

Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

Del-og-kombiner

[CLRS, kapitel 2.3.1-2.3.2, 4.1-4.3, 28.2, problem 30.1.c]



AARHUS UNIVERSITET

Del-og-Kombiner

Algoritme design teknik – virker for mange problemer (man langt fra alle):

- **Opdel** et problem P i mindre problemer P_1, \dots, P_k , der kan løses uafhængigt (små problemer løses direkte)
- Løs delproblemerne P_1, \dots, P_k **rekursivt**
- **Kombiner** løsningerne for P_1, \dots, P_k til en løsning for P

Eksempel: Merge-Sort

MERGE-SORT(A, p, r)

if $p < r$

then $q \leftarrow \lfloor (p + r) / 2 \rfloor$

MERGE-SORT(A, p, q)

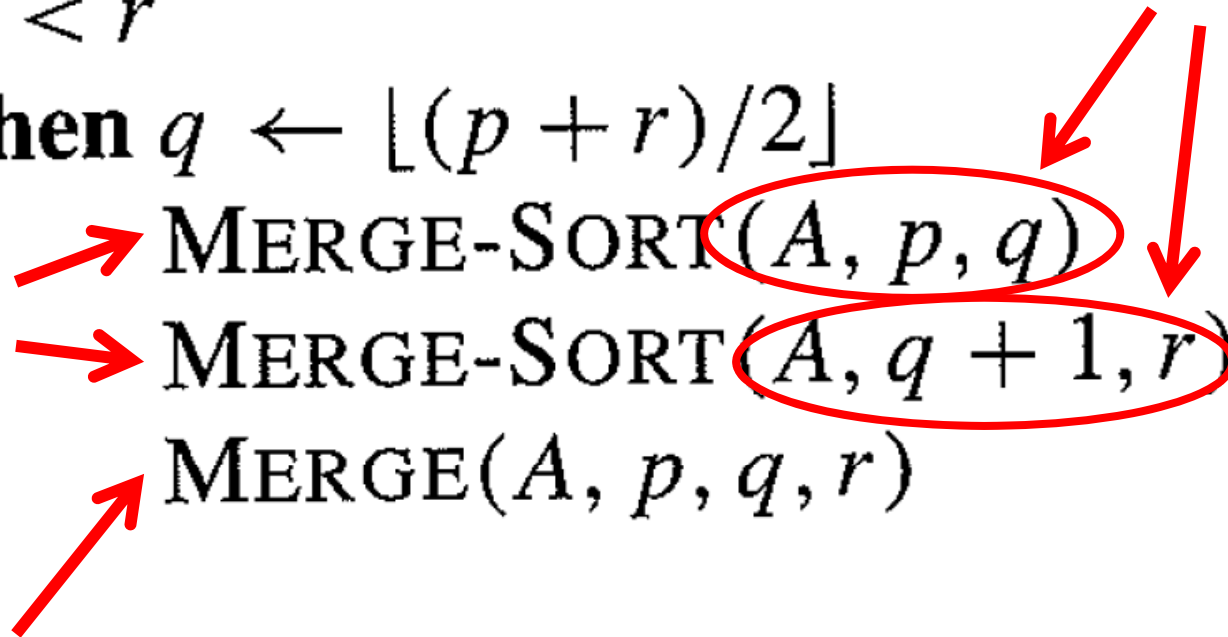
MERGE-SORT($A, q + 1, r$)

MERGE(A, p, q, r)

