Algorithms and Data Structures (CSci 115)

California State University Fresno

College of Science and Mathematics

Department of Computer Science

H. Cecotti

Learning outcomes

- **Complexity**
  - How to measure the performance of an algorithm?
  - Asymptotic notations
    - Θ (Big Theta)
      - Asymptotically **tight** bound
    - O (Big O)
      - Asymptotic **upper** bound
    - Ω (Big Omega)
      - Asymptotic **lower** bound
    - o (little o)
      - **Upper** bound not asymptotically tight
    - ω (little omega)
      - **Lower** bound not asymptotically tight
  - Comparison of functions

Warning

- You will find questions about the definitions corresponding to this class in:
  - **Midterm 1**
  - **Final**
    - → **DO NOT MISS THESE POINTS**
    - Be precise with your answers
    - Be precise with the use of symbols: ∃, ∀ , < , > , <= , >=
- Set
  - Set of numbers
  - Set of **functions**

Different cases

- The **best** case:
  - the inputs of the array were already sorted
- The **worst** case:
  - the inputs of the array were in reverse order
- The **average** case
- Focus on the ***worst-case running time***
  - the longest running time for *any* input of size n.

Different cases

- The worst-case running time
  - An upper bound on the running time for any input.
  - Knowing it provides a **guarantee** that the algorithm will never take any longer!
  - Security:
    - some educated guess about the running time and hope that it never gets much worse.
- For some algorithms
  - the **worst** case occurs fairly often
    - Example: in searching a database for a particular element
    - the searching algorithm's worst case will often occur when the information is **not** present in the database.
    - searches for absent information may be frequent.
- The average case is often roughly as bad as the worst case.

Asymptotic notations

- Defined as functions with domain: $\mathbb{N} = \{0, 1, 2, \ldots\}$.
- Worst case running time: T(n)
  - Defined on int input sizes
- Asymptotic notation
  - To characterize
    - the running times of algorithms (main focus)
    - the amount of space they use
- Different notations
  - To characterize running times
    - Independently of the input

Symbols

- Symbols used in the next slides (reminder)
  - ∃ : there exists
  - { a, b, c} : curly brackets = set
  - | : such that
  - ∀ : for each
    - ∀ $n \geq n_0$: for each n superior or equal to n_0
  - ϵ : belong
    - a ϵ S : a belongs to S
  - Greek letters
    - /!\ if you cannot pronounce the symbol, you cannot remember its meaning later in the definitions
    - Θ (Big Theta)
    - O (Big O)
    - Ω (Big Omega)
    - o (little o)
    - ω (little omega)
    - α : alpha , β: beta , γ: gamma, δ: delta, λ: epsilon
    - µ : mu, π: pi , ρ: rho , σ : sigma , τ : tau , ϕ : phi , ψ : psi

Θ Notation

- Definition
  - $\Theta(g(n))$ the set of functions (Big Theta)
  - $\Theta(g(n))=\{ f(n): \exists \{c_1,c_2,n_0\} \mid 0 \le c_1.g(n) \le f(n) \le c_2.g(n)\ \forall\ n \ge n_0 \}$
- A function f(n) belongs to the set $\Theta(g(n))$
  - If ∃ positive constants c1 and c2 | it can be taken between $c_1.g(n)$ and $c_2.g(n)$, for sufficiently big n.
  - Bounded from above **and** below
- How it is written:
  - $f(n) \in \Theta(g(n)) \rightarrow f(n) = \Theta(g(n))$

