CS240

Haocen Jiang
haocen.jiang@uwaterloo.ca
Prof. Arne Storjohann

University of Waterloo — Fall 2018

Contents

Introduction and Asymptotic Analysis

Random Access Machine (RAM) Model

- The `random access machine` has a set of memory cells, each of which stores one item of data.
- Any `access` to a memory location takes constant time
- Any `primitive operation` takes constant time.
- The `running time` of a program can be computed to be the number of memory accesses plus the number of primitive operations

Order Notation

O-notation:

- $f(n) \in O(g(n))$ if there exist constant $c > 0$ and $n_o > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_o$
- $f(n)$ grows "no faster than" $g(n)$

Example 1

Prove that $(n+1)^5 \in O(n^5)$
we need to prove that $\exists c > 0, n_o > 0$ s.t. $0 \leq f(n) \leq cg(n)$ $\forall n \geq n_o$
Proof. Note that $n + 1 \leq 2n$ $\forall n \geq 1$ Raise both side the the power of 5 gives:

$$(n+1)^5 \leq 32n^5$$

Thus we have found $c = 32$ and $n_o = 1$

- **Properties**: Assume that $f(n)$ and $g(n)$ are both ***asymptotically non-negative***
  1. $f(n) \in O(af(n))$ for any constant a
     pf. $0 \leq f(n) \leq \frac{1}{a}af(n)$ for all $n \geq n_o := N$
  2. if $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$
     pf. $f(n) \in O(g(n)) \Rightarrow \exists c_1, n_1 > 0$ s.t. $f(n) \leq c_1 g(n)\ \forall n \geq n_1$
     $g(n) \in O(h(n)) \Rightarrow \exists c_2, n_2 > 0$ s.t. $g(n) \leq c_2 h(n)\ \forall n \geq n_2$
     $\therefore f(n) \leq c_1 c_2 h(n)$ for all $n \geq \max(\mathrm{n}_1, \mathrm{n}_2)$
  3. a) $\max(f(n), g(n)) \in O(f(n) + g(n))$
        p.f. $0 \leq max(f(n), g(n)) \leq 1 \cdot [f(n) + g(n)]\ \forall n \geq N$
     b) $f(n) + g(n) \in O(max(f(n), g(n)))$
        p.f. $0 \leq f(n) + g(n) \leq 2 \cdot [max(f(n), g(n))]\ \forall n \geq N$
  4. a) $a_0 + a_1 n + \cdots + a_d n^d \in O(n^d)$ if $a_d > 0$
     b) $n^d \in O(a_0 + a_1 n + \cdots + a_d n^d)$

Ω-notation:

- $f(n) \in O(g(n))$ if there exist constant $c > 0$ and $n_o > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_o$
- $f(n)$ grows "no slower than" $g(n)$

> **Example 2**
>
> $n^3 \log(n) \in \Omega(n^3)$ since $\log(n) \geq 1$ for all $n \geq 3$

Θ-notation:

- $f(n) \in O(g(n))$ if there exist constant $c_1, c_2 > 0$ and $n_o > 0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_o$
- $f(n)$ grows "at the same rate as" $g(n)$
- **Fact:** $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ adn $f(n) \in \Omega(g(n))$

> **Example 3**
>
> $2n^3 - n^2 \in \Theta(n^3)$

o-notation:

- $f(n) \in o(g(n))$ if **for all** constants $c > 0$, there exist $n_o > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_o$
- $f(n)$ grows "slower than" $g(n)$

> **Example 4**
>
> **Claim:** $2010n^2 + 1388n \in o(n^3)$ ***proof.***
> let $c > 0$ be given, then
>
> $$\begin{aligned} 2010n^2 + 1388n &< 5000n^2 \\ &= \left(\frac{5000}{n}\right) n^3 \\ &\leq cn^3 \quad \forall n \geq \frac{5000}{c} \end{aligned}$$

ω-notation:

- $f(n) \in \omega(g(n))$ if **for all** constants $c > 0$, there exist $n_o > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_o$
- $f(n)$ grows "faster than" $g(n)$
- $f(n) \in \omega(g(n)) \Leftrightarrow g(n) \in o(f(n))$

Complexity of Algorithm

Common growth rate

- $\Theta(1)$(*constant complexity*)
- $\Theta(\log n)$(*logarithmic complexity*) e.g. binary search
- $\Theta(n)$(*linear complexity*)
- $\Theta(n\log n))$(*linearithmetic complexity*)e.g. merge sort
- $\Theta(n^2)$(*quadratic complexity*)
- $\Theta(n^3)$(*cubic complexity*)e.g. matrix multiplication
- $\Theta(2^n)$(*quadratic complexity*)

