Chapter 7:
Lines, Paragraphs, Paths

Panos Louridas

Athens University of Economics and Business
Real World Algorithms
A Beginners Guide
The MIT Press

Outline

Line Breaking

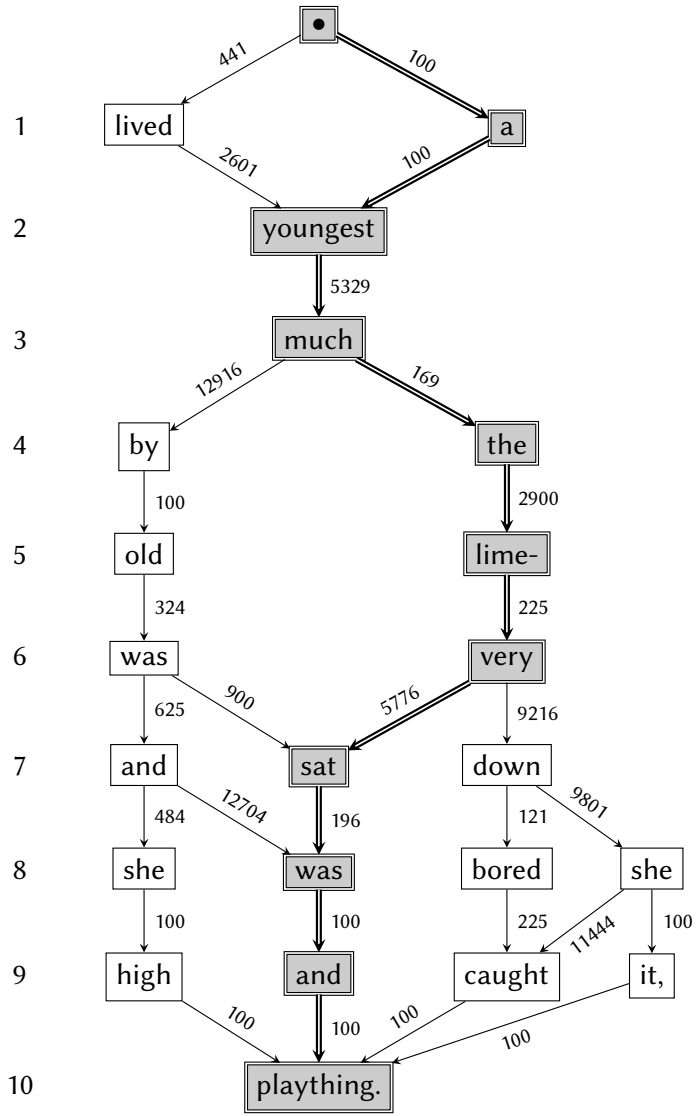
In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain, and when she was bored she took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain, and when she was bored she took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

Another Line Breaking Example

In olden times when wishing still helped one, there lived a king whose daughters were all beautiful, but the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king's castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king's child went out into the forest and sat down by the side of the cool fountain, and when she was bored she took a golden ball, and threw it up on high and caught it, and this ball was her favorite plaything.

Optimum Line Breaking



Another Line Breaking Example

was astonished whenever it shone in her face. Close by was astonished whenever it shone in her face. Close by the

Applications

- TEX (Knuth, Plass).
- LATEX.
- Adobe Corporation.

Outline

The Shortest Paths Problem

Definition

Given a graph $G = (V, E)$ and a vertex s of the graph, find the shortest paths from s to every other vertex of the graph.

Applications

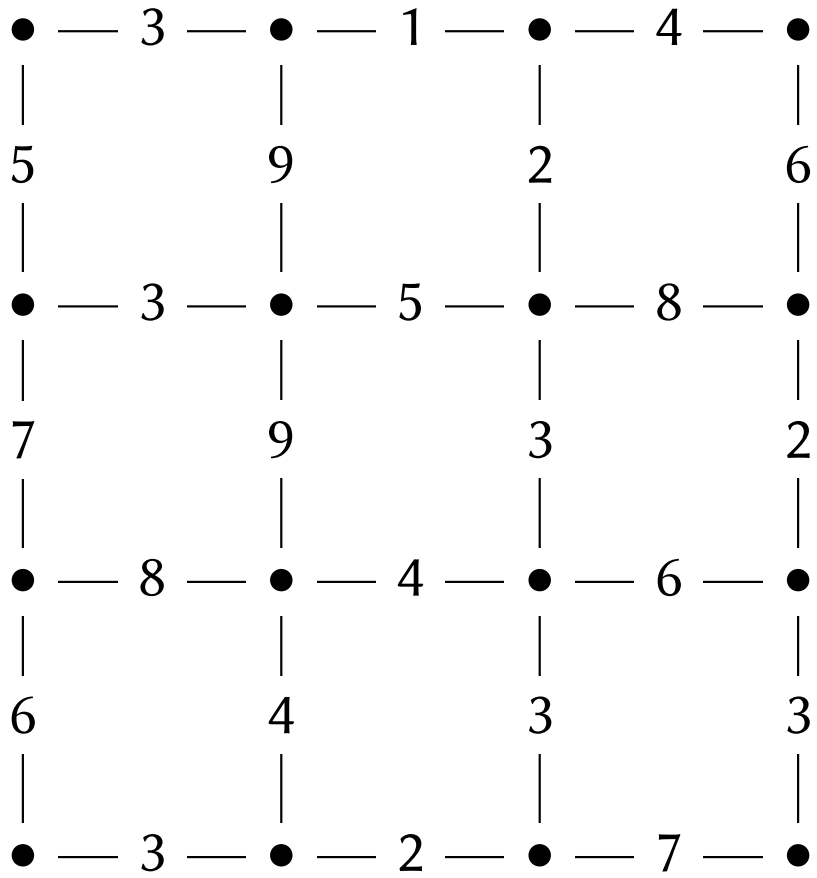
- Line breaking and paragraph making.
- Map navigation.
- Routing in traffic.
- Air travel optimization.
- Routing in communication networks.
- Path finding in robots.
- ...

Outline

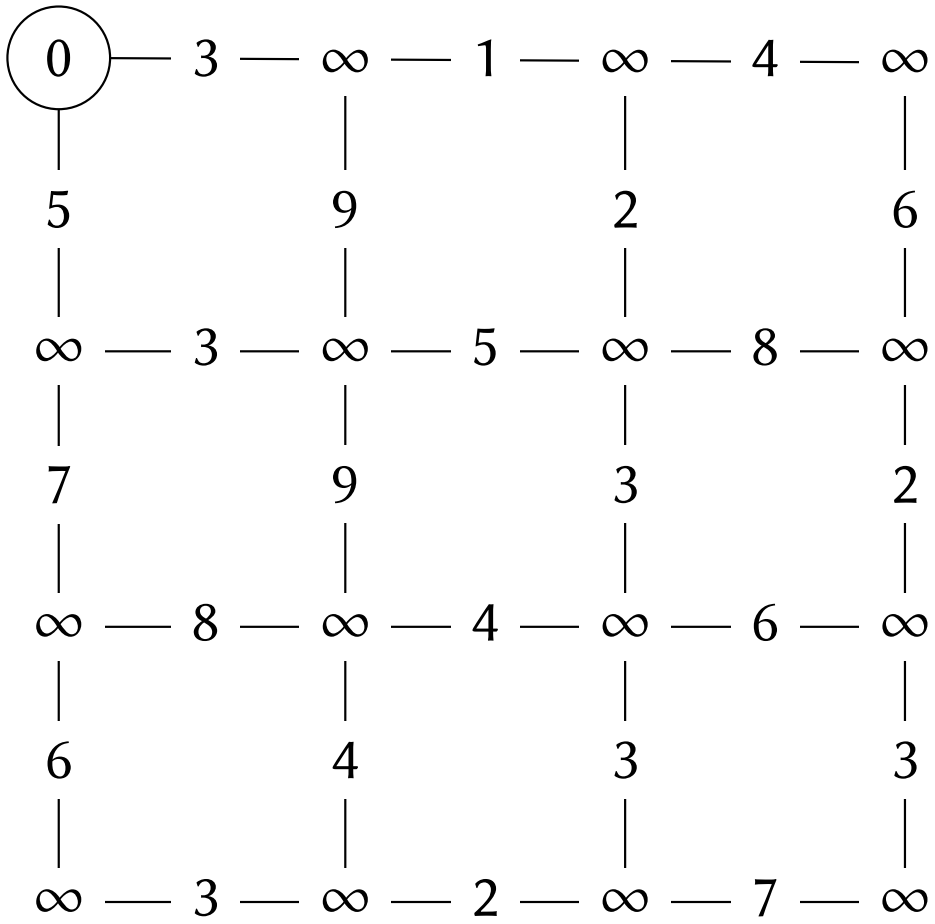
Basic Idea

- It was invented by the Dutch computer scientist Edsger Dijkstra in 1956, was published in 1959.
- We initialize the graph with estimates for the shortest paths from the starting node.
- The estimate for the starting node itself is 0, for all other nodes it is ∞.
- We select the vertex with the best estimate and we update (relax) the estimates for all its neighbors.
- We repeat as many times as there are nodes in the graph.

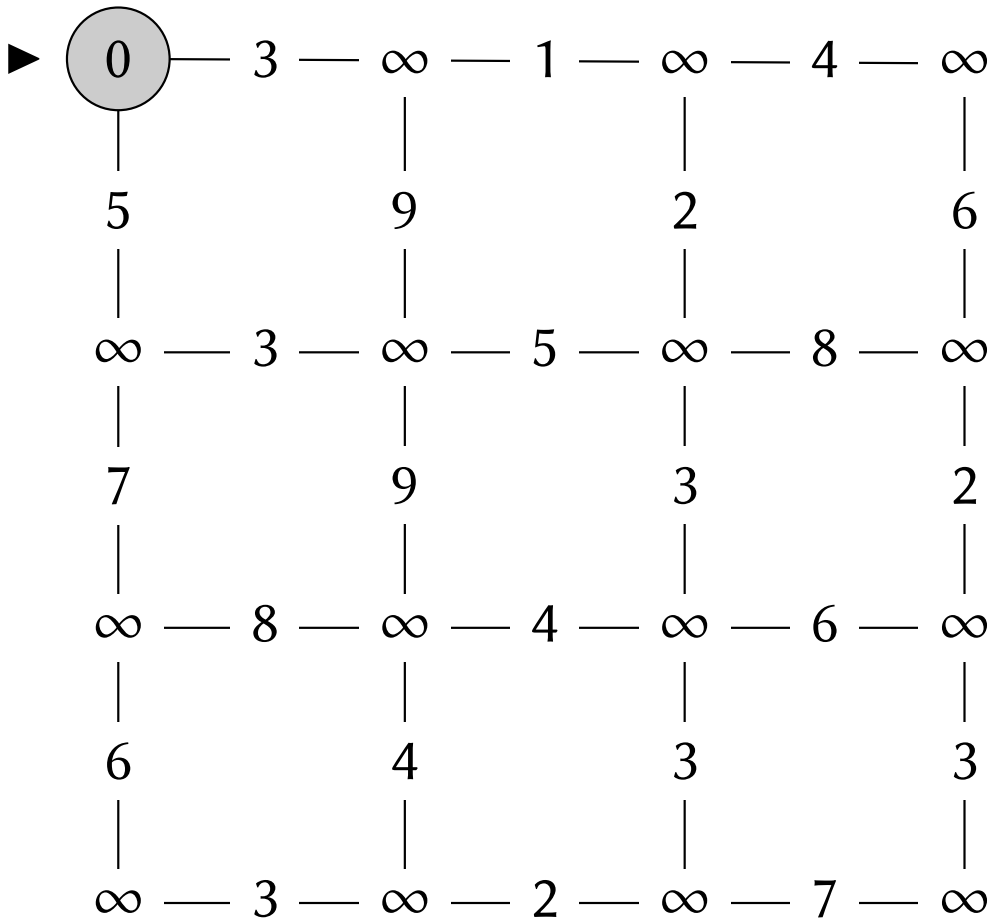
A Traffic Grid



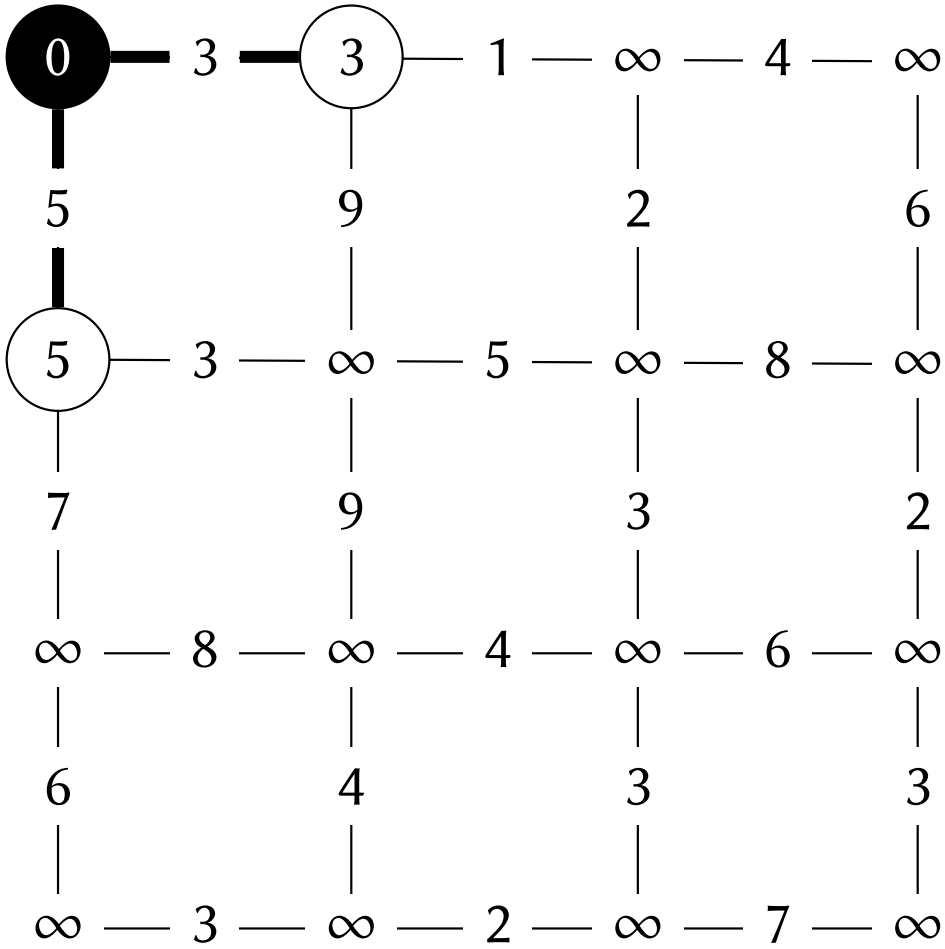
Dijkstra (1)



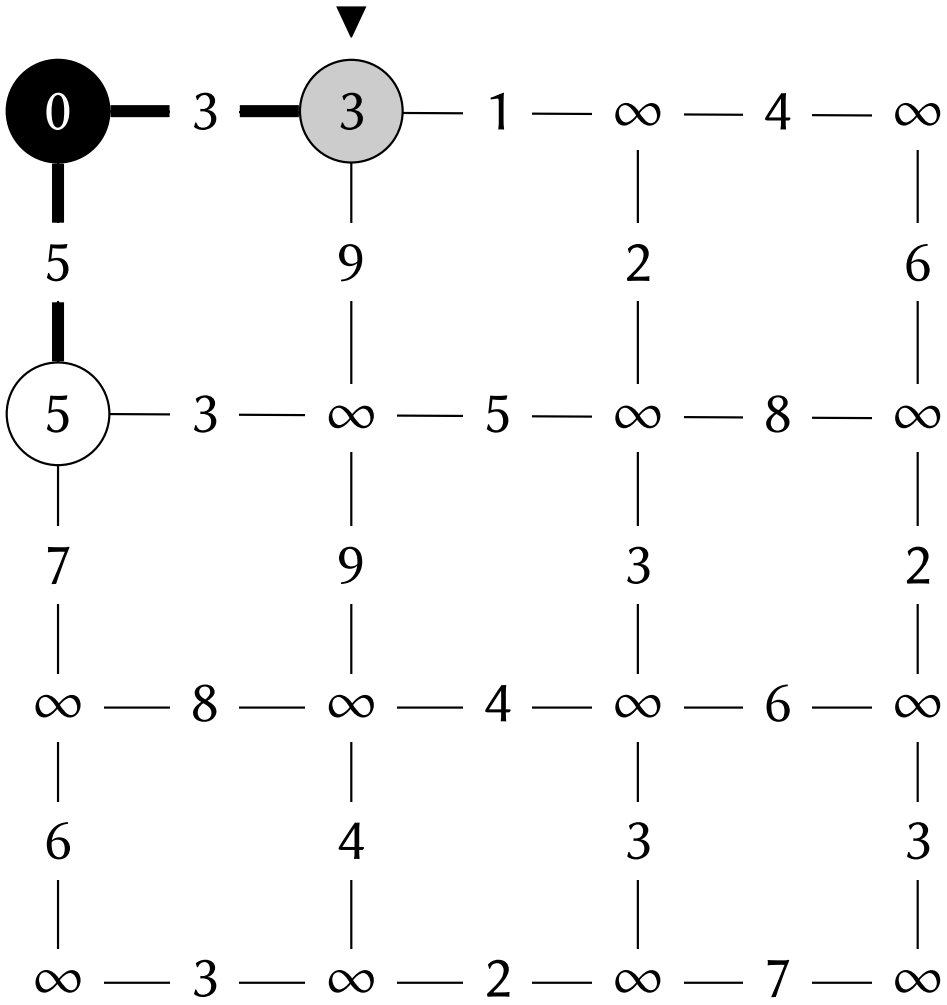
Dijkstra (1.1)