O notation

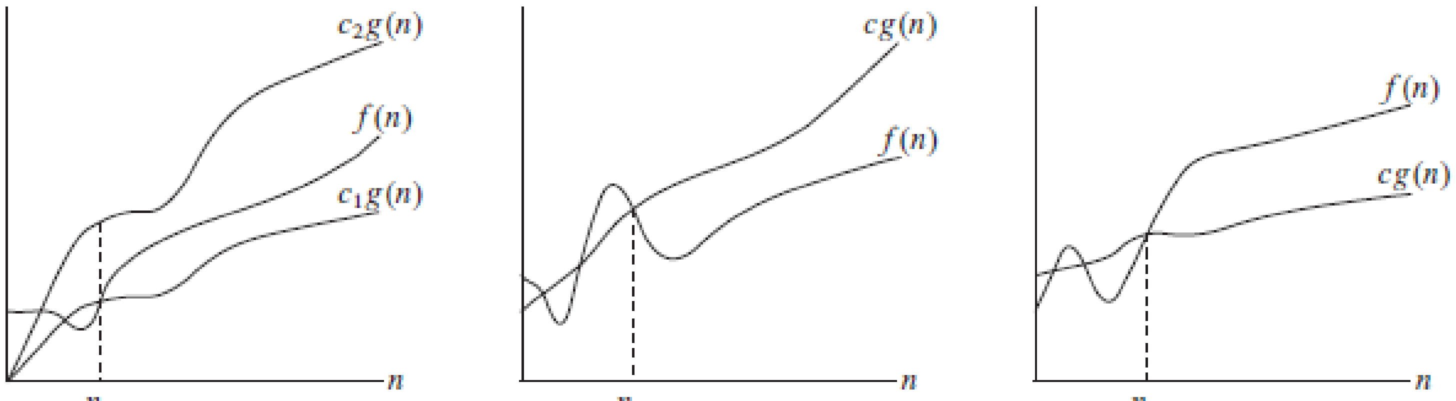
- Goal
  - To give an **upper bound** on a function
  - to bound the worst case running time of an algorithm
    - Within a constant factor
    - O = ‘Big O’
- Only an ***asymptotic upper bound*** $\rightarrow$ O notation
  - $O(g(n))=\{ f(n): \exists \{c,n_0\} \mid 0 \le f(n) \le c.g(n)\ \forall\ n \ge n_0 \}$
- f(n)=O(g(n))
  - to indicate that a function f(n) ϵ the set O(g(n))
  - $f(n)=\Theta(g(n)) \rightarrow f(n)=O(g(n))$
    - Because Θ notation stronger notion than O notation
    - $n=O(n^2)$
- Example
  - Double nested loops
    - a=0; for (i=0;i<n;i++) for (j=0;j<n;j++) a+=i+j;
    - $\rightarrow O(n^2)$

Ω Notation

- Goal
  - To give an lower bound on a function
    - Ω = 'Big Omega'
- Definition
  - $\Omega(g(n))=\{ f(n): \exists \{c,n_0\} \mid 0\leq c.g(n) \leq f(n)\ \forall\ n\geq n_0 \}$
  - → f(n) is **on** or **above** c.g(n)
- Relationship with O and Θ
  - Theorem:
    - For any 2 functions f(n) and g(n), we have f(n)= Θ(g(n)) if and only if
    - f(n)=O(g(n)) and f(n)= Ω(g(n))

Comparisons of the notations

- Graphical examples of the Θ, O, and Ω notations



$f(n) = \Theta(g(n))$ (a) $f(n) = O(g(n))$ (b) $f(n) = \Omega(g(n))$ (c)

o notation

- Goal
  - O notation may be not tight enough
    - $2n^2=O(n^2)$ tight
    - $2n=O(n^2)$ not tight
- Definition
  - $o(g(n))=\{ f(n): \forall c>0 \exists n_0 \mid 0 \leq f(n) < c.g(n) \forall n \geq n_0 \}$
  - $2n^2 \neq o(n^2)$ and $2n=o(n^2)$
- Difference
  - O: there is a c ($\exists c > 0$)
  - o: for each c ($\forall c > 0$)
- Limit
  - $$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

ω notation

- Definition
  - $\omega(g(n))=\{ f(n): \forall\ c>0\ \exists\ n_0 \mid 0\leq c.g(n) < f(n)\ \forall\ n\geq n_0 \}$
  - $f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$
  - ω=little omega
- Example
  - $n^2/2= \omega(n)$ and $n^2/2\neq \omega(n^2)$
- Therefore we have
  - $f(n)=\omega(g(n)) \rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

Comparison of functions

- Transitivity

$$
\begin{aligned}
f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) &\quad \text{imply} \quad f(n) = \Theta(h(n)) \\
f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) &\quad \text{imply} \quad f(n) = O(h(n)) \\
f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) &\quad \text{imply} \quad f(n) = \Omega(h(n)) \\
f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) &\quad \text{imply} \quad f(n) = o(h(n)), \\
f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) &\quad \text{imply} \quad f(n) = \omega(h(n)).
\end{aligned}
$$

