tentamen datastructuren en algoritmen 2013-2014

dinsdag 22 oktober 2013 15.15-18.00

8 exercises – English version

Exercise 1. (*5+4 points*)
This exercise is concerned with linear datastructures.

(a) We have available two stacks, with operations as specified in the abstract datatype (ADT) for stacks. Use them to implement the operations enqueue and dequeue of the ADT for queues.

Ensure that the operation enqueue is in $\mathcal{O}(1)$ is, and the operations dequeue is on average in $\mathcal{O}(1)$ (no justification needed).

(b) Explain why your operation dequeue from the previous question is in $\mathcal{O}(1)$. Use the accounting method (with the 'saving' idea).

Exercise 2. (*4+4+3 points*)
This exercise is concerned with heaps and heap-sort. In heaps the internal leaves are labelled with an integer, and the external leaves are empty.

(a) Apply the bottom-up heap construction to place the following keys in a max-heap:

4 5 7 8 3 4 1

Give your answer in pictures with all bubble-steps explicit.

(b) Apply heap-sort to the max-heap $[-, 4, 3, 2, 1]$.

Give all steps explicitly using either pictures or the array-representation (your choice).

(c) What is the worst-case time complexity of heap-sort?

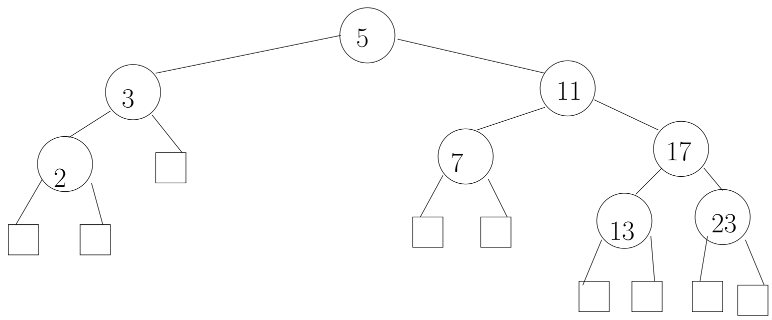
Explain briefly your answer.

Exercise 3. (*3+5+3+4 points*)
This exercise is concerned with binary search trees and AVL-trees. Every node has 0 or 2 children. Internal nodes are labelled with positive integers, and external leaves are empty.

(a) Give all binary search trees with three internal nodes labelled 1, 2, 3.

Indicate which binary search trees are AVL-trees and which are not; indicate for the non-AVL-trees which node(s) is(are) unbalanced.

(b) Given is the following AVL-tree:



Add a node with key 24. Rebalance if necessary, and give all steps explicitly in pictures.

(c) What is the worst-case time complexity for adding a key to (i) a sorted array (ordered dictionary), (ii) a binary search tree, (iii) an AVL-tree? (No motivation needed.)

(d) Give an algorithm (not necessarily in pseudo-code) that takes as input an AVL-tree and gives as output the difference between the largest and the smallest key (i.e. the *range* of the AVL-tree).

Exercise 4. *(4+4+4 points)*
This exercise is concerned with sorting.

(a) Explain the working of bucket sort (no pseudo-code).

(b) Give the merge-sort-tree for sorting the input-sequence $[3, 6, 1, 7, 5, 8, 2, 4]$.

(c) Solve the following recurrent equation for merge-sort:

$$T(n) = \begin{cases} 1 & \text{als } n = 1 \\ 2T(\frac{n}{2}) + n & \text{als } n > 1 \end{cases}$$

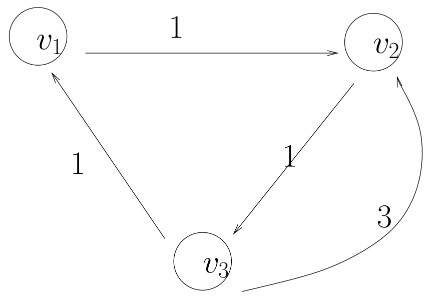
Also, explain how this recurrent equation relates to your merge-sort-tree from the previous exercise.

Exercise 5. *(5+4+4 points)*
This exercise is concerned with trees and graphs. All nodes are labelled.

(a) Give pseudo-code for level-order traversal of a binary tree. (Every node has 0 or 2 children.)

(b) Explain how the algorithm for level-order traversal of a binary tree can be generalised to an algorithm for breadt-first search (BFS) of a graph (which besides the graph also takes a start-node as input).

(c) Apply the dynamic programming algorithm for finding all shortest paths (All-Pair Shortest Paths in the book) to the following graph:



Exercise 6. *(4+4 points)*
This exercise is concerned with dynamic programming.

(a) Give a dynamic programming algorithm fib for calculating the Fibonacci numbers where calculating fib(n) is in $\mathcal{O}(n)$. The Fibonacci numbers are defined as follows: $F_0 = 0$, $F_1 = 1$, en $F_i = F_{i-1} + F_{i-2}$ voor $i \geq 2$.

(b) Apply the algorithm for knapsack01 to the following set S, with items s_i with benefit b_i and weight w_i:

	b	w
s_1	2	1
s_2	5	2
s_3	4	3

with maximal total weight $W = 5$.

Give your answer as a table:

$k \backslash w$	$\ldots$
$\vdots$	

Exercise 7. *(4+4+3 points)*
This exercise is concerned with the Huffman coding. Given is the following table of frequencies of characters, with Fibonacci numbers for the frequencies:

a	b	c	d	e	f	g
1	1	2	3	5	8	13

(a) Give a Huffman-coding tree for obtaining the encodings of the characters from the frequency table, and also give for each character explicitly its encoding as obtained from the tree.

(b) It is possible to adapt the Huffman-coding tree by using the coding alphabet $\{0, 1, 2\}$ (instead of $\{0, 1\}$). Give in this setting a coding-tree for the characters from the frequency table.

(c) Explain briefly how constructing the Huffman-coding tree fits in the paradigm of greedy choice.

Exercise 8. *(4+4+3 points)*
This exercise is concerned with pattern matching.

(a) Give is a pattern P in which all characters are different. Give for this specific case an adaptation of the brute-force pattern machting algorithm in $\mathcal{O}(n)$ with n the length of the text in which we search for P.

(Pseudo-code is allowed but not necessary).

(b) Given are the text T and the pattern P over the alphabet $\Sigma = \{a, b, c, d\}$:

$$\begin{aligned} T &= a\,b\,a\,d\,d\,b\,a\,c\,a\,d\,a\,d\,a\,b\,a\,c \\ P &= a\,b\,a\,c \end{aligned}$$

Apply the Boyer-Moore pattern matching algoritme. First give the function last that gives the last occurrence of a character in P. Give and number all steps in the application of the algorithm.

(c) Give an example of a worst-case input for the Boyer-More pattern matching algorithm, and explain what is (hence) the worst-case time complexity in terms of $\mathcal{O}$.

Het tentamencijfer is (het totaal aantal points plus 10) gedeeld door 10.