Algorithms

- Algorithm: sequence of well-defined computational procedures that takes some value(s) as input and produces some value(s) as output

- Algorithm as a tool for solving a well-specified computational problem
- Any input, meeting the conditions of the problem statement is called an instance of the problem, which is later used to compute a problem
- An algorithm must be correct, by correct it means that the algorithm solves a computational problem
- Choosing an efficient algorithm is as important as choosing a fast hardware

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$	0	5.9	11.8				
$\sqrt{n}$	1	7.79	60				
n	1	60	3600				
$n \lg n$	0	354					
n^2	1	3600					
n^3	1	21600					
2^n	1	1.15×10^{19}					
$n!$	1	8.32×10^{81}					

Insertion Sort

- In-place Algorithm

→ Algo that does not use extra space and produces an output in the same memory that contains data, can use constant extra space for variables

- INSERTION SORT (A):

```
for j ← 2 to A.length
    key ← A[j]
    i ← j-1

    while i > 0 and A[i] > key
        A[i+1] ← A[i]
        i ← i-1
    A[i+1] ← key
```

- Situations to use in
- Already sorted / Nearly sorted data
- Small Datasets

→ Simple

→ Low constant factors

- Systems with minimal memory usage

→ Space complexity of $O(1)$

- Online sorting

→ Sequential data

- Worst-Case : $O(n^2)$
- Average-Case: $\Theta(n^2)$
- Best-case : $\Omega(n)$
- In-place Algo
- Space Complexity: $O(1)$

- RAM Model (Random Access Machine)
- instructions executed one after another, no concurrent execution
- An abstraction, which allows us to ignore hardware specifications, and purely focuses on algorithmic logic only

- To measure efficiency as a "standard":
- Each step takes 1 unit of time in RAM model
- Constants are dropped, and only order of growth is measured

  → Machine differences : a faster computer might execute more steps in the time a slower machine executes 1.

  → Compiler differences: Different languages require different number of machine instructions for the same line of code

- Asymptotic Analysis

  → Standard measurement focuses on performance change as input size approaches infinity

Q- Apply RAM Model and calculate Running Time (Time)

	Individual Cost	Repetition	Total
$x \leftarrow 0$	c_1	1	c_1
for $i \leftarrow 1$ to n	c_2	$n+1$	$c_2(n+1)$
temp $\leftarrow i+1$	c_3	n	$c_3(n)$
for $j \leftarrow 1$ to n	c_4	$n(n+1)$	$c_4(n^2+n)$
If (j mod $2 = 0$) Then	c_5	n^2	$c_5(n^2)$
$x \leftarrow x +$ temp	c_6	$n^2(k)$ $0 \le k \le 1$	$c_6(n^2k)$
Else	c_7	$n^2(1-k)$	$c_7[n^2(1-k)]$
$x \leftarrow x - 1$	c_8	$n^2(1-k)$ $0 \le k \le 1$	$c_8[n^2(1-k)]$

$$T(n) = c_1 + c_2(n+1) + c_3(n) + c_4(n^2+n) + c_5(n^2) + c_6(n^2k) + c_7[n^2(1-k)] + c_8[n^2(1-k)]$$

$$= n^2\underbrace{(c_4 + c_5 + kc_6 + c_7 - kc_7 + c_8 - kc_8)}_{} + n\underbrace{(c_2 + c_3 + c_4)}_{} + \underbrace{(c_1 + c_2)}_{}$$

$$= n^2(c_4 + c_5 + c_7 + c_8 + k(c_6 - c_7 - c_8)) + n(c_2 + c_3 + c_4) + (c_1 + c_2)$$

$T(n) = An^2 + Bn + C$, a quadratic-time algorithm

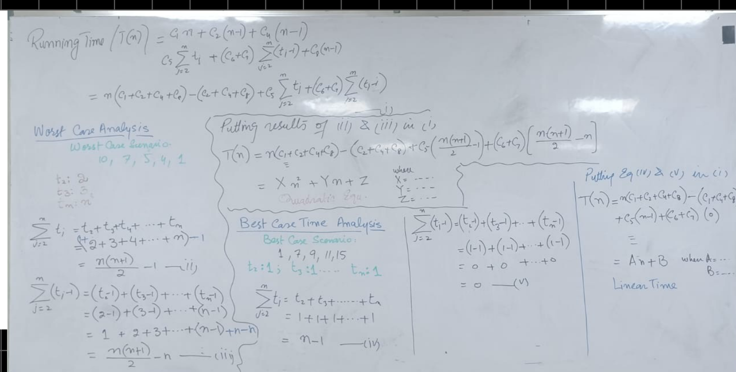
where $A = c_4 + c_5 + c_7 + c_8 + k(c_6 - c_7 - c_8)$

$B = c_2 + c_3 + c_4$

$C = c_1 + c_2$

INSERTION-SORT(A)

		cost	times
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$



Growth of Functions

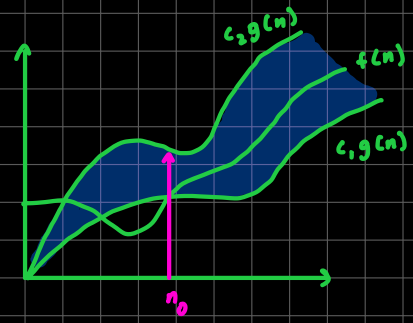
- Asymptotic notation for expressing algorithm's running time.
- Applied on functions
- can be used to represent the amount of space algorithm use

- $\Theta(g(n))$

→ $\Theta(g(n)) = \{ f(n) :$ there exist +ve constants $c_1, c_2,$ and n_0 such that
$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \geq n_0 \}$

- It means a function exists that belongs to the set $o(g(n))$ if there exist +ve constants c_1 and c_2 such that it can be sandwiched b/w $c_1 g(n)$ and $c_2 g(n)$ for large n

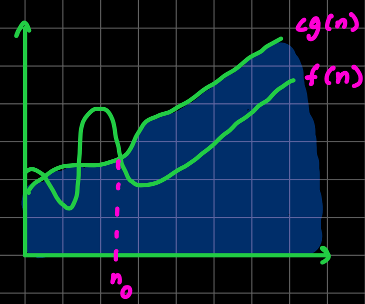
- $f(n) = \theta(g(n)) \approx f(n) \in \theta(g(n))$] same case for all other notations



- Asymptotically tight bound
- Algo grows at a precise rate, sandwiched average case where best and worst case growth rates match

- $O(g(n))$

- $O(g(n)) = \{ f(n) :$ there exist +ve constants c and n_0 such that
$0 \le f(n) \le c\, g(n)$ for all $n \geq n_0 \}$



- Gives asymptotic upper bound
- Algo will not grow any faster than the rate, ceiling on growth

- $\Omega(g(n))$

- $\Omega(g(n)) = \{ f(n) :$ there exist +ve constants c and n_0 such that
$0 \le c g(n) \le f(n)$ for all $n \geq n_0 \}$

- if $f(n) = O(g(n))$ and $f(n) = \Theta(g(n))$, only then $f(n) = \Omega(g(n))$