CS 341: ALGORITHMS
Lecture 2: background and analysis
Readings: CLRS Chapters 2.1, 2.2, 3
Trevor Brown (co-taught with Anna Lubiw)
https://www.student.cs.uwaterloo.ca/~cs341
webrown@uwaterloo.ca

BIG-O NOTATION

$O$-notation:
$f(n) \in O(g(n))$ if there exist constants $c > 0$ and $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.
Here the complexity of $f$ is not higher than the complexity of $g$.

$\Omega$-notation:
$f(n) \in \Omega(g(n))$ if there exist constants $c > 0$ and $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.
Here the complexity of $f$ is not lower than the complexity of $g$.

$\Theta$-notation:
$f(n) \in \Theta(g(n))$ if there exist constants $c_1, c_2 > 0$ and $n_0 > 0$ such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.
Here $f$ and $g$ have the same complexity.

$\omega$-notation:
$f(n) \in \omega(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$.
Here $f$ has lower complexity than $g$.

$\Omega$-notation:
$f(n) \in \Omega(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$.
Here $f$ has higher complexity than $g$.

$O + \Omega = \Theta$
EXERCISE

• Which of the following are true?
  • \( n^2 \in O(n^3) \)
  • \( n^3 \in o(n^2) \)
  • \( n^3 \in \omega(n^4) \)
  • \( \log n \in o(n) \)
  • \( n \log n \in \Omega(n) \)
  • \( n \log n^2 \in \omega(n \log n) \)
  • \( n \in \Theta(n \log n) \)

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

\[
\begin{align*}
 f(n) \in O(g(n)) & \text{ means the growth rate of } f \text{ is } \leq \text{ the growth rate of } g \\
 f(n) \in o(g(n)) & \text{ means the growth rate of } f \text{ is } < \text{ the growth rate of } g \\
 f(n) \in \Omega(g(n)) & \text{ means the growth rate of } f \text{ is } \geq \text{ the growth rate of } g \\
 f(n) \in \omega(g(n)) & \text{ means the growth rate of } f \text{ is } > \text{ the growth rate of } g
\end{align*}
\]

<table>
<thead>
<tr>
<th>( 4n \in O(n^2) )</th>
<th>( 4n \in o(n^2) )</th>
<th>( 7n^2 \in O(n^2) )</th>
<th>( 7n^2 \in \omega(n^2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 7n^2 \in \Omega(n) )</td>
<td>( 7n^2 \in \omega(n) )</td>
<td>( 4n \in \Omega(n) )</td>
<td>( 4n \in \omega(n) )</td>
</tr>
</tbody>
</table>

This is included for your notes.

EXERCISE

• Which of the following are true?
  • \( n^2 \in O(n^3) \) YES
  • \( n^3 \in o(n^2) \) NO
  • \( n^3 \in \omega(n^4) \) 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

Relationships between Order Notations

\[
\begin{align*}
 f(n) \in \Theta(g(n)) & \iff g(n) \in \Theta(f(n)) \\
 f(n) \in O(g(n)) & \Rightarrow g(n) \in \Omega(f(n)) \\
 f(n) \in o(g(n)) & \Rightarrow g(n) \in \omega(f(n)) \\
 f(n) \in \Omega(g(n)) & \Rightarrow f(n) \in O(g(n)) \\
 f(n) \in \omega(g(n)) & \Rightarrow f(n) \in \Omega(g(n))
\end{align*}
\]

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 \( f(n) \not\in O(n) \).
**EXAMPLE 1: \( f(n) = n^2 - 7n - 30 \)**

- **WTP from first principles:** \( f(n) \in O(n^2) \)
  - 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 \( c = 1 \) and \( n = 1 \).

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

Let’s prove this algebraically.

---

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

- **WTP from first principles:** \( f(n) \in O(n^2) \)
  - 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?
- Must show \( n^2 \leq f(n) \leq n^2 - 7n - 30 \)
- Impossible! \( c = 1 \) is too large.
- Let’s try \( c = \frac{1}{2} \).

---

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

- **WTP from first principles:** \( f(n) \in O(n) \), Formally:
  - \((f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)
  - \((- f(n) \in O(n))\)

- Consider any arbitrary \( c > 0, n_0 > 0 \)
- We find some \( n \geq n_0 \) such that \( n^2 + n < 0 \) or \( n^2 + n > cn \)
  - \( n^2 + n > cn \) iff \( n^2 + n - cn > 0 \) iff \( n(n+c-1) > 0 \)
  - For \( n \geq n_0 > 0 \), this holds iff \( n + 1 > c \), equivalently \( n > c - 1 \)
  - So, \( n = \text{max}(c, n_0) \) will suffice

