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

CS-A1140 Data structures and algorithms

Autumn 2017

Exam December 11th, 2017, at 13:00–16:00

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

1. Consider the Scala program on the right. Describe, with one sentence, what kind of a computational problem does the method solve? What are the (i) worst-case and (ii) best-case running times of the method with respect to the length n of the argument a? In each case, justify your answer with few sentences; also explain on which kind of inputs do the worst and best case running times occur.

```
def methodX(a:Array[Int],t:Int):Boolean={
  var i = 0
  var found = false
  while (i < a.length && !found) {
    val v = a(i)
    var j = i+1
    while (j < a.length && !found) {
      if(v + a(j) == t)
        found = true
      j += 1
    }
    i += 1
  }
  return found
}
```

How do the running times change (or do they stay the same) if we modify the first line to be "`def methodX(a: List[Int], t: Int): Boolean = {`"? 7 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? (v) What is the best-case running time of the program? (vi) Do the worst and best case running times change if we uncomment the first line in the method HELPER?
In questions (iv), (v) and (vi), denote the length of the argument array a with n.
Justify each answer with at most few sentences.

```
def sort(a: Array[Int]): Unit = {
  val aux = new Array[Int](a.length)
  def helper(l: Int, m: Int, r: Int): Unit = {
    // if(a(m-1) <= a(m)) return
    var (i, j, d) = (l, m, l)
    while(i < m && j <= r) {
      if(a(i) <= a(j)) {aux(d) = a(i); i += 1}
      else {aux(d) = a(j); j += 1}
      d += 1
    }
    while(i < m) {aux(d) = a(i); i += 1; d += 1 }
    while(j <= r) {aux(d) = a(j); j += 1; d += 1 }
    d = l
    while(d <= r) {a(d) = aux(d); d += 1 }
  }
  def inner(l: Int, r: Int): Unit = {
    if(l < r) {
      val m = l + (r - l)/2
      inner(l, m)
      inner(m+1, r)
      helper(l, m+1, r)
    }
  }
  inner(0, a.length-1)
}
```

13 points

3. Explain how one can implement a mutable set data structure by using hash tables based on chaining (also called "open hashing"). Your explanation should include definitions for the terms "hash function", "collision", and "load factor". Furthermore, it should give answers to the following questions: (i) How do the insertion and removal of elements in the set work? (ii) What is required for the insertion of elements to work in constant time on average? (iii) What

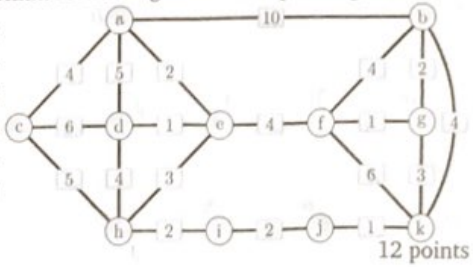
does "rehashing" mean and when/why should one perform it? In your explanation, denote the length of the hash table by m and the number of elements inserted in it by n.

Consider a hash table of size $m = 7$ for storing 32-bit integers. Use chaining and the hash function $h(x) = x \bmod m$. Illustrate the contents of the hash table (after each step) when the keys 15, 9, 8, and 22 are inserted in this order in the table.

Which of the following two functions is probably a better hash function for strings, $h_a(s) = \sum_{i=1}^{|s|} \text{int}(c_i) \bmod m$ or $h_b(s) = \sum_{i=1}^{|s|} 101^{i-1} \times \text{int}(c_i) \bmod m$? In both functions, the argument string s of length $k = |s|$ consists of the characters $c_1 \ldots c_k$ and $\text{int}(c)$ gives the integer representation of a character c, which typically is a value in the range [32,126]. Justify your answer, possibly by using a very small example. 12 points

4. Let us consider connected, edge-weighted undirected graphs. Define the concept of *minimum spanning trees* for such graphs. Describe in pseudocode, or verbally in a clear and structured manner, either (i) Kruskal's or (ii) Prim's algorithm for finding minimum spanning trees.

   What kind of data structures are required for storing such graphs and by the algorithm? Explain/illustrate how your algorithm works on the graph shown on the right. If a graph has n vertices and m edges, then what is the worst-case running time of the algorithm? Justify your answer with few sentences.



   12 points

5. A binary string is a string over the alphabet $\{0,1\}$. Consider the following problem: Given a positive integer n, how many different binary strings of length n there are that do not have two consecutive 1's? For instance, if $n = 3$, the solution is 5 (the strings are 000, 001, 010, 100, and 101). Similarly, if $n = 4$, the solution is 8 (the strings are 0000, 0001, 0010, 0100, 0101, 1000, 1001, and 1010).

   Develop a dynamic programming algorithm that solves the problem. Give both the recursive definition and the algorithm (in Scala or in pseudocode). Your algorithm should work in time $O(n \log n)$. Show how your algorithm solves the problem for $n = 10$ (do not attempt to write down the strings, there are many of them). 8 points

6. Consider the Scala program on the right, computing the value $\text{op}(\ldots \text{op}(\text{op}(a_0, a_1), a_2), \ldots a_{n-1})$ for an array $[a_0, a_1, \ldots, a_{n-1}]$ in parallel, provided that the binary operator op is associative (i.e., $\text{op}(x, \text{op}(y,z)) = \text{op}(\text{op}(x,y),z)$ for all the values x, y, z). The par.parallel(code1,code2) construction is as in the lecture material, executing code1 and code2 in parallel and returning their return values.

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

   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? Why does the program not always work correctly when op is not associative? Justify each answer with at most few sentences. 7 points

7. At what time did you finish answering the exam questions? 1 points