Analyse af algoritmer

Fokus i kurset

Recap fra sidst: i DM507/DM578/DS814/SE4-DMAD vil vi

> Beskrive eksisterende algoritmer, udvikle nye algoritmer og vurdere/analysere algoritmer.

Fokus i kurset

Recap fra sidst: i DM507/DM578/DS814/SE4-DMAD vil vi

> Beskrive eksisterende algoritmer, udvikle nye algoritmer og vurdere/analysere algoritmer.

Spørgsmål: hvad skal vi fokusere på, når vi analyserer en algoritme?

Analyse af algoritmer

Mindstekrav til algoritmer er **korrekthed:**

- Stopper for alle input (aldrig uendelig løkke).
- Giver korrekt output, når den stopper (dvs. giver et svar på vores problem).

Analyse af algoritmer

Mindstekrav til algoritmer er **korrekthed:**

- Stopper for alle input (aldrig uendelig løkke).
- Giver korrekt output, når den stopper (dvs. giver et svar på vores problem).

Korrekte algoritmer kan have forskellig **kvalitet:**

- Hastighed
- Pladsforbrug
- Kompleksitet af implementation
- Ekstra egenskaber (problemspecifikke)

Ingredienser i algoritmeanalyse

Vi har brug for:

- **Model af problem.** Individuelt for hvert problem (men ofte ret ligetil).

Ingredienser i algoritmeanalyse

Vi har brug for:

- **Model af problem.** Individuelt for hvert problem (men ofte ret ligetil).
- **Model af maskine.** Vi bruger RAM-modellen (se næste side).

Ingredienser i algoritmeanalyse

Vi har brug for:

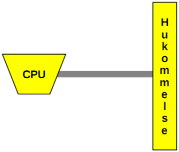
- **Model af problem.** Individuelt for hvert problem (men ofte ret ligetil).
- **Model af maskine.** Vi bruger RAM-modellen (se næste side).
- **Mål for kvalitet.** Vi fokuserer på tidsforbrug.

Ingredienser i algoritmeanalyse

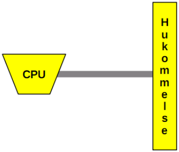
Vi har brug for:

- **Model af problem.** Individuelt for hvert problem (men ofte ret ligetil).
- **Model af maskine.** Vi bruger RAM-modellen (se næste side).
- **Mål for kvalitet.** Vi fokuserer på tidsforbrug.
- **Matematiske værktøjer.** Vi møder i dette kursus bl.a. asymptotisk notation, invarianter, induktion, rekursionsligninger.

RAM-modellen

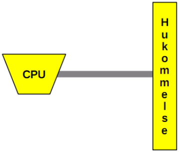


RAM-modellen



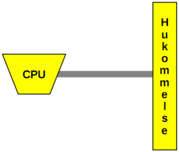
- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)

RAM-modellen



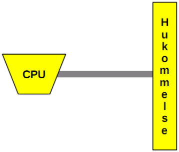
- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:

RAM-modellen



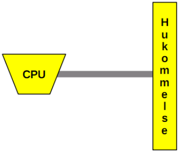
- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, ... to cellers indhold

RAM-modellen



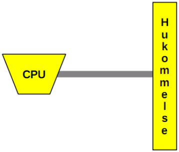
- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, . . . to cellers indhold
  - *flyt* indhold mellem to celler

RAM-modellen



- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, ... to cellers indhold
  - *flyt* indhold mellem to celler
  - *jump i program* (⇒ *løkke*, *forgrening*, *funktions/metode-kald*).

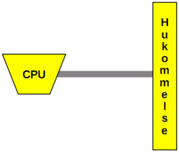
RAM-modellen



- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, ... to cellers indhold
  - *flyt* indhold mellem to celler
  - *jump i program* ($\Rightarrow$ *løkke*, *forgrening*, *funktions/metode-kald*).

Alle basale operationer antages at tage den samme tid.

RAM-modellen

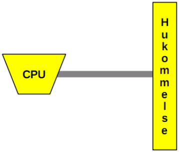


- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, ... to cellers indhold
  - *flyt* indhold mellem to celler
  - *jump i program* ($\Rightarrow$ *løkke*, *forgrening*, *funktions/metode-kald*).

Alle basale operationer antages at tage den samme tid.

- **Tid** for en algoritme: antal basale operationer udført.

