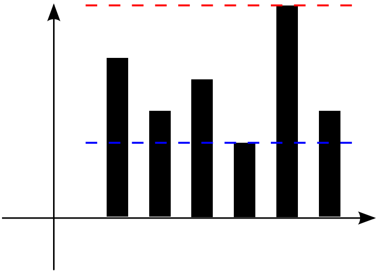
Algoritmer og køretid

Køretider for algoritmer

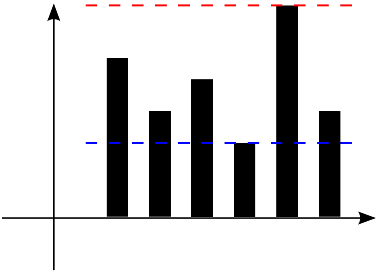
- Worst case
- Best case
- Average case



Køretid for de forskellige input af størrelse n

Køretider for algoritmer

- Worst case
- Best case
- Average case

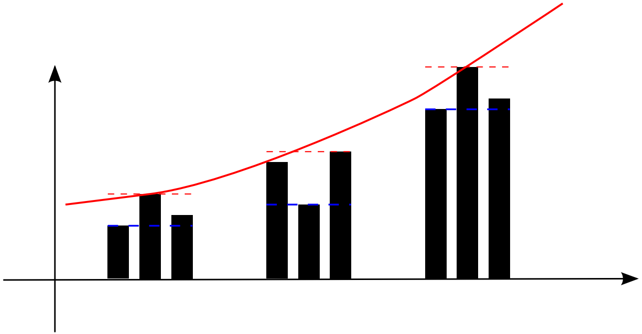


Køretid for de forskellige input af størrelse n

Worst case giver en garanti, og er mest interessant (average case kunne være interessant, men er ofte svær at definere på realistisk måde, og er ofte svær at beregne).

Voksehastighed

Worstcase køretid er normalt en voksende funktion af inputstørrelsen n:



Køretid for de forskellige input af stigende størrelse n

Voksehastighed

Eksempler:

- Lineær søgning

Voksehastighed

Eksempler:

- Lineær søgning: worst case n sammenligninger.

Voksehastighed

Eksempler:

- Lineær søgning: worst case n sammenligninger.
- Binær søgning

Voksehastighed

Eksempler:

- Lineær søgning: worst case n sammenligninger.
- Binær søgning: worst case $\log_2 n$ sammenligninger.

$$(n/2^k = 1 \Leftrightarrow n = 2^k \Leftrightarrow \log_2 n = k)$$

Voksehastighed

Eksempler:

- Lineær søgning: worst case n sammenligninger.
- Binær søgning: worst case $\log_2 n$ sammenligninger.

$$(n/2^k = 1 \Leftrightarrow n = 2^k \Leftrightarrow \log_2 n = k)$$

- InsertionSort

Voksehastighed

Eksempler:

- Lineær søgning: worst case n sammenligninger.
- Binær søgning: worst case $\log_2 n$ sammenligninger.

$$(n/2^k = 1 \Leftrightarrow n = 2^k \Leftrightarrow \log_2 n = k)$$

- InsertionSort: worst case $\sum_{i=1}^{n-1} i = n(n-1)/2 < n^2/2$ sammenligninger.

Forskellige voksehastigheder

$$f(n) \quad = \quad \log_2 n, \quad n, \quad n \cdot \log_2 n, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n.$$

[$f(n)$ = antal operationer (worst case) for input af størrelse n]

Forskellige voksehastigheder

$$f(n) \quad = \quad \log_2 n, \quad n, \quad n \cdot \log_2 n, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n.$$

$[f(n)$ = antal operationer (worst case) for input af størrelse $n]$

Udføres f.eks. 10^9 operationer per sekund på computeren, hvor store input kan man klare på:

- 1 sekund (10^9 operationer)?
- 1 minut ($6 \cdot 10^{10}$ operationer)?
- 1 dag ($9 \cdot 10^{13}$ operationer)?
- 1 år ($3 \cdot 10^{16}$ operationer)?

