Analyse af algoritmer

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

Analyse af algoritmer

- Mål: Bestemme og forudsige ressourceforbrug og korrekthed af algoritmer.
- Eksempel:
  - Virker min algoritme til at beregne korteste veje?
  - Hvor meget tid bruger den på at finde den korteste vej?
  - Kan den skalere til 10.000 forespørgsler per sekund?
  - Løber den tør for hukommelse når kortene er store?
  - Hvordan spiller den sammen med hukommelseshierarkiet? (Ex. cache)
- Dette kursus:
  - Korrekthed, køretid, pladsforbrug.
  - Teoretisk og eksperimentel analyse.

Kørselstid/køretid

- Tid. Antallet af skridt, algoritmen udfører, som funktion af inputstørrelsen.
- Skridt:
  - Tilgå hukommelse (læse/skrive), f.eks. `x=y;`, `i=i+1;`, `A[i]`, ...
  - Aritmetiske/boolske operationer, såsom `+`,`-`,`*`,`/`,`&&`,`||`,`&`,`|`,`^`,`>>`,`<<` ...
  - Sammenligninger, såsom $<$, $>$, $=$, $\leq$, $\geq$, $\neq$, ...
  - Programflow, såsom if-then-else, while, for, goto, funktionskald.
- Terminologi: Kørselstid, køretid, tid, tidskompleksitet.

Tid

- Værstefaldskørestid (worst-case running time)
  Maksimal køretid over alle input af størrelse n.
- Bedstefaldskørestid (best-case running time)
  Minimal køretid over alle input af størrelse n.
- Gennemsnitslig køretid (average case running time)
  Givet en fordeling over input af størrelse n, den gennemsnitslige køretid for input fra fordelingen.
  Når ingen fordeling er specificeret, antag uniform fordeling:
  Gennemsnittet af køretiden over alle input af størrelse n.
- Terminologi: Tid betyder værstefaldskøretid
  (medmindre vi eksplicit skriver andet)
- Andre varianter: Amortiseret, randomiseret, deterministisk, nondeterministisk etc.

Plads

- Plads. Antallet af hukommelsesceller eller lagerpladser som algoritmen bruger.
- Lagerpladser
  - En variabel eller en pointer, en lagerplads.
  - Array af længde k, k lagerpladser.
- Terminologi. Pladsforbrug, plads, pladskompleksitet, lagerplads, ...