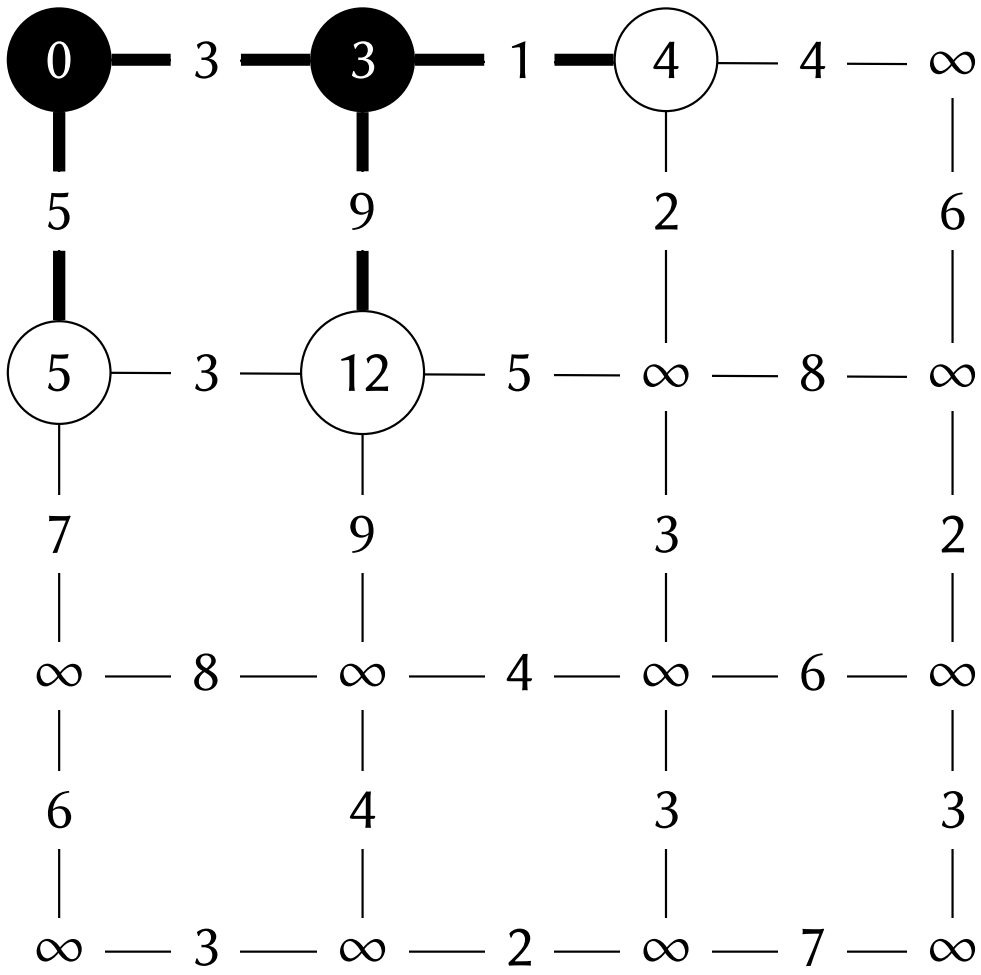
Dijkstra (1.2)



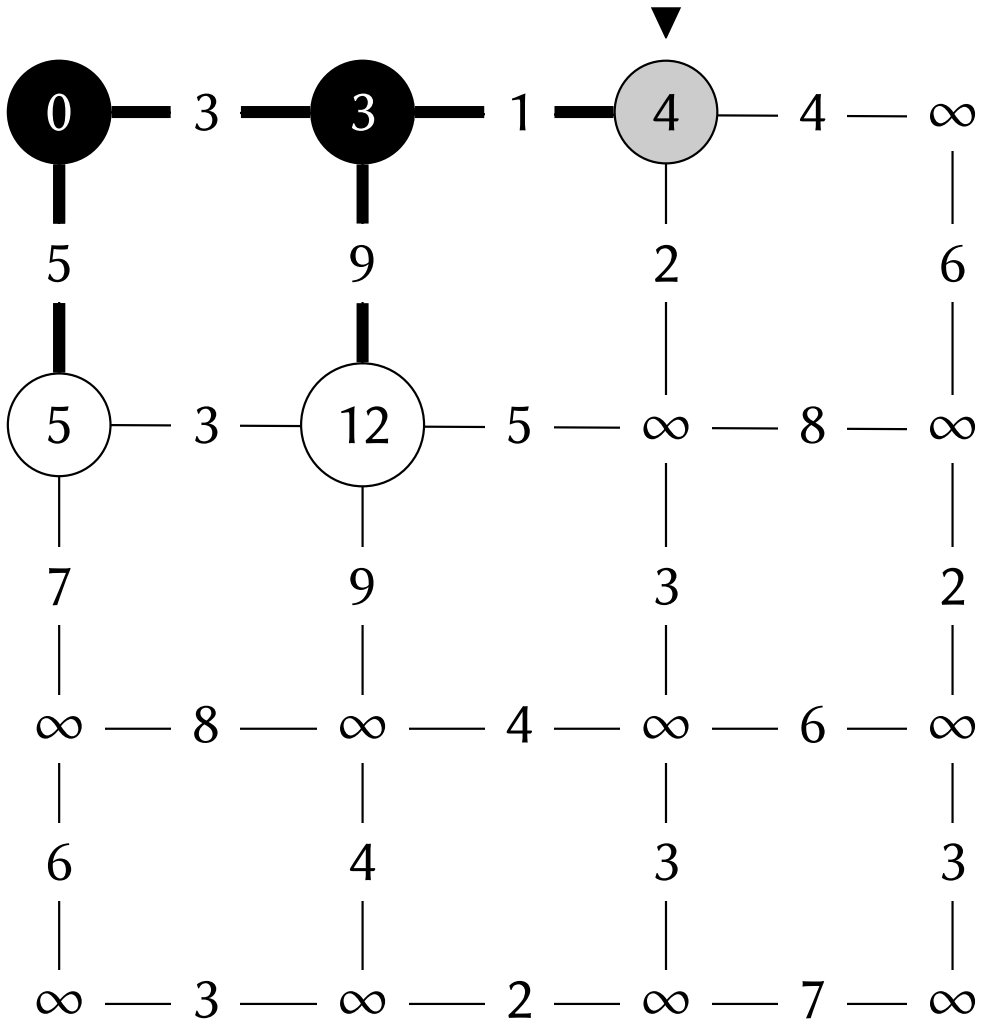
Dijkstra (2.1)



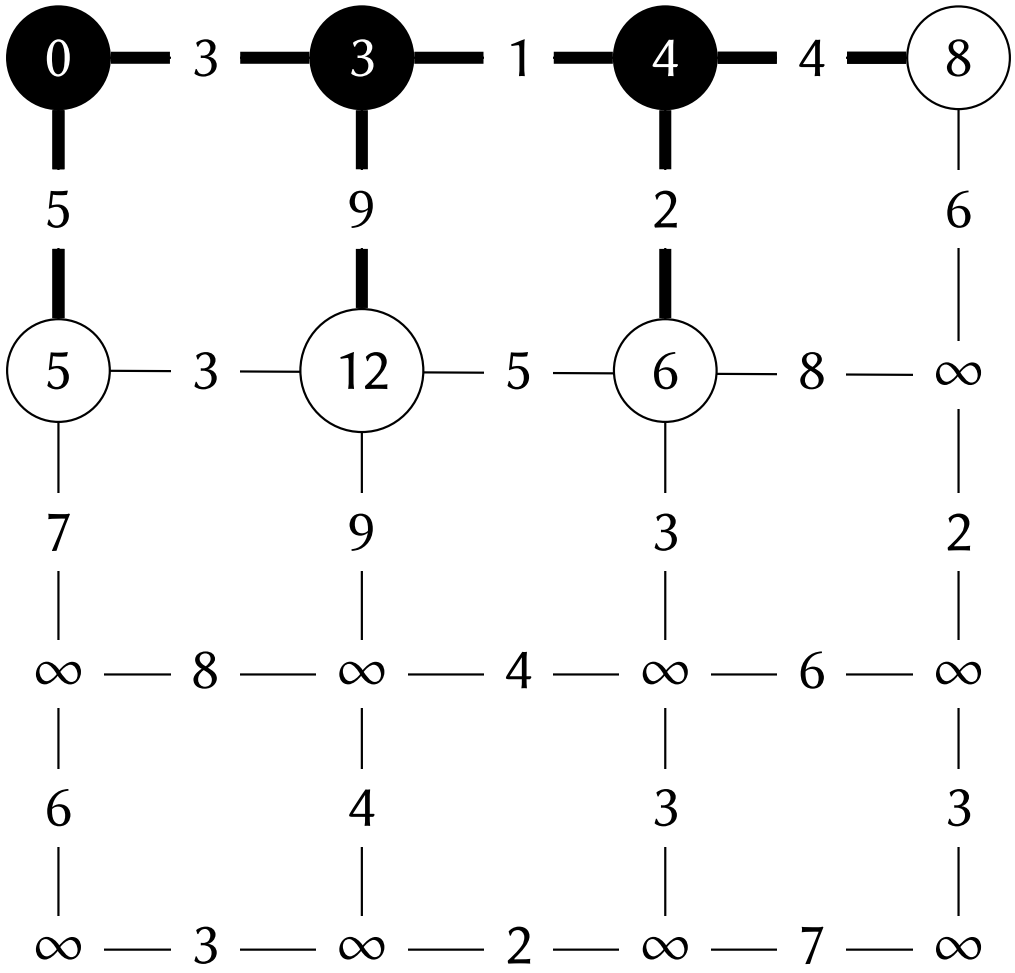
Dijkstra (2.2)



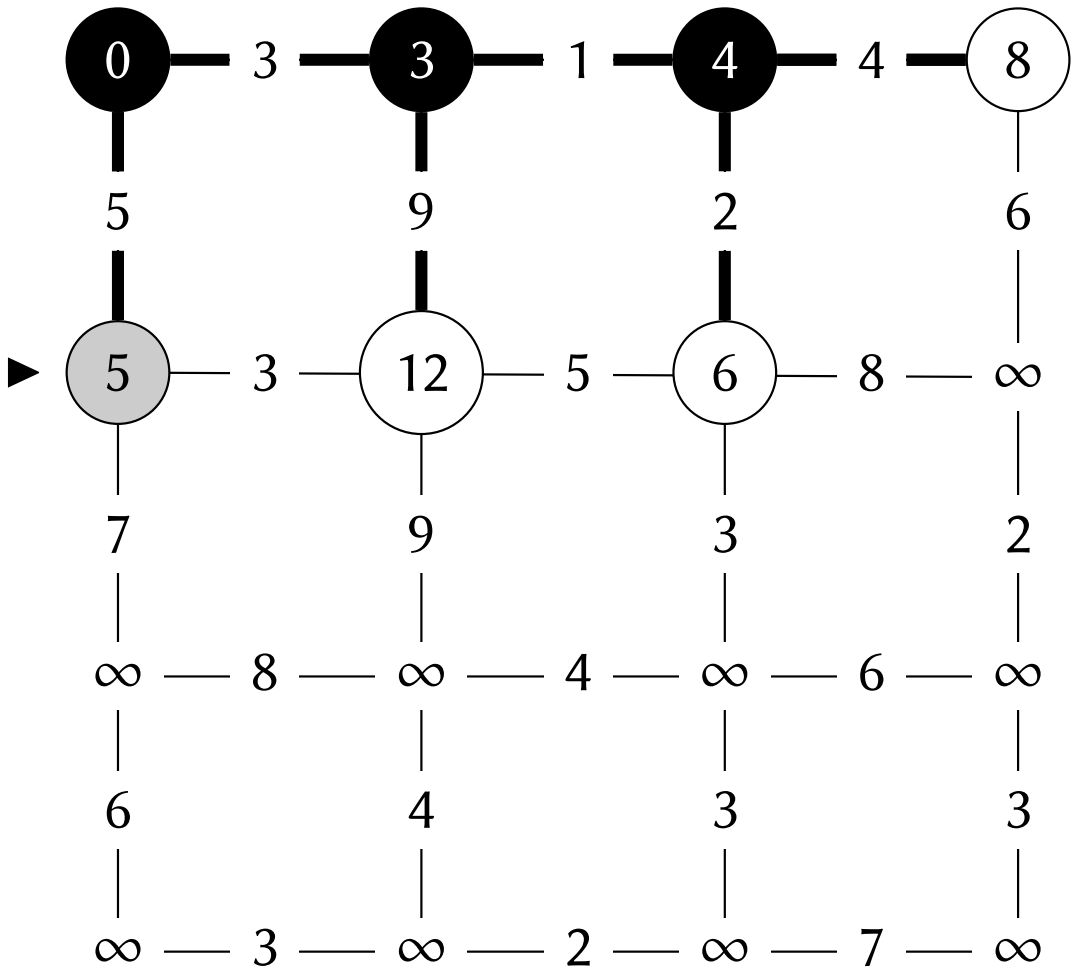
Dijkstra (3.1)



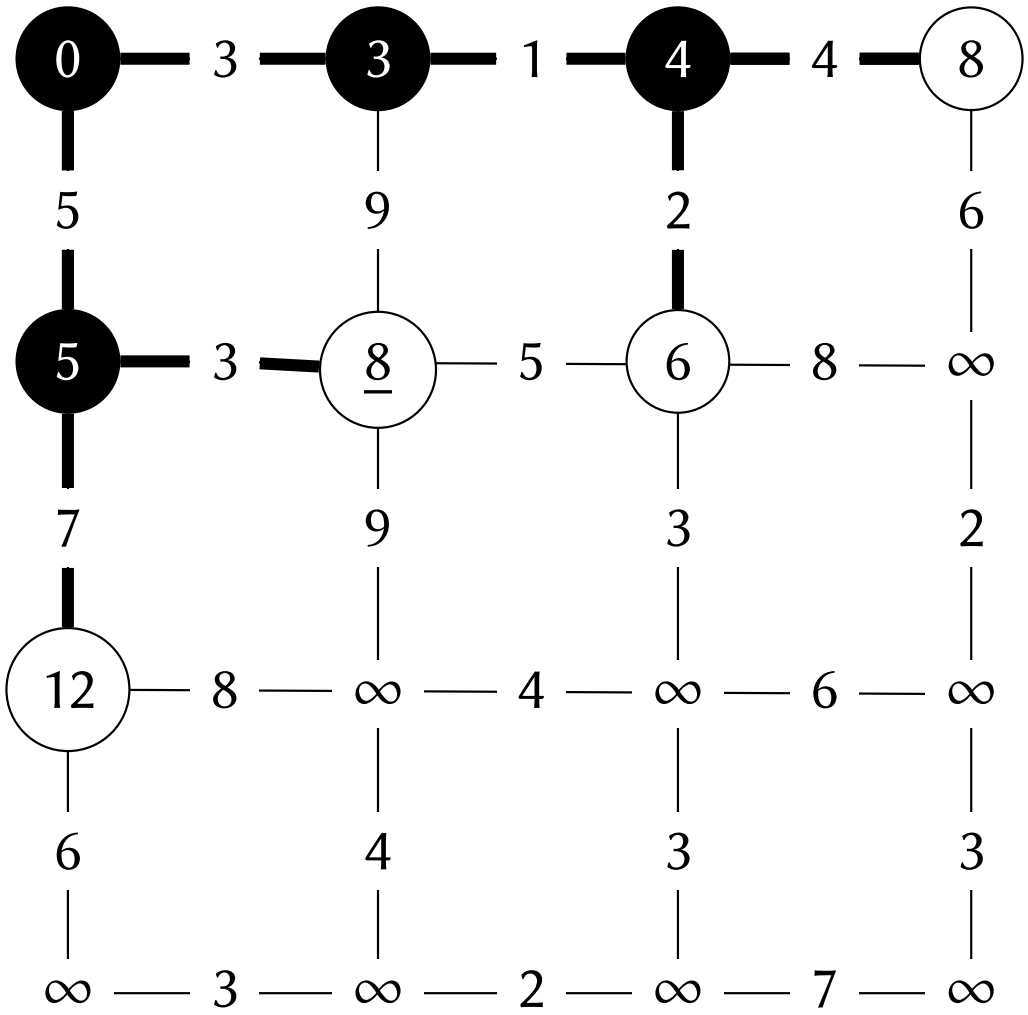
Dijkstra (3.2)