- Reflexibility

$$
\begin{aligned}
f(n) &= \Theta(f(n)) \\
f(n) &= O(f(n)) \\
f(n) &= \Omega(f(n))
\end{aligned}
$$

- Symmetry

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n))$$

- Transpose symmetry

$$
\begin{aligned}
f(n) = O(g(n)) &\quad \text{if and only if} \quad g(n) = \Omega(f(n)) \\
f(n) = o(g(n)) &\quad \text{if and only if} \quad g(n) = \omega(f(n)).
\end{aligned}
$$

Analogy with numbers

- Relationships between 2 functions
  - Analogy with the order between 2 numbers
  - f(n) asymptotically smaller/larger (or equal) then g(n)

$f(n) = O(g(n))$	is like	$a \le b$
$f(n) = \Omega(g(n))$	is like	$a \ge b$
$f(n) = \Theta(g(n))$	is like	$a = b$
$f(n) = o(g(n))$	is like	$a < b$
$f(n) = \omega(g(n))$	is like	$a > b$

Relationships between f and g

- Limits
  - Limits of f/g to asymptotic relationships between f and g

$$
\begin{array}{lcl}
\lim_{n\to\infty} f(n)/g(n) \neq 0, \infty & \Rightarrow & f = \Theta(g) \\
\lim_{n\to\infty} f(n)/g(n) \neq \infty & \Rightarrow & f = O(g) \\
\lim_{n\to\infty} f(n)/g(n) \neq 0 & \Rightarrow & f = \Omega(g)
\end{array}
\qquad
\begin{array}{lcl}
\lim_{n\to\infty} f(n)/g(n) = 1 & \Rightarrow & f \sim g \\
\lim_{n\to\infty} f(n)/g(n) = 0 & \Rightarrow & f = o(g) \\
\lim_{n\to\infty} f(n)/g(n) = \infty & \Rightarrow & f = \omega(g)
\end{array}
$$

- Hospital rule

$$\text{If } \lim_{n\to\infty} f(n) = \infty \text{ and } \lim_{n\to\infty} g(n) = \infty, \text{ then } \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}.$$

- Simple rules
  - Polynomial > Log functions
  - Exponential > Polynomial

Examples

- Let $f(n) = 7n + 8$ and $g(n) = n$. Is $f(n) \in O(g(n))$?

For $7n + 8 \in O(n)$, we have to find c and n_0 such that $7n + 8 \leq c \cdot n$, $\forall n \geq n_0$. By inspection, it's clear that c must be larger than 7. Let $c = 8$.

Now we need a suitable n_0. In this case, $f(8) = 8 \cdot g(8)$. Because the definition of $O()$ requires that $f(n) \leq c \cdot g(n)$, we can select $n_0 = 8$, or any integer above 8 – they will all work.

We have identified values for the constants c and n_0 such that $7n + 8$ is $\leq c \cdot n$ for every $n \geq n_0$, so we can say that $7n + 8$ is $O(n)$.

(But how do we know that this will work for every n above 7? We can prove by induction that $7n+8 \leq 8n$, $\forall n \geq 8$.

Examples

▪

Let $f(n) = 7n + 8$ and $g(n) = n$. Is $f(n) \in o(g(n))$?

In order for that to be true, for any c, we have to be able to find an n_0 that makes $f(n) < c \cdot g(n)$ asymptotically true.

However, this doesn't seem likely to be true. Both $7n + 8$ and n are linear, and $o()$ defines loose upper-bounds. To show that it's not true, all we need is a counter–example.

Because any $c > 0$ must work for the claim to be true, let's try to find a c that won't work. Let $c = 100$. Can we find a positive n_0 such that $7n + 8 < 100n$? Sure; let $n_0 = 10$. Try again!

Let's try $c = \frac{1}{100}$. Can we find a positive n_0 such that $7n + 8 < \frac{n}{100}$? No; only negative values will work. Therefore, $7n + 8 \notin o(n)$, meaning $g(n) = n$ is not a loose upper-bound on $7n + 8$.

Examples

Is $7n + 8 \in o(n^2)$?

Again, to claim this we need to be able to argue that for any c, we can find an n_0 that makes $7n+8 < c \cdot n^2$. Let's try examples again to make our point, keeping in mind that we need to show that we can find an n_0 for any c.