Analyse af algoritmer

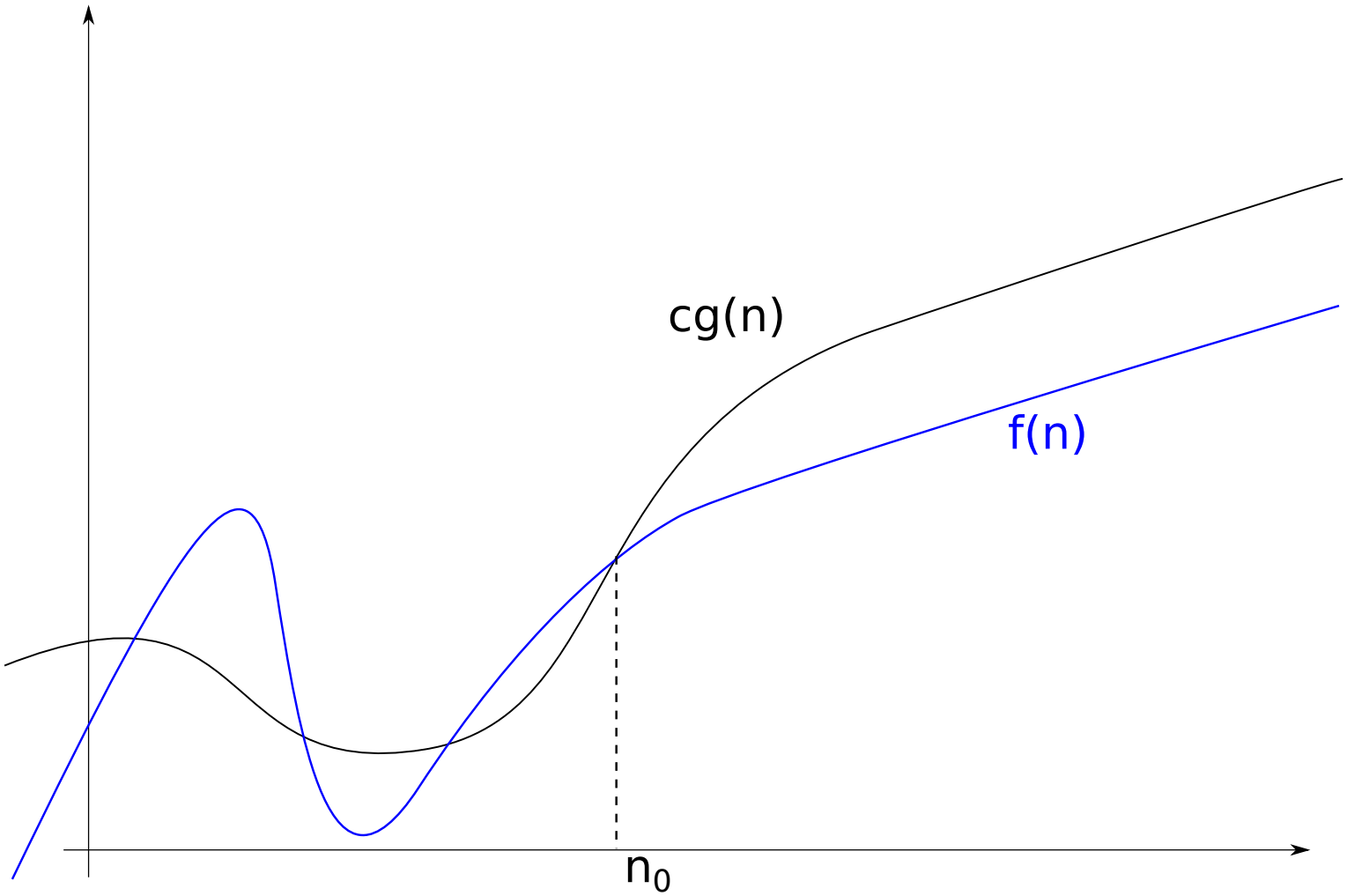
- Analyse af algoritmer
  - Køretid
  - Pladsforbrug
- Asymptotisk notation
  - O, Ω og Θ-notation.
- Eksperimentel analyse af algoritmer

Asymptotisk notation

- Asymptotisk notation.
  - O, Ω, Θ-notation.
  - Angiver grænser for funktioners asymptotiske vækst.
  - Velegnet til analyse af algoritmer.

O-notation

- Definition: $f(n)$ er $O(g(n))$ hvis $f(n) \leq c \cdot g(n)$ for store værdier af n.
  (Dvs. der findes en konstant c og et tal n_0 så $f(n) \leq c \cdot g(n)$ når $n \geq n_0$.)

