



## Uitwerkingen

1. Pseudo-code voor het algoritme `maxSubArray`:

```
Algorithm maxSubArray(A, low, high):  
  if low = high then  
    return (low, high, A[low])  
  else  
    mid := [(low + high)/2]  
    (Llow, Lhigh, Lsum) := maxSubArray(A, low, mid)  
    (Rlow, Rhigh, Rsum) := maxSubArray(A, mid + 1, high)  
    (Clow, Chigh, Csum) := middenMaxSubArray(A, low, mid, high)  
    if Lsum ≥ Rsum and Lsum ≥ Csum then  
      return (Llow, Lhigh, Lsum)  
    else if Rsum ≥ Lsum and Rsum ≥ Csum then  
      return (Rlow, Rhigh, Rsum)  
    else  
      return (Clow, Chigh, Csum)
```

2. Als het input-array alleen negatieve getallen bevat, retourneert het algoritme `maxSubArray`  $(i, i, A[i])$  met  $A[i]$  het grootste getal uit  $A$ .
3. Geef pseudo-code voor het algoritme `maxSubArray` met de brute-force methode in  $\Theta(n^2)$ .  
Input: een array  $A$  bestaande uit integers.

```
Algorithm maxSubArrayBF(A, low, high):  
  m := 0  
  for i := low to high do  
    s := 0  
    for j := i to high do  
      s := s + A[j]  
      if s > m then  
        m := s  
  return m
```

4. Laat zien dat  $T(n) = T(n-1) + n \in \mathcal{O}(n^2)$ . Hier en bij de volgende recurrente betrekkingen nemen we  $T(1) = 1$ .

$$\begin{aligned}
 T(n) &= T(n-1) + n \\
 &= T(n-2) + (n-1) + n \\
 &= T(n-3) + (n-2) + (n-1) + n \\
 &= T(n-4) + (n-3) + (n-2) + (n-1) + n \\
 &= \dots \\
 &= T(n-i) + (n-i+1) + \dots + (n-1) + n
 \end{aligned}$$

Substitueer  $i = n - 1$ . Dat levert:

$$\begin{aligned}
 T(1) + 2 + \dots + (n-1) + n &= \\
 1 + 2 + \dots + (n-1) + n &= \\
 &= \frac{1}{2}n(n-1)
 \end{aligned}$$

Dus  $T(n) \in \mathcal{O}(n^2)$ .

5. Laat zien dat  $T(n) = T(\lceil \frac{n}{2} \rceil) + 1 \in \mathcal{O}(\log(n))$ .

Eerst voor  $n$  een 2-macht:

$$\begin{aligned}
 T(n) &= T\left(\frac{n}{2}\right) + 1 \\
 &= T\left(\frac{n}{4}\right) + 2 \\
 &= T\left(\frac{n}{8}\right) + 3 \\
 &= \dots \\
 &= T\left(\frac{n}{i}\right) + i
 \end{aligned}$$

Substitueer  $i = \log(n)$ . Dat levert  $T(n) = 1 + \log(n)$ . We willen nu laten zien:  $T(n) \in \mathcal{O}(\log(n))$ . We willen dus laten zien:  $T(n) \leq c \log(n)$  vanaf zekere  $n_0$ , voor zekere  $c$ . Stel dat deze grens geldt voor alle  $m < n$ , in het bijzonder voor  $m = \lceil \frac{n}{2} \rceil$ . Dan:

$$\begin{aligned}
 T(n) &= T\left(\lceil \frac{n}{2} \rceil\right) + 1 \\
 &\leq c \log\left(\lceil \frac{n}{2} \rceil\right) + 1 \\
 &\leq c \log\left(\frac{2n}{3}\right) + 1 \\
 &= c \log(n) - c \log\left(\frac{3}{2}\right) + 1 \\
 &\leq c \log(n)
 \end{aligned}$$

