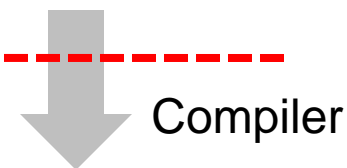
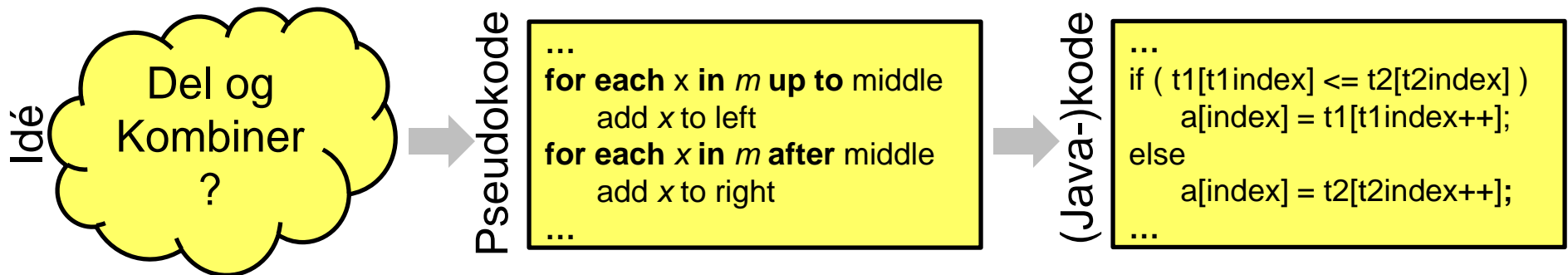


# Algoritmer og Datastrukturer

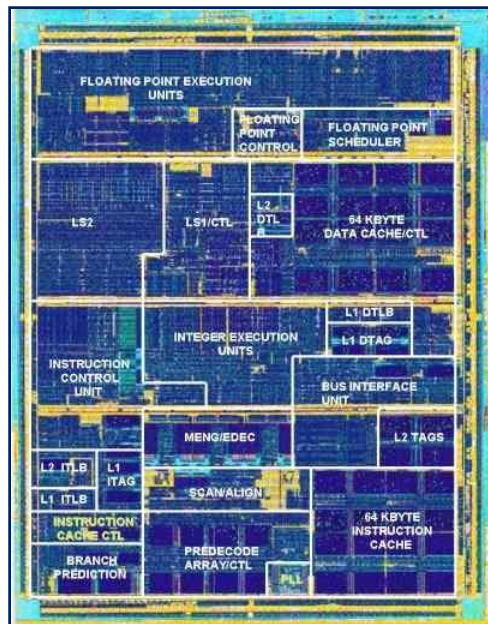
Analyseværktøjer, RAM model,  
Insertion-Sort, O-notation  
[CLRS, 1-3.1]

**Hvad er udførelstiden  
for en algoritme?**

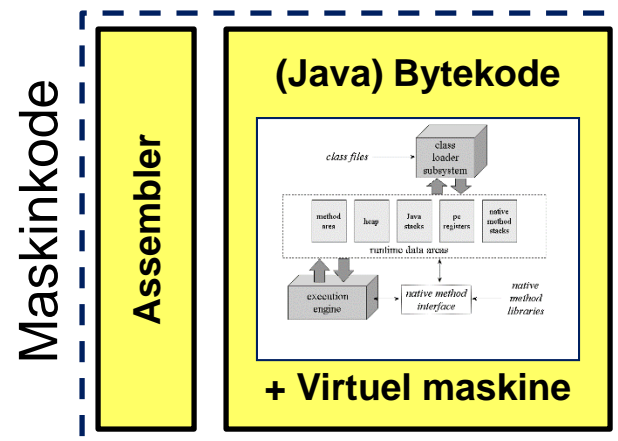
# Fra Idé til Programudførelse



- Mikrokode
- Virtuel hukkomelse/  
TLB
- L1, L2,... cache
- Branch Prediction
- Pipelining
- ...



Program-udførelse



# Hvad er udførselstiden for en algoritme?

- a) Tiden (min/sec) for at køre program
- b) # instruktioner der udføres
- c) # hukommelsesceller der læses/skrives
- d) # procedure kald
- e) Andet

# Maskiner har forskellig hastighed...

Maskine	Tid (sek)
camel19	8.9
molotov	10.2
harald	26.2
gorm	7.8

Tid for at sortere linierne i en 65 MB web log på forskellige maskiner på Institut for Datalogi