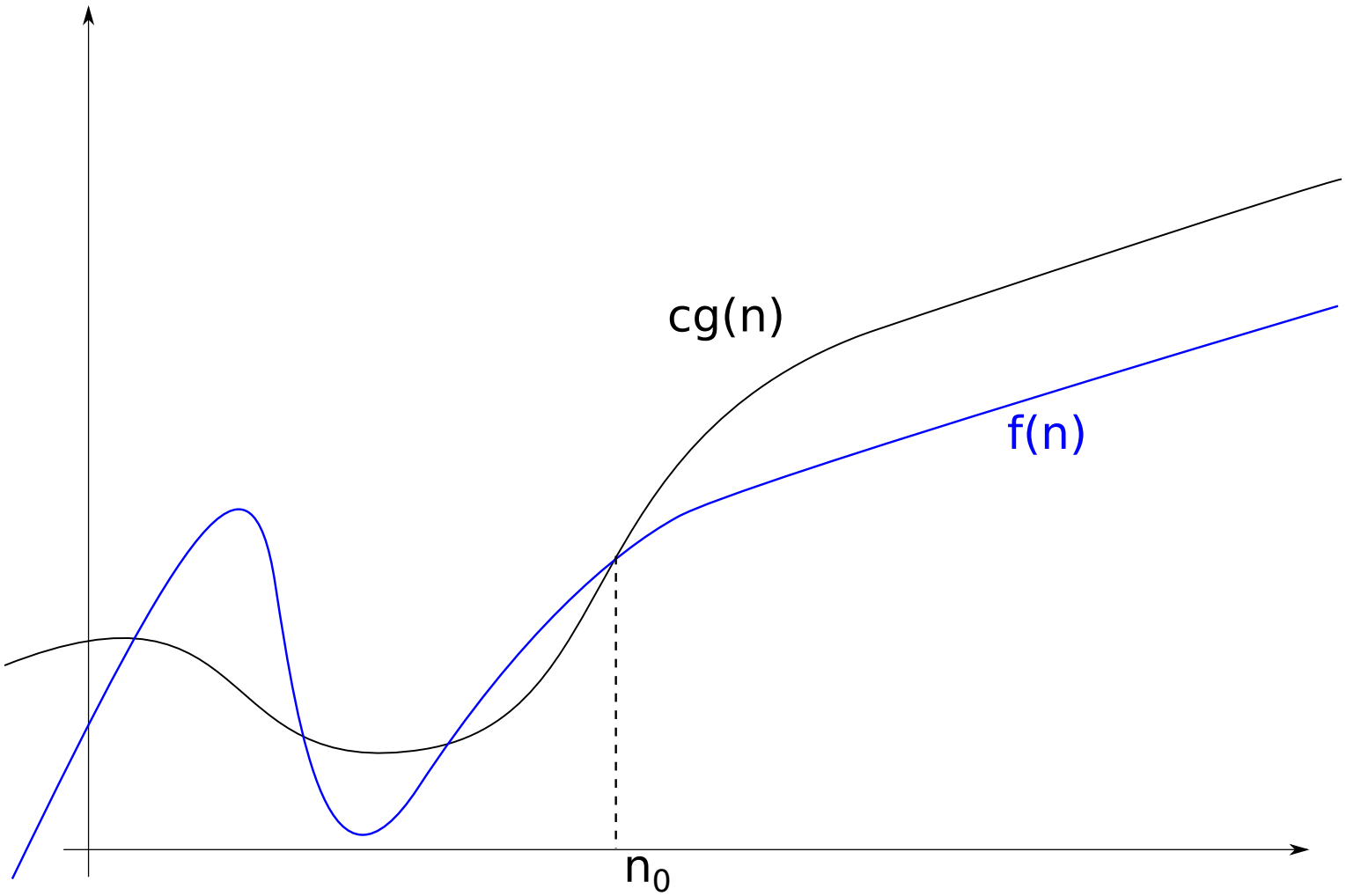


O-notation

- Definition: $f(n)$ er $O(g(n))$ hvis $f(n) \leq c \cdot g(n)$ for store værdier af n.
  (Dvs. der findes en konstant c og et tal n_0 så $f(n) \leq c \cdot g(n)$ når $n \geq n_0$.)
- Notation: $f(n) = O(g(n))$.
  - Eks. $f(n)$ er $O(n^2)$ hvis $f(n) \leq cn^2$ for store n.
  - $5n^2 = O(n^2)$?
    $5n^2 \leq 5n^2$ for store n (faktisk alle 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. (Faktisk alle n.)
  - $5n^3 = O(n^2)$?
    $5n^3 > cn^2$ for store n uanset valg af konstant c.
    Antag for modstrid $5n^3 \leq cn^2 \Rightarrow 5n \leq c \Rightarrow c$ er ikke konstant.

O-notation

- Definition: $f(n)$ er $O(g(n))$ hvis $f(n) \leq c \cdot g(n)$ for store værdier af n.
- Definition: $f(n)$ er $O(g(n))$ hvis der findes en konstant c og et tal n_0 så $f(n) \leq c \cdot g(n)$ når $n \geq n_0$.