Forskellige voksehastigheder

$$f(n) \quad = \quad \log_2 n, \quad n, \quad n \cdot \log_2 n, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n.$$

$$[f(n) = \text{antal operationer (worst case) for input af størrelse } n]$$

Udføres f.eks. 10^9 operationer per sekund på computeren, hvor store input kan man klare på:

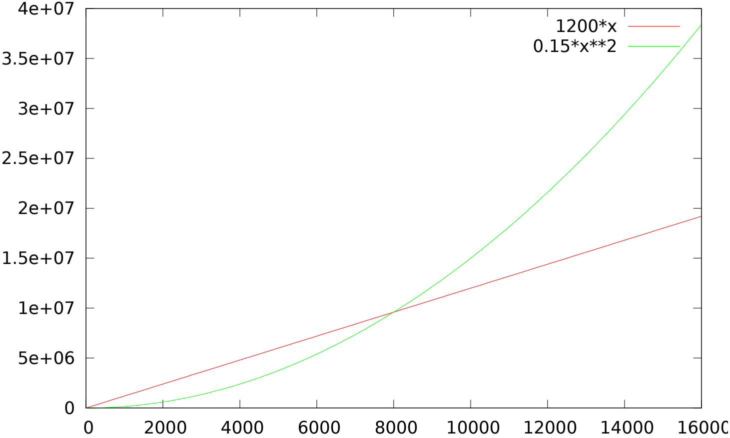
- 1 sekund (10^9 operationer)?
- 1 minut ($6 \cdot 10^{10}$ operationer)?
- 1 dag ($9 \cdot 10^{13}$ operationer)?
- 1 år ($3 \cdot 10^{16}$ operationer)?

	1 sekund	1 minut	1 dag	1 år
n	$1 \cdot 10^9$	$6 \cdot 10^{10}$	$9 \cdot 10^{13}$	$3 \cdot 10^{16}$
$n \cdot \log_2 n$	$4 \cdot 10^7$	$2 \cdot 10^9$	$2 \cdot 10^{12}$	$6 \cdot 10^{14}$
n^2	$3 \cdot 10^4$	$2 \cdot 10^5$	$9 \cdot 10^6$	$2 \cdot 10^8$
n^3	$1 \cdot 10^3$	$4 \cdot 10^3$	$4 \cdot 10^4$	$3 \cdot 10^5$
2^n	30	36	46	55
$10 \cdot n^2 + 2 \cdot n + 50$	$1 \cdot 10^4$	$8 \cdot 10^4$	$3 \cdot 10^6$	$5 \cdot 10^7$

Multiplikative konstanter

Multiplikative konstanter ligegyldige hvis voksehastighed er forskellig:

$$f(n) = 1200n \qquad\qquad g(x) = 0.15n^2$$

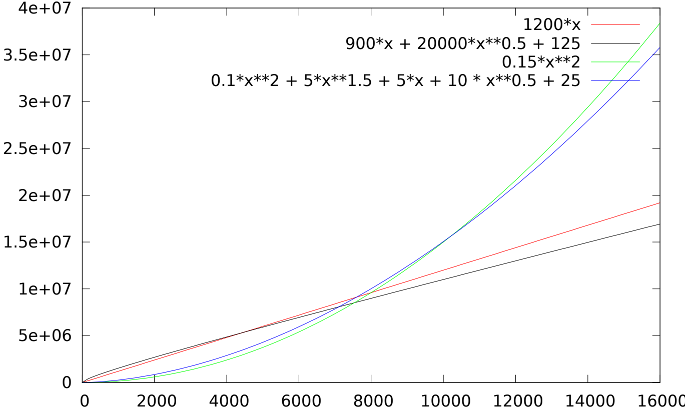


Dominerende led

Dominerende led bestemmer voksehastighed:

$$f(n) = 1200n \qquad g(x) = 0.15n^2$$

$$h(n) = 900n + 20000n^{0.5} + 125 \qquad k(n) = 0.1n^2 + 5n^{1.5} + 5n + 10n^{0.5} + 25$$



Sammenligne voksehastighed

Vi vil sammenligne funktioners essentielle voksehastighed på en måde hvor vi ser bort fra multiplikative konstanter og ikke-dominerende led.

