Orders of magnitude („Big-O Notation“)

TECHNISCHE UNIVERSITÄT DARMSTADT

$$Let\ g : IN \to IR_{\geq 0}.$$

$$O(g) := \{ f : IN \to IR_{\geq 0} : \exists c > 0, n_0 \in IN, \text{ such that } \forall n \geq n_0 : f(n) \leq c \cdot g(n) \}$$

denotes the set of functions f: IN → IN, for that two positive constants $c \in IR_{\geq 0}$ and $n_0 \in IN$ exist, such that for all $n \geq n_0$ it is: $f(n) \leq c^*g(n)$

Remark: This asymptotic notation disregards constants and terms of lower
(One says: if $f \in O(g)$ then, asymptotically, f grows at most as fast as g.)

Claim: For a polynomial $f(n) = a_m n^m + ... + a_0$ of degree m with positive coefficient a_m it is valid: $f \in O(n^m)$ ← [Remark: more precisely $O(n \to n^m)$]

Proof: $f(n) \leq |a_m|\, n^m + ... + |a_1|\, n + |a_0|$
$\leq (|a_m| + |a_{m-1}| / n + ... + |a_0| / n^m) \cdot n^m$
$\leq (|a_m| + |a_{m-1}| + ... + |a_0|) \cdot n^m$

Now, $c = |a_m| + |a_{m-1}| + ... + |a_0|$ and $n_0=1$ implies the claim.

Orders of magnitude, further notations

Further definitions: Again, let $f, g: IN \rightarrow IR_{\geq 0}$

- $f \in \Omega(g) \Leftrightarrow g \in O(f)$
  („asymptotically, *f* grows at least as fast as *g*")

- $f \in \Theta(g) \Leftrightarrow f \in O(g)$ und $g \in O(f)$
  („asymptotically, *f* and *g* grow equally fast")

- $o(g) := \{ f : IN \rightarrow IN : \forall c > 0 \, \exists n_0 \in IN, so\ dass\ \forall n \geq n_0 : f(n) < c \cdot g(n) \}$
  („*f* grows less fast than *g*")

- $F \in \omega(g) \Leftrightarrow g \in o(f)$
  („*f* grows faster than *g*")

Instead of $f \in O(g)$, people sometimes write f = O(g). The same with o, ω, Ω, Θ.

Orders of magnitude, examples

- Let f(n) be the number of comparisons of a sequential search for the maximum of a number-sequence with n elements. Then $f(n) \in O(n)$, because running over the input once finfs the maximum number.

  Then again, every algorithm has at least to inspect each element of the input in order to find the maximum. Therefore, every algorithm for this problem has a running time of $f(n) \in \Omega(n)$.

- Matrix Multiplication: Let A and B be quadratic n x n matrices. The entries c_{ij} of $C = A \cdot B$ result from $c_{ij} = \sum^{n}_{k=1} a_{ik} \cdot b_{kj}$. Seemingly, n multiplications and n additions per entry. As n^2 many entires of C have to be computed , the outcome of the total effort of the „ovbious“ algorithm is: $n^2(n+n-1) = 2n^3 - n^2 \in O(n^3)$. Moreover, each algorithm for this purpose will consume $\Omega(n^2)$ operations.

  The fastest, currently known algorithm consumes $O(n^{2.376})$ operations.

Orders of magnitude, examples

- $n \in o(n^2)$, $n^2 \in O(n^2)$, $n^2 \notin o(n^2)$

- ```
  for i = 1 to n do
      for j = 1 to n do
          perform an operation
      end do
  end do
  ```

  $O(n^2)$ operations

```
for i = 1 to n do
    for j = i+1 to n do
        perform f(n) operations
    end do
end do
```

$O(n^2 \cdot f(n))$ operations
```

Orders of magnitude, remarks

TECHNISCHE UNIVERSITÄT DARMSTADT

a) The relation o(...) is transitive

$$f(n) = o(g(n) \text{ and } g(n)=o(h(n)) \quad \Rightarrow \quad f(n) = o(h(n))$$

b) The relation o(...) can be used for classifiying various functions. E.g. it is valid for $0 < \varepsilon < 1 < c$:

1	$= o(\log \log n)$	constant functions
$\log \log n$	$= o(\log n)$	double logarithmic funktions
$\log n$	$= o(n^{\varepsilon})$	logarithmic funktions
n^{ε}	$= o(n^{c})$	roots
n^{c}	$= o(n^{\log n})$	polynomials
$n^{\log n}$	$= o(c^{n})$	subexponential functions
c^{n}	$= o(n^{n})$	exponential functions
n^{n}	$= o(c^{c^{n}})$	super exponential functions

Orders of magnitude, examples

The following table shows the growth of various functions :

log n	n	n log n	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	4294967296

Orders of magnitude, computing rules

- For a constant c, it is **$c \in O(1)$**
- **$c \cdot f(n) \in O(f(n))$**, clear with definition of O-notation
- **$O(f)+O(f) \subseteq O(f)$**. Let g and h be functions from O(f). Then, there are c_g, c_h, n_g and n_h such that ... (exercise ☺)
- **$O(O(f))=O(f)$** with def.
- **$O(f) \cdot O(g) \subseteq O(f \cdot g)$** (exercise)
- **$O(f+g)=O(\max\{f(n),g(n)\})$**.

  Let $h \in O(f+g)$. Then, there are positive constants c and n_0, such that for all $n \geq n_0$ it is: $h(n) \leq c \cdot (f+g)(n) \leq c \cdot 2 \cdot \max\{f,g\}(n)$. Thus, $h(n) \in O(\max\{f,g\})$.

  The other direction, $h \in O(\max\{f,g\})$. Thus, there are positive constants c and n_0, such that for all $n \geq n_0$ it is valid: $h(n) \leq c \cdot \max\{f,g\}(n) \leq c \cdot (f+g)(n)$, and thus $h \in O(f+g)$.

Orders of magnitude, the O-notationen („Master Theorem“)

Let $a \geq 1$, $b > 1$ constants and let $T(n): IN_0 \rightarrow IR_{\geq 0}$.

Let

$$T(n) = aT(n/b) + f(n)$$

(where n/b either stands for $\lfloor n/b \rfloor$ or for $\lceil n/b \rceil$)

$\rightarrow$ if $\exists \varepsilon > 0$ with $f(n) = O(n^{\log_b a - \varepsilon})$, then

$T(n) = \Theta(n^{\log_b a})$

$\rightarrow$ if $f(n) = \Theta(n^{\log_b a})$, then

$T(n) = \Theta(n^{\log_b a} \cdot \log n)$

$\rightarrow$ if $\exists \varepsilon > 0$ with $f(n) = \Omega(n^{\log_b a + \varepsilon})$, and if there is a c with $0 < c < 1$ such that $a \cdot f(n/b) \leq c \cdot f(n)$ for sufficiently large n, then

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

Note:
There are other than these 3 cases!

Orders of magnitude, the O-notationen („Master Theorem“)

Examples:

$$T(n) = 9T(\lceil n/3 \rceil) + n$$

then is: a=9, b=3, f(n)=n, and thus $n^{\log_b a} = n^{\log_3 9} = n^2$

Therefore, $f(n) = O(n^{\log_3 9 - \varepsilon})$, and we close with case 1:

$T(n) = \Theta(n^2)$

$$T(n) = T(\lceil 2n/3 \rceil) + 1$$

then is: a=1, b=3/2, f(n)=1 and $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Case 2, because $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$

also: $T(n) = \Theta(\log n)$