---

**Comparing Growth Rates**

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

<table>
<thead>
<tr>
<th>O(n^2)</th>
<th>O(n log n)</th>
</tr>
</thead>
</table>

---
Some Common Growth Rates (in increasing order)

| polynomial | \( \Theta(1) \) | \( \Theta(\log n) \) | \( \Theta(n) \) | \( \Theta(n^2) \) | \( \Theta(n^c) \) |
| exponential | \( \Theta(1.1^n) \) | \( \Theta(2^n) \) | \( \Theta(c^n) \) | \( \Theta(n!) \) | \( \Theta(n^n) \) |

**Limit of an Exponential Function**

\[
\lim_{n \to \infty} B^n = B \quad \text{if } B^c \to 1
\]

**Limit of a Logarithm of a Function**

\[
\lim_{n \to \infty} \log B = \log B \quad \text{if } B^c \to 1
\]

**Limit Technique for Comparing Growth Rates**

Suppose that \( f(n) > 0 \) and \( g(n) > 0 \) for all \( n \geq n_0 \). Suppose that

\[
L = \lim_{n \to \infty} \frac{f(n)}{g(n)}.
\]

Then

\[
f(n) \in \begin{cases} 
\Theta(g(n)) & \text{if } L = 0 \\
\omega(g(n)) & \text{if } L = \infty \\
\Theta(g(n)) & \text{if } 0 < L < \infty
\end{cases}
\]

**Limit Rules 1/3**

**Constant Function Rule**

The limit of a constant function is the constant:

\[
\lim_{x \to a} C = C.
\]

**Sum Rule**

This rule states that the limit of the sum of two functions is equal to the sum of their limits:

\[
\lim_{x \to a} [f(x) + g(x)] = \lim_{x \to a} f(x) + \lim_{x \to a} g(x).
\]

All of the identities shown hold only if the limits exist.

**Limit Rules 2/3**

**Product Rule**

This rule says that the limit of the product of two functions is the product of their limits (if they exist):

\[
\lim_{x \to a} [f(x)g(x)] = \lim_{x \to a} f(x) \cdot \lim_{x \to a} g(x).
\]

**Quotient Rule**

The limit of quotient of two functions is the quotient of their limits, provided that the limit in the denominator function is not zero:

\[
\lim_{x \to a} \frac{f(x)}{g(x)} = \frac{\lim_{x \to a} f(x)}{\lim_{x \to a} g(x)}, \quad \text{if } \lim_{x \to a} g(x) \neq 0.
\]

**Limit Rules 3/3**

**Power Rule**

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

**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**

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

(Where base \( b > 0 \))

**L'Hospital's Rule**

- Often we take the limit of \( \frac{f(n)}{g(n)} \) 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:

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{\frac{df(n)}{dn}}{\frac{dg(n)}{dn}}.
\]

- In other words, \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{\frac{df(n)}{dn}}{\frac{dg(n)}{dn}} \).
USING THE LIMIT METHOD: EXERCISE 1

- Compare growth rate of $n^4$ and $n^2 - 7n - 30$
  
  $$\lim_{n \to \infty} \frac{n^4 - 7n - 30}{n^2}$$
  
  $$= \lim_{n \to \infty} \frac{1 - \frac{7}{n} - \frac{30}{n^2}}{1}$$
  
  $$= 1$$

- So $n^2 - 7n - 30 \in o(n^2)$

USING THE LIMIT METHOD: EXERCISE 2

- Compare growth rate of $(\ln n)^2$ and $n^{1/2}$
  
  $$\lim_{n \to \infty} \frac{(\ln n)^2}{n^{1/2}} = \lim_{n \to \infty} \frac{\frac{d}{dn} (\ln n)^2}{\frac{d}{dn} n^{1/2}}$$
  
  $$= \lim_{n \to \infty} \frac{2 \ln n}{n^{1/2}}$$

- So $(\ln n)^2 \in o(n^{1/2})$

Additional Exercises

1. Compare the growth rate of the functions $(3 + (-1)^n)n$ and $n$.

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

Algebra of Order Notations

"Maximum" rules: Suppose that $f(n) > 0$ and $g(n) > 0$ for all $n \geq n_0$.

