

## Algoritmer for Max Sum Problemet

# Maximum Sum problemet

Givet et array (liste) af tal kan vi se på summer af segmenter (del-arrays).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	-1	2	-4	5	3	-1	2	-6	0	8	12	-4	6	8	4



I segmentet ovenfor er summen

$$(-4) + 5 + 3 + (-1) + 2 + (-6) + 0 + 8 = 7$$

# Maximum Sum problemet

Givet et array (liste) af tal kan vi se på summer af segmenter (del-arrays).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	-1	2	-4	5	3	-1	2	-6	0	8	12	-4	6	8	4



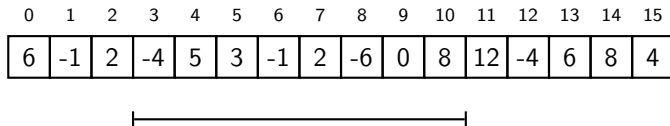
I segmentet ovenfor er summen

$$(-4) + 5 + 3 + (-1) + 2 + (-6) + 0 + 8 = 7$$

Spørgsmål: Hvilket segment har størst sum?

# Maximum Sum problemet

Givet et array (liste) af tal kan vi se på summer af segmenter (del-arrays).



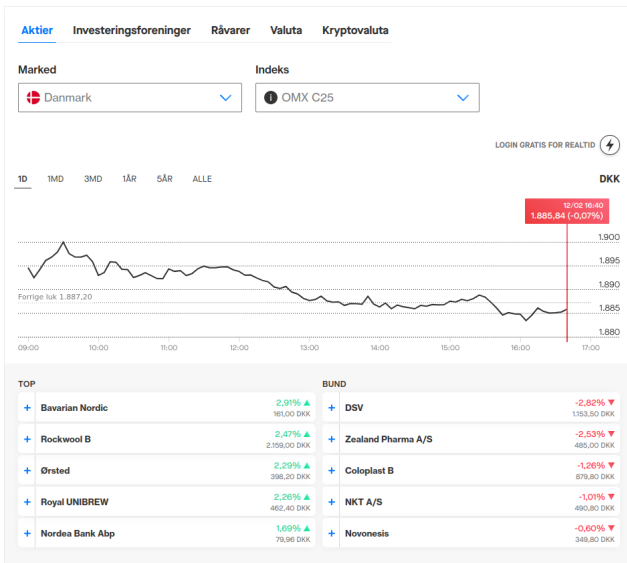
I segmentet ovenfor er summen

$$(-4) + 5 + 3 + (-1) + 2 + (-6) + 0 + 8 = 7$$

Spørgsmål: Hvilket segment har størst sum?

Et simpelt og fundamentalt problem

# Mere motivation for MaxSum problemet: aktieanalyse



(Fra [www.euroinvestor.dk](http://www.euroinvestor.dk))

# Aktieanalyse

Vi har data af følgende type:

Aktie for Firma X:

2023.02.20	2023.02.21	2023.02.22	2023.02.23	2023.02.24	2023.02.25	2023.02.26
+2%	-3%	+8%	-1%	-3%	+3%	+11%



Spørgsmål: I hvilken periode har det været bedst at eje aktien?

# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til

# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.



# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.

Hvis 1000 kr. falder med 2% bliver det til

# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.

Hvis 1000 kr. falder med 2% bliver det til  $1000 \cdot 0.98 = 980$  kr.

# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.

Hvis 1000 kr. falder med 2% bliver det til  $1000 \cdot 0.98 = 980$  kr.

Hvis 1000 kr. først stiger med 3% og derefter falder med 2% bliver det til

# Procentregning

Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.

Hvis 1000 kr. falder med 2% bliver det til  $1000 \cdot 0.98 = 980$  kr.

Hvis 1000 kr. først stiger med 3% og derefter falder med 2% bliver det til

$$1000 \cdot 1.03 \cdot 0.98 (= 1009.40) \text{ kr.}$$

# Procentregning


Hvis 1000 kr. stiger med 3% bliver det til  $1000 \cdot 1.03 = 1030$  kr.

Hvis 1000 kr. falder med 2% bliver det til  $1000 \cdot 0.98 = 980$  kr.

Hvis 1000 kr. først stiger med 3% og derefter falder med 2% bliver det til

$$1000 \cdot 1.03 \cdot 0.98 (= 1009.40) \text{ kr.}$$

2023.02.20	2023.02.21	2023.02.22	2023.02.23	2023.02.24	2023.02.25	2023.02.26
+2%	-3%	+8%	-1%	-3%	+3%	+11%



I perioden ovenfor har aktien forandret sig med en faktor

$$0.97 \cdot 1.08 \cdot 0.99 \cdot 0.97 \cdot 1.03$$

# Aktieanalyse

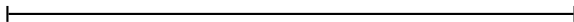
2023.02.20	2023.02.21	2023.02.22	2023.02.23	2023.02.24	2023.02.25	2023.02.26
+2%	-3%	+8%	-1%	-3%	+3%	+11%



Spørgsmål: I hvilken periode har det været bedst at eje aktien?



