(II) Asymptotic Analysis

Sorting Problem

> Input: seq of n numbers $a_1, a_2, \cdots, a_n$
>
> Output: a permutation $a_1', a_2', \cdots, a_n'$ s.t.
>
> $$a_1' \le a_2' \le \cdots \le a_n'$$

$i = n$: cost

$n-1$

$\left.\begin{matrix} 2(n-1) \\ 2(n-1) \end{matrix}\right\} 5(n-1)$

$i = n-1$:

$5(n-2)$ u.c.

$\vdots$

$i = 7$

```
Alg: Bubble-Sort
Input: A = [a_1, ..., a_n]
    for i := n to 2:
        for j = 1 to i-1:
            if A[j] > A[j+1]:
                swap A[j] and A[j+1]
Output: the modified A
```

A : 53, 12, 15, 0, 4, 97, 22

j=1 : 12, 53, 15, 0, 4, 97, 22

j=2 : 12, 15, 53, 0, 4, 97, 22

j=3 : 12, 15, 0, 53, 4, 97, 22

$\vdots$ 53 97

j=6 : 12, 15, 0, 4, 53, 22, 97

▷ Correctness

Invariant:
at the end of i-th iteration,
A[i] to A[n] are sorted.

(At the end of -th iteration)

A : 53, 12, 15, 0, 4, 97, 22

i=7 : 12, 15, 0, 4, 53, 22, 97

i=6 : 12, 0, 4, 15, 22, 53, 97

i=5 : 0, 4, 12, 15, 22, 53, 97

⋮

▷ Efficiency?

Running time?

$$f(n) = \sum_{i=n}^{2} 5(i-1) = 5 \cdot \frac{(n-1+1)(n-1)}{2} = \frac{5n(n-1)}{2}$$

$$\cancel{\tfrac{5}{2}}n^2 \quad \cancel{-\tfrac{5}{2}n}$$

$$= O(n^2)$$

Measure Running Time:

Desired properties:

- A func $f(n)$ scales with n. → e.g. len of the array A.
- Platform / hardware indep.
- not depend on any single input inst.

Exmp. Alg: Trick-Sort

Input: $A = (a_1, \cdots, a_n)$

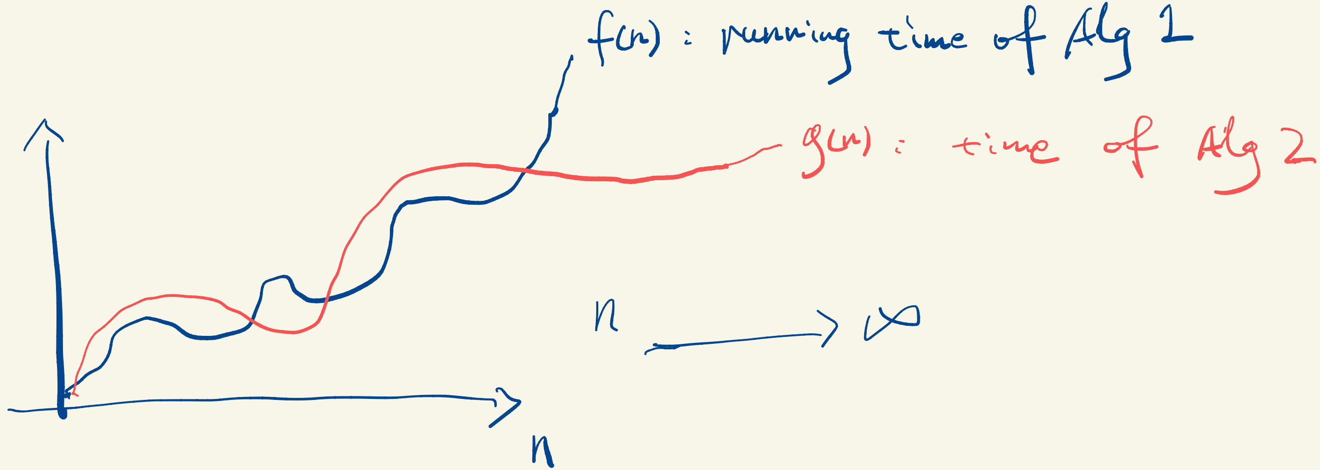
if $A = (10, 9, 8, 7, 6, 5)$:

→ output $(5, 6, 7, 8, 9, 10)$

else:

call Bubble-Sort.

- $f(n)$ v.s $g(n)$. → not depend on any particular n.



- $f(n)$ covers the worst case input:

Alg : Trick-Sorting 2.

If A is sorted
 return A

else:
 call Bubble-Sort(A)

- Faster than bubble sort when A is sorted.
- Doesn't affect worst-case comparison.

▷ How to obtain $f(n)$?

- # of unit operations
- RAM (Random Access Machine)
- Unit operation:
  - Read & Write to any position in the memory.
  - Addition, Subtraction, Multiplication, etc.
  - Every numbers involved can be represented in e.g 64 bits.

▷ Big-O notation

- Def: Given $\forall$ func $g(n), f(n)$. say $f(n) = O(g(n))$ if

▷ $\boxed{\exists\, c > 0,\ n_0 > 0,\ \text{s.t. } f(n) \leq c \cdot g(n),\ \forall n \geq n_0}$.

$\Longleftrightarrow$ $\boxed{\lim_{n\to\infty} f(n)/g(n) \leq C, \text{ for some constant } C \geq 0}$

- Assume $\lim_{n\to\infty} f(n) \geq 0$, $\lim_{n\to\infty} g(n) \geq 0$

- Exmp: $f(n) = 3n^2 + 2n$, $g(n) = n^2 - 10n$.

