February 8, 2000

What's an ~~alligator?~~ algorithm?

A precise definition of the term **algorithm** is difficult, which is quite interesting since it is a central word in late 20^{th} century mathematics and computer science. It is as vital and important to such study as the **sonnet** is to the history and practice of poetry.

Here's some information about **algorithms** from volume 1 of *The Art of Computer Programming* by Donald Knuth (published by Addison Wesley). This is a foundational multi-volume work. I give some quotes from the very beginning of it.

From page 1:

> The word "algorithm" itself is quite interesting; at first glance it may look at though someone intended to write "logarithm" but jumbled up the first four letters. . . . the true origin of the word . . . comes from the name of a famous Persian textbook author, Abu Ja'far Mohammed ibn Mûsâ al-Khowârismî (*c.* 825) – literally, "father of Ja'far, Mohammed, son of Moses, native of Khowârizm." Khowârizm is today the small Soviet city of Khiva. Al-Khowârizmî wrote the celebrated book *Kitab al jabr w'al-muqabala* ("Rules of restoration and reduction"); another word, "algebra", stems from the title of his book, although the book wasn't really very algebraic.

From pages 4, 5, and 6:

> The modern meaning for algorithm is quite similar to that of *recipe, process, method, technique, procedure, routine,* except that the word "algorithm" connotes something just a little different. Besides merely being a finite set of rules which gives a sequence of operations for solving a specific type of problem, an algorithm has five important features:
>
> **1) Finiteness.** An algorithm must always terminate after a finite number of steps. . . .
>
> **2) Definiteness.** Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case. . . .
>
> **3) Input.** An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins. These inputs are taken from specified sets of objects. . . .
>
> **4) Output.** An algorithm has one or more outputs, i.e., quantities which have a specified relation to the inputs. . . .
>
> **5) Effectiveness.** An algorithm is also generally expected to be *effective.* This means that all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time . . .

Knuth continues on the same page to contrast his definition of algorithm with what could be found in a cookbook:

> Let us try to compare the concept of an algorithm with that of a cookbook recipe: A recipe presumably has the qualities of finiteness (although it is said that a watched pot never boils), input (eggs, flour, etc.) and output (TV dinner, etc.) but notoriously lacks definiteness. There are frequently cases in which the definiteness is missing, e.g., "Add a dash of salt." A "dash" is defined as "less than $\frac{1}{8}$ teaspoon"; salt is perhaps well enough defined; but where should the salt be added (on top, side, etc.)? ...

He concludes his comparison by writing:

> ... a computer programmer can learn much by studying a good recipe book.

I sadly report to you that I personally haven't found many good cookbooks. That might, of course, have some connection with my facility as a cook!

Homework

> This assignment is due on Wednesday, February 16. Students may work together in groups of at most 3 people and hand in a joint answer.

Here are the input and output of two algorithms:

Cubing The input is X and the output is X^3.

EXAMPLE If the input is 4, then the output is 64.

Cube inquiry The input is X and the output is **Yes** if X is the cube of an integer and **No** if X is not the cube of an integer.

EXAMPLES If the input is 27, the output is "Yes"; if the input is 23, the output is "No".

The homework problems

1. Use only what we've done in class to estimate the number of elementary arithmetic operations that some implementation of each algorithm will take. *You* describe the implementation – any implementation will do. Try to be precise, though, with your descriptions and your assertions about the number of operations needed: the number of operations will depend on the "size" (the number of digits) of the input (call it S), and may also depend on the input itself (call it X). Briefly explain in standard English sentences why your answer is reasonable and correct.

2. Suppose that it takes 1 second to do **Cubing** for a 100 digit number and 1 minute to do **Cube inquiry** for a 100 digit number, then (using your analysis) approximate the amount of time it will take to do each of these algorithms for a 200 digit number.