Algorithmic Discrete Mathematics
4. Exercise Sheet

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics
Andreas Paffenholz
Silke Horn

SS 2013
31 May 2013

Due to the holiday on Thursday, this week's exercise will take place on

Friday, 13:30 in S1|03 123.

In addition, the Wednesday group will meet as usual. But participants of the Wednesday group are encouraged to also come on Friday since there will be a short discussion on Prim's algorithm in the beginning.

Groupwork

Exercise G1

Prim's algorithm is another algorithm to compute a minimal spanning tree. It starts with some root node r and then successively builds up a tree by adding a minimal edge connected to the partial tree constructed so far.

The algorithm will be presented in the exercise on 31 May 2013.

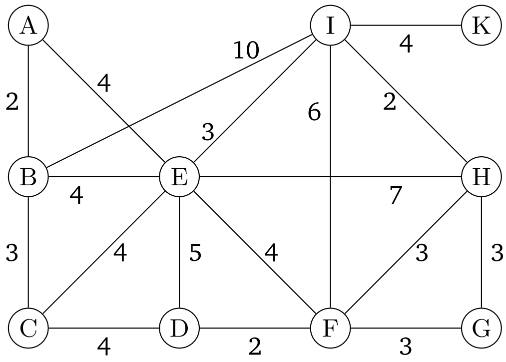
The pseudocode is as follows:

Algorithm 1: Prim's Algorithm

Input: connected graph $G = (V, E)$ given as adjacency list, weight function $w : E \to \mathbb{R}$, root node $r \in V$
Output: minimal spanning tree $T = (V, \{(v, \mathrm{pred}(v)) \mid v \in V \setminus \{r\}\})$ of G

foreach $v \in V$ **do**
    $\mathrm{pred}(v) \leftarrow 0$
    $\mathrm{dist}(v) \leftarrow \infty$ `// distance from tree`
$Q \leftarrow V$ `// priority queue`
$\mathrm{dist}(r) \leftarrow 0$
while $Q \neq \emptyset$ **do**
    $v \leftarrow \mathrm{extract\_min}(Q)$ `// vertex with minimal distance`
    foreach $u \in \mathrm{Adj}(v)$ **do**
        if $u \in Q$ *and* $w(u, v) < \mathrm{dist}(u)$ **then**
            $\mathrm{pred}(u) \leftarrow v$
            $\mathrm{dist}(u) \leftarrow w(u, v)$

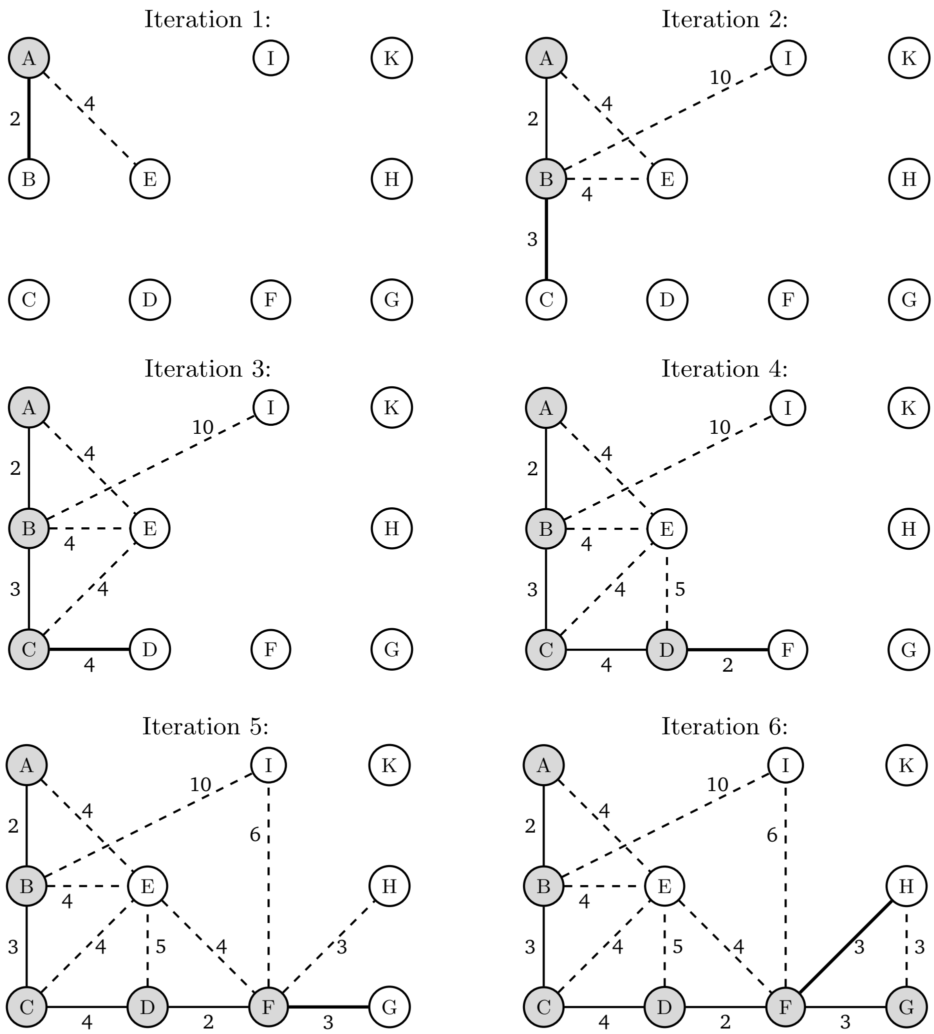
(a) Perform Prim's algorithm on the following graph (starting at node A):

