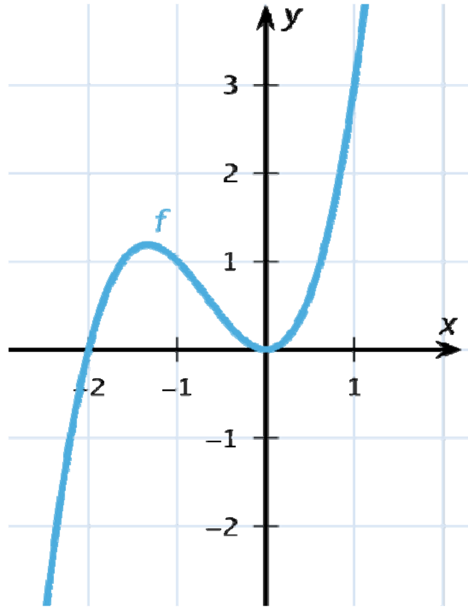
Algoritmer og Datastrukturer

Evaluering af polynomier [Polynomier]
Maximum delsum [Bentley kap. 8]

Evaluering af Polynomier

- Vi har et *n*'te grads polynomium P:

- $P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_1 x + \alpha_0$

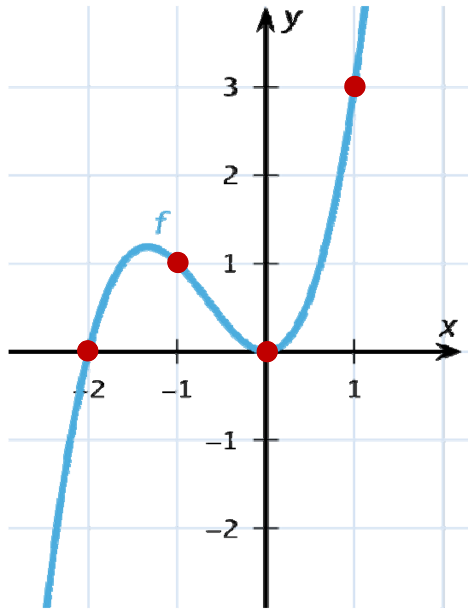


- Vi vil gerne evaluere det i et *x*.
- Hvor meget arbejde?

Evaluering af Polynomier

Eksempel

- $n = 3$
- $P(x) = x^3 + 2x^2$
- $\alpha_3=1,\ \alpha_2=2,\ \alpha_1=0,\ \alpha_0=0$



- P(-2)=0, P(-1)=1, P(0)=0, P(1)=3

Evaluering af Polynomier

- $P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_1 x + \alpha_0$
- **Beregn P(x)**

```
S = 0
for i = 0,…,n:
    B = 1
    for j = 1,…,i:
        B = B * x
    S = S + αi * B
return S
```

i=6 og j=4

Delresultat

Få B til at blive x^i

Udregn bidrag fra $\alpha_i x^i$ og læg til S

$$P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_6 \overbrace{xxxxxx}^{B = x^4} + \overbrace{\alpha_5 xxxxx + \alpha_4 xxxx + \alpha_3 xxx + \alpha_2 xx + \alpha_1 x + \alpha_0}^{S}$$

Evaluering af Polynomier

- $P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \cdots + \alpha_1 x + \alpha_0$
- **Beregn P(x)**

```
S = 0
for i = 0,…,n:
    B = 1
    for j = 1,…,i:
        B = B * x
    S = S + αi * B
return S
```

Hvor mange gange bliver
B = B * x ca. udført?

a) $\log_2(n)$
b) n
c) $n * \log_2(n)$
d) n^2
e) n^3

Evaluering af Polynomier

- **Beregn P(x)**

```
S = 0
for i = 0,…,n:
    B = 1
    for j = 1,…,i:
        B = B * x
    S = S + αi * B
return S
```

Vi laver næsten det samme arbejde for i-1 og i

Lad os prøve at **genbruge resultater**!

Evaluering af Polynomier

- **Beregn P(x)**

  S = 0
  for i = 0,…,n:
      B = 1
      for j = 1,…,i:
          B = B * x
      S = S + α_i * B
  return S

  $n^2/2 + n/2$

- **Beregn P(x)** *ny*

  S = 0
  B = 1
  for i = 0,…,n:
      S = S + α_i * B
      B = B * x
  return S

  2(n+1)

$$P(x) = \alpha_n x^n + \alpha_{n-1}x^{n-1} + \cdots + \alpha_6 x\underbrace{xxxxx}_{B\,*\,x} + \alpha_5\underbrace{xxxxx}_{B} + \underbrace{\alpha_4 xxxx + \alpha_3 xxx + \alpha_2 xx + \alpha_1 x + \alpha_0}_{S}$$

Evaluering af Polynomier

- Hvor stor forskel på moderne computer der kan lave ca. 10^9 instruktioner på 1 sekund?

Degree n:	10^2	10^4	10^6	10^8
Naive ($n^2/2 + n/2$ work):	5 microseconds	50 milliseconds	8 minutes	2 months
Re-use ($2(n+1)$ work):	0.1 microseconds	20 microseconds	2 milliseconds	0.2 seconds

Alternativ
Evaluering af Polynomier

$$P(x) = \alpha_n x^n + \alpha_{n-1}x^{n-1} + \cdots + \alpha_2 x^2 + \alpha_1 x + \alpha_0$$

$$= ((\underbrace{(\cdots\underbrace{(\underbrace{(\underbrace{\alpha_n}x + \alpha_{n-1})}x + \alpha_{n-2})}x + \cdots)}_{S}x + \alpha_2)x + \alpha_1)x + \alpha_0$$

- **Beregn P(x)**

  S = 0

  for i = n,…,0:

