

# Hvilken maskine?



- Forskellige instruktioner.
- Forskellige hastigheder.
- Der kommer en ny model i morgen.

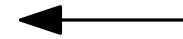
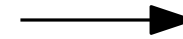
## Løsning: RAM-modellen!

- Abstrakt matematisk model af en computer.
- Simplere end alle virkelige computere.
- Skal aldrig opdateres.
- Alligevel realistisk.

# Løsning: RAM-modellen!

- Abstrakt matematisk model af en computer.
- Simplere end alle virkelige computere.
- Skal aldrig opdateres.
- Alligevel realistisk.

Program:  
Sekvens af basale  
instruktioner



Hukommelse

Plads 0
Plads 1
Plads 2
Plads 3
⋮

⋮

# Løsning: RAM-modellen!

- Abstrakt matematisk model af en computer.
- Simplere end alle virkelige computere.
- Skal aldrig opdateres.
- Alligevel realistisk.

Program:  
Sekvens af basale  
instruktioner

Hukommelse

Plads 0
Plads 1
Plads 2
Plads 3
⋮

⋮

Basale instruktioner:

Læse fra/skrive til hukommelsen

$+$ ,  $-$ ,  $\cdot$ ,  $/$  (aritmetiske op.)

and, or, xor, not (logiske op.)

Sammenligninger:  $==$ ,  $<$ ,  $\leq$ ,  $\neq$

Program flow: if-else, for, while, funktionskald

# Løsning: RAM-modellen!

- Abstrakt matematisk model af en computer.
- Simplere end alle virkelige computere.
- Skal aldrig opdateres.
- Alligevel realistisk.

Program:  
Sekvens af basale  
instruktioner

Hukommelse

Plads 0
Plads 1
Plads 2
Plads 3
⋮

⋮

Basale instruktioner:

Læse fra/skrive til hukommelsen

$+$ ,  $-$ ,  $\cdot$ ,  $/$  (aritmetiske op.)

and, or, xor, not (logiske op.)

Sammenligninger:  $==$ ,  $<$ ,  $\leq$ ,  $\neq$

Program flow: if-else, for, while, funktionskald

Hver instruktion tager konstant tid.

Køretid =  $\#$ instruktioner

# Ord/word

Hver plads kan indeholde et ord/word, fx:

- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Hukommelse  $H$

Plads 0
Plads 1
Plads 2
Plads 3
⋮

⋮

# Ord/word

Hver plads kan indeholde et ord/word, fx:

- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

RAM-model-kode:

Hukommelse  $H$

Plads 0
Plads 1
Plads 2
Plads 3
⋮

⋮

# Ord/word

Hver plads kan indeholde et ord/word, fx:

- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

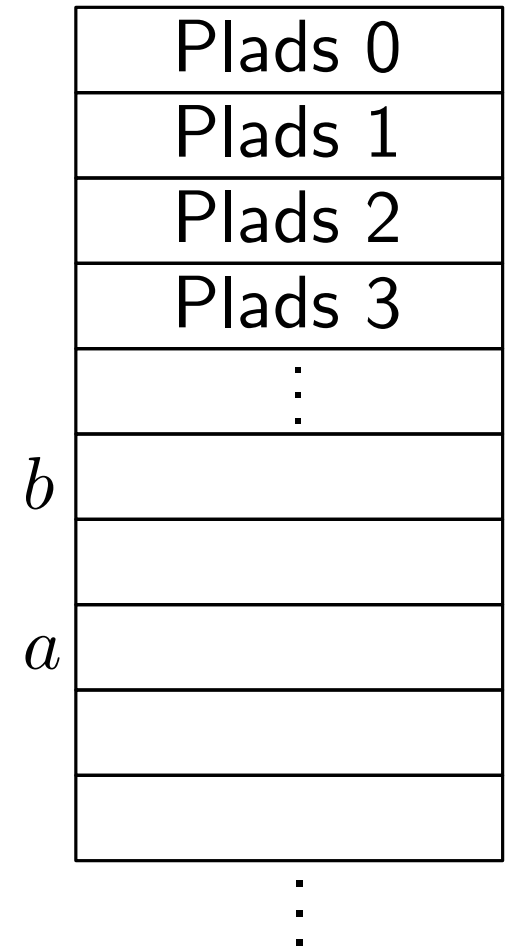
Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

```
 $a = 4$   
 $b = 1$   
 $b = b + 8$   
 $a = a + b$ 
```

RAM-model-kode:

Hukommelse  $H$





# Ord/word

Hver plads kan indeholde et ord/word, fx:

- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

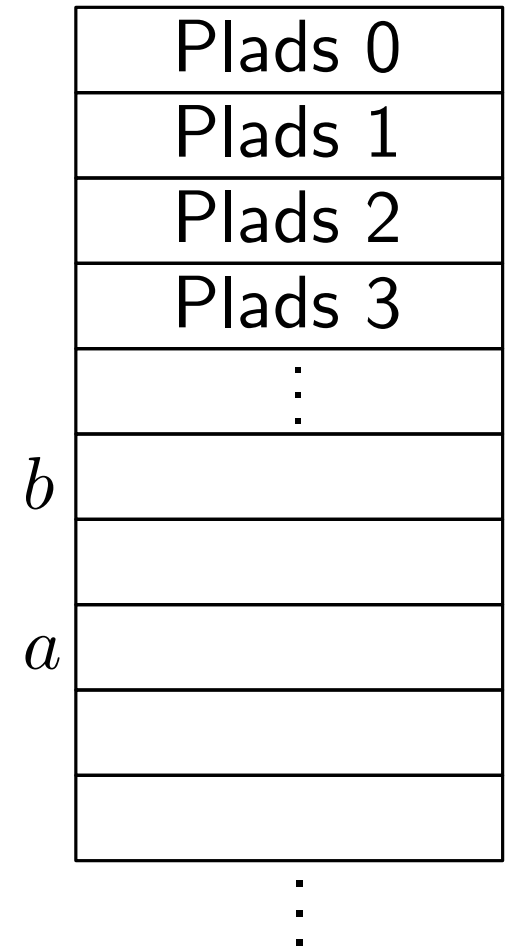
Pseudokode:

```
 $a = 4$   
 $b = 1$   
 $b = b + 8$   
 $a = a + b$ 
```

RAM-model-kode:

```
 $H[7] = 4$   
 $H[5] = 1$   
 $H[5] = H[5] + 8$   
 $H[7] = H[7] + H[5]$ 
```

Hukommelse  $H$



# Ord/word

Hver plads kan indeholde et ord/word, fx:

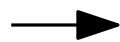
- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

```
 $a = 4$   
 $b = 1$   
 $b = b + 8$   
 $a = a + b$ 
```



RAM-model-kode:

```
 $H[7] = 4$   
 $H[5] = 1$   
 $H[5] = H[5] + 8$   
 $H[7] = H[7] + H[5]$ 
```

Hukommelse  $H$

	Plads 0
	Plads 1
	Plads 2
	Plads 3
	$\vdots$
$b$	
$a$	4
	$\vdots$

# Ord/word

Hver plads kan indeholde et ord/word, fx:

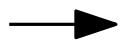
- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

```
 $a = 4$   
 $b = 1$   
 $b = b + 8$   
 $a = a + b$ 
```



RAM-model-kode:

```
 $H[7] = 4$   
 $H[5] = 1$   
 $H[5] = H[5] + 8$   
 $H[7] = H[7] + H[5]$ 
```

Hukommelse  $H$

	Plads 0
	Plads 1
	Plads 2
	Plads 3
	$\vdots$
$b$	1
$a$	4
	$\vdots$

# Ord/word

Hver plads kan indeholde et ord/word, fx:

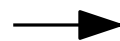
- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

```
 $a = 4$   
 $b = 1$   
 $b = b + 8$   
 $a = a + b$ 
```



RAM-model-kode:

```
 $H[7] = 4$   
 $H[5] = 1$   
 $H[5] = H[5] + 8$   
 $H[7] = H[7] + H[5]$ 
```

Hukommelse  $H$

	Plads 0
	Plads 1
	Plads 2
	Plads 3
	$\vdots$
$b$	<del>1</del> 9
$a$	4
	$\vdots$

# Ord/word

Hver plads kan indeholde et ord/word, fx:

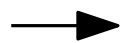
- $w$  bits, konstant  $w$ .
- Et heltal mindre end en konstant  $N$ .
- Et tegn (bogstav/symbol).

Ikke lange tekster, mange tal eller vilkårligt store tal!  
Disse må gemmes som flere ord.

Variable er læsevenlige navne for pladser i hukommelsen:

Pseudokode:

```
a = 4
b = 1
b = b + 8
a = a + b
```



RAM-model-kode:

```
H[7] = 4
H[5] = 1
H[5] = H[5] + 8
H[7] = H[7] + H[5]
```

Hukommelse  $H$

	Plads 0
	Plads 1
	Plads 2
	Plads 3
	⋮
$b$	<del>1</del> 9
$a$	<del>4</del> 13
	⋮

# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

Pseudokode:

$A[0] = 5$

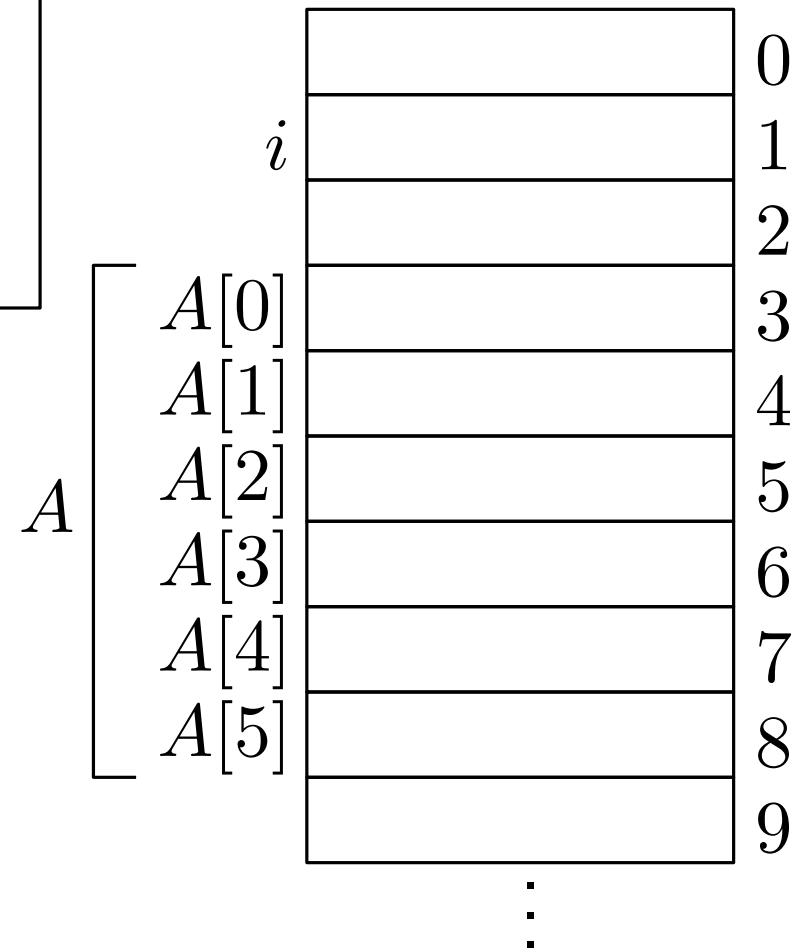
$A[4] = 6$

$i = 2$

$A[i] = 8$

RAM-model-kode:

Hukommelse  $H$



# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

Pseudokode:

$A[0] = 5$

$A[4] = 6$

$i = 2$

$A[i] = 8$

RAM-model-kode:

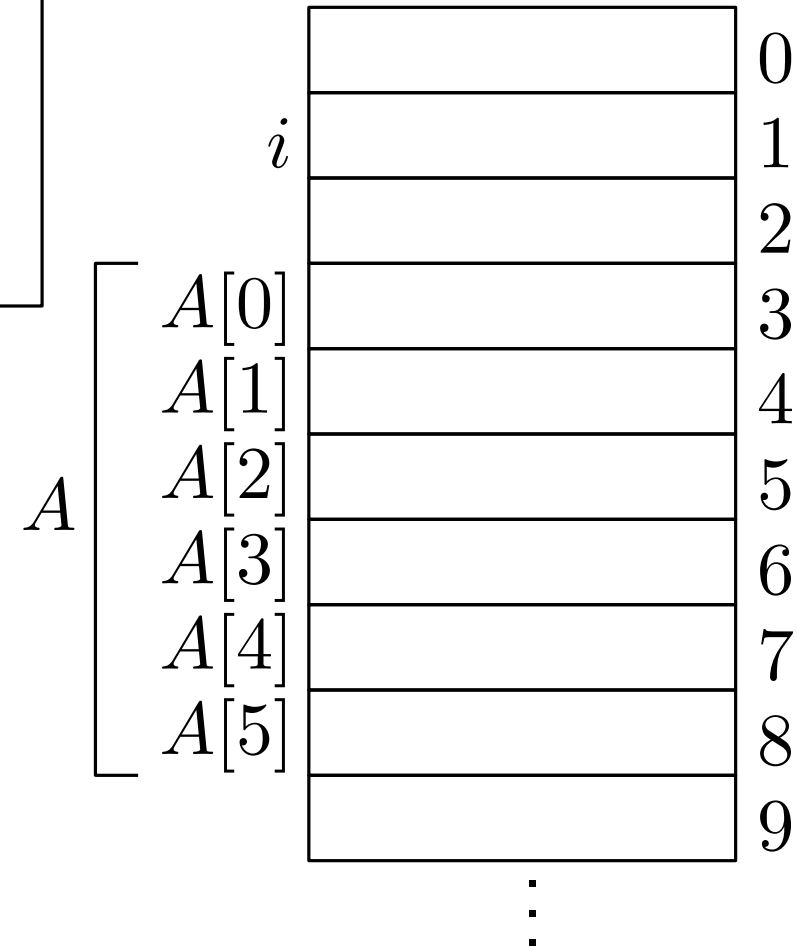
$H[3 + 0] = 5$

$H[3 + 4] = 6$

$H[1] = 2$

$H[3 + H[1]] = 8$

Hukommelse  $H$



# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

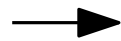
Pseudokode:

$A[0] = 5$

$A[4] = 6$

$i = 2$

$A[i] = 8$



RAM-model-kode:

$H[3 + 0] = 5$

$H[3 + 4] = 6$

$H[1] = 2$

$H[3 + H[1]] = 8$

Hukommelse  $H$

A	$A[0]$		0
	$A[1]$		1
	$A[2]$		2
	$A[3]$	5	3
	$A[4]$		4
	$A[5]$		5
			6
			7
			8
			9
			⋮



# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

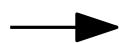
Pseudokode:

$A[0] = 5$

$A[4] = 6$

$i = 2$

$A[i] = 8$



RAM-model-kode:

$H[3 + 0] = 5$

$H[3 + 4] = 6$

$H[1] = 2$

$H[3 + H[1]] = 8$

Hukommelse  $H$

$A$	$A[0]$		0
	$A[1]$		1
	$A[2]$		2
	$A[3]$	5	3
	$A[4]$		4
	$A[5]$		5
			6
		6	7
			8
			9
			$\vdots$

# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

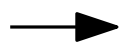
Pseudokode:

$A[0] = 5$

$A[4] = 6$

$i = 2$

$A[i] = 8$



RAM-model-kode:

$H[3 + 0] = 5$

$H[3 + 4] = 6$

$H[1] = 2$

$H[3 + H[1]] = 8$

Hukommelse  $H$

A	[	$A[0]$		0
		$A[1]$		1
		$A[2]$		2
		$A[3]$		3
		$A[4]$		4
		$A[5]$		5
				6
				7
				8
				9
				⋮

$i$

# Arrays/tabeller

Array: Sammenhængende del af hukommelsen.

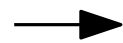
Pseudokode:

$A[0] = 5$

$A[4] = 6$

$i = 2$

$A[i] = 8$



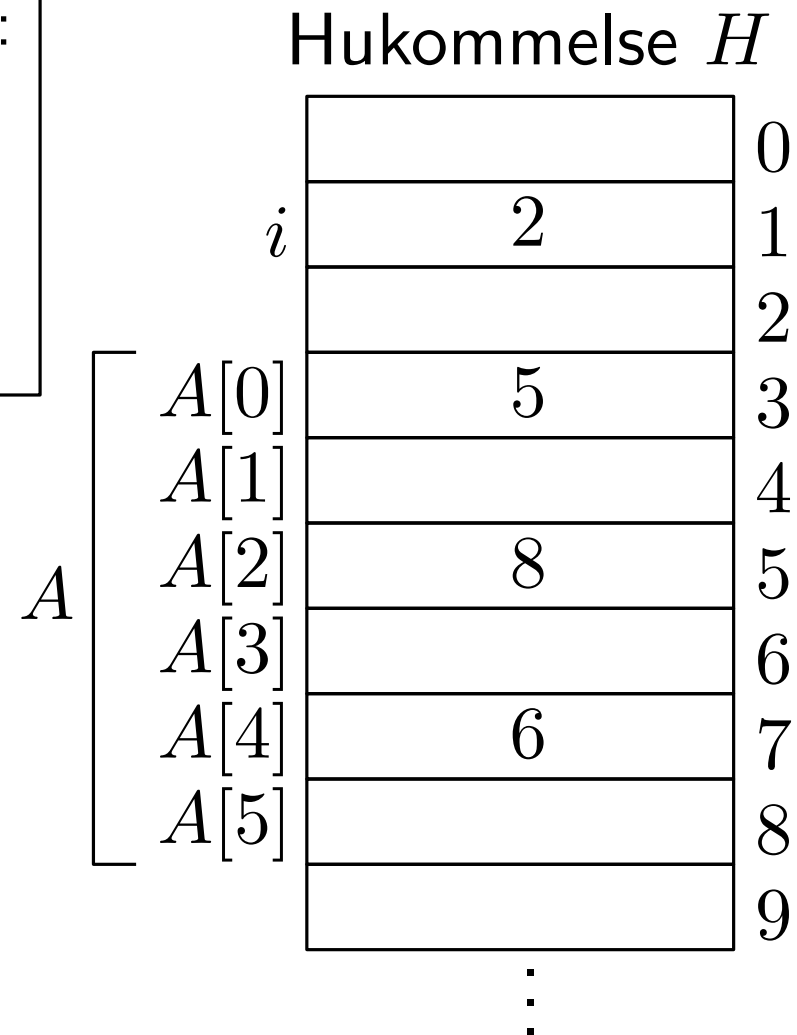
RAM-model-kode:

