5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of n. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

Asymptotic Notation

Formal Definition

Let f, g denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than f)
- $\Omega(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not slower than f)
- $\Theta(f) = \Omega(f) \cap \mathcal{O}(f)$
  (functions that asymptotically have the same growth as f)
- $o(f) = \{g \mid \forall c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow slower than f)
- $\omega(f) = \{g \mid \forall c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow faster than f)

Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). f and g are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \leq \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in \Omega(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} \leq \infty$
- $g \in \Theta(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in o(f)$: $\quad \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$
- $g \in \omega(f)$: $\quad \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$

- Note that for the version of the Landau notation defined here, we assume that f and g are positive functions.
- There also exist versions for arbitrary functions, and for the case that the limes is not infinity.

Asymptotic Notation

Abuse of notation

1. People write $f = \mathcal{O}(g)$, when they mean $f \in \mathcal{O}(g)$. This is **not** an equality (how could a function be equal to a set of functions).
2. People write $f(n) = \mathcal{O}(g(n))$, when they mean $f \in \mathcal{O}(g)$, with $f : \mathbb{N} \to \mathbb{R}^+, n \mapsto f(n)$, and $g : \mathbb{N} \to \mathbb{R}^+, n \mapsto g(n)$.
3. People write e.g. $h(n) = f(n) + o(g(n))$ when they mean that there exists a function $z : \mathbb{N} \to \mathbb{R}^+, n \mapsto z(n), z \in o(g)$ such that $h(n) = f(n) + z(n)$.

2. In this context $f(n)$ does **not** mean the function f evaluated at n, but instead it is a shorthand for the function itself (leaving out domain and codomain and only giving the rule of correspondence of the function).

3. This is particularly useful if you do not want to ignore constant factors. For example the median of n elements can be determined using $\frac{3}{2}n + o(n)$ comparisons.

Asymptotic Notation

Abuse of notation

4. People write $\mathcal{O}(f(n)) = \mathcal{O}(g(n))$, when they mean $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$. Again this is not an equality.

2. In this context $f(n)$ does **not** mean the function f evaluated at n, but instead it is a shorthand for the function itself (leaving out domain and codomain and only giving the rule of correspondence of the function).

3. This is particularly useful if you do not want to ignore constant factors. For example the median of n elements can be determined using $\frac{3}{2}n + o(n)$ comparisons.

Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

Here, $\Theta(n)$ stands for an anonymous function in the set $\Theta(n)$ that makes the expression true.

Note that $\Theta(n)$ is on the right hand side, otw. this interpretation is wrong.

Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + \mathcal{O}(n) = \Theta(n^2)$$

Regardless of how we choose the anonymous function $f(n) \in \mathcal{O}(n)$ there is an anonymous function $g(n) \in \Theta(n^2)$ that makes the expression true.

Asymptotic Notation in Equations

The $\Theta(i)$-symbol on the left represents one anonymous function $f : \mathbb{N} \to \mathbb{R}^+$, and then $\sum_i f(i)$ is computed.

How do we interpret an expression like:

$$\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$

Careful!

"It is understood" that every occurence of an $\mathcal{O}$-symbol (or $\Theta, \Omega, o, \omega$) on the left represents one anonymous function.

Hence, the left side is **not** equal to

$$\Theta(1) + \Theta(2) + \cdots + \Theta(n-1) + \Theta(n)$$

$\Theta(1)+\Theta(2)+\cdots+\Theta(n-1)+\Theta(n)$ does not really have a reasonable interpretation.

Asymptotic Notation in Equations

We can view an expression containing asymptotic notation as generating a set:

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n)$$

represents

$$\left\{ f : \mathbb{N} \to \mathbb{R}^+ \mid f(n) = n^2 \cdot g(n) + h(n) \right.$$
$$\left. \text{with } g(n) \in \mathcal{O}(n) \text{ and } h(n) \in \mathcal{O}(\log n) \right\}$$

Recall that according to the previous slide e.g. the expressions $\sum_{i=1}^{n} \mathcal{O}(i)$ and $\sum_{i=1}^{n/2} \mathcal{O}(i) + \sum_{i=n/2+1}^{n} \mathcal{O}(i)$ generate different sets.

