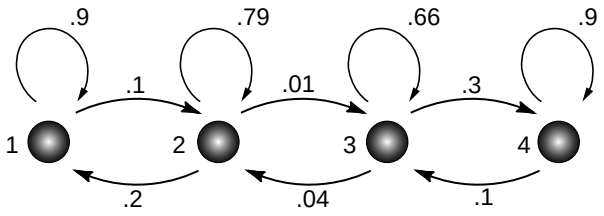
Efficiency Analysis



by Richard Frost

Draft date January 14, 2001

This text is under development by Richard Frost at San Diego State University. Questions and comments should be addressed to:

Richard Frost
Mathematical Sciences
San Diego State University
5500 Campanile Drive
San Diego, CA 92182
frostr@sdsu.edu

Preface

Representation of and operations on basic data structures. Arrays, linked lists, stacks, queues, orthogonal lists, trees; recursion; graphs; hash tables; dynamic storage management and garbage collection.

Column1	Column2
x	A
y	C
z	D

Contents

Chapter 1

Definitions

$$T(n) = O(g(n)) \iff \text{there exists } g(n),\ n_0 \geq 0,\ c > 0 \text{ such that } T(n) \leq c{\cdot}g(n) \text{ for all } n > n_0$$

$$T(n) = \Omega(g(n)) \iff \text{there exists } g(n),\ n_0 \geq 0,\ c > 0 \text{ such that } T(n) \geq c{\cdot}g(n) \text{ for all } n > n_0$$

$$T(n) = \Theta(g(n)) \iff T(n) = O(g(n)) \text{ and } T(n) = \Omega(g(n))$$

$$T(n) = o(g(n)) \iff \text{there exists } g(n),\ n_0 \geq 0,\ c > 0 \text{ such that } T(n) < c{\cdot}g(n) \text{ for all } n > n_0$$

$$T(n) = \omega(g(n)) \iff \text{there exists } g(n),\ n_0 \geq 0,\ c > 0 \text{ such that } T(n) > c{\cdot}g(n) \text{ for all } n > n_0$$

Chapter 2

Using the Definitions in the Left Direction

1. Suppose you have some $T(n)$, for example $T(n) = 3n^2 - n$ which you know is less than or equal to $3n^2$ for all $n \geq 0$ [GAJ86]. Using the definition of $O()$ in the $\Leftarrow$ direction, we have $g(n) = n^2$, $c = 3$, and $n_0 = 0$ so that $T(n) = O(n^2)$.

2. Suppose you have another $T(n)$, for example $T(n) = 5n \cdot \log n + 42$ which you know is less than or equal to $5n \cdot \log n$ for all $n \geq 0$. Using the definition of $\Omega()$ in the $\Leftarrow$ direction, we have $g(n) = n \cdot \log n$, $c = 5$, and $n_0 = 0$ so that $T(n) = \Omega(n \cdot \log n)$.

3. Suppose you have a third $T(n)$ which you know is in **both** $O(n)$ and $\Omega(n)$. Using the definition of $\Theta()$ in the $\Leftarrow$ direction, we have $T(n) = \Theta(n)$.

Chapter 3

Examples

1. Show that

$$T(n) \;=\; 8{\cdot}n^2 - 3{\cdot}n + 5 \;\textit{is in}\; \Omega(n^2)$$

Proof [Knu81]:
We are given $g(n) = n^2$ but we need to find a c and n_0. Notice that

$$8{\cdot}n^2 - 3{\cdot}n + 5 \;>\; 7{\cdot}n^2 \;\textit{for all}\; n \geq 1.$$

This means that if we let

$$c = 7 \;\textit{and}\; n_0 = 0$$

we have

$$T(n) \;=\; \Omega(n^2)$$

2. Show that

$$T(n) \;=\; 8{\cdot}n^2 - 3{\cdot}n + 5 \;\textit{is in}\; O(n^2)$$

Proof:
Again we are given $g(n) = n^2$ but we need to find a c and n_0. Experimenting a little (try plotting) it turns out that

$$8{\cdot}n^2 - 3{\cdot}n + 5 \;<\; 9{\cdot}n^2 \;\textit{for all}\; n \geq 1.$$

This means that if we let

$$c = 9 \;\textit{and}\; n_0 = 0$$

we have

$$T(n) \;=\; O(n^2)$$

3. Show that

$$T(n) \;=\; 8{\cdot}n^2 - 3{\cdot}n + 5 \;\textit{is in}\; \Theta(n^2)$$

From our work above:

$$T(n) \;=\; O(n^2) \;\textit{and}\; T(n) \;=\; \Omega(n^2) \;\Rightarrow\; T(n) \;=\; \Theta(n^2)$$

