**BIG-O NOTATION**

- **$O$ notation:**
  
  \[ f(n) = O(g(n)) \text{ if there exist constants } c > 0 \text{ and } n_0 > 0 \text{ such that} \]
  
  \[ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0. \]

  Here, the complexity of $f$ is **not higher** than the complexity of $g$.

- **$\Omega$ notation:**
  
  \[ f(n) = \Omega(g(n)) \text{ if there exist constants } c > 0 \text{ and } n_0 > 0 \text{ such that} \]
  
  \[ 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0. \]

  Here, the complexity of $f$ is **not lower** than the complexity of $g$.

- **$\Theta$ notation:**
  
  \[ f(n) = \Theta(g(n)) \text{ if there exist constants } c_1, c_2 > 0 \text{ and } n_0 > 0 \text{ such that} \]
  
  \[ c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0. \]

  Here, $f$ and $g$ have **the same complexity**.

- **$o$ notation:**
  
  \[ f(n) = o(g(n)) \text{ if for all constants } c > 0, \text{ there exists a constant } n_0 \text{ such that} \]
  
  \[ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0. \]

  Here, $f$ has **lower complexity** than $g$.

- **$\omega$ notation:**
  
  \[ f(n) = \omega(g(n)) \text{ if for all constants } c > 0, \text{ there exists a constant } n_0 \text{ such that} \]
  
  \[ 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0. \]

  Here, $f$ has **higher complexity** than $g$.

- **$\Theta$ notation:**
  
  \[ f(n) = \Theta(g(n)) \text{ if for all constants } c_1, c_2 > 0, \text{ there exists a constant } n_0 \text{ such that} \]
  
  \[ c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0. \]

  Here, $f$ and $g$ have **the same complexity**.

- **$O + \Omega = \Theta$:**
  
  \[ O(f(n)) + \Omega(g(n)) = \Theta(f(n) + g(n)). \]

**Example:**

- If $f(n) = n^2 + 3n + 1$ and $g(n) = 2n^2$, then $f(n) = O(g(n))$ because

  \[ n^2 + 3n + 1 \leq 2n^2 \text{ for all } n \geq 1. \]

**Exercise:**

- Consider $f(n) = 3n^2 + 5n + 2$ and $g(n) = n^2$. Determine whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. Justify your answer.
EXERCISE

• Which of the following are true?
  • $n^2 \in \Theta(n^3)$
  • $n^3 \in o(n^2)$
  • $n^3 \in \omega(n^2)$
  • $\log n \in o(n)$
  • $n \log n \in \Omega(n)$
  • $n \log n^2 \in \omega(n \log n)$
  • $n \in \Theta(n \log n)$

EXERCISE

• Which of the following are true?
  • $n^2 \in \Theta(n^3)$ YES
  • $n^3 \in o(n^2)$ YES
  • $n^3 \in \omega(n^2)$ NO
  • $\log n \in o(n)$ YES
  • $n \log n \in \Omega(n)$ YES
  • $n \log n^2 \in \omega(n \log n)$ NO
  • $n \in \Theta(n \log n)$ NO

Intuitively, we have the following correspondences between order notation and growth rates:

- $f(n) \in O(g(n))$ means the growth rate of $f$ is $\leq$ the growth rate of $g$
- $f(n) \in \omega(g(n))$ means the growth rate of $f$ is $< \omega$ the growth rate of $g$
- $f(n) \in \Theta(g(n))$ means the growth rate of $f$ is $= \omega$ the growth rate of $g$
- $f(n) \in \Omega(g(n))$ means the growth rate of $f$ is $\geq \omega$ the growth rate of $g$

Relationships between Order Notations

- $f(n) \in \Theta(g(n)) \Rightarrow g(n) \in \Theta(f(n))$
- $f(n) \in O(g(n)) \Rightarrow g(n) \in \Theta(f(n))$
- $f(n) \in \omega(g(n)) \Rightarrow g(n) \in \Omega(f(n))$
- $f(n) \in o(g(n)) \Rightarrow g(n) \in \omega(f(n))$
- $f(n) \in \Omega(g(n)) \Rightarrow f(n) \in O(g(n))$
- $f(n) \in \omega(g(n)) \Rightarrow f(n) \in \Omega(g(n))$

This is included for your notes

Prove that $f(n) \in \Theta(g(n))$ implies $g(n) \in \Theta(f(n))$.

