MIDTERM, WINTER 1997 — COMPUTER SCIENCE 308-251

Examiner: Luc Devroye
Date: February 20, 2:30 pm.
Time limit: 90 minutes.
Weight: Questions 1,4,5,7: 3 credits.
Questions 2,3,6,8: 2 credits.

Instructions: Calculators and computers are not allowed.
Answer in the examination booklets.
Do your scratchwork elsewhere.
Note the suggested lengths of the answers.

Question 1. Linear time selection. In a computational model in which one comparison takes one time unit, the recurrence for the worst-case time of the median-of-3 algorithm for finding the k-th smallest of n elements is

$$T_n = T_{n/3} + T_{2n/3} + 2n \ .$$

A. Draw the recursion tree for this. (Half a page)
B. Use it to derive a Θ expression for T_n. (5 lines)
C. What is the expected time of this median-of-3 algorithm in Θ notation if we form the groups of 3 randomly? (5 lines)
D. Write the recurrence for the worst-case time complexity of a median-of-13 version of the algorithm. (1 line)

Question 2. Suffix tries, suffix trees. Consider a text consisting of n symbols followed by the terminal symbol \$. Let T be the suffix trie and S the suffix tree.
A. How many leaves does T have? (1 line)
B. How many internal nodes does S have? (1 line)
C. How many edges does S have? (1 line)
D. The storage of T is easily seen to be $O(n^2)$. Give an example of a binary text (all symbols are 0 or 1 except the terminal symbol) for which the storage grows at least as $\Omega(n^2)$. (2 lines)

Question 3. Game trees. Here is a modified Nim game: the second player always plays two turns in a row. As usual, the player who picks the last stick loses. Who wins the following games: Nim (2,2,2) and Nim (2,2,3)? (Credit only if *both* answers are correct.) (1 line)

Question 4. Ternary trees. Consider a complete ternary tree in which every level of nodes is full. Let l be the number of levels (the level of the root is included in the count), and let n be the number of nodes. For a node `x`, let `left[x]`, `middle[x]` and `right[x]` denote the three children of `x`. Let `parent[x]` be its parent. Answer the following questions (no explanations necessary):
A. How large is n as a function of l? (No sums in the answer, please!) (1 line)
B. If we use the standard implicit way of storing complete trees, and `x` is the node in position j, then in what position do we find `parent[x]`? (1 line)

C. The following algorithm `partialvisit (t)` (where `t` is a node, initially the root of the tree) traverses only part of the tree. Give an estimate of the complexity of the algorithm as a function of n. (1 line)

```
function partialvisit(t:  treenode);
  if (t != null) then partialvisit(left[t])
                      print(key[t])
                      partialvisit(right[t])
```

Question 5. 2-d trees. 2-d trees are k-d trees in dimension 2. Each node `x` has attributes `left[x]`, `right[x]`, `xcoordinate[x]`, `ycoordinate[x]`, `splittype[x]`, which are self-explanatory; `splittype[x]` is `(horizontal)` or `(vertical)` depending upon the position of the node in the tree.

A. Write a short recursive algorithm for determining whether the tree with root `t` has at least one node with x-coordinate exactly equal to `xvalue`. (Half a page)

B. Assuming that the tree is complete and all levels are full, what is the worst-case complexity of the algorithm? (1 line)

Question 6. Symbolic multiplication of polynomials. Assume that a polynomial

$$\sum_{i=0}^{n} a_i x^i$$

is stored symbolically by placing $(a_0, a_1, \ldots, a_n)$ in an array, where the a_i's are real numbers. Let $(b_0, b_1, \ldots, b_n)$ be another polynomial. In a RAM model with uniformly bounded computation times, how would you multiply both polynomials and compute the resulting polynomial $(c_0, c_1, \ldots, c_{2n})$? (7 lines)

Question 7. Treaps. Give a 10 to 12-line recursive algorithm for the function `deleteroot(t)` in a treap in which each node `x` has key `key[x]`, time index `time[x]`, and children `left[x]` and `right[x]`. The algorithm should not have any explicit loop, should never use a comparison between keys, and indeed should not use the word "key". The function recursively fixes the subtree of `t`, and returns a pointer (possibly null) to the root of `t`. *Add a drawing.* (1 Page)

Question 8. Binary search trees. Consider a random binary search tree constructed by random insertion of n randomly permuted data $X_1, \ldots, X_n$.

A. What is the expected size (as a function of n) of the size of the subtree rooted at the leftmost grandchild? (1 line)

B. What is the expected size of the subtree rooted at X_{27} (as a function of n, assuming $n > 27$)? (1 line)