4. Show that

$$T(n) \;=\; a{\cdot}3^n + b{\cdot}n^3 + t \;\textit{with}\; a > 0 \;\textit{is in}\; O(3^n)$$

Proof:
We are given $g(n) = 3^n$ but as usual we need to find a c and n_0. We have to worry about whether or not b and t are positive or negative. Notice that

$$b{\cdot}n^3 \;\leq |b|{\cdot}n^3 \;\leq\; |b|{\cdot}3^n$$

and

$$t \;\leq |t| \;\leq\; |t|{\cdot}3^n$$

so that

$$a{\cdot}3^n + b{\cdot}n^3 + t \;\leq\; a{\cdot}3^n + |b|{\cdot}3^n + |t|{\cdot}3^n \;=\; (a + |b| + |t|){\cdot}3^n \;\textit{for all}\; n > 0$$

This means that if we let

$$c \;=\; (a + |b| + |t|) \;\textit{and}\; n_0 = 0$$

we have

$$T(n) \;=\; O(3^n)$$

5. Show that

$$T(n) \;=\; a{\cdot}3^n + b{\cdot}n^3 + t \;\textit{with}\; a > 0 \;\textit{is in}\; \Omega(3^n)$$

Proof:
We are given $g(n) = 3^n$ and we need to find a c and n_0 for this case. We again have to worry about whether or not b and t are positive or negative. Let's try to find c. We require:

$$a{\cdot}3^n + b{\cdot}n^3 + t \;\geq\; c{\cdot}3^n \;>\; 0$$

$$\Rightarrow\; a + b{\cdot}\frac{n^3}{3^n} + \frac{t}{3^n} \;\geq\; c \;>\; 0 \; for \; n > 0$$

Consider this last expression in two extreme cases: when b and t are both negative and when b and t are both positive. For $n = 1$ in the first case (both negative) this formula reduces to:

$$a - |b| - |t|$$

This value might be positive or negative.
For $n = 1$ in the second case (both positive) this formula will produce:

$$a + |b| + |t|$$

This value is positive.

Since the terms involving b and t will grow very small as n grows large, we have:

$$a - |b| - |t| \;\leq\; a + b{\cdot}\frac{n^3}{3^n} + \frac{t}{3^n} \;\leq\; a + |b| + |t| \; for \; all \; n > 0$$

We are looking for a $c > 0$ that is less than or equal to the expression (in the middle) for all n above some n_0 to be determined. Notice that

$$\lim_{n\to\infty} a + b{\cdot}\frac{n^3}{3^n} + \frac{t}{3^n} \;=\; a$$

If b and t are both positive, then the expression in the limit is greater than a for all finite n. If b and t are both negative, then the limit approaches a from the left on the number line. **More importantly**, there must be an n_0 such that the expression in the limit exceeds any positive fraction of a. So let's choose c to be some positive fraction of a, say $a/2$, and choose our n_0 to be the first value of n such that the expression is greater or equal to c:

$$a + b{\cdot}\frac{{n_0}^3}{3^{n_0}} + \frac{t}{3^{n_0}} \;\geq\; \tfrac{a}{2} \;=\; c \;> 0$$

$$\Rightarrow a + b \cdot \frac{n^3}{3^n} + \frac{t}{3^n} \;\geq\; \frac{a}{2} \;=\; c \;> 0 \; for \; all \; n > n_0$$

$$\Rightarrow a \cdot 3^n + b \cdot n^3 + t \;\geq\; \frac{a}{2} \cdot 3^n \;=\; c \cdot 3^n \;>\; 0 \; for \; all \; n > n_0$$