**Idé:**

**Argumenter om algoritmer uafhængig af maskine**

# Design af Algoritmer

## Korrekt algoritme

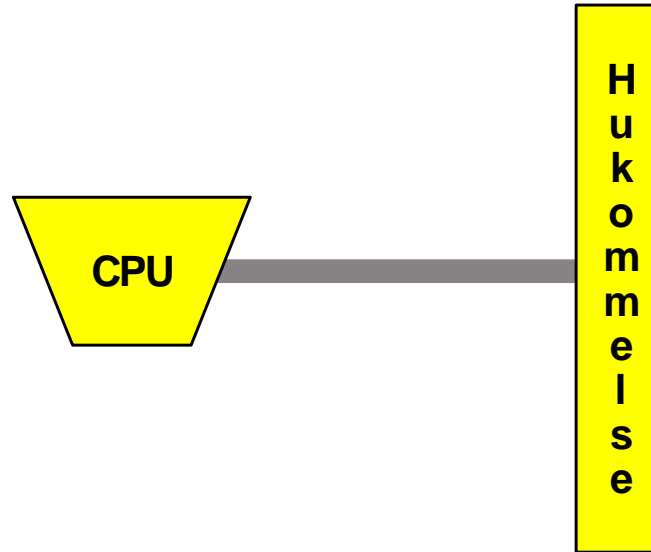
- algoritmen **standser** på alle input
- output er det **rigtige** på alle input

## Effektivitet

- Optimer algoritmerne mod at bruge **minimal** tid, plads, additioner,... eller **maximal** parallelisme...
- $\sim n^2$  er bedre end  $\sim n^3$  : **asymptotisk tid**
- Mindre vigtigt : **konstanterne**
- Resouceforbrug: **Worst-case** eller **gennemsnitlig**?

# RAM Modellen

## (Random Access Machine)



- Beregninger sker i CPU
- Data gemmes i hukommelsen
- Basale operationer tager **1 tidsenhed**:  
+, -, \*, AND, OR, XOR, **get(i)**, **set(i,v)**, ...
- Et maskinord indeholder  **$c \cdot \log n$  bits**


# Eksempel: Insertion-Sort

INSERTION-SORT( $A$ )

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert  $A[j]$  into the sorted sequence  $A[1 .. j - 1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```



# Hvor mange gange udføres linie 6 ?

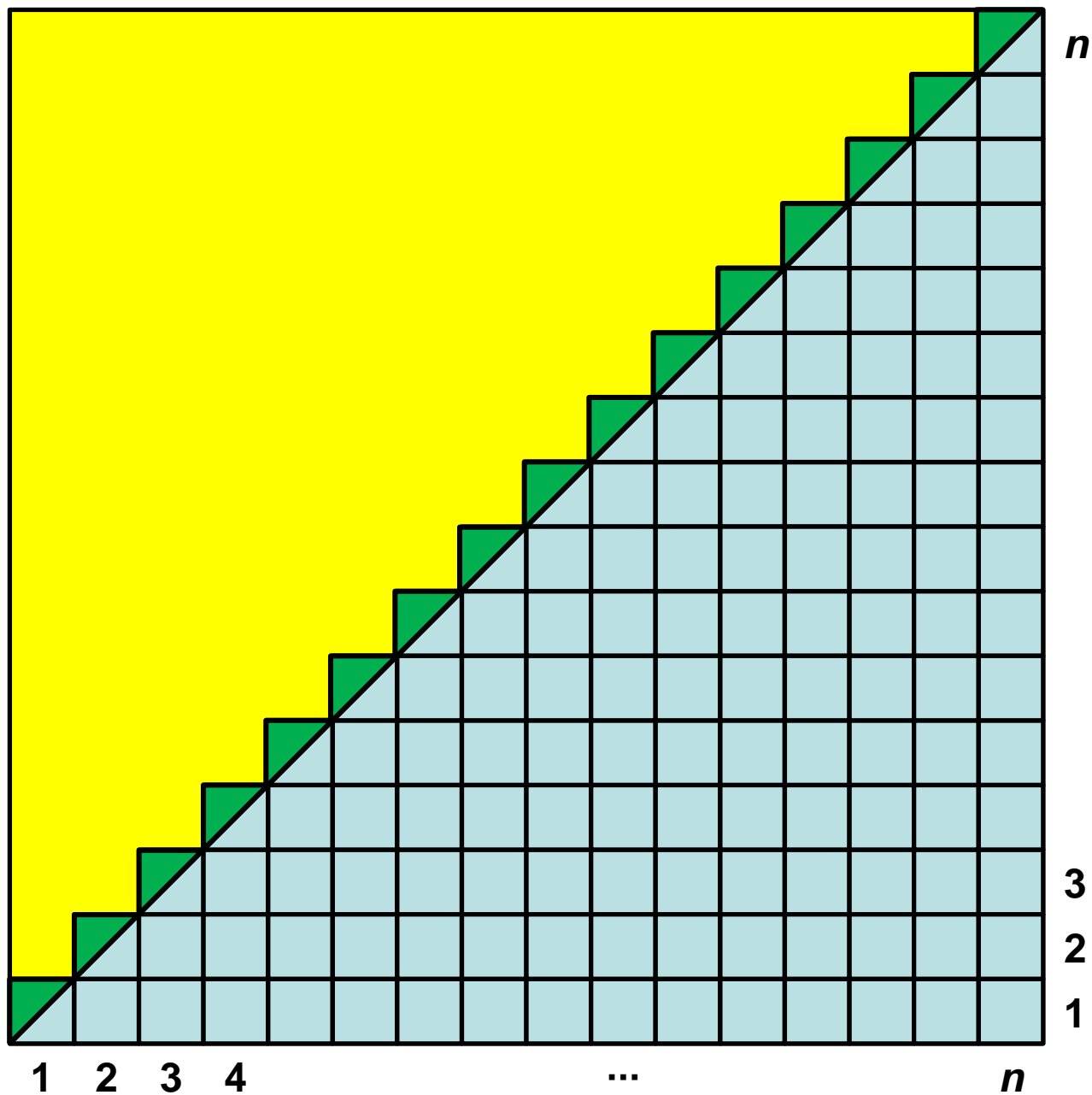
- a)  $\sim n$
- b)  $\sim n \log n$
-  c)  $\sim n^2$
- d)  $\sim n^3$
- e) Ved ikke

```
INSERTION-SORT(A)
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert  $A[j]$  into the sorted sequence  $A[1 .. j - 1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```

Input 

$n$	$n-1$	$n-2$	...	4	3	2	1
-----	-------	-------	-----	---	---	---	---

$$\begin{aligned} 1 + 2 + \dots + n \\ &= \frac{n^2}{2} + \frac{n}{2} \\ &= \frac{n(n+1)}{2} \end{aligned}$$



# Insertion-Sort (C)

