CS 341, Fall 2025 T. Brown, É. Schost

ASSIGNMENT 1

DUE: Thu Sep 25, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies.

Note: All logarithms default to base 2 (i.e., $\log x$ is defined as $\log_2 x$). **Note:** "Giving" an algorithm means doing the four parts (i)–(iv) as described on the course web page.

1. [10 marks] **Order notation.**

   For each of the following questions, please justify your answers.

   (a) [3 marks] Give a function that is $o(n^3)$ but $\Omega(n^{3-\epsilon})$ for all $\epsilon > 0$ (and justify this).

   (b) [2 marks] Use the limit technique to compare the growth of 1.01^n and n^2 (and report the relationship in terms of $\omega, \Omega, o, O, \Theta$ as appropriate).

   (c) [3 marks] Suppose $f(n) \in o(g(n))$. Does it follow that $\log(f(n)) \in o(\log(g(n)))$?

   (d) [2 marks] Compare the growth of $\log(n!)$ and $n \log n$ (and report their relationship).

2. [13 marks + bonus] **Recurrence relations.** In all these questions, if you use the master theorem, you of course don't have to reprove it.

   (a) [2 marks] Consider an $n \times n$ upper triangular matrix M, with 1's on the diagonal (you can assume that n is a power of 2). We can write

   $$M = \begin{bmatrix} A & B \\ 0 & C \end{bmatrix}$$

   with A, B, C matrices of size $n/2 \times n/2$. Then, one can verify (you don't have to do it) that the inverse of M is

   $$M^{-1} = \begin{bmatrix} A^{-1} & -A^{-1}BC^{-1} \\ 0 & C^{-1} \end{bmatrix}$$

   Suppose you use Strassen's algorithm to multiply matrices. Write the recurrence for the cost of inverting M, and solve it (give a Θ).

   (b) [2 marks] Set $T(n) = 3T(n-4) + 1$ for $n > 3$ and $T(n) = 0$ for $n \leq 3$. Give a Θ bound for $T(n)$, and **prove inductively** that your bound is correct.

   (c) [2 marks] A well-known algorithm for multiplying polynomials is based on the following recursive approach, for n a power of a power of 2: do $2\sqrt{n}$ calls in size $\sqrt{n}$ and $n \log(n)$ extra operations. The base case is $n = 2$, with $T(2) = 0$. Give a Theta bound for $T(n)$.

   (d) [3 marks + bonus] Suppose that Algorithm A solves a problem of size n by dividing it into 10 subproblems of size $n/5$ and 4 subproblems of size $n/4$, recursively solving each subproblem, and then combining the solutions in time n. Give an Ω bound and a (not necessarily matching) O bound on its runtime (you will not get any credit for answering $T(n) \in \Omega(n)$).

   Bonus, worth 3 extra marks: give and prove a Θ bound on the runtime. You cannot use any theorem that was not proved in class.

(e) [2 marks] Set $T(n) = 2T(n/2) + n\log^k(n)$ for $n > 1$, $T(1) = 0$, where k is a constant. Give a Theta bound for $T(n)$. You can assume that n is a power of 2 if you want.

(f) [2 marks] Let $T(n) = 4T(n/2) + n^{2.5}\log n$, and $T(0) = 0$. Solve using the general master theorem.

3. [10 marks] **Divide & Conquer.**

The number of bits in the binary representation of an integer a is given by

$$\lg a = \begin{cases} 1 & \text{if } a = 0; \\ 1 + \lfloor \log |a| \rfloor, & \text{otherwise.} \end{cases}$$

Consider a software library for multi-precision nonnegative integers (whose type we will call `LongInt`) that includes the following two functions:

- `LongInt(x)` – takes as input a primitive type integer $x, 0 \leq x < 2^{64}$, and returns a `LongInt` with the value x.
  Run-time: $O(1)$.
- `Mul(A,B)` – takes as input two `LongInt`'s and returns a `LongInt` with the value $A \times B$.
  Run-time: $O(\max(\lg A, \lg B)\min(\lg A, \lg B)^{\log_2 3-1})$ (Karatsuba's algorithm)

In this question you will consider the problem of multiplying together n primitive type integers to obtain a single large integer.

As input you are given an array $a[1\ldots n]$ of length $n \geq 1$, each entry being a primitive integer (e.g., fits into 64 bits). Your task is to write a procedure that computes the product of all the integers in a and returns the answer as a `LongInt`. Here is one solution that simply accumulates the product:

```
MultiMul($a[1\ldots n], n$)
    $P$ := LongInt($a[1]$)
    for $i$ from 2 to $n$ do
        $P$ := Mul($P$, LongInt($a[i]$))
    return $P$
```

To estimate the run-time of algorithms using the multi-precision library we sum the cost of the calls to the functions defined above. Note that operations with primitive types like array indices and loop variables are assumed to have unit cost.

(a) [2 marks] Analyse the run-time of procedure `MultiMul`, that is, derive a tight big-O bound for the run-time in terms of n.

(b) [8 marks] Design and analyse a divide and conquer algorithm that does the same job as `MultiMul` but with run-time $O(n^{\log_2 3})$.