# Foundations:
# **asymptotic analysis**

CH08-320201
Algorithms and Data Structures

Michael Sedlmair / Jacobs University / Spring 2018

# Updates

- New web server: vis.jacobs-university.de/teaching/ads/18s/
- Homework #1 online: http://vis.jacobs-university.de/teaching/ads/18s/homeworks/01_homework.pdf
- Moodle: will be set up this week

# Insertion Sort

```
INSERTION-SORT(A, n)
  for j = 2 to n
      key = A[j]
      // Insert A[j] into the sorted sequence A[1 .. j - 1].
      i = j - 1
      while i > 0 and A[i] > key
          A[i + 1] = A[i]
          i = i - 1
      A[i + 1] = key
```

# Running time

- The running time depends on the input: an already sorted sequence is easier to sort.
- Parameterize running time by the size of the input: short sequences are easier to sort than long ones.
- Generally, we seek upper bounds on the running time: we would like to have a guarantee.

# 1.3 Asymptotic analysis

# Types of analyses

- **Worst case**: (usually)
  T(n) = maximum time of algorithm on any input of size n.
- **Average case**: (sometimes)
  T(n) = expected time of algorithm over all inputs of size n. (Need assumption of statistical distribution of inputs.)
- **Best case**: (almost never)
  Does not make much sense, e.g., we can start with the solution.

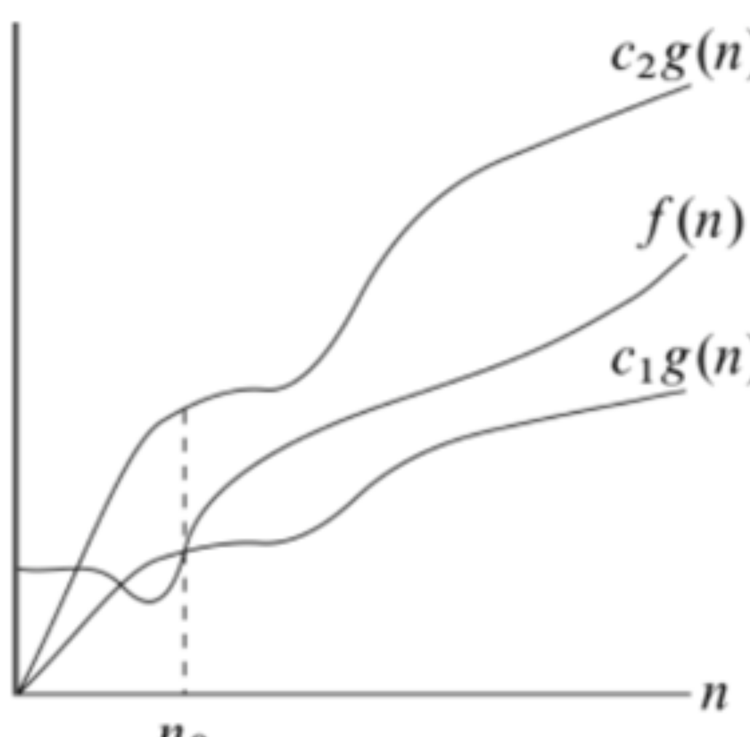
# Asymptotic analysis

- *What is INSERTION SORT's worst-case time?*
  - It depends on the speed of our computer: relative speed (on the same machine), absolute speed (on different machines).
- Idea:
  - Ignore machine-dependent constants. Look at growth of $T(n)$ as $n \to \infty$.

# Asymptotically tight bound: Θ-notation

For a given asymptotically non-negative function *g(n)*, we define

$$\Theta(g(n)) = \{f(n) \mid \exists \text{ positive constants } c_1, c_2, \text{ and } n_0, \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}.$$



$f(n) = \Theta(g(n))$

We write $f(n) = \Theta(g(n))$ instead of $f(n) \in \Theta(g(n))$.

