TIN093 EXERCISE WEEK 1

JOSEFIN ONDRUS

What does $n^{O(1)}$ mean?
Since

Is it correct to say $O(n+n+...+n) = O(n)$ or not? Explain

If we reformulate the expression we get $n+n+...+n = cn$ where $c \in \mathbb{N}$, $0 \leq c$. The purpose of $O(\cdot)$ is to define an upper bound based on the dominant factor as n approaches infinity. Since c is a constant value, n is the dominant factor. The execution time ($O(cn)$) will be greater than $O(n)$ for a finite number of instances but will eventually not grow faster.

Suppose you have three algorithms (for whatever problem) that need $O(n^2)$, $O(2^n)$, and $O(n!)$ time, respectively. When the speed of your machine is doubled, how does this affect the size of inputs you can handle?

Suppose you have an algorithm with two nested for-loops where the loop indices run through at most n different values (such as: `for i = 1 to n do for j = i to n do [some elementary operation with` i `and` j`] endfor endfor`), and another part of the algorithm is just one such for-loop with at most n values. What would be the time complexity in O-notation? Explain in detail.

Lets call the double nested for-loop (1) and the single for-loop (2). Since the for-loops can run through at most n different values, worst case scenario for (1) will be reading $f(n) = n * n$ times, and worst case for (2) is reading $g(n) = n$ times. This gives us a time complexity of $O(f(n)+g(n))$ (+ some constants representing the elementary operation). From the properties of O we know that

$$O(f(n)+g(n)) = O(max\{f(n), g(n)\})$$

and for this algorithm: $g(n) \leq f(n)$ since $0 \leq n, n \in n^2$

which gives us the time complexity $O(f(n)) = O(n^2)$. Since the time for executing constants depends on the computer hardware we choose to disregard this.

What is the worst-case time complexity of adding 1 to (or subtracting 1 from) an integer with n digits, if we count operations with digits as elementary operations?

What is the worst-case time complexity of comparing two integers x, y with n digits, if we count operations with digits as elementary operations? Comparing means to decide whether $x < y, x = y, or x < y$.

Date: September 4, 2015.

How many digit operations do you need to compute the sum $1+2+3+\ \ +n$? Try to come up with a bound in O-notation which is as good as possible.

Give a rigorous proof that m integers of n digits can be added in O(mn) time. (Intuitively this should be true, but the details are not so obvious.)