1.02	0.97	1.08	0.99	0.97	1.03	1.11
------	------	------	------	------	------	------



Spørgsmål: Hvilket segment har **størst produkt**?

# Fra maximum produkt til maximum sum

Logaritmer er voksende funktioner. Så

$$0.94 \cdot 1.05 \cdot 0.99 \leq 0.96 \cdot 1.03 \cdot 1.01$$

hvis og kun hvis

$$\log(0.94 \cdot 1.05 \cdot 0.99) \leq \log(0.96 \cdot 1.03 \cdot 1.01)$$

## Fra maximum produkt til maximum sum

Logaritmer er voksende funktioner. Så

$$0.94 \cdot 1.05 \cdot 0.99 \leq 0.96 \cdot 1.03 \cdot 1.01$$

hvis og kun hvis

$$\log(0.94 \cdot 1.05 \cdot 0.99) \leq \log(0.96 \cdot 1.03 \cdot 1.01)$$

Da  $\log(x \cdot y) = \log(x) + \log(y)$ , gælder ovenstående hvis og kun hvis

$$\log(0.94) + \log(1.05) + \log(0.99) \leq \log(0.96) + \log(1.03) + \log(1.01)$$



## Fra maximum produkt til maximum sum

Så segmentet, der har **størst produkt** i dette array:

1.02	0.97	1.08	0.99	0.97	1.03	1.11
------	------	------	------	------	------	------

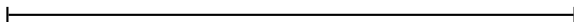


er det samme som segmentet der har **størst sum** i dette array:

$\log 1.02$	$\log 0.97$	$\log 1.08$	$\log 0.99$	$\log 0.97$	$\log 1.03$	$\log 1.11$
-------------	-------------	-------------	-------------	-------------	-------------	-------------

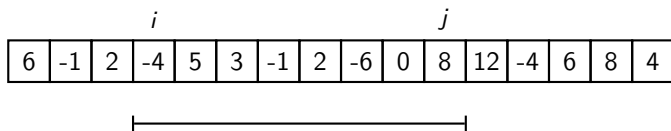


0.0286	-0.0439	0.1110	-0.0145	-0.0439	0.0426	0.1506
--------	---------	--------	---------	---------	--------	--------



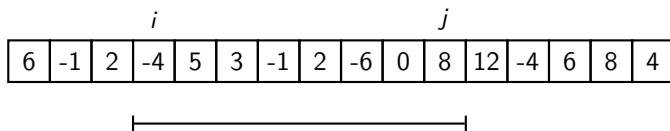
# Algoritmer for MaxSum

Vi skal finde summen for alle segmenter:



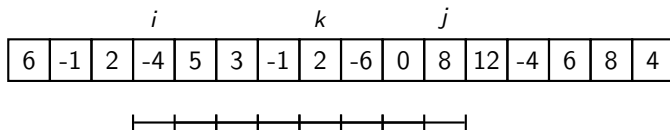
# Algoritmer for MaxSum

Vi skal finde summen for alle segmenter:



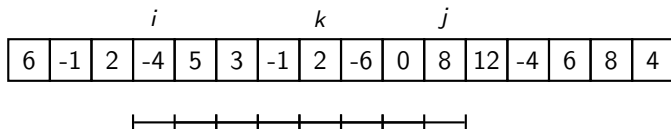
Naturlig algoritme ud fra definitionen:

For alle  $i$  og for alle  $j \geq i$ , lav summen fra  $i$  til og med  $j$ .



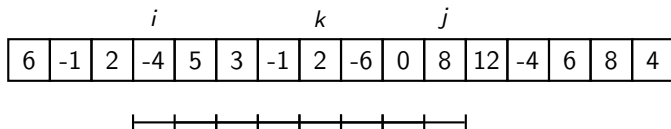
# Første algoritme for MaxSum

```
MAXSUM1( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    for  $j = i$  to  $n - 1$   
      sum = 0  
      for  $k = i$  to  $j$   
        sum +=  $A[k]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```



## Første algoritme for MaxSum

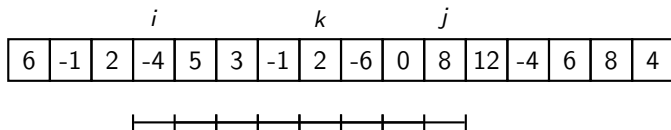
```
MAXSUM1( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    for  $j = i$  to  $n - 1$   
      sum = 0  
      for  $k = i$  to  $j$   
        sum +=  $A[k]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```



Korrekt? Følger af definition af problem.

