2.2 Intro. To Algorithm Analysis

Tuesday, February 4, 2025 12:52 PM

"the toolbox"

- Algorithm Analysis:

DEF: To investigate algorithm efficiency with respect to time and space.

- Analytical not Empirical

Suppose:

$\boxed{A}$ $\boxed{B}$

¿which one is better?

Analytical:
we will model performance using a mathematical function
the "runtime" function

$R(\ )$: input :- the size of the input data.
output :- the number of "operations" performed by the program

operations: $\leftarrow$ $+$ $*$ $=$ $[\,]$ $<$ $>$

$R_A(n)$ vs $R_B(n)$

To compare performance we compare functions

E.G.

```
FUNCTION Sum( A[0..n-1])
    s ← 0          - 1
    i ← 0          - 1
    while i < n do
        s ← s + A[i]   - 3
        i ← i + 1      - 2
    return s - 1
```

$n(3+2+1) = 6n$

$$R_{sum}(n-1) = 6n+3$$

- BUT: there are many inputs of size n

- BUT: there are many inputs of size n
  which one to choose?

E.G

```
FUNCTION Find ( A[0..n-1], x )
    FOR i ← 0 to n-1 DO
        IF x = A[i] THEN
            RETURN i
    RETURN -1
```

$\left.\begin{matrix}\text{IF } x = A[i] \text{ THEN} \\ \text{RETURN } i\end{matrix}\right\} 2 \quad \Big\} n \cdot 4$

$\text{Find}(\boxed{1|2|3}, 2)$

$\text{Find}(\boxed{1|2|3}, 4)$

$R_{worst\,Find}(n) = 4 \cdot n + 1$

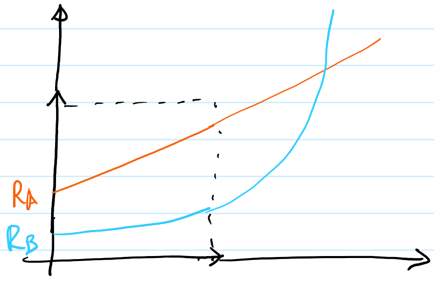
☆ - Worst_input :- the one that takes most steps.

- best_input :- optimistic underestimate.
- all inputs, and then average.
- amortize cost over all inputs

- BUT: which size n to use?
  - no specific values !!!

we want to know what happens to the runtime function as n becomes really large.

- Graphs are deciefull



- We are going to care about a function's "rate-of-growth" or "order-of-growth"
  what happens $n \to \infty$

- When you work with "rate-of-growth"
  you ignore constant factors.

$R_{sum}(n-1) = \cancel{6}n + \cancel{3}$ $\quad R_{worst\,Find}(n) = \cancel{4} \cdot n + \cancel{1}$

- ASYMPTOTIC NOTATION

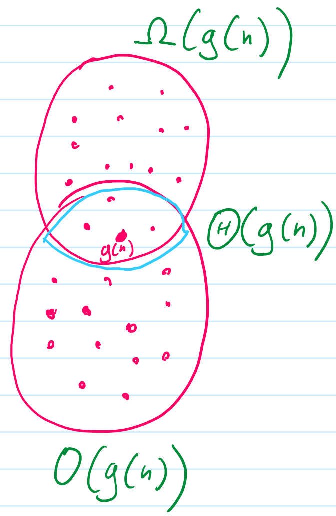
(informal definition)

Let $g(n)$ be a function:

$O(g(n))$ is the set of all functions with lower or same "rate-of-growth"

$\Omega(g(n))$ is the set of all functions with larger or same "rate-of-growth"

$\Theta(g(n))$ is the set of functions with the same "rate-of-growth"



E.g. Functions in $O(n^2)$

- $n \in O(n^2)$ • $27n+3 \in O(n^2)$ • $3n^2 \in O(n^2)$ • $3\cancel{27}n^2 + \underline{1024n} + \underline{10{,}972} \in O(n^2)$

  - note: ignore constant factors.

more.

$n^3 \in \Omega(n^2)$ • $3n^2 \in \Omega(n^2)$ • $2^n \in \Omega(n^2)$ • $n! \in \Omega(n^2)$

• $3n^2 \in \Theta(n^2)$

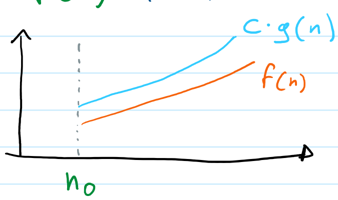
DEF:

We say that $f(n) \in O(g(n))$ if there exist constants C and n_0 such that for all $n > n_0$

$$f(n) \leq C \cdot g(n)$$

- no restriction on C and n_0
- C is what allows us to ignore constant factors.

in English: $f(n)$ is bound from above by a multiple of $g(n)$



E.G.