O-notation

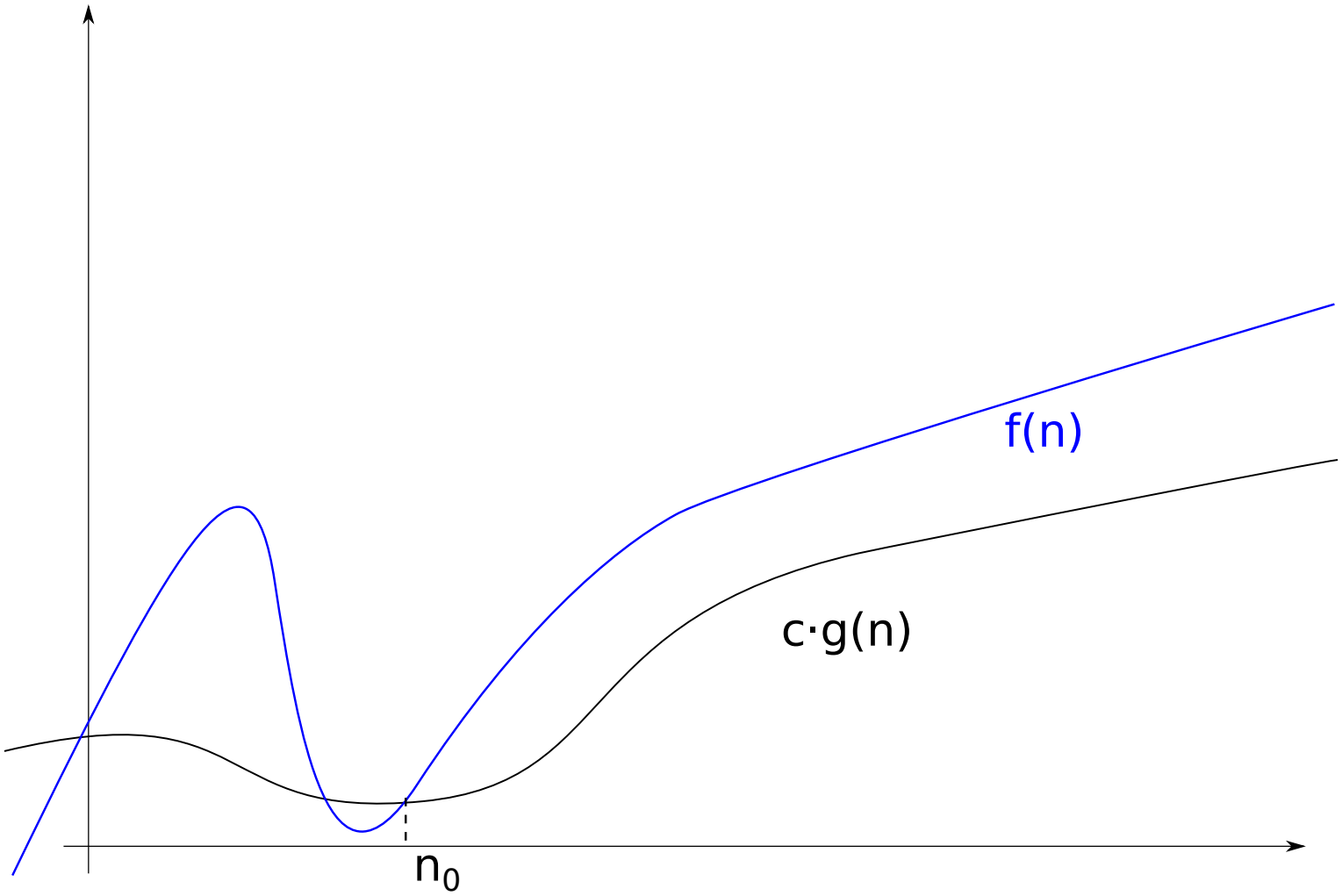
- Notation.
  - $O(g(n))$ er en mængde af funktioner.
  - Tænk på $f(n) = O(g(n))$ som notation for $f(n) \in O(g(n))$.
  - Man skriver $f(n) = O(g(n))$ men aldrig $O(g(n)) = f(n)$.