Example:

$f(n) = 3n^3 + 90n^2 - 5n + 6046$

$3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$

# Example

$$c_1 n^3 \quad \leq \quad 3n^3 + 90n^2 - 5n + 6046 \quad \leq \quad c_2 n^3$$

$$c_1 \quad \leq \quad 3 + \frac{90}{n} - \frac{5}{n^2} + \frac{6046}{n^3} \quad \leq \quad c_2$$

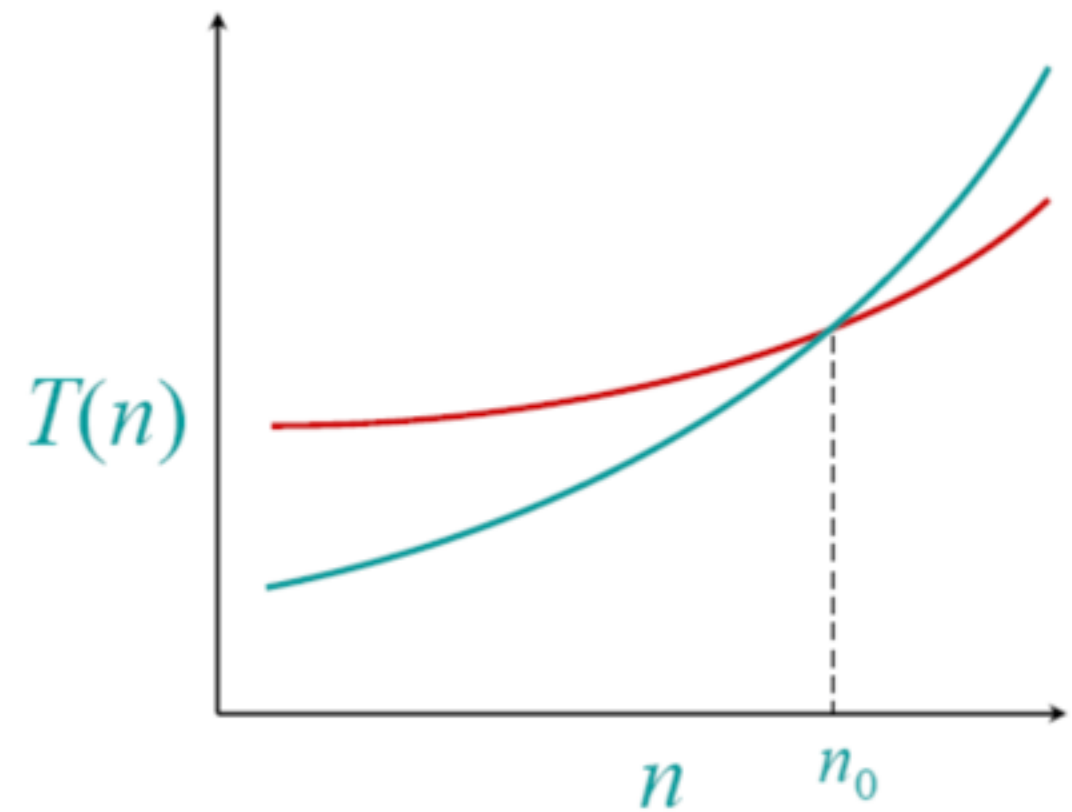
**try:** $c_1 = 2$; $c_2 = 4$; $n_0 = 100$; —> $f(n) = 3.906546$

- **Intuitively:**
  - set $c_1$ to a value smaller than the coefficient of the highest-order term
  - and $c_2$ to a value that is slight larger

# Asymptotic performance

- When *n* gets large enough, a $\Theta(n^2)$ algorithm always beats a $\Theta(n^3)$ algorithm.

- **Informal notion:**
  - throw away lower-order terms
  - ignore the leading coefficient of the highest-order term




$$\cancel{3}\mathbf{n^3} + \cancel{90n^2} - \cancel{5n} + \cancel{6046} = \boldsymbol{\Theta(n^3)}$$