  $\quad S = S * x + \alpha_i$

  return S

n+1

Programming Pearls

Second Edition

JON BENTLEY

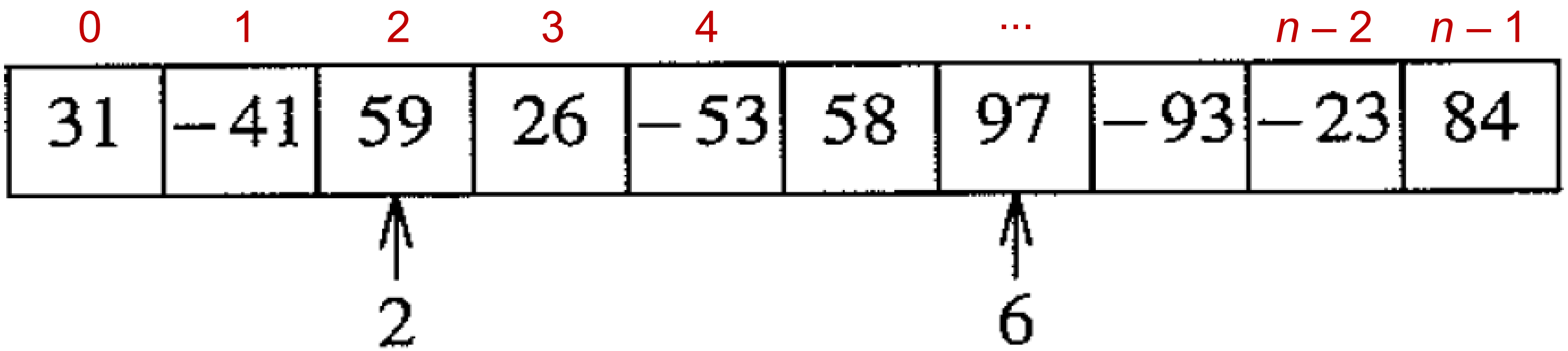
Bell Labs, Lucent Technologies
Murray Hill, New Jersey

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-Delsum



Hvad er Max-Delsum ?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	5	-4	-5	2	-3	4	2	-3	5	6	-2	3	-7	2	-6	10

a) 10
b) 11
c) 14
d) 15
e) 17
f) 20
g) 30
h) ved ikke

Algoritme 1

```
maxsofar = 0
for i = [0, n)
    for j = [i, n)
        sum = 0
        for k = [i, j]
            sum += x[k]
        /* sum is sum of x[i..j] */
        maxsofar = max(maxsofar, sum)
```

Antal additioner:

$$\sum_{l=1}^{n} l(n-l+1) = (n+1)\sum_{l=1}^{n} l - \sum_{l=1}^{n} l^2 = (n+1)\frac{n(n+1)}{2} - \frac{n(n+1)(2n+1)}{6} = \frac{n^3+3n^2+2n}{6}$$

www.cs.au.dk/~gerth/slides/math.pdf

Algoritme 2

```
maxsofar = 0
for i = [0, n)
    sum = 0
    for j = [i, n)
        sum += x[j]
        /* sum is sum of x[i..j] */
        maxsofar = max(maxsofar, sum)
```

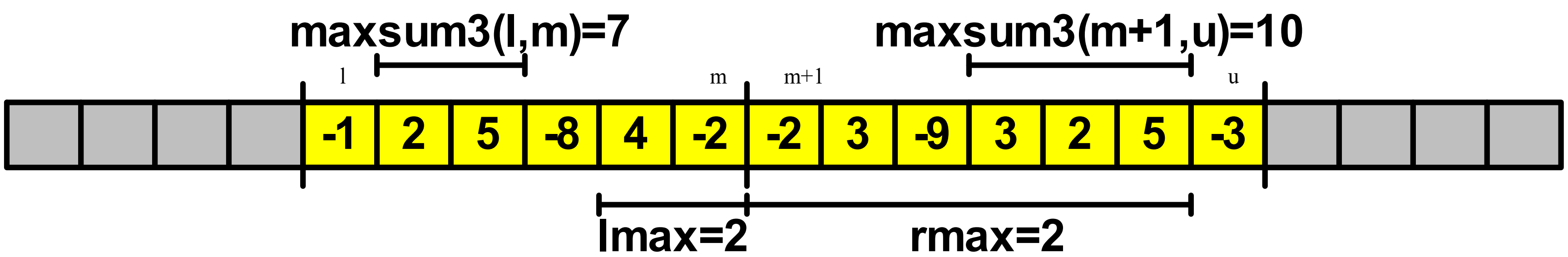
Algoritme 2b

```
cumarr[-1] = 0
for i = [0, n)
    cumarr[i] = cumarr[i-1] + x[i]
maxsofar = 0
for i = [0, n)
    for j = [i, n)
        sum = cumarr[j] - cumarr[i-1]
        /* sum is sum of x[i..j] */
        maxsofar = max(maxsofar, sum)
```

Algoritme 3

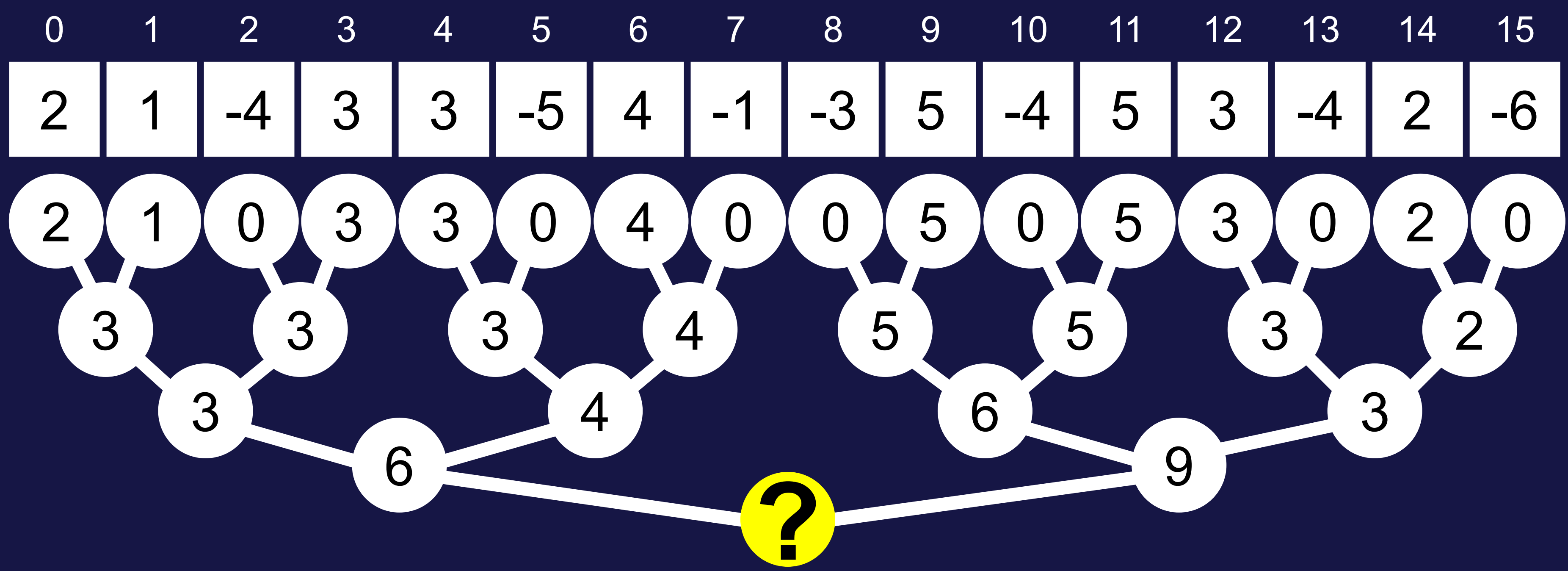
```
answer = maxsum3(0, n-1)
float maxsum3(l, u)
    if (l > u)  /* zero elements */
        return 0
    if (l == u)  /* one element */
        return max(0, x[l])

    m = (l + u) / 2
    /* find max crossing to left */
    lmax = sum = 0
    for (i = m; i >= l; i--)
        sum += x[i]
        lmax = max(lmax, sum)
    /* find max crossing to right */
    rmax = sum = 0
    for i = (m, u]
        sum += x[i]
        rmax = max(rmax, sum)

    return max(lmax+rmax, maxsum3(l, m), maxsum3(m+1, u))
```

