Models of Computation

K. Subramani
Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV
ksmani@csee.wvu.edu

1 Introduction

The efficiency of an algorithm is determined by the number of steps it takes to terminate as a function of *input size*. Consequently, analyzing the running time of an algorithm requires a formal specification of *size of input*, i.e. how large the input is. There are 2 broad ways of measuring input size, viz. the *Uniform-Cost Model* and the *Log-cost Model* [2, 1].

Definition: 1.1 *A function $g(n) = O(f(n))$ if $g(n) \leq c.f(n)$ for some $c > 0$ and all $n = 1, 2, \ldots \infty$.*

Thus, $25.n = O(n^2)$, since $25.n \leq 25.n^2$, for all n; likewise $3.n = O(n)$, since $3.n \leq 4.n$, for all n. In other words, $O(g(n))$ is the set of functions $f(n)$ such that $c.g(n)$ sits above $f(n)$.

Definition: 1.2 *A function $g(n) = \Omega(f(n))$, if $g(n) \geq c.f(n)$ for some $c > 0$ and all $n = 1, 2, \ldots, \infty$.*

Thus, $\frac{1}{2}.n^2 = \Omega(n)$, since $\frac{1}{2}.n^2 \geq \frac{1}{2}.n$; likewise $n = \Omega(n)$. In other words, $\Omega(g(n))$ is the set of functions $f(n)$ such that $f(n)$ sits above $c.g(n)$.

Observe the symmetry in the definitions of $O()$ and $\Omega()$; it is clear that $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$.

2 The Uniform-Cost Model

In the Uniform-Cost Model, also called the *algebraic model*, every input element is counted as 1; thus if 10 elements are read into an array, the input size is 10 regardless of how large these numbers are. Further, every basic operation such as addition, subtraction, multiplication and division are assumed to take unit time. This model is not very realistic; however it is adequate to analyze operations in which arithmetic operations do not blow up the size of the constituent numbers. For instance, analyzing the MERGE-SORT() algorithm (see [2]) in the Uniform-Cost model gives a running time of $O(n.\log n)$, since there are $O(n.\log n)$ comparisions. An algorithm which takes a polynomial number of operations in the Uniform-Cost model is said to be *strongly polynomial*.

3 The Log-Cost Model

In the Log-Cost Model, also called the *bitwise model*, the number of bit operations are counted. Thus reading in 10 elements into an array requires $\log(a_1)+\log(a_2)+\ldots\log(a_{10})$ operations, where $a_1, a_2, \ldots, a_{10}$ are the elements that are read in. Further, in case of arithmetic operations, we count the number of bits involved. For instance, adding 2 $M - bit$ numbers in the log-cost model results in a cost of M, whereas in the Uniform-Cost model, the cost would be 2. This model is considered more realistic.

4 Differences between the models

There are problems such as linear programming [3] for which there are polynomial algorithms in the Log-Cost model, but no strongly polynomial algorithms. In fact, finding a strongly polynomial algorithm for linear programming is a famous open problem.

Consider the following algorithm to test if a number is prime:

Function PRIME-TESTER(N)
1: **for** ($i = 2$ **to** $N - 1$) **do**
2: **if** (N *mod* $i = 0$) **then**
3: N is composite
4: **end if**
5: **end for**
6: N is prime

Algorithm 4.1: An algorithm for testing primality

In the Uniform-Cost model, the input is of size 1 and the algorithm takes $N - 1$ steps; thus as a function of input size, the algorithm takes unbounded time. In the Log-Cost model, the input is of size $\log N$. Each *mod* operation takes time proportional to a division operation. Dividing a number P by a number Q takes $\lceil \frac{P}{Q} \rceil . \log P$ bit operations [1]. Accordingly, the total number of bit operations is:

$$\frac{N}{2}.\log N + \frac{N}{3}.\log N + \ldots \frac{N}{N-1}.\log N \quad (1)$$

$$\leq \sum_{i=1}^{N} .\log N.\frac{N}{i} \quad (2)$$

$$\leq N.(\log N)^2 \quad (3)$$

Thus as a function of input size, Algorithm (4.1) takes exponential time; since $N.(\log N)^2 = \Omega(2^{\log N})$.

References

[1] William Cook, William H. Cunningham, William Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization.* John Wiley & Sons, 1998.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[3] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1987.

[1] Assume that $P > Q$ and that division is implemented through repeated subtraction!