- $n^2 \in O(3n^2 + n)$

need to show that $n^2 \leq C \cdot (3n^2 + n)$ for $n > n_0$

$C = 1$ $\qquad n^2 \leq 3n^2 + n$ $\qquad$ for $n > 1$

$n_0 = 1$ $\qquad n^2 \leq 3n^2$

$C = 1$ $\qquad$ $n^2 \le 3n^2 + n$ $\qquad$ for $n > 1$
$n_0 = 1$ $\qquad$ $n^2 \le 3n^2$ ✓

E.G.

- $3n^2 + n \in O(n^2)$

need to show that $3n^2 + n \le C \cdot (n^2)$ for $n > n_0$

$C = 4$ $\qquad$ $3n^2 + n \le 4n^2$
$n_0 = 1$ $\qquad$ $\cancel{3n^2} + n \le \cancel{3n^2} + 1n^2$

$$n \le n^2$$ ✓

DEF:
we say that $f(n) \in \Omega(g(n))$
if there exists constants C and n_0
such that for every $n > n_0$

$$f(n) \ge C \cdot (g(n))$$

In English: $f(n)$ is bounded from below by a multiple of $g(n)$

DEF:
we say that $f(n) \in \Theta(g(n))$
if there exists constants C_1, C_2 and n_0
such that for every $n > n_0$

$$C_1 \cdot (g(n)) \le f(n) \le C_2 \cdot (g(n))$$

In English: $f(n)$ is bound from above and below by multiples of $g(n)$

In English: $f(n)$ and $g(n)$ have the same "order-of-growth"

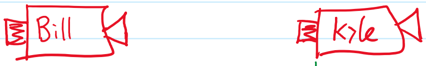
Note: IF $f(n) \in O(g(n))$ AND $f(n) \in \Omega(g(n))$ THEN $f(n) \in \Theta(g(n))$

- BASIC EFFICIENCY CLASSES:

n^2 $\quad$ $3n^2+n$ $\quad$ $4n^2$ $\quad$ $2n^2-n$ $\quad$ $5n^2+40n+12$ $\quad$ n^2

When comparing functions:

When comparing functions:



¿ which one is better?

$$R_{Bill}(n) = 5n^3 + 2n^2 \qquad R_{Kyle}(n) = \frac{1}{2}n^2 + 3.$$

we place functions in their class, and then compare classes.

The Complexity Hierarchy

	$n!$	"Factorial" :- generate all permutations of a collection of size n.
	2^n	"exponential" :- generate all subsets of a collection of size n.
	$\vdots$	
	n^5	
	n^4	
$R_{Bill}(n) \longrightarrow$	n^3	"cubic" :- 3 nested loops.
$R_{Kyle}(n) \longrightarrow$	n^2	"cuadratic" :- 2 nested for loops.
	$n \cdot \log n$	"linear-logarithmic" :- "divide and conquer" algorithms.
	n	"linear" :- do one thing for every element in input
	$\log n$	"logarithmic" :- split input in half, and continue on one half.
	1	"constant" :- programs without loops.

E.g. searching in a sorted array

n = number of elements

- How many times can you cut n elements in half?
- Same as $2^k = n$? $\quad k = \log_2 n$

- MORE MATH : USEFUL PROPERTIES OF ORDER-OF-GROWTH

what about running Bill after kyle.

$$R_{BK}(n) = R_{Bill}(n) + R_{Kyle}(n)$$

- If you have functions $f_1(n)$ $f_2(n)$ $g_1(n)$ $g_2(n)$

and you know that $\cdot\, f_1(n) \in O(g_1(n))$
and $\cdot\, f_2(n) \in O(g_2(n))$

then $f_1(n) + f_2(n) \in O\left(\max\{g_1(n), g_2(n)\}\right)$

e.g. $R_{Bill}(n) \in O(n^3)$ $\quad R_{Kyle} \in O(n^2)$

$R_{BK}(n) \in O\left(\max\{n^3, n^2\}\right)$

$R_{BK}(n) \in O(n^3)$

- A polynomial of degree k is $O(n^k)$

- Limits and the order-of-growth

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 :- f(n) \text{ has a smaller order-of-growth than } g(n) \\ C :- f(n) \text{ has the same order-of-growth as } g(n) \\ \infty :- f(n) \text{ has a larger order-of-growth than } g(n) \end{cases}$$

these allows us to use convenient results from calculus

- L'Hopital rule:

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

- Stirling's formula:

$$n! \approx \sqrt{2\cdot\pi\cdot n}\cdot\left(\frac{n}{e}\right)^n \quad \text{for large values of } n$$

—o— EOF