① To mindre delproblemer

② Løs rekursivt

③ Kombiner

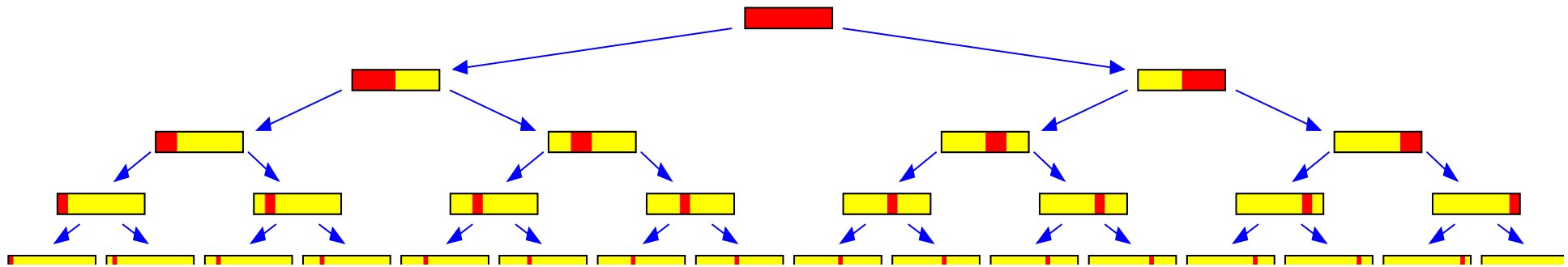


MERGE(A, p, q, r)

```
1   $n_1 \leftarrow q - p + 1$ 
2   $n_2 \leftarrow r - q$ 
3  create arrays  $L[1..n_1 + 1]$  and  $R[1..n_2 + 1]$ 
4  for  $i \leftarrow 1$  to  $n_1$ 
5      do  $L[i] \leftarrow A[p + i - 1]$ 
6  for  $j \leftarrow 1$  to  $n_2$ 
7      do  $R[j] \leftarrow A[q + j]$ 
8   $L[n_1 + 1] \leftarrow \infty$ 
9   $R[n_2 + 1] \leftarrow \infty$ 
10  $i \leftarrow 1$ 
11  $j \leftarrow 1$ 
12 for  $k \leftarrow p$  to  $r$ 
13     do if  $L[i] \leq R[j]$ 
14         then  $A[k] \leftarrow L[i]$ 
15              $i \leftarrow i + 1$ 
16         else  $A[k] \leftarrow R[j]$ 
17              $j \leftarrow j + 1$ 
```

Merge-Sort : Analyse

Rekursionstræet



Observation

Samlet arbejde per lag er $O(n)$

Arbejde

$$O(n \cdot \# \text{ lag}) = O(n \cdot \log_2 n)$$

Del-og-kombiner, dADS 1 eksempler:

- **MergeSort**
 - Del op i to lige store dele
 - Rekursiv sortering
 - Kombiner = fletning
- **QuickSort**
 - Opdel efter tilfældigt pivot (**tilfældig opdeling**)
 - Rekursive sortering
 - Kombiner = ingen (konkatener venstre og højre)
- **QuickSelect**
 - Opdel efter tilfældigt pivot (**tilfældig opdeling**)
 - Rekursiv select
 - Kombiner = ingen

Analyse af Del-og-Kombiner

= analyse af en rekursiv procedure

Essentielt to forskellige måder:

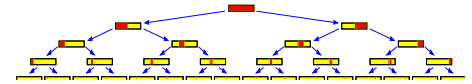
1. Argumenter direkte om **rekursionstræet** (analyser dybde, #knuder på hvert niveau, arbejde i knuderne/niveauerne/træet)
2. Løs en matematisk **rekursionsligning**, f.eks.

$T(n) \leq a$	hvis $n \leq c$
$T(n) \leq 2 \cdot T(n/2) + a \cdot n$	ellers

Bevises f.eks. vha. induktion.

Løsning af rekursionsligninger

- Fold rekursionsligningen ud og argumenter om **rekursionstræet**
- Gæt en løsning og vis den ved induktion efter voksende n



$$T(n) \leq a$$

hvis $n \leq c$

$$T(n) \leq 2 \cdot T(n/2) + a \cdot n$$

ellers

Rekursionsligninger: Faldgrubber

- Ulige opdelinger glemmes (n ulige, så er de rekursive kald typisk $\lfloor n/2 \rfloor$ og $\lceil n/2 \rceil$)
- Analyserer typiske kun for $n=2^k$
- Brug aldrig O-udtryk i rekursionsformlen – brug konstanter (~~$T(n) = O(n) + O(T(n/3))$~~)

Master Theorem

(Simplificering af [CLRS, Theorem 4.1])

