

Algoritmer og Datastrukturer 1

Gerth Stølting Brodal



Kursusbeskrivelsen...

Kursusbeskrivelsen:

Algoritmer og datastrukturer 1

Formål

Deltagerne vil efter kurset have indsigt i **algoritmer** som model for **sekventielle beregningsprocesser** og som basis for formelle **korrekthedsbeviser** og analyse af **ressourceforbrug** ved beregningerne, samt detaljeret kendskab til adskillige konkrete implementationer af fundamentale datastrukturer.

Indhold

Datastrukturer: Lister, træer, hashtabeller; *Dataabstraktioner*: Stakke, køer, prioritetskøer, ordbøger, mængder; *Algoritmer*: Søgning, sortering, selektion, fletning; *Analyse og syntese*: Worst-case, amortiseret og forventet udførelsestid, udsagn, invarianter, gyldighed, terminering og korrekthed.

Læringsmål

Deltagerne skal ved afslutningen af kurset kunne:

- **formulere** og **udføre** algoritmer og datastrukturer i pseudo code.
- **analysere** og **sammenligne** tid og pladsforbruget af algoritmer.
- **identificere** gyldige invarianter for en algoritme.
- **bevise** korrektheden af simple programmer og transitionssystemer.

Kursusbeskrivelsen:

Algoritmer og datastrukturer 1

Forudsætningskrav

dlIntProg

Undervisningsformer

Forelæsninger: 4 timer/uge

Obligatorisk program

6 opgaver

Evaluering

Sprog

Dansk

Eksamensterminer

Eksamen: 3. kvarter

Reeksamen: August

Vi kan antage at I ved hvordan man programmerer detaljerne – så dem springer vi over

Forelæsningerne gennemgår stoffet fra bogen. I øvelserne arbejder man med stoffet.

Stilles 6 opgaver – alle skal afleveres for at kunne gå til eksamen. Opgaverne afleveres i grupper på 1-3 personer.

Eksamen består af ca. 25 korte spørgsmål – se eksempler på kursushjemmesiden

DatLab

Fælles Q3-studiecafé i Ada0
hver fredag kl. 13-15

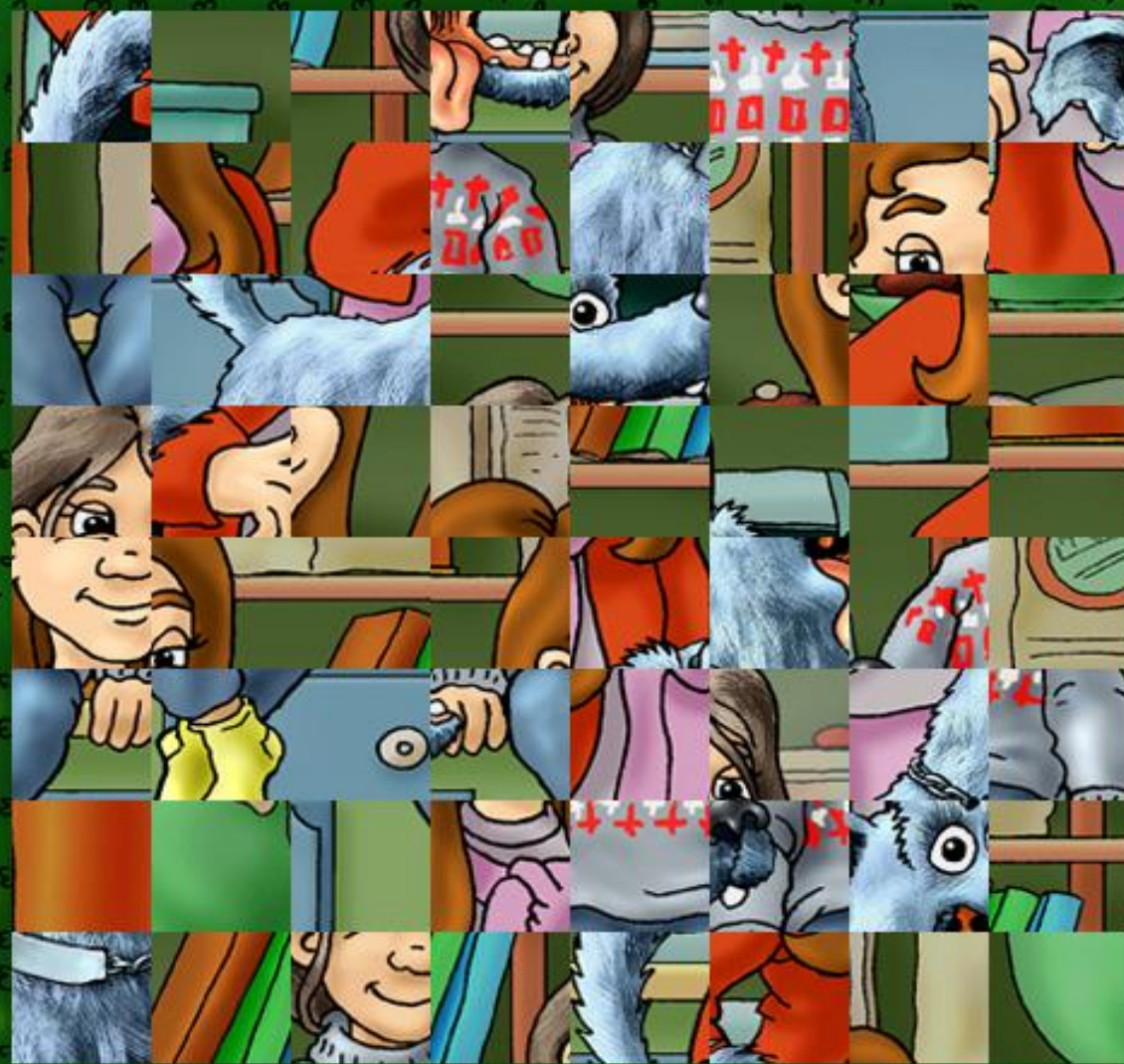
- ✓ Webteknologi
- ✓ Interaktionsdesign
- ✓ Algoritmer og datastrukturer (dat.)
- ✓ Fysisk Design (it)

Spørgsmål ?



TV2

500





Præsenteres i samarbejde med



- › Forside
- › Om Valhal
- › Konkurrencer
- › Spil & Hiscore
- › Downloads
- › På mobilen
- › TV-Guide

Hiscore er du på?

