# Algoritmeanalyse

# Algoritmeanalyse

Mindstekrav til algoritmer er korrekthed:

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

# Algoritmeanalyse

Mindstekrav til algoritmer er korrekthed:

- Stopper for alle input (aldrig uendelig løkke).
- Giver det søgte 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), f.eks. stabilitet af sortering.

# Algoritmeanalyse

Mindstekrav til algoritmer er korrekthed:

- Stopper for alle input (aldrig uendelig løkke).
- Giver det søgte 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), f.eks. stabilitet af sortering.

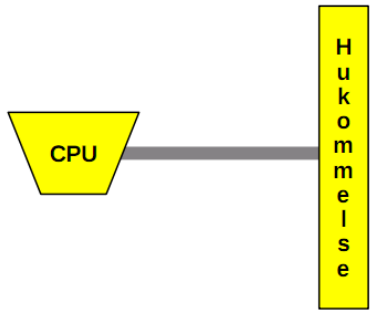
I dette kursus vil vi beskrive eksisterende algoritmer, udvikle nye algoritmer og analysere algoritmer for korrekthed og kvalitet.

# Ingredienser i algoritmeanalyse

Vi har brug for:

- Model af problem. Individuelt for hvert problem.
- Model af maskine. Vi bruger RAM-modellen.
- Mål for kvalitet. Vi fokuserer (mest) på tidsforbrug.
- Matematiske analyseværktøjer: Løkkeinvarianter, induktion, rekursionsligninger, asymptotisk notation.

# RAM-modellen




- En CPU
- En hukommelse ($\sim$ uendeligt array af celler).
- Et antal basale operationer: *add*, *sub*, *mult*, *shift*, *compare*, *flyt dataelement*, *jump i program* (*løkke*, *forgrening*, *metodekald*). Disse antages alle at tage samme tid.

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

# Måle tidsforbrug

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

Eksempel: for sortering af $n = 8$ tal er følgende nogle af de 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

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

Eksempel: for sortering af $n = 8$ tal er følgende nogle af de 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

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

Eksempel: for sortering af $n = 8$ tal er følgende nogle af de 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

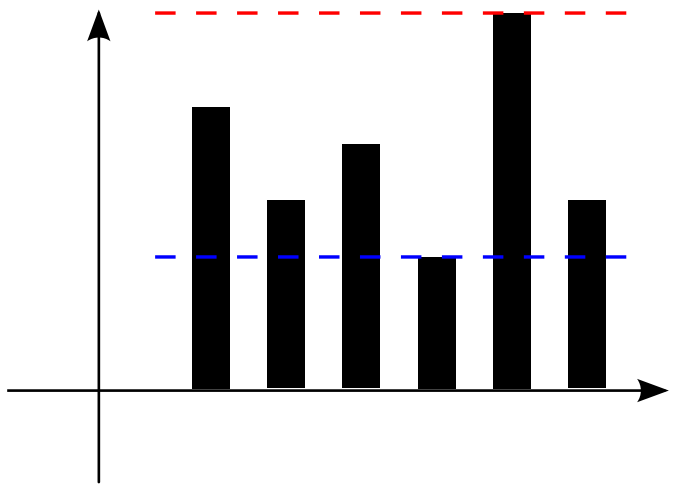
En algoritme har som regel forskelligt tidsforbrug på hver af disse.

Hvilke input skal vi bruge til at vurdere tidsforbruget?

# Måle tidsforbrug

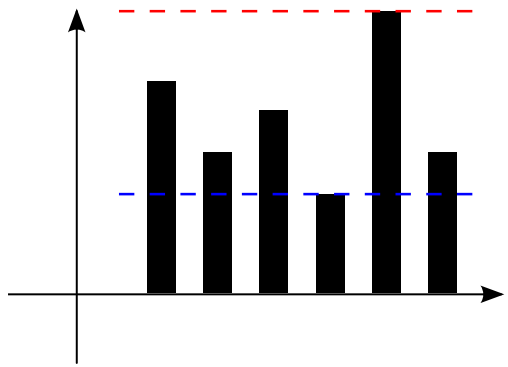
# Måle tidsforbrug

- 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 $n$)



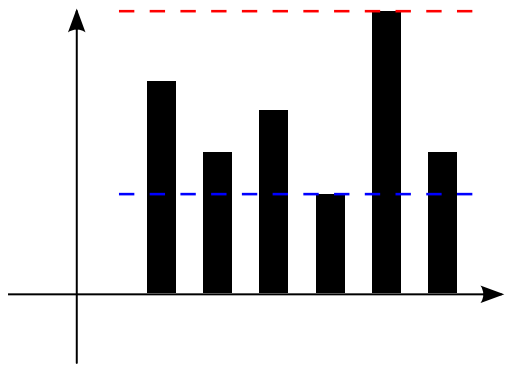
Køretid for de forskellige input af størrelse $n$