Proof: Suppose $f(n) \in \Theta(g(n))$. Then there exist constants $c_1, c_2, n_0$ such that

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

if $n \geq n_0$. Thus

$0 \leq (1/c_2) f(n) \leq g(n) \leq (1/c_1) f(n)$

if $n \geq n_0$. Define $c'_1 = 1/c_2, c'_2 = 1/c_1$ and $n'_0 = n_0$. Then

$0 \leq c'_2 f(n) \leq g(n) \leq c'_1 f(n)$

if $n \geq n'_0$.

WORKED EXERCISES

1. Let $f(n) = n^2 - 7n - 30$. Prove from first principles that $f(n) \in O(n^2)$.
2. Let $f(n) = n^2 - 7n - 30$. Prove from first principles that $f(n) \in \Omega(n^2)$.
3. Suppose $f(n) = n^2 + n$. Prove from first principles that $f(n) \notin O(n)$.
EXAMPLE 1: \( f(n) = n^2 - 7n - 30 \)

- Want to prove (WIP) from first principles: \( f(n) \in \Omega(n^2) \)
  - More formally: there exist constants \( c > 0, n_0 > 0 \)
    such that for all \( n \geq n_0 \), we have \( 0 \leq f(n) \leq cn^2 \)
- Pick a value for \( c \); how about 1?
- Let’s visualize \( c = 1 \)

Seems plausible that \( c = 1 \) will work

Let’s prove this algebraically

\[ 0 \leq f(n) \leq cn^2 \]

So, the claim holds with \( c = 1, n_0 = 10 \)

EXAMPLE 2: \( f(n) = n^2 - 7n - 30 \)

- WIP from first principles: \( f(n) \in \Omega(n^2) \)
  - More formally: there exist constants \( c > 0, n_0 > 0 \)
    such that for all \( n \geq n_0 \), we have \( 0 \leq cn^2 \leq f(n) \)
- Solution:
  - Pick a value for \( c \).
  - How about 1?
  - Must show \( n^2 \leq n^2 - 7n - 30 \)
  - Impossibly! \( c = 1 \) is too large.
  - Let’s try \( c = \frac{1}{2} \)

\[ 0 \leq \frac{1}{2}n^2 \leq n^2 - 7n - 30 \]

Result: \( c = \frac{1}{2}, n_0 = 18 \) works!

EXAMPLE 3: \( f(n) = n^2 + n \)

- WIP from first principles \( f(n) \in O(n) \). Formally:
  - \( f(n) \in \Theta(n) \)
  - \( \forall c > 0, n_0 > 0 \) \( \forall n \geq n_0 \) \( f(n) < cn \) or \( f(n) > cn \)
  - Consider any arbitrary \( c > 0, n_0 > 0 \)
  - We find some \( n \geq n_0 \) such that \( n^2 + n < 0 \) or \( n^2 + n > cn \)
  - \( n^2 + n > cn \) if \( n^2 + n - cn > 0 \) if \( n(n + 1 - c) > 0 \)
  - For \( n \geq n_0 > 0 \), this holds if \( n + 1 - c > 0 \), equivalently \( n > c = 1 \)
  - So, \( n = \text{max}(c, n_0) \) will suffice

you vs. the guy she tells you not to worry about

\[ O(n^2) \quad O(n \log n) \]

COMPARING GROWTH RATES
All of the identities shown hold only if the limits exist.

**Limit of an Exponential Function**

\[
\lim_{x \to a} b^f(x) = b^{\lim_{x \to a} f(x)}.
\]

**Limit of a Logarithm of a Function**

\[
\lim_{x \to a} \log_b f(x) = \log_b \left( \lim_{x \to a} f(x) \right)
\]

(Where base \( b > 0 \))

---

**L'Hospital's Rule**

- Often we take the limit of \( \frac{f(x)}{g(x)} \) where both \( f(n) \) and \( g(n) \) tend to \( \infty \), or both \( f(n) \) and \( g(n) \) tend to \( 0 \).
- Such limits require L'Hospital's rule.
  - This rule says the limit of \( \frac{f(n)}{g(n)} \) in this case is the same as the limit of the derivative.
  - In other words, \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{f'(n)}{g'(n)} \).
**Using the Limit Method: Exercise 1**