rekursive metodekald



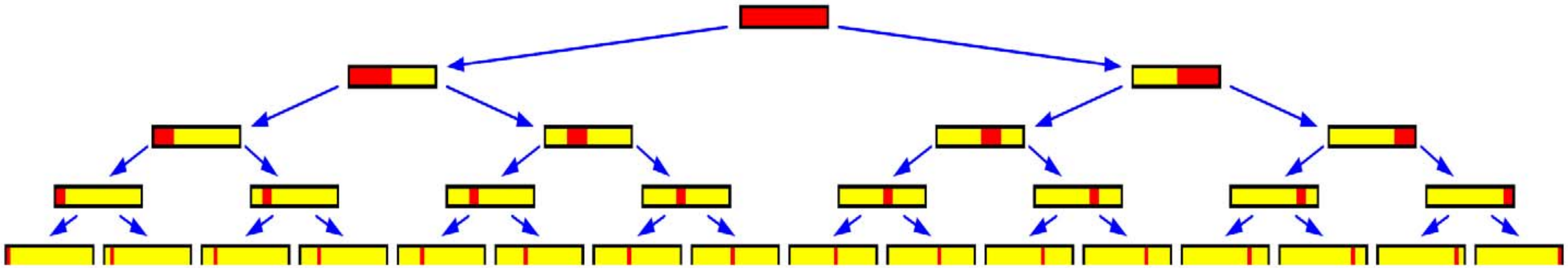
Algoritme 3

	lmax	rmax	maxsum3
a)	6	9	10
b)	5	8	13
c)	0	0	9
d)	4	6	10
e)	6	9	15
f)	ved ikke		



Algoritme 3 : Analyse

Rekursionstræet



Observation

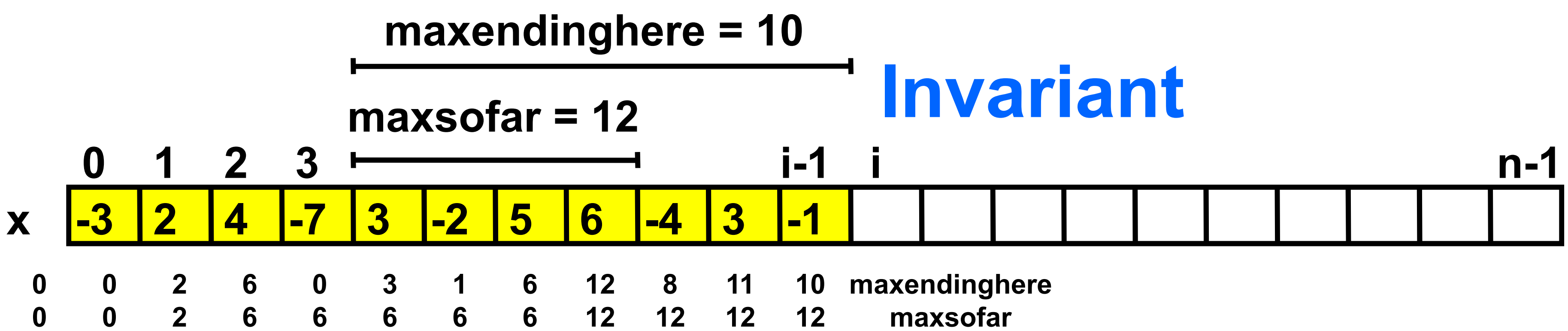
Samlet mængde additioner per lag er $\sim n$

