References

- Cormen/Leiserson/Rivest/Stein: ***Introduction to Algorithms***
- Knuth: *The Art of Computer Programming*
- Sipser: *Introduction to the Theory of Computation*

Course page **http://kaist.theoryofcomputation.asia/22cs300**

E-Learning: **http://klms.kaist.ac.kr** No email requests, please!

Homework: weekly assignments, individually, hand-written; random grading

\+ programming in *ELICE* **http://kaist.elice.io/**

Pledge for integrity and academic honesty to be signed and submitted!

Overview

- Motivation
- Searching
- Sorting
- Data
- Graphs
- Strings
- Paradigms

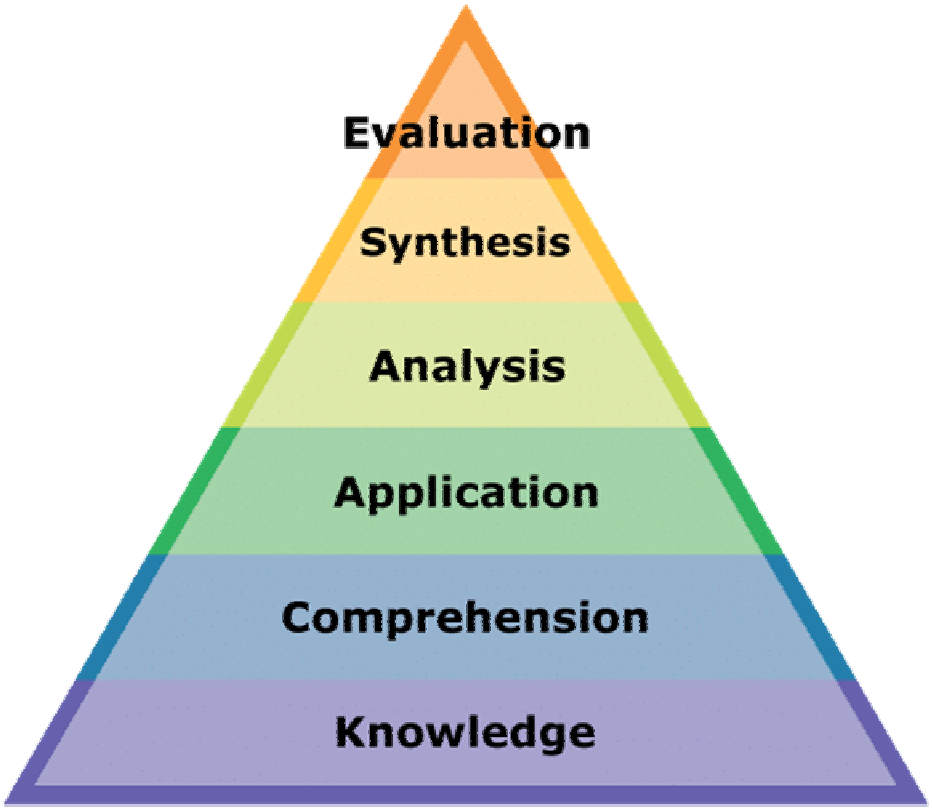
Pedagogical Concept

Education vs. Collection of Facts

This course focuses
on *Comprehension*,
Application, and *Analysis*

and takes for granted
your ability to amass
(or lookup) *Knowledge*!

Bloom's Taxonomy



Syllabus of §1 Introduction

1. Computational Problems/Algorithmic Solutions

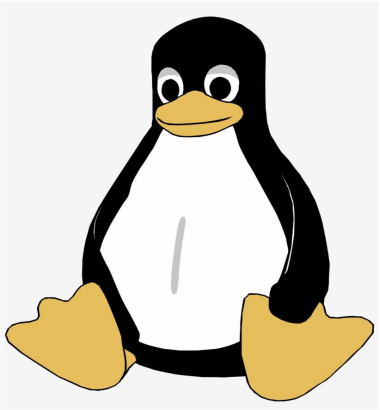
- “Virtues” of Computer Science:
  - Specification
  - Algorithm Design
  - and Analysis
  - Optimality, Example: *Square and Multiply*
- Semantics&cost for various primitive operations:
  - Four algorithms computing Fibonacci Numbers
- Mathematics: Recurrences and the *Master Theorem*
- Polynomial Multiplication: Long, Karatsuba, Toom, Cook
- Matrix Multiplication: complexity as open problem