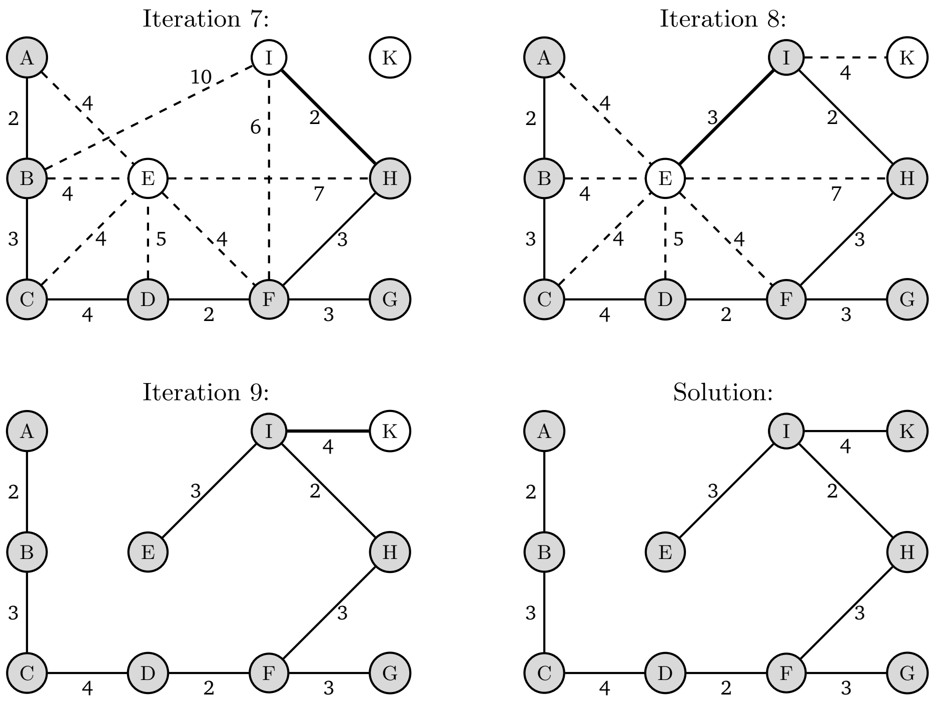


(b) Prove that Prim's Algorithm correctly computes a minimal spanning tree.

Solution:

(a) We start at node A; if there are several edges of the same weight we choose the one with smaller node number.





weight of the spanning tree: 26

(b) We have to show that the output T of Prim's Algorithm is connected, acyclic and minimal:

Connected: T is connected by the same argument we used for Kruskal's Algorithm.