Additioner

$\#$ additioner $\sim n \cdot \#$ lag $\sim n \cdot \log_2 n$

Algoritme 4

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



Max-Delsum: Algoritmiske idéer

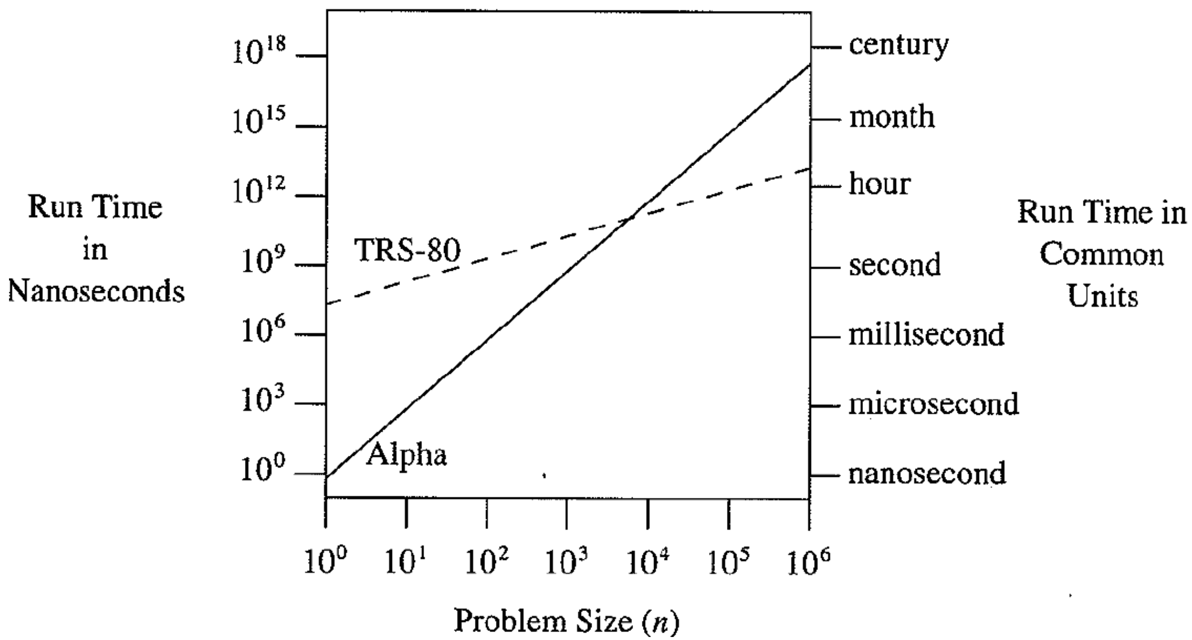
Algoritme	# additioner	Idé
1	~ n^3	**Naive løsning**
2 + 2b	~ n^2	**Genbrug beregninger** sum(x[i..j]) = sum(x[i..j-1]) + x[j] sum(x[i..j]) = sum(x[0..j]) - sum(x[0..i-1])
3	~ $n \cdot \log n$	**Del-og-kombiner**
4	~ n	**Inkrementel**

Sammenligning

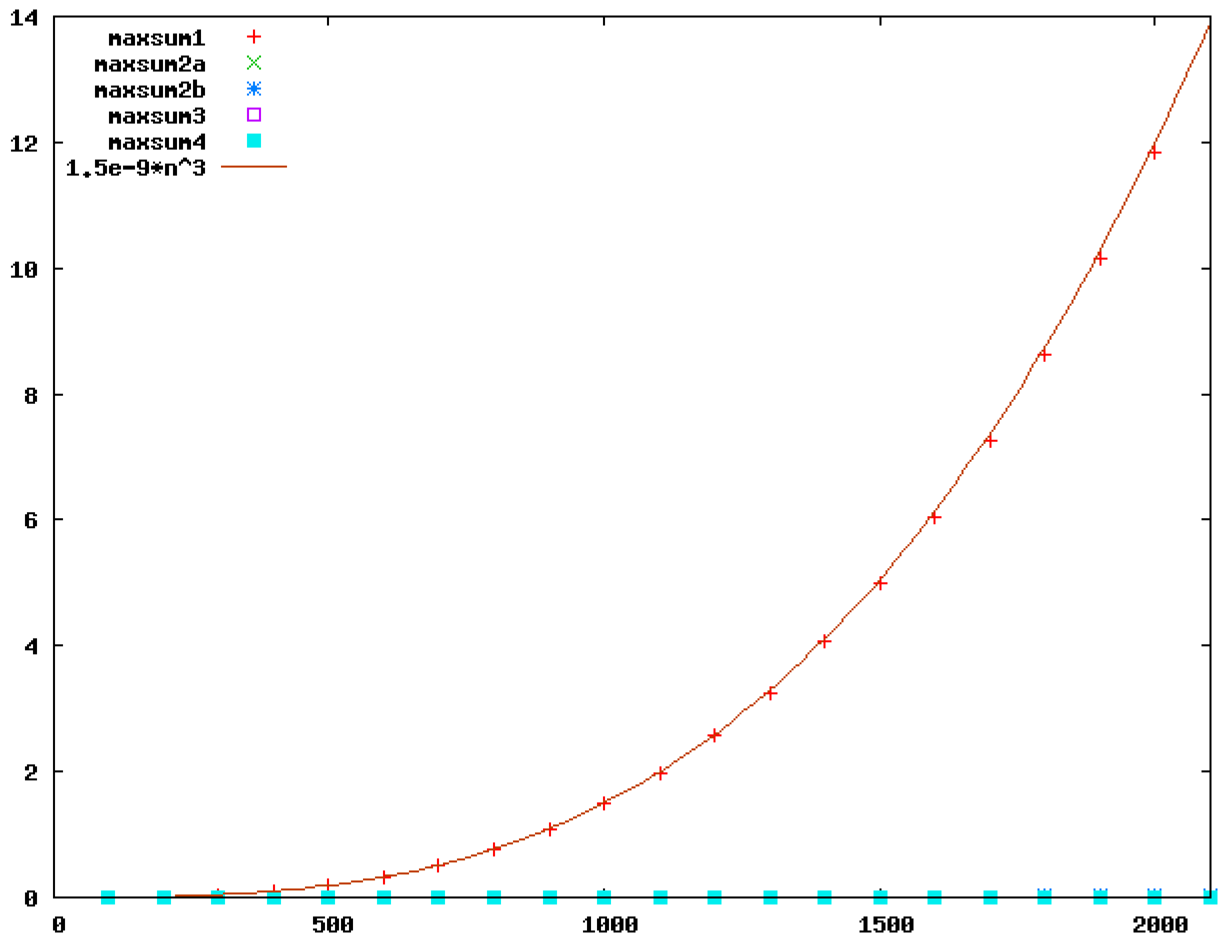
ALGORITHM		1	2	3	4
Run time in nanoseconds		$1.3n^3$	$10n^2$	$47n \log_2 n$	$48n$
Time to solve a problem of size	10^3	1.3 secs	10 msecs	.4 msecs	.05 msecs
	10^4	22 mins	1 sec	6 msecs	.5 msecs
	10^5	15 days	1.7 min	78 msecs	5 msecs
	10^6	41 yrs	2.8 hrs	.94 secs	48 msecs
	10^7	41 millennia	1.7 wks	11 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 microsecs	200 millisecs
100	0.6 millisecs	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



