Basics of Algorithm Analysis

- **We measure running time as a function of n, the size of the input (in bytes assuming a reasonable encoding).**
- **We work in the RAM model of computation. All "reasonable" operations take "1" unit of time. (e.g. +, *, -, /, array access, pointer following, writing a value, one byte of I/O...)**

What is the running time of an algorithm

- **Best case (seldom used)**
- **Average case (used if we understand the average)**
- **Worst case (used most often)**

We measure as a function of n, and ignore low order terms.

- $5n^3 + n - 6$ **becomes** n^3
- $8n \log n - 60n$ **becomes** $n \log n$
- $2^n + 3n^4$ **becomes** 2^n

Asymptotic notation

big-O

$$O(g(n)) = \{f(n): \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}\ .$$

Alternatively, we say

$$f(n) = O(g(n)) \text{ if there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0\}$$

Informally, $f(n) = O(g(n))$ **means that** $f(n)$ **is asymptotically less than or equal to** $g(n)$**.**

big-Ω

$$\Omega(g(n)) = \{f(n): \text{ there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\}\ .$$

Alternatively, we say

$$f(n) = \Omega(g(n)) \text{ if there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0\}\ .$$

Informally, $f(n) = \Omega(g(n)$ **means that** $f(n)$ **is asymptotically greater than or equal to** $g(n)$**.**

big-Θ

$f(n) = \Theta(g(n))$ **if and only if** $f(n) = O(g(n))$ **and** $f(n) = \Omega(g(n))$**.**

Informally, $f(n) = \Theta(g(n)$ **means that** $f(n)$ **is asymptotically equal to** $g(n)$**.**

INFORMAL summary

- $f(n) = O(g(n))$ **roughly means** $f(n) \leq g(n)$
- $f(n) = \Omega(g(n))$ **roughly means** $f(n) \geq g(n)$
- $f(n) = \Theta(g(n))$ **roughly means** $f(n) = g(n)$
- $f(n) = o(g(n))$ **roughly means** $f(n) < g(n)$
- $f(n) = w(g(n))$ **roughly means** $f(n) > g(n)$

We use these to classify algorithms into classes, e.g. n**,** n^2**,** $n \log n$**,** 2^n**.**

See chart for justification