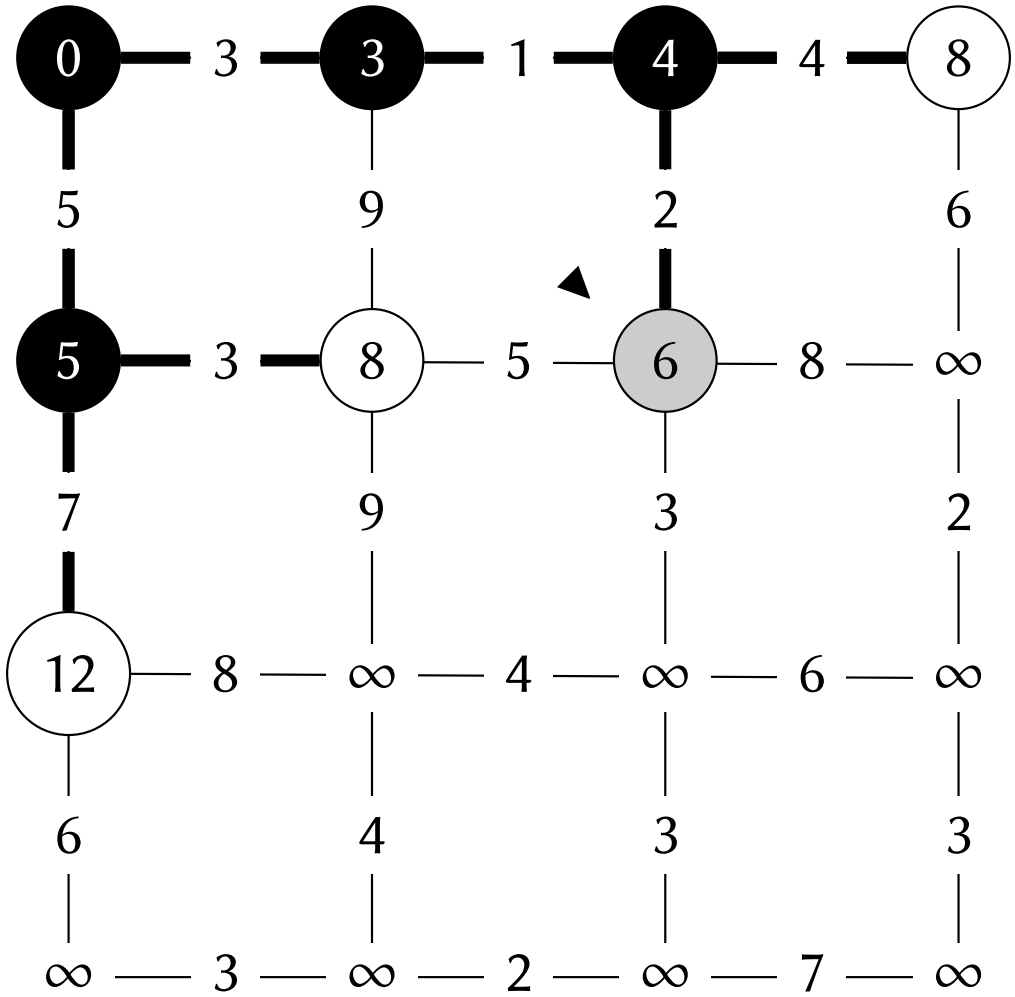
Dijkstra (4.1)



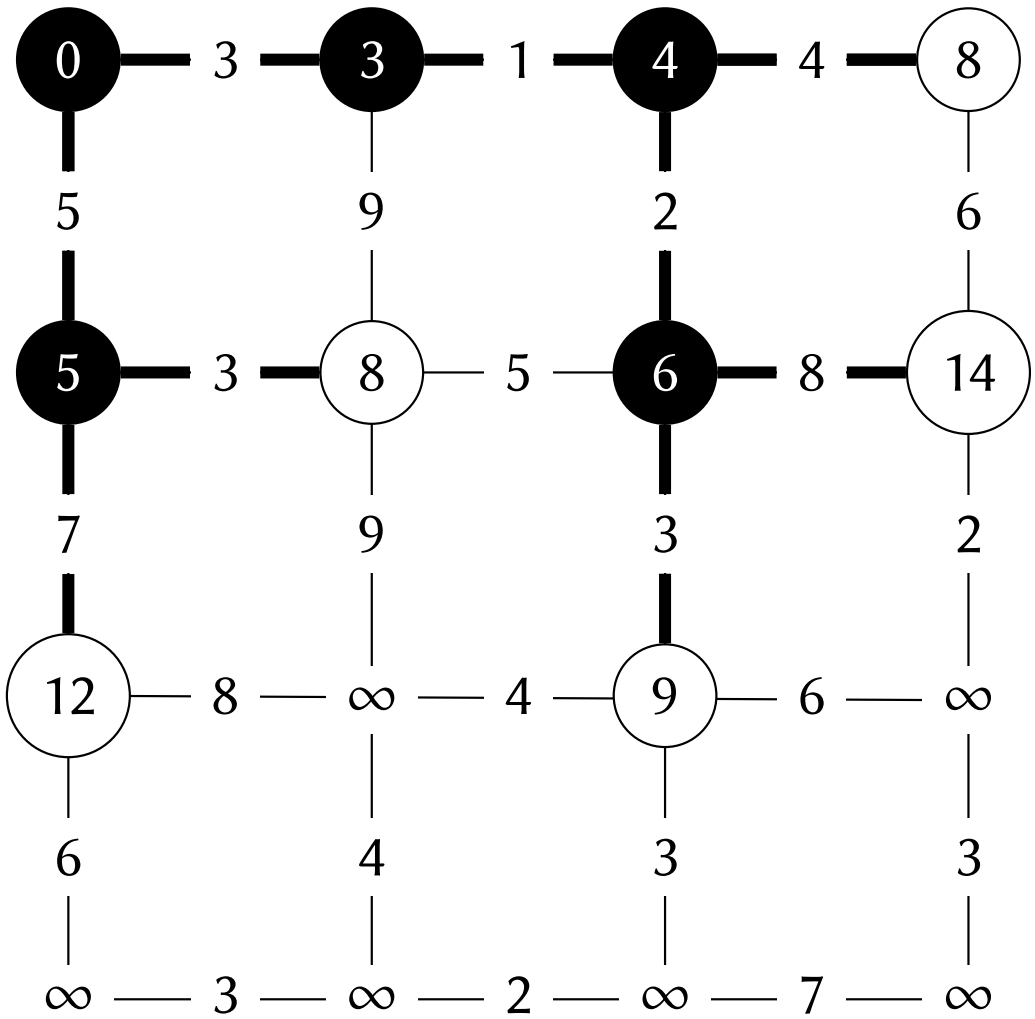
Dijkstra (4.2)



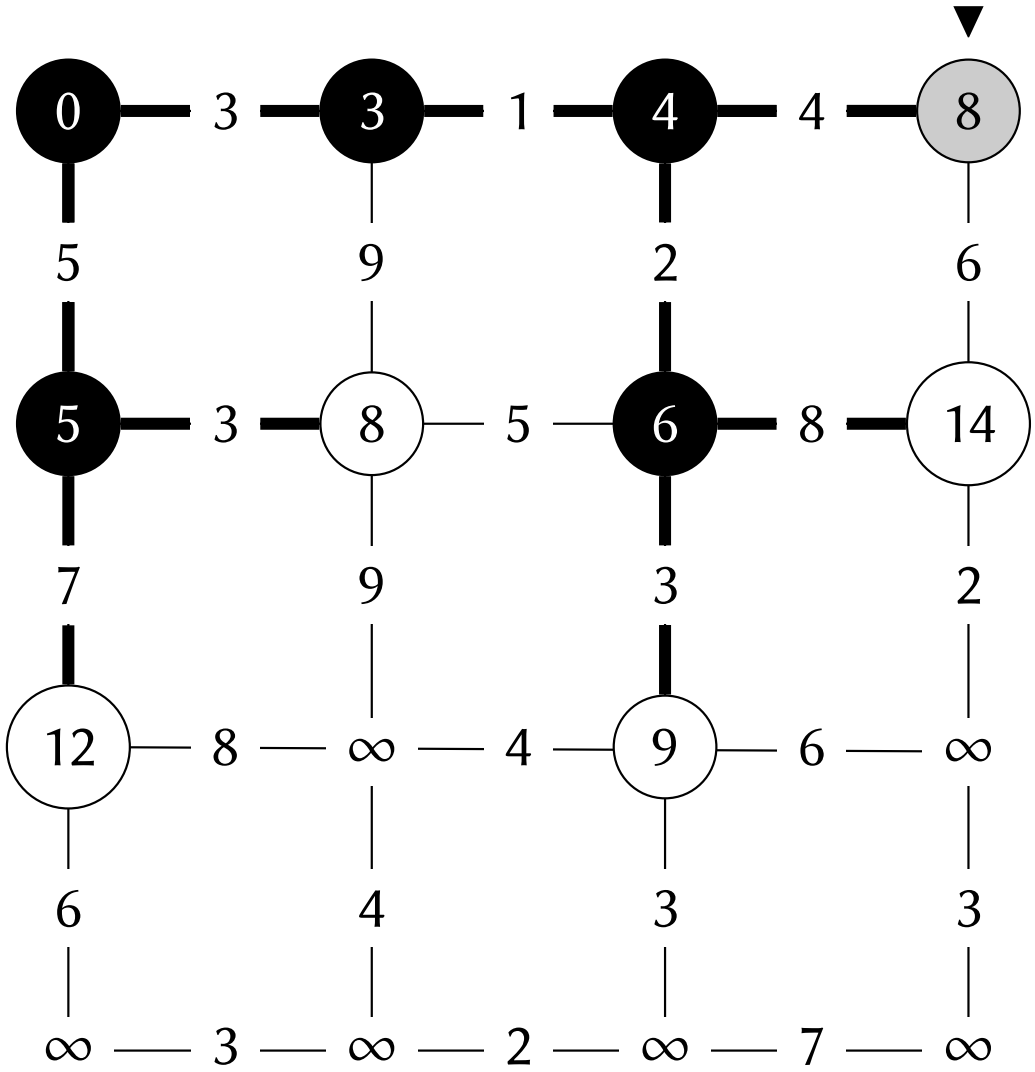
Dijkstra (5.1)



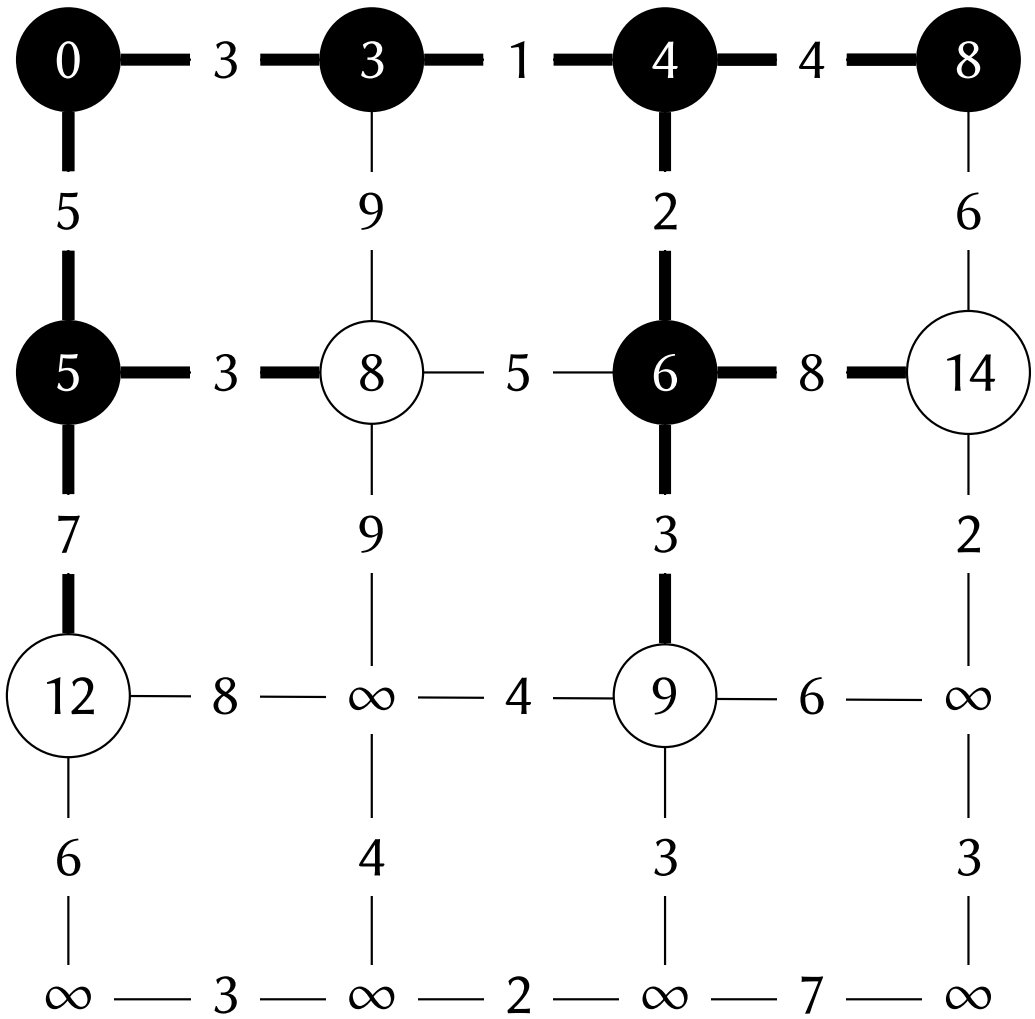
Dijkstra (5.2)



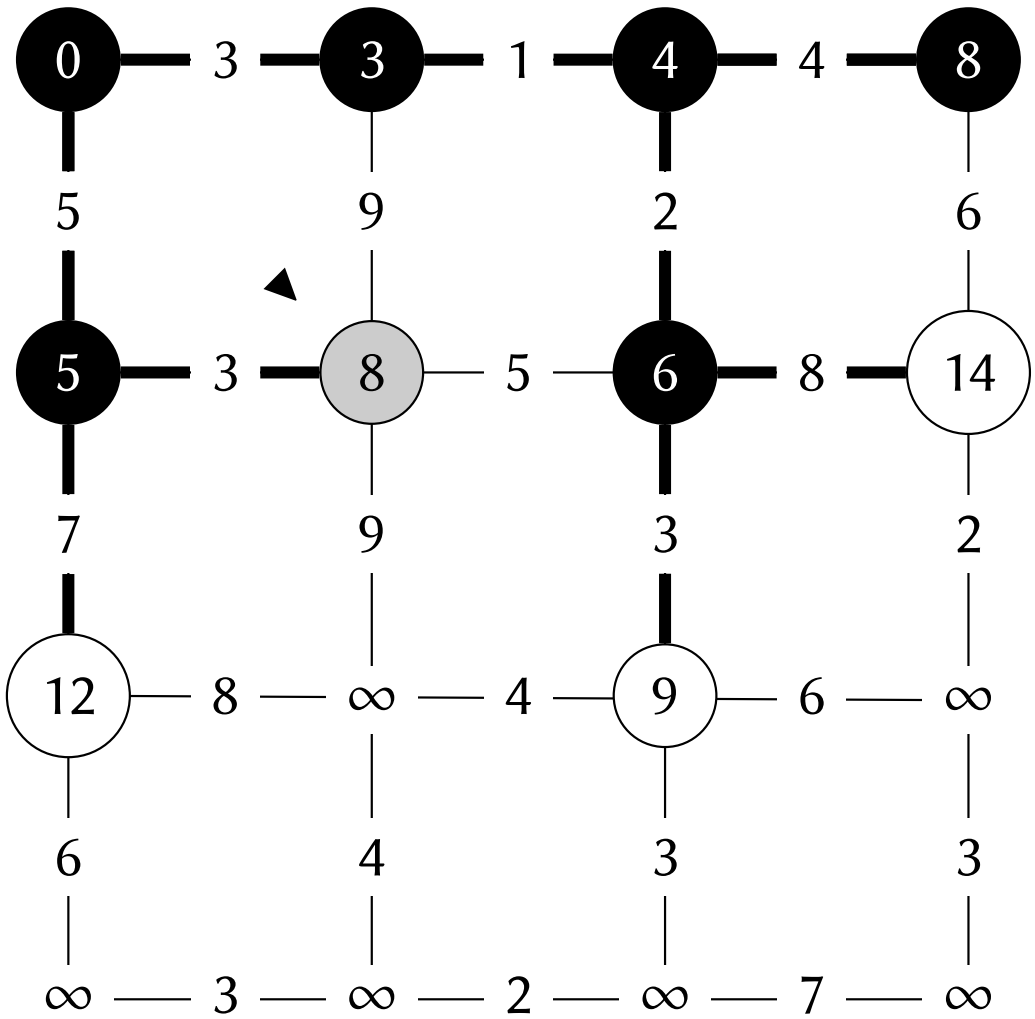
Dijkstra (6.1)



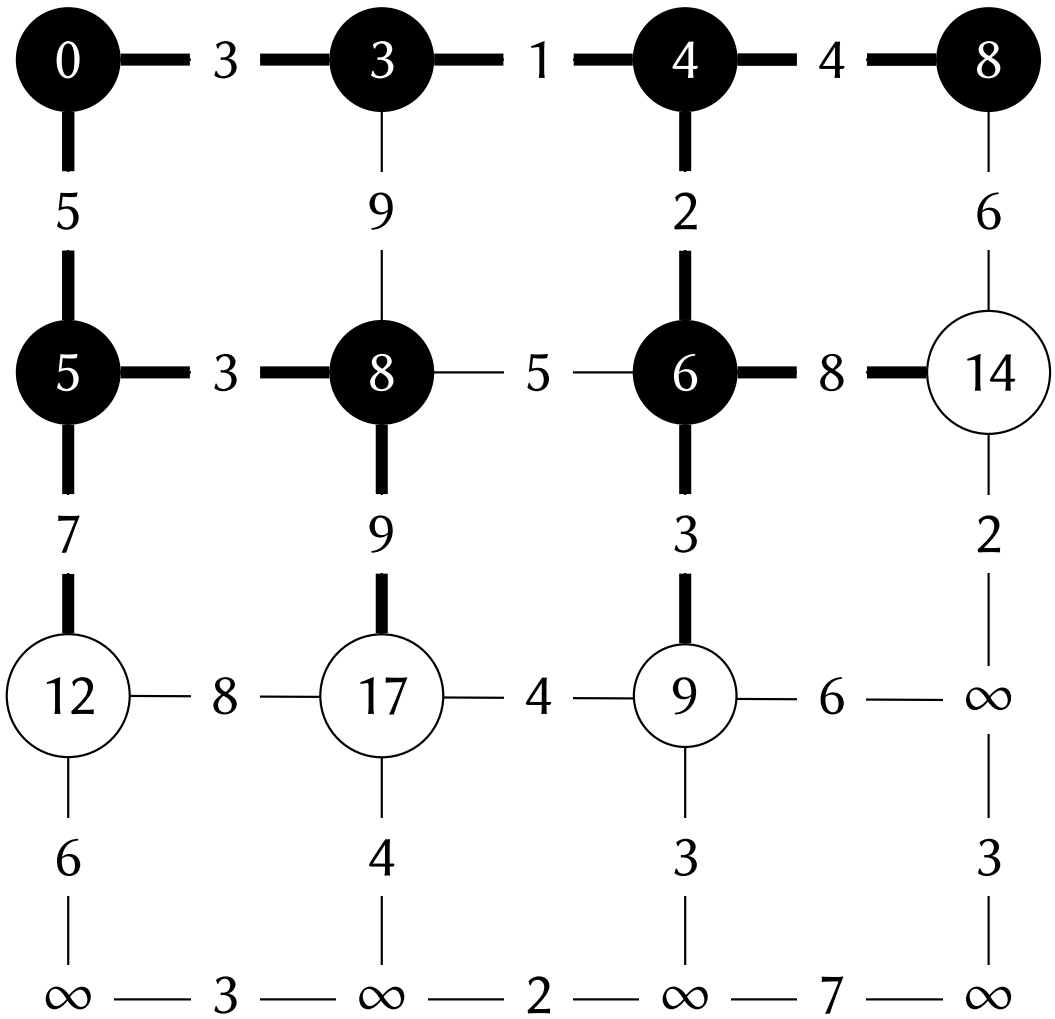
Dijkstra (6.2)



