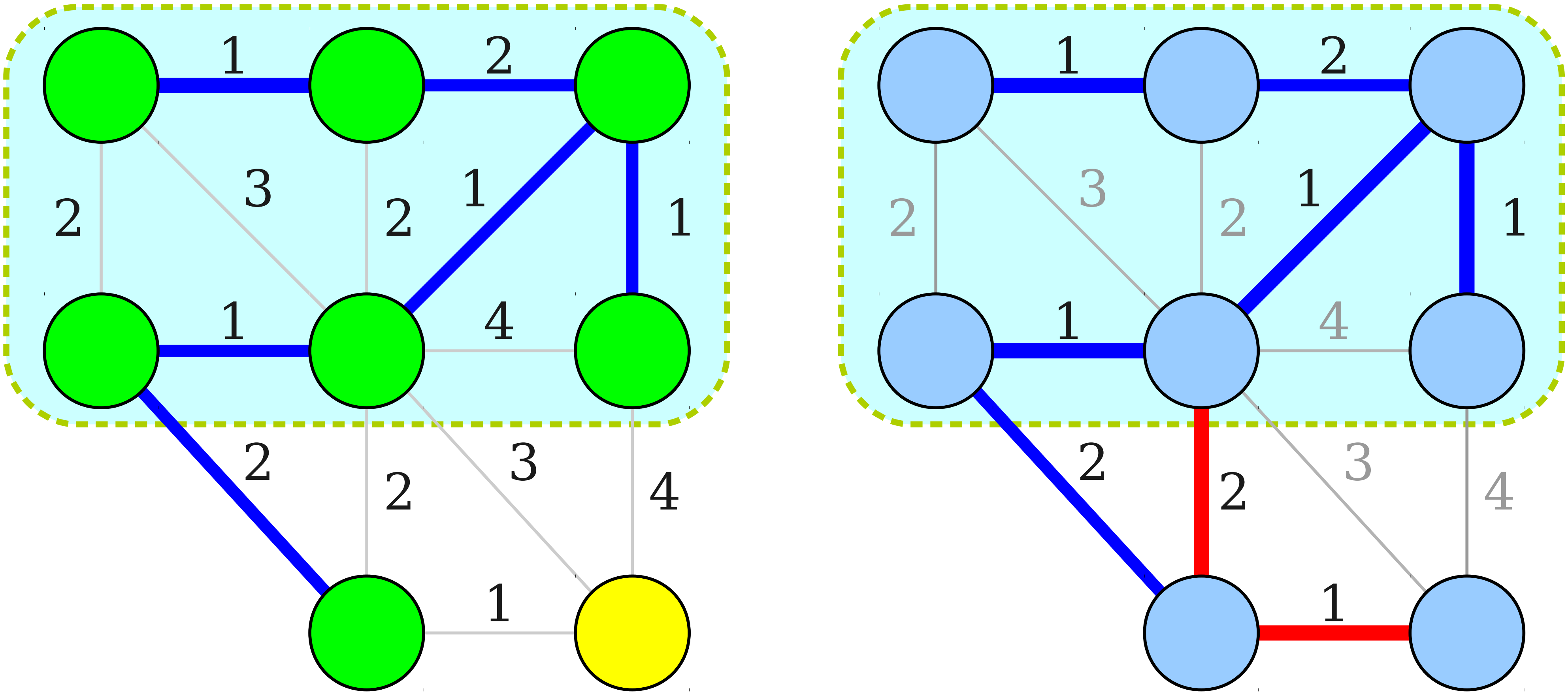

1
2
3
2
1
2
1
1
4
2
2
3
4
1

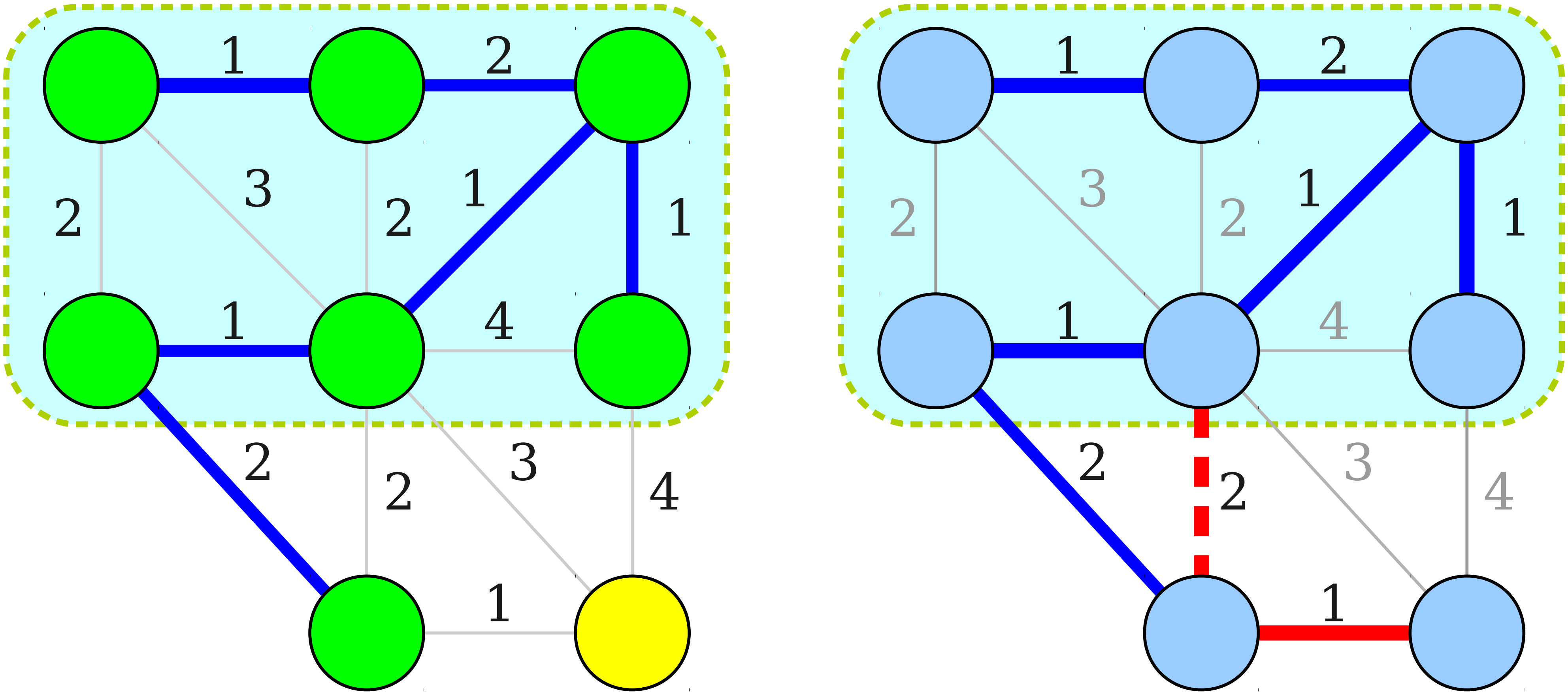

1
2
3
2