Valhal spillene findes på den cd-rom, som følger med lågekalenderen. Find din egen score herunder. Husk at vælge et specielt spille-navn, så du kan kende dig blandt alle de andre. Hi-scores bliver genstartet hver dag! Kan du blive nr. 1 på et de 24 spil?

Klik på spilnavnet for at se alle scores!

Se også

- › Hotline
- › Thors Torden Race
- › Anders And Hiscore



1. Pebernødder til Snifer

| | | |
|---|-----|-------------|
| 1 | 499 | andreas |
| 2 | 470 | Mads12345 |
| 3 | 246 | Ikke oplyst |
| 4 | 63 | DANIEL |
| 5 | 53 | mathiastp |

2. Lokes høj

| | | |
|---|-----|------------|
| 1 | 450 | Anne.K.Nie |
| 2 | 449 | Kimingen88 |
| 3 | 448 | morten.fly |
| 4 | 448 | MiaMaria |
| 5 | 448 | RONNIE |

NYT ALBUM
UDE NU

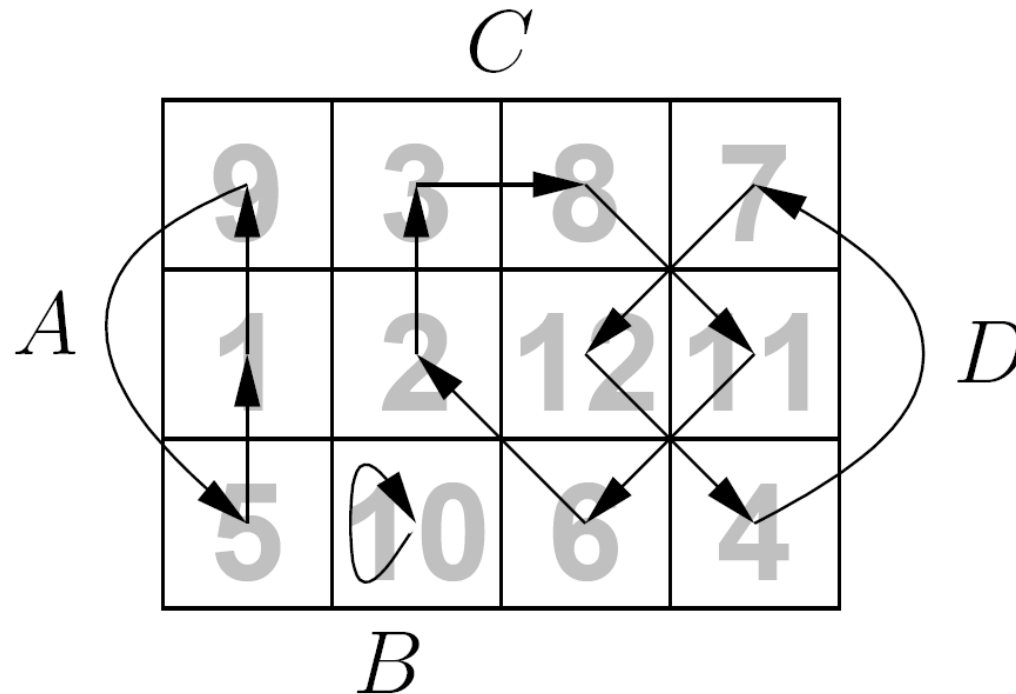
INKL.
DET DU GØR &
DRENGE SOM MIG

”Lokes Høj”

- 64 brikker
- Hiscore 450
- Antal ombytninger $500 - 450 = 50$

**Hvordan opnår man et lavt antal ombytninger
– held eller dygtighed ?**

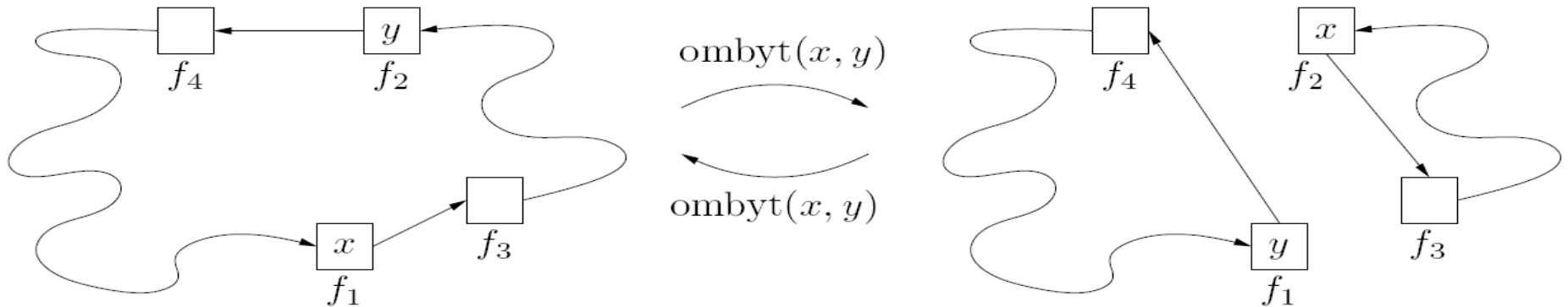
Cykler (Permutationer)



Hver pil peger på brikkens korrekte plads

Definerer en mængde af cykler (fx cyklerne A,B,C,D)

Ombytninger og Cykler



Lemma

- En ombytning af to brikker i **samme cykel** øger antallet af cykler med én.
- En ombytning af to brikker fra to **forskellige cykler** reducerer antallet af cykler med én.

Lemma

Når alle n brikker er korrekt placeret er der præcis n cykler.

Lemma

For at løse et puslespil med n brikker og k cykler I starten kræves $\geq n - k$ ombytninger.

Har vist en **nedre grænse** for
ALLE algoritmer der løser problemet

En (grådig) algoritme

Algoritme Puslespil

while der findes en brik x som ikke er placeret korrekt **do**

 lad y være brikken på x 's korrekte plads

 ombyt(x, y)

od

Lemma

Algoritmen bytter aldrig om på brikker der står korrekt.

Lemma

Algoritmen udfører $\leq n - 1$ ombytninger

Lemma

For at løse et puslespil med n brikker og k cykler I starten udfører algoritmen præcis $n - k$ ombytninger.