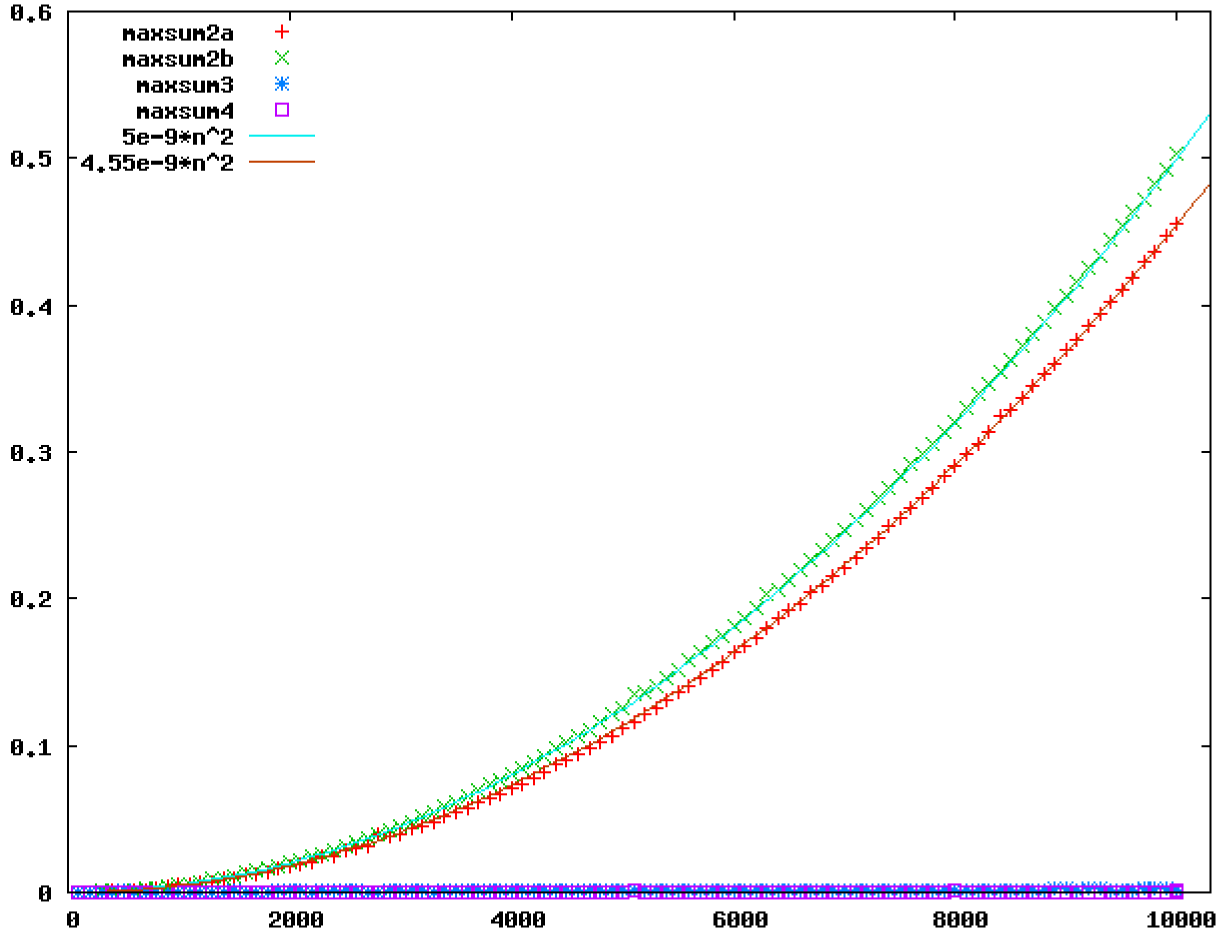
Sammenligning 2009



maxsum1 ≈ n^3

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

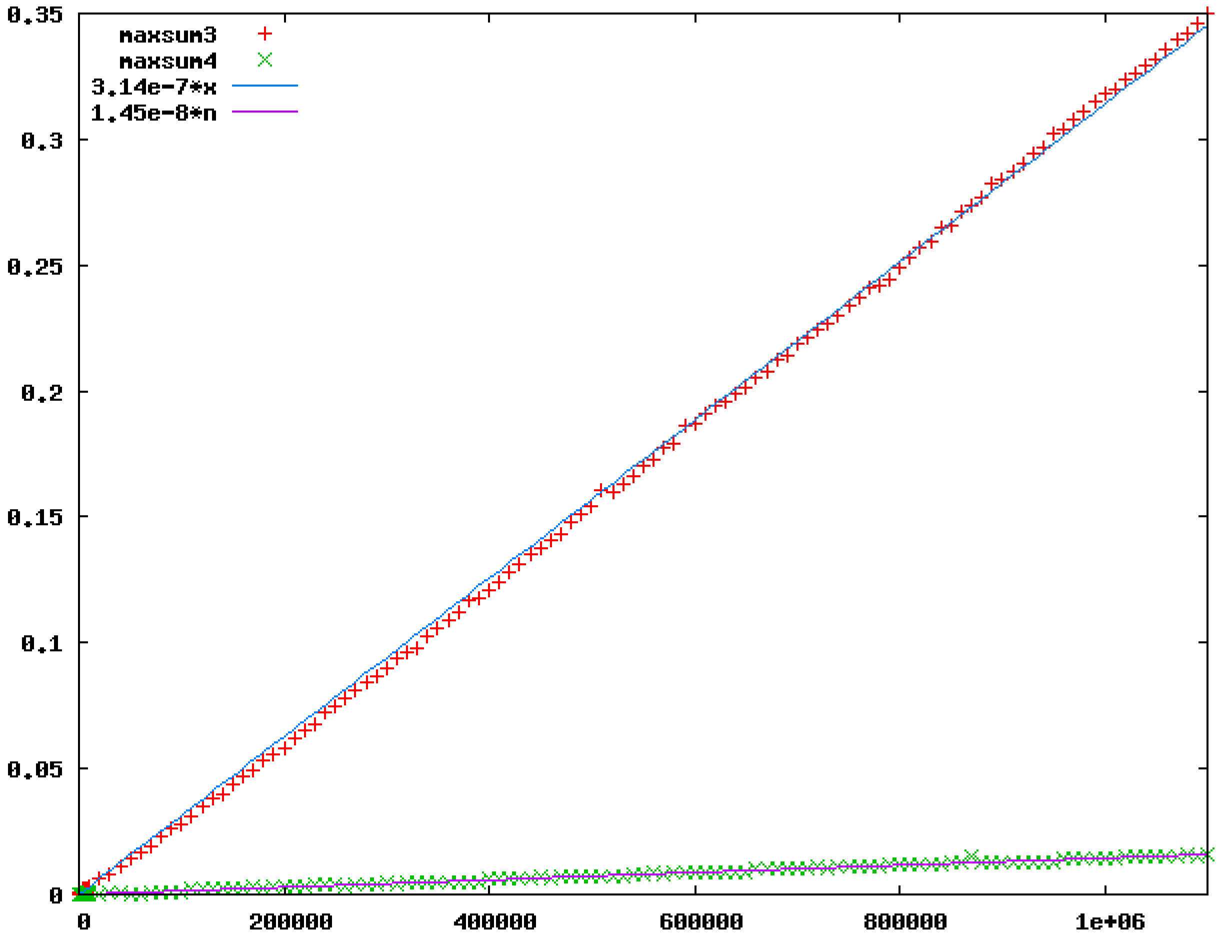
Sammenligning 2009



