

Analyse af algoritmer

- Analyse af algoritmer
 - Køretid
 - Pladsforbrug
- Asymptotisk notation
 - O , Θ og Ω -notation.
- Eksperimentiel analyse af algoritmer

Philip Bille

Analyse af algoritmer

- Analyse af algoritmer
 - Køretid
 - Pladsforbrug
- Asymptotisk notation
 - O , Θ og Ω -notation.
- Eksperimentiel analyse af algoritmer

Analyse af algoritmer

- **Mål.** At **bestemme** og **forudsige** resourceforbrug og korrekthed af algoritmer
- **Eks.**
 - Virker min algoritme til at beregne korteste veje i grafer?
 - Hvor hurtigt kører min algoritme til at søge efter approximativt matchende billeder i en billeddatabase? Skaleres den til 100000+ billeder?
 - Løber min indekseringsalgoritme til DNA-sekvenser tør for hukommelse når jeg indekserer 1000 menneskegenomer?
 - Hvor mange cache-misses genererer min sorteringsalgoritme?
- **Primært fokus.**
 - Korrekthed, køretid og pladsforbrug.
 - Teoretisk og eksperimentiel analyse.

Køretid

- **Køretid.** Antallet af **skridt** som en algoritmen udfører på et input af størrelse n .
- **Skridt.**
 - Læsning/skrivning til hukommelse ($x := y$, $A[i]$, $i = i + 1$, ...)
 - Aritmetiske/boolske operationer ($+$, $-$, $*$, $/$, $\%$, $\&\&$, $\|$, $\&$, $|$, \wedge , \sim)
 - Sammenligninger ($<$, $>$, $=<$, $=>$, $=$, \neq)
 - Programflow (if-then-else, while, for, goto, funktionskald, ..)
- **Terminologi.** Køretid, tid, tidskompleksitet.

Køretid

- **Værstefalds-køretid** (*worst-case running time*). Maksimal køretid over alle input af størrelse n .
- **Bedstefalds-køretid** (*best-case running time*). Minimal køretid over alle input af størrelse n .
- **Gennemsnitlige-køretid** (*average-case running time*). Gennemsnitlig køretid over alle input af størrelse n .
- **Terminologi**. Køretid = værstefalds-køretid (medmindre vi eksplicit skriver andet).
- **Andre varianter**. Amortiseret, randomiseret, deterministisk, non-deterministisk, etc.

Pladsforbrug

- **Pladsforbrug**. Antallet af **lagerpladser** som algoritmen bruger.
- **Lager**.
 - Variable og pointere/referencer = 1 lagerplads.
 - Tabel af længde k = k lagerpladser.
- **Terminologi**. Pladsforbrug, plads, pladskompleksitet, lagerplads, lager.

Analyse af algoritmer

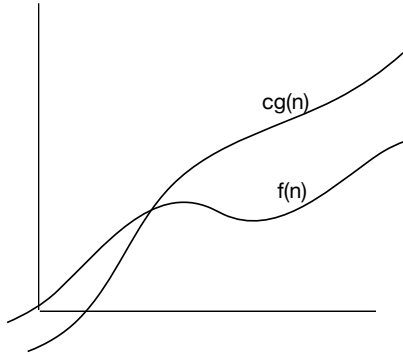
- Analyse af algoritmer
 - Køretid
 - Pladsforbrug
- Asymptotisk notation
 - O , Θ og Ω -notation.
- Eksperimentiel analyse af algoritmer

Asymptotisk notation

- **Asymptotisk notation**.
 - O , Θ og Ω -notation.
 - Notation til at give **grænser** for funktioners **asymptotiske** vækst.
 - Velegnet til analyse af algoritmer.

O-notation

- Def. $f(n) = O(g(n))$ hvis $f(n) \leq cg(n)$ for store n .