1. Computational problems and algorithmic solutions

Power of Abstraction

high-level program.

obj.library



Computer Hardware

1. Computational problems and algorithmic solutions

"Virtues" of Computer **Science**

- problem specification
- algorithm design
  (primitives, semantics, cost)
- and analysis
  (correctness, efficiency)
- optimality proof

1. Computational problems and algorithmic solutions

specification

- Find me a good friend/spouse!
- What movie shall we watch later?
- How many angels can dance on the head of a pin?
- How many prime numbers twins are there?
- Is „*no*“ the only correct answer to this question?
- What is the smallest positive integer
  that can*not* be described in English using 100 characters?

1. Computational problems and algorithmic solutions

algorithm vs. code

```
// sort list…list_end
      mov esi, offset list
top:  mov edi, esi
inner:      mov eax, [edi]
      mov edx, [edi+4]
      cmp eax, edx
      jle no_swap
      mov [edi+4], eax
      mov [edi], edx
no_swap: add edi, 4
      cmp edi, list_end - 4
      jb inner
      add esi, 4
      cmp esi, list_end - 4
      jb top
```

- primitive operations
- their semantics
- their costs

Java:

```
Arrays.sort(numbers);
```

1. Computational problems and algorithmic solutions

pseudo-code

```
bubbleSort ( A[n] )
while n≠0 do
   newn := 0
    for i := 1 to n-1 do
            if A[i-1] > A[i]  then
                    swap(A[i-1], A[i])
                    new n := i
            endif
    endfor
    n := newn
endwhile
```

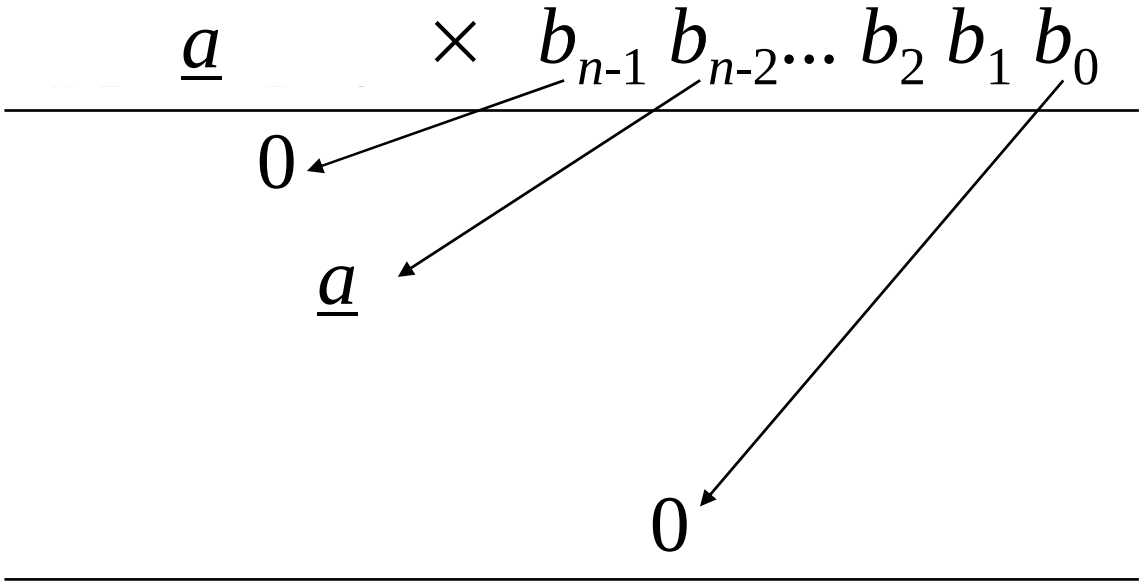
- primitive operations
- their semantics
- their costs

„Premature optimization is the root of all evil“ (D.Knuth)

1. Computational problems and algorithmic solutions

Bit-cost vs. instruct.count

Problem: Multiply two n-bit integers



- primitive operations
- their semantics
- their costs

Theorem (Harvey, v.d. Hoeven'20):

$O(n \cdot \log n)$ bit operations suffice!