```
insertion(int a[], int N)
{ int i, j, key;

  for(j=1; j < N; j++)
  { key = a[j];
    i = j-1;
    while( i>=0 && a[i] > key )
      { a[i+1] = a[i];
        i--;
      }
    a[i+1] = key;
  }
}
```

```
insertion:
pushl   %ebp
movl   %esp, %ebp
pushl   %edi
pushl   %esi
pushl   %ebx
subl   $12, %esp
cmpl   $1, 12(%ebp)
jle    .L3
movl   8(%ebp), %edx
xorl   %ebx, %ebx
movl   8(%ebp), %eax
movl   $1, -16(%ebp)
movl   4(%edx), %edx
addl   $4, %eax
movl   %eax, -20(%ebp)
movl   %edx, -24(%ebp)
.p2align 4,,7
.L6:
movl   8(%ebp), %ecx
leal   0(%ebx,4), %esi
movl   (%ecx,%ebx,4), %eax
cmpl   -24(%ebp), %eax
jle    .L8
movl   %ecx, %edi
leal   -4(%esi), %ecx
leal   (%ecx,%edi), %edx
jmp    .L9
.p2align 4,,7
.L16:
movl   (%edx), %eax
movl   %ecx, %esi
subl   $4, %edx
subl   $4, %ecx
cmpl   -24(%ebp), %eax
jle    .L8
.L9:
movl   -20(%ebp), %edi
subl   $1, %ebx
movl   %eax, (%edi,%esi)
jns    .L16
.L8:
movl   -16(%ebp), %edi
movl   8(%ebp), %edx
leal   (%edx,%edi,4), %eax
.L5:
movl   -24(%ebp), %ecx
movl   -20(%ebp), %edx
addl   $1, -16(%ebp)
movl   -16(%ebp), %edi
movl   %ecx, (%edx,%ebx,4)
cmpl   %edi, 12(%ebp)