# Worst case tidsforbrug



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

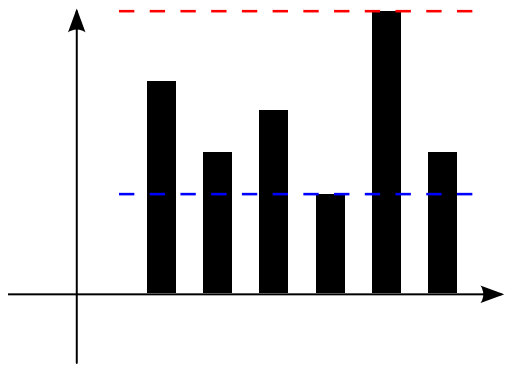
# Worst case tidsforbrug



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

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

# Worst case tidsforbrug

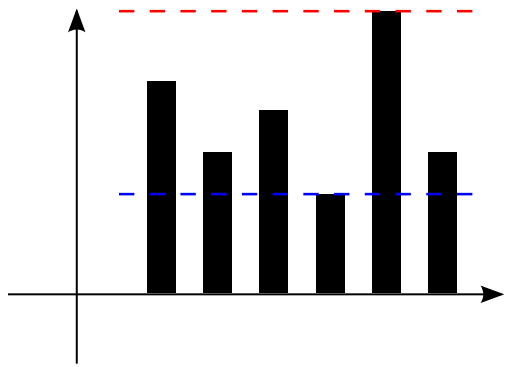


Worst case giver **garanti**. Er desuden ofte repræsentativ for average case (men kan også betydeligt mere pessimistisk).

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 betydeligt mere pessimistisk).

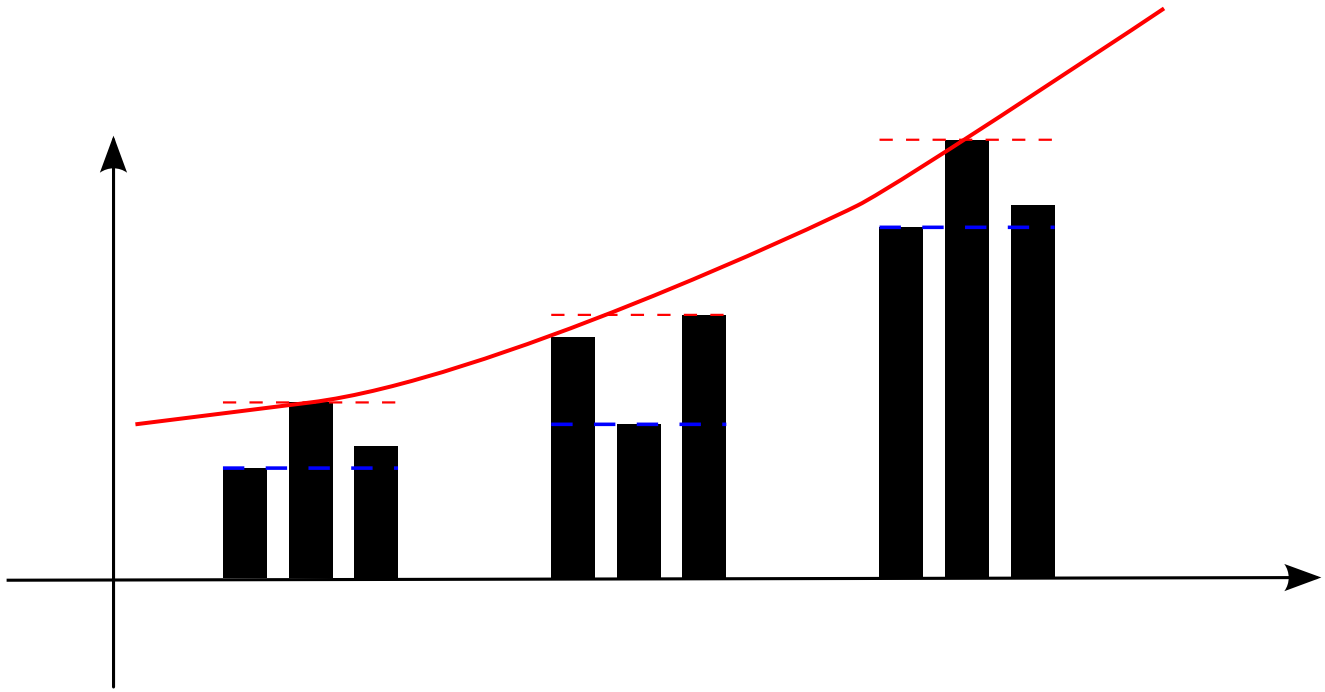
Average case: Hvilken fordeling af input? Hvorfor er denne fordeling realistisk/relevant? Analyse er ofte (matematisk) svær at gennemføre.

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 $n$

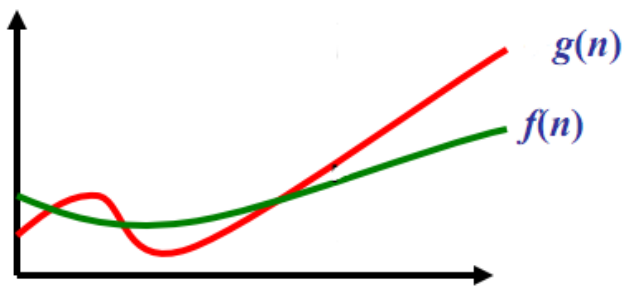
# Voksehastighed

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

# Voksehastighed

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

Vi har derfor brug for at sammenligne funktioner. Det relevante mål er voksehastighed af $f(n)$ når $n$ stiger - en hurtigere voksende funktion vil altid overhale en langsomt voksende funktion, når $n$ bliver stor nok. Og for små $n$ er (næsten) alle algoritmer hurtige.



# Voksehastighed

Eksempler (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 (stigende voksehastighed):

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

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

Senere: definition af asymptotisk voksehastighed for funktioner og sammenligninger heraf.