$\leq n$ additions: $O(n)$

of $2n$ bits each: $O(n^2)$

1. Computational problems and algorithmic solutions

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	

Algorithm *analysis*, as opposed to empirical program evaluation

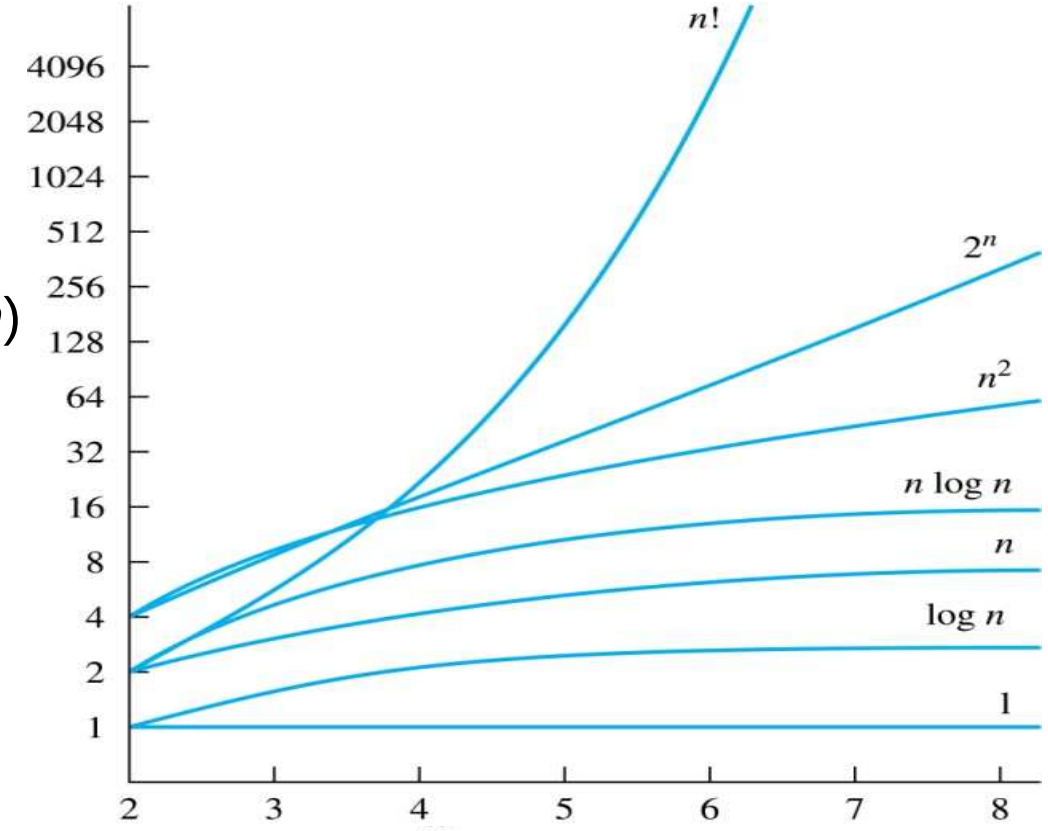
„*Premature optimization is the root of all evil*“ (D.Knuth)

1. Computational problems and algorithmic solutions

asymptotic growth

Landau Big-Oh notation $f = O(g)$

- constant 17
- logarithmic $\log(n)$
- square root $\sqrt{n}$
- linear n
- quasilinear $n \cdot \log(n)$
- quadratic n^2
- cubic n^3
- polynomial $c \cdot n^c$
- superpolynomial $n^{\log(n)}$
- exponential 2^n
- doubly exponential 2^{2^n}
- tetration $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$



1. Computational problems and algorithmic solutions

binary length vs. unary/value

Compare writing $N=2^{999}$

a) in binary:

length $1000 \approx \log_2(N)$ logarithmic in value

b) in unary: length $2^{999} \approx$ value

Deciding whether N is prime:

Sieve of Eratosthenes takes $O(\sqrt{N})$ operations.

