# CS 240E
# Personal Notes

Marcus Chan

Taught by Therese Biedl

UW CS '25

# Chapter 1:
# Algorithm Analysis

## HOW TO "SOLVE" A PROBLEM

When solving a problem, we should

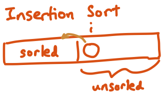
① Write down exactly what the problem is;

eg Sorting Problem

→ given $n$ numbers in an array, put them in sorted order

② Describe the idea;

eg Insertion Sort




Idea: repeatedly move one item into the correct space of the sorted part.

③ Give a detailed description; usually pseudocode.

eg Insertion Sort:

```
for i=1, ..., n-1
    j=i
    while j>0 and A[j-1]>A[j]
        swap A[j] and A[j-1]
        j--
```

④ Argue the correctness of the algorithm.

→ In particular, try to point out loop invariants & variants.

⑤ Argue the run-time of the program.

→ We want a theoretical bound. (using asymptotic notation).

To do this, we count the # of primitive operations.

## PRIMITIVE OPERATIONS

1. In our computer model,
   ① our computer has memory cells
   ② all cells are equal
   ③ all cells are big enough to store our numbers

2. Then, "primitive operations" are $+, -, *, \div$, load & following references.

3. We also assume each primitive operation takes the same amount of time to run.

## ASYMPTOTIC NOTATION

### BIG-O NOTATION: $O(f(n))$

1. We say that "$f(n) \in O(g(n))$" if there exist $c>0$, $n_0>0$ s.t.

$$|f(n)| \leq c|g(n)| \quad \forall n \geq n_0.$$

eg $f(n) = 75n+500$ & $g(n) = 5n^2$, $c=1$ & $n_0=20$

2. Usually, "$n$" represents input size.

### SHOW $2n^2+3n^2+11 \in O(n^2)$

To show the above, we need to find $c, n_0$ such that $0 \leq 2n^2+3n+11 \leq cn^2 \quad \forall n \geq n_0$.

Sol$^{\underline{n}}$. Consider $n_0=1$. Then

$$1 \leq n \Rightarrow 1 \leq n^2 \Rightarrow 11 \leq 11n^2$$
$$1 \leq n \Rightarrow n \leq n^2 \Rightarrow 3n \leq 3n^2$$
$$(+) \quad 2n^2 \leq 2n^2$$
$$\Rightarrow 2n^2+3n+11 \leq 11n^2+2n^2+3n^2 = 16n^2$$

Hence $c=16$ & $n_0=1$, so $2n^2+3n+11 \in O(n^2)$. ∎

### $\Omega$-NOTATION (BIG OMEGA): $f(n) \in \Omega(g(n))$

We say "$f(n) \in \Omega(g(n))$" if there exist $c>0$, $n_0>0$ such that

$$c|g(n)| \leq |f(n)| \quad \forall n \geq n_0.$$

### $\Theta$-NOTATION (BIG THETA): $f(n) \in \Theta(g(n))$

1. We say "$f(n) \in \Theta(g(n))$" if there exist $c_1, c_2>0$, $n_0>0$ such that

$$c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|.$$

2. Note that

$$f(n) \in \Theta(g(n)) \Leftrightarrow f(n) \in O(g(n)) \text{ \& } f(n) \in \Omega(g(n)).$$

### o-NOTATION (SMALL o): $f(n) \in o(g(n))$

1. We say "$f(n) \in o(g(n))$" if for any $c>0$, there exists some $n_0>0$ such that

$$|f(n)| < c|g(n)| \quad \forall n \geq n_0.$$

2. If $f(n) \in o(g(n))$, we say $f(n)$ is "asymptotically strictly smaller" than $g(n)$.

### $\omega$-NOTATION (SMALL OMEGA): $f(n) \in \omega(g(n))$

We say $f(n) \in \omega(g(n))$ if for all $c>0$, there exists some $n_0>0$ such that

$$0 \leq c|g(n)| < |f(n)| \quad \forall n \geq n_0.$$

# FINDING RUNTIME OF A PROGRAM

To evaluate the run-time of a program, given its pseudocode, we do the following:

