ASYMPTOTIC NOTATION

ROMI FADILLAH RAHMAT

- Having the expressions for best case, average case and worst case, for all the three cases we need to identify the upper bound, lower bounds.
- In order to represent these upper bound and lower bounds we need some syntax. Let us assume that the given algorithm is represented in the form of function $f(n)$.
- Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.
  - O Notation
  - Ω Notation
  - θ Notation

BIG-O NOTATION

- This notation gives tight the upper bound of the given function. Generally we represent it as $f(n) = O(g(n))$. That means, at larger values of n , the upper bound of $f(n)$ is $g(n)$.
- Example : For example, if $f(n) = n^4 + 100n^2 + 10n + 50$ is the given algorithm, then n^4 is $g(n)$. That means $g(n)$ gives the maximum rate of growth for $f(n)$ at larger values of n .
- It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.

OMEGA - Ω NOTATION

- The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time.
- It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

THETA-Θ NOTATION

- This notation decides whether the upper and lower bounds of a given function (algorithm) are same or not.
- The average running time of algorithm is always between lower bound and upper bound. If the upper bound (O) and lower bound (Ω) gives the same result then θ notation will also have the same rate of growth.

GUIDELINES FOR ASYMPTOTIC ANALYSIS

- There are some general rules to help us in determining the running time of an algorithm. Below are few of them.
- Loops : The running time of a loop is, at most, the running time of the statements inside the loop (including tests) multiplied by the number of iterations.

```
// executes n times
for (i=1; i<=n; i++)
{
        m = m + 2; // constant time, c
}
```

- Total Time = a constant c x n = c n = O(n).

- Nested Loops : Analyze from inside out. Total running time is the product of the sizes of all the loops.

```
//outer loop executed n times
for (i=1; i<=n; i++)
{
    // inner loop executed n times
    for (j=1; j<=n; j++)
    {
        k = k+1; //constant time
    }
}
```

Total Time = $c \times n \times n = cn^2 = O(n^2)$

- Consecutive Statements
- If then Else Statement
- Logarithmic Complexity

COMMONLY USED LOGARITHMS AND SUMMATIONS

Logarithms

$$\log x^y = y \log x$$
$$\log n = \log_e^n$$
$$\log xy = \log x + \log y$$
$$\log^k n = (\log n)^k$$
$$\log \log n = \log(\log n)$$
$$\log \frac{x}{y} = \log x - \log y$$
$$a^{\log_b^x} = x^{\log_b^a} \quad \log_b^x = \frac{\log_a^x}{\log_a^b}$$

Arithmetic series

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

Geometric series

$$\sum_{k=1}^{n} x^k = 1 + x + x^2 \ldots + x^n = \frac{x^{n+1} - 1}{x - 1} (x \neq 1)$$

Harmonic series

$$\sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \ldots + \frac{1}{n} \approx \log n$$

Other important formulae

$$\sum_{k=1}^{n} \log k \approx n \log n$$

$$\sum_{k=1}^{n} k^p = 1^p + 2^p + \cdots + n^p \approx \frac{1}{p+1} n^{p+1}$$