RAM-modellen



- En hukommelse: En stor række af celler, hver med plads til et tal eller et tegn. (NB: liste/array = mange naboceller)
- En CPU med et antal basale operationer:
  - *plus*, *minus*, *gange*, *sammenlign*, ... to cellers indhold
  - *flyt* indhold mellem to celler
  - *jump i program* ($\Rightarrow$ *løkke*, *forgrening*, *funktions/metode-kald*).

Alle basale operationer antages at tage den samme tid.

- **Tid** for en algoritme: antal basale operationer udført.
- **Plads** for en algoritme: maks antal optagne hukommelsesceller.

Måle tidsforbrug: hvilket input?

For en givet størrelse n af input er der ofte mange forskellige konkrete input.

Måle tidsforbrug: hvilket input?

For en givet størrelse n af input er der ofte mange forskellige konkrete input. Eksempel:

$$\boxed{\textbf{Sortering} = \text{stille } n \text{ elementer i stigende orden}}$$

Måle tidsforbrug: hvilket input?

For en givet størrelse n af input er der ofte mange forskellige konkrete input. Eksempel:

$$\boxed{\textbf{Sortering} = \text{stille } n \text{ elementer i stigende orden}}$$

For $n = 8$ er følgende lister fire ud af mange mulige input:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

8,7,6,5,4,3,2,1

Måle tidsforbrug: hvilket input?

For en givet størrelse n af input er der ofte mange forskellige konkrete input. Eksempel:

$\boxed{\textbf{Sortering} = \text{stille } n \text{ elementer i stigende orden}}$

For $n = 8$ er følgende lister fire ud af mange mulige input:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

8,7,6,5,4,3,2,1

En algoritme har som regel *forskelligt* tidsforbrug på hver af disse.

Måle tidsforbrug: hvilket input?

For en givet størrelse n af input er der ofte mange forskellige konkrete input. Eksempel:

Sortering = stille n elementer i stigende orden

For $n = 8$ er følgende lister fire ud af mange mulige input:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

8,7,6,5,4,3,2,1

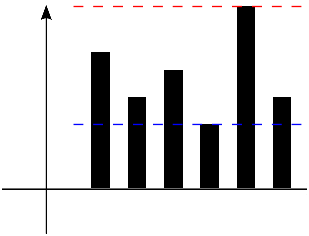
En algoritme har som regel *forskelligt* tidsforbrug på hver af disse.

Hvilke input skal vi bruge til at vurdere tidsforbruget?

Måle tidsforbrug: hvilket input?

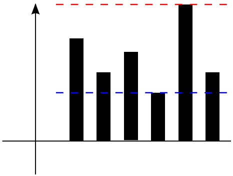
Måle tidsforbrug: hvilket input?

- Worst case (max over alle input af størrelse n)
- Average case (gennemsnit over en fordeling af input af størrelse n)
- Best case (min over alle input af størrelse n)



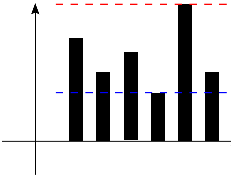
Køretid for de forskellige input af (samme) størrelse n

Worst case tidsforbrug



Worst case giver **garanti**. Er desuden ofte repræsentativ for average case (men kan også være mere pessimistisk).

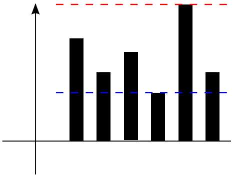
Worst case tidsforbrug



Worst case giver **garanti**. Er desuden ofte repræsentativ for average case (men kan også være mere pessimistisk).

Average case: Hvilken fordeling af input? Hvorfor er denne fordeling realistisk/relevant? Analyse er ofte (matematisk) svær at gennemføre.

Worst case tidsforbrug

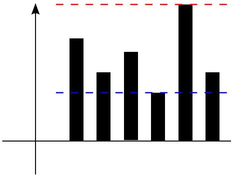


Worst case giver **garanti**. Er desuden ofte repræsentativ for average case (men kan også være mere pessimistisk).

Average case: Hvilken fordeling af input? Hvorfor er denne fordeling realistisk/relevant? Analyse er ofte (matematisk) svær at gennemføre.

Best case: Giver ofte ikke så megen relevant information.

Worst case tidsforbrug



Worst case giver **garanti**. Er desuden ofte repræsentativ for average case (men kan også være mere pessimistisk).

