Algorithms are the most important and durable part of computer science because they can be studied in a language- and machine-independent way. This means that we need techniques that enable us to compare the efficiency of algorithms without implementing them. Our two most important tools are (1) the RAM model of computation and (2) the asymptotic analysis of worst-case complexity.

Assessing algorithmic performance makes use of the “big Oh” notation that, proves essential to compare algorithms and design more efficient ones. While the hopelessly practical person may blanch at the notion of theoretical analysis, we present the material because it really is useful in thinking about algorithms.

This method of keeping score will be the most mathematically demanding part of this book. But once you understand the intuition behind these ideas, the formalism becomes a lot easier to deal with.

2.1 The RAM Model of Computation

Machine-independent algorithm design depends upon a hypothetical computer called the Random Access Machine or RAM. Under this model of computation, we are confronted with a computer where:

- Each simple operation (+, *, -, =, if, call) takes exactly one time step.

- Loops and subroutines are not considered simple operations. Instead, they are the composition of many single-step operations. It makes no sense for sort to be a single-step operation, since sorting 1,000,000 items will certainly take much longer than sorting 10 items. The time it takes to run through a loop or execute a subprogram depends upon the number of loop iterations or the specific nature of the subprogram.
2. ALGORITHM ANALYSIS

- Each memory access takes exactly one time step. Further, we have as much memory as we need. The RAM model takes no notice of whether an item is in cache or on the disk.

Under the RAM model, we measure run time by counting up the number of steps an algorithm takes on a given problem instance. If we assume that our RAM executes a given number of steps per second, this operation count converts naturally to the actual running time.

The RAM is a simple model of how computers perform. Perhaps it sounds too simple. After all, multiplying two numbers takes more time than adding two numbers on most processors, which violates the first assumption of the model. Fancy compiler loop unrolling and hyperthreading may well violate the second assumption. And certainly memory access times differ greatly depending on whether data sits in cache or on the disk. This makes us zero for three on the truth of our basic assumptions.

And yet, despite these complaints, the RAM proves an excellent model for understanding how an algorithm will perform on a real computer. It strikes a fine balance by capturing the essential behavior of computers while being simple to work with. We use the RAM model because it is useful in practice.

Every model has a size range over which it is useful. Take, for example, the model that the Earth is flat. You might argue that this is a bad model, since it has been fairly well established that the Earth is in fact round. But, when laying the foundation of a house, the flat Earth model is sufficiently accurate that it can be reliably used. It is so much easier to manipulate a flat-Earth model that it is inconceivable that you would try to think spherically when you don’t have to.\textsuperscript{1}

The same situation is true with the RAM model of computation. We make an abstraction that is generally very useful. It is quite difficult to design an algorithm such that the RAM model gives you substantially misleading results. The robustness of the RAM enables us to analyze algorithms in a machine-independent way.

\textbf{Take-Home Lesson:} Algorithms can be understood and studied in a language- and machine-independent manner.

2.1.1 Best, Worst, and Average-Case Complexity

Using the RAM model of computation, we can count how many steps our algorithm takes on any given input instance by executing it. However, to understand how good or bad an algorithm is in general, we must know how it works over all instances.

To understand the notions of the best, worst, and average-case complexity, think about running an algorithm over all possible instances of data that can be

\textsuperscript{1}The Earth is not completely spherical either, but a spherical Earth provides a useful model for such things as longitude and latitude.