HOMEWORK 2

Solve as many of the following problem as you can. Your writeup should reflect your own work !

You should send solutions by email to gabriel.istrate@gmail.com.

DEADLINE: Thursday 29 November, noon (FIRM).

1 Counting inversions

An inversion in an array $A[1], A[2], \ldots, A[n]$ storing a permutation is a pair $i < j$ with $A[i] > A[j]$.

Modify the MERGESORT algorithm to obtain an algorithm of complexity $O(n \log n)$ that counts the number of inversions in the array.

Sketch the analysis of the algorithm to show that the running time is really $O(n \log n)$.

2 Integer Multiplication in two's complement notation

The *two's complement notation* is a way to represent integers in some range $[-2^{n-1}, 2^{n-1}]$, where $n \geq 2$, by n bits.

The most significant bit is 0 if the represented number is nonnegative and 1 otherwise.

The value w of an N-bit integer $a_{n-1} \ldots a_1 a_0$ is given by the following formula:

$$-a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots + 2a_1 + a_0.$$

In other words, when the most significant bit a_{n-1} is 0 the representation is the same as in the binary representation.

Example: 3 is represented on 4 bits as 0011.

When the most significant bit is we subtract 2^{n-1} from the binary representation of the

Example: -5 is represented on 4 bits as 1011.

Devise an algorithm that multiplies two integers written in two's complement notation, obtaining the result in such a representation.

3 Big-Oh notation

For all f, g below write which of the following relations is true as $x \to \infty$.
A: $f = O(g)$, B: $f = \theta(g)$, C: $f = o(g)$, D: $g = O(f)$, E: none of them,

1. $f(x) = x \log_2(x)$, $g(x) = x \log_2(x) + x$.
2. $f(x) = x \log_2(x)$, $g(x) = x \log_2(x) + x^{1.01}$.
3. $f(x) = 0.1 \cdot x^2$, $g(x) = 10!/x$
4. $f(x) = sin(x)$, $g(x) = cos(x)$

4 Three-way Mergesort

Describe the pseudocode of a version of Mergesort where instead of dividing your input into two equal halves you divide it into **three** equal parts.

How does the code for the Merge operation change ?

Analyze the complexity of the Merge subroutine and then use the Master theorem to derive the complexity of Three-Way Mergesort.

5 Sorting $X + Y$

Given two vectors $X = (X_1, X_2, \ldots, X_n)$ and $Y = (Y_1, Y_2, \ldots Y_n)$, define set $Z = (X_i + Y_j : 1 \leq i, j \leq n)$.

Define and analyze a recursive algorithm A_1 for sorting Z. This algorithm should be different from the algorithm A_2 where we explicitly compute all the sums then sort.

How does the complexity of A_2 compare with that of A_1 ?

NOTE: *(for your information only) This problem, the $X + Y$* **sorting problem***, is one whose exact complexity is not known, and interesting. Talk to me if you want to know more.*