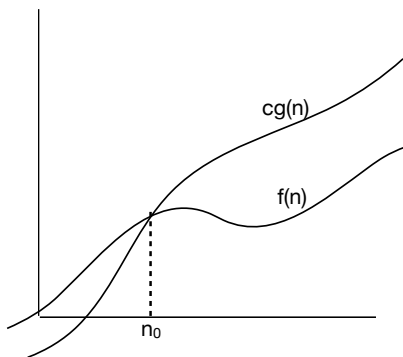


O-notation

- Eks. $f(n) = O(n^2)$ hvis $f(n) \leq cn^2$ for store n .
- $5n^2 = O(n^2)$?
 - $5n^2 \leq 5n^2$ for store n .
- $5n^2 + 3 = O(n^2)$?
 - $5n^2 + 3 \leq 6n^2$ for store n .
- $5n^2 + 3n = O(n^2)$?
 - $5n^2 + 3n \leq 6n^2$ for store n .
- $5n^2 + 3n^2 = O(n^2)$?
 - $5n^2 + 3n^2 = 8n^2 \leq 8n^2$ for store n .
- $5n^3 = O(n^2)$?
 - $5n^3 \geq cn^2$ for alle konstanter c for store n .

O-notation

- Def. $f(n) = O(g(n))$ hvis $f(n) \leq cg(n)$ for store n .
- Def. $f(n) = O(g(n))$ hvis der findes konstanter $c, n_0 > 0$, således at for alle $n \geq n_0$ gælder at $f(n) \leq cg(n)$



O-notation

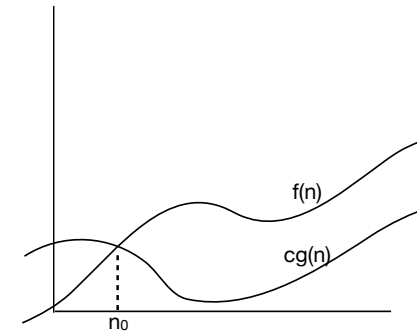
- Notation.
 - $O(g(n))$ er en mængde af funktioner.
 - Tænk på $=$ som \in eller \subseteq .
 - Man kan skrive $f(n) = O(n^2)$ men *ikke* $O(n^2) = f(n)$.

O-notation

- $f(n) = O(g(n))$ hvis $f(n) \leq cg(n)$ for store n .
- **Opgave.**
 - Lad $f(n) = 3n + 2n^3 - n^2$ og $h(n) = 4n^2 + \log n$ (Basen på logaritmer er altid 2 medmindre andet er angivet).
 - Hvilke af følgende udsagn er korrekte?
 - $f(n) = O(n)$
 - $f(n) = O(n^3)$
 - $f(n) = O(n^4)$
 - $h(n) = O(n^2 \log n)$
 - $h(n) = O(n^2)$
 - $h(n) = O(f(n))$
 - $f(n) = O(h(n))$

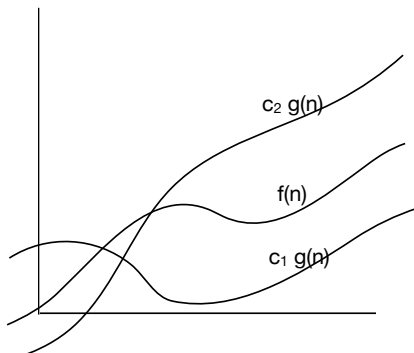
Ω -notation

- **Def.** $f(n) = \Omega(g(n))$ hvis der findes konstanter $c, n_0 > 0$, således at for alle $n \geq n_0$ gælder at $f(n) \geq cg(n)$