maxsum2a og maxsum2b ≈ n^2

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

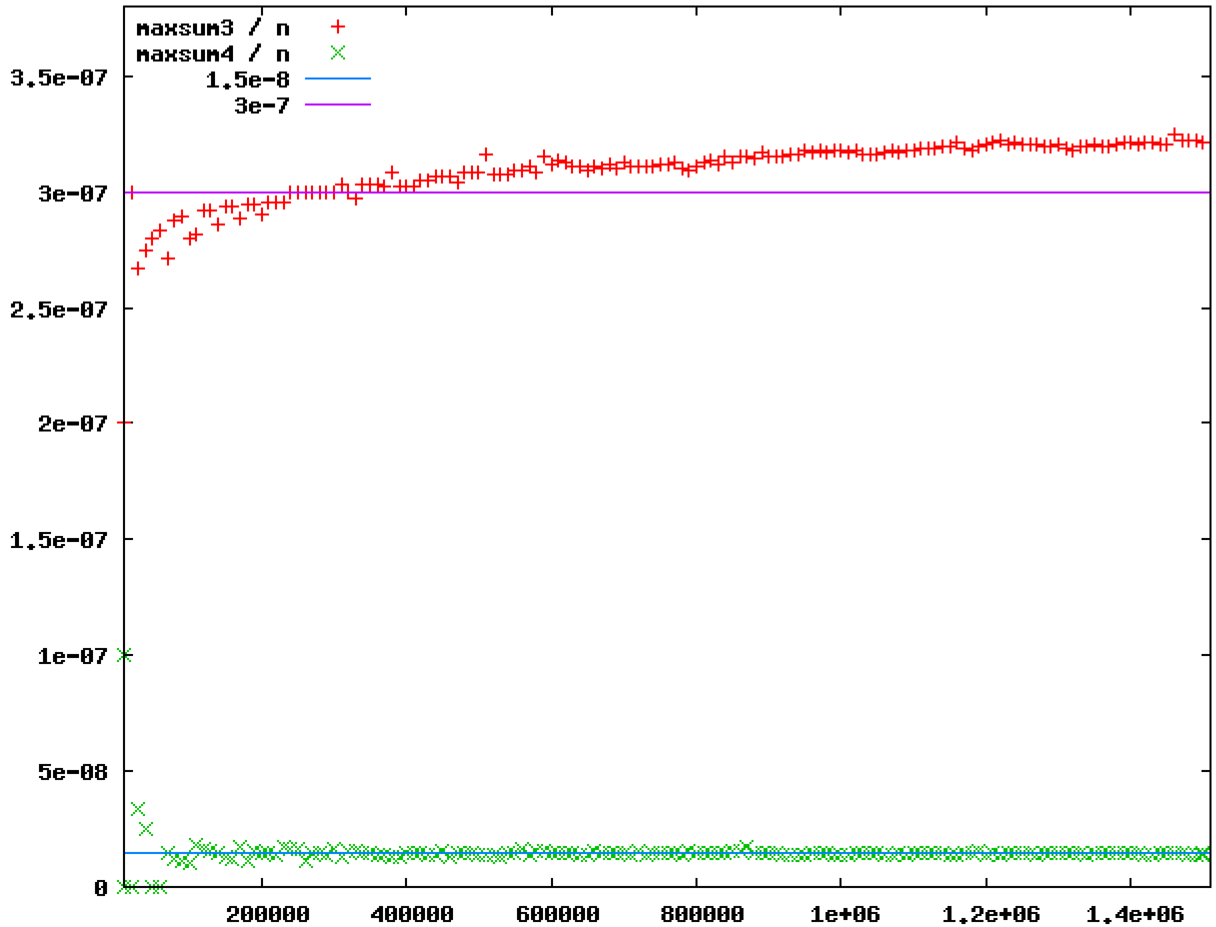
Sammenligning 2009



maxsum3 og maxsum4 ≈ *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)