[AKS'02]: time $O(\log^6 N)$

Factoring N:

not known feasible in time $O(\text{polylog } N)$

Basis of **RSA crypto**

1. Computational problems and algorithmic solutions

Optimality

Example *Powering Problem*: Fix some monoid $\mathcal{M}$.

Input: $X \in \mathcal{M}$ and $n \in \mathbb{N}$. Output: X^n.

Cost: #multiplications

- $X^n = X \cdot X \cdot \ldots \cdot X$: n-1 multiplications
- Recursive algorithm: compute

$X^n = (X^{n/2})^2$ if n even $T(n) \le T(\lfloor n/2 \rfloor)+1$

$X^n = (X^{(n-1)/2})^2 \cdot X$ if n odd $T(n) \le T(\lfloor n/2 \rfloor)+2$

- #multiplications $T(1)=0$, $T(n) \le T(\lfloor n/2 \rfloor)+2$,
- Proof by induction: $T(n) \le 2 \cdot \log_2(n)$

1. Computational problems and algorithmic solutions

Optimality

Example *Powering Problem*: Fix some monoid $\mathcal{M}$.

Fix $n \in \mathbb{N}$. Input: $X \in \mathcal{M}$. Output: X^n.

Cost: #multiplications

Devised & analysed algorithm using $\leq 2 \cdot \log_2 n$ multiplications

$$X^n = \mathrm{e}^{n \cdot \ln(X)}$$

3 operations!

Lemma a) Any algorithm starting with input X, and using only multiplications, produces only monomials in X as (intermediate) results.

b) After ℓ multiplications, all intermediate results (monomials) have degree $n \leq 2^\ell$.

So computing X^n requires $\ell \geq \log_2 n$ multiplications.

1. Computational problems and algorithmic solutions

Example: Fibonacci Numbers

```
FibIter(n)
if n=0 return 0;
fib := 1;  fibL := 0;
while n>1 do
    tmp:=fibL;
    fibL := fib;
    fib := fibL+tmp;
    n := n-1  ;  end
return fib;
```

$F_0=0$

$F_1=1$

$F_n = F_{n-1} + F_{n-2}$

$\geq 2^{n/2\,-1}$, $n\geq 1$

```
FibRek(n)
if n=0 return 0;  if n=1 return 1;
return FibRek(n-1)+FibRek(n-2);
```

$\geq F_n$ additions

$\leq O(n)$ additions+assignments

Observe: Computing F_n produces $\geq n/2$ bits of output.

1. Computational problems and algorithmic solutions

Example:
Fibonacci with *Multiplication*

```
FibIter(n)
if n=0 return 0;
fib := 1;  fibL := 0;
while n>1 do
    tmp:=fibL;
    fibL := fib;
    fib := fibL ⊕ tmp;
    n := n-1  ;  end
return fib;
```

$$F_0=0$$

$$F_1=1$$

$$F_n = F_{n-1} + F_{n-2}$$

```
FibRek(n)
if n=0 return 0;  if n=1 return 1;
return FibRek(n-1) ⊕ FibRek(n-2);
```

$$\begin{vmatrix} F_n \\ F_{n-1} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} \cdot \begin{vmatrix} F_{n-1} \\ F_{n-2} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}^{\oplus k} \cdot \begin{vmatrix} F_{n-k} \\ F_{n-k-1} \end{vmatrix}$$

$k := n-1$

1. Computational problems and algorithmic solutions

Example:
Fibonacci with
Exponentiation

$\varphi := (1+\sqrt{5})/2$ *Golden Ratio*

$$F_n = (\varphi^n-(-1/\varphi)^n) / \sqrt{5}$$

(*Binet*, proof by induction)

$F_0=0$

$F_1=1$

$F_n = F_{n-1} + F_{n-2}$

Four different algorithms, accelerating from exponential to constant "time"!

$$\varphi^n = \exp(n \cdot \ln \varphi)$$

Repeated squaring: $O(\log k)$ multiplications of 2×2 matrices, each expressed using 8 integer multiplications and 4 additions

$$\begin{vmatrix} F_n \\ F_{n-1} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} \cdot \begin{vmatrix} F_{n-1} \\ F_{n-2} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix}^k \cdot \begin{vmatrix} F_{n-k} \\ F_{n-k-1} \end{vmatrix} \qquad k := n-1$$

