University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Schneider

Algorithms and Data Structures
Winter Term 2019/2020
Exercise Sheet 9

Remark: For this exercise, the material of the 13^{th} *video lecture is relevant.*

Exercise 1: "Reverse" Connected Components

(a) Let $G = (V, E)$ be a directed graph with n nodes and m edges given as *adjacency list*. Let $v \in V$ be a node. Give an algorithm with runtime $\mathcal{O}(n+m)$ that computes the set $U = \{u \in V : \exists \text{ Path from } u \text{ to } v\}$, i.e., all nodes u for which a path from u to v exists.

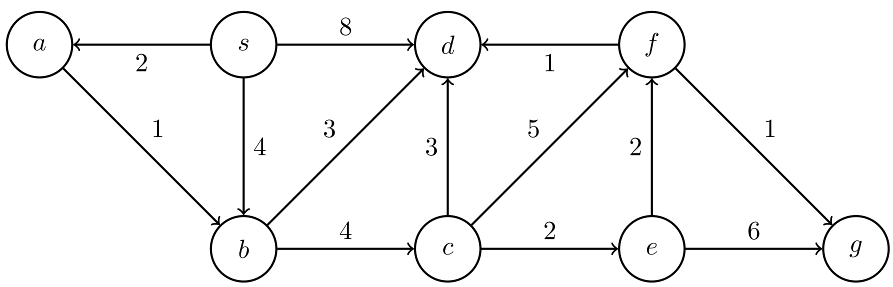
(b) Analyze the running time and argue the correctness of your algorithm.

Exercise 2: Priority Queue with Decrease Key Operation

A heap data structure offers a simple implementation of the functionality of a priority queue. We already know that we can insert elements with keys (i.e. priorities) into a binary tree and then call `heapify` to make a valid heap out of it. We can also insert elements individually using the `insert` operation. Furthermore, we can get the element with the highest priority (that is, the one with the smallest key) with the `delete-min` operation.
For Dijkstras' algorithm, we also require an operation `decrease-key`(p, k) which gets a *pointer* p to directly access an element in the binary tree, and a key k to which the key of that element is lowered, provided that it is not already lower and subsequently restores the heap condition. Give pseudocode that implements `decrease-key`(p, k) in $\mathcal{O}(\log n)$ time if n is the number of elements in the heap.

Exercise 3: Dijkstras' Algorithm

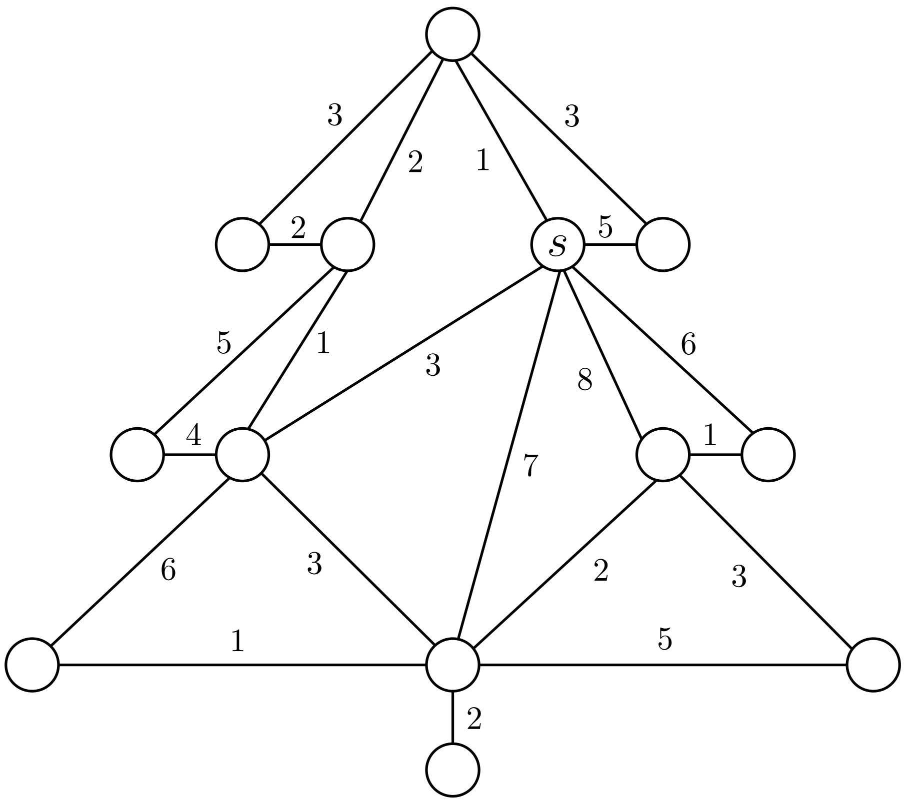
Execute Dijkstras' Algorithm on the following weighted, directed graph, starting at node s. Into the table further below, write the distances from each node to s that the algorithm stores in the priority queue after each iteration.



Initialization	s	a	b	c	d	e	f	g
$\delta(s,\cdot) =$	0	∞	∞	∞	∞	∞	∞	∞
1. Step $(u = s)$	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
2. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
3. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
4. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
5. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
6. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
7. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								
8. Step ($\boldsymbol{u =}$)	s	a	b	c	d	e	f	g
$\boldsymbol{\delta(s,\cdot) =}$								

Exercise 4: More of Dijkstras' Algorithm

In the following graph execute Dijkstras' Algorithm starting from node s. Write the distance of each node into the respective node. Mark the order in which nodes are settled by the algorithm and mark all edges belonging to the *shortest path tree*.



Enjoy the holidays and have a happy new year!