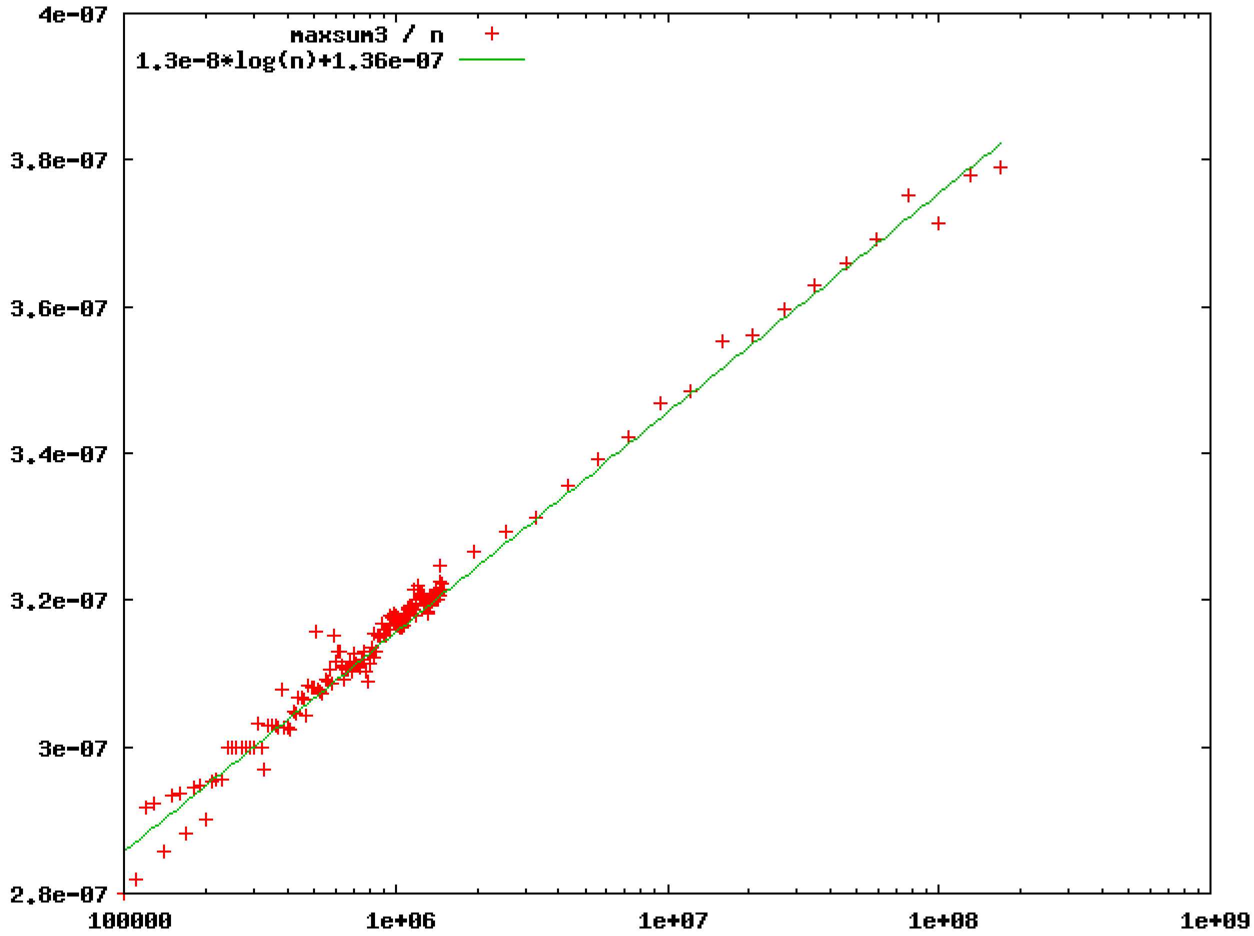
Sammenligning 2009



maxsum4 ≈ *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)