jle    .L3
movl   4(%eax), %edx
movl   %edi, %ebx
addl   $4, %eax
subl   $1, %ebx
movl   %edx, -24(%ebp)
jns    .L6
jmp    .L5
.L3:
addl   $12, %esp
popl   %ebx
popl   %esi
popl   %edi
popl   %ebp
ret
```

# Eksempel: Insertion-Sort

- Eksempel på **pseudo-kode**
- Detaljeret analyse – stort arbejde
- Tid: **worst-case** ( $\sim n^2$ ) og **best-case** ( $\sim n$ )  
meget forskellige
- Tid: **gennemsnitlige** ( $\sim n^2$ )
- Hurtigere på  $\sim$  sorterede input: **adaptive**

# Asymptotisk notation

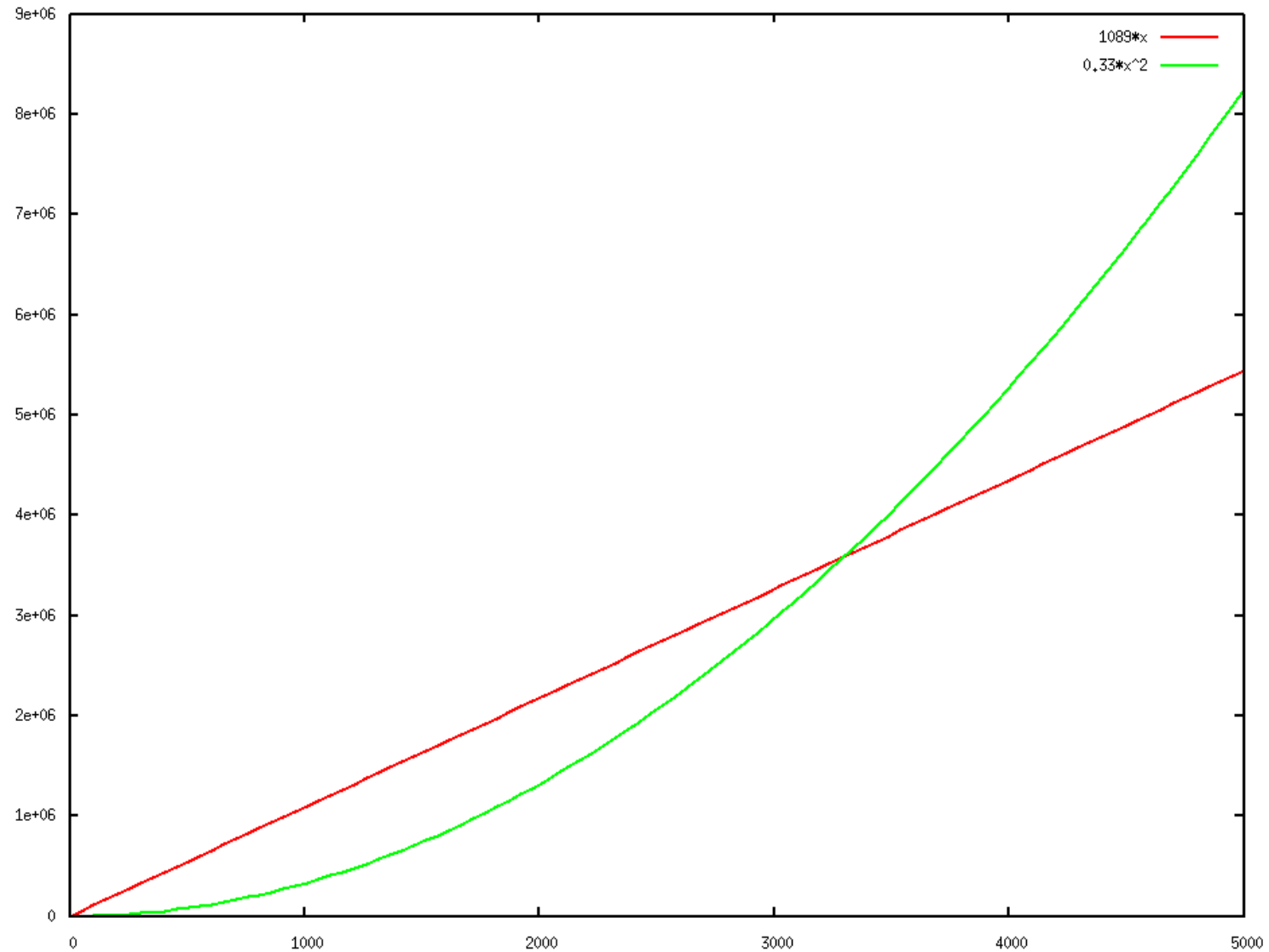
- Grundlæggende antagelse:
  - $\sim n^2$  er bedre end  $\sim n^3$
  - Konstanter ikke vigtige
- **Matematisk formel** måde at arbejde med " $\sim$ "
- Eksempler:

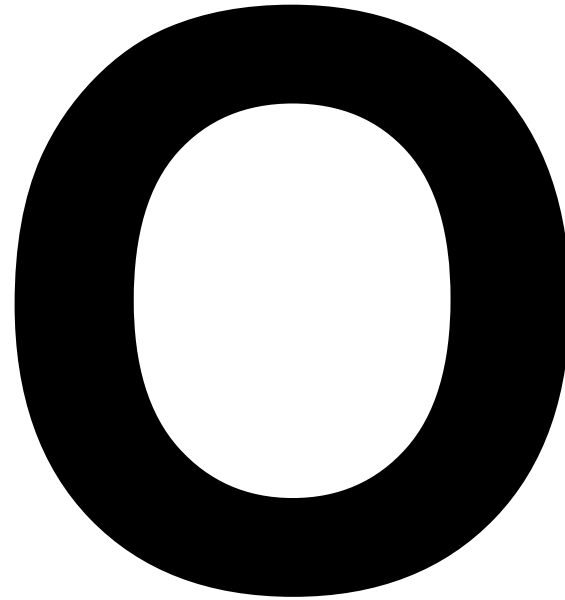
$$87 \cdot n^2 \quad " \leq " \quad 12 \cdot n^3$$

$$1089 \cdot n \quad " \leq " \quad 0.33 \cdot n^2$$

$$7 \cdot n^2 + 25 \cdot n \quad " \leq " \quad n^2$$

$1089 \cdot x$  VS  $0.33 \cdot x^2$





**- notation**

**... og vennerne**

**$\Omega$  (store omega)**

**$\Theta$  (theta)**

**$\omega$  (lille omega)**

**o (lille o)**

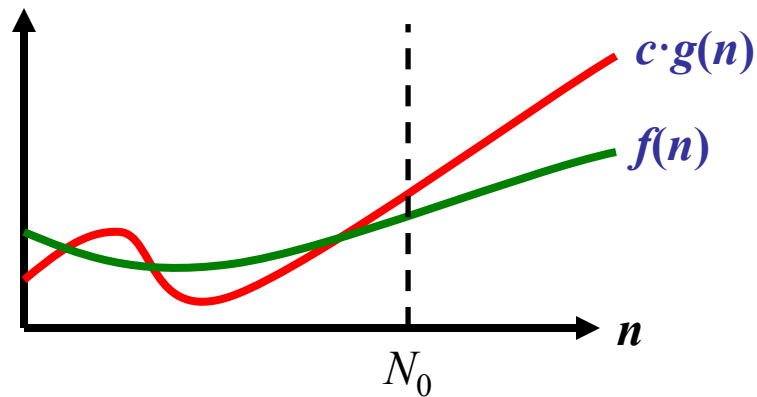