Har vist en **øvre grænse** for en konkret algoritme

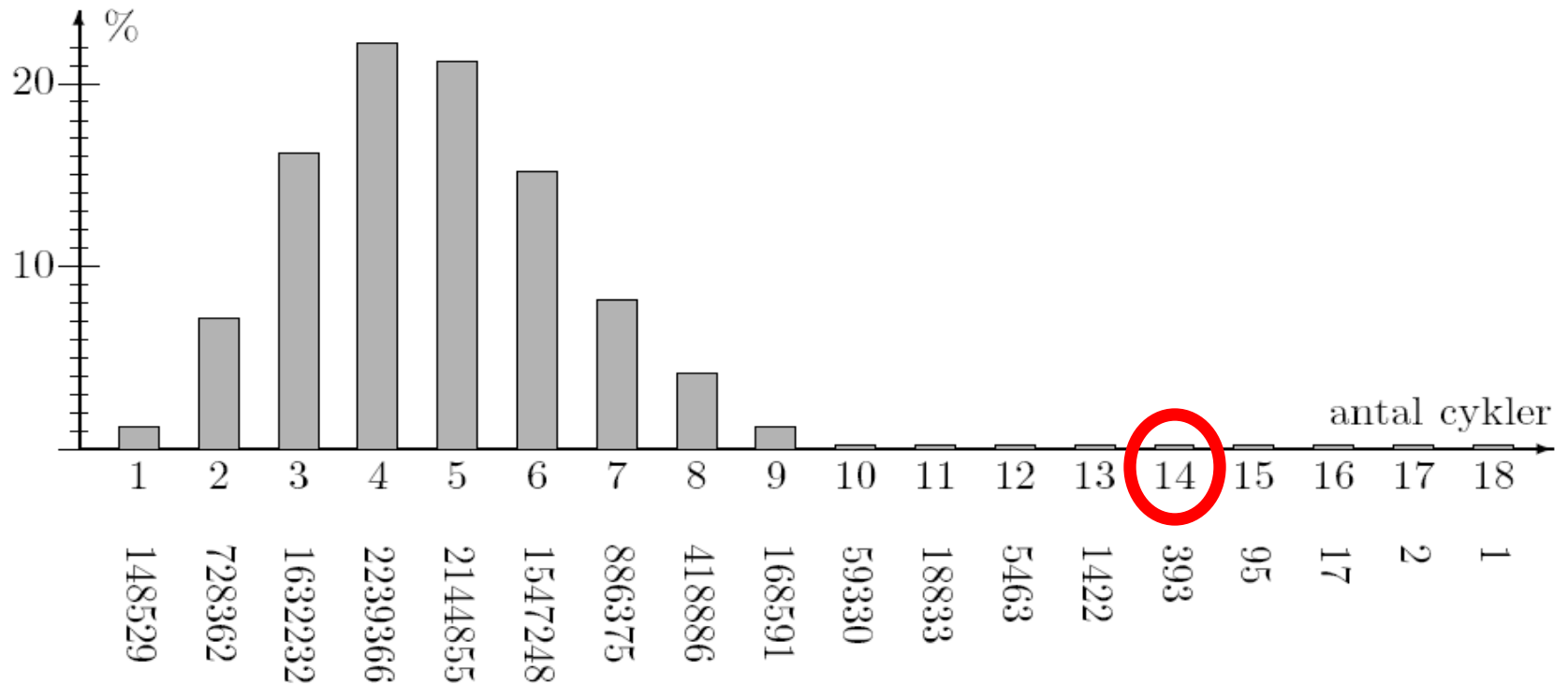
Algoritmen er **optimal** da antal ombytninger er bedst mulig

Sætning

For at løse et puslespil med n brikker og k cykler i starten kræves præcis $n - k$ ombytninger

Fordelingen af antal cykler

$n = 64$, 10.000.000 permutationer

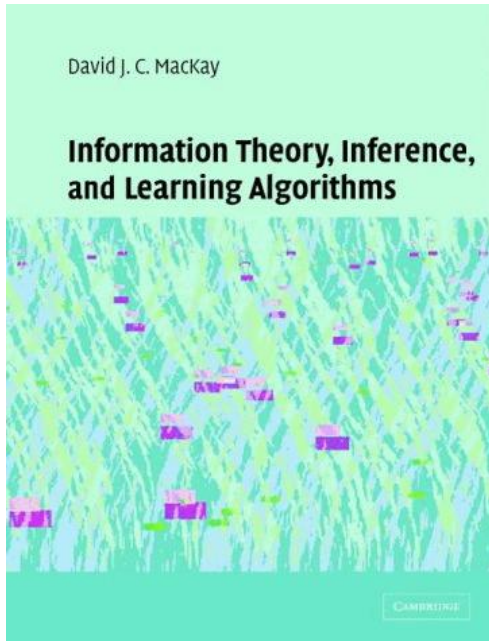


Hvad har vi så lært... ?

Algoritmisk indsigt...

- **Matematisk indsigt** (cykler)
- **Resourceforbrug** (antal ombytninger)
- **Nedre grænse** ($\geq n - k$ ombytninger)
- **Grådig algoritme**
- **Analyseret algoritmen** ($\leq n - k$ ombytninger)
- **Optimal algoritme** (argumenteret bedst mulig)
- **Input afhængig resourceforbrug**

Tilfældige permutationer...



Yderligere information kan findes i David J.C. MacKay, tillæg til *Information Theory, Inference, and Learning Algorithms*, om "Random Permutations", 4 sider.