O-notation

- $f(n) = O(g(n))$ hvis $f(n) \leq c \cdot g(n)$ for store værdier af n. (Dvs. der findes en konstant c og et tal n_0 så $f(n) \leq c \cdot g(n)$ når $n \geq n_0$.)
- Opgave
  - $f(n) = 3n + 2n^3 - n^2$ og $h(n) = 4n^2 + \log_2(n)$.
  - Hvilke af følgende udsagn er korrekte?
  - $f(n) = O(n)$
  - $f(n) = O(n^3)$
  - $f(n) = O(n^4)$
  - $h(n) = O(n^2 \log_2 n)$
  - $h(n) = O(n^2)$
  - $h(n) = O(f(n))$
  - $f(n) = O(h(n))$

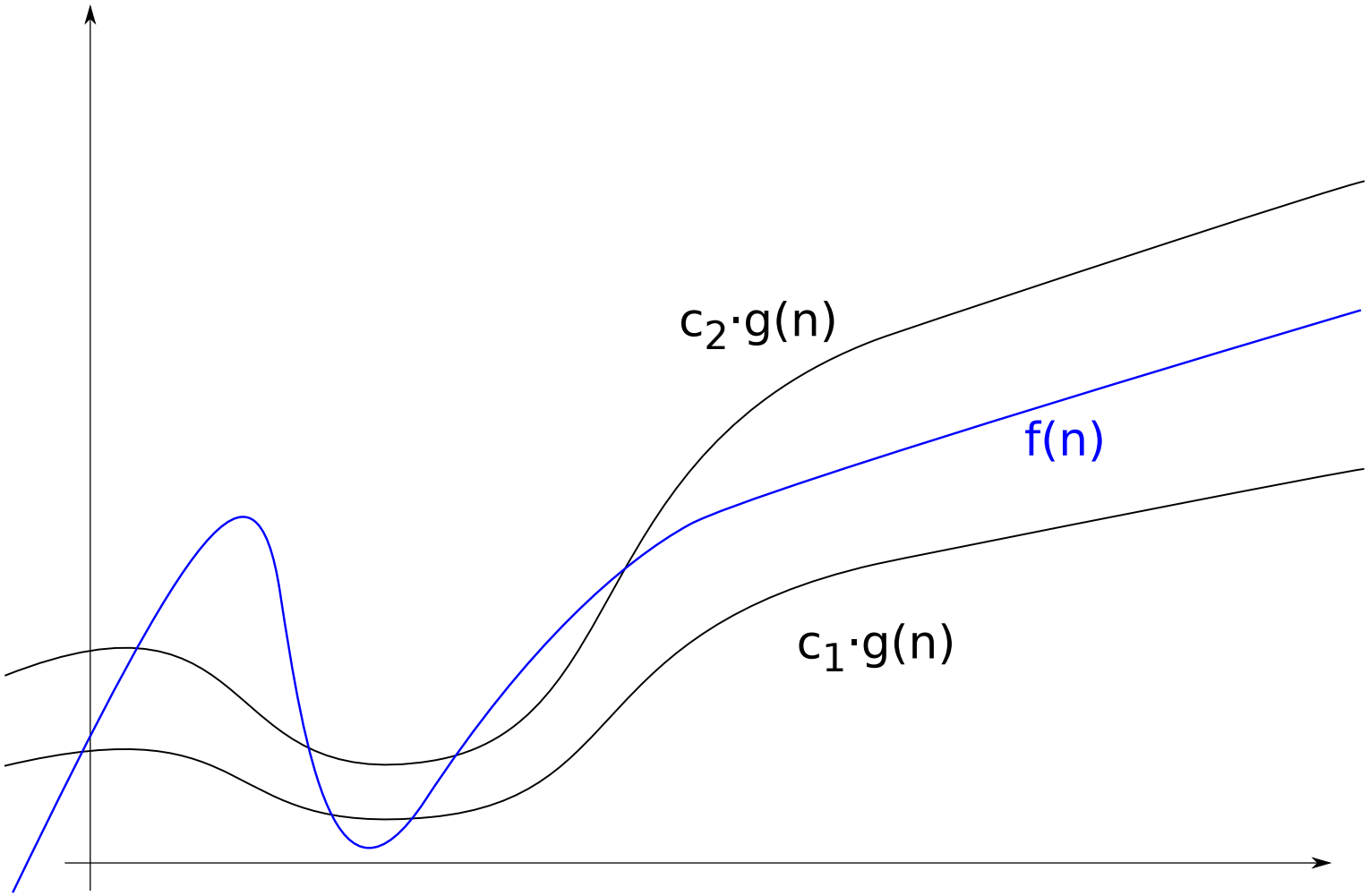
Ω-notation

- $f(n) = \Omega(g(n))$ hvis $f(n) \geq c \cdot g(n)$ for store værdier af n. Dvs. der findes en konstant c og et tal n_0 så $f(n) \geq c \cdot g(n)$ når $n \geq n_0$.



Θ-notation

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



Asymptotisk notation

- $f(n) = O(g(n))$ hvis $f(n) \leq c \cdot g(n)$ for store værdier af n.
- $f(n) = \Omega(g(n))$ hvis $f(n) \geq c \cdot g(n)$ for store værdier af n.
- $f(n) = \Theta(g(n))$ hvis både $f(n) = O(g(n))$ og $f(n) = \Omega(g(n))$.
- Opgave: hvilke udsagn er sande? (notation: $\log^k n$ betyder $(\log n)^k$)
  - $n \log^3 n = O(n^2)$?
  - $2^n + 5n^7 = \Omega(n^3)$?
  - $n^2 \cdot (n-5)/5 = \Theta(n^2)$?
  - $4n^{1/1000} = \Omega(n)$?
  - $n^3/300 + 15 \log_2 n = \Theta(n^3)$?
  - $2^{\log_2 n} = O(n)$?
  - $(\log_2 n)^2 + n + 7 = \Omega(\log n)$?

Asymptotisk notation

- Nyttige egenskaber
  - Et polynomium vokser proportionalt med sit største led.

$$a_0 + a_1 n + a_2 n^2 + \ldots + a_d n^d = \Theta(n^d)$$

