Design and Analysis of Algorithms

Schedule: Tue.+Thu. 14h30—15h45 in E11 #309

Language: English

Prerequisites: CS300 CS204+CS206

TAs: 박세원, 이원영, 임준성, 최규현

Attendance: 10 points for missing <5 lectures, 9 when missing 5, 8 when missing 6, and so on.

Grading: Homework 20%, Midterm exam 30%, Final exam 40%, Attendance 10%.

Homework: Assigned roughly every 2nd week, 7 days to solve, individual handwritten solutions.

Literature, slides, assignments etc:
`http://theoryofcomputation.asia/16CS500/`

Exams: Midterm April 21, Final exam June 16

Background Check

? CS204 *Discrete Mathematics*

? CS206 *Data Structures*

? CS300 *Introduction to Algorithms*

? CS320 *Programming Languages*

? CS322 *Formal Languages and Automata*

? MAS275 *Discrete Mathematics*

? MAS365 *Intro. to Numerical Analysis*

? MAS477 *Introduction to Graph Theory*

? MAS480 *Topics in Mathematics*

? graduate courses (at KAIST)

? non-KAIST courses

Asymptotic Efficiency

n	$\log_2 n \cdot 10$s	$n \cdot \log n$ sec	n^2 msec	n^3 μsec	2^n nsec
10	33sec	33sec	0.1sec	1msec	1msec
100	≈1min	11min	10sec	1sec	40 Mrd. Y
1000	≈1.5min	≈3h	17min	17min	
10 000	≈2min	1.5 days	≈1 day	11 days	
100 000	≈2.5min	19 days	4 months	32 years	

- Running times of some sorting algorithms
  - **BubbleSort**: $O(n^2)$ comparisons and copy instructions
  - **QuickSort**: typically $O(n \cdot \log n)$ steps
    but $O(n^2)$ in the worst-case
  - **HeapSort**: always at most $O(n \cdot \log n)$ operations
  - **BucketSort**: $O(n)$ operations
  - SORT primitive: $O(1)$
- Worst-case vs. average-case vs. best case
- w.r.t. input size =: $n \to \infty$

Example: Powering

Fix $n \in \mathbb{N}$. Given x calculate x^n with few multiplications

Applications: RSA ($x \in \mathbb{Z}_{p \cdot q}$), parallel programming, …

- Naïve algorithm: n-1 multiplications
- Inductive improvement:
  For $k := \lfloor n/2 \rfloor$ calculate x^k, then $(x^k)^2 = x^n$ or $= x^{n-1}$
- #multiplications $T(n) \leq T(n/2) + 2$, $T(n) \leq 2 \cdot \log_2(n)$
- Asympt. optimality: Each multiplication at most doubles the degree of the intermediate results;
  so computing x^n requires at least $\log_2 n$ of them.

Example: Matrix Multiplication

KAIST
CS500 M. Ziegler

- Input: entries of $n \times n$-matrices A,B — $O(n^3)$,
- Wanted: entries of $n \times n$-matrix $C := A+B$ — optimal
- High school: n^2 inner products á $O(n)$:

7 multiplications +18 additions of $(n/2)\times(n/2)$-matrizes

$T_1 := (A_{2,1}+A_{2,2})\cdot B_{1,1}$

$T_2 := (A_{1,1}+A_{1,2})\cdot B_{2,2}$

$T_3 := A_{1,1}\cdot(B_{1,2}-B_{2,2})$

$T_4 := A_{2,2}\cdot(B_{2,1}-B_{1,1})$

$$\begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix}$$

$C_{1,1} = T_5 + T_4 - T_2 + T_7$, $C_{1,2} = T_3 + T_2$

$L(n) = 7\cdot L(\lceil n/2 \rceil)$ — asymptotics dominated by #multiplications

World record: $O(n^{2.37})$ [Coppersmith&Winograd'90, François Le Gall'14]

$L(n) = O(n^{\log_2 7})$, $\log_2 7 \approx 2.81$

Algorithm

KAIST
CS500 M. Ziegler

A *finite* sequence of *primitive* instructions that, executed according to their well-specified semantics, provide a mechanical solution to the *infinitely* many instances of a possibly *complex* mathematical problem.

1. fully specified (input/output)
2. guaranteed correct (no heuristic/recipe)
3. analysis of cost (runtime, memory, …)
4. optimality proof (wrt model of computation)

"An algorithm is a finite, definite, effective procedure, with some input and some output." — Donald Knuth