Sammenligne voksehastighed

Vi vil sammenligne funktioners essentielle voksehastighed på en måde hvor vi ser bort fra multiplikative konstanter og ikke-dominerende led.

Derved behøver vi ikke diskutere helt præcis hvor mange basale operationer, en algoritme bruger.

Sammenligne voksehastighed

Vi vil sammenligne funktioners essentielle voksehastighed på en måde hvor vi ser bort fra multiplikative konstanter og ikke-dominerende led.

Derved behøver vi ikke diskutere helt præcis hvor mange basale operationer, en algoritme bruger.

Dette er en fordel:

- Det er nærmest umuligt at beregne helt præcist.
- Det afhænger alligevel af implementation, compiler, maskine.
- Hvis algoritme A har dårligere essentiel voksehastighed end algoritme B, vil A vinde over B når input bliver stort nok (uanset implementation, compiler, maskine).

Asymptotisk notation

Når vi sammenligner

$$= \qquad \leq \qquad <$$

voksehastighed for funktioner bruger vi af historiske årsager flg. betegnelser:

$$\Theta \qquad O \qquad o$$

Hvilket udtales således:

"Store Theta" "Store O" "lille o"

Eksempler

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^2)$

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^3)$

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^3)$
- $10 \cdot n^2 + 2 \cdot n + 50 = o(n^3)$

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^3)$
- $10 \cdot n^2 + 2 \cdot n + 50 = o(n^3)$
- $n \log n = o(n^2)$

Eksempler

- $10 \cdot n^2 + 2 \cdot n + 50 = \Theta(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^2)$
- $10 \cdot n^2 + 2 \cdot n + 50 = O(n^3)$
- $10 \cdot n^2 + 2 \cdot n + 50 = o(n^3)$
- $n \log n = o(n^2)$
- $n = o(n \log n)$

Generel metode

Som regel kan man sammenligne to voksehastigheder ved at dividere dem med hinanden og se hvad der sker når n vokser.

Generel metode

Som regel kan man sammenligne to voksehastigheder ved at dividere dem med hinanden og se hvad der sker når n vokser.

Eksempler:

Generel metode

Som regel kan man sammenligne to voksehastigheder ved at dividere dem med hinanden og se hvad der sker når n vokser.

Eksempler:

$$\frac{20n^2 + 17n + 312}{n^2} = \frac{20 + 17/n + 312/n^2}{1} \to 20 \text{ for } n \to \infty$$

Generel metode

Som regel kan man sammenligne to voksehastigheder ved at dividere dem med hinanden og se hvad der sker når n vokser.

Eksempler:

$$\frac{20n^2 + 17n + 312}{n^2} = \frac{20 + 17/n + 312/n^2}{1} \rightarrow 20 \text{ for } n \rightarrow \infty$$

$$\frac{20n^2 + 17n + 312}{n^3} = \frac{20/n + 17/n^2 + 312/n^3}{1} \rightarrow 0 \text{ for } n \rightarrow \infty$$

Generel metode

Som regel kan man sammenligne to voksehastigheder ved at dividere dem med hinanden og se hvad der sker når n vokser.

Eksempler:

$$\frac{20n^2 + 17n + 312}{n^2} = \frac{20 + 17/n + 312/n^2}{1} \to 20 \text{ for } n \to \infty$$

$$\frac{20n^2 + 17n + 312}{n^3} = \frac{20/n + 17/n^2 + 312/n^3}{1} \to 0 \text{ for } n \to \infty$$

$$\frac{n}{n \log n} = \frac{1}{\log n} \to 0 \text{ for } n \to \infty$$

Voksehastighed

Eksempler fra tidligere:

- Sekventiel søgning $= \Theta(n)$
- Binær søgning $= \Theta(\log_2 n)$
- InsertionSort $= \Theta(n^2)$