1
2
1
1
4
2
2
3
4
1



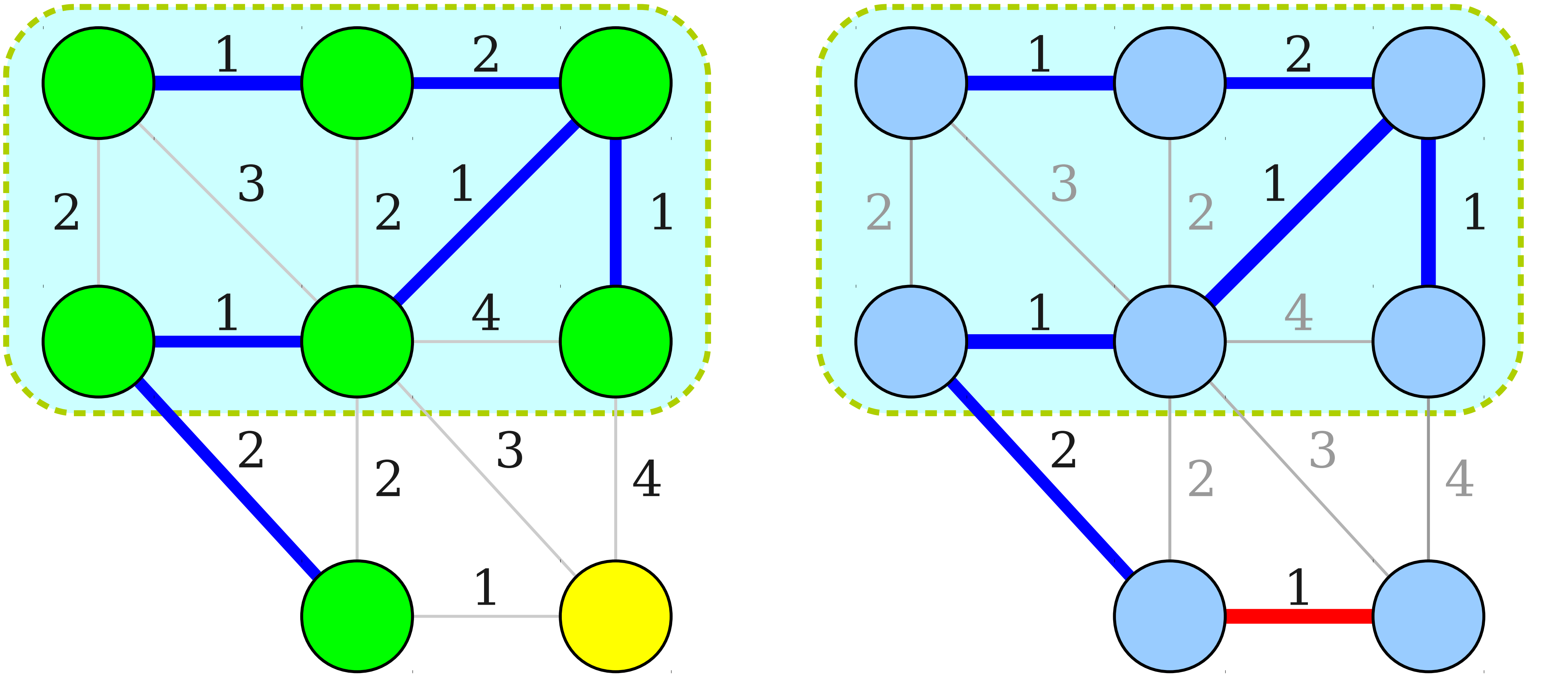
1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