$H[3 + 0] = 5$

$H[3 + 4] = 6$

$H[1] = 2$

$H[3 + H[1]] = 8$



# Eksempel på algoritme: Maksimum i array

**Input:** Array  $A$  af størrelse  $n$ , dvs.  $A[0 \dots n - 1]$ .

**Output:** Et index  $i$ ,  $0 \leq i < n$ , så  $A[i] = \max\{A[0], \dots, A[n - 1]\}$ .

# Eksempel på algoritme: Maksimum i array

**Input:** Array  $A$  af størrelse  $n$ , dvs.  $A[0 \dots n - 1]$ .

**Output:** Et index  $i$ ,  $0 \leq i < n$ , så  $A[i] = \max\{A[0], \dots, A[n - 1]\}$ .

## Algoritme:

Gennemløb  $A$  og vedligehold index af hidtil største indgang. Returnér index.

# Eksempel på algoritme: Maksimum i array

**Input:** Array  $A$  af størrelse  $n$ , dvs.  $A[0 \dots n - 1]$ .

**Output:** Et index  $i$ ,  $0 \leq i < n$ , så  $A[i] = \max\{A[0], \dots, A[n - 1]\}$ .

## Algoritme:

Gennemløb  $A$  og vedligehold index af hidtil største indgang. Returnér index.

```
public static int findMax(int[] A, int n) {  
    int max = 0;  
    for (int i = 0; i < n; i++)  
        if (A[i] > A[max])  
            max = i;  
    return max;  
}
```

# Eksempel på algoritme: Maksimum i array

**Input:** Array  $A$  af størrelse  $n$ , dvs.  $A[0 \dots n - 1]$ .

**Output:** Et index  $i$ ,  $0 \leq i < n$ , så  $A[i] = \max\{A[0], \dots, A[n - 1]\}$ .

## Algoritme:

Gennemløb  $A$  og vedligehold index af hidtil største indgang. Returnér index.

```
public static int findMax(int[] A, int n) {  
    int max = 0;  
    for (int i = 0; i < n; i++)  
        if (A[i] > A[max])  
            max = i;  
    return max;  
}
```

```
findMax( $A$ ,  $n$ )  
     $max = 0$   
    for  $i = 0$  to  $n - 1$   
        if  $A[i] > A[max]$   
             $max = i$   
    return  $max$ 
```

# Eksempel på algoritme: Maksimum i array

**Input:** Array  $A$  af størrelse  $n$ , dvs.  $A[0 \dots n - 1]$ .

**Output:** Et index  $i$ ,  $0 \leq i < n$ , så  $A[i] = \max\{A[0], \dots, A[n - 1]\}$ .

## Algoritme:

Gennemløb  $A$  og vedligehold index af hidtil største indgang. Returnér index.

```
public static int findMax(int[] A, int n) {  
    int max = 0;  
    for (int i = 0; i < n; i++)  
        if (A[i] > A[max])  
            max = i;  
    return max;  
}
```

```
findMax( $A$ ,  $n$ )  
     $max = 0$   
    for  $i = 0$  to  $n - 1$   
        if  $A[i] > A[max]$   
             $max = i$   
    return  $max$ 
```

Ligesom rigtig kode, men uden besværet.

Skrevet til at blive læst af *mennesker*, ikke en computer.

Meget mere præcis end rent sproglig beskrivelse.



# Køretid/tidskompleksitet

$T(n)$  = # skridt/instruktioner/operationer som algoritmen laver på et input af størrelse  $n$

# Køretid/tidskompleksitet

$T(n)$  = # skridt/instruktioner/operationer som algoritmen laver på et input af størrelse  $n$

Skridt:

- Læse fra/skrive til hukommelse ( $x = y$ ,  $A[i]$ ,  $i = i + 1, \dots$ )
- Aritmetiske/logiske operationer ( $+$ ,  $-$ ,  $\cdot$ ,  $/$ ,  $\%$ , and, or)
- Sammenligninger ( $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ,  $==$ ,  $\neq$ )
- Programflow (if-else, while, for, funktionskald)

# Køretid/tidskompleksitet

$T(n)$  = # skridt/instruktioner/operationer som algoritmen laver på et input af størrelse  $n$

Skridt:

- Læse fra/skrive til hukommelse ( $x = y$ ,  $A[i]$ ,  $i = i + 1, \dots$ )
- Aritmetiske/logiske operationer ( $+$ ,  $-$ ,  $\cdot$ ,  $/$ ,  $\%$ , and, or)
- Sammenligninger ( $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ,  $==$ ,  $\neq$ )
- Programflow (if-else, while, for, funktionskald)

Tidskompleksitet i værste fald (worst-case complexity):

Maximal køretid for alle input af størrelse  $n$ .

