Exam Data Structures and Algorithms 2015-2016

Tuesday October 20, 2015, 15.15-18.00

7 exercises

Answers may be given in English or in Dutch.

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

(a) Assume given two queues, both with operations enqueue and dequeue. Give in this setting an implementation of a stack with operations push and pop.

(b) Give and explain the worst-case time complexity in terms of $\mathcal{O}$ of your operations push and pop from (a).

(c) Assume given a singly linked list. In a node v we have operations $v.next$ and $v.element$ with the suggested meaning. For a list L we have operations $L.first$ and $L.last$ with the suggested meaning.

Give in this setting an implementation of a stack with operations push and pop.

(d) Give and explain the worst-case time complexity in terms of $\mathcal{O}$ of your operations push and pop from (c).

Exercise 2. *(4+4+5 points)*
This exercise is concerned with sorting.

```
Algorithm selectionSort(A, n):
    for i := 1 to n − 1 do
        m := i
        for j = i + 1 to n do
            if A[j] < A[m] then
                m := j
        x := A[m]
        A[m] := A[i]
        A[i] := x
```

(a) Apply selection sort to the array $[5, 3, 1, 4, 2]$.

(b) Give and explain the worst-case time complexity of selection sort in terms of $\mathcal{O}$.

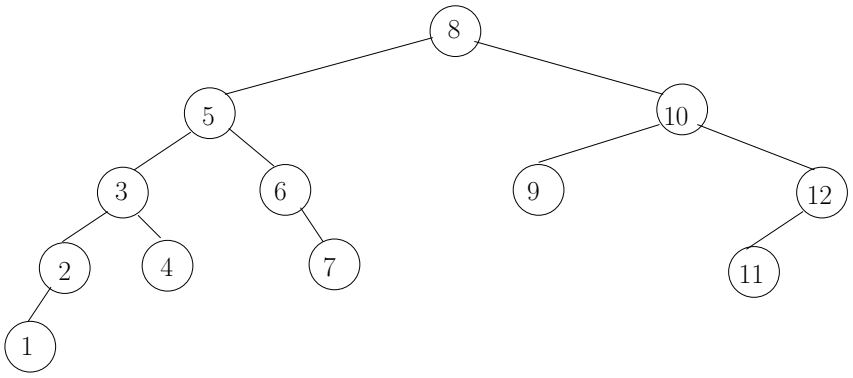
(c) Argue that selection sort is correct, using an invariant.

Exercise 3. (*3+3+4 points*)
This exercise is concerned with hash tables and tree-like data structures.

(a) What is the worst-case time complexity of adding an item to a hash table of length n, solving collision using probing?

(b) Give an example of a binary search tree with nodes labeled $1, 2, 3, 4$ and maximal height.

(c) Remove from the following AVL-tree the node with label 9. Give all steps.



Exercise 4. (*5+4+4 points*)
This exercise is concerned with traversals of graphs and trees represented using a linked structure.

(a) Give pseudo-code for a non-recursive algorithm for level-order traversal of a binary tree, using a queue.

(b) Explain how your algorithm can be adapted to an algorithm for breadth-first search (BFS) through a graph.

(c) Give and explain the worst-case time complexity for the algorithm for BFS, assuming that the input-graph is represented using an adjacency-list.

Exercise 5. *(4+4+5 points)*
Consider the algorithm for knapsack01:

Algorithm knapsack01(S, W):
  new $B[0 \dots n, 0 \dots W]$
  for $w := 0$ **to** W **do**
    $B[0, w] := 0$
  for $k := 1$ **to** n **do**
    $B[k, 0] := 0$
    for $w := 1$ **to** W **do**
      if $w_k \leq w$ **then**
        $B[k, w] := \mathsf{max}(B[k-1, w], B[k-1, w-w_k] + b_k)$
      else
        $B[k, w] := B[k-1, w]$

(a) Apply the algorithm to maximal weight $W = 5$ and the following set S with items with benefit and weight:

	b	w
s_1	3	2
s_2	2	1
s_3	4	3

Give your answer in the form of a table

$k \backslash w$	$\dots$
$\vdots$	

(b) Give and explain the worst-case time complexity of the algorithm for knapsack01 in terms of $\mathcal{O}$.

(c) Adapt the algorithm so that is uses only an array $B[0 \dots W]$.

Exercise 6. *(4+3+3+4 points)*
This exercise is concerned with the Huffman algorithm for computing optimal prefix code. The input is a set of characters C where every character c in C has frequency $c.freq$. We use a priority queue Q.

Algorithm HuffmanCode(C):
  $n := |C|$
  $Q := C$
  for $i = 1$ **to** $n - 1$ **do**
    new *node* z
    $z.left := x :=$ removeMin(Q)
    $z.right := y :=$ removeMin(Q)
    $z.freq := x.freq + y.freq$
    insert(Q, z)
  return removeMin(Q)

(a) Apply Huffman's algorithm to the following set of characters with frequencies:

$$a : 1 \quad b : 1 \quad c : 3 \quad d : 3 \quad e : 5 \quad f : 11$$

Give step by step (in pictures) the construction of your coding tree, and finally give the encoding of every character.

(b) Explain why Huffman's algorithm is greedy.

(c) Explain that Huffman's algorithm is non-deterministic.

(d) Give and explain the worst-case time complexity of Huffman's algorithm for an input C consisting of n characters, in terms of $\mathcal{O}$. Explain if necessary implementation choices.

Exercise 7. *(4+5 points)*

(a) What is the worst-case time complexity of brute-force pattern matching? Give an example of a pattern of length 3 and a text of length 10 that illustrate that the bound is tight.

(You do not have to apply the algorithm to your pattern and text.)

(b) The brute-force pattern matching algorithm can be improved for the special case that the input pattern is known to consist of all different symbols.

Give pseudo-code for this algorithm.

The mark for the midterm is (the total number of points plus 10) divided by 10.