1. Computational problems and algorithmic solutions

recurrences

Master Theorem: Let $f:\mathbb{N}\to\mathbb{R}$ be increasing and satisfy the recurrence relation

$$f(n) = a \cdot f(n/b) + O(n^d)$$

whenever n is an integral power of b, where $a,b\geq 1$ are integers and $d>0$. Then

$$f(n) \text{ is } \begin{cases} O(n^d) & \text{if } a < b^d, \\ O(n^d \log n) & \text{if } a = b^d, \\ O(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

1. Computational problems and algorithmic solutions

Long Multiplication

Input: coefficients of polynomials

$A(x)=a_0+a_1x+a_2x^2+\ldots+a_{N-1}x^{N-1}$ and $B(x)$ of degree $<N$.

Output: coefficients of polynomial

$C(x) := A(x) \cdot B(x)$ of degree $\leq K:=2N-2$.

w.l.o.g. $2|N$

e.g. $\times(-5)$ or +

$T(N)$:= #arithmetic operations (multiplications, linear combinat.s)

Recursive algorithm, Distributive law: $T(N) = 4 \cdot T(N/2) + O(N)$

"Naîve" $c_k = \sum_j a_j \cdot b_{k-j}$

$$(A_0(x)+ A_1(x)\cdot x^{N/2}) \cdot (B_0(x)+ B_1(x)\cdot x^{N/2}) = C_0(x) + C_1(x) \cdot x^{N/2} + C_2(x) \cdot x^N$$

$C_0(x)= A_0(x)\cdot B_0(x)$ $C_1(x)= A_0(x)\cdot B_1(x) + A_1(x)\cdot B_0(x)$ $C_2(x)= A_1(x)\cdot B_1(x)$

$C(x)=$ $c_0, \ldots c_{N/2-1}$ | $c_{N/2}, \ldots c_{N-1}$ | $c_N, \ldots c_{3/2\cdot N-1}$ | $c_{3/2\cdot N}, \ldots c_{2N-2}$

1. Computational problems and algorithmic solutions

Karatsuba Multiplication

Input: coefficients of polynomials

$A(x)=a_0+a_1x+a_2x^2+\ldots+a_{N-1}x^{N-1}$ and $B(x)$ of degree $<N$.

Output: coefficients of polynomial

$C(x) := A(x) \cdot B(x)$ of degree $\leq K:=2N-2$.

w.l.o.g. $2|N$

e.g. $\times(-5)$ or $+$

$T(N)$:= #arithmetic operations (multiplications, linear combinat.s)

Recursive algorithm, based on:

Distributive law: $T(N) = 4\cdot T(N/2) + O(N)$

„Karatsuba law“: $T(N) = 3\cdot T(N/2) + O(N)$

$$(A_0(x)+ A_1(x)\cdot x^{N/2}) \cdot (B_0(x)+ B_1(x)\cdot x^{N/2}) = C_0(x) + C_1(x) \cdot x^{N/2} + C_2(x) \cdot x^N$$

$$C_0(x) := A_0(x) \cdot B_0(x)\,, \qquad C_2(x) := A_1(x) \cdot B_1(x)$$

$$C_1(x) := (A_0(x) + A_1(x)) \cdot (B_0(x) + B_1(x)) - C_0(x) - C_2(x)$$

1. Computational problems and algorithmic solutions

Toom Multiplication

$$T_1 := (A_0+2A_1+4A_2) \circ (B_0+2B_1+4B_2)$$
$$T_2 := (A_0+A_1+A_2) \circ (B_0+B_1+B_2)$$
$$T_3 := (4A_0+2A_1+A_2) \circ (4B_0+2B_1+B_2)$$

w.l.o.g. $3|N$

$$C_3 = -C_0 + \tfrac{1}{3}T_1 - 2T_2 + \tfrac{1}{6}T_3 - 3\tfrac{1}{2}C_4$$
$$C_2 = 3\tfrac{1}{2}C_0 - \tfrac{1}{2}T_1 + 5T_2 - \tfrac{1}{2}T_3 + 3\tfrac{1}{2}C_4$$
$$C_1 = -3\tfrac{1}{2}C_0 + \tfrac{1}{6}T_1 - 2T_2 + \tfrac{1}{3}T_3 - C_4$$
$$C_0 = A_0 \circ B_0\,,$$
$$C_4 = A_2 \circ B_2$$