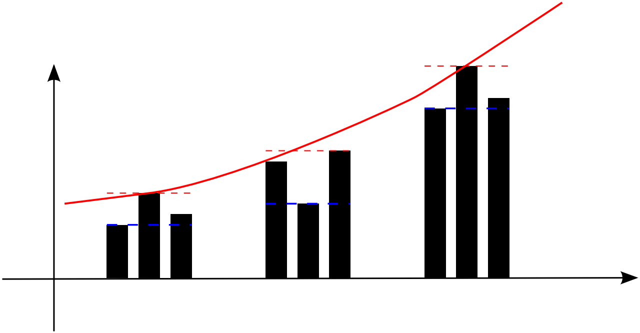
Average case: Hvilken fordeling af input? Hvorfor er denne fordeling realistisk/relevant? Analyse er ofte (matematisk) svær at gennemføre.

Best case: Giver ofte ikke så megen relevant information.

Næsten alle analyser i dette kursus er worst case.

Forskellige inputstørrelser

Worstcase køretid er normalt en voksende funktion af inputstørrelsen n:



Køretid for de forskellige input af stigende størrelse n

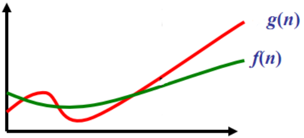
Voksehastighed

En analyse må derfor give en funktion $f(n)$ af inputstørrelsen n.

Voksehastighed

En analyse må derfor give en funktion $f(n)$ af inputstørrelsen n.

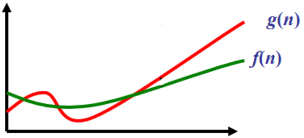
For at sammenligne algoritmers køretid har vi derfor brug for at sammenligne funktioner. Mere præcist vil vi gerne undersøge, hvordan to funktioner $f(n)$ og $g(n)$ vokser, når n stiger.



Voksehastighed

En analyse må derfor give en funktion $f(n)$ af inputstørrelsen n.

For at sammenligne algoritmers køretid har vi derfor brug for at sammenligne funktioner. Mere præcist vil vi gerne undersøge, hvordan to funktioner $f(n)$ og $g(n)$ vokser, når n stiger.

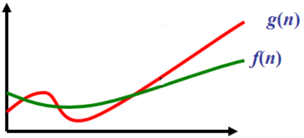


Specielt vil vi gerne kunne genkende, hvis f.eks. $g(n)$ altid vil overstige $f(n)$, når n bliver stor nok. Vi siger så, at $g(n)$ har større voksehastighed end $f(n)$, og vi vil normalt foretrække algoritmen med køretid $f(n)$.

Voksehastighed

En analyse må derfor give en funktion $f(n)$ af inputstørrelsen n.

For at sammenligne algoritmers køretid har vi derfor brug for at sammenligne funktioner. Mere præcist vil vi gerne undersøge, hvordan to funktioner $f(n)$ og $g(n)$ vokser, når n stiger.



Specielt vil vi gerne kunne genkende, hvis f.eks. $g(n)$ altid vil overstige $f(n)$, når n bliver stor nok. Vi siger så, at $g(n)$ har større voksehastighed end $f(n)$, og vi vil normalt foretrække algoritmen med køretid $f(n)$.

For små n er (næsten) alle algoritmer hurtige, så det er køretiden for store n, som er interessant.

Voksehastighed

Eksempler på funktioner listet efter stigende voksehastighed:

$$1, \quad \log n, \quad \sqrt{n}, \quad n, \quad n \log n,$$

$$n\sqrt{n}, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n$$

Voksehastighed

Eksempler på funktioner listet efter stigende voksehastighed:

$$1, \quad \log n, \quad \sqrt{n}, \quad n, \quad n \log n,$$

$$n\sqrt{n}, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n$$

Disse har ret forskellig effektivitet i praksis:

	n	$n \log n$	n^2	n^3	n^{10}	2^n
1 minut	$6,0 \cdot 10^{10}$	$1,9 \cdot 10^{9}$	245.000	3.910	12	36
1 måned	$2,6 \cdot 10^{15}$	$5,7 \cdot 10^{13}$	50.900.000	137.000	35	51

Tabellen viser, hvilke inputstørrelser n man kan klare, hvis algoritmen skal udføre $f(n)$ CPU-operationer, og den skal være færdig efter henholdsvis et minut og en måned. Det er antaget, at en CPU kan lave 10^9 operationer per sekund. Se udleveret program for beregningerne.

Asymptotisk voksehastighed

Vi laver senere en formel definition af begrebet

$$\boxed{\text{asymptotisk voksehastighed}}$$

for funktioner. Dette bliver vores værktøj til at sammenligne køretider af algoritmer.