Lecture Notes: Computation Model

Yufei Tao
Department of Computer Science and Engineering
Chinese University of Hong Kong
taoyf@cse.cuhk.edu.hk

Computer science is a subject under mathematics. From your undergraduate study, you should have learned that, before you can even start to analyze the "running time" of an algorithm, you need to first define a computation model properly.

The random access machine (RAM) model. This is perhaps the model you are most familiar with. In the RAM model, the *memory* is an infinite sequence of *cells*, where each cell is a sequence of w bits for some integer w, and is indexed by an integer *address*. Each cell is also called a *word*; and accordingly, the parameter w is often referred to as the *word length*. The *CPU*, on the other hand, has a (constant) number of cells, each of which is called a *register*. The CPU can perform only the following *atomic* operations:

- Set a register to some constant, or to the content of another register.
- Compare two numbers in registers.
- Perform $+, -, \cdot, /$ on two numbers in registers.
- Perform the AND, OR, XOR on two registers.
- When an address x has been stored in a register, read the content of the memory cell at address x into a register, or conversely, write the content of a register into the memory cell.

The *time* (or *cost*) of an algorithm is measured by the number of atomic operations it performs. Note that the time is an integer.

A remark is in order about the word length w: it needs to be long enough to encode all the memory addresses! For example, if your algorithm uses n^2 memory cells for some integer n, then the word length will need to have at least $2 \log_2 n$ bits.

Dealing with real numbers. In the model defined earlier, the (memory/register) cells can only store integers. Next, we will slightly modify the model in order to deal with real values.

Note that simply "allowing" each cell to store a real value does not give us a satisfactory model because it creates several nasty issues. For example, how many bits would you use for a real value? In fact, even if the number of bits *were* infinite, still we would not be able to represent all the real values even in a short interval like $[0, 1]$ — the set of real values in the interval is *uncountably* infinite! If we cannot even specify the word length for a "real-valued" cell, how to properly define the atomic operations for performing shifts and the logic operations AND, OR, and XOR?

We can alleviate this issue by introducing the concept of *black box*. We still allow a (memory/register) cell c to store a real value x, but in this case, the algorithm is forbidden to look *inside* c, that is, the algorithm has no control over the representation of x. In other words, c is now a black box, holding the value x *precisely* (by magic).

A black box remains as a black box after computation. For example, suppose that two registers are both storing $\sqrt{2}$. We can calculate their product 2, but the product must still be understood as a real value (even though it is an integer). This is similar to the requirement in C++ that the product of two float numbers remains as a float number.

Now we can formally extend the RAM model as follows:

- Each cell can store either an integer or a real value.
- For operations $+, -, *, /$, if one of the operand numbers is a real value, the result is a real value.
- Among the atomic operations mentioned earlier, shifting, AND, OR, and XOR cannot be performed on registers that store real values.

We should note that, although mathematically sound, the resulting model — often referred to as the *real RAM* model — is not necessarily a realistic model in practice because no one has proven that it is polynomial-time equivalent to Turing machines (it would be surprising if it was). We must be very careful *not to abuse the power of real value computation.* For example, in the standard RAM model (with only integers), it is still open whether a polynomial time algorithm exists for the following problem:

Input: integers $x_1, x_2, ..., x_n$ and k
Output: whether $\sum_{i=1}^{n} \sqrt{x_i} \geq k$.

It is rather common, however, to see people design algorithms by assuming that the square root operator can be carried out in polynomial time — in that case, the above problem can obviously be settled in polynomial time under the real-RAM model!

Randomness. All the atomic operations are *deterministic* so far. In other words, our models so far do not permit *randomization*, which is important to certain algorithmic techniques (such as hashing).

To fix the issue, we introduce one more atomic operation for both the RAM and real-RAM models. This operation, named *RAND*, takes two non-negative integer parameters x and y, and returns an integer chosen uniformly at random from $[x, y]$. In other words, every integer in $[x, y]$ can be returned with probability $1/(y - x + 1)$. The values of x, y should be in $[0, 2^w - 1]$ because they each need to be encoded in a word.

Math conventions. We will assume that you are familiar with the notations of $O(.), \Omega(.)$, $\Theta(.)$, $o(.)$, and $\omega(.)$. The notation $\tilde{O}(f(n_1, n_2, ..., n_x))$ represents the class of functions that are $O(f(n_1, n_2, ..., n_x) \cdot \text{polylog}(n_1 + n_2 + ... + n_x))$, namely, $\tilde{O}(.)$ hides a polylogarithmic factor. The symbol $\mathbb{R}$ denotes the set of real values.