If $c = 100$, the inequality is clearly true. If $c = \frac{1}{100}$, we'll have to use a little more imagination, but we'll be able to find an n_0. (Try $n_0 = 1000$.) From these examples, the conjecture *appears* to be correct.

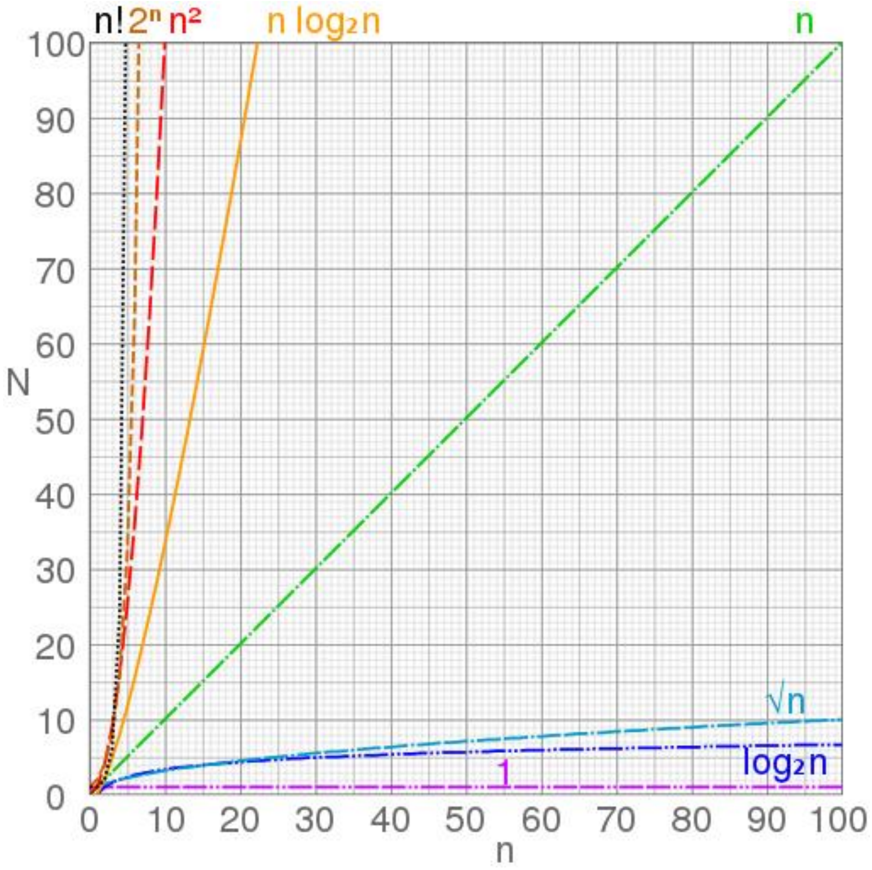
To prove this, we need calculus. For $g(n)$ to be a loose upper-bound on $f(n)$, it must be the case that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Here, $\lim_{n\to\infty} \frac{7n+8}{n^2} = \lim_{n\to\infty} \frac{7}{2n} = 0$ (by l'Hôpital). Thus, $7n + 8 \in o(n^2)$.

Examples

- Compute Big O of
  - $T(n)=5n^3+5n^2+10$
  - $T(n)=8n*\log(6n)+n^2$
  - $T(n)=10n^2*\log(n)+2n^3$
- Exercise:
  - Show $2^n=O(n!)$

Complexity

- Graphical representation:



