Asymptotic Analysis

It is just a way to define a limiting behavior
{ n → ∞ }

Time taken =	$c * \text{(number of commands used)}$		
Time taken =	$c * \{$		$\}$
	100	10,000	10^6

$c * n$ ← $c * n^2$

$[1 < \log n < n^{\frac{1}{2}} < n^2 < 2^n < n!]$

Common complexity.

$n + 2 \approx n^2$
$n + 100 \approx n$] Drop-off the constants

$(2^n + n) \approx 2^n$
$n^2 + n \approx n^2$] Drop the lower order terms

$n^2 + m \approx n^2 + m$

$n^2 + n * m \approx n^2 * m$

Big-O:

Gives an upper bound to a given algorithm

$T(n) = c * n$

TC: $O(n)$
$= O(n^2)$
$= O(n!)$

Big-Omega:

Gives a lower-bound to the algo.

TC: $\Omega(n)$
$\Omega(2\log(n))$
$\Omega(1)$

Big Theta:

Gives the exact bound.

TC: $\Theta(n)$

$\Theta(1, 10^9) \to \Theta(2^n)$

Best / Worst / Avg. Case.

Linear Search

[1, 5, 4, 2, 3, 2]
target = 4
target = 7

Best-case: $O(1)$

Worst-case: $O(n)$

Avg. case: $O(n)$

1st idx → 1
1st idx → 2
3rd idx → 3
⋮
$(n-1)^{th}$ idx → n
not found → n+1

(n+1) cases

Worst-case → (n+1)

Total work done = $1 + 2 + 3 + 4 - - - - + (n+1)$

$= \frac{(n+1)(n+2)}{2}$

$\text{Avg} = \frac{\frac{(n+1)(n+2)}{2}}{(n+1)} = \frac{n+2}{2} = O(n)$

Note: For "a-lot" of algos, best & avg case complexities are the same.
But not always: eg Quick Sort.
↳ Worst case = $O(n)$
Avg. case = $O(n \log n)$

Space Complexity.

→ Input space
+
Auxiliary space ←
+
Output space.

Input space

size = n^2

```
vector<int> func(vector<vector<int>> mat) {
    vector<int> res(m);
    vector<int> temp(n^2);
    - - -
    - - -
    return res;
}
```

v/s space = $O(n^2)$
o/p space = $O(m)$
← Aux space = $O(m^2)$

$SC = O(n^2 + m)$

$i = 0$: $O(B)$
$i = 1$: $O(B)$
⋮
$i = A-1$: $O(B)$

$= O(A * B)$

```
void function(n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0 * 1; j < n; j++) {
            cout << i << j << endl;
        }
    }
}
```

$i = 0$: $n-1$
$i = 1$: $n-2$
$i = 2$: $n-3$
⋮
$i = n-2$: 1
$i = n-1$: 0

Total work = $1 + 2 + 3 - - - - + n-1$

$= \frac{(n-1)(n)}{2} = O(n^2)$

$= \frac{n^2}{2} - \frac{n}{2}$

no. of chars in string = m
time to print = $O(m)$

chars → int
strings → list of int #

int: [- - - 32 bits - - - -] ×2

["...30...", "...40...", "...9...", "...10..."]

$O(m * \text{max_length of any string in the input})$

[acb, dfg, bca, ddb]
s1 = [abc, dfg, abc, bdd]
s2 = [abc, abc, bdd, dfg]

abcd, abcf
abcd, abcf

$s1 \to (m \log m + m * \log n)$
$s2 \to (m \log m + m * \log n)$

s1 → $O(m \log m)$
s2 → $O(m \log n)$
] → $O(m(\log m + \log n))$
$O(m \log m + m \log n)$

Time to sort a list of numbers = $O(n \log n)$
Time to compare 2 numbers = $O(1)$
Hence, no. of comparisons in sorting = $O(n \log n)$
Time to compare 2 strings = $O(m)$
Time to sort a list strings = $O(m(n \log n))$