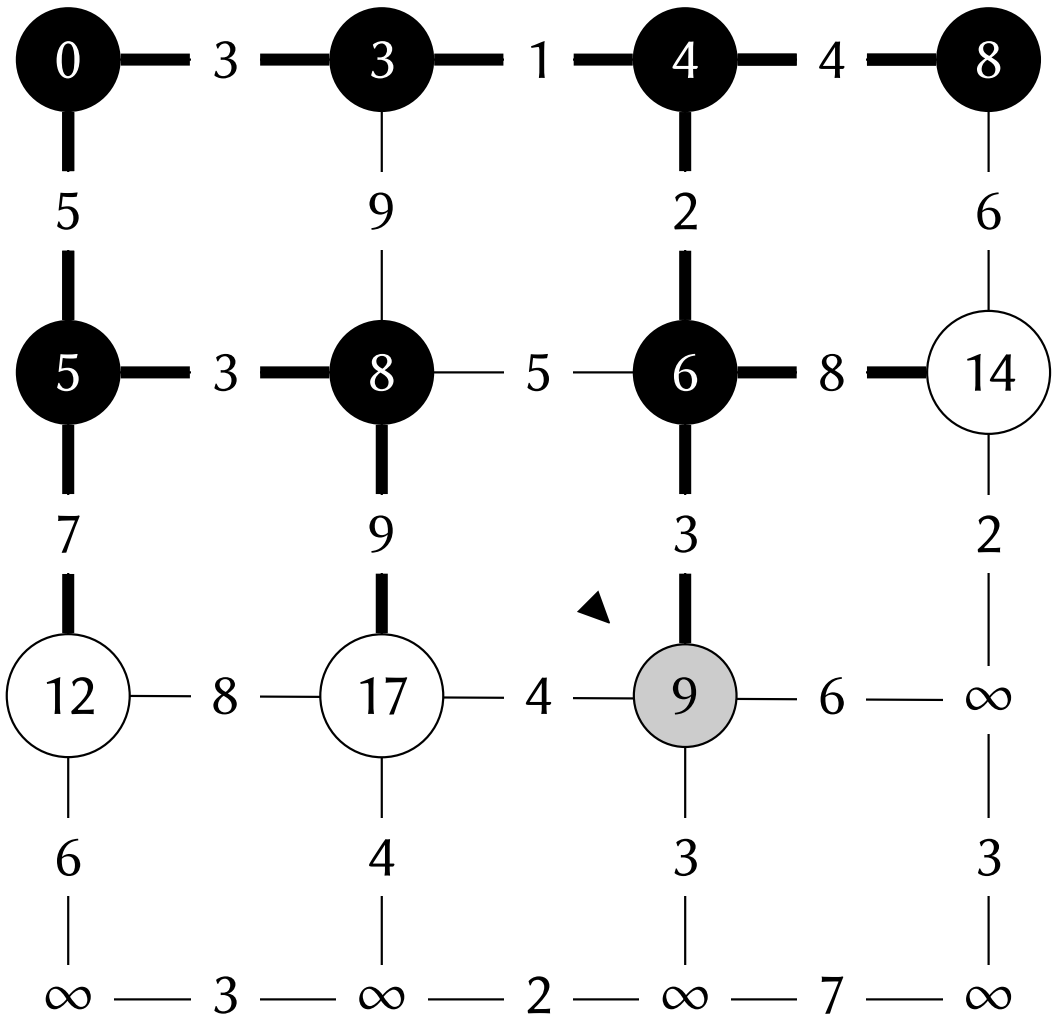
Dijkstra (7.1)



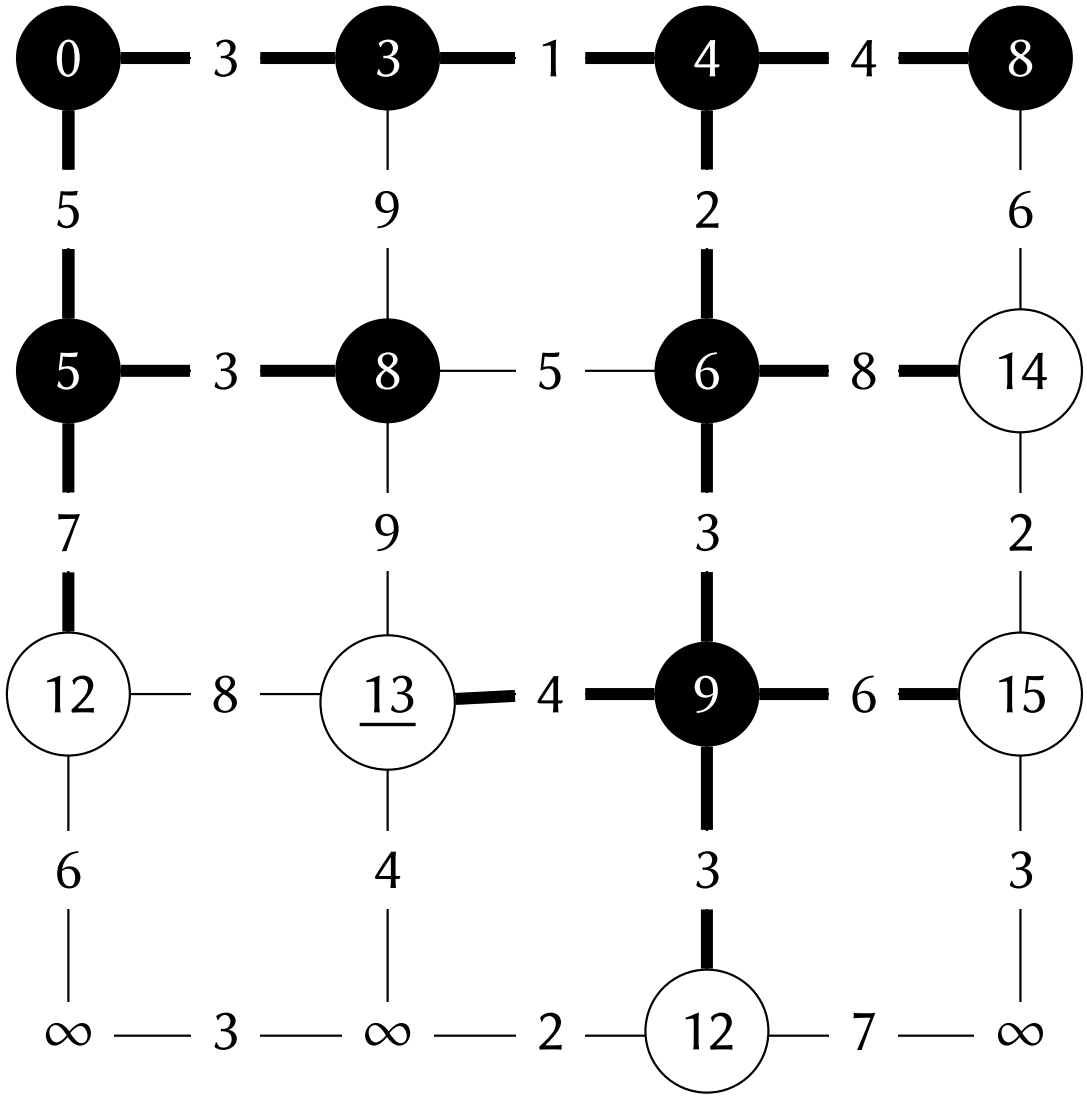
Dijkstra (7.2)



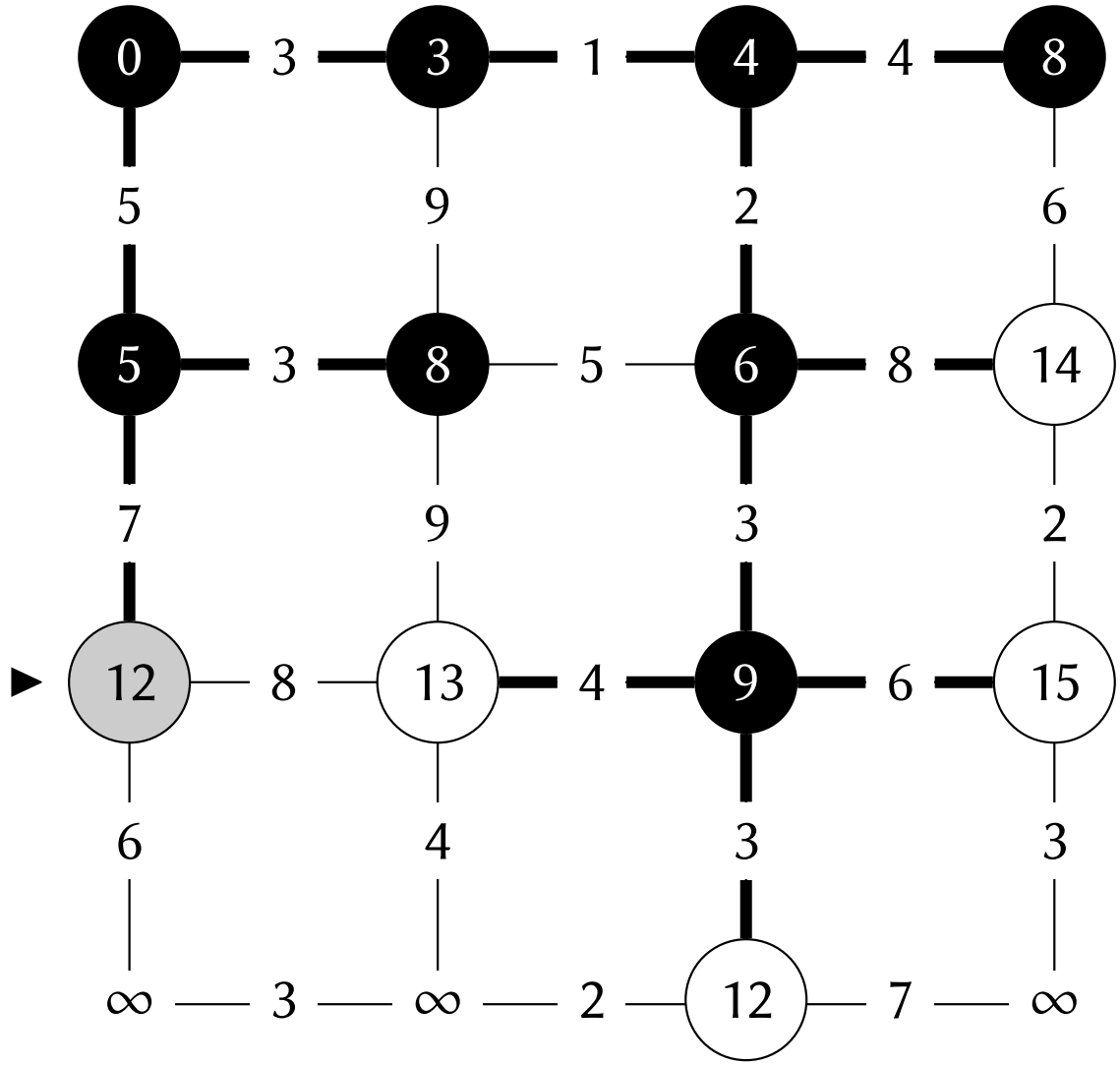
Dijkstra (8.1)



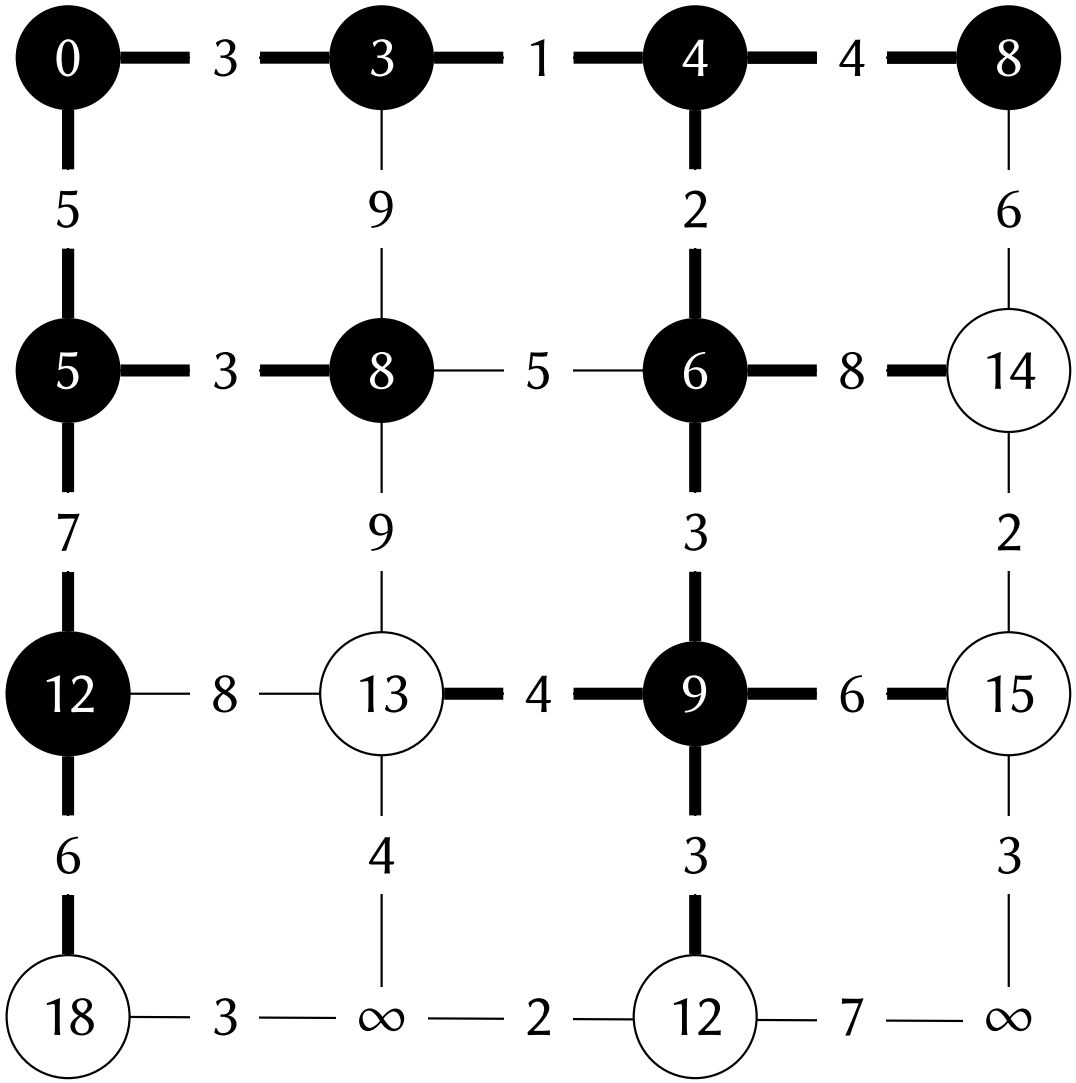
Dijkstra (8.2)