Claim: $f(n) = O(g(n))$

(1) Pf: Let $c = 100$, $n_0 = 2000$, $f(n) = 3n^2 + 2n \leq c \cdot g(n) = 100n^2 - 1000n$.

$\Rightarrow c \cdot g(n) - f(n) = 97n^2 - 1002n \geq 0 \quad \forall n \geq n_0 = 2000$ □

(2) Pf: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{3n^2 + 2n}{n^2 - 10n} = \lim_{n\to\infty} \frac{3 + 2 \cdot \frac{1}{n}}{1 - \frac{10}{n}} = 3 \leq 4$ □

▷ Ω- notation and Θ-notation

· Def: Given $\forall$ func $g(n)$, $f(n)$. say $f(n) = \Omega(g(n))$ if

$$\boxed{\exists\, c > 0,\ n_0 > 0.\ \text{s.t. } c\cdot f(n) \geq g(n),\ \forall n > n_0}$$

$$\Longleftrightarrow \boxed{\exists\, c > 0,\ \text{s.t. } \lim_{n\to\infty} f(n)/g(n) \geq c}$$

· Exmp: $f(n) = 3n^2 + 2n$, $g(n) = n^2 - 10n$.

claim: $f(n) = \Omega(g(n))$

▷ · Pf: Let $c = 100$, $n_0 = 1$. $c \cdot f(n) = 300n^2 + 200n \geq g(n) = n^2 - 10n$

for any $n > 1$.

▷ Pf: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 3 \geq 1 = c > 0$

$\square$

$f(n) = O(g(n))$, $f(n) = \Omega(g(n))$

Def : $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $\Omega(g(n))$

O	Ω	Θ	o	ω
$\leq$	$\geq$	$=$	$<$	$>$

▷ little-o and ω notation: stricter version of O and Ω.

- Def: Given $\forall$ func $g(n)$, $f(n)$. say $f(n) = o(g(n))$ if

$$\boxed{\lim_{n \to \infty} f(n)/g(n) = 0}$$

E.g. $f(n) = 3n$, $g(n) = 0.1n^2$

- Def: Given $\forall$ func $g(n)$, $f(n)$. say $f(n) = \omega(g(n))$ if

$$\boxed{\lim_{n \to \infty} g(n)/f(n) = 0}$$

E.g. $f(n) = 0.1n^2$, $g(n) = 3n$

O	Ω	Θ	o	ω
$\leq$	$\geq$	$=$	$<$	$>$

① $f = O(g) \iff g = \Omega(f)$

② $f = \Theta(g) \iff f = O(g)$ and $f = \Omega(g)$

③ $f = o(g) \Rightarrow f = O(g)$, $\not\Leftarrow$

④ $f = \omega(g) \Rightarrow f = \Omega(g)$, $\not\Leftarrow$

e.g. $f = n^2 - 10$, $g = 2n^2$

$f = O(g)$ $\quad f \neq o(g)$

$f = \Omega(g)$ $\quad f \neq \omega(g)$

▷ Typical functions

① Polynomials: n^c, c const. e.g.: $2n$, $3n^2-4$, $5n^7-3n^5+2n^2$

② Logarithmic: $\log_b^c n := (\log_b n)^c$. b, c, const >0

e.g. $\log_2 n$, $\log_{10} n$, $\log_2^2 n$. etc.

③ Exponential: c^n, c, const >1. e.g. 2^n, 3^n, e^n.

④ Hybrid of ①②③:

$n^{\log n}$, 2^{n^2}, $n^{\frac{n^2}{\log n}}$, $\frac{n^2}{\log n}$, $n+\log n$, $n\log n$

Comparison of different type of functions

① Logarithmic < Polynomial < Exponential.

$$(\log n)^{100} = o(n^{0.0001})$$

$$n^{1000} = o(2^{0.0001n})$$

② Polynomial v.s. Polynomials

- Only highest order matter.

e.g. $\boxed{3}n^2 \cancel{+2n-1}$ v.s. $0.1n^2$, $\quad \lim\limits_{n\to\infty} \frac{3n^2+2n-1}{0.1n^2} = \lim\limits_{n\to\infty} \frac{3n^2}{0.1n^2}$

$$3n^2+2n-1 = \Theta(0.1n^2)$$

③ Exponential: c_1^n v.s. c_2^n

If $c_1 < c_2 \Rightarrow c_1^n = o(c_2^n)$

④ Hybrid:

e.g. $n\log n$ v.s. n^2

Claim: $n\log n = o(n^2)$

$n \cdot \log n$ v.s. $n \cdot n$ $\qquad \lim_{n\to\infty} \frac{n\log n}{n\cdot n} = \lim_{n\to\infty} \frac{\log n}{n} = 0.$

Q: $n^{\log n}$ v.s. 2^n ?

▷ Conventions

- Ignore lower-order terms

$f(n) = O(g(n))$. $g(n) = 3n^2 - n + \log n$

$f(n) = n^2$

$f(n) = O(3n^2 \cancel{- n + \log n}) = O(3n^2)$

- Ignore constant factors

$f(n) = O(3n^2) = O(n^2)$

- $f(n) = O(g(n))$ is "tight" $\Longleftrightarrow$ $f(n) = \Theta(g(n))$

$= \Omega(g(n))$

- $\cancel{O(g(n)) = f(n)}$