- Compare growth rate of $n^2$ and $n^2 - 7n - 30$.
  
  \[ \lim_{n \to \infty} \frac{2^n - n^2}{n^2} = 1 \]
  
  \[ \text{So } n^2 - 7n - 30 \in \Theta(n^2) \]

**Using the Limit Method: Exercise 2**

- Compare growth rate of $(\ln n)^2$ and $n^{1/2}$.
  
  \[ \lim_{n \to \infty} \frac{\ln n}{n^{1/2}} = \lim_{n \to \infty} \frac{\ln n}{n^{1/2}} = \frac{\ln n}{n^{1/2}} = \frac{\ln n}{n^{1/2}} = 0 \]
  
  \[ \text{So, } (\ln n)^2 \in \Theta(n^{1/2}) \]

**Additional Exercises**

- Compare the growth rate of the functions $(3 + (-1)^n)n$ and $n$.
  
  \[ \lim_{n \to \infty} \frac{3 + (-1)^n}{n} = \lim_{n \to \infty} \frac{3 + (-1)^n}{n} = 0 \]

- Compare the growth rates of the functions $f(n) = n|\sin \pi n/2| + 1$ and $g(n) = \sqrt{n}$.

**Algebra of Order Notations**

- "Monotonic" rules: Suppose that $f(n) > 0$ and $g(n) > 0$ for all $n \geq n_0$.
  
  \[ k \cdot f(n) = O(f(n)) \]
  \[ (f(n) + g(n)) = \Theta(f(n)) \]
  \[ f(n) \cdot g(n) = \Omega(g(n)) \]

- "Summation" rules: Suppose $f$ is a finite set. Then
  
  \[ \sum_{i \in f} k = \sum_{i \in f} k_i = k \cdot \sum_{i \in f} \tilde{k} \]
  \[ \sum_{i \in f} \tilde{k} = \sum_{i \in f} \tilde{k} \cdot \sum_{i \in f} \tilde{k} \]
  \[ \sum_{i \in f} \tilde{k} = \sum_{i \in f} \tilde{k} \cdot \sum_{i \in f} \tilde{k} \]
Summation rules are commonly used in loop analysis.
Example:
\[ \sum_{i=1}^{n} O(i) = O \left( \sum_{i=1}^{n} i \right) = O \left( \frac{n(n+1)}{2} \right) = O(n^2). \]

**SEQUENCES**

Arithmetic sequence:
\[ \sum_{d=0}^{n-1} (a + di) = na + \frac{dn(n - 1)}{2} \in \Theta(n^2). \]

Geometric sequence:
\[ \sum_{l=0}^{n-1} ar^l = \begin{cases} a \frac{r^n - 1}{r - 1} \in \Theta(r^n) & \text{if } r > 1 \\ na \in \Theta(n) & \text{if } r = 1 \\ a \frac{1 - r^n}{1 - r} \in \Theta(1) & \text{if } 0 < r < 1 \end{cases} \]

**SEQUENCES CONTINUED**

Arithmetic-geometric sequence:
\[ \sum_{d=0}^{n-1} (a + di)r^l = \frac{a}{1 - r} \frac{(a + (n - 1)d)r^n}{1 - r} + \frac{dr(1 - r^{n-1})}{(1 - r)^2} \]
provided that \( r \neq 1 \).

Harmonic sequence:
\[ H_n = \sum_{i=1}^{n} \frac{1}{i} \in \Theta(\log n) \]

**Miscellaneous Formulae**

\[ n! \in \Theta(n^{n/2} e^{-n}) \]

\[ \text{limit } \in \Theta(n \log n) \]

Another useful formula is
\[ \sum_{i=1}^{n} 1 = \frac{n^2}{2} \]
which implies that
\[ \sum_{i=1}^{n} \frac{1}{i^2} \in \Theta(1) \]
A sum of powers of integers when \( c \geq 1 \):
\[ \sum_{i=1}^{n} i^c \in \Theta(n^{c+1}) \]

**LOGARITHM RULES**

**Logarithm Formulae**
1. \( \log_b xy = \log_b x + \log_b y \)
2. \( \log_b \frac{x}{y} = \log_b x - \log_b y \)
3. \( \log_b \frac{1}{x} = -\log_b x \)
4. \( \log_b x^y = y \log_b x \)
5. \( \log_b a = \frac{1}{\log_a b} \)
6. \( \log_b a = \frac{\log_b c}{\log_b a} \)
7. \( a \log_b c = \log_b a^c \)
We typically omit the base, and just write \( \Theta(\log x) \) for this reason.