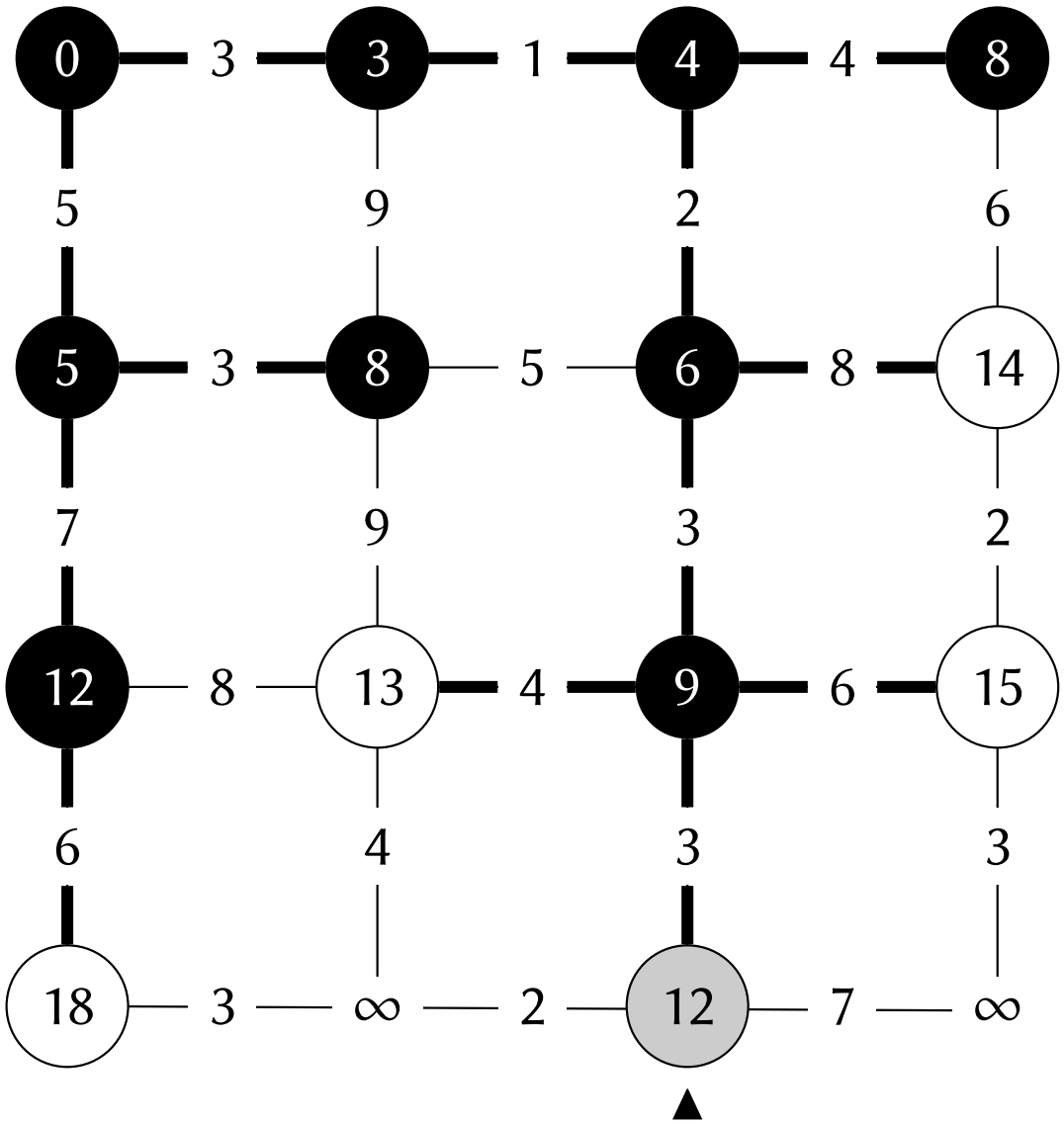
Dijkstra (9.1)



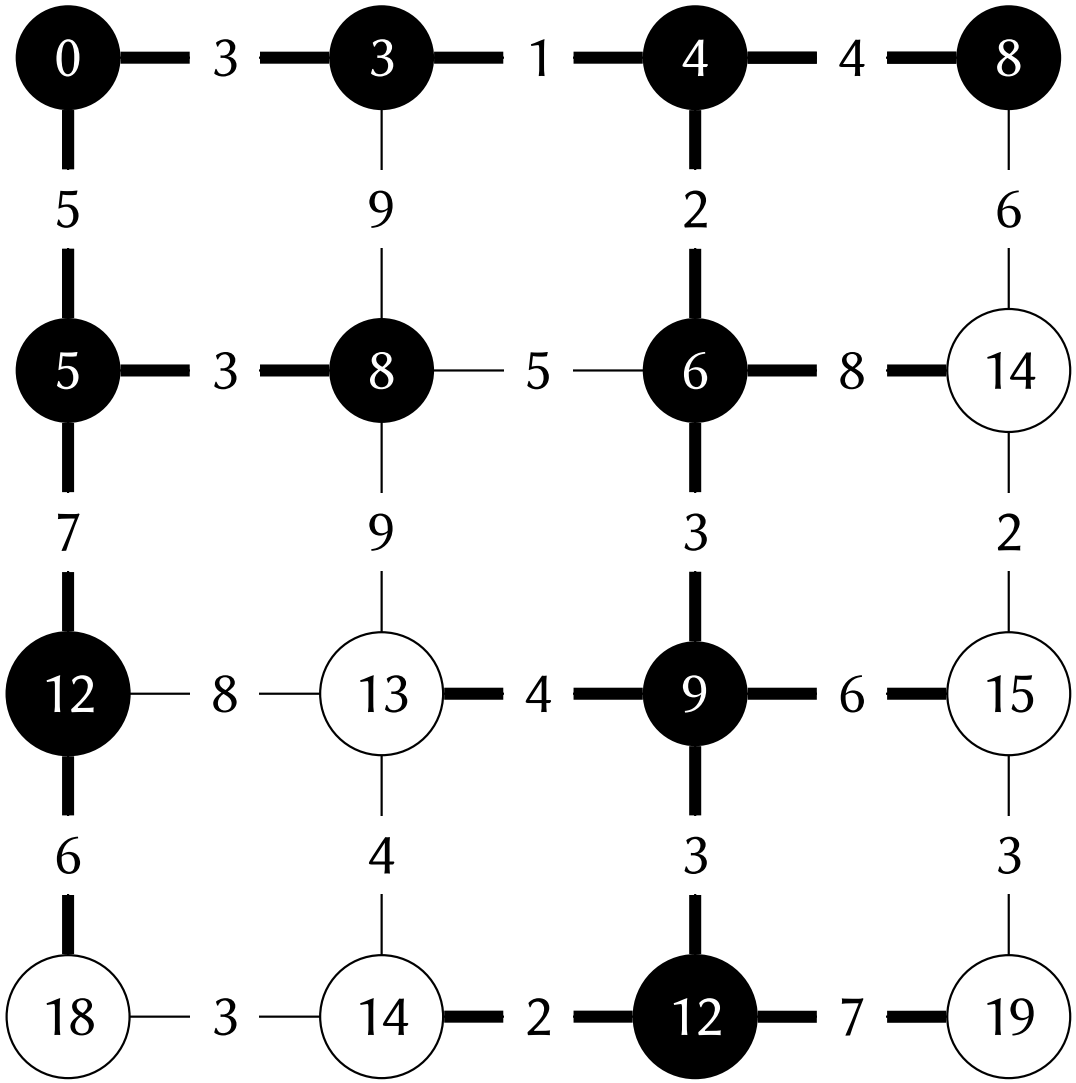
Dijkstra (9.2)



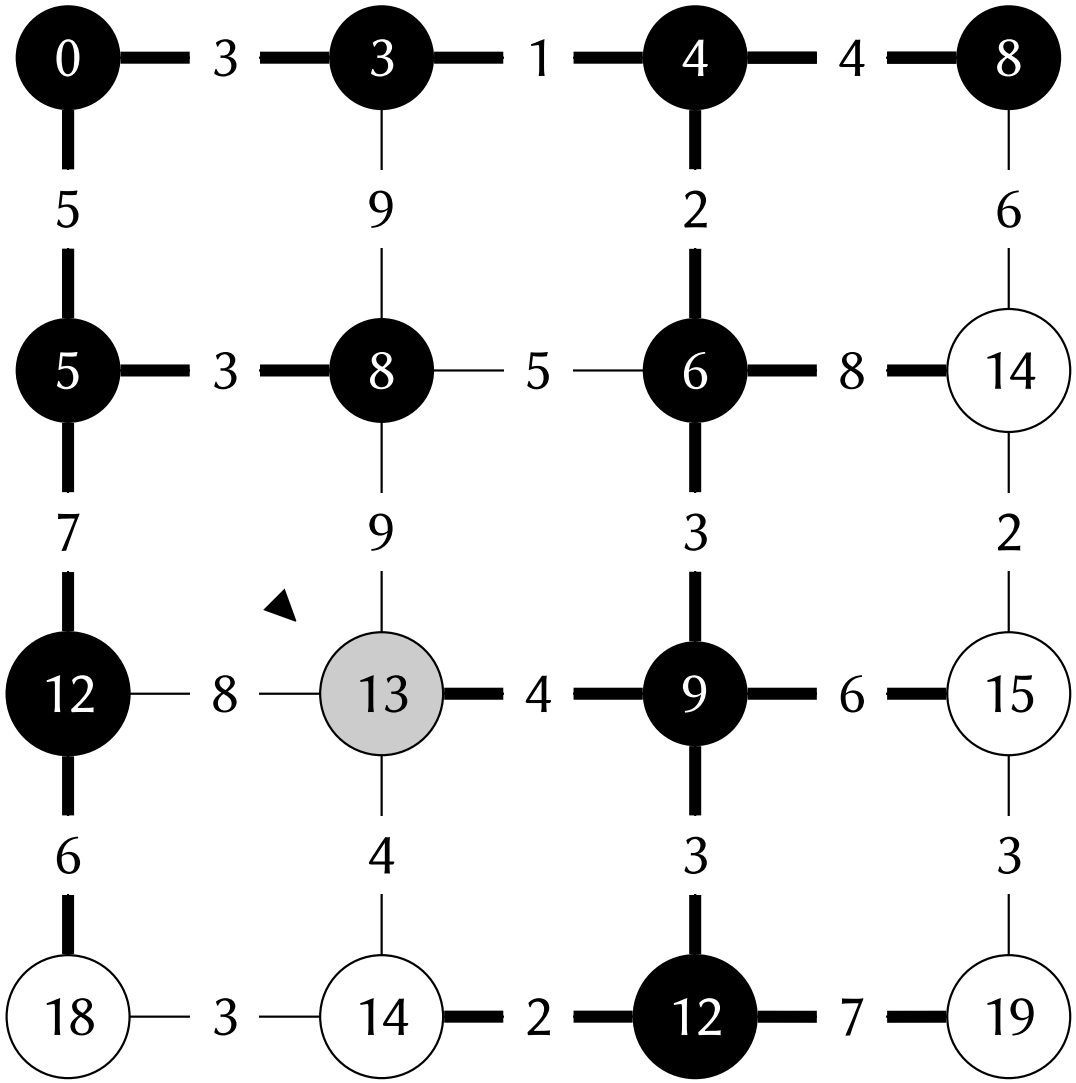
Dijkstra (10.1)



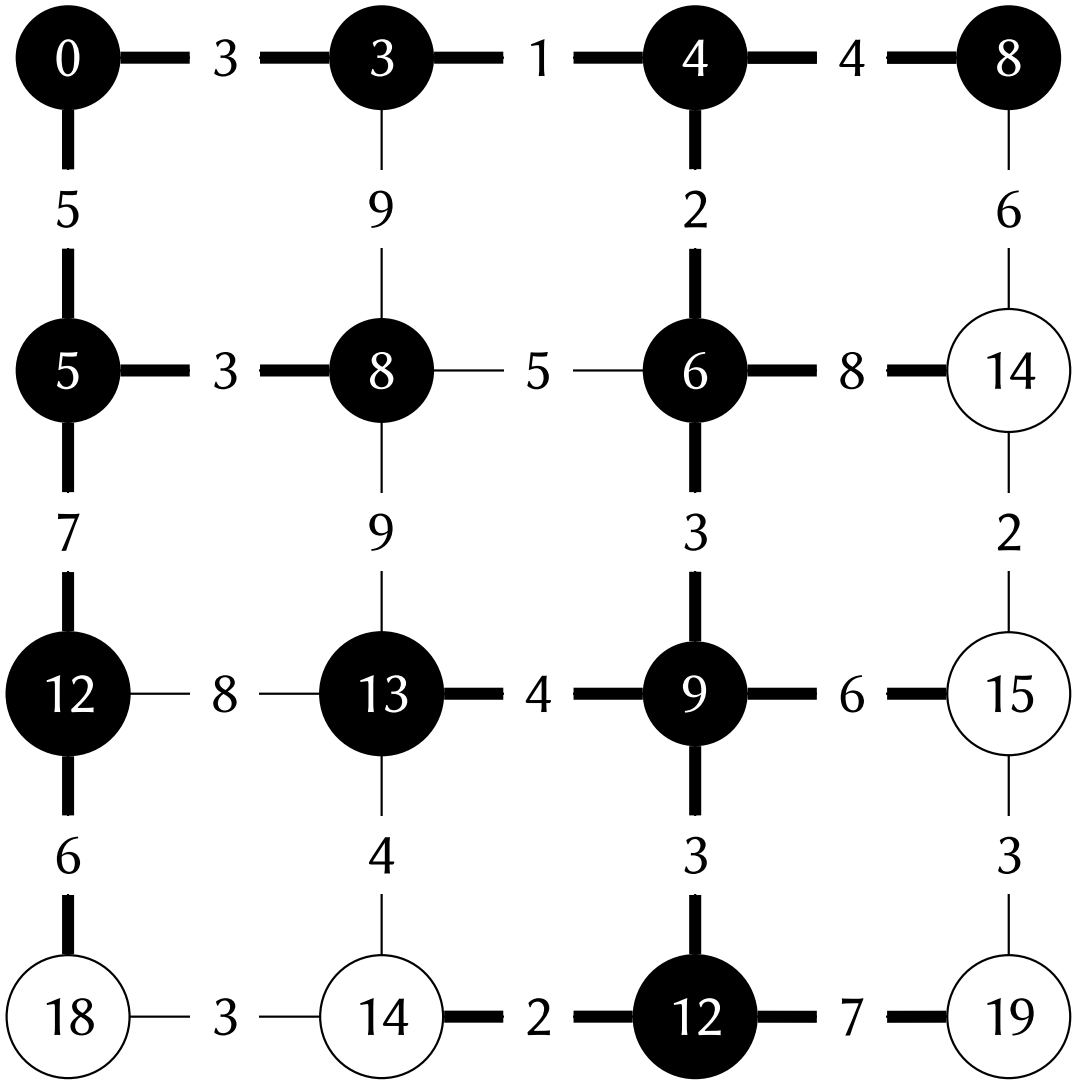
Dijkstra (10.2)



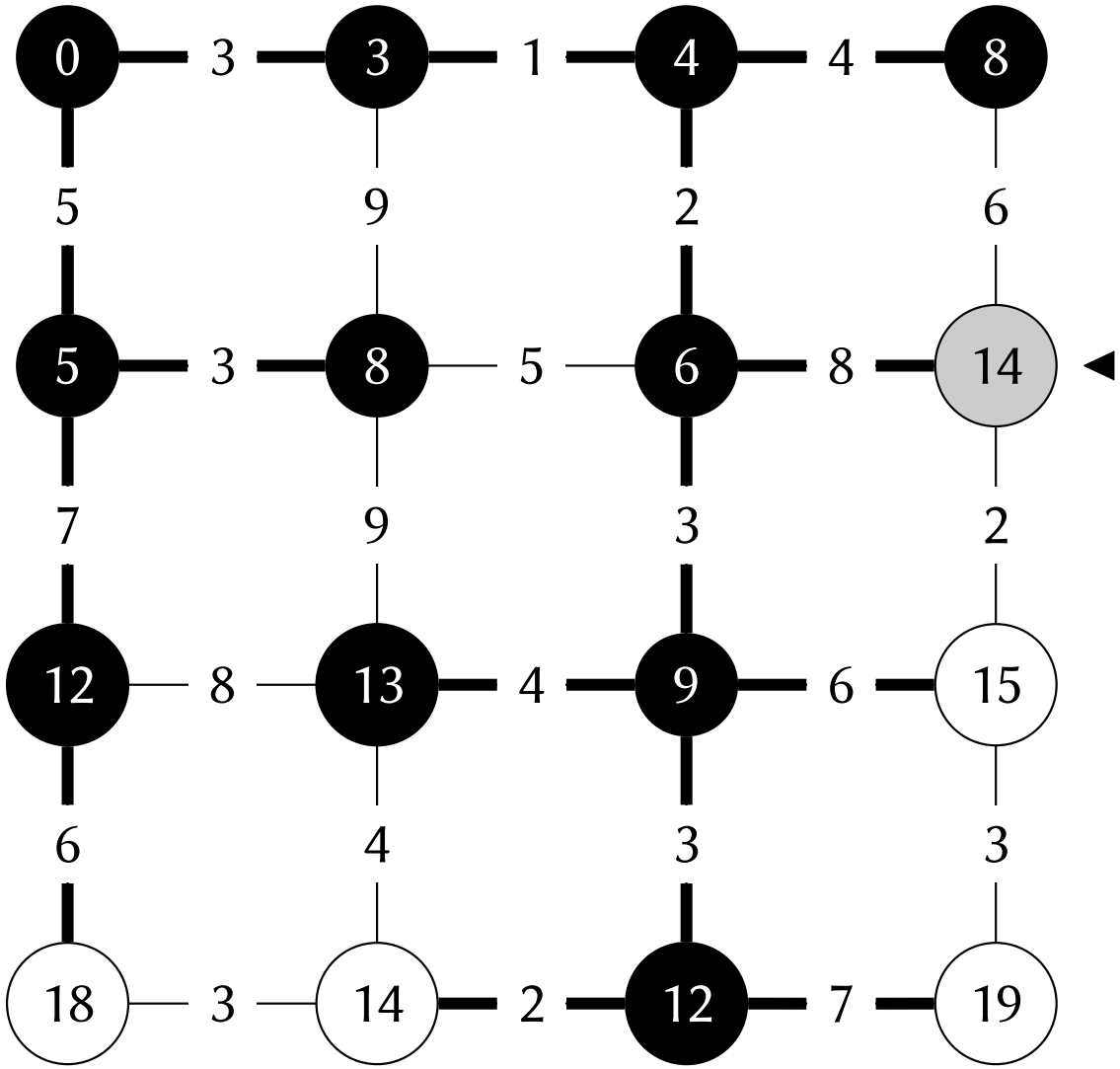
Dijkstra (11.1)