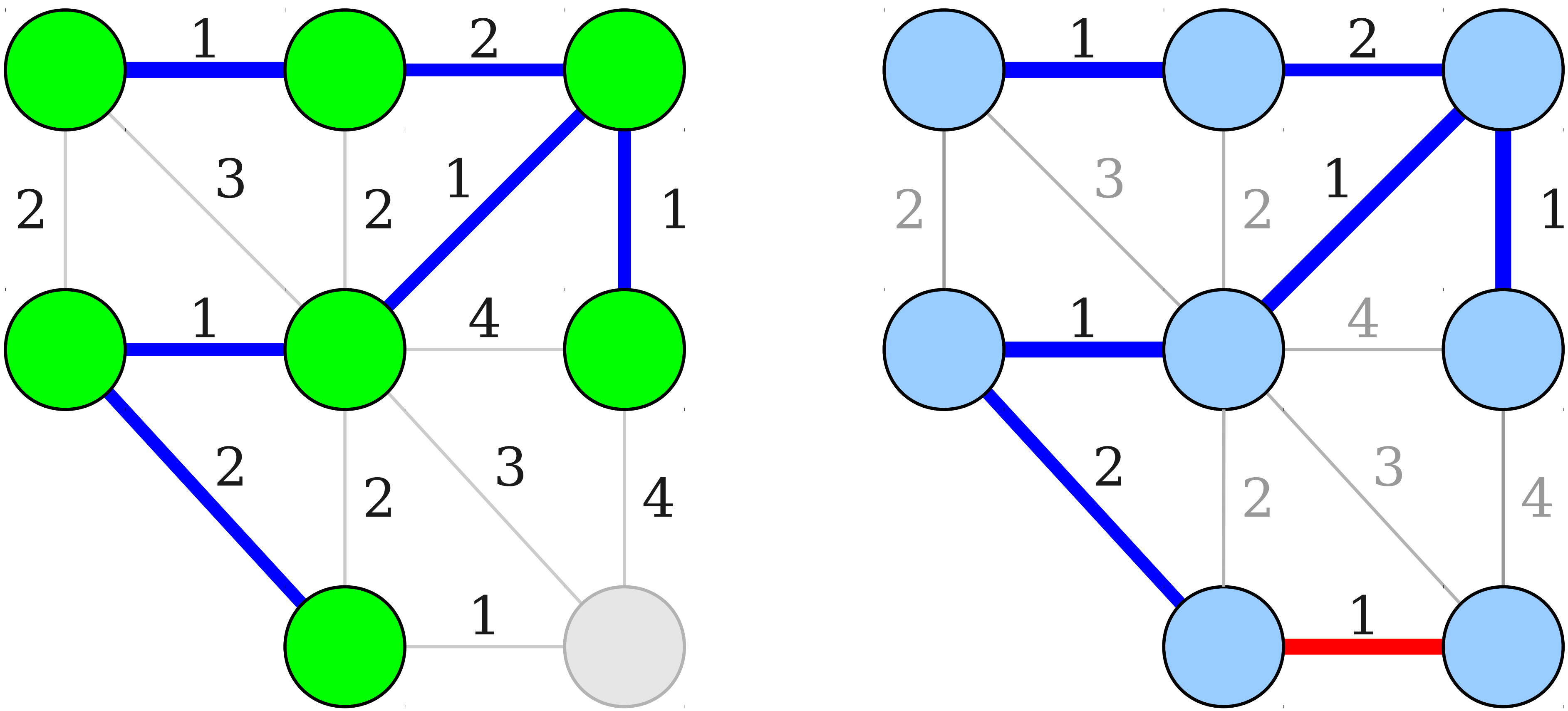

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1


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