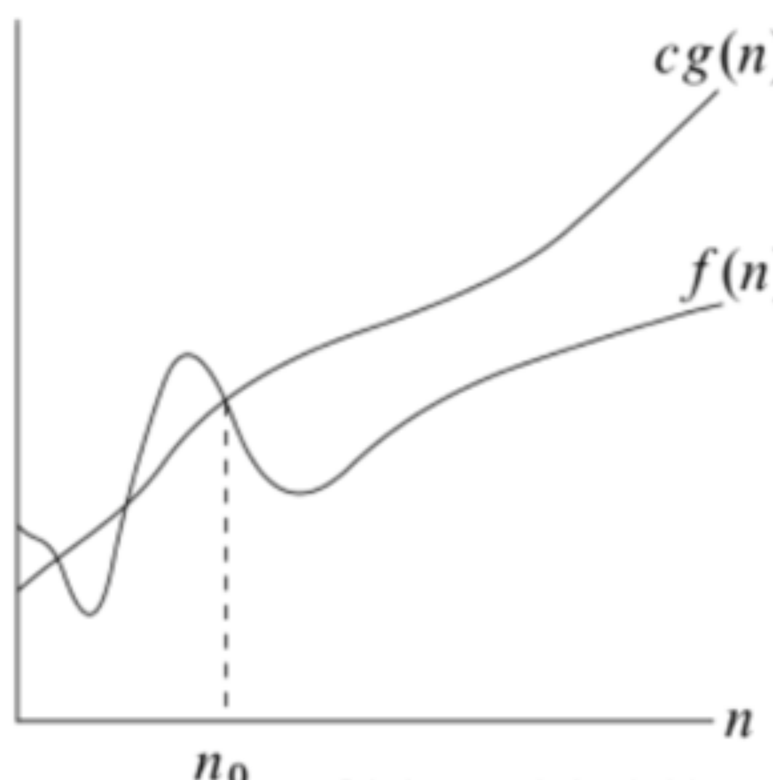
# Asymptotic analysis

- We should **not** ignore asymptotically slower algorithms, however.
- Real-world design situations often call for a careful balancing of engineering objectives.
- Asymptotic analysis is a useful tool to help to structure our thinking.

# Asymptotically upper bound: O-notation

For a given asymptotically non-negative function *g(n)*, we define

$$O(g(n)) = \{f(n) \mid \exists \text{ positive constants } c \text{ and } n_0, \text{ such that } 0 \le f(n) \le cg(n), \forall n \ge n_0\}.$$




$f(n) = O(g(n))$

Example: We say that *f(n)* is polynomially bounded if *f(n)* = $O(n^k)$ for some constant k.

# Examples

- $f(n) = 3n^3 + 90n^2 – 5n + 6046$

  —> $f(n) = O(n^3)$

- $f(n) = n$

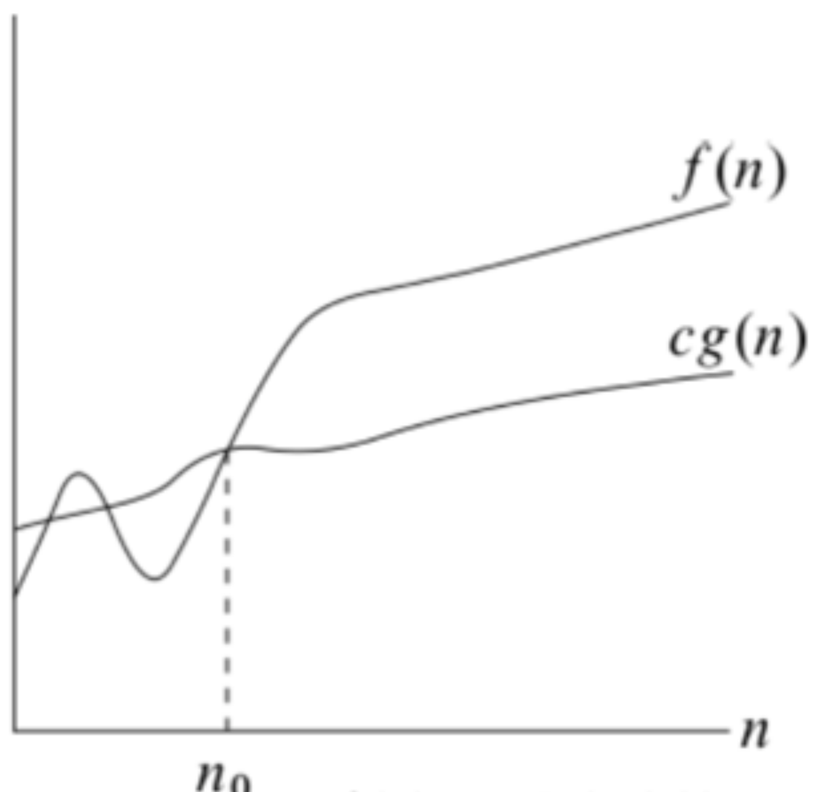
  —> $f(n) = O(n^2)$ **???**

  —> $f(n) = O(n)$ **also true**

# Asymptotically lower bound: Ω-notation

For a given asymptotically non-negative function *g(n)*, we define

$$\Omega(g(n)) = \{f(n) \mid \exists \text{ positive constants } c \text{ and } n_0, \text{ such that } 0 \leq cg(n) \leq f(n), \forall n \geq n_0\} .$$




