Linnaeus University
Department of Computer Science
Ola Petersson

Exam in Algorithms and Advanced Data Structures, 1DV516, 4.5 credits
Saturday 19 November 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.
Notice: I will not be able to visit the exam to answer questions. If anything is unclear, make reasonable assumptions and state them clearly.

1. (a) Formally show that $n/100 + \sqrt{n} = \Omega(n)$. (2p)

(b) Assuming that addition takes constant time, state the recurrence relation for the *running time* of f(n), and solve it:

```
int f(n)
  if (n == 1)
    return 1
  else
    x = f(n − 1)
    return x + x
```

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

(c) Assume instead that addition takes $\Theta(b)$ time, where b is the number of bits in the larger of the two numbers being added. Again, state and solve the recurrence relation for the running time of f(n) above. Use Θ-notation. (3p)

(d) What is the minimum and maximum number of nodes *at depth d* in a binary search tree. The root is at depth 0. (3p)

2. (a) We are to store integers in a open addressing hash table, in which collisions are resolved by means of double hashing. The size of the hash table is 11. The first hash function is $h_1(x) = x \mod 11$, and the second is $h_2(x) = 5 - (x \mod 5)$. Show the contents of the hash table after inserting the integers 16, 23, 34, 12, 56, in that order. (5p)

(b) What is the *load factor* of the hash table in (a). Explain how the load factor affects lookup time. What load factor would you recommend, and why? (2p)

(c) Suppose we are using Quicksort to sort an array of numbers. What is the maximum number of times that the largest element can be moved? Motivate! (3p)

3. There are exactly five different binary trees with three nodes:

(a) Give a recursive method int nTrees(n) that calculates the number of different binary trees with n nodes for $n \geq 0$. (5p)

(b) Your recursive algorithm is probably quite inefficient, because during the course of the computation on a large value of n, it calls itself on smaller values of n many times over. It can be made much more efficient by caching values already computed in a data structure and just consulting the data structure to see if the value has already been computed. Describe briefly how and where you would modify your code to take advantage of this idea. State what data structure you would use. (5p)

4. Let A be an empty max-heap, and let x be the root of a binary tree T of size n. Each node can be assumed to be a real number. Consider the following algorithm to build a max-heap from T:

```
TreeToHeap(x)
  if (x ≠ NIL)
    A.MaxHeapInsert(x)
    TreeToHeap(x.right)
    TreeToHeap(x.left)
```

where MaxHeapInsert is the insert we have learned in class.

(a) Give the time complexity of TreeToHeap. (2p)

(b) Assuming that T is a binary *search* tree, give an algorithm for finding the closest pair of nodes in T; i.e., return two nodes u and v, such that $|u.key - v.key|$ is minimized. Give the time complexity of your algorithm and justify your answer. (3p)

(c) Show that if T is a binary search tree, TreeToHeap can be modified to run in linear time. (5p)

5. (a) Given a weighted, directed graph $G = (V, E, w)$ and the shortest path distances $d(s, u)$ from a source node s to every other node u in G. However, you are not given the actual paths. Give an algorithm that computes the shortest path from s to an input node t in $O(|V| + |E|)$ time. Argue why your algorithm is correct. (5p)

(b) Given a weighted, directed graph $G = (V, E, w)$ with no negative-weight cycles. The *diameter* of the graph is the maximum-weight shortest path between any two nodes in the graph. Give a polynomial-time algorithm for finding the diameter of the graph. What is the running time of your algorithm? Does it matter whether it's a dense or sparse graph? Motivate! (5p)

6. (a) Give a linear-time, constant-space algorithm for determining whether an array $H[1..n]$ is a min-heap. (3p)

(b) A *palindrome* is a string that reads the same way from left to right as it does right to left. The longest palindrome subsequence of a string is its longest subsequence (of not necessarily consecutive symbols) that is a palindrome. Give an algorithm to determine the length of the longest palindrome subsequence of a string x in $O(|x|^2)$ time. What is the space complexity of your algorithm?(7p)