Asymptotic Notation in Equations

Then an asymptotic equation can be interpreted as containement btw. two sets:

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n) = \Theta(n^2)$$

represents

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n) \subseteq \Theta(n^2)$$

Note that the equation does not hold.

Asymptotic Notation

Lemma 1
Let f, g be functions with the property $\exists n_0 > 0\, \forall n \geq n_0 : f(n) > 0$ (the same for g). Then

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant* c
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

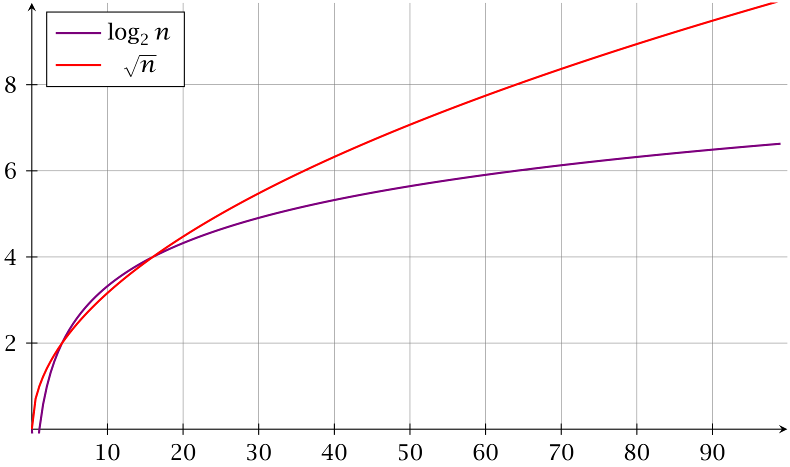
The expressions also hold for Ω. Note that this means that $f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.

Asymptotic Notation

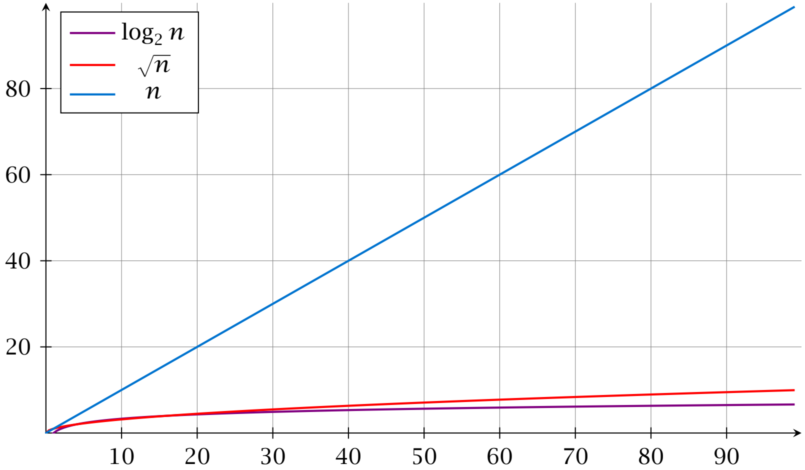
Comments

- Do not use asymptotic notation within induction proofs.
- For any constants a, b we have $\log_a n = \Theta(\log_b n)$. Therefore, we will usually ignore the base of a logarithm within asymptotic notation.
- In general $\log n = \log_2 n$, i.e., we use 2 as the default base for the logarithm.