Acyclic: There is no cycle since we always connect a node of the partial tree to a previously isolated node.

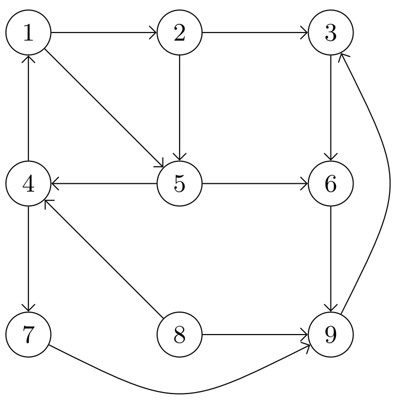
Minimal: Suppose there is a spanning tree T' of smaller weight than T. Let $e = \{u, v\}$ be the first edge of T that is added by Prim's Algorithm and is not contained in T'. Let S denote the set of nodes connected by edges that are added before e. We can without loss of generality assume that $u \in S, v \notin S$. Since T' is a spanning tree, it contains a path from u to v. Let e' be the first edge on this path that connects a vertex in S to a vertex not in S. Now in Prim's Algorithm when e is added we could also have added e'. Since we chose e with $\mathsf{dist}(e)$ minimal, this implies that $w(e) = w(e')$. We can thus replace e by e' to obtain another spanning tree of the weight as T. By iterating this process we see that T has the same weight as T'.

Exercise G2

A directed graph is called *strongly connected* if there is a path from each vertex to every other vertex. It is called *weakly connected* if the underlying undirected graph is connected.

The *strongly connected components (strong components)* of a directed graph are its maximal strongly connected subgraphs.

(a) Determine the strong components of the following graph.



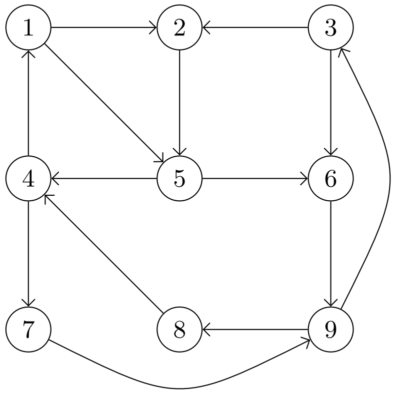
(b) Can you redirect some of the edges so that the graph becomes strongly connected?

(c) Show that a directed graph is *acyclic*, *i.e.,* it does not contain any directed cycle, if and only if all strong components consist of only one vertex. (Note that cycles in directed graphs may consists of only two nodes.)

Solution:

(a) The strong components are $\{8\}, \{1,2,4,5\}, \{7\}, \{3,6,9\}$.

(b) By redirecting the edges $(2,3)$ and $(8,9)$ we obtain:



This is strongly connected.

(c) First assume that there are two nodes u, v in the same strong component. Then there are directed paths p from u to v and p' from v to u. But then (p, p') contains a directed cycle.

Conversely, assume that there is some directed cycle. Then all nodes in the cycle are contained in the same strong component.

Exercise G3

A *bridge* in an undirected graph $G = (V, E)$ is an edge e such that $G - e$ has more connected components than G. Prove that a graph in which every vertex has even degree does not have a bridge.

Solution: *First solution:* If every vertex had even degree, then removing the bridge would create exactly two vertices of odd degree in two different connected components of G. This contradicts the fact that every connected component has an even number of vertices with odd degree.

Second solution: If every vertex has even degree, then the connected component of $e = \{u, v\}$ contains an Eulerian trail. Hence there is another path from u to v.

