Asymptotic Analysis of Algorithms

In a nutshell:

- Suppresses constant factors (that are system dependent)
- Suppresses lower order terms (that are irrelevant for large inputs)

Asymptotic Order of Growth

Upper bounds (Big Oh). \( T(n) = O(f(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( T(n) \leq c \cdot f(n) \).

Lower bounds (Big Omega). \( T(n) = \Omega(f(n)) \) if there exist constants \( c > 0 \) and \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( T(n) \geq c \cdot f(n) \).

Tight bounds (Theta). \( T(n) = \Theta(f(n)) \) if \( T(n) \) is both \( O(f(n)) \) and \( \Omega(f(n)) \).

Little oh. \( T(n) = o(f(n)) \) if for all constants \( c > 0 \) there is \( n_0 \geq 0 \) such that for all \( n \geq n_0 \) we have \( T(n) \leq c \cdot f(n) \).

Ex: \( T(n) = 32n^2 + 17n + 32 \).
- \( T(n) \) is \( O(n^2) \), \( O(n^3) \), \( o(n^3) \), \( \Omega(n^2) \), \( \Omega(n) \), and \( \Theta(n^2) \).
- \( T(n) \) is not \( O(n) \), \( o(n^2) \), \( \Omega(n^3) \), \( \theta(n) \), or \( \Theta(n^3) \).