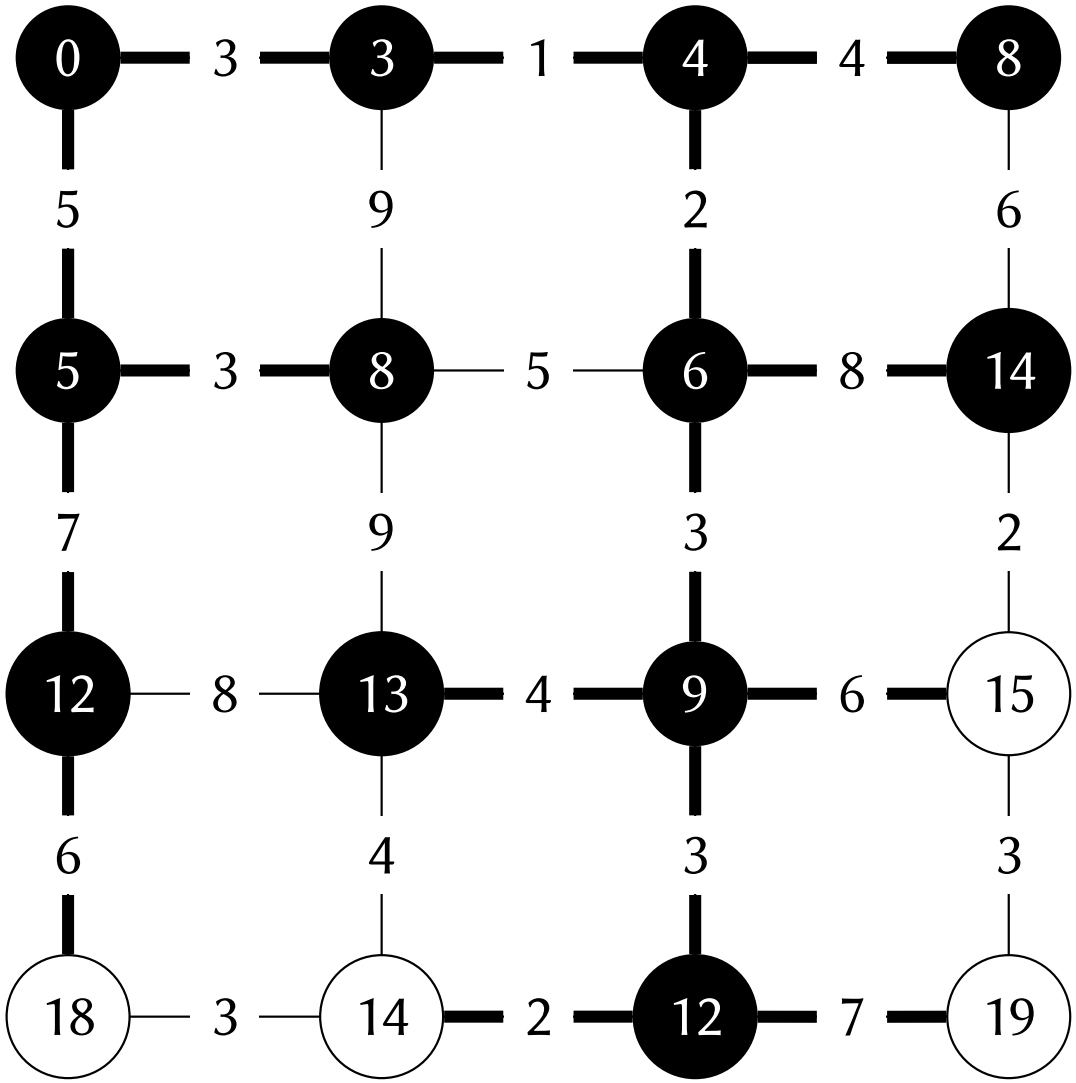
Dijkstra (11.2)



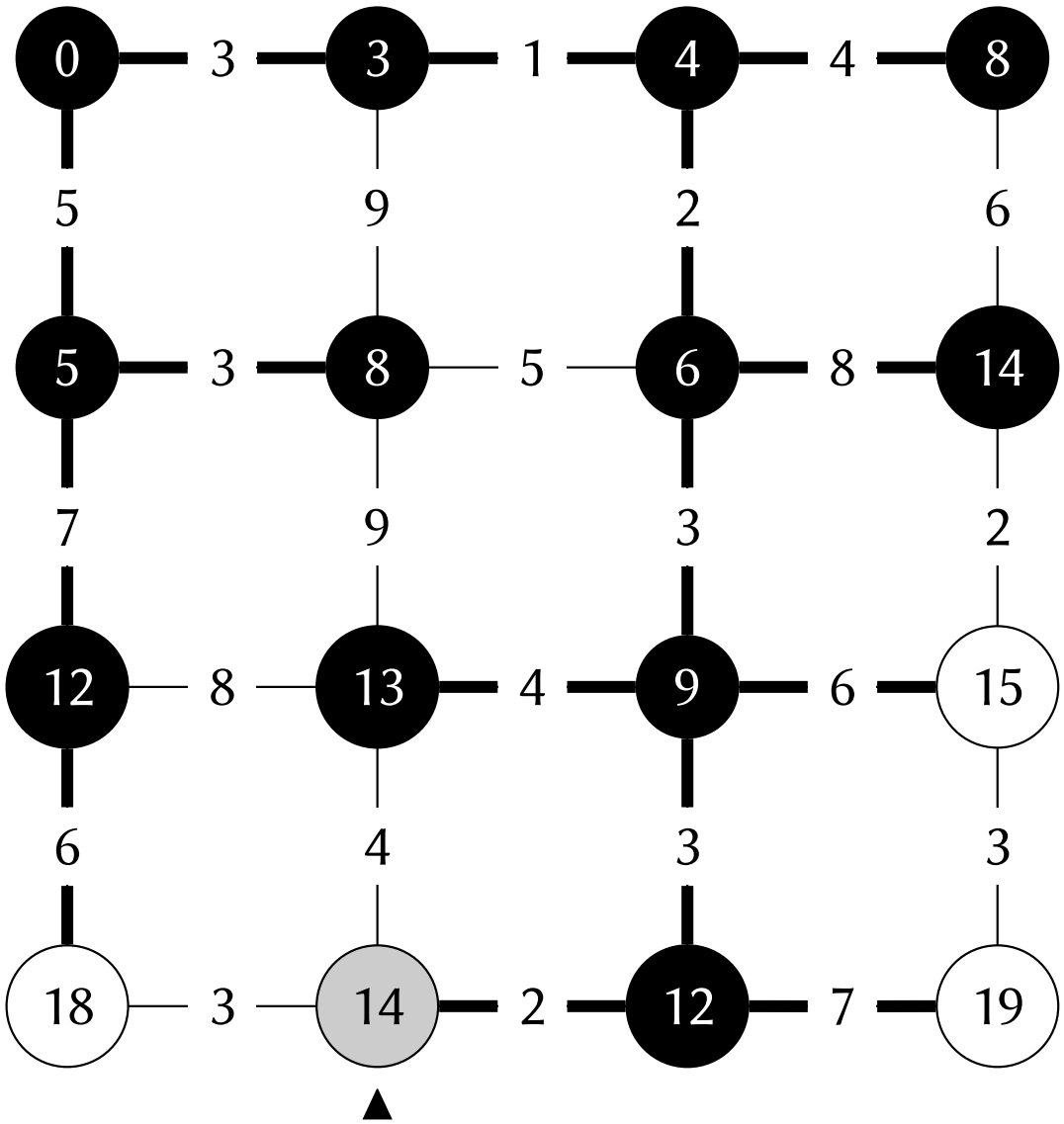
Dijkstra (12.1)



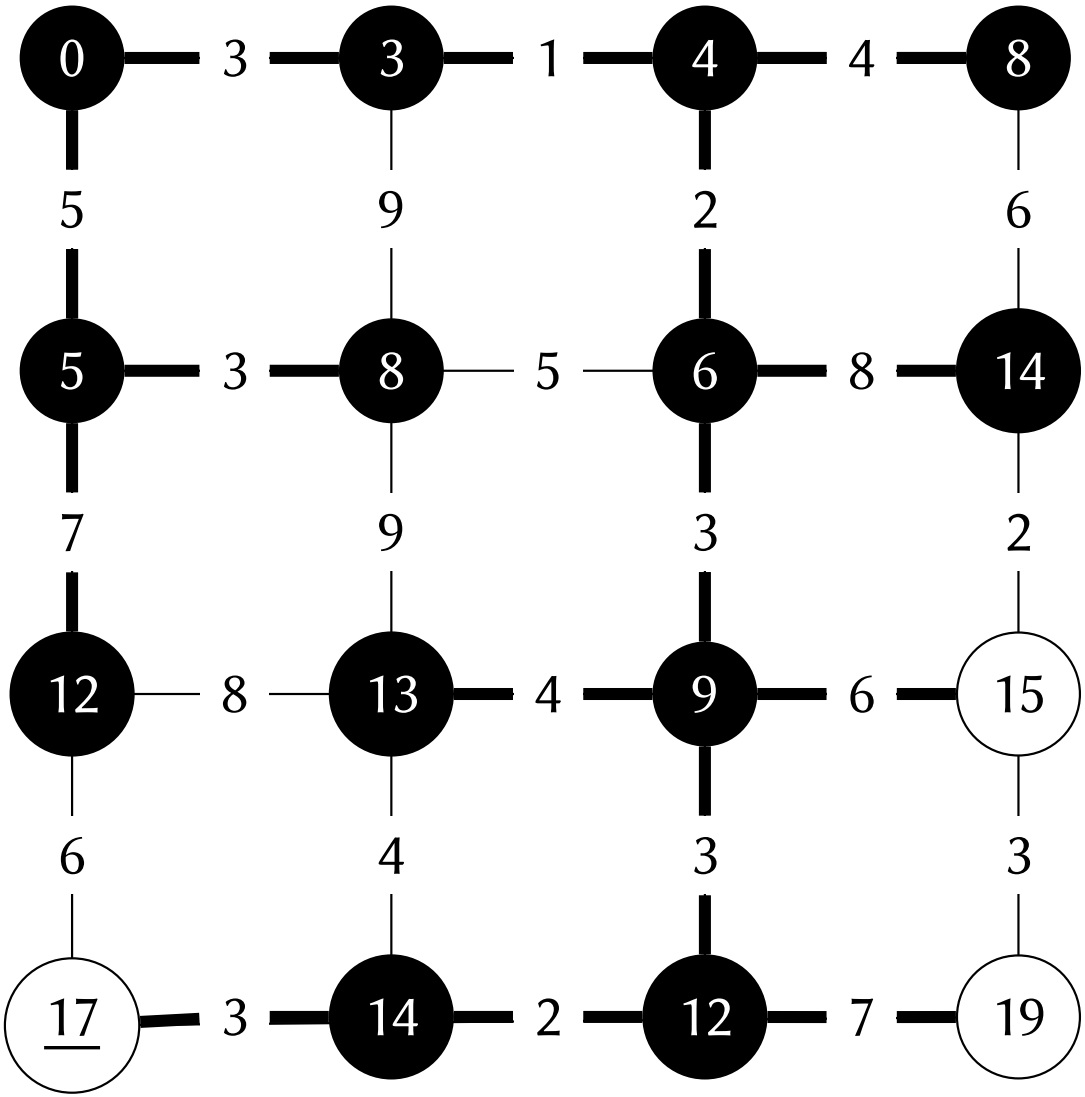
Dijkstra (12.2)



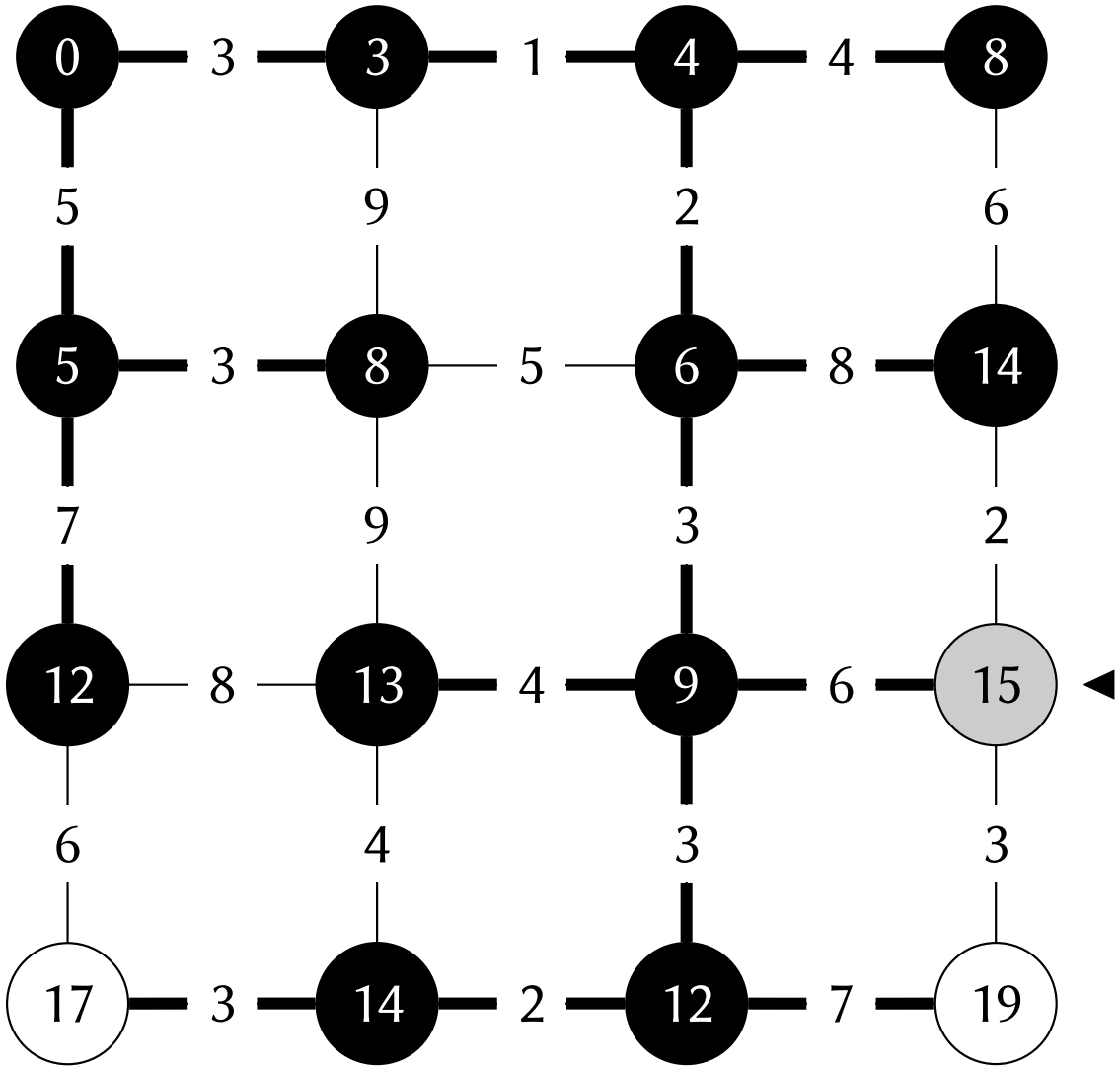
Dijkstra (13.1)



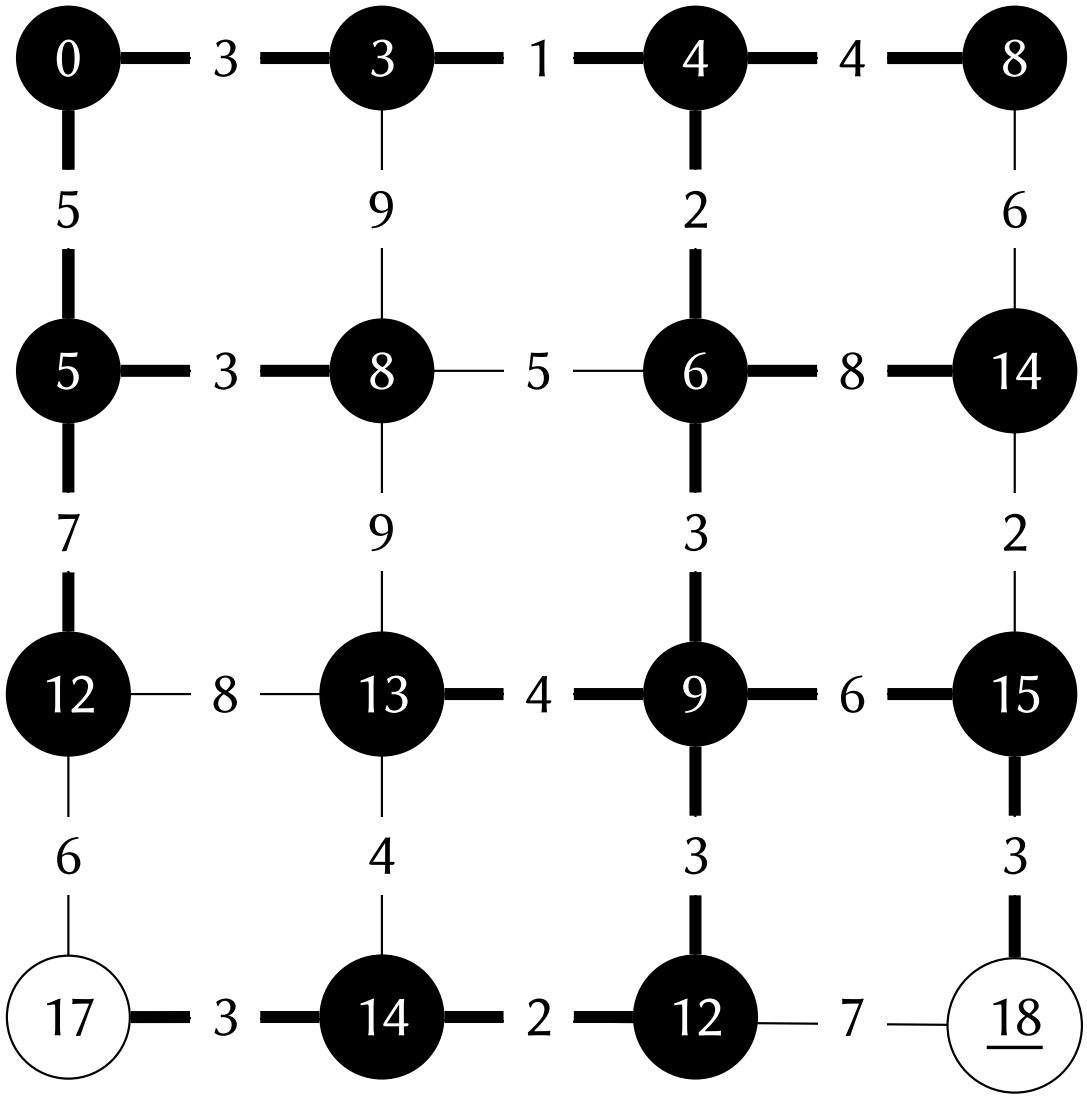
Dijkstra (13.2)