# Første algoritme for MaxSum

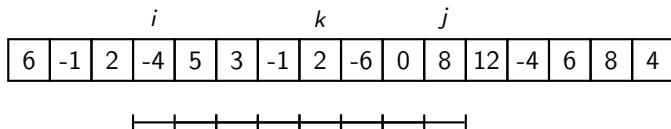
```
MAXSUM1( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    for  $j = i$  to  $n - 1$   
      sum = 0  
      for  $k = i$  to  $j$   
        sum +=  $A[k]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```



Korrekt? Følger af definition af problem. Køretid?

# Første algoritme for MaxSum

```
MAXSUM1( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    for  $j = i$  to  $n - 1$   
      sum = 0  
      for  $k = i$  to  $j$   
        sum +=  $A[k]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```



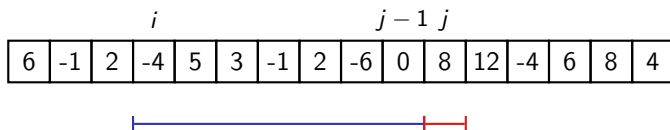
Korrekt? Følger af definition af problem. Køretid?  $\Theta(n^3)$ , med samme argument som for Algoritme 3 fra asymptotisk analyse eksemplerne (de har helt samme struktur).

# Observation

$$(-4) + 5 + 3 + (-1) + 2 + (-6) + 0 = (-1)$$

$\Downarrow$

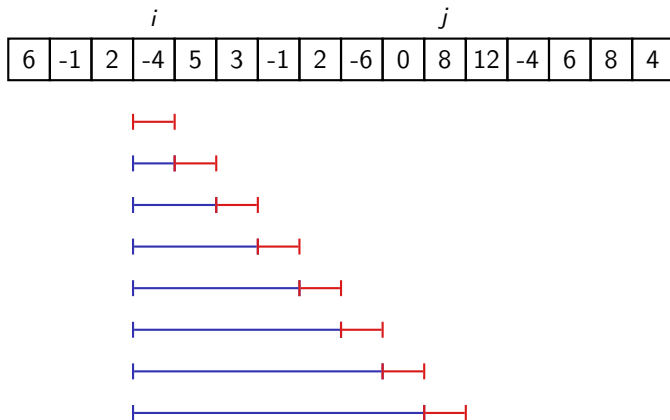
$$(-4) + 5 + 3 + (-1) + 2 + (-6) + 0 + 8 = (-1) + 8 = 7$$





## Idé til forbedret algoritme

Algoritme: For hvert  $i$ , beregn summer for stigende  $j$  med én ny addition per sum.



## Anden algoritme for MaxSum

```
MAXSUM2( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    sum = 0  
    for  $j = i$  to  $n - 1$   
      sum +=  $A[j]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```

## Anden algoritme for MaxSum

```
MAXSUM2( $n$ )  
  maxSoFar = 0  
  for  $i = 0$  to  $n - 1$   
    sum = 0  
    for  $j = i$  to  $n - 1$   
      sum +=  $A[j]$   
      maxSoFar = max(maxSoFar, sum);  
  return maxSoFar
```

Korrekt? Følger af definition af problem samt observationen ovenfor.

## Anden algoritme for MaxSum

```
MAXSUM2( $n$ )  
    maxSoFar = 0  
    for  $i = 0$  to  $n - 1$   
        sum = 0  
        for  $j = i$  to  $n - 1$   
            sum +=  $A[j]$   
            maxSoFar = max(maxSoFar, sum);  
    return maxSoFar
```

Korrekt? Følger af definition af problem samt observationen ovenfor.

Køretid?

## Anden algoritme for MaxSum