Techniques for Order Notation

Suppose that $f(n) > 0$ and $g(n) > 0$ for all $n \geq n_o$. Suppose that

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)}$$

Then

$$f(n) \in \begin{cases} o(g(n)) & \text{if } L = 0 \\ \Theta(g(n)) & \text{if } 0 < L < \infty \\ \omega(g(n)) & \text{if } L = \infty \end{cases}$$

Example 5

Compare the growth rates of $\log n$ and n^i (where $i > 0$ is a real number).

$$\lim_{n\to\infty} \frac{\log n}{n^i} = \lim_{n\to\infty} \frac{1/n}{in^{i-1}} = \lim_{n\to\infty} \frac{1}{in^i} = 0$$

This implies that $\log n \in o(n^i)$

Example 6 A1P3

Prove of disprove the following statements

(a) $f(n) \notin o(g(n))$ and $f(n) \notin \omega(g(n)) \Rightarrow f(n) \in \Theta(g(n))$

disprove: Counter example, consider $f(n) := n$ and $g(n) := \begin{cases} 1 & \text{n is odd} \\ n^2 & \text{n is even} \end{cases}$.

- For any odd number $n_1 > c$, we have $f(n_1) = n_1 > c = cg(n_1)$, showing that $f(n) \notin O(g(n))$, and therefore, $f(n) \notin o(g(n))$ - Similarly, for any even number $n_1 > 1/c$ we have $cg(n_1) = cn_1^2 > n_1 = f(n_1)$, showing that $f(n) \notin \Omega(g(n))$ and therefore, $f(n) \notin \omega(g(n))$ - However, since $f(n) \notin \Omega(g(n))$, it has to be the case that $f(n) \notin \Theta(g(n))$

(b) $min(f(n), g(n) \in \Theta\left(\frac{f(n)g(n)}{f(n)+g(n)}\right)$

Proof. We will show that $\frac{f(n)g(n)}{f(n)+g(n)} \leq min(f(n), g(n)) \leq 2\frac{f(n)g(n)}{f(n)+g(n)}$ for all $n \geq 1$. The desired result will then follow from the definition of Θ using $c_1 = 1, c_2 = 2$ and $n_0 = 1//$ By assumption, f and g are positive, so $fg/(f+g) = \min(f,g)\max(f,g)/(f+g)$, which is less than $\min(f,g)$ since $\max(f,g)/(f+g) < 1$. Similarly, $\min(f,g) = 2fg/(2\max(f,g)) \leq 2fg/(f+g)$

Example 7

Prove that $n(2 + \sin(n\pi/2)$ is $\Theta(n)$. Note that $\lim_{n\to\infty}(2 + \sin n\pi/2)$ does not exist

Proof. $n \leq n(2 + \sin\frac{n\pi}{2}) \leq 3n$

Relationships between Order Notations

- $f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$
- $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$
- $f(n) \in o(g(n)) \Leftrightarrow g(n) \in \omega(f(n))$
- $f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$
- $f(n) \in o(g(n)) \Rightarrow f(n) \in O(g(n))$
- $f(n) \in o(g(n)) \Rightarrow f(n) \notin \Omega(g(n))$
- $f(n) \in \omega(g(n)) \Rightarrow f(n) \in \Omega(g(n))$
- $f(n) \in \omega(g(n)) \Rightarrow f(n) \notin O(g(n))$

"Maximum" rules

- $O(f(n) + g(n)) = O(max\{f(n), g(n)\})$
- $\Theta(f(n) + g(n)) = \Theta(max\{f(n), g(n)\})$
- $\Omega(f(n) + g(n)) = \Omega(max\{f(n), g(n)\})$

Transitivity

If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in O(h(n))$ If $f(n) \in \Omega(g(n))$ and $g(n) \in O(h(n))$, then $f(n) \in \Omega(h(n))$

Techniques for Algorithm Analysis

Two general strategies are as follows.

- Use Θ-bounds throughout the analysis and obtain a Θ-bound for the complexity of the algorithm.
- Prove a O-bound and a matching Ω-bound separately. Use upper bounds (for O-bounds) and lower bounds (for Ω-bound) early and frequently. This may be easier because upper/lower bounds are easier to sum.

