Linnaeus University
Department of Computer Science
Ola Petersson

Exam in Algorithms and Advanced Data Structures, 1DV516, 4.5 credits
Thursday 27 October 2016, 14.00–19.00

The exam consists of **6** questions, worth 10 points each. To pass you need 30 points. Write your name, personal identity number and the course code on each paper you hand in. You may answer the questions in Swedish or English.
When an algorithm is asked for, it should be understood that it should be as efficient as possible, and it should be expressed in such a way that it is understandable.
Allowed aids: Pen, pencil, eraser, and paper.
Not allowed: Consultation of any media outside of this document.

1. (a) State and solve the recurrence relation corresponding to the time complexity of the following algorithm:

```
int f($n$)
    if ($n$ == 1)
        return 1
    else
        return f($n-1$) + f($n-1$)
```

Express your answer in Θ-notation. (3p)

(b) State and solve the recurrence relation corresponding to the time complexity of the following algorithm:

```
int f($n$)
    if ($n$ == 1)
        return 1
    else
        return 2 · f($n-1$)
```

Express your answer in Θ-notation. (2p)

(c) Solve the following recurrence relation using telescoping

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n/2) + n & \text{otherwise} \end{cases}$$

Express your answer in Θ-notation. (2p)

(d) Solve the following recurrence relation using substitution

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n-1) + \log n & \text{otherwise} \end{cases}$$

Express your answer in Θ-notation. (3p)

2. (a) In a max-heap with n distinct elements, where can the smallest element reside? Give both the location in the array and the location in the implicit tree structure. (5p)

(b) Let A and B be two sets of numbers, represented as unordered linked lists of distinct numbers. You do have pointers to the heads of the lists, but *you do not know the list lengths.* Give an $O(\min\{|A|, |B|\})$ expected-time algorithm for determining whether $A = B$. (5p)

3. Let A be a sequence of numbers. Let *Greedy-Pick* be a function that extracts an increasing sequence from A, using the following method:

```
1. Create an empty list $L$;
2. Scan the first element in $A$, say $x$.
3. Remove $x$ from $A$, and append $x$ at the end of $L$;
4. while (there is unscanned element in $A$)
5. {   Scan the next element in $A$, say $y$;
6.     if ($y$ is greater than the last element in $L$)
7.     { Remove $y$ from $A$, and append $y$ at the end of $L$; }
8. }
9. return $L$;
```

For example, if $A = (1, 4, 3, 5, 2, 6)$, *Greedy-Pick* will pick the sequence $L = (1, 4, 5, 6)$ and leave $A = (3, 2)$.

Now, consider the following sorting algorithm, *Greedy-Sort*, based on *Greedy-Pick*:

```
1. Create two empty lists $A$ and $B$;
2. Put the original input sequence into $A$;
3. while ($A$ is not empty)
4. {   Greedy-Pick a sequence from $A$, called $L$;
5.     Merge $L$ into $B$;
6. }
7. return $B$;
```

(a) Show how *Greedy-Sort* sorts the sequence $A = (3, 1, 5, 4, 2)$ (2p)

(b) Argue why *Greedy-Sort* is correct, that is, that it actually sorts any sequence. (3p)

(c) What is the worst-case time complexity of *Greedy-Sort*? Motivate! (3p)

(d) Give a worst-case input for *Greedy-Sort*? (2p)

4. Let $G = (V, E)$ be a connected undirected graph where each edge is given a distinct label from 1 to $|E|$. Give an algorithm that finds the minimum number m such that G is still connected after removing all edges with labels greater than m. For 10p, your algorithm must run in $O(|E|)$ time; for 6p, it must run in $O(|E| \log |E|)$ time; and for 2p, it must run in $O(|E|^2)$ time.

5. Kruskal's algorithm for computing a minimum spanning tree (MST) for a weighted connected graph starts with an empty tree and repeatedly adds a remaining edge of smallest weight, which does not introduce a cycle, to the tree. An alternative algorithm starts with the original graph and repeatedly *removes* an edge of *largest weight* which doesn't disconnect the graph.

(a) Describe the algorithm in pseudo-code. (2p)

(b) Give the time complexity of the algorithm. Motivate! (3p)

(c) Argue that the algorithm produces a spanning tree. (2p)

(d) Argue that the algorithm produces an *minimum* spanning tree. (3p)

6. (a) Show that determining whether a list of numbers contains any duplicates is polynomially reducible to sorting. (5p)

(b) Consider an n-by-n grid where (i, j) contains $c(i, j) \geq 0$ gold coins. Starting in $(1, 1)$ and ending in (n, n), in each step you can move one step down or one step to the right. In each cell you visit, you get to collect the gold coins in that cell. Give an efficient algorithm to find the path earning you the maximum number of coins. (5p)