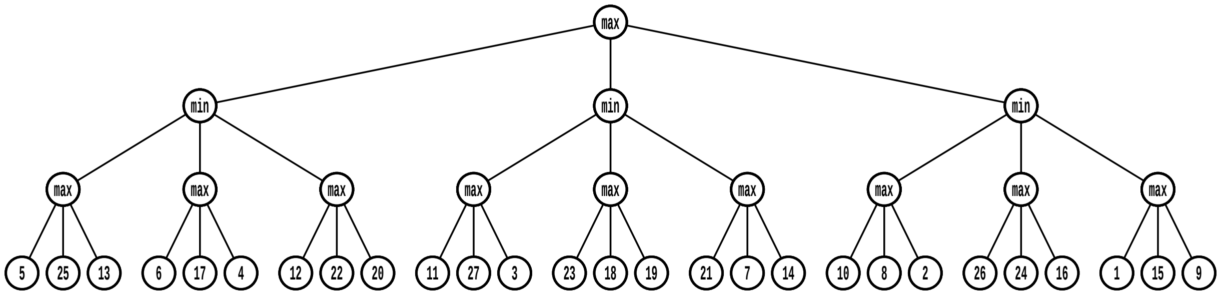
MIDTERM, WINTER 1996 — COMPUTER SCIENCE 308-251

Examiner: Luc Devroye
Date: February 15, 2:30 pm.
Time limit: 90 minutes.
Instructions: Calculators are not allowed. Answer in the examination booklets.

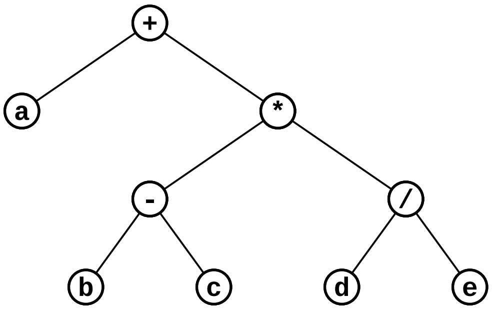
Question 1. GAME TREES. A game tree is shown below. Give the number of leaves visited (looked at) by $\alpha - \beta$ search when the value of the root is computed and children are always visited from left to right.



Question 2. SUFFIX TREES. Draw the suffix tree for the string `AAAAAAA`.

Question 3. THE SPARSE MATRIX ADT. Suppose that we have a 3000x4000 matrix of real values of which no more than 1% are nonzero. Suppose also that we wish to resolve any request to retrieve or assign a value to the element in position (i,j) in an average of about three probes into an underlying hash table implementation. Which hash table implementation would you choose? Provide a rationale for your choice of hash function and table size.

Question 4. CASE STUDY: EXPRESSION TREES. A binary expression tree is a binary tree that represents expressions in which all operators are binary (such as `+`, `-`, and `*`). An example is shown below for a tree that represents the expression `a+((b-c)*(d/e))`. The symbols `a, b,...` represent constants with certain numerical values. They are called operands, and correspond to leaves in the tree. The internal nodes represent values of subexpressions (example: the `-` node represents the value `b-c`). The value of the root is the value of the expression.



A. How many binary expression trees are there with n operands?
B. Given a binary expression tree stored with nodes having left and right pointers only, give a simple recursive algorithm for evaluating the value of an expression.
C. Now look at the evaluation of the expression in assembler language. As you may know, arithmetic operations are best carried out in registers in CPU. So, for example, to implement `b-c`, we do
   - Put `b` in register 1.
   - Put `c` in register 2.
   - Subtract register 2 from register 1. Leave result in register 1.

   Communication with memory is rather slow, so to execute `d/e`, we use registers 2 and 3, leaving the result in register 2. To do `*` in the example, we then need only multiply registers 1 and 2 and leave the result in register 1. Thus, the entire expression can be evaluated using only 3 registers and without ever storing intermediate results in memory. For a given expression tree, let the minimal number of registers needed be called M if intermediate results are always left in these registers. Registers and CPU are expensive, so M must be kept small. Tell me why it is always possible to evaluate an expression (tree) with $M \leq 1 + \log_2 n$ where n is the number of operands. Explain in a paragraph how you would do this.

Question 5. RECURRENCES. Draw the first few levels of the recursion tree for linear time selection when we use a median-of-9 approach instead of a median-of-5.

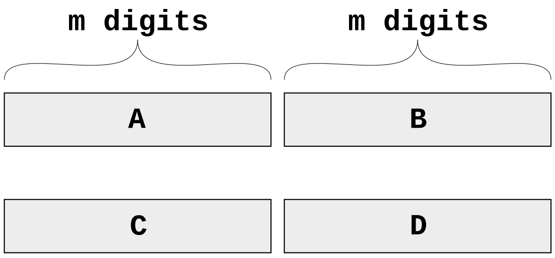
Question 6. INFORMATION AND BITS. Propose an efficient way of representing (the shape of) a ternary position tree with n nodes as a sequence of bits. How many bits does your representation require? Do the same question for a ternary ordered tree, i.e., an ordered tree with up to three children per node.

Question 7. MULTIPLYING VERY LARGE INTEGERS. Let `x` and `y` be two very large integers consisting of n decimal digits each. Here n could be very large. In parts B, C and D, you'll explore three different ways of computing the product `x.y`.

A. How could you store these integers if the size of n is not known beforehand? Give a PASCAL or C type declaration and draw the data structure.

B. If you use the representation of part A, and if you multiply `x` and `y` using calculations that mimic ordinary multiplication as seen in high school, then what is the worst-case complexity as a function of n in Θ notation?

C. Consider $m = \lceil n/2 \rceil$. Then `x` and `y` may be broken into pieces as shown in the figure below:



represents x = A.10^m+B

represents y = C.10^m+D

In that case, we may also write a recursive program that uses the following identity when n, the maximal number of digits in `x` and `y` is greater than 1:

$$xy = (AC)\ 10^{2m} + (AD+BC)\ 10^{m} + (BD)\ .$$

If n=1, we do not recurse, but compute the result directly. If T_n denotes the worst-case time for this method, write down the recurrence relation for T_n and give a solution in Θ notation. How did you guess and/or get that solution?

D. Assume that we base our recursive solution instead on this:

$$xy = (AC)\ 10^{2m} + ((A+B)(C+D)-AC-BD)\ 10^{m} + (BD)\ .$$

Repeat the previous question. The recurrence you'll get is close to one you'll recognize, so the solution may at first be "guessed" by this proximity.

Question 8. HASHING. In open addressing with a table of size n, we use a totally random probing sequence for collision resolution. In Θ notation, answer the following questions. No explanations needed.

A. What is the expected time needed to insert n/2 elements in an empty table?

B. What is the expected time needed to insert n elements in an empty table? (Note: at the end, the table is full.)

C. What is the expected time needed to insert 10 elements in a table that already holds n-10 elements? (Note: at the end, the table is full.)

Question 9. BINARY SEARCH TREES. Let us permute the integers 1 , 2 , . . . , n as follows. Start with an array A[1..n] with A[i]=i for all i. Form a new distorted array A[i..j] A[1..i-1] A[j+1..n], where we use concatenation. Thus, elements in positions i through j are moved to the front. This is like grabbing a number of consecutive cards from the body of a card deck and placing them on top. Repeat this operation 10 times. Each time, you are completely free to pick i and j. Feed the integer array A to a procedure that constructs a binary search tree by consecutive insertions in the standard manner. Thus, A[1] is the root, and so forth. How high is this tree *in the best case* in Θ notation? Why?