Worst-case complexity of an algorithms:

The worst-case running time of an algorithm A is a function $f : \mathbb{Z}^+ \to \mathbb{R}$ mapping n (the input size) to the ***longest*** running time for any input instance of size n:

$$T_A(n) = \max\{T_A(I) : Size(I) = n\}.$$

Average-case complexity of an algorithm:

The average-case running time of an algprith A is a function $f : \mathbb{Z}^+ \to \mathbb{R}$ mapping n (the input size) to the ***average*** running time over all instances of size n:

$$T_A^{avg}(n) = \frac{1}{|\{I : Size(I) = n\}|} \sum_{\{I:Size(I)=n\}} T_A(I).$$

Notes on O-notation

- It is important not to try to make comparisons between algorithms using O-notations.
- For example, suppose algorithm A_1 and A_2 both solve the same problem, A_1 has worst-case run-time $O(n^3)$ and A_2 has worst-case run-time $O(n^2)$. We **cannot** conclude that A_2 is more efficient

> **NOTE**
> 1.The worst-case run-time may only be achieved on some instances.
> 2. O-notation is an upper bound. A_1 may well have worst-case run-time $O(n)$. If we want to be able to compare algorithms, we should always use Θ-notation.

Example 8

Goal: Use asymptotic notation to simplify run-time analysis.

```
Test1(n)
1.    sum ← 0
2.    for i ← 1 to n do
3.        for j ← i to n do
4.            sum ← sum + (i − j)^2
5.    return sum
```

- size of instance is n
- line1 and line5 execute only once: $\Theta(1)$
- running time proportional to: number of iterations if the j-loop

Direct Method:

$$\#\text{ of iteration} = \sum_{i=1}^{n}(n-i+1) = \sum_{i=1}^{n} i = \frac{n(n+1}{2}$$

$\Rightarrow$ # of iterations of j-loop is $\Theta(n^2)$
$\Rightarrow$ Complexity of Test 1 is $\Theta(n^2)$

Sloppy Method:

$$\#\text{ of iteration} = \sum_{i=1}^{n}(n-i+1) \leq \sum_{i=1}^{n} n = n^2$$

$\Rightarrow$ Complexity of Test 1 is $O(n^2)$

Merge Sort

```
MergeSort(A, ℓ ← 0, r ← n − 1)
A: array of size n, 0 ≤ ℓ ≤ r ≤ n − 1
    if (r ≤ ℓ) then
        return
    else
        m = (r + ℓ)/2
        MergeSort(A, ℓ, m)
        MergeSort(A, m + 1, r)
        Merge(A, ℓ, m, r)
```

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + cn \\
&= 2\left(2T\left(\frac{n}{4}\right) + c\left(\frac{n}{2}\right)\right) + cn \\
&= 4T\left(\frac{n}{4}\right) + c\left(2\left(\frac{n}{2}\right) + n\right) \\
&= 4\left(2T\left(\frac{n}{8}\right) + c\left(\frac{n}{4}\right)\right) + c\left(2\left(\frac{n}{2}\right) + n\right) \\
&= 8T\left(\frac{n}{8}\right) + c\left(4\left(\frac{n}{4}\right) + 2\left(\frac{n}{2}\right) + n\right) \\
&= \ldots \\
&= nc + c\left(n + 2\left(\frac{n}{2}\right) + 4\left(\frac{n}{4}\right) + \cdots + \left(\frac{n}{2}\right)\left(\frac{n}{\frac{n}{2}}\right)\right) \\
&= nc + cn\log(n)
\end{aligned}$$

Some Recurrence Relations

Recursion	resolves to	example
$T(n) = T(n/2) + \Theta(1)$	$T(n) \in \Theta(\log n)$	Binary search
$T(n) = 2T(n/2) + \Theta(n)$	$T(n) \in \Theta(n \log n)$	Mergesort
$T(n) = 2T(n/2) + \Theta(\log n)$	$T(n) \in \Theta(n)$	Heapify ($\rightarrow$ later)
$T(n) = T(cn) + \Theta(n)$ for some $0 < c < 1$	$T(n) \in \Theta(n)$	Selection ($\rightarrow$ later)
$T(n) = 2T(n/4) + \Theta(1)$	$T(n) \in \Theta(\sqrt{n})$	Range Search ($\rightarrow$ later)
$T(n) = T(\sqrt{n}) + \Theta(1)$	$T(n) \in \Theta(\log\log n)$	Interpolation Search ($\rightarrow$ later)

Helpful Formulas

Arithmetic Sequence

$$\sum_{i=0}^{n-1}(a + di) = na + \frac{dn(n-1)}{2} \in \Theta(n^2)$$

Geometric Sequence

$$\sum_{i=0}^{n-1} ar^i = \begin{cases} a\frac{r^n-1}{r-1} \in \Theta(r^n) & \text{if } r > 1 \\ na \in \Theta(n) & \text{if } r = 1 \\ a\frac{1-r^n}{1-r} \in \Theta(1) & \text{if } 0 < r < 1 \end{cases}$$

A few more

$\sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6}$ $\qquad$ $\sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$