Theorem

Hvis a, b, c, d, p er konstanter som opfylder $a, c, p > 0$, $d \geq 1$, og $b > 1$, så har rekursionsligningen

$$T(n) = \begin{cases} c & \text{hvis } n \leq d \\ a \cdot T(n/b) + c \cdot n^p & \text{hvis } n > d \end{cases}$$

følgende løsning

$$\begin{array}{ll} O(n^p) & \text{hvis } a < b^p \\ O(n^p \log n) & \text{hvis } a = b^p \\ O(n^{\log_b a}) & \text{hvis } a > b^p \end{array}$$

Matrix Multiplication

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1..m} a_{ik} \cdot b_{kj}$$

Naive implementation: tid $O(npm)$

(Kvadratisk) Matrix Multiplikation

$$\begin{pmatrix} I & J \\ K & L \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

$$I = AE + BG$$

$$J = AF + BH$$

$$K = CE + DG$$

$$L = CF + DH$$

- A, B, \dots, K, L er $n/2 \times n/2$ -matricer
- I, J, K, L kan beregnes med 8 rekursive multiplication på $n/2 \times n/2$ -matricer + 4 matrix additioner
- $T(n) \leq 8 \cdot T(n/2) + c \cdot n^2$ for $n \geq 2$
 $T(n) \leq c$ for $n = 1$
- $T(n) = O(n^3)$

Strassen's Matrix Multiplikation

1969

$$\begin{pmatrix} I & J \\ K & L \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

$$\begin{aligned} I &= S_5 + S_6 + S_4 - S_2 \\ &= (A + D)(E + H) + (B - D)(G + H) + D(G - E) - (A + B)H \\ &= AE + DE + AH + DH + BG - DG + BH - DH + DG - DE - AH - BH \\ &= AE + BG. \end{aligned}$$

$$\begin{aligned} J &= S_1 + S_2 \\ &= A(F - H) + (A + B)H \\ &= AF - AH + AH + BH \\ &= AF + BH. \end{aligned}$$

$$\begin{aligned} K &= S_3 + S_4 \\ &= (C + D)E + D(G - E) \\ &= CE + DE + DG - DE \\ &= CE + DG. \end{aligned}$$

$$\begin{aligned} L &= S_1 - S_7 - S_3 + S_5 \\ &= A(F - H) - (A - C)(E + F) - (C + D)E + (A + D)(E + H) \\ &= AF - AH - AE + CE - AF + CF - CE - DE + AE + DE + AH + DH \\ &= CF + DH. \end{aligned}$$

S_1	$=$	$A(F - H)$
S_2	$=$	$(A + B)H$
S_3	$=$	$(C + D)E$
S_4	$=$	$D(G - E)$
S_5	$=$	$(A + D)(E + H)$
S_6	$=$	$(B - D)(G + H)$
S_7	$=$	$(A - C)(E + F)$

7 rekursive multiplikationen

Strassen's Matrix Multiplikation

- Bruger 18 matrix additioner (tid $O(n^2)$) og 7 rekursive matrix multiplikationer

$$T(n) \leq 7 \cdot T(n/2) + c \cdot n^2 \quad \text{for } n \geq 2$$

$$T(n) \leq c \quad \text{for } n = 1$$

- $T(n) = O(n^{2.81})$ hvor $2.81 = \log_2 7$

Multiplikation af lange heltal

- I og J hver heltal med n bits
- Naive implementation kræver $O(n^2)$ bit operationer
- Lad $I = I_h \cdot 2^{n/2} + I_l$ og $J = J_h \cdot 2^{n/2} + J_l$
- $I \cdot J = I_h \cdot J_h \cdot 2^n + ((I_h - I_l) \cdot (J_l - J_h) + I_l \cdot J_l + I_h \cdot J_h) \cdot 2^{n/2} + I_l \cdot J_l$

$$T(n) \leq 3 \cdot T(n/2) + c \cdot n \quad \text{for } n \geq 2$$

$$T(n) \leq c \quad \text{for } n = 1$$

- $T(n) = O(n^{\log_2 3})$