Exercise G4

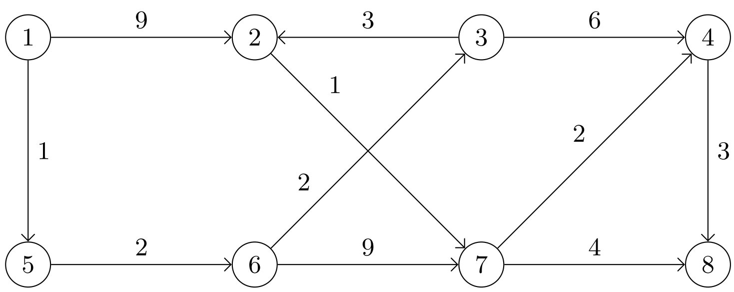
Two graphs are called *isomorphic* ($G \cong H$) if there is a bijective map between their vertex sets that preserves adjacency. Recall that the complementary graph $\overline{G}$ of a graph $G = (V, E)$ is defined as $\overline{G} = (V, \binom{V}{2} \setminus E)$. If $G \cong \overline{G}$ then we say that G is *self-complementary*. Prove: Every self-complementary graph has $4k$ or $4k+1$ vertices for some $k \in \mathbb{N}$.

Solution: A self-complementary graph has $\frac{1}{2}\binom{n}{2} = \frac{n(n-1)}{4}$ edges. This should be integral.

Homework

Exercise H1 (5 points)

(a) Consider the following graph:



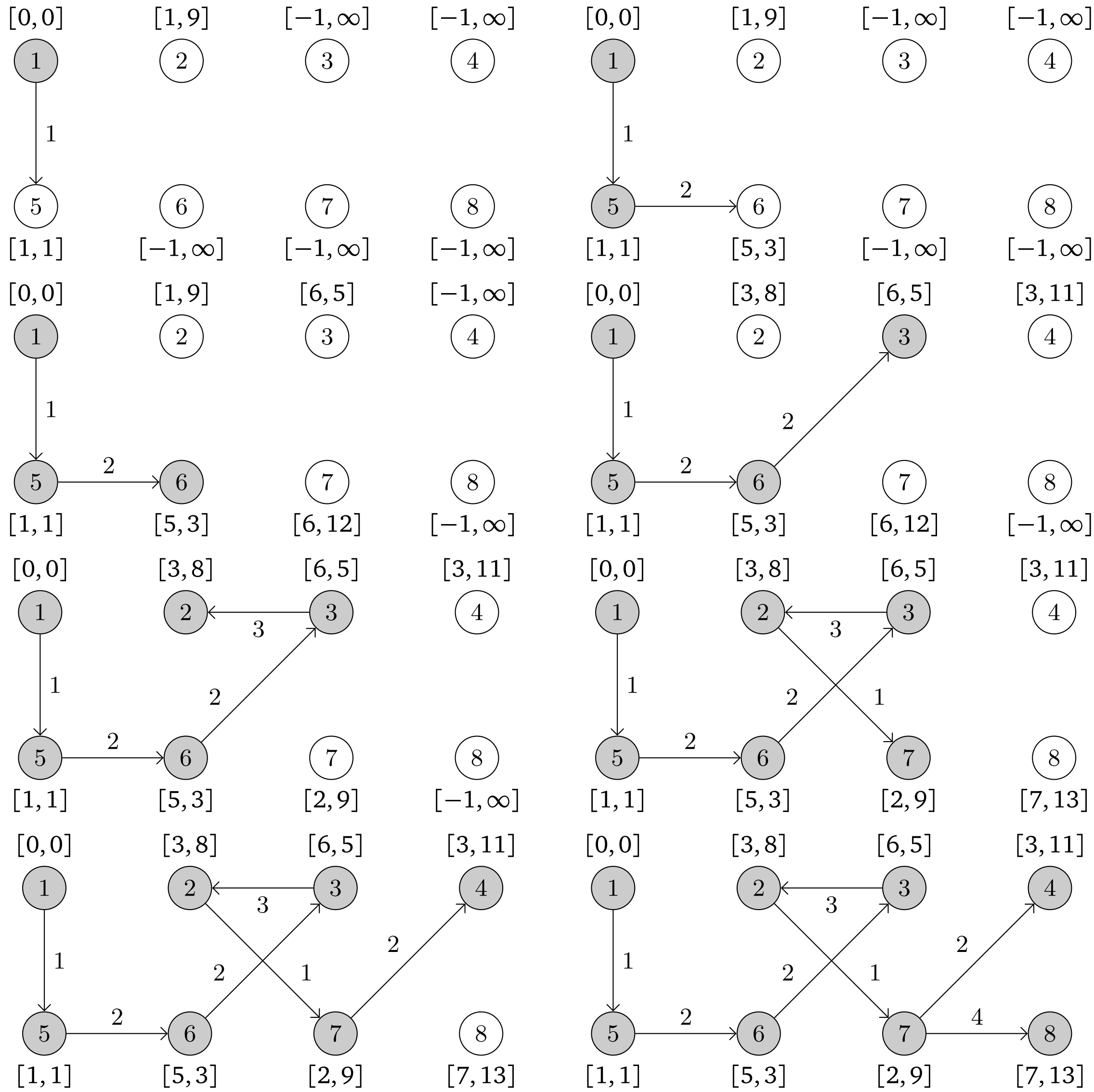
Using Dijkstra's Algorithm compute a shortest path from $r = 1$ to all other nodes and determine the shortest path tree.

