

Schedule: Wed+Fri 9h00—10h15 in E11 #309

Language: English **Prerequisites:** CS300

TAs: 박세원, 조준희, Ivan Koswara CS204+CS206

Attendance: 10 points for missing <15% lectures
9 when missing <19%, 8 when missing <23% ...

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

Homework: Assigned roughly every 2nd week, 7
days to solve, handwritten solutions in groups à 3

Literature, slides, assignments etc:

<http://complexity.kaist.edu/18CS500/>

Exams: Midterm April 16 9am, final exam June 11

Asymptotic Efficiency

n	$\log_2 n \cdot 10s$	$n \cdot \log n \text{ sec}$	$n^2 \text{ msec}$	$n^3 \text{ } \mu\text{sec}$	2^n nsec
10	33sec	33sec	0.1sec	1msec	1msec
100	$\approx 1\text{min}$	11min	10sec	1sec	40 Mrd. Y
1000	$\approx 1.5\text{min}$	$\approx 3h$	17min	17min	
10 000	$\approx 2\text{min}$	1.5 days	$\approx 1 \text{ day}$	11 days	
100 000	$\approx 2.5\text{min}$	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 =: $n \rightarrow \infty$

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
a model of computation)

A program is an implementation of an algorithm.

Algorithmic Cost vs. Complexity

Algorithm, on inputs of length n :

- absolute runtime NOT
- (asymptotic) # instructions executed
- (asymptotic) amount of memory used
- (asymptotic) # parallel processors/
communication volume

→ rigorous comparison of algorithms!

Then hybridization / heuristics...

Problem Complexity

= asymptotic cost of *optimal* algorithm solving it.

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 $(x^k)^2 = x^{2k}$ 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

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

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

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

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

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

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

$$T_5 := (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \quad C_{2,1} = T_1 + T_4, \quad C_{2,2} = T_5 - T_1 + T_3 + T_6$$

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

$C_{1,1}$	$C_{1,2}$
$C_{2,1}$	$C_{2,2}$

$$=$$

$A_{1,1}$	$A_{1,2}$
$A_{2,1}$	$A_{2,2}$

$$\cdot$$

$B_{1,1}$	$B_{1,2}$
$B_{2,1}$	$B_{2,2}$

$$L(n) = 7 \cdot L(\lceil n/2 \rceil) + 18 \cdot (n/2)^2$$

World record: $O(n^{2.37})$

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

[Coppersmith & Winograd '90,
François Le Gall '14]