# O-notation

**Definition:**  $f(n) = O(g(n))$

hvis  $f(n)$  og  $g(n)$  er funktioner  $N \rightarrow R$  og

findes  $c > 0$  og  $N_0$  så for alle  $n \geq N_0$  :


$$f(n) \leq c \cdot g(n)$$

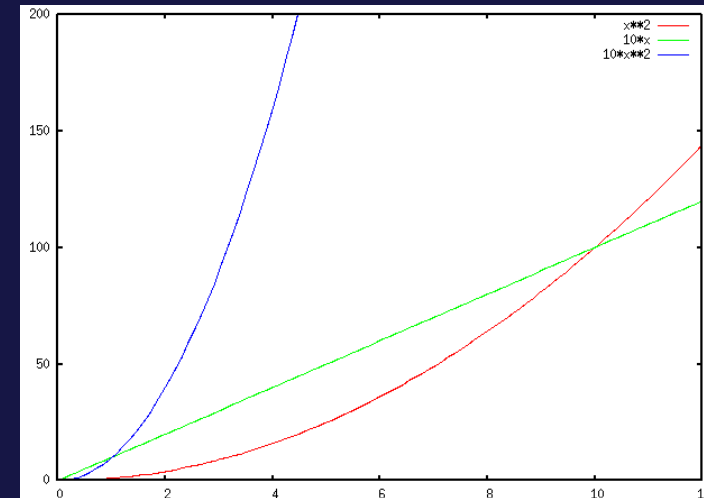


Intuitivt:  $f(n)$  er "mindre end er lig med"  $g(n)$ , eller  $g(n)$  "dominerer"  $f(n)$



$$10 \cdot n = O(n^2) ?$$

- a) Nej
- b) Ja –  $c=1$  og  $N_0=1$
- c) Ja –  $c=10$  og  $N_0=1$
- d) Ja –  $c=1$  og  $N_0=10$
-  e) Både c) og d)
- f) Ved ikke



# O-notation

$O(g(n))$  er **mængden af funktioner** der asymptotisk er begrænset af  $g(n)$

Datalogi notation	Matematik notation
$f(n) = O(g(n))$	$f(n) \in O(g(n))$
$O(f(n)) = O(g(n))$	$O(f(n)) \subseteq O(g(n))$
Eksempler: <ul style="list-style-type: none"><li>• <math>n^2 = O(n^3)</math></li><li>• <del><math>O(n^3) = n^2</math></del></li><li>• <math>O(n^2) = O(n^3)</math></li><li>• <del><math>O(n^3) = O(n^2)</math></del></li></ul>	

# Eksempel: Insertion-Sort

INSERTION-SORT( $A$ )

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert  $A[j]$  into the sorted sequence  $A[1 .. j - 1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```