Sammenligning 2009

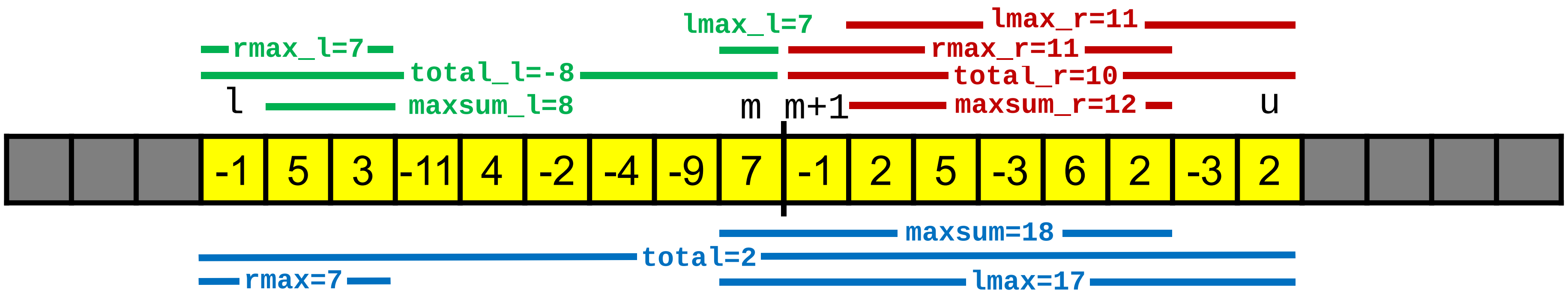


$$\text{maxsum3} \approx c_1 \cdot n \cdot \log n + c_2 \cdot n$$

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

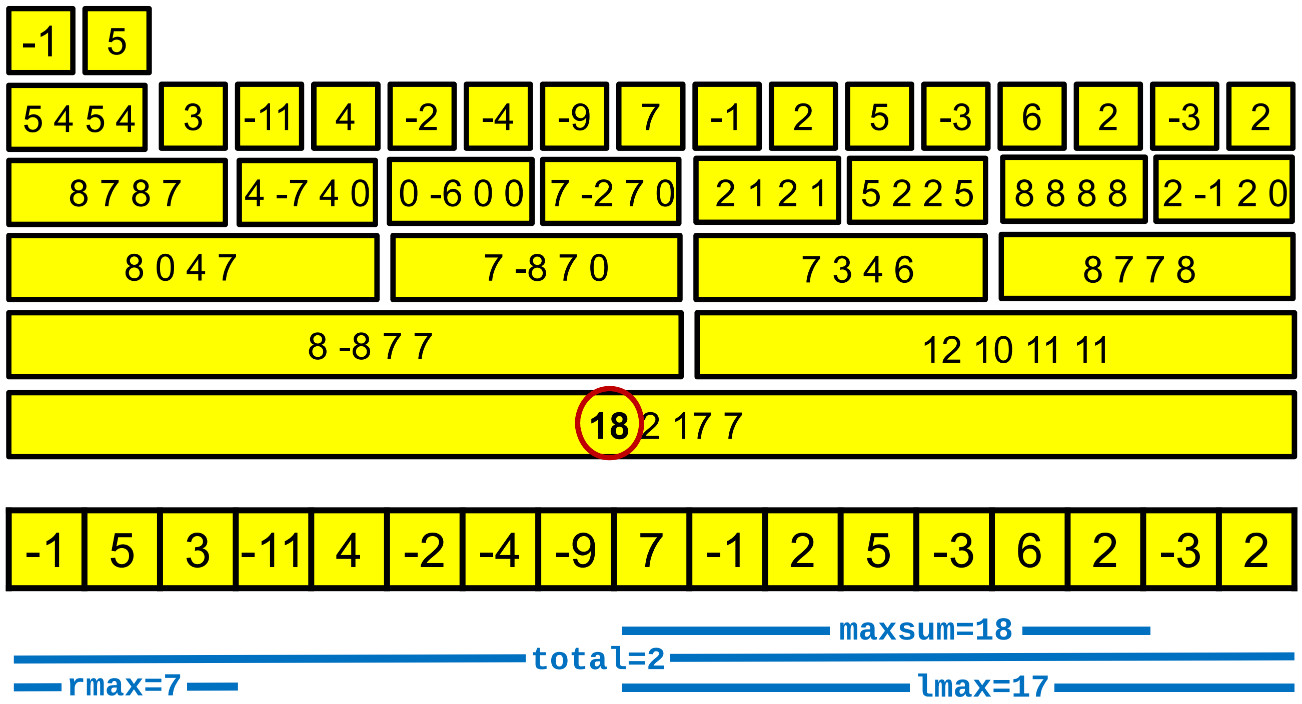
Algoritme 5 (del-og-kombiner)

```
proc maxsum5(l, u)  /* return (maxsum, total, lmax, rmax) */
    if (u < l)  /* zero elements */
        return (0, 0, 0, 0)
    if (u == l)  /* one element */
        maxsum = max(0, x[l])
        return (maxsum, x[l], maxsum, maxsum)
    m = (l + u) / 2
    (maxsum_l, total_l, lmax_l, rmax_l) = maxsum5(l, m)
    (maxsum_r, total_r, lmax_r, rmax_r) = maxsum5(m + 1, u)
    maxsum = max(maxsum_l, maxsum_r, lmax_l + rmax_r)
    total = total_l + total_r
    lmax = max(lmax_l + total_r, lmax_r)
    rmax = max(rmax_r + total_l, rmax_l)
    return (maxsum, total, lmax, rmax)
(maxsum, total, lmax, rmax) = maxsum5(0, n - 1)
answer = maxsum
```



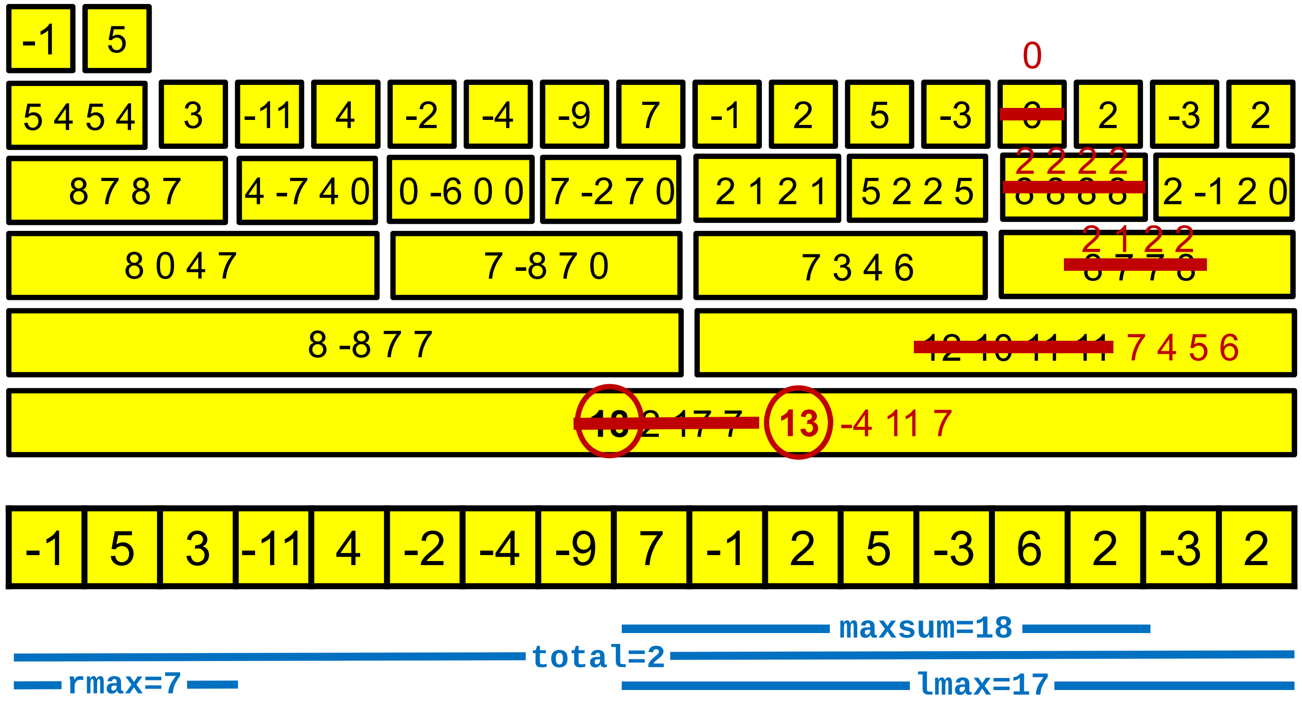
Algoritme 5

Beregn for hvert rekursivt kald (**maxsum**, **total**, **lmax**, **rmax**)



Totalt 2n-1 delproblemer → tid ~ n

Dynamisk maksimum delsum



Husk alle udregninger af Algoritme 5(`maxsum`, `total`, `lmax`, `rmax`)

Totalt ~log n delproblemer skal genberegnes for én ændring af input

→ tid ~ log n

Algoritmisk indsigt...

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

Afleveringsopgave

- Bentley 8.7.13:

-10	5	24	3	-100	4
56	5	-13	-16	80	-10
3	-2	0	-10	19	45
-34	-20	100	4	-5	10
18	8	-6	-4	-50	-50
3	14	-42	-33	15	7

- Find største sum i et del-rektangel.
- Input *n* x *n*. Alt fra n^6 og ned er OK.
- Prøv at argumentere for tid og korrekthed.