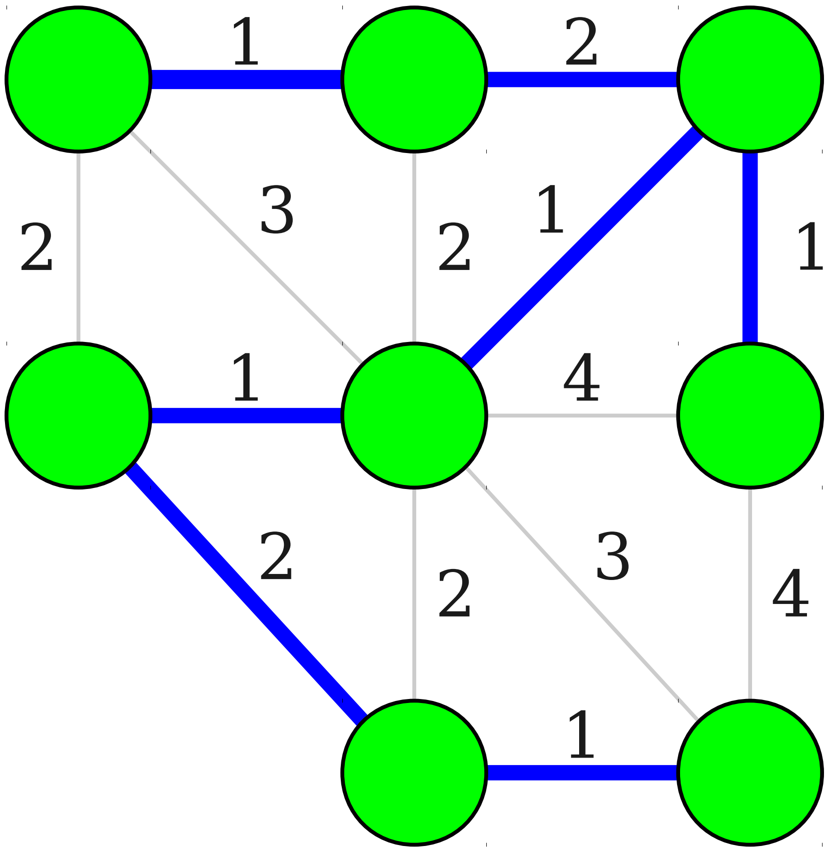


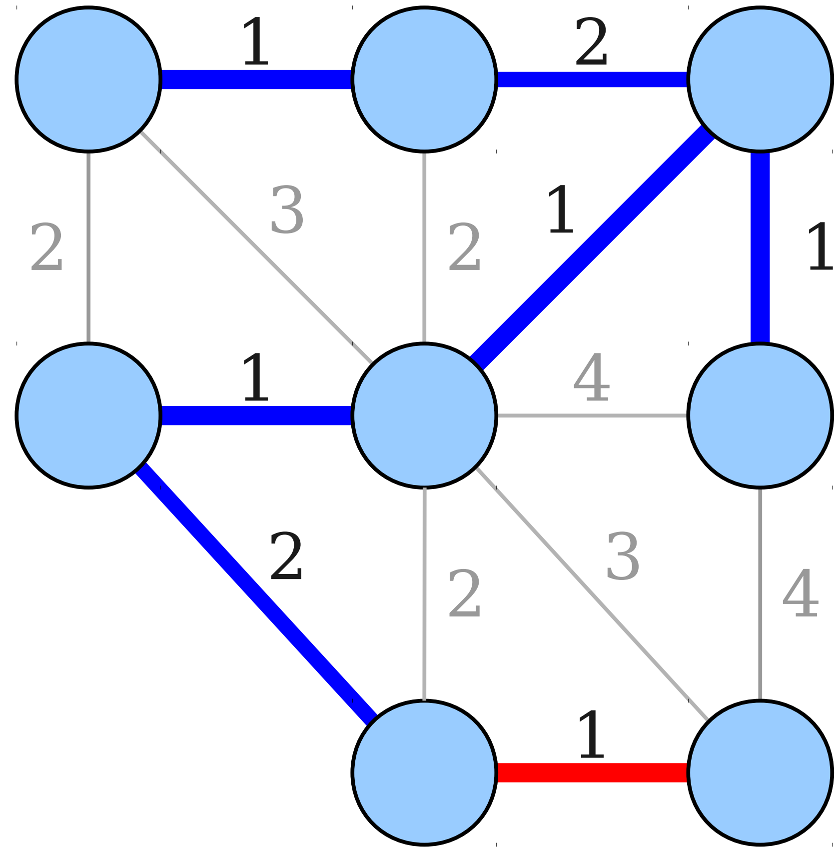
1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