So finally,

$$T(n) \;=\; \Omega(3^n)$$

Chapter 4

Analyzing Efficiencies

1. Suppose you work as the IT manager at an accounting firm. Suppose that the nightly accounting runs are composed of processing n client data files. You've been evaluating the run times of accounting runs on your old hardware and some new hardware. You've found:

$$\textit{Old machine}: \quad T(n) = 5 \cdot n$$

$$\textit{New machine}: \quad T(n) = 2 \cdot n$$

It's great that the new machine takes less time to finish the processing than the old machine. However, this just means that the night operators are left with nothing to do for part of their shift. Instead, you'd like to either save money by cutting back on the operator labor hours or (better) do more processing in the same amount of time and increase the corporate profits [Tér88]. Let's look at the latter scenario.

What we'd like to do is: given that n data files took $5 \cdot n$ time to process on the old machine, how many m files can we process in the same amount of time on the new machine? We need:

$$5 \cdot n = 2 \cdot m$$

Right away we get

$$m = 2.5 \cdot n$$

The new machine can handle 2.5 the workload of the old machine.

2. Suppose you are evaluating a data processing task on two machines. The task is known to be $\Theta(n^2)$. You want to know how much work load (in terms of n) you can put on machine 2 so that it takes the same amount of time as machine 1. You've found:

$$\textit{machine \#1}: \quad T(14000) = 3 \textit{ hours}$$

$$\textit{machine \#2}: \quad T(14000) = 2 \textit{ hours}$$

Since the task is $\Theta(n^2)$, we know that $T(n) \approx c \cdot n^2$ for some constant c. We want to find the appropriate load m for machine 2 in terms of the n used for machine 1:

$$\begin{aligned}
\frac{c_1 \cdot 14000^2}{c_2 \cdot 14000^2} &= \frac{3}{2} \\
\Rightarrow \frac{c_1}{c_2} &= \frac{3}{2} \\
c_1 \cdot n^2 &= c_2 \cdot m^2 \\
\Rightarrow \frac{c_1}{c_2} \cdot n^2 &= m^2 \\
\Rightarrow m &= n \cdot \sqrt{1.5}
\end{aligned}$$

3. Suppose you are evaluating a data processing task on two machines. The task is $O(n^3)$ on machine 1, but because of new hardware features is $O(n^2)$ on machine 2. You want to know how much work load (in terms of n) you can put on machine 2 so that it takes the same amount of time as machine 1. You've found:

$$\textit{machine \#1}: \quad T(10000) = 6 \textit{ hours}$$

$$\textit{machine \#2}: \quad T(10000) = 1 \textit{ hour}$$

We want to find the appropriate load m for machine 2 in terms of the n used for machine 1. Since the $T(n)$ functions are different on the two machines, we have to compute the constants separately:

$$\begin{aligned} c_1 \cdot 10000^3 &= 6 \\ \Rightarrow\ c_1 &= 6 \cdot 10^{-12} \\ c_2 \cdot 10000^2 &= 1 \\ \Rightarrow\ c_2 &= 10^{-8} \\ c_1 \cdot n^3 &= c_2 \cdot m^2 \\ \Rightarrow\ \frac{c_1}{c_2} \cdot n^3 &= m^2 \\ \Rightarrow\ m &= \sqrt{n^3 \cdot 6 \cdot 10^{-4}} \end{aligned}$$

In the case of $n = 10000$, $m = 10000 \cdot \sqrt{6} \approx 24500$.
For the case of $n = 100$, $m = \sqrt{6 \cdot 10^5} \approx 775$.

Acknowledgements

Special thanks to ...

Bibliography

[GAJ86] *G-Animal's Journal*, 41(7), July 1986. The entire issue is devoted to gnats and gnus (this entry is a cross-referenced ARTICLE (journal)).

[Knu81] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1981. This is a full BOOK entry.

[Tér88] Tom Térrific. An $O(n \log n / \log \log n)$ sorting algorithm. Wishful Research Result 7, Fanstord University, Computer Science Department, Fanstord, California, October 1988. This is a full TECHREPORT entry.

Index