**Tid**  $O(n^2)$

# Eksempler : O - regneregler

$$f(n) = O(g(n)) \rightarrow c \cdot f(n) = O(g(n))$$

$$f_1(n) = O(g_1(n)) \text{ og } f_2(n) = O(g_2(n)) \rightarrow$$

$$f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$$

$$f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$$

$$c_k \cdot n^k + c_{k-1} \cdot n^{k-1} + \dots + c_2 \cdot n^2 + c_1 \cdot n + c_0 = O(n^k)$$

$$f_1(x) = O(g_1(x)) \text{ og } f_2(x) = O(g_2(x))$$

gælder så

$$f_1(x) - f_2(x) = O(g_1(x) - g_2(x)) ?$$

a) Ja



b) Nej

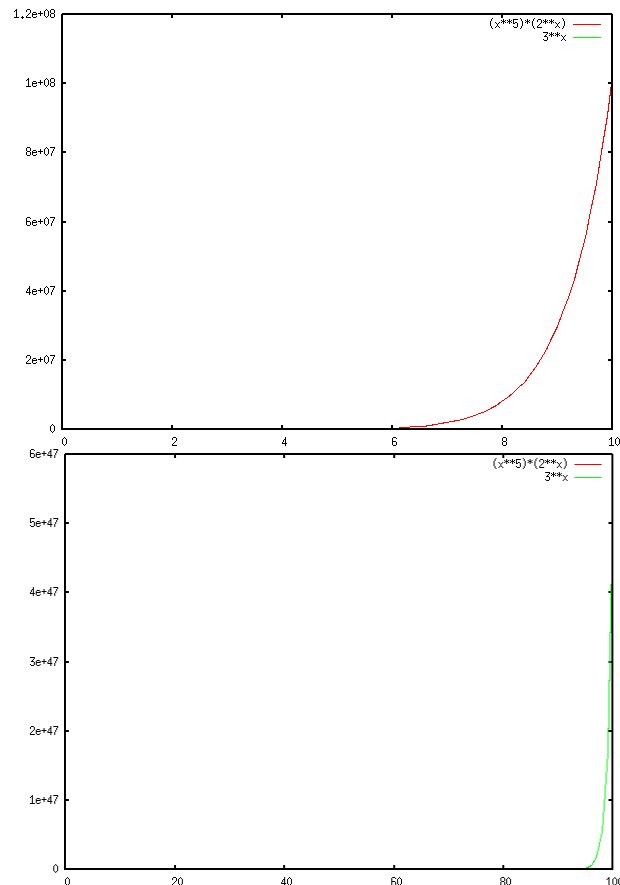
c) Ved ikke

# Eksempler : O

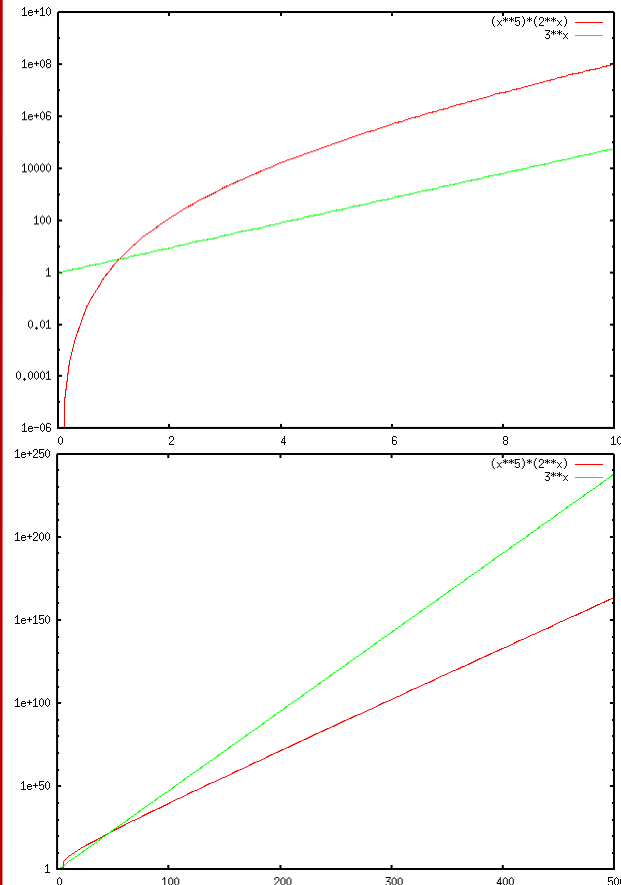
- $3 \cdot n^2 + 7 \cdot n = O(n^2)$
- $n^2 = O(n^3)$
- $\log_2 n = O(n^{0.5})$
- $(\log_2 n)^3 = O(n^{0.1})$
- $n^2 \cdot \log_2 n + 7 \cdot n^{2.5} = O(n^{2.5})$
- $2^n = O(3^n)$
- $n^5 \cdot 2^n = O(3^n)$

# Visuel test af $n^5 \cdot 2^n = O(3^n)$ ?

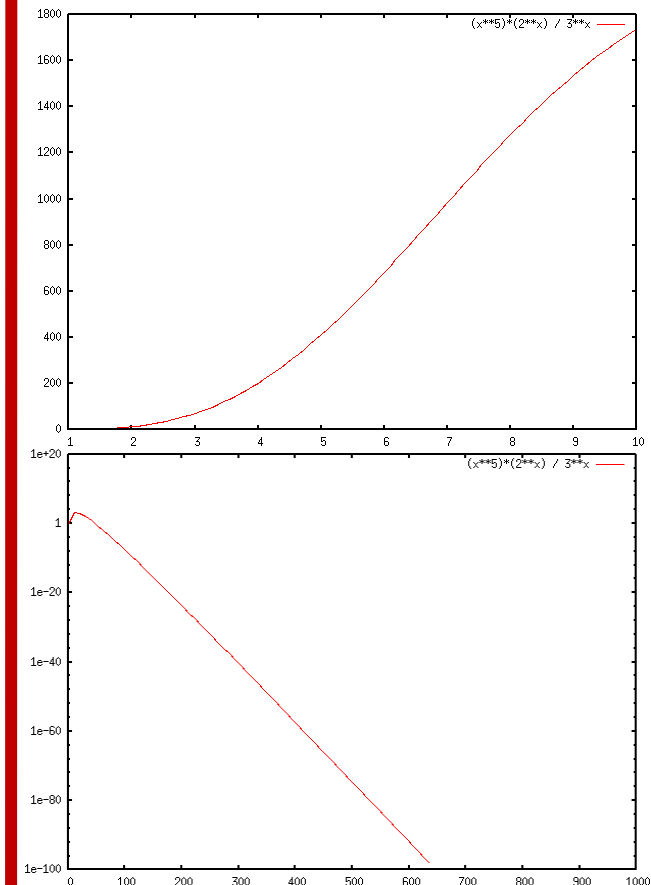
Plot af de to funktioner  
– ikke særlig informativ



Plot af de to funktioner  
med logaritmisk y-akse  
– første plot misvisende