- With large value of n

The difference is **substantial**

→worth to study Csci 115 ☺

→Big data → Large n

Complexity

- Some figures related to the time

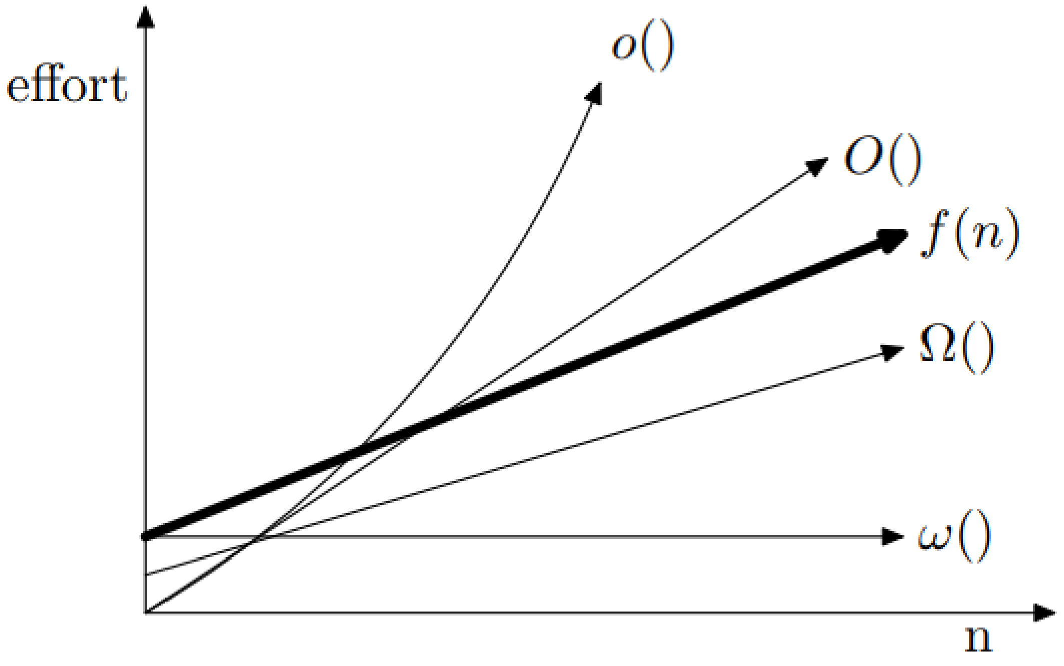
n	O(1)	O($\log_2 n$)	O(n)	O($n \log_2 n$)	O(n^2)
10^2	1 μsec	1 μsec	1 μsec	1 μsec	1 μsec
10^3	1 μsec	1.5 μsec	10 μsec	15 μsec	100 μsec
10^4	1 μsec	2 μsec	100 μsec	200 μsec	10 msec
10^5	1 μsec	2.5 μsec	1 msec	2.5 msec	1 sec
10^6	1 μsec	3 μsec	10 msec	30 msec	1.7 min
10^7	1 μsec	3.5 μsec	100 msec	350 msec	2.8 hr
10^8	1 μsec	4 μsec	1 sec	4 sec	11.7 d

n	O(n^2)	O(2^n)
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec
130	1.7 μsec	18 min
140	2.0 μsec	13 d
150	2.3 μsec	37 yr
160	2.6 μsec	37,000 yr

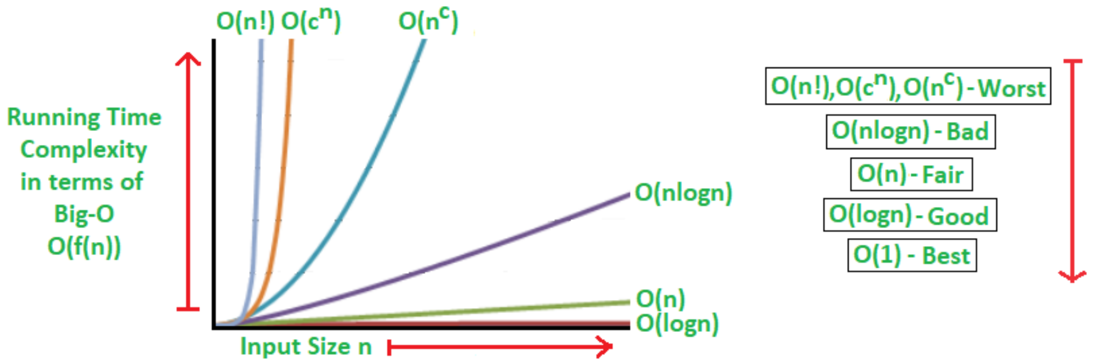
- Know the order of growth:
  - 1 < log n < sqrt(n) < n < n.log n < n^2 < n^3 < 2^n

Conclusion

- Asymptotic notation:



(n= number of elements in the data structure)



Questions ?

- Attendance on Canvas
- Reading:
  - ➢Section 1.3 in Csci115 book
  - ➢Chapter 4: Introduction to algorithms