Time complexity will be 1 of the below.

$$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots$$
$$\cdots < 2^n < 3^n < \cdots < n^n$$

Big Oh The function $f(n) = O(g(n))$ if there exist +ve constants C and n_0 such that

$$0 \leq f(n) \leq C \cdot g(n)$$ where ~~for~~ all $n \geq n_0$

eg. $f(n) = 2n+3$

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

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

$2n+3 \leq 10n$ even $2n+3 \leq 7n$

↳ multiply by n

or

$\therefore \boxed{2n+3 \leq 2n+3n}$

$2n+3 \leq 2n^2+3n^2$ also true.

$2n+3 \leq 5n^2$ ← true.

$\boxed{2n+3 \leq 5n}$ → Here $C=5$

↳ $O(n)$ $g(n)=n$

$\therefore \boxed{f(n) = O(n)}$

↳ I can write. $f(n) = O(n^2)$

as $\boxed{2n+3 \leq 5n^2}$ is also true.

But we consider lowest limit.

$1 < \log n < \sqrt{n} < n$ ← lower bound

$n \log n < n^2 < n^3 < \cdots < 2^n < 3^n$ ← upper bound

Big Omega (Ω)

$f(n) = 2n+3$

$f(n) = \Omega(n)$

If there exists c and n_0 such that $c \cdot g(n) \leq f(n)$ $\forall$ $n \geq n_0$

$c \cdot g(n) \leq 2n+3$

$1 \times n \leq 2n+3$

$c = 1 \Rightarrow g(n) = n$

$\therefore f(n) = \Omega(n)$

lower bound

True $\Rightarrow f(n) = \Omega(\log n)$

But not $f(n) = \Omega(n^2)$ X

Big Q Theta.

$f(n) = 2n+3$

$c_1 \cdot g(n) \leq f(n) < c_2 \cdot g(n)$

$1 \times n \leq 2n+3 \leq 5n$

$c_1 g(n)$ $\qquad$ $c_2 g(n)$

$f(n) = Q(n)$

exact

not below

not above

(2) $T(n) = 10n^2 + 2n + 1$

Big Oh $f(n) \leq c \cdot g(n)$

$10n^2 + 2n + 1 \leq 10n^2 + 2n^2 + n^2$

$\therefore 10n^2 + 2n + 1 \leq 13n^2$ → $c \cdot g(n)$

$\therefore \boxed{f(n) = O(n^2)}$

Big Omega $c \cdot g(n) \leq f(n)$

$10 \times n^2 \leq 10n^2 + 2n + 1$

$c = 1 \quad g(n) = n^2$

$\boxed{f(n) = \Omega(n^2)}$ ← lower bound

Big Θ $c_1 g(n) \leq f(n) \leq c_2 g(n)$

$10 \times n^2 \leq 10n^2 + 2n + 1 \leq 13n^2$

$\boxed{f(n) = \Theta(n^2)}$

Mathematical background for Algorithm Analysis *

① $\sum_{i=1}^{n} 1 = 1+1+1+\cdots+1 = n = O(n)$

② $\sum_{i=1}^{n} i = 1+2+3+\cdots+n$

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

$= O(n^2)$

③ $\sum_{i=1}^{n} i^k = 1+2^k+3^k+\cdots+n^k = \frac{n^k}{k+1}$

$= O(n^{k+1})$

④ $\sum_{i=1}^{n} k^i = k+k^2+k^3+\cdots+k^n = \frac{k^{n+1}}{k-1}$

$= O(k^n)$

⑤ $\sum_{i=k}^{n} 1 = n-k+1$

Big Oh

① show that $4n^2 = O(n^3)$

$0 \le f(n) \le cg(n)$

$0 \le f(n) \le cn^3$

$0 \le 4n^2 \le cn^3$ divide by n^3

$0 \le 4/n \le c$ for $n = 1$

c will be max.

$\therefore c = 4$

To determine value of n_0

$0 \le 4/n \le c$

$0 \le 4/n_0 \le 4$

$\therefore 0 \le 4/4 \le$

$\therefore 0 \le \frac{4}{4} \le n_0$

$\therefore n_0 = 1$

$\therefore 0 \le 4n^2 \le 4n^3 \quad \forall \quad n \ge n_0 = 1$

(2) Show that $400n^3 + 20n^2 = O(n^3)$

$0 \le f(n) \le c.g(n)$

$0 \le 400n^3 + 20n^2 \le c.g(n)$

$0 \le 400n^3 + 20n^2 \le cn^3$

Dividing by n^3

$$\frac{0}{n^3} \le \frac{400n^3 + 20n^2}{n^3} \le c$$

$$\therefore \boxed{0 \le 400 + 20/n \le c} \quad \text{— (1)}$$

$20/n \to 0$ when $n = \infty$

$20/n \to \max(20)$ i.e. when $n = 1$

$$\therefore \boxed{0 \le 400 + 20/1 \le c} \quad \text{— (2)}$$

$$\therefore \boxed{c = 420}$$ ↺ put in (1)

$$0 \le 400 + 20/n_0 \le 420$$

$$-400 \le 20/n_0 \le 20$$

$$-20 \le 1/n_0 \le 1$$

$$-20n_0 \le 1 \le n_0 \to \text{i.e. } \boxed{n_0 = 1}$$

Hence $0 \le 400n^3 + 20n^2 \le 420n^3 \; \forall \; n \ge n_0 = 1$

Time complexity range

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \cdots < 2^n < 3^n < \cdots < n^n$

categories of algorithms

Running complexity

1. Constant time algorithm → $O(1)$
2. Linear time algorithm → $O(n)$
3. Logarithmic time algorithm → $O(\log n)$
4. Polynomial time Algo → $O(n^k)$ where $k \geq 1$
5. Exponential time Algo → $O(2^n)$

Framwork for analysis of Non-recursive functions.

- Finding complexity of non-recursive algorithm is simpler than that of recursive algorithm.

Step (1) Determine size of problem / input
Step (2) Find primitive / elementary operation.
Step (3) Find count of primitive operations for best, worst and average case.
Step (4) Simplify the summation by dropping multiplicative and divisive constants of highest degree polynomial term in sum.