<http://www.inference.phy.cam.ac.uk/mackay/itila/cycles.pdf>

**Et andet eksempel på en
beregningsprocess...**

Programming Pearls

Second Edition

JON BENTLEY

Bell Labs, Lucent Technologies
Murray Hill, New Jersey

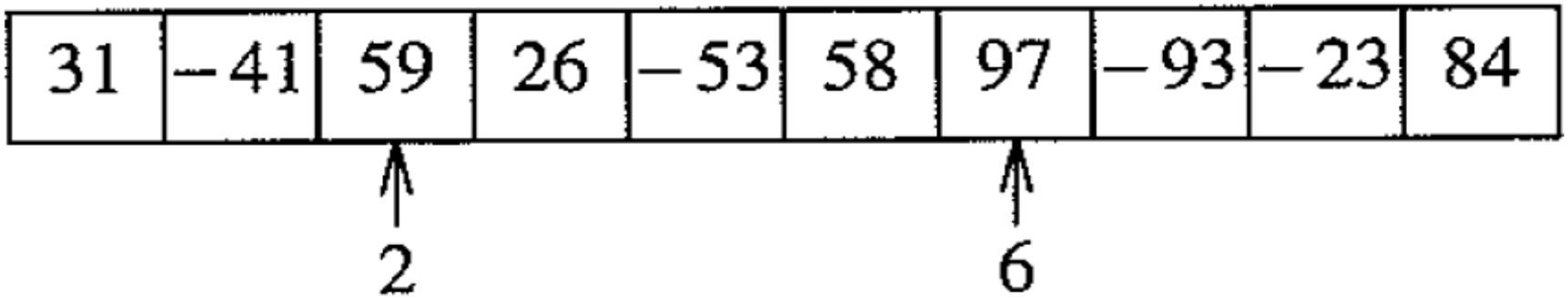


ACM Press
New York, New York

◆ Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Max-DeIsum



Algorithme 1

```
1 maxsofar = 0
2 for i = [0, n)
3     for j = [i, n)
4         sum = 0
5         for k = [i, j]
6             sum += x[k]
7         /* sum is sum of x[i..j] */
8         maxsofar = max(maxsofar, sum)
```

Algorithme 2

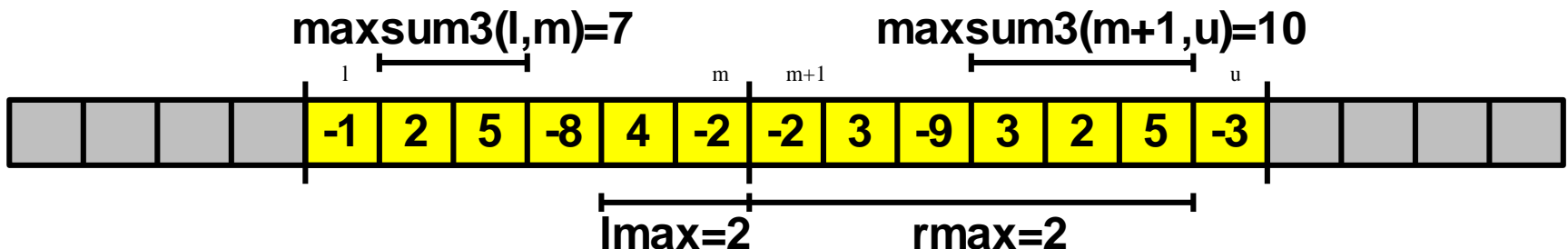
```
1 maxsofar = 0
2 for i = [0, n)
3     sum = 0
4     for j = [i, n)
5         sum += x[j]
6         /* sum is sum of x[i..j] */
7         maxsofar = max(maxsofar, sum)
```

Algorithme 2b

```
1 cumarr[-1] = 0
2 for i = [0, n)
3     cumarr[i] = cumarr[i-1] + x[i]
4 maxsofar = 0
5 for i = [0, n)
6     for j = [i, n)
7         sum = cumarr[j] - cumarr[i-1]
8         /* sum is sum of x[i..j] */
9         maxsofar = max(maxsofar, sum)
```

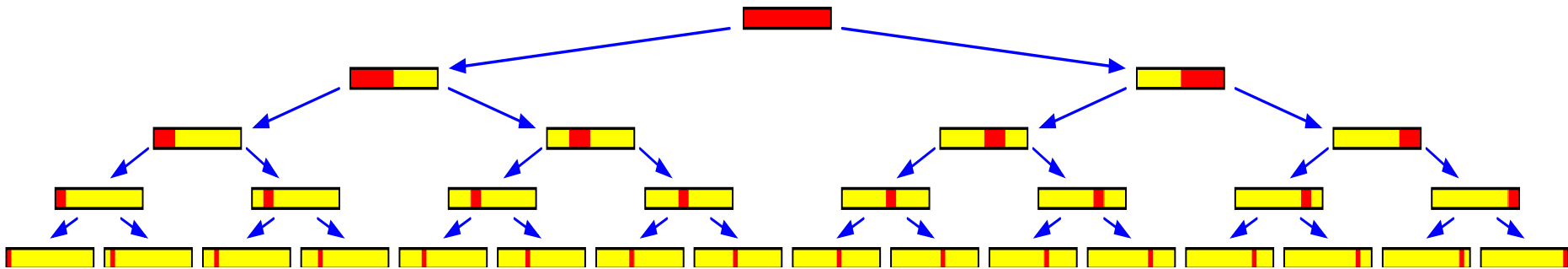
Algorithme 3

```
1 answer := maxsum3(0, n-1)
2 float maxsum3(l, u)
3     if (l > u) /* zero elements */
4         return 0
5     if (l == u) /* one element */
6         return max(0, x[l])
7
8     m = (l + u) / 2
9     /* find max crossing to left */
10    lmax = sum = 0
11    for (i = m; i >= l; i--)
12        sum += x[i]
13        lmax = max(lmax, sum)
14    /* find max crossing to right */
15    rmax = sum = 0
16    for i = (m, u]
17        sum += x[i]
18        rmax = max(rmax, sum)
19
20    return max(lmax+rmax, maxsum3(l, m), maxsum3(m+1, u))
```



Algoritme 3 : Analyse

Rekursionstræet



Observation

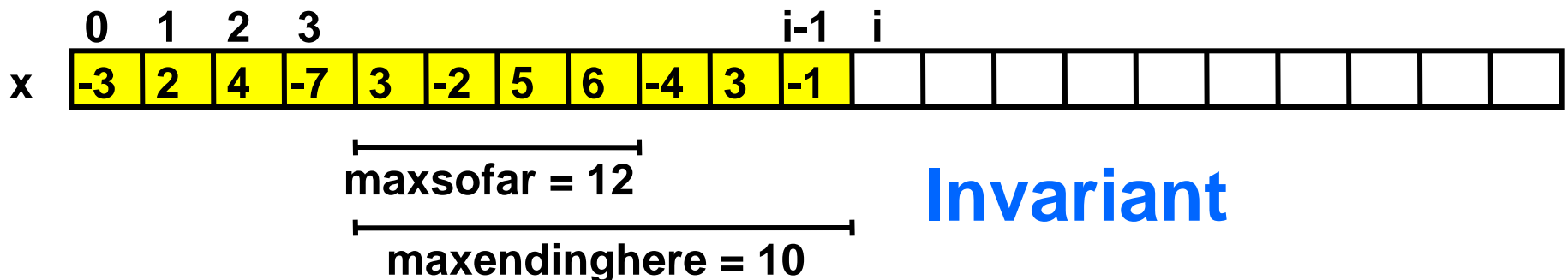
Samlet mængde additioner per lag er $\sim n$