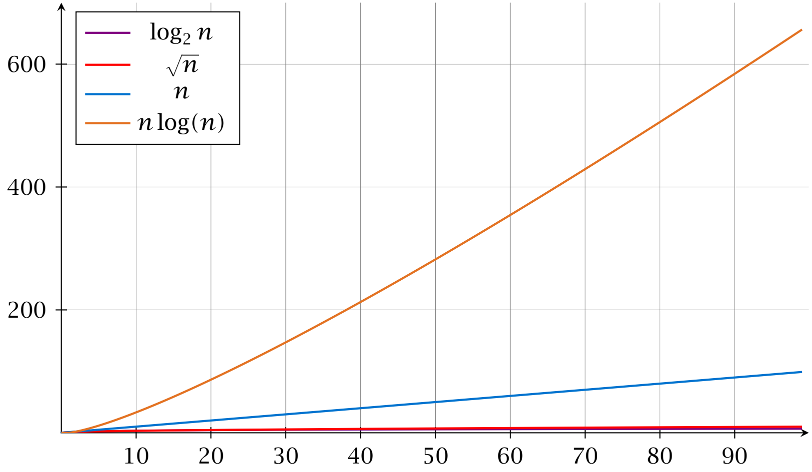
Funktionen



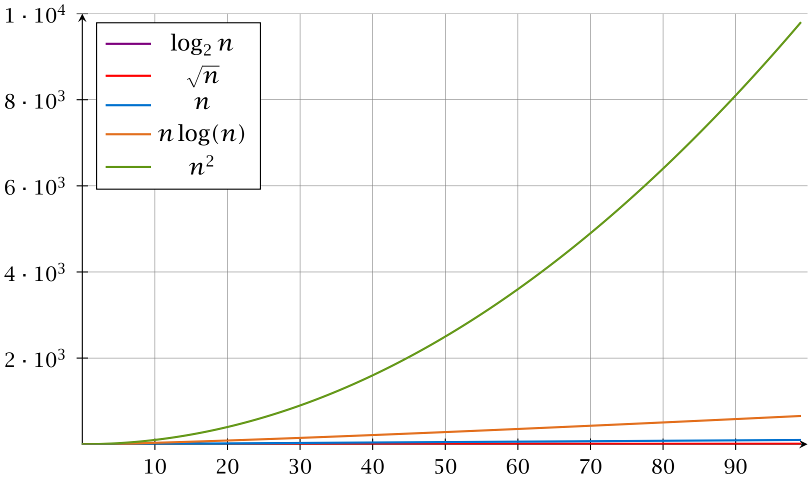
Funktionen



Funktionen

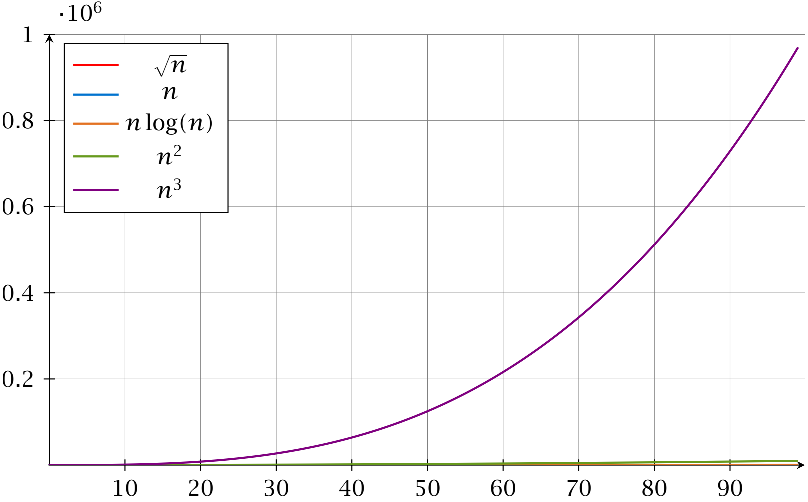


Funktionen

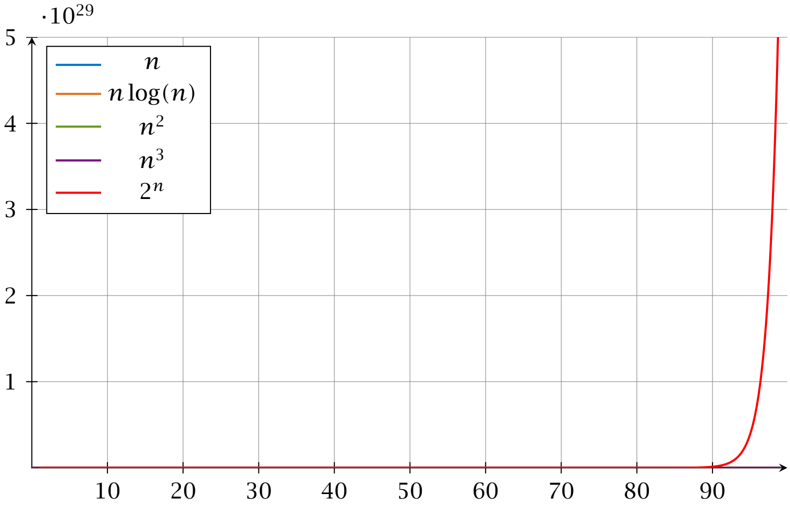


$1 \cdot 10^4$
$8 \cdot 10^3$
$6 \cdot 10^3$
$4 \cdot 10^3$
$2 \cdot 10^3$
10
20
30
40
50
60
70
80
90
$\log_2 n$
$\sqrt{n}$
n
$n \log(n)$
n^2

Funktionen



Funktionen



Laufzeiten

Funktion $f(n)$	**Eingabelänge n** 10	10^2	10^3	10^4	10^5	10^6	10^7	10^8
$\log n$	33ns	66ns	0.1µs	0.1µs	0.2µs	0.2µs	0.2µs	0.3µs
$\sqrt{n}$	32ns	0.1µs	0.3µs	1µs	3.1µs	10µs	31µs	0.1ms
n	100ns	1µs	10µs	0.1ms	1ms	10ms	0.1s	1s
$n \log n$	0.3µs	6.6µs	0.1ms	1.3ms	16ms	0.2s	2.3s	27s
$n^{3/2}$	0.3µs	10µs	0.3ms	10ms	0.3s	10s	5.2min	2.7h
n^2	1µs	0.1ms	10ms	1s	1.7min	2.8h	11d	3.2y
n^3	10µs	10ms	10s	2.8h	115d	317y	$3.2 \cdot 10^5$y	
1.1^n	26ns	0.1ms	$7.8 \cdot 10^{25}$y					
2^n	10µs	$4 \cdot 10^{14}$y						
$n!$	36ms	$3 \cdot 10^{142}$y						

1 Operation = 10ns; 100MHz

Alter des Universums: ca. $13.8 \cdot 10^9$y

Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of n.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000 \log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

  Clearly $f = o(g)$. However, as long as $\log n \leq 1000$ Algorithm B will be more efficient.

Multiple Variables in Asymptotic Notation

