Asymptotic Analysis & Notation — Comprehensive Notes

1) Motivation

- When analyzing algorithms, we want to compare **growth rates of functions** that describe time or space complexity.
- Input size: $n \in \mathbb{N}$ (natural numbers).
- Complexity functions map $n \to \mathbb{R}^+$ (nonnegative reals).
  → Negative time/space is meaningless (no algorithm runs in −2 steps or uses negative memory).
- Focus: **asymptotic growth** → behavior as $n \to \infty$.

---

2) What is asymptotic notation?

- A **system for comparing functions** (growth rates).
- Analogy: comparing numbers
  - 20 ≥ 15
  - 14 = 14
  - 12 ≤ 18
- Here: compare functions **f(n), g(n)** for large n.

---

3) First Notation: Big-O (O)

Intuition

- Big-O ≈ "≤" (upper bound).
- **f(n) = O(g(n))** means:
  Beyond some threshold N_0, $f(n)$ grows no faster than a constant multiple of $g(n)$.

Formal Definition

$f(n) = O(g(n)) \Leftrightarrow$
$\exists$ constants $K > 0$, $N_0 \geq 0$ such that
$\forall\ n \geq N_0$: $f(n) \leq K \cdot g(n)$.

Notes

1. Only matters for **large n** (ignore small inputs).
2. Constants are ignored (we can multiply g by any positive K).
3. f may initially be larger than g, but eventually g overtakes f permanently.

Diagram (in words)

- Plot f and g.
- At some **overtake point N_0**, $K \cdot g(n)$ lies above f(n) forever.

---

4) Examples for Big-O

1. **$f(n) = 0.5n^2$, $g(n) = 0.1n^3$**
   - Eventually n^3 overtakes n^2.
   - So $f(n) = O(g(n))$.
2. **Huge vs tiny constants**

- $f(n) = 2 \times 10^{10} n^2$, $g(n) = 0.000000002 n^3$
- Despite constants, `n³` always wins for large n.
- So still `f(n) = O(g(n))`.

3. **Linear vs linear**
   - $f(n) = 2.2n$, $g(n) = 1.5n$
   - f is always above g. But with K = 10, g can be scaled to overtake f.
   - So `f = O(g)` and also `g = O(f)` → they grow at the same rate.

---

5) Second Notation: Big-Ω (Ω)

Intuition

- Big-Ω ≈ "≥" (lower bound).
- **f(n) = Ω(g(n))** means:
  Beyond some N_0, f is always above a constant multiple of g.

Formal Definition

```
f(n) = Ω(g(n)) ⇔
∃ constants K > 0, N₀ ≥ 0 such that
   ∀ n ≥ N₀: f(n) ≥ K · g(n).
```

Diagram (in words)

- f eventually lies **above** K·g(n).
- Mirror image of Big-O.

---

6) Third Notation: Big-Θ (Θ)

Intuition

- Big-Θ ≈ "=" (tight bound).
- **f(n) = Θ(g(n))** means:
  f is both O(g) and Ω(g).
  (g grows neither faster nor slower than f asymptotically).

Formal Definition

$f(n) = \Theta(g(n)) \Leftrightarrow$
$\exists\ k_1, k_2 > 0$ and $N_0 \geq 0$ such that
$\forall\ n \geq N_0: k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n).$

Diagram (in words)

- f is "sandwiched" between $k_1 g(n)$ and $k_2 g(n)$, beyond some point N_0.
- Captures asymptotic equivalence.

---

7) Rules of Thumb

- **Ignore constants**: factors like 0.5, 2, 100 don't matter.
- **Ignore lower-order terms**:
  - $f(n) = 2n^2 + 3n + 5 \rightarrow$ dominated by n^2.
- **Compare leading terms only**:
  - n^3 dominates $n^2 \log n$, which dominates n^2, which dominates $n \log n$, etc.

---

8) Worked Examples

Example Set

- $F(n) = 2n^2 + 3 \log n + 4\sqrt{n} + 15$ → **dominated by n^2**
- $G(n) = 200\sqrt{n} + 15 \log n + 14n^{1.5}$ → **dominated by $n^{1.5}$**
- $H(n) = n^2 + 2n + 3 \log \log n$ → **dominated by n^2**
- $L(n) = n^3 + 15n^2 + 2.5n$ → **dominated by n^3**
- $M(n) = 4n^2 + 13n^2 \log n$ → **dominated by $n^2 \log n$**

Comparisons

- F vs G: n^2 vs $n^{1.5}$ → **$F = \Omega(G)$, $G = O(F)$** (not Θ).
- F vs H: both n^2 → **$F = \Theta(H)$**.
- L vs M: n^3 vs $n^2 \log n$ → **$M = O(L)$, $L = \Omega(M)$** (not Θ).
- G vs M: $n^{1.5}$ vs $n^2 \log n$ → **$G = O(M)$, $M = \Omega(G)$** (not Θ).

---

9) Practical Guide for Algorithms

- To classify f(n):
    1. Find **leading term**.
    2. Discard constants and lower terms.
    3. Compare leading terms.

Growth rate hierarchy (common functions):

```
1 < log n < √n < n < n log n < n² < n³ < … < 2^n < n!
```

-

---

10) Summary

- **Big-O:** asymptotic upper bound (≤).
- **Big-Ω:** asymptotic lower bound (≥).
- **Big-Θ:** tight bound (=).
- Constants and additive terms are irrelevant.
- Use leading terms to compare functions.
- Typical workflow:
    - Identify leading term.
    - Compare growth orders.
    - Classify as O, Ω, or Θ.

✅ These notes cover everything from **definitions, intuition, rules, and examples** through to **formal statements and diagrams-in-words**, following your transcript closely.