Additioner

additioner $\sim n \cdot \# \text{ lag} \sim n \cdot \log_2 n$

Algorithme 4

```
1 maxsofar = 0
2 maxendinghere = 0
3 for i = [0, n)
4     /* invariant: maxendinghere and maxsofar
5        are accurate for x[0..i-1] */
6     maxendinghere = max(maxendinghere + x[i], 0)
7     maxsofar = max(maxsofar, maxendinghere)
```



Max-Delsum: Algoritmiske idéer

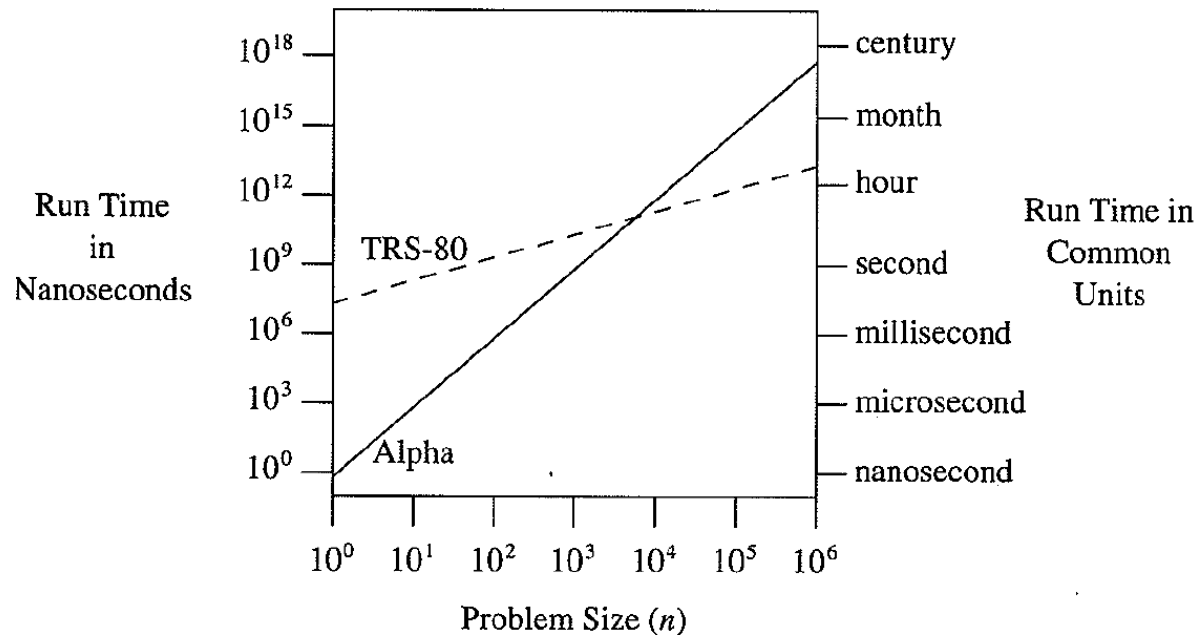
| Algoritme | # additioner | Idé |
|-----------|-----------------------|---------------------|
| 1 | $\sim n^3$ | Naive løsning |
| $2 + 2b$ | $\sim n^2$ | Genbrug beregninger |
| 3 | $\sim n \cdot \log n$ | Del-og-kombiner |
| 4 | $\sim n$ | Inkrementel |