(b) Is the shortest path tree for any given graph and root node unique?

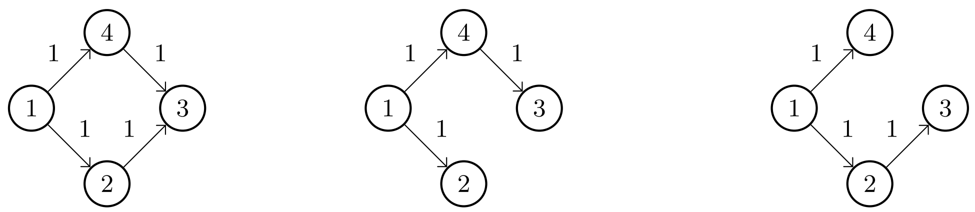
(c) Show by an example that Dijsktra's Algorithm does not work correctly for negative edge weights.

Solution:

(a) The labels are $[\text{pred}(v), d(v)]$. The vertices in Q are gray.

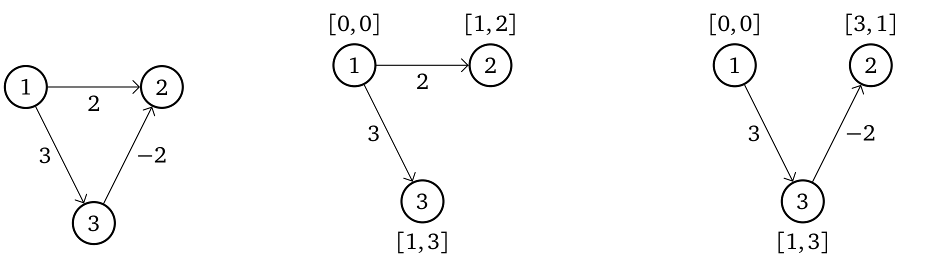


(b) No, in general the tree is not unique; consider the following counter-example:



graph — one shortest path tree — another shortest path tree

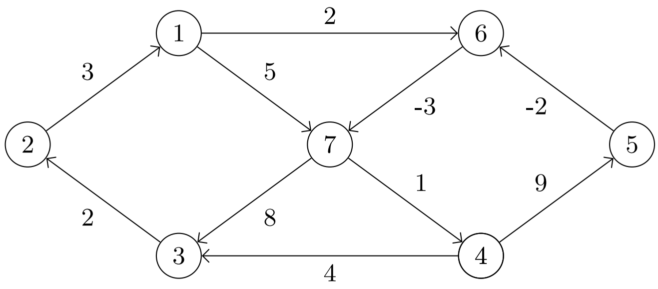
(c) This is a counter-example:



graph $G = (V, E)$ — output of Dijkstra — correct shortest path tree

Exercise H2 (5 points)

(a) Consider the following graph:



Using the Algorithm of Bellman-Ford compute a shortest path from $r = 1$ to all other nodes. In each step specify the order in which you process the edges. Also draw the shortest path tree.

(b) Is the shortest path tree of a graph a minimal spanning tree (of the underlying undirected graph)?