6. Laat zien dat  $T(n) = 4T\left(\frac{n}{2}\right) + n \in \mathcal{O}(n^2)$ :

$$\begin{aligned}
 T(n) &= 4 \cdot T\left(\frac{n}{2}\right) + n \\
 &= 4 \cdot \left(4 \cdot T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \\
 &= 16 \cdot T\left(\frac{n}{4}\right) + 3 \cdot n \\
 &= 64 \cdot T\left(\frac{n}{8}\right) + 7 \cdot n \\
 &= \dots \\
 &= 2^{(2^i)} \cdot T\left(\frac{n}{2^i}\right) + (2^i - 1) \cdot n
 \end{aligned}$$

Substitueer  $n = 2^i$  oftewel  $i = \log n$ . Dat levert:

$$2^{2 \log n} \cdot 1 + (n - 1) \cdot n = 2n^2 - n$$

Het bijbehorende algoritme is in  $\mathcal{O}(n^2)$ .

7. Laat zien dat  $T(n) = 3T(\frac{n}{2}) + n \in \mathcal{O}(n^{\log(3)})$ .

Nog toevoegen (niet geheel representatief).

8. Laat zien dat  $T(n) = 8T(\frac{n}{2}) + n^2 \in \mathcal{O}(n^3)$ .

$$\begin{aligned} T(n) &= 8T(\frac{n}{2}) + n^2 \\ &= 8^2T(\frac{n}{4}) + \frac{n^2}{4} + n^2 \\ &= 8^3T(\frac{n}{8}) + \frac{n^2}{16} + \frac{n^2}{4} + n^2 \\ &= 2^{3i}T(\frac{n}{2^i}) + n^2 \sum_{j=0}^i \frac{1}{4} \\ &\leq 2^{3i}T(\frac{n}{2^i}) + 2n^2 \end{aligned}$$

Substitueer  $i = \log(n)$ , dat levert  $T(n) = n^3 + 2n^2$ . Dus het bijbehorende algoritme is in  $\mathcal{O}(n^3)$ .

9. Laat zien dat  $T(n) = 7T(\frac{n}{2}) \in \mathcal{O}(n^{\log(7)})$ .

Nog toevoegen (niet geheel representatief).

10. (Dit is ongeveer opgave R-5.5.)

We berekenen  $A \cdot B$  met  $A = 10110011$  en  $B = 10111010$ . De deelresultaten die we elk twee keer gebruiken zijn:

$$\begin{aligned} A_h \cdot B_h &= 1011 \cdot 1011 = 1111001 \\ A_l \cdot B_l &= 0011 \cdot 1010 = 11110 \end{aligned}$$

(In decimale termen:  $11 \cdot 11 = 121$  en  $3 \cdot 10 = 30$ .) Dan:

$$\begin{aligned} A \cdot B &= 1011 \cdot 1011 \cdot 2^8 \\ &\quad + (1011 - 0011) \cdot (1010 - 1011) \cdot 2^4 \\ &\quad + 1011 \cdot 1011 \cdot 2^4 \\ &\quad + 0011 \cdot 1010 \cdot 2^4 \\ &\quad + 0011 \cdot 1010 \\ &= 1111001 \cdot 2^8 \\ &\quad + (-1000) \cdot 2^4 \\ &\quad + 1111001 \cdot 2^4 \\ &\quad + 11110 \cdot 2^4 \\ &\quad + 11110 \\ &= 1111001 \cdot 2^8 \\ &\quad + 10010110 \cdot 2^4 \\ &\quad + 11110 \\ &= 1000001000001110 \end{aligned}$$

De laatste optelling is:

$$\begin{array}{r} 111100100000000 \\ 100011110000 \\ 11110 \\ \hline 1000001000001110 \end{array}$$

11. Pas de methode van Strassen toe om het product te berekenen van

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 2 \\ 4 & 2 \end{pmatrix}$$

$$\begin{aligned} S_1 &= 0 \\ S_2 &= 8 \\ S_3 &= 72 \\ S_4 &= -10 \\ S_5 &= 48 \\ S_6 &= -12 \\ S_7 &= -48 \end{aligned}$$

en dan dus

$$\begin{aligned} I &= 18 \\ J &= 8 \\ K &= 62 \\ L &= 24 \end{aligned}$$

12. We sorteren de klanten met  $t_i$  niet-dalend. In het voorbeeld dus 312 met totale tijd 29.