Then:

$O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$

$\Theta(f(n) + g(n)) = \Theta(\max\{f(n), g(n)\})$

$\Omega(f(n) + g(n)) = \Omega(\max\{f(n), g(n)\})$

"Summation" rules: Suppose $f$ is a finite set. Then

$$O \left( \sum_{i \in S} f(i) \right) = \sum_{i \in S} O(f(i))$$

$$\Theta \left( \sum_{i \in S} f(i) \right) = \sum_{i \in S} \Theta(f(i))$$

$$\Omega \left( \sum_{i \in S} f(i) \right) = \sum_{i \in S} \Omega(f(i))$$

This is included for your notes.
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:
\[
\sum_{i=0}^{n} (a + di) = na + \frac{dn(n-1)}{2} \in \Theta(n^2).
\]

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

**SEQUENCES CONTINUED**

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

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

**Miscellaneous Formulae**
\[
n! \in \Theta(n^{n+1/2}e^{-n})
\]
\[
\log n! \in \Theta(n \log n)
\]
Another useful formula is
\[
\sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6}.
\]
which implies that
\[
\sum_{i=1}^{\infty} \frac{1}{i^2} \in \Theta(1).
\]
A sum of powers of integers when \( c \geq 1 \):
\[
\sum_{i=1}^{n} i^c \in \Theta(n^{c+1}).
\]

**LOGARITHM RULES**

1. \( \log_b(xy) = \log_b x + \log_b y \)
2. \( \log_b \frac{x}{y} = \log_b x - \log_b y \)
3. \( \log_b 1/x = -\log_b x \)
4. \( \log_b x^y = y \log_b x \)
5. \( \log_b a = \frac{1}{\log_x b} \)
6. \( \log_b a = \frac{\log_x a}{\log_x b} \)
7. \( a^{\log_b n} = n^{\log_b a} \)
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: $\log_b x = \log_c x / \log_c b$
  - But $\log_c b$ is a constant!
  - So $\log_c x \in \Theta(\log_b x)$

**RUNNING TIME ANALYSIS**

**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? 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)
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.

**Strategy 1: big-O and big-Ω bounds**

We focus on the two nested for loops (i.e., (2)). The total number of iterations is $\sum_{i=1}^{n} i$, with $\Theta(1)$ time per iteration.

**Upper bound**:

\[
\sum_{i=1}^{n} O(i) \leq \sum_{i=1}^{n} O(n) - O(n^2).
\]

**Lower bound**:

\[
\sum_{i=1}^{n} \Omega(i) \geq \sum_{i=1}^{\log n} \Omega(n/\log n) = \Omega(n/\log n) = \Omega(n^2).
\]

Since the upper and lower bounds match, the complexity is $\Theta(n^2)$.

**Algorithm: LoopAnalysis**

1. $\sum \gets 0$
2. for $i \leftarrow 1$ to $n$
   1. for $j \leftarrow 1$ to $i$
      1. $\sum \leftarrow \sum + (i - j)^2$
      2. $\sum \leftarrow \lceil \sum / j \rceil$
   3. return $\sum$

**Strategy 2: use $\Theta$-bounds throughout the analysis**

**Algorithm: LoopAnalysis**

1. $\sum \leftarrow 0$
2. for $i \leftarrow 1$ to $n$
   1. for $j \leftarrow 1$ to $i$
      1. $\sum \leftarrow \sum + (i - j)^2$
      2. $\sum \leftarrow \lceil \sum / j \rceil$
   3. return $\sum$

**$\Theta$-bound analysis**

1. $\Theta(1)$
2. Complexity of inner for loop $\Theta(i)$
3. Complexity of outer loop $\Theta(n^2)$
4. Total $\Theta(n^2)$

**Example 2**

Consider this loop alone...

```
sum := 0;
for i := 1 to n do
  j := 1;
  while j <= i do
    sum := sum + 1/j;
    j := floor(j/2);
  print(sum);
```

- $\sum_{i=1}^{n} \log i = \Theta(n \log n)$
- $\sum_{i=1}^{n} \log \log i = \Theta(n \log n)$

... ANOTHER EXERCISE IN LOOP ANALYSIS?

Olive Garden waiter: Sir, you've already had 5 baskets of breadsticks.
Me:

We're good, thanks! I have none done.
EXAMPLE 3  (BENTLEY’S PROBLEM, SOLUTION 1)

max := 0;
for i := 1 to n do
  for j := i to n do
    sum := 0;
    for k := i to j 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-Ω bounds

\[ T(n) = \Theta(1) + \sum_{i=1}^{n} \left( \Theta(1) + \sum_{j=i}^{n} \Theta(1) \right) \]
\[ T(n) = \Omega(\sum_{i=1}^{n} \sum_{j=i}^{n} (j - i)) \]
\[ T(n) = \Omega\left( \sum_{i=1}^{n} \sum_{j=i}^{n} (j - i) \right) \]

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

Proving a big-Ω bound...

Recall:
\[ T(n) = \Omega\left( \sum_{i=1}^{n} \sum_{j=i}^{n} (j - i) \right) \]
\[ \geq \Omega\left( \sum_{i=1}^{n} \sum_{j=i}^{n} (n - j) \right) \]
\[ \geq \Omega\left( \sum_{i=1}^{n} \sum_{j=i}^{n} j \right) \]

Intuition: \( j - i = \Omega(n) \) in some iterations. How many iterations? Lots!

To get a good Ω-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 \( n \)).

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