# Eksempel

```
findMax( $A$ ,  $n$ )  
   $max = 0$   
  for  $i = 0$  to  $n - 1$   
    if  $A[i] > A[max]$   
       $max = i$   
  return  $max$ 
```

# Eksempel

```
findMax( $A$ ,  $n$ )
```

```
   $max = 0$ 
```

```
  for  $i = 0$  to  $n - 1$ 
```

```
    if  $A[i] > A[max]$ 
```

```
       $max = i$ 
```

```
  return  $max$ 
```

skridt i linje

1

7

6

2

2

# Eksempel

```
findMax( $A$ ,  $n$ )  
   $max = 0$   
  for  $i = 0$  to  $n - 1$   
    if  $A[i] > A[max]$   
       $max = i$   
  return  $max$ 
```

skridt i linje

1

7

6

2

2

skridt hver gang:

for

læse  $i$

udregnet  $i + 1$

gemme  $i + 1$  som  $i$

læse  $n$

udregne  $n - 1$

sammenligne ny  $i$  med  $n - 1$

# Eksempel

```
findMax( $A$ ,  $n$ )  
   $max = 0$   
  for  $i = 0$  to  $n - 1$   
    if  $A[i] > A[max]$   
       $max = i$   
  return  $max$ 
```

skridt i linje

1

7

6

2

2

max. udførsler af linje

1

$n + 1$

$n$

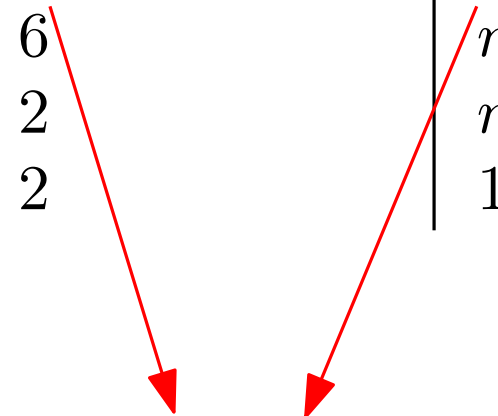
$n$

1

# Eksempel

```
findMax( $A$ ,  $n$ )  
   $max = 0$   
  for  $i = 0$  to  $n - 1$   
    if  $A[i] > A[max]$   
       $max = i$   
  return  $max$ 
```

skridt i linje	max. udførsler af linje
1	1
<span style="border: 1px solid red;">7</span>	<span style="border: 1px solid red;"><math>n + 1</math></span>
6	$n$
2	$n$
2	1



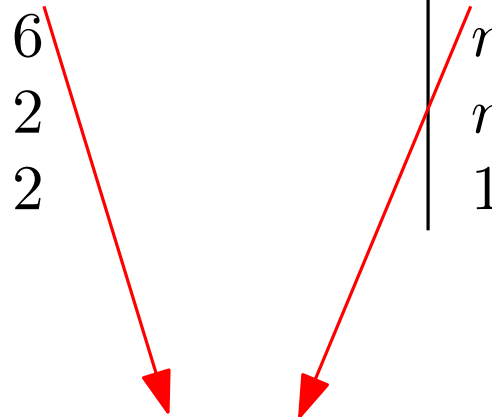
$$T(n) = \text{køretid} = \text{skridt i alt} = 1 + 7(n + 1) + 6n + n + 2 = 14n + 10 = \Theta(n)$$



# Eksempel

```
findMax( $A$ ,  $n$ )  
   $max = 0$   
  for  $i = 0$  to  $n - 1$   
    if  $A[i] > A[max]$   
       $max = i$   
  return  $max$ 
```

skridt i linje	max. udførsler af linje
1	1
<span style="border: 1px solid red;">7</span>	<span style="border: 1px solid red;"><math>n + 1</math></span>
6	$n$
2	$n$
2	1



$$T(n) = \text{køretid} = \text{skridt i alt} = 1 + 7(n + 1) + 6n + n + 2 = 14n + 10 = \textstyle\boxed{\Theta(n)}$$

Asymptotisk notation: Køretiden er lineær. Langsomt voksende led og konstanter ignoreres.

