Exam Data Structures and Algorithms 2016-2017

Tuesday October 25, 2016, 15.15–18.00

7 exercises

**Explain your answers unless otherwise speficied!**

---

**Exercise 1.** (*4+4+5 points*)
This exercise is concerned with selection sort.

**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 'in detail' selection sort to the input $A = [4, 3, 1, 5, 2]$ and $n = 5$.
Indicate clearly what happens for every $i$ and $j$.

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

(c) Give an invariant that can be used to show correctness of selection sort.
Explain why your invariant gives correctness.

**Exercise 2.** (*4+4+4+4 points*)

(a) Apply 'on the fly' the algorithm for bottom-up max-heap construction to the array $[1, 2, 3, 4, 5, 6, 7]$. You may use pictures.

(b) Start from the max-heap obtained as answer to (a), and continue with applying heapsort 'on the fly'. You may use pictures.

(c) What is the worst-case, and what is the best-case time complexity of heapsort in terms of $\mathcal{O}$? No motivation needed.

(d) Explain informally (no pseudo-code needed) but clearly how heapsort can be adapted to sort an array of natural numbers in *decreasing* order.

**Exercise 3.** (*4+4+5 points*)
This exercise is concerned with singly linked lists. 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. Do not assume predefined operations on lists.

(a) 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) Give pseudo-code for a $\Theta(n)$-time non-recursive procedure that reverses a singly-linked list of $n$ elements, only using a constant amount of additional storage. Briefly indicate why your algorithm is in $\Theta(n)$.

**Exercise 4.** (*4+4+4 points*)
This exercise is concerned with binary search trees (BSTs) and AVL-trees.

(a) What is the worst-case height of a BST with $n$ keys?

What is (approximately) the worst-case height of a AVL-tree with $n$ keys?

For both: no motivation needed.

(b) Give all possible BSTs with keys 1, 2, and 3.

Indicate for every BST whether it is an AVL-tree or not.

(c) Construct an AVL-tree by inserting one by one the numbers

$$6 \quad 4 \quad 5 \quad 3 \quad 1 \quad 7 \quad 2$$

starting from the empty tree. After each insertion, rebalance the tree if needed. Give your answer in pictures (with comments if needed).

**Exercise 5.** *(5+4+5 points)*
Consider the algorithm for a longest common subsequence (LCS) of input sequences $X = \langle x_1, \ldots, x_m \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$:

```
Algorithm LCS(X, Y):
  new array C[0...m, 0...n]
  for i := 0 to m do
    C[i, 0] := 0
  for j := 0 to n do
    C[0, j] := 0
  for i := 1 to m do
    for j := 1 to n do
      if x_i = y_j then
        C[i, j] := C[i − 1, j − 1] + 1
      else
        C[i, j] := max(C[i, j − 1], C[i − 1, j]
  return C
```

(a) Apply the LCS algorithm to the following sequences:
$X = \langle R, O, B, O, T, S \rangle$ and $Y = \langle D, R, O, N, E, S \rangle$.

Give also explicitly the longest common subsequence(s) that is (are) found.

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

(c) Can we improve the space complexity of the LCS algorithm?

Explain why and (informally) how, or why not.

**Exercise 6.** *(4+5 points)*
Given a set $S$ of activities $a_i$, each with start time $s_i$ and finish time $f_i$. Two activities $a_i$ and $a_k$ are *compatible* if $f_i \leq s_k$ or $f_k \leq s_i$. The activity selection problem is to find a maximal-size subset of $S$ consisting of compatible activities.

(a) Give an example showing that repeatedly choosing a compatible task with shortest duration time does not necessarily yield an optimal solution for the activity selection problem.

(b) Show the correctness of the greedy choice for an activity with smallest finish time.

**Exercise 7.** *(4+4+5 points)*
This exercise is concerned with string matching and varia.

(a) What is the number of steps used by the brute-force string matching algorithm applied to the pattern $P = a\,a\,b$ and the text $T = a^{1000}b$?

Here $a^{1000}b$ is the string consisting of first thousand $a$'s and then one $b$.

(b) Apply 'on the fly' the Knuth-Morris-Pratt string matching algorithm to the pattern $P = a\,a\,a$ and the text $T = b\,a\,b\,a\,a\,b\,a\,a\,a$.

Give and number all steps.

You do not have to give the failure function explicitly.

(c) We consider arrays of length $n$ containing in increasing order all numbers $0, \ldots, n$ except for one (the 'missing number'). An example with $n = 5$ is $A = [0, 1, 3, 4, 5]$, where 2 is the missing number.

Give (not necessarily in pseudo-code) an algorithm in $\mathcal{O}(n)$ that takes as input such an array $A$ and its length $n$, and gives back as output the missing number.

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