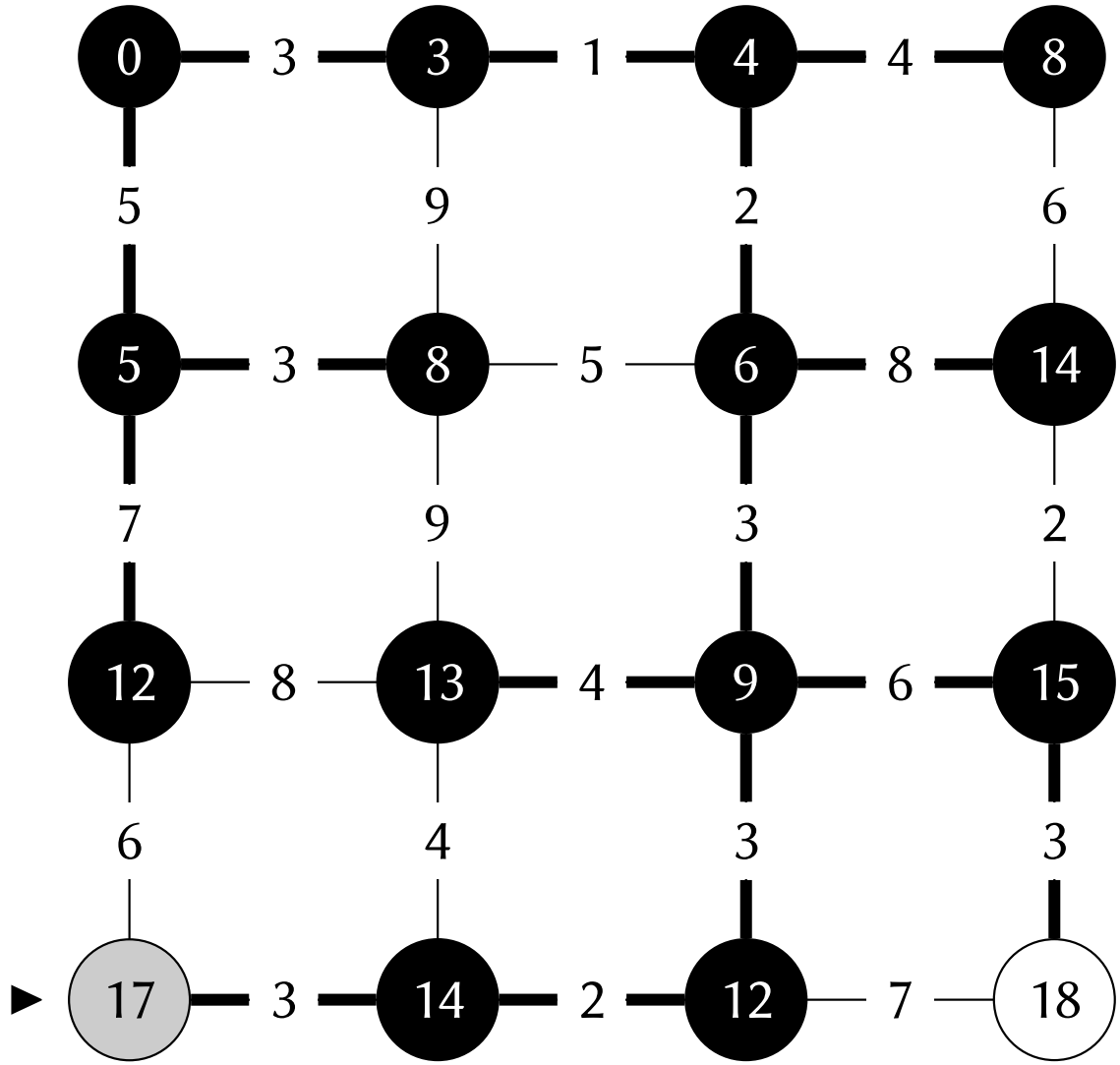
Dijkstra (14.1)



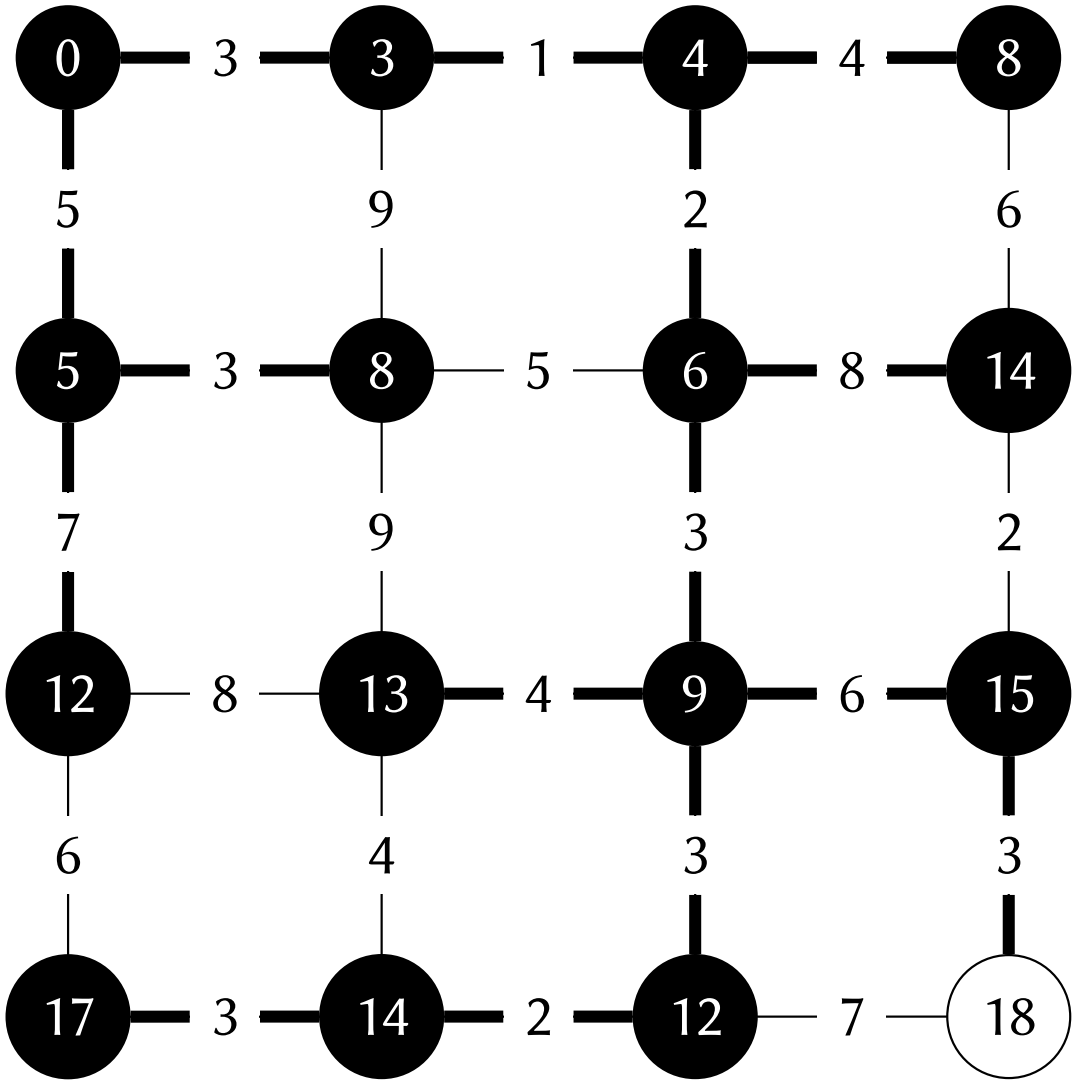
Dijkstra (14.2)



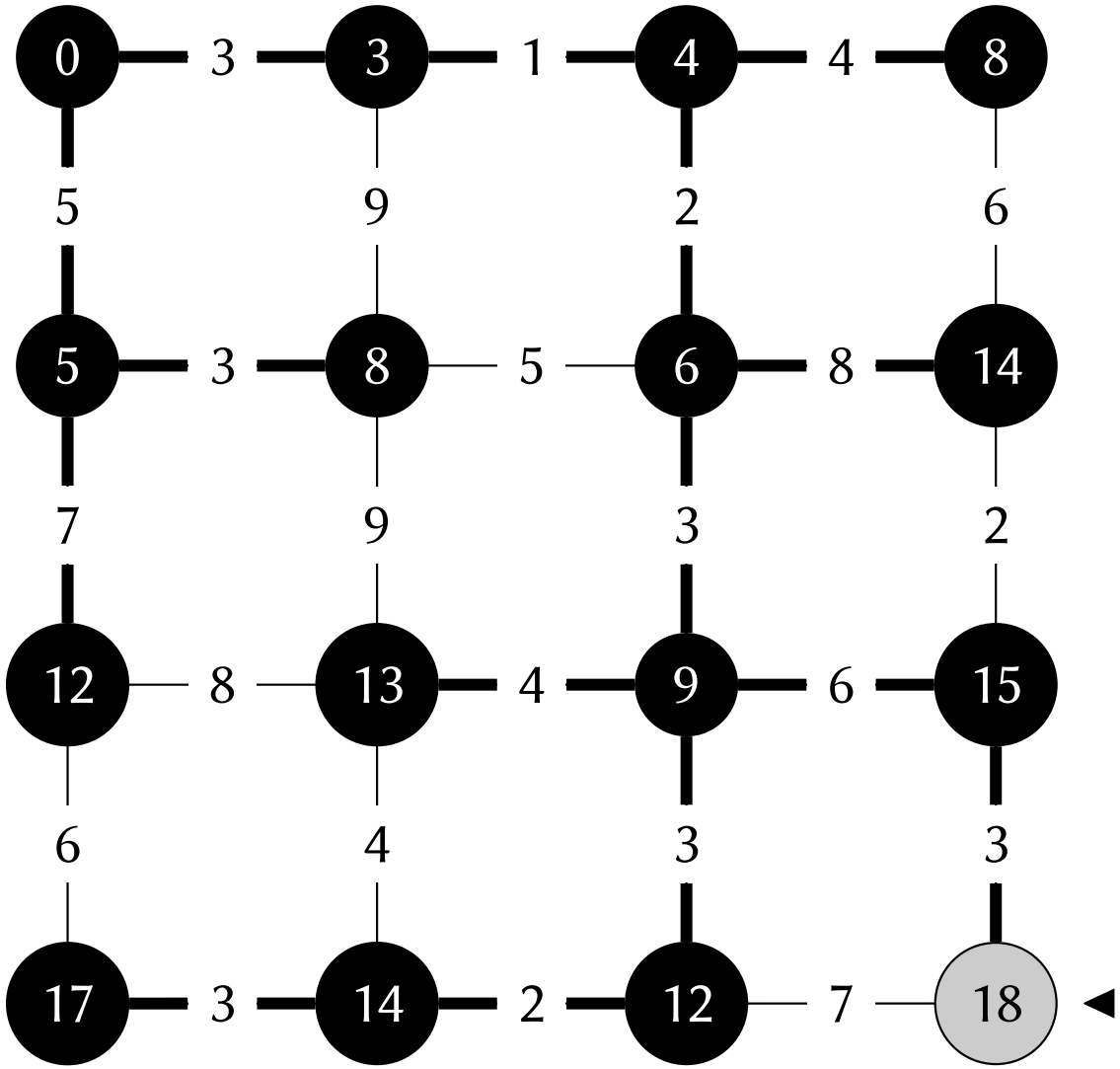
Dijkstra (15.1)



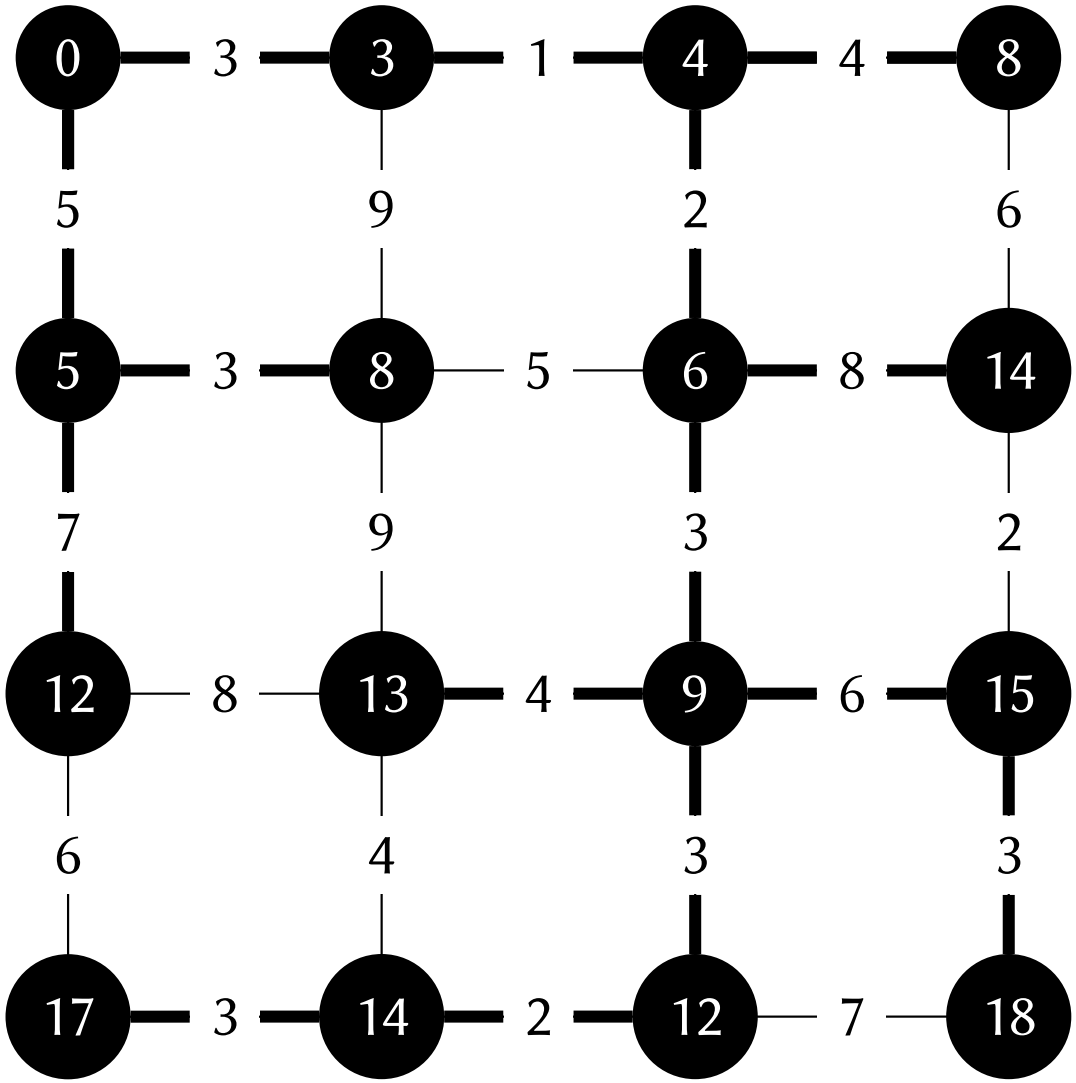
Dijkstra (15.2)



