HW1 – Bonus

There are three things that we have learned in this week that are really important:

(1). Input size. (All complexity functions are measured on input sizes hence you need figure out input sizes correctly.)

(2). Input doesn't take memory when it is read-only. Memory (in the context of space complexity of an algorithm) refers to "additional" memory needed.

(3). I keep saying that C-programs are algorithms but actually this is not completely right. I say this only for those students who hate Turing machines. Algorithms are Turing machines that halt on all inputs. Therefore, If you are given an algorithm, you have many many C programs to implement it; some are good some are bad. However, if I give you a C program, you usually don't have many drastically different "programs" to implement the C program (at most you use different compilers, but the assembly code you get is roughly the same.)

Many people also think to make an algorithm any times faster (on almost all inputs! not just a single one, not necessarily all inputs) is a mystery (Blum's speedup theorem). Of course, if you can make it two times faster, you can make it any times faster. Now, let us look at an idea that makes an algorithm two times faster.

For a natural number x represented in digits (so the size of input is $\log_{10} x$), we learned in our grammar school how to do arithmetic operations like addition, subtraction, multiplication, etc., on those digits. Being in digits, you are talking about alphabet $\{0, 1, 2, ..., 9\}$. Because of this, we were taught to do those arithmetics digit-by-digit. Now, suppose that you do two digits a time. Hence, in order to multiply

$$1234 \times 5678$$

you do this

$$(12)(34) \times (56)(78)$$

and now your alphabet is $\{00, ..., 99\}$ (if we humans have 100 fingers to begin with, then we would do math in this way thousands years ago) — you might have already noticed that the input size now is $\log_{100} x$ where x is the natural number mentioned at the beginning). This will make our multiplication at least two times faster. Can you figure it out and sketch how we would do this for n digit by n digit multiplication and why it is at least two times faster? This is why Blum's theorem works.