Sometimes the input for an algorithm consists of several parameters (e.g., nodes and edges of a graph (n and m)).

If we want to make asympotic statements for $n \to \infty$ and $m \to \infty$ we have to extend the definition to multiple variables.

Formal Definition

Let f, g denote functions from $\mathbb{N}^d$ to $\mathbb{R}_0^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists N \in \mathbb{N}_0\ \forall \vec{n}$ with $n_i \geq N$ for some i : $[g(\vec{n}) \leq c \cdot f(\vec{n})]\}$

  (set of functions that asymptotically grow not faster than f)

Multiple Variables in Asymptotic Notation

Example 2

- $f : \mathbb{N} \to \mathbb{R}_0^+$, $f(n,m) = 1$ und $g : \mathbb{N} \to \mathbb{R}_0^+$, $g(n,m) = n - 1$
  then $f = \mathcal{O}(g)$ does not hold
- $f : \mathbb{N} \to \mathbb{R}_0^+$, $f(n,m) = 1$ und $g : \mathbb{N} \to \mathbb{R}_0^+$, $g(n,m) = n$
  then: $f = \mathcal{O}(g)$
- $f : \mathbb{N}_0 \to \mathbb{R}_0^+$, $f(n,m) = 1$ und $g : \mathbb{N}_0 \to \mathbb{R}_0^+$, $g(n,m) = n$
  then $f = \mathcal{O}(g)$ does not hold

5 Asymptotic Notation

Bibliography

[MS08] Kurt Mehlhorn, Peter Sanders:
Algorithms and Data Structures — The Basic Toolbox,
Springer, 2008

[CLRS90] Thomas H. Cormen, Charles E. Leiserson, Ron L. Rivest, Clifford Stein:
Introduction to algorithms (3rd ed.),
McGraw-Hill, 2009

Mainly Chapter 3 of [CLRS90]. [MS08] covers this topic in chapter 2.1 but not very detailed.