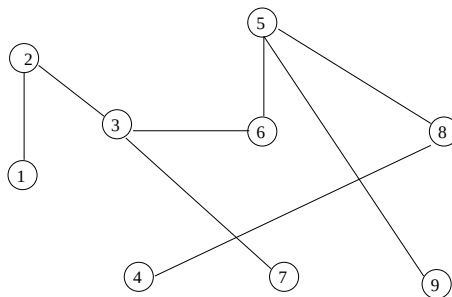


Exercise 3.8

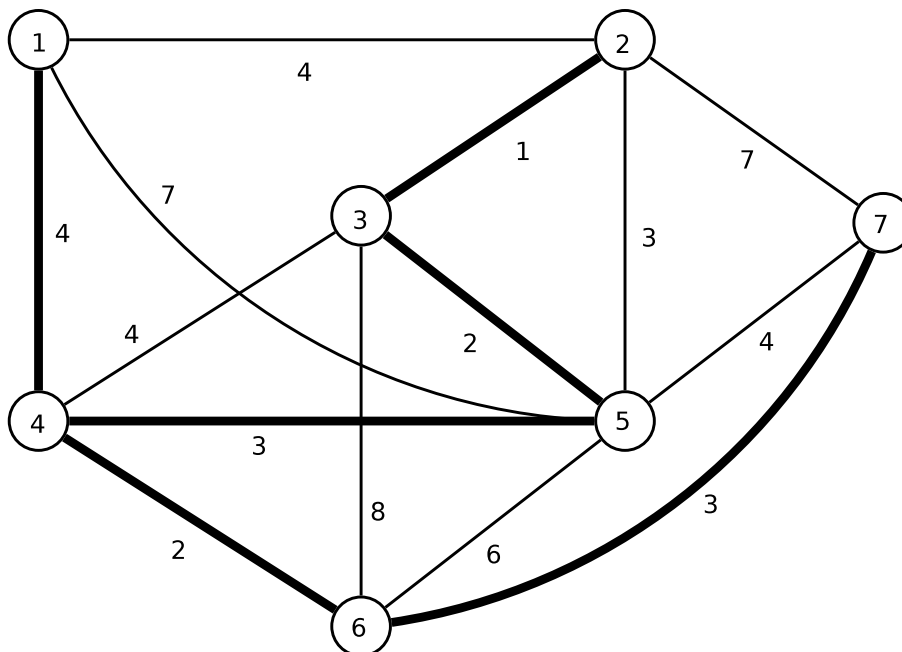
Use the algorithm on page 44 but order the edges in non-decreasing order of length, since we would like to find a minimum weight spanning tree. The edge order used is: $c_{36} \leq c_{23} \leq c_{56} \leq c_{25} \leq c_{12} \leq c_{59} \leq c_{69} \leq c_{58} \leq c_{37} \leq c_{16} \leq c_{79} \leq c_{89} \leq c_{48} \leq c_{13} \leq c_{14} \leq c_{47}$.

We start with $T = \emptyset$ and builds the tree one edge at a time. An edge is added to T if it does not introduce a cycle. The tree shown in the following figure is obtained. It has a value of 34. Alternative minimum spanning trees with the same value exists.



LC3.1: Constructing a MST

There exists several minimum spanning trees in the given graph. One possibility is the one shown below:



The value of an optimal solution is 15..

LC 3.2: An Alternative Algorithm for MST

This proof can be constructed in a way similar to the proof of the correctness of the selection rule.

Given a set H . Assume that is before an iteration contains the MST T . Now we need to prove that after an iteration H still contains an MST.

Let e be the edge selected in the current iteration. Two cases now emerge:

1. $e \notin T$. In this case we can delete e without any problems.
2. $e \in T$. Now we want to generate a new MST T' contained in the “new” H to replace T . This can be done with the following “cut and paste” argument.

We have identified $e = (i, j)$. As the graph is still connected after removing e there must exist an alternative path P from i to j . As $e \in T$ removing e it T is partitioned into two parts uniquely defining a cut. At least one edge from P has to cross the cut as i and j are on opposite sides of the cut. Let this edge be f . As e was the “heaviest” edge we have $w_f \leq w_e$. So by constructing $T' = T \setminus \{e\} \cup \{f\}$ we get $w_{T'} \leq w_T$. As T is a MST we have $w_T \leq w_{T'}$ so $w_{T'} = w_T$ and therefore T' is a MST. As T' is contained in the “new” H , that is, $H \setminus \{e\}$, we have proven the correctness of the algorithm.

LC 3.3: The MiniMax Spanning Tree Problem

Assume T is an MST but is *not* an MMST. Thus a tree S exists, which does not include the longest edge $e \in T$, i.e. $e \notin S$. Taking out e gives two disconnected components. By assumption T is not an MMST, and hence an alternative edge f which is shorter than e , reconnecting the two components, must exist. But this tree will be shorter than T contradicting the fact that T is an MST.

The following figure gives an example that an MMST is not necessarily an MST. The MMST is indicated with bold and is *not* an MST.

