Asymptotic Analysis → A way to define a dominating behavior or algorithms.

↓

Assume inputs → ∞.

Time — Space.

1) $O(n \log n)$ 1) $O(n^2)$

Rules to follow.

1) Drop-off the constants.
2) Drop-off the insignificant terms.

$T(n) = n^2 + \cancel{7}$ $T(n, m) = n^2 + m + \cancel{500}$.

$T(n) = n^2 + \cancel{9} + \cancel{500}$ $T(n, m) = n^2 + m + \cancel{500}$

$T(n, m) = n^2 + m^2 + \cancel{9} + \cancel{9} + \cancel{500}$.

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$

$< O(2^n) < O(n!)$.

$\underbrace{2 \cdot 2 \cdot 2 \cdot 2 \cdots \cdots}_{n}$

$\underbrace{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdots \cdots}_{n}$

Big-O → Defines an upper bound of the algo.

$T(n) = n + c$

⇒ $O(n)$ / $O(n \log n)$ / $O(n^2)$ ... / $O(n!)$

Big-Θ → Exact Bound.

$T(n) = n + c$.

⇒ $\Theta(n)$

Big-Omega → Lower Bound.

$T(n) = n + c$.

⇒ $\Omega(n)$ / $\Omega(\log n)$ / $\Omega(1)$

$\Theta(1.17^n)$ → $O(2^n)$.

Best / Worst / Avg. Case.

```
(arr, target)
{
    for (int i=0; i<n ; i++) {
        if (arr[i] == target)
            return true;
    }
    return false;
}
```

Best-case : TC → $O(1)$.

Worst-case : TC → $O(n)$.

Avg. case :

Case	Work done
0th	1
1st	2
2nd	3
⋮	⋮
(n-1)th	n
not found	n+1

Avg → $\dfrac{\text{Total work done}}{n+1}$

$= \dfrac{1+2+3+ \cdots + (n+1)}{(n+1)}$

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

⇒ $O(n)$

Space Complexity → I/P Space + [Auxiliary / Extra space +].

O/P Space.

```
vector<int> func ( vector<int> arr) {
    vector<int> res(n);
    vector<int> temp(n^2);
    ⋮
    return res;
}
```

size = n.

I/P space = $O(n)$.

O/P space = $O(n)$.

Aux. Space = $O(n^2)$

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

$i = 0 \to n-1$

$i = 1 :$

$i = 0 :$ $j : 1 \to n-1$ ⇒ $n-1$

$i = 1 :$ $j : 2 \to n-1$ ⇒ $n-2$.

$i = 2 :$ $j : 3 \to n-1$ ⇒ $n-3$

⋮

$i = n-1$ $j : n \to n-1$ ⇒ 0

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

$= \dfrac{(n-1)(n)}{2}$ ⇒ $O(n^2)$.

---

print int.

$O(1)$

print string.

$O(\text{len})$

int ~ char.

Vector<int> ~ string.

```
void function(int n, string s) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            cout << s << endl; // printing a string
        }
    }
}
```

$i = 0.$

$j = 1 \to O(m)$

$j = 2 \to O(m)$

$j = 3 \to O(m)$

$j = 4 \to O(m)$

⋮

$j = n-1 \to O(m)$

→ Total = $(n-1)\, m = O(nm)$

- - - → TC = $O(n^2 m)$