Design and Analysis of Algorithms

Instructor: Sharma Thankachan

Lecture 2: Growth of Function
About this lecture

• Introduce Asymptotic Notation
  - $\Theta()$, $O()$, $\Omega()$, $o()$, $\omega()$
Recall that for input size $n$,

- **Insertion Sort**’s running time is:
  \[ An^2 + Bn + C, \quad (A,B,C \text{ are constants}) \]
- **Merge Sort**’s running time is:
  \[ Dn \log n + En + F, \quad (D,E,F \text{ are constants}) \]
- To compare their running times for large $n$, we can just focus on the dominating term (the term that grows fastest when $n$ increases)
  - $An^2$ vs $Dn \log n$
• If we look more closely, the leading constants in the dominating term does not affect much in this comparison
  - We may as well compare $n^2$ vs $n \log n$
    (instead of $A n^2$ vs $D n \log n$)

• As a result, we conclude that Merge Sort is better than Insertion Sort when $n$ is sufficiently large
Asymptotic Efficiency

• The previous comparison studies the asymptotic efficiency of two algorithms.

• If algorithm P is asymptotically faster than algorithm Q, P is often a better choice.

• To aid (and simplify) our study in the asymptotic efficiency, we now introduce some useful asymptotic notation.
Big-O notation

Definition: Given a function \( g(n) \), we denote \( O(g(n)) \) to be the set of functions

\[
\left\{ f(n) \mid \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq c \cdot g(n) \text{ for all } n \geq n_0 \right\}
\]

Rough Meaning: \( O(g(n)) \) includes all functions that are upper bounded by \( g(n) \)
Big-O notation (example)

- $4n \in O(5n)$ [proof: $c = 1$, $n \geq 1$]
- $4n \in O(n)$ [proof: $c = 4$, $n \geq 1$]
- $4n + 3 \in O(n)$ [proof: $c = 5$, $n \geq 3$]
- $n \in O(0.001n^2)$ [proof: $c = 1$, $n \geq 100$]
- $\log_e n \in O(\log n)$ [proof: $c = 1$, $n \geq 1$]
- $\log n \in O(\log_e n)$ [proof: $c = \log e$, $n \geq 1$]

Remark: Usually, we will slightly abuse the notation, and write $f(n) = O(g(n))$ to mean $f(n) \in O(g(n))$
Big-Omega notation

Definition: Given a function $g(n)$, we denote $\Omega(g(n))$ to be the set of functions

$$\{ f(n) \mid \text{there exists positive constants } c \text{ and } n_0 \text{ such that } 0 \leq c \cdot g(n) \leq f(n) \text{ for all } n \geq n_0 \}$$

Rough Meaning: $\Omega(g(n))$ includes all functions that are lower bounded by $g(n)$
Similar to Big-O, we will slightly abuse the notation, and write $f(n) = \Omega(g(n))$ to mean $f(n) \in \Omega(g(n))$

Relationship between Big-O and Big-Ω:

$$f(n) = \Omega(g(n)) \iff g(n) = O(f(n))$$
Big-$\Omega$ notation (example)

- \(5n = \Omega(4n)\) [proof: \(c = 1, n \geq 1\)]
- \(n = \Omega(4n)\) [proof: \(c = 1/4, n \geq 1\)]
- \(4n + 3 = \Omega(n)\) [proof: \(c = 1, n \geq 1\)]
- \(0.001n^2 = \Omega(n)\) [proof: \(c = 1, n \geq 100\)]
- \(\log n = \Omega(\log \log n)\) [proof: \(c = 1/\log e, n \geq 1\)]
- \(\log n = \Omega(\log_\log n)\) [proof: \(c = 1, n \geq 1\)]
Definition: Given a function $g(n)$, we denote $\Theta(g(n))$ to be the set of functions

\[
\{ f(n) \mid \text{there exists positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \}
\]

Meaning: Those functions which can be both upper bounded and lower bounded by $g(n)$.
**Big-O, Big-Ω, and Θ**

- Similarly, we write \( f(n) = \Theta(g(n)) \) to mean
  \[ f(n) \in \Theta(g(n)) \]

**Relationship between Big-O, Big-Ω, and Θ:**

\[ f(n) = \Theta(g(n)) \iff f(n) = \Omega(g(n)) \text{ and } f(n) = O(g(n)) \]
O notation (example)

- $4n = \Theta(n)$ \[ c_1 = 1, c_2 = 4, n \geq 1 \]
- $4n + 3 = \Theta(n)$ \[ c_1 = 1, c_2 = 5, n \geq 3 \]
- $\log_e n = \Theta(\log n)$ \[ c_1 = 1/\log e, c_2 = 1, n \geq 1 \]
- Running Time of Insertion Sort = $\Theta(n^2)$
  - If not specified, running time refers to the worst-case running time
- Running Time of Merge Sort = $\Theta(n \log n)$
Definition: Given a function $g(n)$, we denote $o(g(n))$ to be the set of functions

$$\{f(n) \mid \text{for any positive } c, \text{ there exists positive constant } n_0 \text{ such that } 0 < f(n) \leq c g(n) \text{ for all } n \geq n_0\}$$

Note the similarities and differences with Big-O
Little-o (equivalent definition)

Definition: Given a function $g(n)$, $o(g(n))$ is the set of functions

$$\{ f(n) \mid \lim_{n \to \infty} \left( \frac{f(n)}{g(n)} \right) = 0 \}$$

Examples:

- $4n = o(n^2)$
- $n \log n = o(n^{1.000001})$
- $n \log n = o(n \log^2 n)$
Definition: Given a function \( g(n) \), we denote \( \omega(g(n)) \) to be the set of functions

\[
\{ f(n) \mid \text{for any positive } c, \text{ there exists positive constant } n_0 \text{ such that } \\
0 < c g(n) < f(n) \text{ for all } n \geq n_0 \}
\]

Little-omega notation

Note the similarities and differences with the Big-Omega definition.
Little-omega (equivalent definition)

Definition: Given a function \( g(n) \), \( \omega(g(n)) \) is the set of functions

\[
\left\{ f(n) \mid \lim_{n \to \infty} \left( \frac{g(n)}{f(n)} \right) = 0 \right\}
\]

Relationship between Little-o and Little-\( \omega \):

\[
f(n) = \omega(g(n)) \iff g(n) = o(f(n))
\]
To remember the notation:

\( \mathcal{O} \) is like \( \leq \) : \( f(n) = \mathcal{O}(g(n)) \) means \( f(n) \leq cg(n) \)

\( \Omega \) is like \( \geq \) : \( f(n) = \Omega(g(n)) \) means \( f(n) \geq cg(n) \)

\( \Theta \) is like \( = \) : \( f(n) = \Theta(g(n)) \) \( \iff \) \( g(n) = \Theta(f(n)) \)

\( o \) is like \( < \) : \( f(n) = o(g(n)) \) means \( f(n) < cg(n) \)

\( \omega \) is like \( > \) : \( f(n) = \omega(g(n)) \) means \( f(n) > cg(n) \)

Note: Not any two functions can be compared asymptotically (E.g., \( \sin x \) vs \( \cos x \))
Your friend, after this lecture, has tried to prove $1+2+\ldots+n = O(n)$

- His proof is by induction:
  - First, $1 = O(n)$
  - Assume $1+2+\ldots+k = O(n)$
  - Then, $1+2+\ldots+k+(k+1) = O(n) + (k+1)$
    $$ = O(n) + O(n) = O(2n) = O(n)$$

So, $1+2+\ldots+n = + O(n)$  [where is the bug??]