1. Annotate any primitive operations with just "$\Theta(1)$";
2. For any loops, find the worst-case bound for how many times it will execute;
3. Calculate the big-O run time of the program;
4. Argue this bound is tight (ie show program is also in $\Omega(g(n))$, so runtime $\in \Theta(g(n))$.)

eg insertion sort

```
for i = 1, ..., n-1
    j = i                       Θ(1)
    while j > 0 and A[j-1] > A[j]
        swap A[j] and A[j-1]    Θ(1)   Θ(i)   Θ(n)   Θ(n)
        j--                     Θ(1)
```

Then, let $c$ be a const s.t. the upper bounds all the times needed to execute one line.

So run-time $\leq n \cdot n \cdot c = c \cdot n^2 \in O(n^2)$.

Next, consider the worst pos. case of insertion sort.

$\boxed{10}\boxed{9}\boxed{8}\boxed{7}\boxed{6} \cdots \boxed{0}$

For each $A[i]$, we need $i-1$ swaps.

So

$$\text{runtime} \geq \sum_{i=1}^{n-1} (i-1) = \sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{2} \in \Omega(n^2),$$

and so

runtime $\in \Theta(n^2)$.

## $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L < \infty \Rightarrow f(n) \in O(g(n))$

## <<LIMIT RULE I>> (L1.1(2))

Let $f(n), g(n)$ be such that $\lim_{n\to\infty}\frac{f(n)}{g(n)} = L < \infty$.

Then necessarily $f(n) \in O(g(n))$.

*Proof.* We know $\lim_{n\to\infty}\frac{f(n)}{g(n)} = L$.

$\Rightarrow \forall \varepsilon > 0: \exists n_\varepsilon$ s.t. $\left|\frac{f(n)}{g(n)} - L\right| < \varepsilon \quad \forall n \geq n_\varepsilon$.

We want to show

$\exists c > 0, \exists n_0 \ni \forall n \geq n_0, f(n) \leq c \cdot g(n)$.

Choose $\varepsilon = 1$. Then there exists a $n_1$ s.t.

$\forall n \geq n_1, \quad \left|\frac{f(n)}{g(n)} - L\right| \leq 1$.

$\Leftrightarrow \frac{f(n)}{g(n)} - L \leq \left|\frac{f(n)}{g(n)} - L\right| \leq 1$.

$\Leftrightarrow \frac{f(n)}{g(n)} \leq L + 1$

$\Leftrightarrow f(n) \leq Lg(n) + g(n) \quad$ (since $g(n) > 0$)

Choose $c = L + 1$. Note $f(n), g(n) > 0$, so $L + 1 > 0$, and since $L < \infty$, thus $c < \infty$.

Now, for all $n \geq n_1$, $f(n) \leq c \cdot g(n)$, and so $f(n) \in O(g(n))$. ∎

## $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f(n) \in o(g(n))$

## <<LIMIT RULE II>> (L1.1(1))

Let $f(n), g(n)$. Then $\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0$ iff $f(n) \in o(g(n))$.

## $\lim_{n\to\infty} \frac{f(n)}{g(n)} = L > 0 \Rightarrow f(n) \in \Omega(g(n))$

## <<LIMIT RULE III>> (L1.1(3))

Let $f(n), g(n)$ such that $\lim_{n\to\infty}\frac{f(n)}{g(n)} = L > 0$.

Then necessarily $f(n) \in \Omega(g(n))$.

## $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f(n) \in \omega(g(n))$

## <<LIMIT RULE IV>> (L1.1(4))

Let $f(n), g(n)$ such that $\lim_{n\to\infty}\frac{f(n)}{g(n)} = \infty$.

Then necessarily $f(n) \in \omega(g(n))$.