```
MAXSUM2( $n$ )  
    maxSoFar = 0  
    for  $i = 0$  to  $n - 1$   
        sum = 0  
        for  $j = i$  to  $n - 1$   
            sum +=  $A[j]$   
            maxSoFar = max(maxSoFar, sum);  
    return maxSoFar
```

Korrekt? Følger af definition af problem samt observationen ovenfor.

Køretid?  $\Theta(n^2)$ , med ca. samme argument som for Algoritme 2 fra asymptotisk analyse eksemplerne.

## Ny observation

$$x_1 \leq x_2$$

$$\Leftrightarrow$$

$$x_1 + 2 \leq x_2 + 2$$

## Ny observation

$$\begin{aligned}x_1 &\leq x_2 \\ \Updownarrow \\ x_1 + 2 &\leq x_2 + 2\end{aligned}$$

Heraf følger:

$$\max\{x_1 + 2, x_2 + 2, \dots, x_i + 2\} = \max\{x_1, x_2, \dots, x_i\} + 2$$

## Ny observation

$$\begin{aligned}x_1 &\leq x_2 \\ \Updownarrow \\ x_1 + 2 &\leq x_2 + 2\end{aligned}$$

Heraf følger:

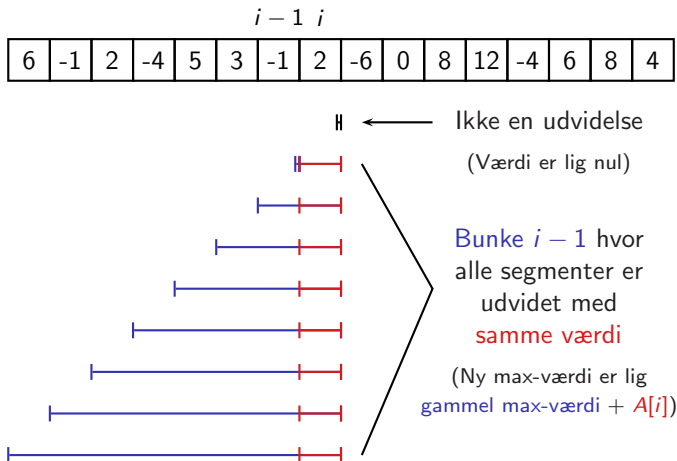
$$\max\{x_1 + 2, x_2 + 2, \dots, x_i + 2\} = \max\{x_1, x_2, \dots, x_i\} + 2$$

Idé: Kan vi kigge på segmenter i bunker, således at den nye bunke er lig den gamle bunke med alle segmenter udvidet med den samme værdi?



# Idé til forbedret algoritme

Lad bunke  $i$  være alle segmenter, som ender ved  $A[i]$  (s højre kant). Så er bunke  $i$  det samme som bunke  $i - 1$  med alle segmenter udvidet med den samme værdi, plus det tomme segment:



## Tredie algoritme for MaxSum

```
MAXSUM3( $n$ )  
  maxSoFar = 0  
  maxEndingHere = 0  
  for  $i = 0$  to  $n - 1$   
    maxEndingHere = max(maxEndingHere +  $A[i]$ , 0)  
    maxSoFar = max(maxSoFar, maxEndingHere);  
  return maxSoFar
```

## Tredie algoritme for MaxSum

```
MAXSUM3( $n$ )  
    maxSoFar = 0  
    maxEndingHere = 0  
    for  $i = 0$  to  $n - 1$   
        maxEndingHere = max(maxEndingHere +  $A[i]$ , 0)  
        maxSoFar = max(maxSoFar, maxEndingHere);  
    return maxSoFar
```

**Korrekt?** Følger af definition af problem samt den nye observation ovenfor, som sikrer, at vi tager maksimum over alle segmenter (da ethvert segment er med i en bunke, nemlig bunken for det  $i$ , hvor segmentet ender).

## Tredie algoritme for MaxSum

```
MAXSUM3( $n$ )  
    maxSoFar = 0  
    maxEndingHere = 0  
    for  $i = 0$  to  $n - 1$   
        maxEndingHere = max(maxEndingHere +  $A[i]$ , 0)  
        maxSoFar = max(maxSoFar, maxEndingHere);  
    return maxSoFar
```

**Korrekt?** Følger af definition af problem samt den nye observation ovenfor, som sikrer, at vi tager maksimum over alle segmenter (da ethvert segment er med i en bunke, nemlig bunken for det  $i$ , hvor segmentet ender).

**Køretid?**

## Tredie algoritme for MaxSum

```
MAXSUM3( $n$ )  
    maxSoFar = 0  
    maxEndingHere = 0  
    for  $i = 0$  to  $n - 1$   
        maxEndingHere = max(maxEndingHere +  $A[i]$ , 0)  
        maxSoFar = max(maxSoFar, maxEndingHere);  
    return maxSoFar
```

**Korrekt?** Følger af definition af problem samt den nye observation ovenfor, som sikrer, at vi tager maksimum over alle segmenter (da ethvert segment er med i en bunke, nemlig bunken for det  $i$ , hvor segmentet ender).

**Køretid?** Der er  $n$  iterationer, som hver tager  $\Theta(1)$  tid. Det giver alt i alt  $\Theta(n)$ .