For tight bounds, we get
*Θ(g(n)) = O(g(n)) ∩ Ω(g(n))*

# Non-tight upper bound: o-notation

For a given asymptotically non-negative function *g(n)*, we define

$$o(g(n)) = \{f(n) \mid \text{ for any constant } c > 0, \exists n_0 > 0, \text{ such that } 0 \le f(n) \le cg(n), \forall n \ge n_0\}.$$

- *f(n) = o(g(n))* implies $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

# Examples

- $2n = o(n^2)$
- $2n^2 \neq o(n^2)$ ???
- $n^b = o(a^n)$ for $a > 1$

# Non-tight lower bound: ω-notation

For a given asymptotically non-negative function *g(n)*, we define

$$f(n) \in \omega(g(n)) \text{ iff } g(n) \in o(f(n)).$$

- $f(n) = \omega(g(n))$ implies $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty.$

# Asymptotic analysis of Insertion Sort

| INSERTION-SORT$(A, n)$ | *cost* | *times* |
|---|---|---|
| **for** $j = 2$ **to** $n$ | $c_1$ | $n$ |
| $\quad key = A[j]$ | $c_2$ | $n-1$ |
| $\quad$ // Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$. | 0 | $n-1$ |
| $\quad i = j-1$ | $c_4$ | $n-1$ |
| $\quad$ **while** $i > 0$ and $A[i] > key$ | $c_5$ | $\sum_{j=2}^{n} t_j$ |
| $\quad\quad A[i+1] = A[i]$ | $c_6$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| $\quad\quad i = i-1$ | $c_7$ | $\sum_{j=2}^{n}(t_j - 1)$ |
| $\quad A[i+1] = key$ | $c_8$ | $n-1$ |

- $c_k$ is the number of steps a computer needs to perform instruction $k$ once (e.g., a Random Access Machine).
- $t_j$ is the number of times the while-loop is executed in for-iteration $j$.

# Asymptotic analysis of Insertion Sort

Best case:
Input series was ordered in reverse.
Then, $t_j = 1$.

$\Theta(n)$

# Asymptotic analysis of Insertion Sort

Worst case:
Input series was ordered in reverse.
Then, $t_j = j$.

With the arithmetic series

$$\sum_{k=1}^{n} k = \frac{1}{2} n (n+1)$$

the worst-case asymptotic complexity is

$$T(n) = \sum_{j=2}^{n} \Theta(j) = \Theta(n^2)$$

# Asymptotic analysis of Insertion Sort

Average case:
All permutations are equally likely.
Then, tj is expected to be j/2 on average.

Hence, the average-case asymptotic complexity is

$$T(n) = \sum_{j=2}^{n} \Theta(j/2) = \Theta(n^2)$$

# Asymptotic analysis of Insertion Sort

- Is Insertion Sort fast?
- For small $n$, it is still moderately fast.
- For large $n$, it is actually slow.

# Summary: Asymptotic analysis

- **O-notation** — Asymptotically upper bound
- **Ω-notation** — Asymptotically lower bound
- **Θ-notation** — Asymptotically tight bound
- **o-notation** — Non-tight upper bound
- **ω-notation** — Non-tight lower bound

$$
\begin{array}{lll}
f(n) = O(g(n)) & \text{is like} & a \le b \,, \\
f(n) = \Omega(g(n)) & \text{is like} & a \ge b \,, \\
f(n) = \Theta(g(n)) & \text{is like} & a = b \,, \\
f(n) = o(g(n)) & \text{is like} & a < b \,, \\
f(n) = \omega(g(n)) & \text{is like} & a > b \,.
\end{array}
$$