# OTHER LIMIT RULES

The following are corollaries of the limit rules:

1. $f(n) \in \Theta(f(n))$ } (Identity)
2. $K \cdot f(n) \in \Theta(f(n)) \quad \forall K \in \mathbb{R}$ } (Constant multiplication)
3. $f(n) \in O(g(n)), \; g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$
4. $f(n) \in \Omega(g(n)), \; g(n) \in \Omega(h(n)) \Rightarrow f(n) \in \Omega(h(n))$ } (Transitivity)
5. $f(n) \in O(g(n)), \; g(n) \leq h(n) \; \forall n \geq N \Rightarrow f(n) \in O(h(n))$
6. $f(n) \in \Omega(g(n)), \; g(n) \geq h(n) \; \forall n \geq N \Rightarrow f(n) \in \Omega(h(n))$ } (Dominance)
7. $f_1(n) \in O(g_1(n)), \; f_2(n) \in O(g_2(n)) \Rightarrow f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$
8. $f_1(n) \in \Omega(g_1(n)), \; f_2 \in \Omega(g_2(n)) \Rightarrow f_1(n) + f_2(n) \in \Omega(g_1(n) + g_2(n))$ } (Addition)
9. $h(n) \in O(f(n) + g(n)) \Rightarrow h(n) \in O(\max(f(n), g(n)))$ } (Maximum-rule for $O$)
10. $h(n) \in \Omega(f(n) + g(n)) \Rightarrow h(n) \in \Omega(\max(f(n), g(n)))$ } (Maximum-rule for $\Omega$)

## $f(n) \in P_d(\mathbb{R}) \Rightarrow f(n) \in \Theta(n^d)$ <<POLYNOMIAL RULE>>

Let $f(n) \in P_d(\mathbb{R})$, ie of the form $f(n) = c_0 + c_1 n + \cdots + c_d n^d$.

Then necessarily $f(n) \in \Theta(n^d)$.

## $b > 1; \; \log_b(n) \in \Theta(\log n)$ <<LOG RULE I>>

Let $b > 1$. Then necessarily $\log_b(n) \in \Theta(\log n)$.

> *convention: "log" = "$\log_2$".

*Proof.* Note

$$\lim_{n\to\infty}\frac{\log_b(n)}{\log n} = \lim_{n\to\infty}\frac{\left(\frac{\log(n)}{\log(b)}\right)}{\log(n)} = \lim_{n\to\infty}\frac{1}{\log(b)} > 0,$$

so by Limit Rules 1 & 3, $\log_b(n) \in \Theta(\log n)$. ∎

## $c, d > 0; \; \log^c n \in o(n^d)$ <<LOG RULE II>>

Let $c, d > 0$. Then necessarily $\log^c n \in o(n^d)$,

where $\log^c n \equiv (\log n)^c$.

*Proof.* See that

$$\lim_{n\to\infty}\frac{\ln^k n}{n} \overset{L'H}{=} \lim_{n\to\infty}\frac{k\ln^{k-1}(n)\cdot\frac{1}{n}}{1} \overset{L'H}{=} \cdots \overset{L'H}{=} \lim_{n\to\infty}\frac{k!}{n} = 0,$$

so $\ln^k n \in o(n)$.

Fix $c, d > 0$. Then

$$\lim_{n\to\infty}\frac{\ln^c n}{n^d} = \left(\lim_{n\to\infty}\frac{\ln^{\frac{c}{d}} n}{n}\right)^d \leq \left(\lim_{n\to\infty}\frac{\ln^{\lceil\frac{c}{d}\rceil} n}{n}\right)^d = 0^d = 0.$$

As $\log^c n = \left(\frac{1}{\ln 2}\right)^c \ln^c n$, thus $\lim_{n\to\infty}\frac{\log^c n}{n^d} = \left(\frac{1}{\ln 2}\right)^c \cdot 0 = 0$.

Proof follows from the limit rule. ∎