Θ -notation

- **Def.** $f(n) = \Theta(g(n))$ hvis $f(n)$ er både $O(g(n))$ og $\Omega(g(n))$



Asymptotisk notation

- $f(n) = O(g(n))$ hvis $f(n) \leq cg(n)$ for store n .
- $f(n) = \Omega(g(n))$ hvis $f(n) \geq cg(n)$ for store n .
- $f(n) = \Theta(g(n))$ hvis $f(n)$ er både $O(g(n))$ og $\Omega(g(n))$
- **Opgave.** Hvilke af følgende udsagn er korrekte? ($\log^k n$ betegner $(\log n)^k$)
 - $n \log^3 n = O(n^2)$
 - $2^n + 5n^7 = \Omega(n^3)$
 - $n^2(n - 5)/5 = \Theta(n^2)$
 - $4n^{1/100} = \Omega(n)$
 - $n^3/300 + 15 \log n = \Theta(n^3)$
 - $2^{\log n} = O(n)$
 - $\log^2 n + n + 7 = \Omega(\log n)$

Asymptotisk notation

- **Nyttige egenskaber.**

- Ethvert polynomie vokser proportionalt med det største led.

$$a_0 + a_1n + a_2n^2 + \dots + a_dn^d = \Theta(n^d)$$

- Alle logaritmer er asymptotisk ens.

$$\log_a(n) = \frac{\log_b n}{\log_b a} = \Theta(\log_c(n)) \quad \text{for alle konstanter } a, b > 0$$

- Enhver logaritme vokser langsommere end ethvert polynomie

$$\log(n) = O(n^d) \quad \text{for alle } d > 0$$

- Ethvert polynomie vokser langsommere end enhver eksponentialfunktion

$$n^d = O(r^n) \quad \text{for alle } d > 0 \text{ og } r > 1$$

Typiske køretider

```
for i = 1 to n  
<  $\Theta(1)$  tids operation >
```

```
for i = 1 to n  
  for j = 1 to n  
    <  $\Theta(1)$  tids operation >
```

```
for i = 1 to n  
  for j = i to n  
    <  $\Theta(1)$  tids operation >
```

Typiske køretider

$$T(n) = \begin{cases} T(n/2) + \Theta(1) & \text{hvis } n > 1 \\ \Theta(1) & \text{hvis } n = 1 \end{cases}$$

$$T(n) = \begin{cases} 2T(n/2) + \Theta(n) & \text{hvis } n > 1 \\ \Theta(1) & \text{hvis } n = 1 \end{cases}$$

$$T(n) = \begin{cases} T(n/2) + \Theta(n) & \text{hvis } n > 1 \\ \Theta(1) & \text{hvis } n = 1 \end{cases}$$

$$T(n) = \begin{cases} 2T(n/2) + \Theta(1) & \text{hvis } n > 1 \\ \Theta(1) & \text{hvis } n = 1 \end{cases}$$

Analyse af algoritmer

- Analyse af algoritmer
 - Køretid
 - Pladsforbrug
- Asymptotisk notation
 - O , Θ og Ω -notation.
- Eksperimentiel analyse af algoritmer

Eksperimentiel analyse

- **Udfordring.** Kan man eksperimentielt bestemme (et godt bud på) køretiden?
- **Fordoblingsteknik.**
 - Kør programmet for forskellige størrelser og mål køretiden (manuelt eller automatisk).
 - Kig på hvilken faktor køretiden vokser med når man **fordobler** størrelsen af input.

- **Eks.**

- Input vokser med faktor 2 \Rightarrow Tid vokser med en faktor ~ 4 .

- \Rightarrow Algoritmen har kvadratisk køretid:

- $T(n) = cn^2$
- $T(2n) = c(2n)^2 = c2^2n^2 = c4n^2$
- $T(2n)/T(n) = 4$

n	Tid i min	Ratio
5000	0	-
10000	0.2	-
20000	0.6	3
40000	2.3	3.8
80000	9.4	4
160000	37.8	4

Analyse af algoritmer

- Analyse af algoritmer
 - Køretid
 - Pladsforbrug
- Asymptotisk notation
 - O , Θ og Ω -notation.
- Eksperimentiel analyse af algoritmer