# Eksempel

```
doubleLoop( $n$ )  
   $x = 0$   
  for  $i = 0$  to  $n - 1$   
     $x = x + 1$   
    for  $j = 0$  to  $n - 1$   
       $x = x + 1$   
  return  $x$ 
```

## Eksempel

```
doubleLoop( $n$ )
```

```
   $x = 0$ 
```

```
  for  $i = 0$  to  $n - 1$ 
```

```
     $x = x + 1$ 
```

```
    for  $j = 0$  to  $n - 1$ 
```

```
       $x = x + 1$ 
```

```
  return  $x$ 
```

# skridt i linje

1

7

3

7

3

2

## Eksempel

```
doubleLoop( $n$ )  
   $x = 0$   
  for  $i = 0$  to  $n - 1$   
     $x = x + 1$   
    for  $j = 0$  to  $n - 1$   
       $x = x + 1$   
  return  $x$ 
```

# skridt i linje	max. # udførsler af linje
1	1
7	$n + 1$
3	$n$
7	$n(n + 1)$
3	$n^2$
2	1

## Eksempel

```
doubleLoop(n)  
  x = 0  
  for i = 0 to n − 1  
    x = x + 1  
    for j = 0 to n − 1  
      x = x + 1  
  return x
```

# skridt i linje	max. # udførsler af linje
1	1
7	$n + 1$
3	$n$
7	$n(n + 1)$
3	$n^2$
2	1

$$T(n) = 1 + 7(n + 1) + 3n + 7n(n + 1) + 3n^2 + 2 = 10n^2 + 17n + 10 = \Theta(n^2)$$

## Eksempel

```
doubleLoop(n)  
  x = 0  
  for i = 0 to n - 1  
    x = x + 1  
    for j = 0 to n - 1  
      x = x + 1  
  return x
```

# skridt i linje	max. # udførsler af linje
1	1
7	$n + 1$
3	$n$
7	$n(n + 1)$
3	$n^2$
2	1

$$T(n) = 1 + 7(n + 1) + 3n + 7n(n + 1) + 3n^2 + 2 = 10n^2 + 17n + 10 = \Theta(n^2)$$

Asymptotisk notation: Køretiden er kvadratisk. Langsomt voksende led og konstanter ignoreres.

## Eksempel

	# skridt i linje	max. # udførsler af linje
<code>doubleLoop(<math>n</math>)</code>		
$x = 0$	1	1
for $i = 0$ to $n - 1$	7	$n + 1$
$x = x + 1$	3	$n$
for $j = 0$ to $n - 1$	7	$n(n + 1)$
$x = x + 1$	3	$n^2$
return $x$	2	1

$$T(n) = 1 + 7(n + 1) + 3n + 7n(n + 1) + 3n^2 + 2 = 10n^2 + 17n + 10 = \Theta(n^2)$$

Asymptotisk notation: Køretiden er kvadratisk. Langsomt voksende led og konstanter ignoreres.

	# skridt i linje
<code>twoLoops(<math>n</math>)</code>	
$x = 0$	1
for $i = 0$ to $n - 1$	7
$x = x + 1$	3
for $j = 0$ to $n - 1$	7
$x = x + 1$	3
return $x$	2

## Eksempel

	# skridt i linje	max. # udførsler af linje
<code>doubleLoop(<math>n</math>)</code>		
$x = 0$	1	1
for $i = 0$ to $n - 1$	7	$n + 1$
$x = x + 1$	3	$n$
for $j = 0$ to $n - 1$	7	$n(n + 1)$
$x = x + 1$	3	$n^2$
return $x$	2	1

$$T(n) = 1 + 7(n + 1) + 3n + 7n(n + 1) + 3n^2 + 2 = 10n^2 + 17n + 10 = \Theta(n^2)$$

Asymptotisk notation: Køretiden er kvadratisk. Langsomt voksende led og konstanter ignoreres.

	# skridt i linje	max. # udførsler af linje
<code>twoLoops(<math>n</math>)</code>		
$x = 0$	1	1
for $i = 0$ to $n - 1$	7	$n + 1$
$x = x + 1$	3	$n$
for $j = 0$ to $n - 1$	7	$n + 1$
$x = x + 1$	3	$n$
return $x$	2	1

$$T(n) = 1 + 7(n + 1) + 3n + 7(n + 1) + 3n + 2 = 20n + 20 = \Theta(n)$$



# Quiz

```
Q( $A, n$ ) \\
  \\antag  $\log_2 n$  er et helta
   $s = 0$ 
  for  $i$  from 1 to  $n$ 
    for  $j$  from 1 to  $\log_2 n$ 
       $s = s + A[i] + A[j]$ 
  return  $s$ 
```

# Quiz

```
Q(A, n) \\ antag  $\log_2 n$  er et heltal  
  s = 0  
  for i from 1 to n  
    for j from 1 to  $\log_2 n$   
      s = s + A[i] + A[j]  
  return s
```

Hvad er køretiden af  $Q(A, n)$ ?

socrative.com → Student login.

Room name: ABRAHAMSEN3464

- (a)  $\Theta(n)$
- (b)  $\Theta(n^2)$
- (c)  $\Theta(n + \log_2 n)$
- (d)  $\Theta(n \cdot \log_2 n)$
- (e)  $\Theta(n\sqrt{n})$

# Hvorfor er asymptotisk køretid så praktisk?

Antag  $S1(A, n)$  og  $S2(A, n)$  begge sorterer array  $A$  af længde  $n$ .