Plot af brøken mellem  
de to funktioner  
– første plot misvisende



# Bevis for $n^5 \cdot 2^n = O(3^n)$

**Vis**  $n^5 \cdot 2^n \leq c \cdot 3^n$  for  $n \geq N_0$  for passende valg af  $c$  og  $N_0$

Bevis:

$$(5/\log_2(3/2))^2 \leq n \quad \text{for } n \geq 73$$

$$\Downarrow 5/\log_2(3/2) \leq \sqrt{n} = n/\sqrt{n} \leq n/\log_2 n \quad \text{da } \sqrt{n} \geq \log_2 n \text{ for } n \geq 17$$

$$\Downarrow 5 \cdot \log_2 n \leq n \cdot \log_2(3/2)$$

$$\Downarrow \log_2(n^5) \leq \log_2(3/2)^n$$

$$\Downarrow n^5 \leq (3/2)^n = 3^n / 2^n$$

$$\Downarrow n^5 \cdot 2^n \leq 3^n$$

Dvs. det ønskede gælder for  $c = 1$  og  $N_0 = 73$ . □

**Sætning**  $\text{poly}(n) \cdot a^n = O(b^n)$  for alle  $1 \leq a < b$

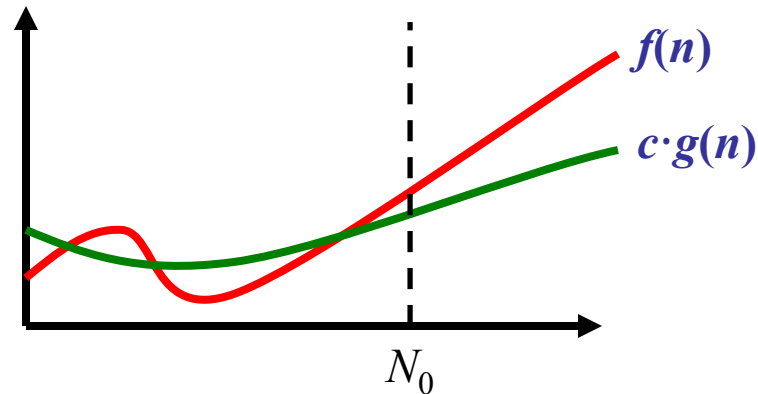


# $\Omega$ -notation

**Definition:**  $f(n) = \Omega(g(n))$

hvis  $f(n)$  og  $g(n)$  er funktioner  $N \rightarrow R$  og findes  $c > 0$  og  $N_0$  så for alle  $n \geq N_0$  :

$$f(n) \geq c \cdot g(n)$$

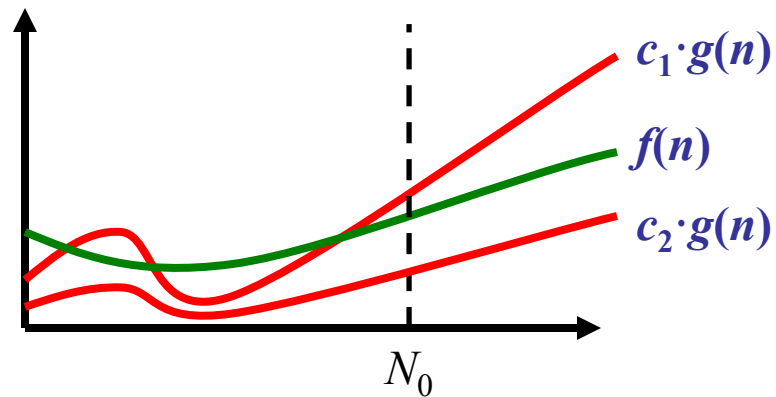


Intuitivt:  $f(n)$  er "større end er lig med"  $g(n)$ , eller  $g(n)$  er "domineret af"  $f(n)$