Solution:

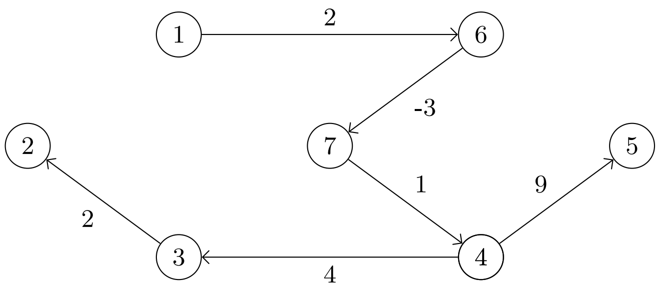
(a) We always go through the edges in lexicographical order.

The steps of the algorithm are as follows:

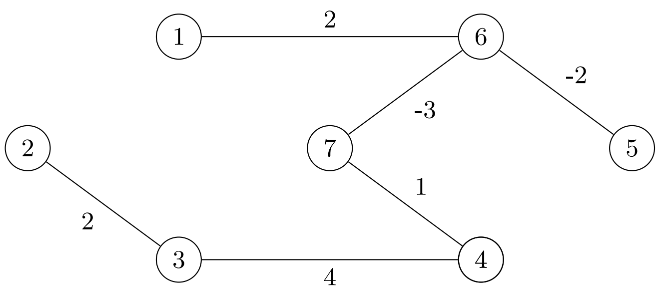
i	1	2	3	4	5	6	7
Init	$(0,0)$	$(\infty,0)$	$(\infty,0)$	$(\infty,0)$	$(\infty,0)$	$(\infty,0)$	$(\infty,0)$
1						$(2,1)$	$(5,1)$
2			$(7,7)$	$(0,7)$			$(-1,6)$
3		$(9,3)$	$(4,4)$		$(9,4)$		
4		$(6,4)$					

After that there will be no more changes.

The shortest path tree looks as follows:



(b) No, in this case a minimal spanning tree could look as follows:



This has smaller weight.

Exercise H3 (5 points)

Let $G = (V, E)$ be a connected undirected graph and $T = (V, E(T))$ a spanning tree of G. A *swap* is a pair (e, f) of edges with $e \in E(T), f \notin E(T)$ such that $T' = T - e + f$ is a spanning tree.

(a) Can any spanning tree of G be transformed into any other spanning tree via a finite sequence of swaps?

(b) What is the maximal number of swaps needed for this?

Solution:

(a) Yes, see (b).

(b) We need at most $|V|-1$ swaps. To this end, let T, T' be two spanning trees with edges $E(T), E(T')$. Let $E_1 = E(T) \setminus E(T') = \{e_1, \dots, e_k\}$ and $E_2 = E(T') \setminus E(T) = \{e'_1, \dots, e'_{k'}\}$. Then $|E_1| = |E_2|$; in particular, $k = k'$.
If we add e'_1 to T we create exactly one cycle C in $T + e'_1$. Then there is some edge $e \in C \cap E_1$. (Otherwise C would be contained in T'.) So we obtain a swap (e, e'_1). By iterating this we get a sequence of k swaps that transform T into T'. Since $k \leq |V|-1$, this proves the claim.

Exercise H4 (5 points)

(a) Let G be a graph with $n \geq 2$ vertices and $m > \binom{n-1}{2}$ edges. Show that G is connected.

(b) Show that (up to isomorphisms) there is only one disconnected graph with n vertices and $\binom{n-1}{2}$ edges.

Solution:

(a) A graph with two connected components of sizes $a, b \geq 1$ with $a + b = n$ has at least

$$ab = a(n-a)$$

missing edges. A graph with $m > \binom{n-1}{2}$ edges misses

$$\binom{n}{2} - m < \binom{n}{2} - \binom{n-1}{2} = n-1.$$

But $a(n-a) < n-1$ only holds for $a < 1$ or $a > n-1$.

(b) With the argument in (a) we see that such a graph has to consist of a complete graph on $n-1$ edges and one isolated node.