Aalto University, School of Science
Department of Computer Science
Tommi Junttila (050-4300861)

CS-A1140 Data structures and algorithms — Autumn 2018

Exam December 12th, 2018, at 09:00–12:00

Use of calculators is not allowed in the exam.
Note: your answers should be clear, well structured and concise.

1. Use the big O notation or precise terms such as "constant time on average" when answering the following questions. As usual, the time limits should be as precise as possible, but *you don't need to justify your answers* in this assignment.

   (a) If a is a dynamic array (`ArrayBuffer` in Scala) containing n integer elements, then how much time does it take to (i) find the element at an index i, (ii) find the smallest element in the array, (iii) append an element at the end of the array?

   (b) If q is a mutable priority queue implemented with max-heaps and contains n integer elements, then how much time does it take, in the worst case, to (i) add a new element in the queue, (ii) find the largest element in the queue, and (iii) find the smallest element in the queue?

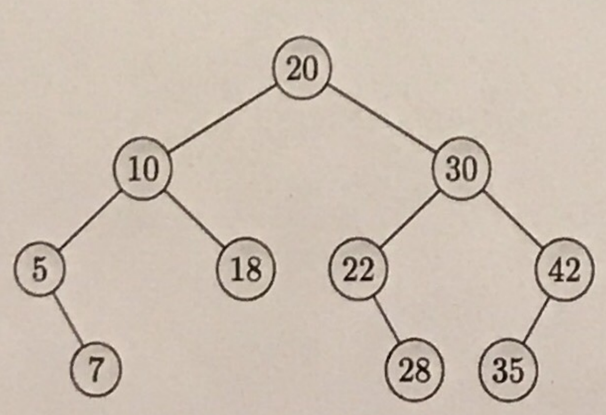
   (c) If s is a mutable set implemented with balanced binary search trees, such as AVL trees, and contains n integer elements, then how much time does it take, in the worst case, to (i) add a new element in the set, (ii) remove an element from the set, (iii) find the smallest element in the set, and (iv) list all the elements in the set? 10 points

2. Define the following concepts: (a) a sorting algorithm that *works in-place*, and (b) a *stable* sorting algorithm.
   Consider the Scala program on the right. (i) Which well-known sorting algorithm does it implement? (ii) Is the algorithm stable? (iii) Does the algorithm work in-place? (iv) What is the worst-case running time of the program and on what kind of input arrays does it occur? (v) How could the program be improved so that its worst-case running time either becomes smaller or occurs very unlikely? You don't have to provide code but only a description of the idea.
   In questions (iv) and (v), denote the length of the argument array a with n.
   Justify each answer with at most few sentences.

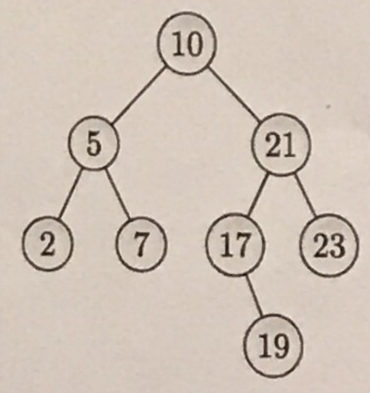
```
def sort(a: Array[Int]): Unit = {
  def swap(i: Int, j: Int): Unit = {
    val t = a(i); a(i) = a(j); a(j) = t
  }
  def helper(lo: Int, hi: Int): Int = {
    val p = a(hi)
    var i = lo - 1
    var j = lo
    while (j < hi) {
      if (a(j) <= p) { i += 1; swap(i, j) }
      j += 1
    }
    swap(i + 1, hi)
    i + 1
  }
  def inner(lo: Int, hi: Int): Unit = {
    val q = helper(lo, hi)
    if (lo < q - 1) inner(lo, q - 1)
    if (q + 1 < hi) inner(q + 1, hi)
  }
  if (a.length >= 2) inner(0, a.length - 1)
}
```

11 points

3. Consider the binary search tree A below (the keys are integers and drawn inside the nodes).



tree A



tree B

Perform the following operations (in the given order!), in the way described in the course material, to the tree A and, after each operation, illustrate the resulting binary search tree: (i) insert the key 29, (ii) remove the key 22, and (iii) remove the key 20.

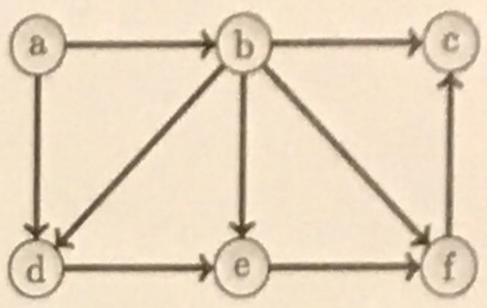
Define what is required of a binary search tree so that it is an AVL tree. Is the binary search tree A above an AVL tree? If not, why not?

Consider the AVL tree B above. Insert the key 18 to it in the way described in the course material that preserves the AVL property; describe what kind of transformations your insertion applies and why.

10 points

4. Give pseudo-code, or a clearly structured compact verbal description, of the *breadth-first search* algorithm for finding a path from a source vertex to a target vertex in a directed graph. What kind of data structures are required by the algorithm?

Consider the graph on the right. Illustrate its adjacency list representation. Explain/illustrate how the breadth-first search algorithm works on the graph when the source vertex is a and the target vertex is f. If a directed graph $G = (V, E)$ has n vertices (i.e., $|V| = n$) and m edges (i.e., $|E| = m$), then what its the worst-case running time of the algorithm? Justify your answer with few sentences.



Define the concept "topological ordering" (or "topological sort") for directed graphs. What its required of a directed graph in order to such ordering to exist? Does the graph shown above have a topological ordering? If it does not, why not? If it does, give (one) such ordering.

12 points

5. A "red-green-blue string" is a string over the alphabet {r, g, b}. Consider the following problem: Given a positive integer n, how many different red-green-blue strings of length n there are in which no consecutive positions contain the characters 'r' and 'g' (in either order). For instance, if $n = 1$, the solution is 3 (the strings are "r", "g", "b"). Similarly, if $n = 2$, the solution is 7 (the strings are "rr", "rb", "gg", "gb", "br", "bg", and "bb").

Develop a dynamic programming algorithm that solves the problem. Give both the recursive definition and the algorithm (in Scala or in pseudo-code). Your algorithm should work in time $O(n \log n)$, where the $\log n$ term is allowed for representing the numbers (they grow large quickly). Show how your algorithm solves the problem for $n = 5$ (do not attempt to write down the strings, there are too many of them).

8 points

6. Consider the parallel Scala program on the right. Describe with one or two sentences, what kind of value the program computes for the given argument arrays $a = [a_0, a_1, ..., a_{n-1}]$ and $b = [b_0, b_1, ..., b_{n-1}]$ and the binary function f.

The par.parallel(code1,code2) construction is as in the lecture material: it executes the codes code1 and code2 in parallel and returns their return values in a pair.

What are the (i) span, (ii) work, and (iii) amount of parallelism of the program? How could the program be improved to work faster in practise? Justify each answer with at most few sentences.

```
def g[T](a:Array[T], b:Array[T], f:(T,T)=>Double):
    Double = {
  require(a.nonEmpty)
  require(a.length == b.length)
  def inner(start: Int, end: Int): Double = {
    if(start == end) f(a(start), b(start))
    else {
      val mid = start + (end - start)/2
      val (l, r) = par.parallel(
        inner(start, mid),
        inner(mid+1, end)
      )
      l + r
    }
  }
  inner(0, a.length-1)
}
```

8 points

7. At what time did you finish answering the exam questions?

1 points