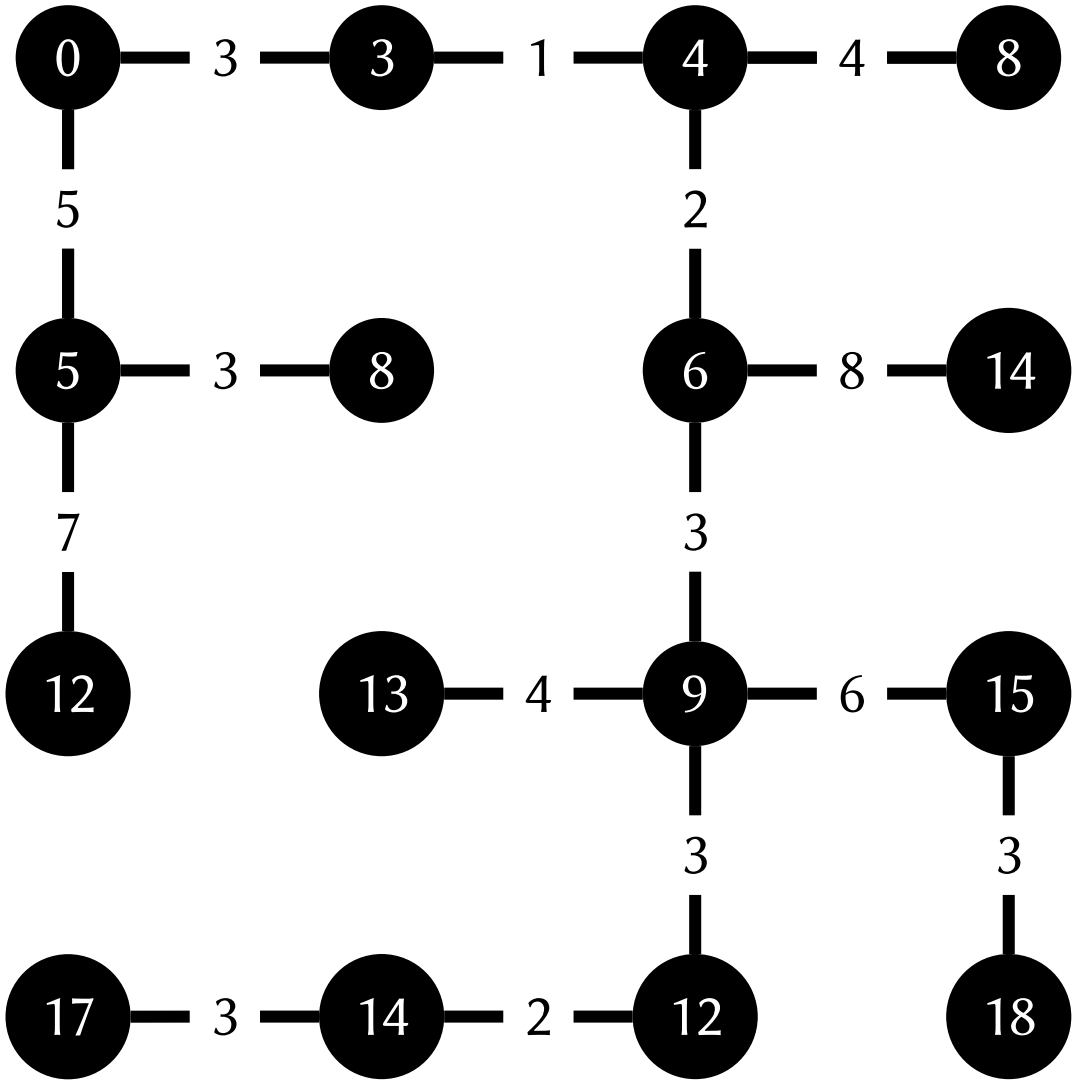
Dijkstra (16.1)



Dijkstra (16.2)



Spanning Tree



Priority Queue Application

To implement the algorithm, we need a priority queue with the following functions:

- `CreatePQ()`, creates a new, empty priority queue.
- `InsertInPQ(`pq, n, d`)` inserts item n with value d in priority queue pq.
- `ExtractMinFromPQ(`pq`)` removes from the queue and returns the node with the minimum distance.
- `SizePQ(`pq`)` returns the number of elements in the priority queue pq.
- `UpdatePQ(`pq, n, d`)` updates the priority queue by changing the stored value associated with n to the value d.

Dijkstra's Algorithm

Algorithm: Dijkstra's algorithm.

`Dijkstra`$(G, s) \rightarrow (pred, dist)$

Input: $G = (V, E)$, a graph
s, the starting node

Output: $pred$, an array of size $|V|$ such that $pred[i]$ is the predecessor of node i in the shortest path from s
$dist$, an array of size $|V|$ such that $dist[i]$ is the length of the shortest path calculated from node s to i

```
1  pred ← CreateArray(|V|)
2  dist ← CreateArray(|V|)
3  pq ← CreatePQ()
4  foreach v in V do
5      pred[v] ← −1
6      if v ≠ s then
7          dist[v] ← ∞
8      else
9          dist[v] ← 0
10     InsertInPQ(pq, v, dist[v])
11 while SizePQ(pq) ≠ 0 do
12     u ← ExtractMinFromPQ(pq)
13     foreach v in AdjacencyList(G, u) do
14         if dist[v] > dist[u] + Weight(G, u, v) then
15             dist[v] ← dist[u] + Weight(G, u, v)
16             pred[v] ← u
17             UpdatePQ(pq, v, dist[v])
18 return (pred, dist)
```

Implementation

- The priority queue can be implemented with an array.
- Then, `InsertInPQ(`pq, n, d`)` is $pq[n] \leftarrow d$, which requires constant time, $O(1)$.
- The function `UpdatePQ(`pq, n, d`)` is also $pq[n] \leftarrow d$ and requires time $O(1)$.
- The function `ExtractMinFromPQ(`pq`)` requires searching through the array, and as this is not ordered, it requires time $O(n)$ for an array with n elements.

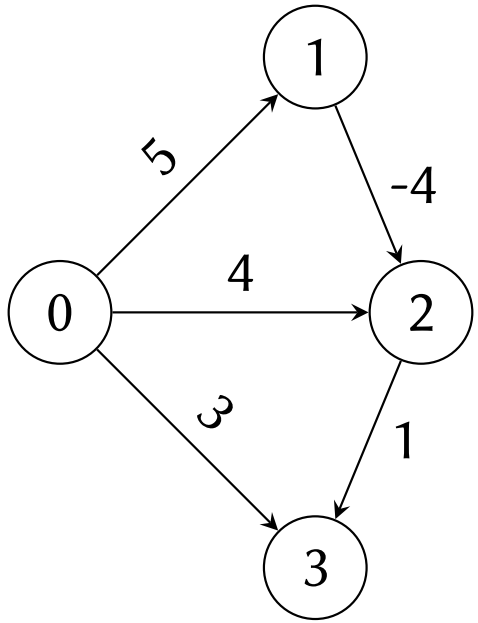
Complexity

- Initialization, in lines 1–10, is executed $|V|$ times.
- All initialization operations require constant time, so the whole initialization requires time $O(|V|)$.
- The algorithm extracts each node as the minimum element exactly once, in line 12, so we have $|V|$ calls of `ExtractMinFromPQ(`pq`)`, each one requiring $O(|V|)$ time, so in total this is $O(|V|^2)$.
- The relaxation sequence in lines 14–17 is performed at most $|E|$ times, once per edge, so we have $|E|$ calls of `UpdatePQ(`pq, n, d`)`, requiring $O(|E|)$ time.
- In total, Dijkstra's algorithm requires $O(|V| + |V|^2 + |E|) = O(|V|^2)$ time.

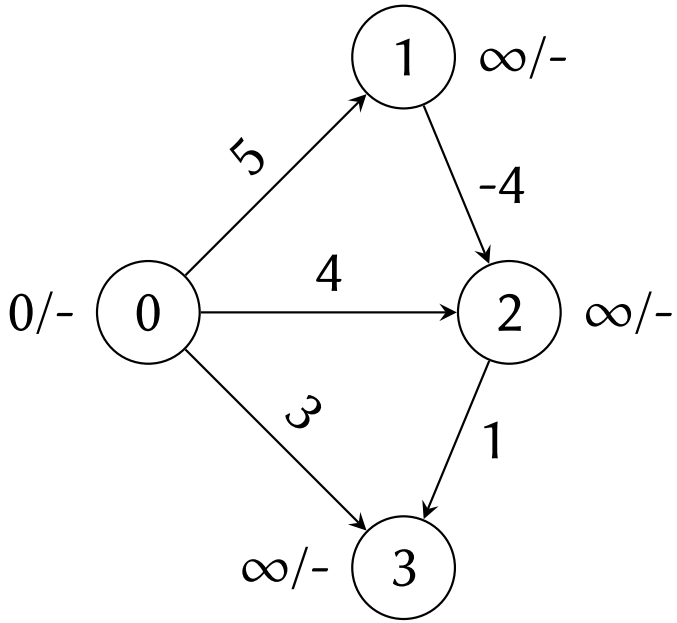
Complexity—Improvements

- Using a more efficient priority queue, we can reduce the time to $O((|V| + |E|) \lg |V|)$.
- If all graph nodes are reachable from the source, then the number of nodes minus the source cannot be more than the number of edges, and therefore the algorithm runs in time $O(|E| \lg |V|)$.

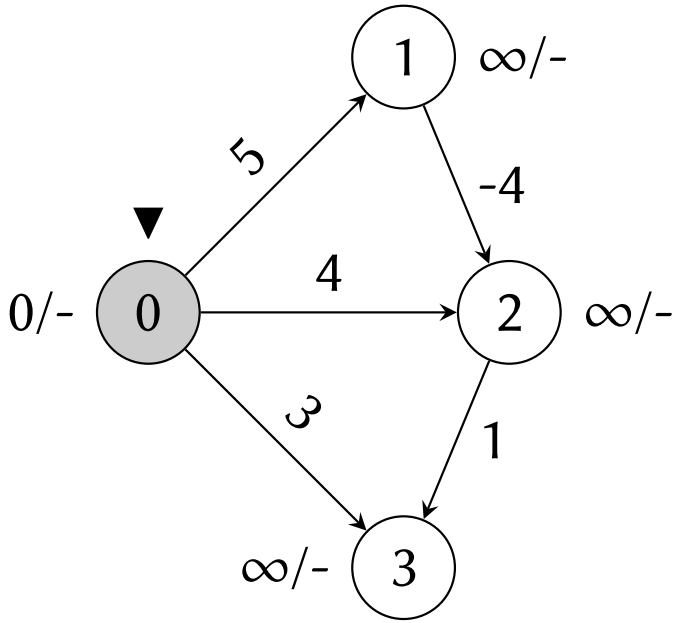
When Does Dijkstra's Not Work?



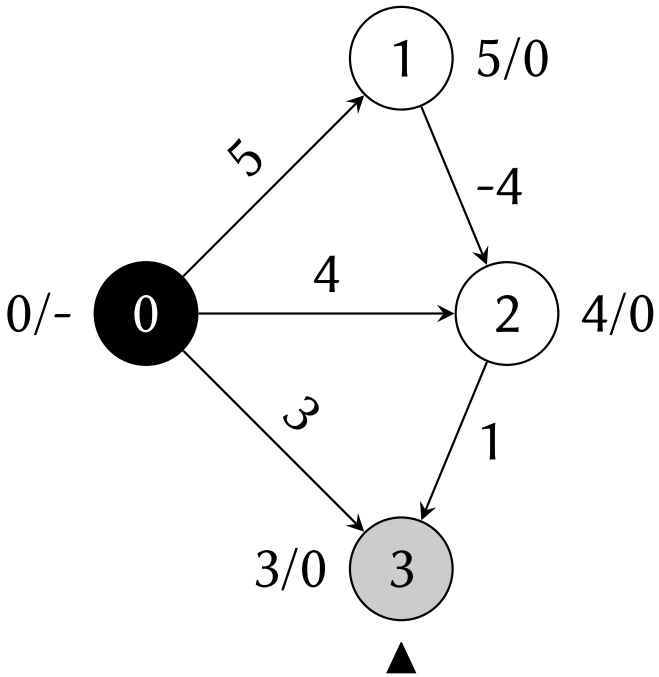
Graph with Negative Weights (1)



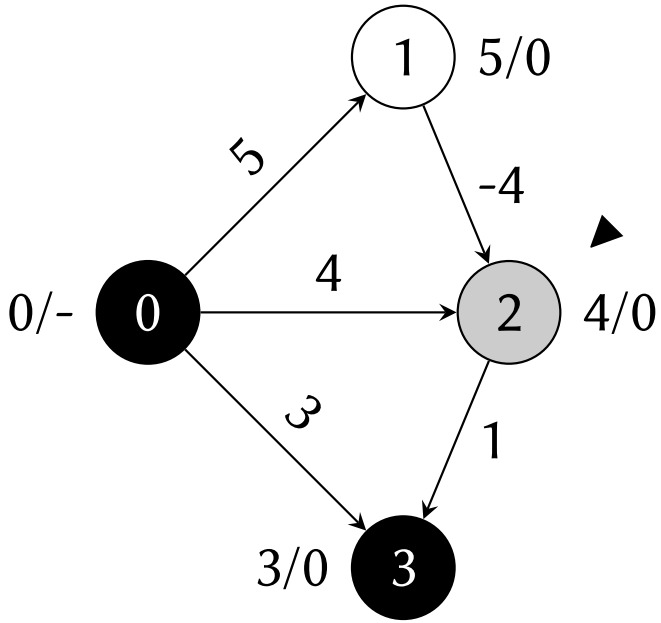
Graph with Negative Weights (2)



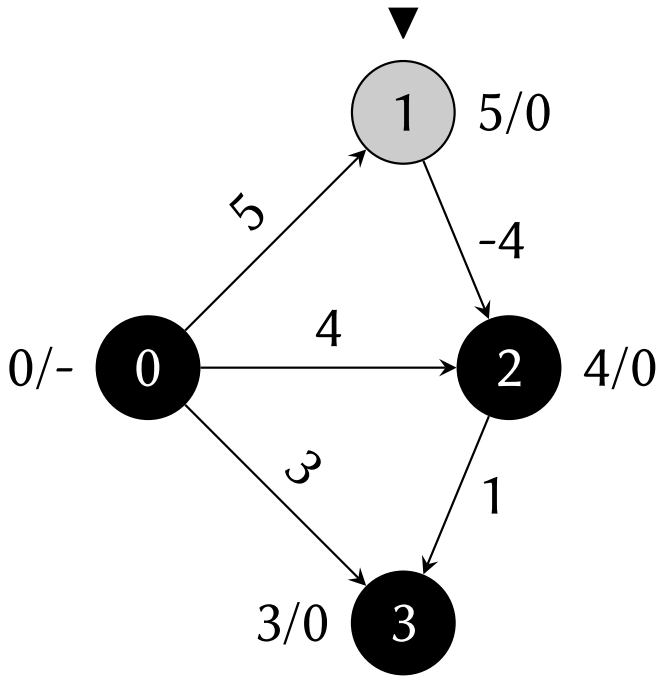
Graph with Negative Weights (3)



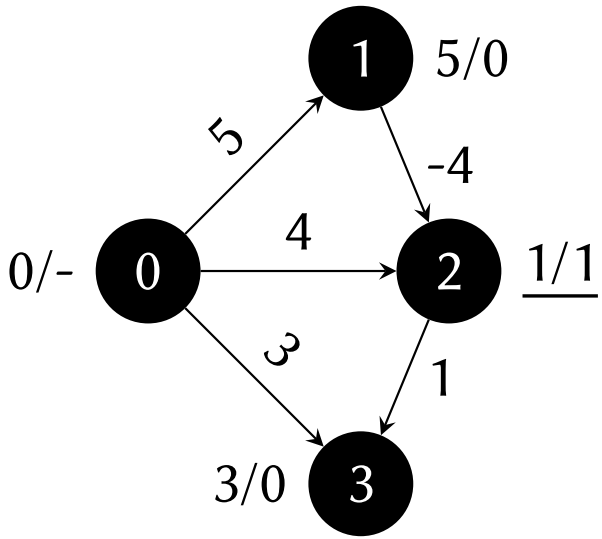
Graph with Negative Weights (4)



Graph with Negative Weights (5)



Graph with Negative Weights (6)



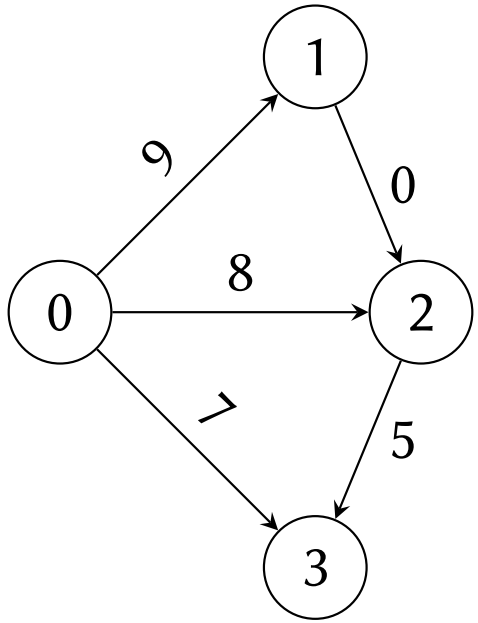
The shortest path from node 0 to node 3 is:

$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3$$

but the algorithm will report the path:

$$0 \longrightarrow 3.$$

Reweighted Graph



Unfortunately, it does not work. The algorithm will still report $0 \longrightarrow 3$ as the shortest path between 0 and 3.

All Pairs Shortest Paths Algorithm

Algorithm: All pairs shortest paths.

`AllPairsShortestPaths`$(G) \rightarrow (preds, dists)$

Input: $G = (V, E)$, a graph

Output: *preds*, an array of size $|V|$ such that $pred[u]$ is the array of predecessors resulting from calling `Dijkstra` for node u
dists, an array of size $|V|$ such that $dist[u]$ is the array of distances resulting from calling `Dijkstra` for node u

```
1  preds ← CreateArray(|V|)
2  dists ← CreateArray(|V|)
3  foreach u in V do
4      (preds[u], dists[u]) ← Dijkstra(G, u)
5  return (preds, dists)
```

Graph Diameter

Definition

The *diameter of a graph* is the length of the longest shortest path between two nodes.

To calculate the diameter of a graph, we need to calculate all pairs shortest paths and find the longest path.