$S1$  har køretid  $T_1(n) = n^2$  og  $S2$  har køretid  $T_2(n) = 100 \cdot n \log_2 n$ .

$T_1(n) = \Theta(n^2)$  og  $T_2(n) = \Theta(n \log n)$ .

# Hvorfor er asymptotisk køretid så praktisk?

Antag  $S1(A, n)$  og  $S2(A, n)$  begge sorterer array  $A$  af længde  $n$ .

$S1$  har køretid  $T_1(n) = n^2$  og  $S2$  har køretid  $T_2(n) = 100 \cdot n \log_2 n$ .

$T_1(n) = \Theta(n^2)$  og  $T_2(n) = \Theta(n \log n)$ .

$n = 100$ :  $T_1(100) = 10000$  og  $T_2(100) \approx 66400$ , så

$$\frac{T_1(100)}{T_2(100)} \approx 0.15.$$

# Hvorfor er asymptotisk køretid så praktisk?

Antag  $S1(A, n)$  og  $S2(A, n)$  begge sorterer array  $A$  af længde  $n$ .

$S1$  har køretid  $T_1(n) = n^2$  og  $S2$  har køretid  $T_2(n) = 100 \cdot n \log_2 n$ .

$T_1(n) = \Theta(n^2)$  og  $T_2(n) = \Theta(n \log n)$ .

$n = 100$ :  $T_1(100) = 10000$  og  $T_2(100) \approx 66400$ , så

$$\frac{T_1(100)}{T_2(100)} \approx 0.15.$$

$n = 5.8 \cdot 10^6$ :  $T_1(5.8 \cdot 10^6) \approx 3.4 \cdot 10^{13}$  og  $T_2(5.8 \cdot 10^6) \approx 1.3 \cdot 10^{10}$ , så

$$\frac{T_1(5.8 \cdot 10^6)}{T_2(5.8 \cdot 10^6)} \approx 2600.$$

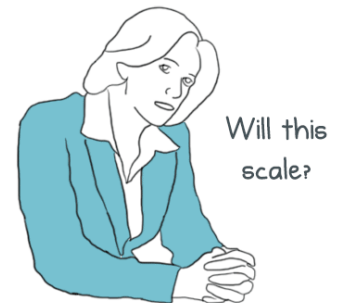
# Hvorfor er vi ligeglade med konstanter?

Hvis køretiden er  $T(n) = 100n \log n$  skriver vi  $T(n) = \Theta(n \log n)$ .

Vi ignorerer konstanten 100 fordi:

- Konstanten afhænger af præcis hvordan vi tæller skridt.
- I praksis er det forskelligt hvor lang tid de basale skridt tager.
- Når  $n$  bliver stor er det vigtigste den asymptotiske opførsel.
- “Will this scale?”
- Derfor ignorerer vi også langsomt voksende led.

ASK “WILL  
THIS SCALE?”  
NO MATTER  
WHAT IT IS



# Hvorfor er vi ligeglade med konstanter?

Hvis køretiden er  $T(n) = 100n \log n$  skriver vi  $T(n) = \Theta(n \log n)$ .

Vi ignorerer konstanten 100 fordi:

- Konstanten afhænger af præcis hvordan vi tæller skridt.
- I praksis er det forskelligt hvor lang tid de basale skridt tager.
- Når  $n$  bliver stor er det vigtigste den asymptotiske opførsel.
- “Will this scale?”
- Derfor ignorerer vi også langsomt voksende led.

ASK “WILL  
THIS SCALE?”  
NO MATTER  
WHAT IT IS



P.S:

- I praksis kan vi ikke ignorere astronomiske konstanter.
- En asymptotisk langsommere algoritme kan foretrækkes hvis
  - $n$  aldrig bliver meget stor, eller
  - den langsommere algoritme er meget simplere og hurtig nok.

# $\Theta$ , $\Omega$ og $O$

Vi skriver:

- $T(n) = \Omega(n^2)$  hvis

$$c_1 \cdot n^2 \leq T(n)$$

for en konstant  $c_1 > 0$ .

- $T(n) = O(n^2)$  hvis

$$T(n) \leq c_2 \cdot n^2$$

for en konstant  $c_2 > 0$ .

- $T(n) = \Theta(n^2)$  hvis

$$c_1 \cdot n^2 \leq T(n) \leq c_2 \cdot n^2$$

for konstanter  $c_1, c_2 > 0$ , dvs.  $T(n) = \Omega(n^2)$  og  $T(n) = O(n^2)$ .

Disse definitioner er ikke helt præcise. Vi vender tilbage til dem senere.