**BASE OF LOGARITHM DOES NOT MATTER!**

- Big-O notation does not distinguish between log bases.
- Proof:
  - Fix two constant logarithm bases \( b \) and \( c \).
  - From log rules, we can change from \( \log_b x \) to \( \log_c x \) by using formula: \( \log_b x = \frac{\log_c x}{\log_c b} \).
  - But \( \log_b \) is a constant!
  - So \( \log_b x \in \Theta(\log_2 x) \).

But how do we know how much time \( M \) will take on input \( I \)?

We don’t know how much time an individual step in the program takes.

**MODEL OF COMPUTATION**

- Before we can analyze the running time of code, we need a precise model of computation.
- We use the Word-RAM model.
  - Each memory location is a word that can hold an integer.
  - Accessing a word of memory takes constant time.
  - Basic operations (arithmetic, shifting, logical operators) take constant time.

Is a word large enough to hold any integer?

List large enough to hold an address of an object in a data structure? Yes if the data structure fits in RAM...

**META-ALGORITHM FOR ANALYZING LOOPS**

- Identify operations that require only constant time.
- The complexity of a loop is the sum of the complexities of all iterations.
- Analyze independent loops separately and add the results.
- If loops are nested, it often helps to start at the innermost, and proceed outward... but:
  - Sometimes you must express several nested loops together in a single equation (using nested summations).
  - And actually evaluate the nested summations... (can be hard.)

**LOOP ANALYSIS**
TWO BIG-O ANALYSIS STRATEGIES

- **Strategy 1**:
  - Prove a \( O \) bound and a matching \( \Omega \) bound separately to get a \( \Theta \) bound. *Often easier (but not always)*

- **Strategy 2**:
  - Use \( \Theta \) bounds throughout the analysis and thereby obtain a \( \Theta \) bound for the complexity of the algorithm.

EXAMPLE 1
algorithm: loopAnalysis(n : integer)
(1) sum \( \leftarrow 0 \)
(2) for \( i \leftarrow 1 \) to \( n \)
  do 
    for \( j \leftarrow 1 \) to \( i \)
      do 
        sum \( \leftarrow \text{sum} + (i - j)^2 \)
    sum \( \leftarrow \lfloor \text{sum} / i \rfloor \)
(3) return (sum)

\[ \sum_{i=1}^{n} \Theta(i) = \sum_{i=1}^{n} \Theta(\log i) \]
EXAMPLE 3 (BENTLEY’S PROBLEM, SOLUTION 1)

max := 0;
for i := 1 to n do
  for j := i to n do
    sum := sum + A[k];
    if sum > max then max := sum;

Try to analyze this yourself!
One possible solution is given in these slides...

Strategy 1: big-O and big-\(\Omega\) bounds

\[
T(n) \in \Theta(1) + \sum_{k=1}^{n} \sum_{j=0}^{n} \Theta(1) + \Theta(1) \\
= \sum_{k=1}^{n} \sum_{j=0}^{n} \Theta(1) \\
= \Theta(n^2)
\]

This is the maximum number of iterations that could be performed in this loop.

Recall: Smallest possible value of \(j - i\) for these bounds on \(i, j\)

We will perform at least this much work in every iteration!

Since we have \(\Theta(n^2)\) and \(\Omega(n^2)\), we have proved \(\Theta(n^2)\).

Proving a big-\(\Omega\) bound...

Recall: \(T(n) \in \Omega(\sum_{k=1}^{n} \sum_{j=0}^{n} (j-0) )\)

Intuition: \(j - i \in \Theta(n)\) in some iterations. How many iterations? Latent!

To get a good \(\Omega\)-bound, we ask questions like: When do our loops have many iterations? When is our dominant term large?

Many iterations: when our \(j\) loop does \(\Omega(n)\) iterations! For example, when \(i \leq n/2\)

Large dominant term: when \(j\) is much larger than \(i\) (i.e., by a factor of \(n\))

This term does not depend on the loop indexes, so just multiply by the total number of loop iterations...

Smallest possible value of \(j - i\) for these bounds on \(i, j\)

We will perform at least this much work in every iteration!