Sammenligning

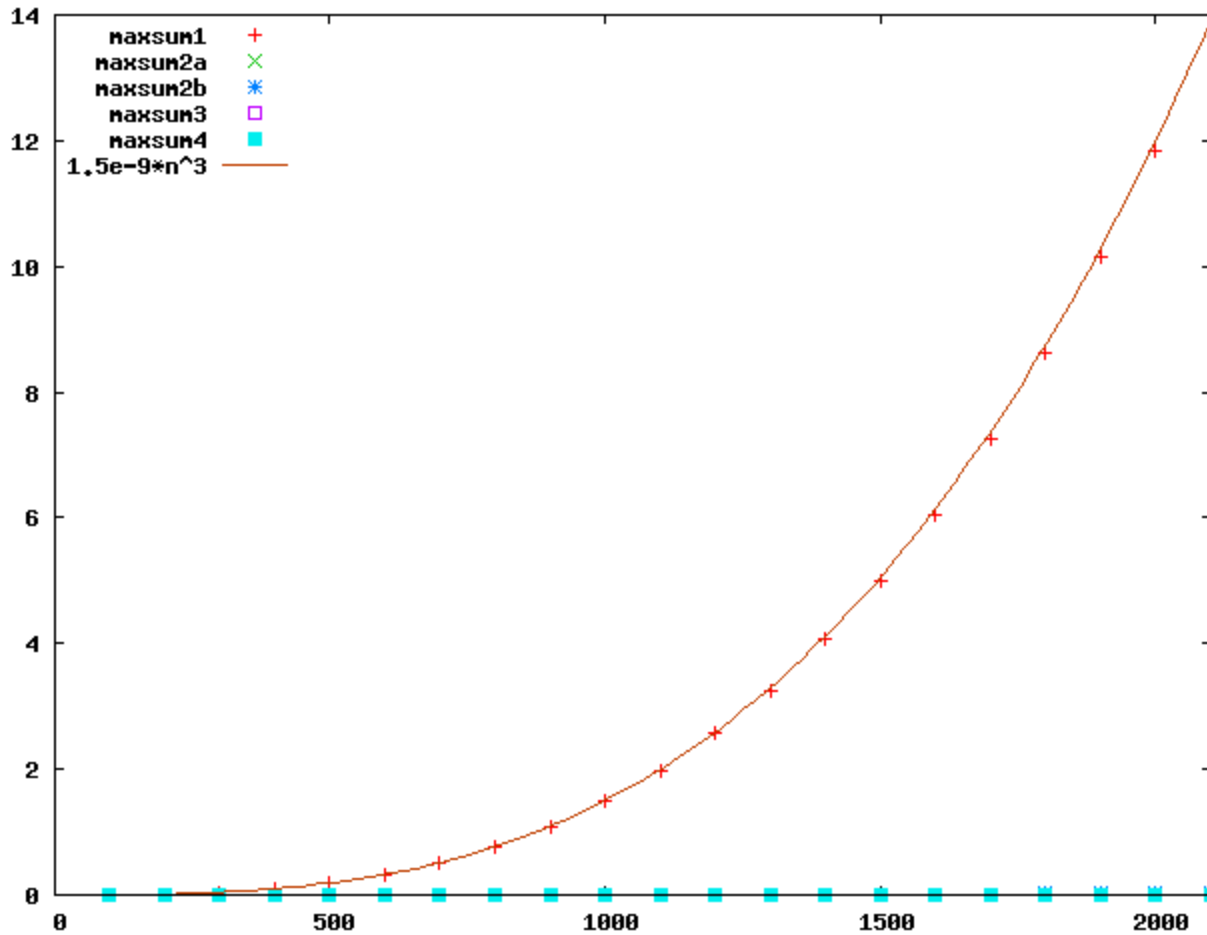
| ALGORITHM | | 1 | 2 | 3 | 4 |
|---|--------|--------------|-------------------|----------------------|----------------------|
| Run time in nanoseconds | | $1.3n^3$ | $10n^2$ | $47n \log_2 n$ | $48n$ |
| Time to solve a problem of size | 10^3 | 1.3 secs | 10 msecs | .4 msecs | .05 msecs |
| | 10^4 | 22 mins | 1 sec | 6 msecs | .5 msecs |
| | 10^5 | 15 days | 1.7 min | 78 msecs | 5 msecs |
| | 10^6 | 41 yrs | 2.8 hrs | .94 secs | 48 msecs |
| | 10^7 | 41 millennia | 1.7 wks | 11 secs | .48 secs |
| Max size problem solved in one | sec | 920 | 10,000 | 1.0×10^6 | 2.1×10^7 |
| | min | 3600 | 77,000 | 4.9×10^7 | 1.3×10^9 |
| | hr | 14,000 | 6.0×10^5 | 2.4×10^9 | 7.6×10^{10} |
| | day | 41,000 | 2.9×10^6 | 5.0×10^{10} | 1.8×10^{12} |
| If n multiplies by 10, time multiplies by | | 1000 | 100 | 10+ | 10 |
| If time multiplies by 10, n multiplies by | | 2.15 | 3.16 | 10- | 10 |

Sammenligning: n^3 og n

| n | ALPHA 21164A, C, CUBIC ALGORITHM | TRS-80, BASIC, LINEAR ALGORITHM |
|-----------|--|---------------------------------------|
| 10 | 0.6 microseconds | 200 millisecs |
| 100 | 0.6 milliseconds | 2.0 secs |
| 1000 | 0.6 secs | 20 secs |
| 10,000 | 10 mins | 3.2 mins |
| 100,000 | 7 days | 32 mins |
| 1,000,000 | 19 yrs | 5.4 hrs |

