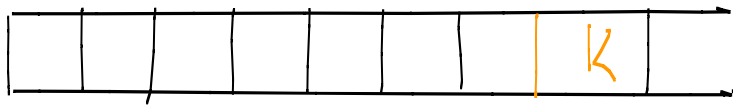


# Entwurf & Analyse von Algorithmen

Note Title

20.10.2008



\* leeres Feld wenn sonst keine pos. Teilfelder

→ k ist sicher immer dabei  
⇒  $T_{k-1}$  ist das längste  
das noch Sinn macht

Ziel ist es das größte zusammenhängende Summenfeld zu finden die am Stelle k endet;  
jede Stelle wird als k betrachtet und berechnet.  
Die Scanline arbeitet somit nur lokal,

Beispiel:

100	-70	23		50	-70	23		50	-70	150
$T_k$ 100	30	53		50	0	23		50	0	150

# Algorithmus für Scanlines:

```
max := 0;
von := 0;
bis := 0;
v := 1;
T := 0;

for k:= 1 to n do
  T:= T+A[k];
  if T < 0 then
    T:= 0;
    v = k + 1;
  end if
  if T > max then
    max = T;
    bis := k;
    von := v ;
  end if
end for
```

Es muss jedes Element min. einmal durchlaufen werden  $\Rightarrow O(n)$

Multiplizieren langer Zahlen:

$$\begin{array}{cccc} P_1 & P_2 & \dots & P_n \\ \downarrow & \downarrow & & \downarrow \\ 10^{n-1} & 10^{n-2} & & 10^0 = 1 \end{array}$$

$$p = \sum_{i=1}^n P_i \cdot 10^{n-i}$$

$$q = \sum_{i=1}^n Q_i \cdot 10^{n-i}$$

Divide & conquer:

$$p = \underbrace{a}_{2^k} \underbrace{b}_{2^k} = 10^{2^k} \cdot a + b$$

$$q = \underbrace{c}_{2^k} \underbrace{d}_{2^k} = 10^{2^k} c + d$$

$$p \cdot q = a \cdot c \cdot 10^n + (a \cdot d + c \cdot b) \cdot 10^{\frac{n}{2}} + b \cdot d$$

$\hookrightarrow$  realisiert als Bitshift  $\leftarrow$

Beispiel:

$$T(n) = 4T\left(\frac{n}{2}\right) + \mathcal{O}(n)$$

$$T\left(\frac{n}{2}\right) = 4T\left(\frac{n}{4}\right) + \mathcal{O}\left(\frac{n}{2}\right)$$

$$\begin{aligned} T(n) &= 4 \left( 4T\left(\frac{n}{4}\right) + \mathcal{O}\left(\frac{n}{2}\right) \right) + \mathcal{O}(n) \\ &= 2^3 T\left(\frac{n}{2^2}\right) + 2^2 \mathcal{O}\left(\frac{n}{2^1}\right) + \mathcal{O}(n) \end{aligned}$$

$$T\left(\frac{n}{4}\right) = 4T\left(\frac{n}{8}\right) + \mathcal{O}\left(\frac{n}{4}\right)$$

$$\begin{aligned} T(n) &= 2^4 T\left(\frac{n}{2^3}\right) + 2^4 \mathcal{O}\left(\frac{n}{2^2}\right) + 2 \mathcal{O}(n) + \mathcal{O}(n) \\ &= 2^{k+1} T\left(\frac{n}{2^k}\right) + 2^2 \mathcal{O}(n) + 2^1 \mathcal{O}(n) + \mathcal{O}(n) \\ &= 2^{k+1} T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} 2^i \mathcal{O}(n) \end{aligned}$$

$$\frac{n}{2^k} \stackrel{!}{=} 1 \Rightarrow k = \log_2(n) \quad | \quad A \log B = B \log A$$

$4 \log_2(n) = n \log_2(4) = n^2$

$$T(n) = \mathcal{O}(n^2) + \mathcal{O}(n^2) = \mathcal{O}(n^2)$$

Beispiel:  $17 \times 23$ :  $17 = a = 1$   $b = 7$   
 $23 = c = 2$   $d = 3$

$$\Rightarrow 1 \cdot 2 \cdot 100 + (1 \cdot 3 + 2 \cdot 7) \cdot 10 + 7 \cdot 3$$

$$= 391$$

bessere Idee:

$$u = a \cdot c$$

$$v = b \cdot d$$

$$w = (a+b)(c+d)$$

$$(a+d + b+c) = w - u - v$$

$$\Rightarrow T(n) = O(n^{\log(3)})$$

Exponentiation:  $(x^n, n \in \mathbb{N}, x \in \mathbb{R})$

z.B.:  $x^{23} = x^{16} \cdot x^4 \cdot x^2 \cdot x^1$

$$T(n) = O(\log(n))$$

```
function EXP(x,n): real
e := 1;
while n > 0 do
  if (ungerade(n)) then
    e := shift Right(n)
    x := x * x
  end while
EXP := e
```

ungerade Zahl:

division mit 2  $\neq 0$

Ist die Methode optimal?

z.B.:  $x^{62} = x^2 \cdot x^4 \cdot x^8 \cdot x^{16} \cdot x^{32} \Rightarrow 9$  Multiplikation

$x^{62} = x^{20} \cdot x^{20} \cdot x^{20} \cdot x^2 \Rightarrow 8$  Multiplikation

asymptotisch gut, minimal nein!

O-Notationsklassen:

konstant

logarithmisch

Wurzelfunktion

linear

linearlogarithmisch

polynomial

exponential

Fakultät

$$\bullet \Theta(f(n) - g(n)) = \Theta(f(n))$$

$$\bullet \Theta(f(n) + g(n)) = \Theta(\max(f(n), g(n)))$$

$$\bullet \Theta(f(n) \cdot g(n)) =$$