Analysis of Algorithms

- Analysis of algorithms
  - Running time
  - Space
- Asymptotic notation
  - O, Θ og Ω-notation.
- Experimental analysis of algorithms

Philip Bille

Analysis of Algorithms

- Analysis of algorithms
  - Running time
  - Space
- Asymptotic notation
  - O, Θ og Ω-notation.
- Experimental analysis of algorithms

Analysis of Algorithms

- Goal. Determine and predict computational resources and correct of algorithms.
- Ex.
  - Does my route finding algorithm work?
  - How quickly can I answer a query for a route?
  - Can it scale to 10k queries per second?
  - Will it run out of memory with large maps?
  - How many cache-misses does the algorithm generate per query? And how does this affect performance?
- Primary focus
  - Correctness, running time, space usage.
  - Theoretical and experimental analysis.

Running time

- Running time. Number of steps an algorithm performs on an input of size n.
- Steps.
  - Read/write to memory (x := y, A[i], i = i + 1, ...)
  - Arithmetic/boolean operations (+, -, *, /, %, &&, ||, &, |, ^, ~)
  - Comparisons (<, >, =<, =>, =, ≠)
  - Program flow (if-then-else, while, for, goto, function call, ..)
- Terminologi. Running time, time, time complexity.

Running time

- Worst-case running time. Maximal running time over all input of size n.
- Best-case running time. Minimal running time over all input of size n.
- Average-case running time. Average running time over all input of size n.

- Terminologi. Time = worst-case running time (unless otherwise stated).
- Other variants. Amortized, randomized, determinstic, non-deterministic, etc.

Space

- Space. Number of memory cells used by the algorithm
- Memory cells.
  - Variables and pointers/references = 1 memory cells.
  - Array of length k = k memory cells.
- Terminologi. Space, memory, storage, space complexity.

Analysis of Algorithms

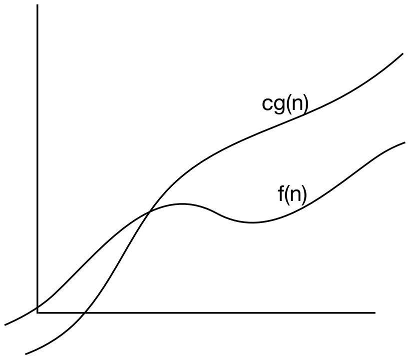
- Analysis of algorithms
  - Running time
  - Space
- Asymptotic notation
  - O, Θ og Ω-notation.
- Experimental analysis of algorithms

Asymptotic Notation

- Asymptotic notation.
  - O, Θ og Ω-notation.
  - Notation to bound the asymptotic growth of functions.
  - Fundamental tool for talking about time and space of algorithms.

O-notation

- Def. f(n) = O(g(n)) hvis f(n) ≤ cg(n) for large n.



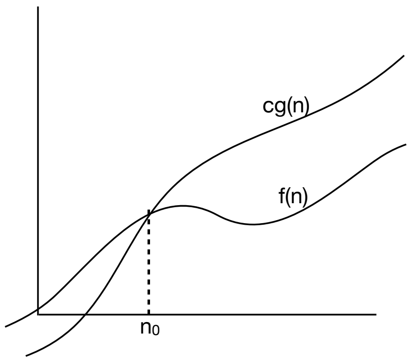
O-notation

- Ex. f(n) = $O(n^2)$ if f(n) ≤ cn^2 for large n.

- $5n^2 = O(n^2)$?
  - $5n^2 \leq 5n^2$ for large n.
- $5n^2 + 3 = O(n^2)$?
  - $5n^2 + 3 \leq 6n^2$ for large n.
- $5n^2 + 3n = O(n^2)$?
  - $5n^2 + 3n \leq 6n^2$ for large n.
- $5n^2 + 3n^2 = O(n^2)$?
  - $5n^2 + 3n^2 = 8n^2 \leq 8n^2$ for large n.
- $5n^3 = O(n^2)$?
  - $5n^3 \geq cn^2$ for all constants c for large n.

O-notation

- Def. f(n) = O(g(n)) if f(n) ≤ cg(n) for large n.
- Def. f(n) = O(g(n)) if exists constants c, $n_0 > 0$, such that for all $n \geq n_0$, f(n) ≤ cg(n).



O-notation

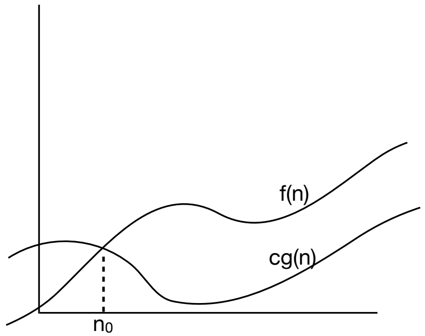
- Notation.
  - O(g(n)) is a er set of functions.
  - Think of = as ∈ or ⊆.
  - f(n) = $O(n^2)$ is ok. $O(n^2)$ = f(n) is not!

