Department for computer science 1
Prof. Dr. Gerhard Woeginger
Janosch Fuchs

SS 2020
15.07.2020

Exam
Algorithms and Data Structures

Name: ....

First name: ....

Matriculation number: ....

Course of studies: ....

Note:

- You have 90 minutes for the exam.
- Please write your name and matriculation number on each sheet.
- Please write clearly. Illegible parts are not corrected and rated as incorrect.
- Cross out concept calculations that should not be counted or make them otherwise identifiable. If several attempts are made to solve a problem, the worst is scored.
- Please use a document-proof pen with blue or black ink and do not use an ink killer or similar. Use only the paper provided.
- Please turn off your electronic devices!

I declare that I have completed the exam myself and I am aware that the exam will be rated as "failed" if attempted to deceive.

....
(Signature)

Exercise	1	2	3	4	Gesamt
Points	20	20	20	30	90
Scored					

(Name/Matriculation number)

Exercise 1:

(a) Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be two functions. State the (mathematical) definition of "$f(n) \in \Omega(g(n))$". **(2 Points)**

(b) The function $T : \mathbb{N} \to \mathbb{N}$ satisfies $T(n) = n$ for $n \leq 100$ and **(4 Points)**

$$T(n) = 64\,T(n/4) + 5n^3$$

for $n \geq 101$. What does the **Master theorem** tell us about the asymptotic behavior of $T(n)$?

(c) Perform an analysis of the Quicksort algorithm. Answer the following questions: **(7 Points)**

- Explain the Divide-step in the QuickSort algorithm.

- Explain the Conquer-step in the QuickSort algorithm.

- State (without further argument) the time complexity of the Divide-step on an array with n integers.

(Name/Matriculation number)

(d) Consider a set of n integers, each with $b = 1000$ bits. Explain how `Radix-Sort` with parameter $r = 10$ sorts these integers. State (without further argument) the resulting time complexity as a function of n. **(7 Points)**

(Name/Matriculation number)

Exercise 2:

(a) Explain the term **open addressing** in hashing. **(2 Points)**

(b) Explain the term **linear probing** in hashing. **(2 Points)**

(c) Consider a hash table $T[0 \ldots 10]$ with $m = 11$ entries, and the hash function **(10 Points)**

$$h(k, i) \;=\; (2k \bmod 11 \;+\; i + 7) \bmod 11.$$

Insert the six keys 11, 46, 25, 47, 24, 13 step by step into the initially empy hash table T. Use open addressing and linear probing with the given hash function $h(k, i)$.

T[0]	**T**[1]	**T**[2]	**T**[3]	**T**[4]	**T**[5]	**T**[6]	**T**[7]	**T**[8]	**T**[9]	**T**[10]

(d) Explain the problem of **primary clustering** that arises under linear probing. Describe one approach that helps to avoid primary clustering. **(6 Points)**

(Name/Matriculation number)

Exercise 3:

In the **longest common subsequence** problem, we are given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$ and a sequence $Y = \langle y_1, y_2, \ldots, y_n \rangle$. The goal is to find a longest subsequence that occurs in X as well as in Y. (Recall that a subsequence of X results from sequence X by deleting an arbitrary subset of terms.)

(a) Explain the **dynamic programming** algorithm for computing a longest common subsequence of X and Y. **(8 Points)**

(b) Perform a time complexity analysis of the dynamic programming algorithm under (a). **(6 Points)**

- State the resulting time complexity as a function of m and n.

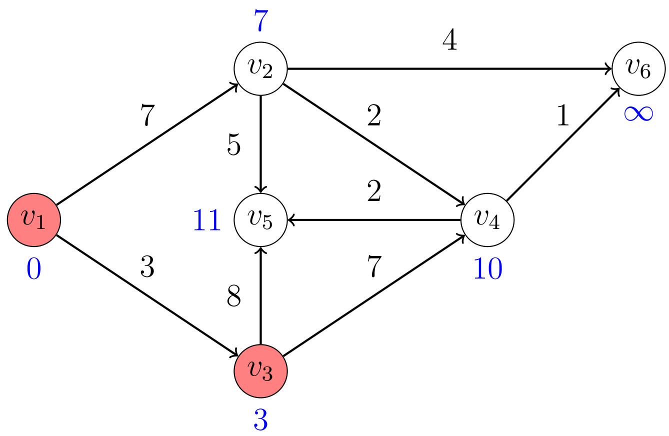
- State the resulring space complexity as a function of m and n.

(Name/Matriculation number)

(c) Construct the dynamic programming table for the sequences $Y = BCDABA$ and $X = ABDBCAB$. State a longest common subsequence of X and Y. **(6 Points)**

(Name/Matriculation number)

Exercise 4:



The figure above shows an intermediate state of the Dijkstra algorithm, starting at the vertex v_1. The red vertices are already visited. The blue numbers next to the vertices indicate the current distance to v_1.

(a) Insert the missing three lines in the following source code snippet: **(4 Points)**

Initialize-Single-Source(G, s);
$S \leftarrow \emptyset$;
$Q \leftarrow V[G]$;
while $\boldsymbol{Q \neq \emptyset}$ **do**

;

;

for each vertex $\boldsymbol{v \in Adj[u]}$ **do**

;

end

end

Algorithm 1: Dijkstra algorithm

(b) Consider the next step of the Dijkstra algorithm in the graph presented above: **(6 Points)**

- Which vertex will be selected and handled ?
- State the resulting distances $d[v]$ for all vertices $v \in V$.

(c) Give the shortest path between v_1 and v_6 at the end of the computation. **(2 Points)**

(Name/Matriculation number)

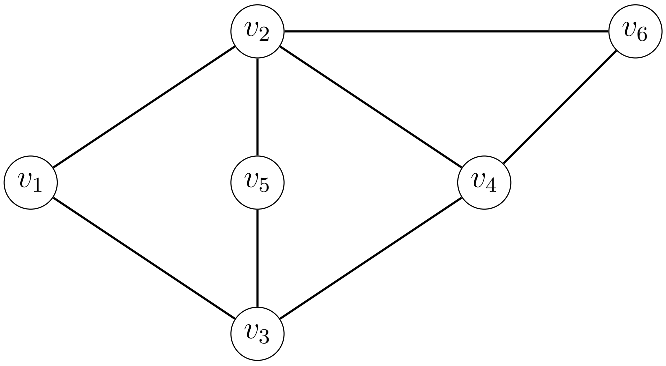
(d) Explain why the Dijkstra algorithm can not properly handle edges with negative weight. Give an example that illustrates this problem. **(4 Points)**

(e) Does the Bellman-Ford algorithm suffer from the same problem with negative edge weights? **(2 Points)**

(f) What is the running time of the Dijkstra algorithm in terms of $|V|$ and $|E|$? **(2 Points)**

(g) Give the proof idea for the time complexity analysis. **(6 Points)**

(Name/Matriculation number)



(h) Draw a Breadth-first search tree, starting at v_6. **(4 Points)**