  - Alle logaritmer er asymptotisk ens.

$$\log_a(n) = \frac{\log_b(n)}{\log_b(a)} = \Theta(\log_b(n)) \quad \text{for konstante } a, b$$

  - Enhver logaritme vokser langsommere end ethvert polynomium

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

  - Ethvert polynomium vokser langsommere end enhver eksponentialfunktion

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

Typiske køretider

for $i = 1$ to n **do**
    $\Theta(1)$-tids operation

for $i = 1$ to n **do**
    for $j = 1$ to n **do**
        $\Theta(1)$-tids operation

for $i = 1$ to n **do**
    for $j = i$ to n **do**
        $\Theta(1)$-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, Ω og Θ-notation.
- Eksperimentel analyse af algoritmer

Eksperimentel analyse af algoritmer

- Udfordring: Kan man eksperimentelt bestemme (et godt bud på) køretiden?
- Fordoblingsteknik:
  - Kør programmet for forskellige størrelser og mål køretiden (manuelt eller automatisk).
  - Hvilken faktor vokser køretiden med, når størrelsen af input fordobles?

n	tid	forhold
5000	0	-
10000	0,2	-
20000	0,6	3
40000	2,3	3,8
80000	9,4	4
160000	37,8	4

- Eksempel:

  Input vokser med faktor 2 $\Rightarrow$ Tid vokser med cirka en faktor 4

  Algoritmen ser ud til at have kvadratisk køretid:

  - $T(n) = c \cdot n^2$
  - $T(2n) = c \cdot (2n)^2 = c \cdot 2^2 \cdot n^2 = c \cdot 4 \cdot n^2$
  - $T(2n)/T(n) = 4$

Analyse af algoritmer

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