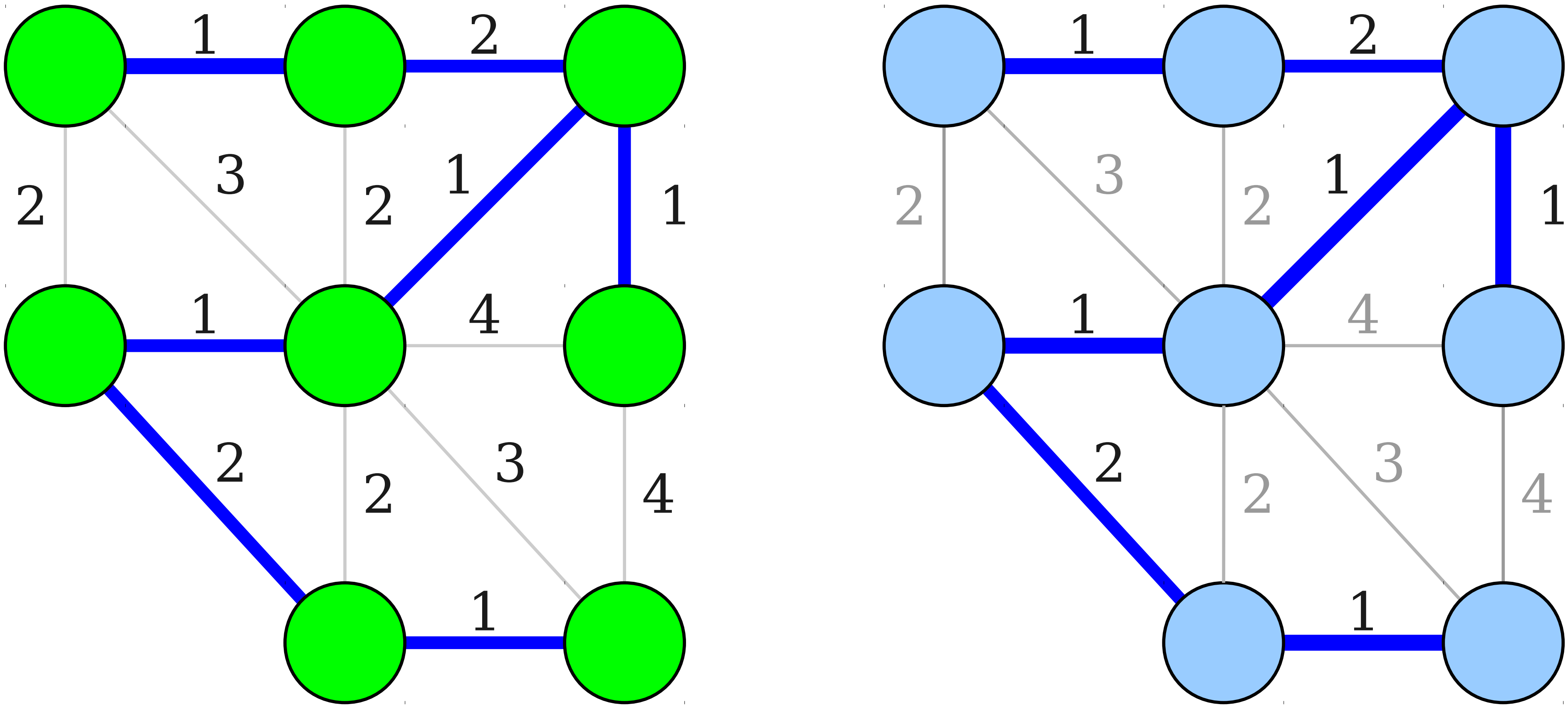

1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1

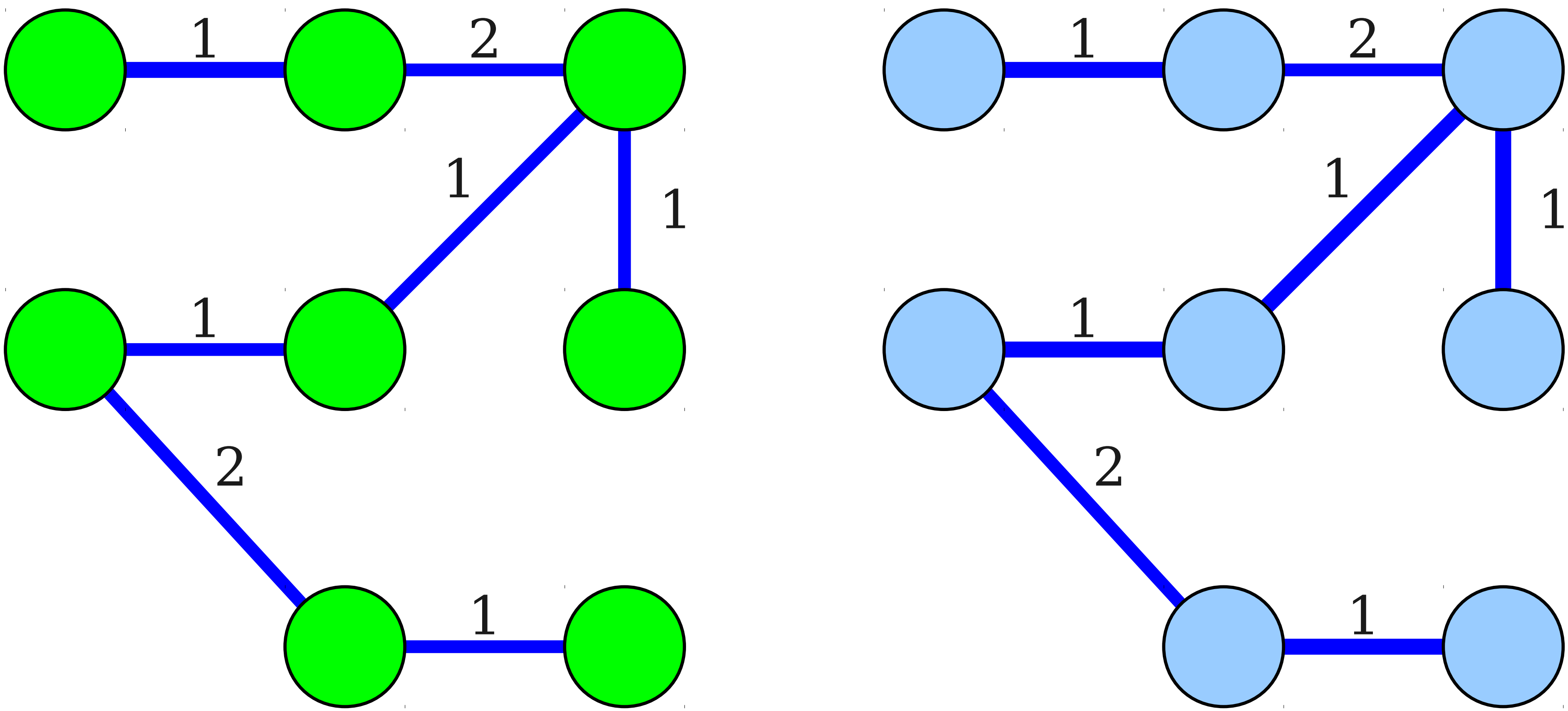


1
2
2
3
2
1
1
1
4
2
2
3
4
1
1
2
2
3
2
1
1
1
4
2
2
3
4
1


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


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


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


1
2
1
1
1
2
1
1
2
1
1
1
2
1

Exchange Arguments

- A more general version of an exchange argument is as follows.
  - Let X be the object produced by a greedy algorithm and X^* be *any* optimal solution.
  - If $X = X^*$, the algorithm is optimal.
  - Otherwise, show that you can *exchange* some piece of X^* for some piece of X without deteriorating the quality of X^*.
  - Argue that this process can be iterated repeatedly to turn X^* into X without changing its cost.
  - Conclude that X is optimal.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. Notice $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) \leq c(T^*)$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. Notice $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) \leq c(T^*)$. Since T^* is an MST, this means $c(T^{*\prime}) \geq c(T^*)$, so $c(T^*) = c(T^{*\prime})$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. Notice $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) \leq c(T^*)$. Since T^* is an MST, this means $c(T^{*\prime}) \geq c(T^*)$, so $c(T^*) = c(T^{*\prime})$.

Note that $|T - T^{*\prime}| = |T - T^*| - 1$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. Notice $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) \leq c(T^*)$. Since T^* is an MST, this means $c(T^{*\prime}) \geq c(T^*)$, so $c(T^*) = c(T^{*\prime})$.

Note that $|T - T^{*\prime}| = |T - T^*| - 1$. Therefore, if we repeat this process once for each edge in $T - T^*$, we will have converted T^* into T while preserving $c(T^*)$.

Theorem: If G is a connected, weighted graph, Prim's algorithm correctly finds an MST in G.

Proof: Let T be the spanning tree found by Prim's algorithm and T^* be any MST of G. We will prove $c(T) = c(T^*)$. If $T = T^*$, then $c(T) = c(T^*)$ and we are done.

Otherwise, $T \neq T^*$, so we have $T - T^* \neq \emptyset$. Let (u, v) be any edge in $T - T^*$. When (u, v) was added to T, it was a least-cost edge crossing some cut $(S, V - S)$. Since T^* is an MST, there must be a path from u to v in T^*. This path begins in S and ends in $V - S$, so there must be some edge (x, y) along that path where $x \in S$ and $y \in V - S$. Since (u, v) is a least-cost edge crossing $(S, V - S)$, we have $c(u, v) \leq c(x, y)$.

Let $T^{*\prime} = T^* \cup \{(u, v)\} - \{(x, y)\}$. Since (x, y) is on the cycle formed by adding (u, v), this means $T^{*\prime}$ is a spanning tree. Notice $c(T^{*\prime}) = c(T^*) + c(u, v) - c(x, y) \leq c(T^*)$. Since T^* is an MST, this means $c(T^{*\prime}) \geq c(T^*)$, so $c(T^*) = c(T^{*\prime})$.

Note that $|T - T^{*\prime}| = |T - T^*| - 1$. Therefore, if we repeat this process once for each edge in $T - T^*$, we will have converted T^* into T while preserving $c(T^*)$. Thus $c(T) = c(T^*)$. ■

A Note on the Proof

- Our proof worked as follows:
  - Find a way to replace one piece of T^* with one piece of T without increasing $c(T^*)$.
  - Note that this makes T^* "less different" than T as before.
  - Conclude that we could iterate this process until eventually T^* became T, at which point we have $c(T) = c(T^*)$.
- This is inherently an inductive argument, but typically it is not presented as such.
  - It's fine to say "repeat this process" rather than writing out a base case and inductive step.

Next Time

- Kruskal's Algorithm
- Disjoint-Set Forests