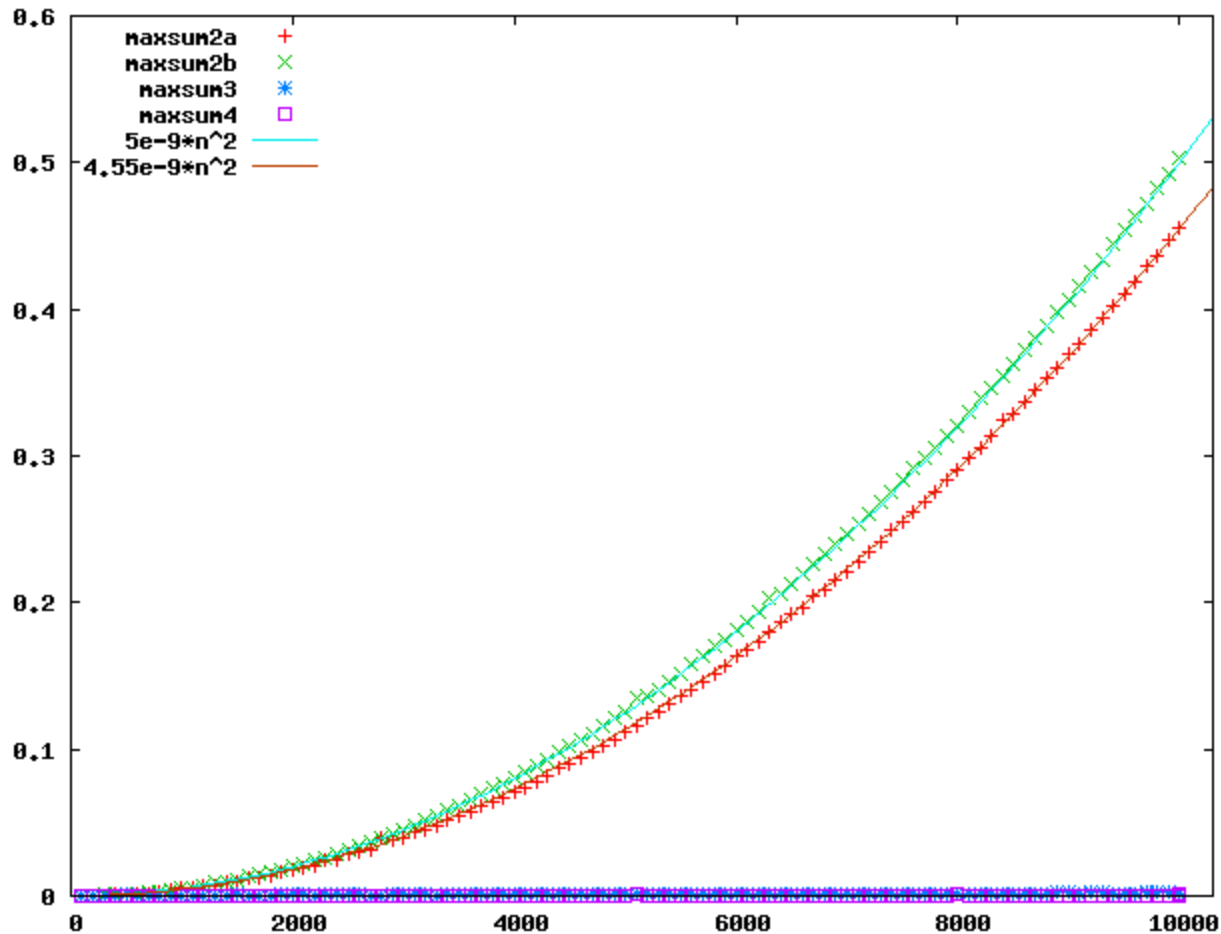


Sammenligning 2009



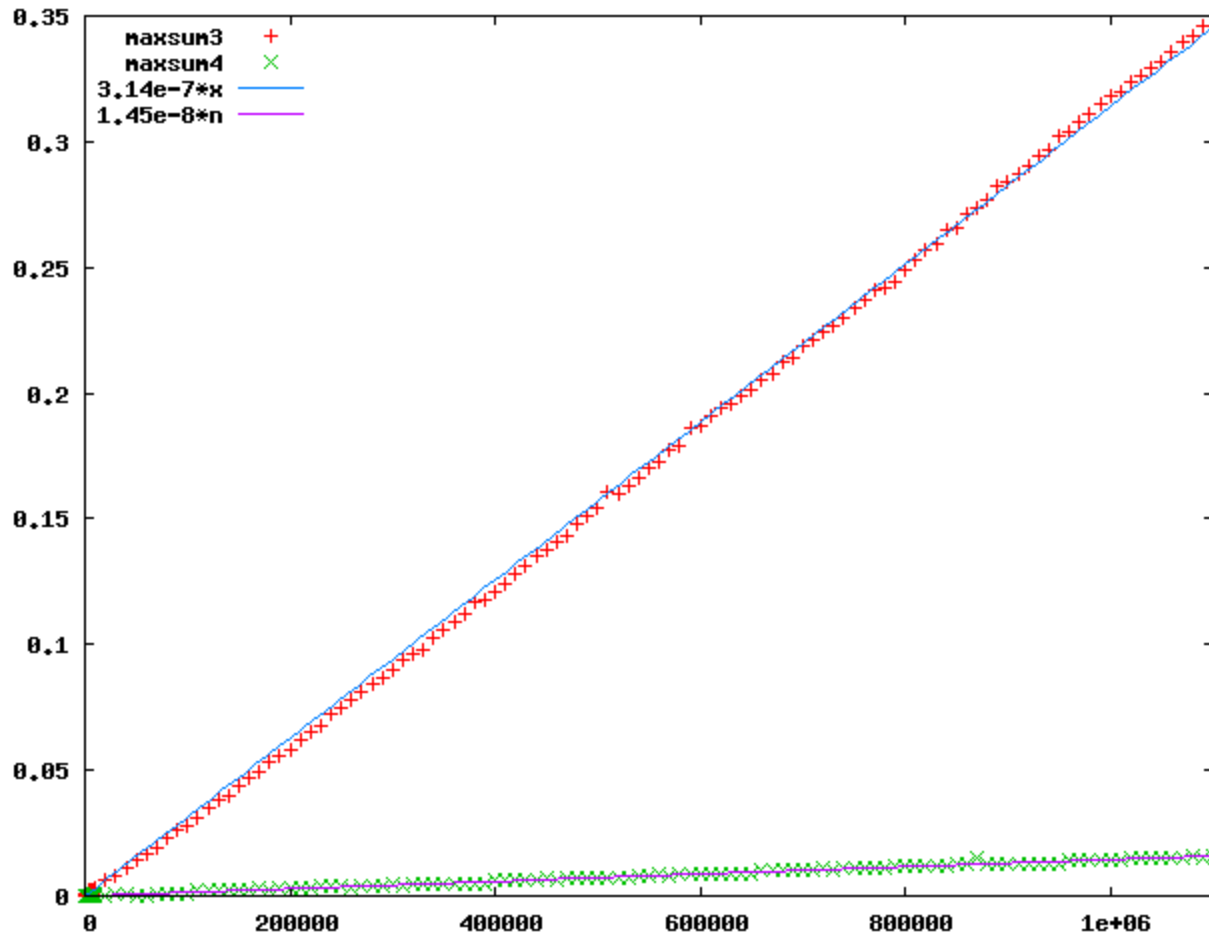
$$\text{maxsum1} \approx n^3$$

Sammenligning 2009



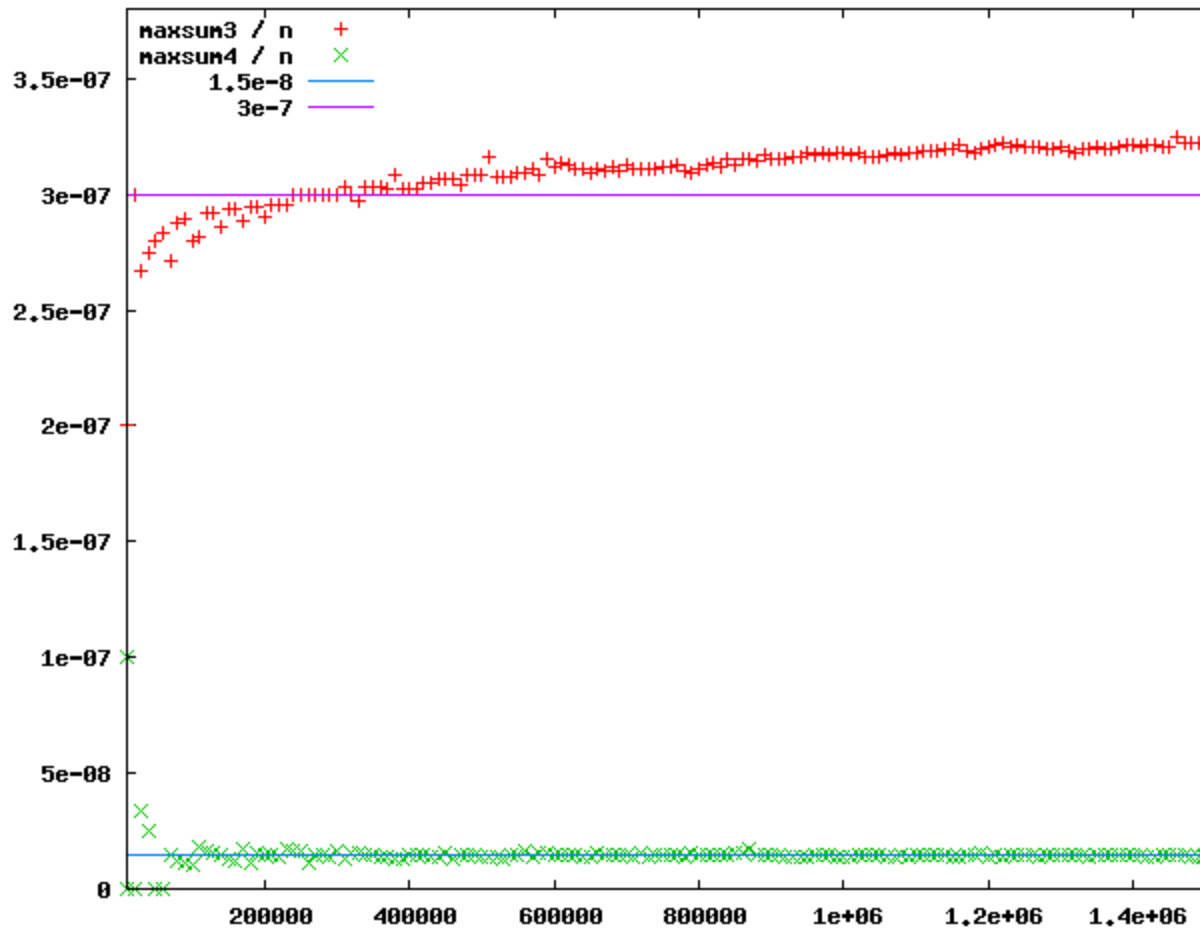
maxsum2a og maxsum2b $\approx n^2$

Sammenligning 2009



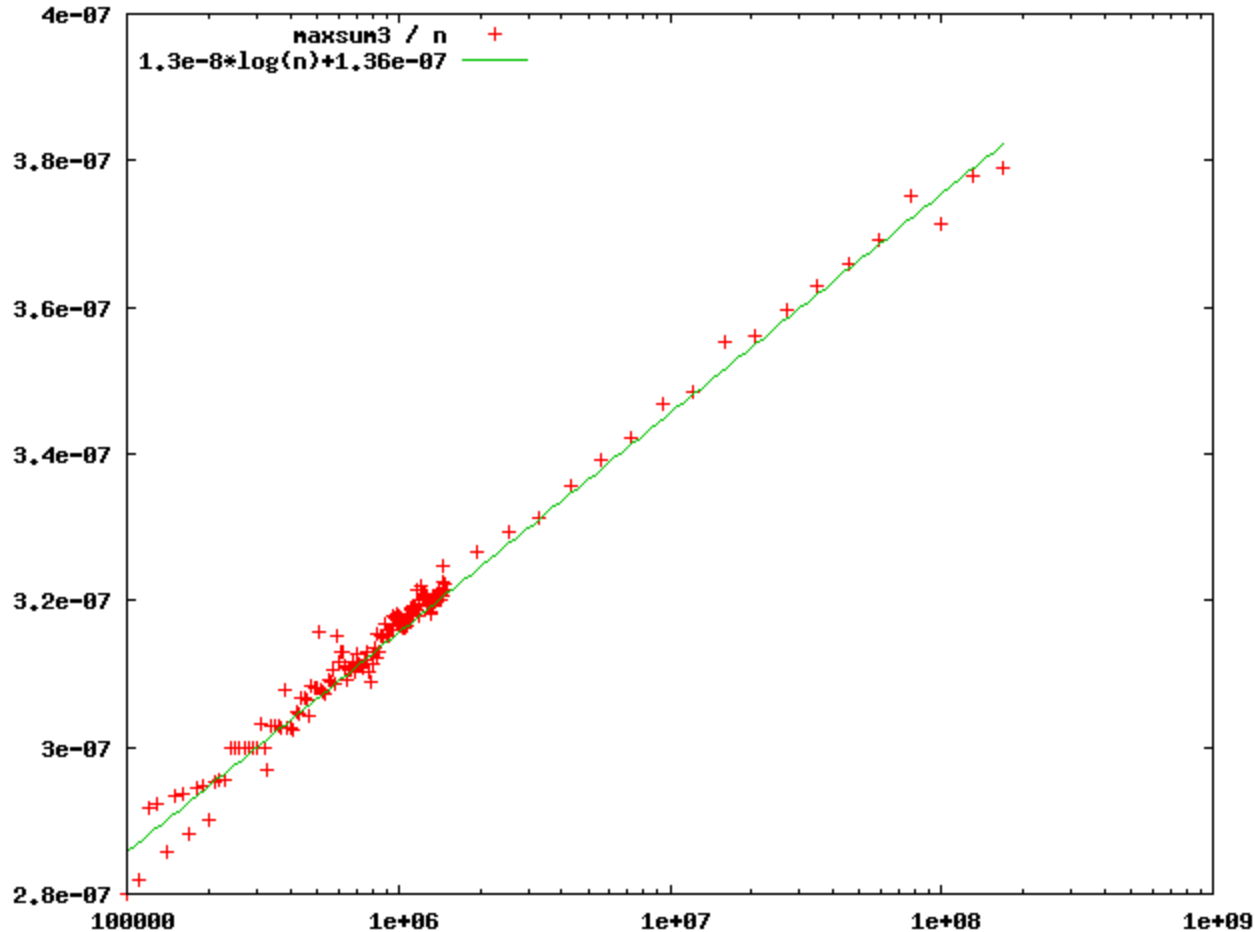
maxsum3 og maxsum4 $\approx n$???

Sammenligning 2009



$$\text{maxsum4} \approx n$$

Sammenligning 2009



$$\text{maxsum3} \approx c_1 \cdot n \cdot \log n + c_2 \cdot n$$

x-akse = n , y = sekunder, hvert eksperiment gennemsnit af 10 kørsler (gcc 4.1.2, C, Linux 2.6.18, Intel Xeon 3 GHz)

Algoritmisk indsigt...

- Gode idéer kan give hurtige algoritmer
- Generelle algoritme teknikker
 - Del-og-kombiner
 - Inkrementel
- Analyse af udførelsestid
- Argumenteret for korrektheden
- Invarianter