Distributive law: $T(N) = 4\cdot T(N/2) + O(N)$

„*Karatsuba law*“: $T(N) = 3\cdot T(N/2) + O(N)$

„*Toom's law*“: $T(N) = \mathbf{5}\cdot T(N/\mathbf{3}) + O(N)$

$$(A_0(x)+A_1(x)\cdot x^{N/3} + A_2(x)\cdot x^{2N/3}) \times (B_0(x)+B_1(x)\cdot x^{N/3} + B_2(x)\cdot x^{2N/3})$$
$$= C_0(x) + C_1(x)\cdot x^{N/3} + C_2(x)\cdot x^{2N/3} + C_3(x)\cdot x^{3N/3} + C_4(x)\cdot x^{4N/3}$$

1. Computational problems and algorithmic solutions

Toom-Cook

Input: coeff. of $A(x)=a_0+a_1x+a_2x^2+\dots$ of $\deg<N$, $B(x)$ of $\deg<M$

Output: coeffients $c_0, \dots c_K$ of $C(x) := A(x) \cdot B(x)$, $\deg(C) \le K$

Now $K \to k := n+m-1$

Long Multiplication: $N \cdot M$ products & linear combinations

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^K \\ 1 & x_1 & x_1^2 & \dots & x_1^K \\ & & \vdots & & \\ 1 & x_K & x_K^2 & \dots & x_K^K \end{pmatrix}^{-1} \cdot \begin{pmatrix} A(x_0)\cdot B(x_0) \\ A(x_1)\cdot B(x_1) \\ \vdots \\ A(x_K)\cdot B(x_K) \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_K \end{pmatrix}$$

Vandermonde Matrix invertible for any distinct fixed $x_0,\dots x_K$

Evaluation/Interpolation: $K+1$ products, $O(NK+MK+K^2)$ lin.combs

$$T(N,M) = (n+m-1) \cdot T(N/n,M/m) + O((N+M)\cdot(n+m)^2)$$

$$\left(A_0(x) + A_1(x)\cdot x^{N/n} + A_2(x)\cdot x^{2N/n} + \dots + A_{n-1}(x)\cdot x^{(n-1)\cdot N/n}\right)$$
$$\times \left(B_0(x) + B_1(x)\cdot x^{M/m} + B_2(x)\cdot x^{2M/m} + \dots + B_{m-1}(x)\cdot x^{(m-1)\cdot M/m}\right)$$

1. Computational problems and algorithmic solutions

matrix multiplication

- Input: entries of $n \times n$-matrices A, B
- Output: entries of $C := A \times B$

$T(n)$ = #arithmetic operations

$O(n^3)$

7 multiplications +18 lin.comb.s of $(n/2)\times(n/2)$-matrizes

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

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

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

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

$T_5 := (A_{1,1}+A_{2,2}) \cdot (B_{1,1}+B_{2,2})$

$T_6 := (A_{2,1}-A_{1,1}) \cdot (B_{1,1}+B_{1,2})$

$T_7 := (A_{1,2}-A_{2,2}) \cdot (B_{2,1}+B_{2,2})$

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

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

$C_{2,1} = T_1+T_4$, $C_{2,2} = T_5-T_1+T_3+T_6$

$T(n) = 7 \cdot T(n/2) + 18 \cdot (n/2)^2 = O(n^{\log_2 7})$

World record: $O(n^{2.37})$ [F.Le Gall'14]

Recap of §1 Introduction

1. Computational Problems/Algorithmic Solutions

- “Virtues” of Computer Science:
  - Specification
  - Algorithm Design
  - and Analysis
  - Optimality, Example: *Square and Multiply*
- Semantics&cost for various primitive operations:
  - Four algorithms computing Fibonacci Numbers
- Mathematics: Recurrences and the *Master Theorem*
- Polynomial Multiplication: Long, Karatsuba, Toom, Cook
- Matrix Multiplication: complexity as open problem