Comp 251: Assignment 4

Instructor: Jérôme Waldispühl

Due on April 8th at 11h59

- Your solution must be returned electronically on MyCourse.
- Written answers and programming questions must be returned in separate submission folders on MyCourse.
- The only format accepted for written answers are PDF or text files (e.g. `.txt` or `.rtf`). PDF files must open on SOCS computers. Any additional files (e.g. images) must be included in the PDF.
- Do not submit a compressed files (e.g. zip files). Upload instead each PDF or text file individually.
- The solution of programming questions must be written in java. Submit the Java source file only (i.e. `.java`). Your program should compile and execute of SOCS computers in a terminal. Java files that do not compile or execute properly on SOCS computer will not be graded.
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. You must indicate on your assignments the names of the persons with who you collaborated or discussed your assignments (including members of the course staff). If you did not collaborate with anyone, you write "No collaborators" at the beginning of your document. If asked, you should be able to orally explain your solution to a member of the course staff.
- Unless specified, all answers must be justified.
- When applicable, your pseudo-code should be commented and indented.
- The clarity and presentation of your answers is part of the grading. Be neat!
- Violation of all rules above may result in penalties or even absence of grading (Please, refer to the course webpage for a full description of the policy).
- Partial answers will receive credits.
- The course staff will answer questions about the assignment during office hours or in the online forum at `https://osqa.cs.mcgill.ca/`. We urge you to ask your questions as early as possible. We cannot guarantee that questions asked less than 24h before the submission deadline will be answered in time.

1. (50 points) We want to compare the naive and Karatsuba divide-and-conquer methods to multiply two integers `x` and `y`. Download the java file `multiply.java` from the course web page. Here, your task is to implement a recursive version of these algorithms in the methods

`naive(int size, int x, int y)` and `karatsuba(int size, int x, int y)`, and to use them to compare the efficiency of each algorithm (i.e. number of arithmetic operations). The variable `size` is the size of the integers `x` and `y`, and is defined as the number of bits used to encode them (Note: we assume that `x` and `y` have the same size).
Each method (i.e. `naive` and `karatsuba`) will return an integer array `result` that stores the value of the multiplication in the first entry of the array (i.e. `result[0]`), and the cost of this computation in the second entry (i.e. `result[1]`). We define the cost as the number of brute force arithmetic operations of the (addition, subtraction, or multiplication) executed by the algorithm multiplied by the size (in bits) of the integers involved in this operation (Note: We ignore the multiplication by powers of 2 which can be executed using a bit shift. Of course, this is a crude approximation).
In particular, for the base case (i.e. when the size of the integers is 1 bit), this cost will be 1 (brute force multiplication of two integers of size 1). In the induction case, the naive method executes 3 arithmetic operations of integer of size m (i.e. cost is $3 \cdot m$), in addition of the number of operations executed by each recursive call to the function. By contrast, the Karatsuba algorithm requires 6 arithmetic operations of size m on the top of the cost of the recursion.
The output of your program will print a list of numbers such that, the first number of each row is the size of the integers that have been multiplied, the second number is the cost of the naive method, and the third number the cost of the Karatsuba method.

2. (25 points) We remind the master method for determining the asymptotical behaviour of a recursive function.

   Theorem 1 (Master method) *Let $a \geq 1$ and $b \geq 1$ be two constants, and $f(n)$ a function. $\forall n \in \mathbb{N}^+$ we define $T(n)$ as:*

   $$T(n) = aT\big(\frac{n}{b}\big) + f(n), \textit{where } \tfrac{n}{b} \textit{ is interpreted as } \lfloor \tfrac{n}{b} \rfloor \textit{ or } \lceil \tfrac{n}{b} \rceil.$$

   Then, we can find asymptotical bounds for $T(n)$ such that:

   1. *If $f(n) = O(n^{\log_b a - \epsilon})$ with $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.*
   2. *If $f(n) = \Theta(n^{\log_b a} \cdot \log^p n)$, then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$.*
   3. *If $f(n) = \Omega(n^{\log_b a + \epsilon})$ with $\epsilon > 0$, and $a \cdot f\big(\frac{n}{b}\big) \leq cf(n), \forall n > n_0$ with $c < 1$ and $n_0 > 0$. Then $T(n) = \Theta(f(n))$.*

   When possible, apply the master theorem to find the asymptotic behaviour of $T(n)$ from the following recurrences. Show your work and justify your answer for each recurrence.

   (a) (5 points) $T(n) = 25 \cdot T(\frac{n}{5}) + n$

   (b) (5 points) $T(n) = 2 \cdot T(\frac{n}{3}) + n \cdot \log(n)$

   (c) (5 points) $T(n) = T(\frac{3n}{4}) + 1$

   (d) (5 points) $T(n) = 7 \cdot T(\frac{n}{3}) + n^3$

   (e) (5 points) $T(n) = T(n/2) + n(2 - \cos n)$

3. (25 points) Let T_A and T_B be two function returning the running time of algorithms A and B, defined by the recusions $T_A(n) = 7T_A(\frac{n}{2}) + n^2$ and $T_B(n) = \alpha T_B(\frac{n}{4}) + n^2$. Find the largest integer value of α for which algorithm B is asymptotically faster than A.