# $\Theta$ -notation

**Definition:**  $f(n) = \Theta(g(n))$

hvis  $f(n) = O(g(n))$  og  $f(n) = \Omega(g(n))$



Intuitivt:  $f(n)$  og  $g(n)$  er "asymptotisk ens"

# o-notation (lille o)

**Definition:**  $f(n) = o(g(n))$

hvis  $f(n)$  og  $g(n)$  er funktioner  $N \rightarrow R$  og

**for alle**  $c > 0$ , findes  $N_0$  så for alle  $n \geq N_0$  :

$$f(n) \leq c \cdot g(n)$$

Intuitivt:  $f(n)$  er "skarpt mindre end"  $g(n)$

# $\omega$ -notation

**Definition:**  $f(n) = \omega(g(n))$

hvis  $f(n)$  og  $g(n)$  er funktioner  $N \rightarrow R$  og

**for alle**  $c > 0$ , findes  $N_0$  så **for alle**  $n \geq N_0$  :

$$f(n) \geq c \cdot g(n)$$

Intuitivt:  $f(n)$  er "skarpt større end"  $g(n)$

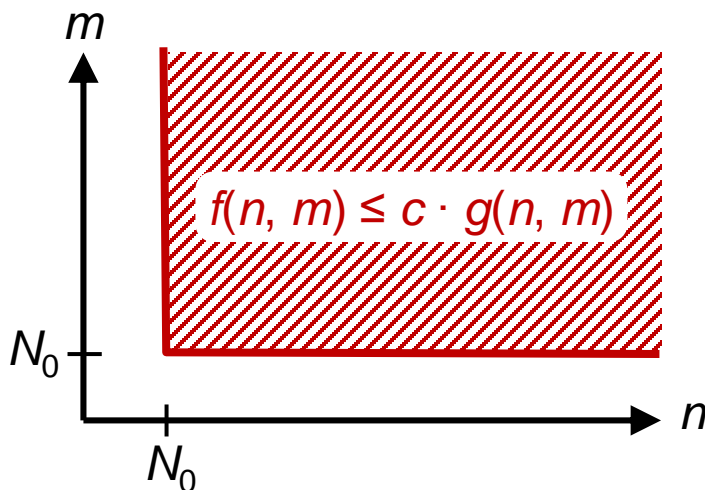
# O-notation i flere variable

Vi vil gerne kunne skrive

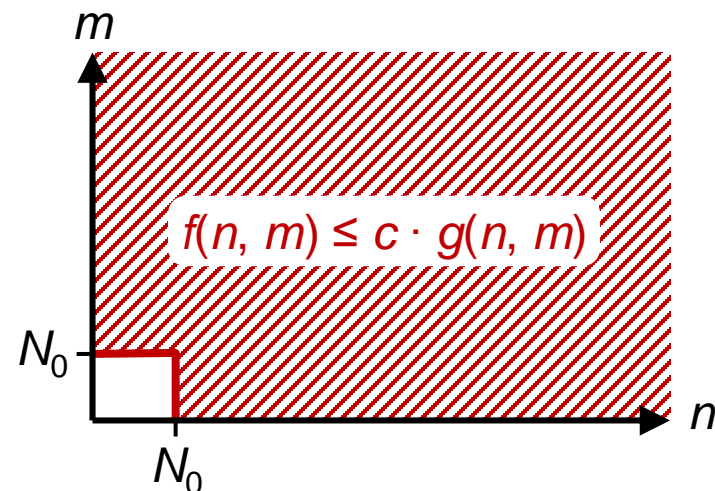
$$3n^2 \cdot m + 7n \cdot m = O(n^2 \cdot m)$$

Litteraturen er dog ikke præcis / konsistent...

$$f(n, m) = O(g(n, m)) \quad ?$$



eller  
?



# Algoritme Analyse

- RAM model
- O-notation

... behøver ikke at beskrive og analysere algoritmer i detaljer !