O-notation

- $f(n) = O(g(n))$ if $f(n) \leq cg(n)$ for large n.
- Exercise.
  - Let $f(n) = 3n + 2n^3 - n^2$ and $h(n) = 4n^2 + \log n$
  - Which are true?
  - $f(n) = O(n)$
  - $f(n) = O(n^3)$
  - $f(n) = O(n^4)$
  - $h(n) = O(n^2 \log n)$
  - $h(n) = O(n^2)$
  - $h(n) = O(f(n))$
  - $f(n) = O(h(n))$

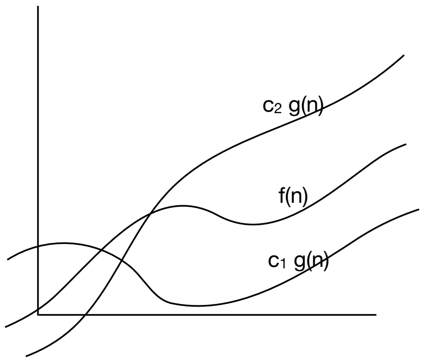
Ω-notation

- Def. $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ for large n.
- Def. $f(n) = \Omega(g(n))$ if exists constants c, $n_0 > 0$, such that for all $n \geq n_0$, $f(n) \geq cg(n)$



Θ-notation

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



Asymptotic Notation

- $f(n) = O(g(n))$ if $f(n) \leq cg(n)$ for large n.
- $f(n) = \Omega(g(n))$ if $f(n) \geq cg(n)$ for large n.
- $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ *and* $f(n) = \Omega(g(n))$.

- Exercise. Which are true? ($\log^k n$ is $(\log n)^k$)
  - $n \log^3 n = O(n^2)$
  - $2^n + 5n^7 = \Omega(n^3)$
  - $n^2(n - 5)/5 = \Theta(n^2)$
  - $4\ n^{1/100} = \Omega(n)$
  - $n^3/300 + 15 \log n = \Theta(n^3)$
  - $2^{\log n} = O(n)$
  - $\log^2 n + n + 7 = \Omega(\log n)$

Asymptotic Notation

- Basic properties.
  - Any polynomial grows proportional to it's leading term.

$$a_0 + a_1 n + a_2 n^2 + \cdots + a_d n^d = \Theta(n^d)$$

  - All logarithms are asymptotically the same.

$$\log_a(n) = \frac{\log_b n}{\log_b a} = \Theta(\log_c(n)) \qquad \text{for all constants } a, b > 0$$

  - All logarithms grows slower than all polynomials.

$$\log(n) = O(n^d) \qquad \text{for all } d > 0$$

  - All polynomials grow slower than all exponentials.

$$n^d = O(r^n) \qquad \text{for all } d > 0 \text{ and } r > 1$$

Typical Running Times

```
for i = 1 to n
  < Θ(1) time operation >
```

```
for i = 1 to n
  for j = 1 to n
    < Θ(1) time operation >
```

```
for i = 1 to n
  for j = i to n
    < Θ(1) time operation >
```

Typical Running Times

$$T(n) = \begin{cases} T(n/2) + \Theta(1) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

$$T(n) = \begin{cases} 2T(n/2) + \Theta(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

$$T(n) = \begin{cases} 2T(n/2) + \Theta(1) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

$$T(n) = \begin{cases} T(n/2) + \Theta(n) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

Analysis of Algorithms

- Analysis of algorithms
  - Running time
  - Space
- Asymptotic notation
  - O, Θ og Ω-notation.
- Experimental analysis of algorithms

Experimental Analysis

- Challenge. Can we experimentally estimate the theoretical running time?
- Doubling technique.
  - Run program and measure time for different input sizes.
  - Examine the time increase when we double the size of the input.
- Ex.
- Input size x 2 and time x 4.
- $\Longrightarrow$ Algorithm probably runs in quadratic time.
  - $T(n) = cn^2$
  - $T(2n) = c(2n)^2 = c2^2n^2 = c4n^2$
  - $T(2n)/T(n) = 4$

n	time	ratio
5000	0	-
10000	0,2	-
20000	0,6	3
40000	2,3	3,8
80000	9,4	4
160000	37,8	4

Analysis of Algorithms

- Analysis of algorithms
  - Running time
  - Space
- Asymptotic notation
  - O, Θ og Ω-notation.
- Experimental analysis of algorithms