Algorithms

Jiří Vyskočil, Marko Genyk-Berezovskyj

2010-2014

Introduction

- Course pages:

  https://cw.felk.cvut.cz/doku.php/courses/ae4b33alg/start

- Course goals

  The course is concerned with the ability to implement effectively solutions of various problems arising in elementary computer science. Main topics of the course include sorting and searching algorithms and related data structures. The course stresses correct algorithms choice and effective implementation as an unique tool for successful problems solving.

- Prerequisites (informal)

  The student of the course is expected to be capable of **programming** in at least one of the languages C/C++/Java. Integral part of the seminaries are practical programming homeworks on algorithm topics. The student should be familiar with basic structures like arrays, lists, files and should be able to manipulate data stored in those structures.

Problems and algorithms

- Computational problem P
  - The task is to transform input data IN to the output data OUT which satisfy some prescribed conditions.
- Algorithm A
  - Computational process reflecting the progress of solution of problem P.
  - Exact and unambiguous description of the sequence of computational steps which transforms input data IN step by step to output data OUT satisfying the conditions prescribed in P.
- Problem instance
  - Problem P taken together with one set of particular input data.
- Correctness of algorihtm A for problem P
  - Algorithm A is correct, if it can produce an **appropriate** output in **finite** time for **each** instance of problem P.

How to compare algorithms?

- For each algorithm write a program which implements it and run it on a computer with some set of data instances.
- Compare the speed and memory demands of each implementation.
- What happens when we use a different computer or OS? What happens when we choose other data instances or other programming language?
- The comparison will yield very probably a different result.
- → There is a pressing need for a comparison method which is independent on programmer, programming language, OS, computer, HW in general...

Rate of growth of functions

- Data size: n = Number of data items -- integers, vectors etc. One operation takes time 1μs (microsecond = 10^{-6} sec). $T(n)$: Number of operations required to complete the algorithm.

$n/T(n)$	20	40	60	80	100
$\log(n)$	4.3 μs	5.3 μs	5.9 μs	6.3 μs	6.6 μs
n	20 μs	40 μs	60 μs	80 μs	0.1 ms
$n\log(n)$	86 μs	0.2 ms	0.35 ms	0.5 ms	0.7 ms
n^2	0.4 ms	1.6 ms	3.6 ms	6.4 ms	10 ms
n^3	8 ms	64 ms	0.22 sec	0.5 sec	1 sec
n^4	0.16 sec	2.56 sec	13 sec	41 sec	100 sec
2^n	1 sec	12.7 days	36600 yrs	10^{11} yrs	10^{16} yrs
$n!$	77100 yrs	10^{34} yrs	10^{68} yrs	10^{105} yrs	10^{144} yrs

Asymptotic estimations

- Upper asymptotic estimate (capital omikron estimate):

$$f(n) \in O(g(n))$$

- Meaning:

f is asymptotically **bounded above** by function g
(disregarding an additive or multiplicative constant).

- Definition:

$$(\exists c > 0)(\exists n_0)(\forall n > n_0) : f(n) \leq c \cdot g(n)$$

where $c \in \mathbb{R}^{>0} \quad n_0, n \in \mathbb{N} \quad f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotic estimations

- Upper asymptotic estimate for function of more variables:

$$f(n_1, \cdots, n_k) \in O(g(n_1, \cdots, n_k))$$

- Definition:

$$(\exists c > 0)(\exists n_0)(\forall n_1 > n_0) \cdots (\forall n_k > n_0) :$$

$$f(n_1, \cdots, n_k) \leq c \cdot g(n_1, \cdots, n_k)$$

where $c \in \mathbb{R}^{>0} \quad n_0, n_1, \cdots, n_k \in \mathbb{N} \quad f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotic estimations

- Lower asymptotic estimate (capital omega estimate):

$$f(n) \in \Omega(g(n))$$

- Meaning:

  f is asymptotically **bounded below** by function g (disregarding an additive or multiplicative constant).

- Definition:

$$(\exists c > 0)(\exists n_0)(\forall n > n_0) : c \cdot g(n) \leq f(n)$$

where $c \in \mathbb{R}^{>0}$ $n_0, n \in \mathbb{N}$ $f, g \in \mathbb{N} \rightarrow \mathbb{R}^{\geq 0}$

Asymptotic estimations

- Optimal asymptotic estimation (capital theta estimate):

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

f is asymptotically **bounded above and below** by function g (disregarding an additive or multiplicative constant).

- Definition: $\Theta(g(n)) \overset{\text{def}}{=} O(g(n)) \cap \Omega(g(n))$
- Or alternatively:

$$(\exists c_1, c_2 > 0)(\exists n_0)(\forall n > n_0): c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

where $c_1, c_2 \in \mathbb{R}^{>0}$ $n_0, n \in \mathbb{N}$ $f, g \in \mathbb{N} \to \mathbb{R}^{\geq 0}$

Asymptotic estimations

- Example: Let there be a 2D array of MxN numbers. What is the asymptotic complexity of finding maximum value in this array?

- Upper bounds:
  - O((M+N)²) ✓
  - O(max(M,N)²) ✓
  - O(N²) ✗
  - O(M□N) ✓

- Lower bounds:
  - Ω(1) ✓
  - Ω(M) ✓
  - Ω(M□N) ✓

- Optimal:
  - Θ(M□N)

Asymptotic estimations

- Let $f(n)$ be the complexity of algorithm A. A is said to be

logarithmic, if	$f(n) \in \Theta(\log(n))$
linear, if	$f(n) \in \Theta(n)$
quadratic, if	$f(n) \in \Theta(n^2)$
cubic, if	$f(n) \in \Theta(n^3)$
polynomial, if	$f(n) \in \Theta(n^k)$, $k \in \mathbb{N}$
exponential, if	$f(n) \in \Theta(k^n)$, $k \in \mathbb{N}$

- Note: Logarithmic complexities do not require to include the base of the logarithm because for any $a,b > 1$ holds

$$\log_a(n) \in \Theta(\log_b(n))$$

Properties of symptotic estimations

$$\begin{aligned}
n^m &\in O\left(n^{m'}\right) \; if \; m \leq m' \\
f(n) &\in O(f(n)) \\
c \cdot O(f(n)) &= O(c \cdot f(n)) = O(f(n)) \\
O\big(O(f(n))\big) &= O\big(f(n)\big) \\
O(f(n)) + O(g(n)) &= O(\max\{f(n), g(n)\}) \\
O(f(n)) \cdot O(g(n)) &= O\big(f(n) \cdot g(n)\big) \\
O(f(n) \cdot g(n)) &= f(n) \cdot O(g(n))
\end{aligned}$$

- The class of he complexity of a polynomial is given by the term with highest exponent:

$$\sum_{i=0}^{k} a_i \cdot n^{k-i} \in \sum_{i=0}^{k} O(n^k) = k \cdot O(n^k) = O(k \cdot n^k) = O(n^k)$$

Properties of symptotic estimations

- Theorem: If functions $f(n)$, $g(n)$ are always positive then for the limit in the infinity holds:
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n) \in O(g(n))$, but **not** $f(n) \in \Theta(g(n))$
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a$, where $0 < a < \infty$, then $f(n) \in \Theta(g(n))$
  - $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $g(n) \in O(f(n))$, but **not** $g(n) \in \Theta(f(n))$
- Corollary: Let $i \in \mathbb{N}$ be a fixed integer. Then
- $$(\log(n))^i \in O(n)$$
- You may use L'Hôpital's rule to prove the corollary.