

# Dynamisk programmering

## Flere eksempler

# Eksempel 1: Længste fælles delstreng

Alfabet = mængde af tegn:

$\{a,b,c,\dots,z\}$ ,  $\{A,C,G,T\}$ ,  $\{0,1\}$

# Eksempel 1: Længste fælles delstreng

**Alfabet** = mængde af tegn:

$\{a,b,c,\dots,z\}$ ,  $\{A,C,G,T\}$ ,  $\{0,1\}$

**Streng** = sekvens  $x_1x_2x_3\dots x_n$  af tegn fra et alfabet:

helloworld

GATAAATCTGGTCTTATTTCC

00101100101010001111

# Eksempel 1: Længste fælles delstreng

**Alfabet** = mængde af tegn:

$\{a,b,c,\dots,z\}$ ,  $\{A,C,G,T\}$ ,  $\{0,1\}$

**Streng** = sekvens  $x_1x_2x_3\dots x_n$  af tegn fra et alfabet:

helloworld

GATAAATCTGGTCTTATTTCC

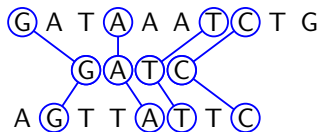
00101100101010001111

**Delstreng** = delmængde af tegnene i streng, i uændret rækkefølge:

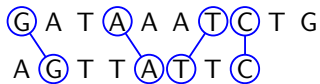


# Længste fælles delstreng

Fælles delstreng for to strenge:

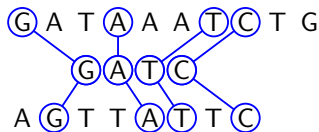


Eller:

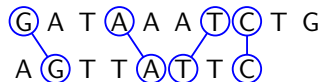


# Længste fælles delstreng

Fælles delstreng for to strenge:



Eller:



Længste fælles delstreng:

Givet to strenge

$$X = x_1 x_2 x_3 \dots x_m$$

$$Y = y_1 y_2 y_3 \dots y_n$$

af længde  $m$  og  $n$ , find en **længste** fælles delstreng for dem.

Længden af denne kan ses som et mål for similaritet mellem strenge (f.eks. dna-streng).

# Rekursiv løsning?

Vi vil arbejde på at lave en rekursiv løsning. Vi definerer derfor mindre problemstørrelser:

# Rekursiv løsning?

Vi vil arbejde på at lave en rekursiv løsning. Vi definerer derfor mindre problemstørrelser:

- ▶  $X_i = x_1x_2x_3 \dots x_i$  for  $1 \leq i \leq m$ .
- ▶  $Y_j = y_1y_2y_3 \dots y_j$  for  $1 \leq j \leq n$ .
- ▶  $X_0$  og  $Y_0$  er den tomme streng.
- ▶  $\text{lcs}(i,j)$  er *længden* af længste fælles delstreng af  $X_i$  og  $Y_j$ .



# Rekursiv løsning?

Vi vil arbejde på at lave en rekursiv løsning. Vi definerer derfor mindre problemstørrelser:

- ▶  $X_i = x_1x_2x_3 \dots x_i$  for  $1 \leq i \leq m$ .
- ▶  $Y_j = y_1y_2y_3 \dots y_j$  for  $1 \leq j \leq n$ .
- ▶  $X_0$  og  $Y_0$  er den tomme streng.
- ▶  $\text{lcs}(i,j)$  er *længden* af længste fælles delstreng af  $X_i$  og  $Y_j$ .

Vi vil gerne finde  $\text{lcs}(m, n)$ .

# Rekursiv løsning?

Vi vil arbejde på at lave en rekursiv løsning. Vi definerer derfor mindre problemstørrelser:

- ▶  $X_i = x_1x_2x_3 \dots x_i$  for  $1 \leq i \leq m$ .
- ▶  $Y_j = y_1y_2y_3 \dots y_j$  for  $1 \leq j \leq n$ .
- ▶  $X_0$  og  $Y_0$  er den tomme streng.
- ▶  $\text{lcs}(i,j)$  er *længden* af længste fælles delstreng af  $X_i$  og  $Y_j$ .

Vi vil gerne finde  $\text{lcs}(m, n)$ .

Mere generelt: Vi søger en rekursiv formel for  $\text{lcs}(i, j)$ .

# Rekursiv løsning?

Vi vil arbejde på at lave en rekursiv løsning. Vi definerer derfor mindre problemstørrelser:

- ▶  $X_i = x_1x_2x_3 \dots x_i$  for  $1 \leq i \leq m$ .
- ▶  $Y_j = y_1y_2y_3 \dots y_j$  for  $1 \leq j \leq n$ .
- ▶  $X_0$  og  $Y_0$  er den tomme streng.
- ▶  $\text{lcs}(i,j)$  er *længden* af længste fælles delstreng af  $X_i$  og  $Y_j$ .

Vi vil gerne finde  $\text{lcs}(m, n)$ .

Mere generelt: Vi søger en rekursiv formel for  $\text{lcs}(i, j)$ .

Basistilfælde: Det er klart at  $\text{lcs}(0, j) = \text{lcs}(i, 0) = 0$ .

# Optimale delproblemer I

Formel for  $\text{lcs}(i, j)$ :

Case I:  $x_i = y_j$

Observation: en fælles delstreng  $Z$  for  $X_i$  og  $Y_j$  består af

- ▶ Et sidste tegn  $z_k$ .
- ▶ En streng  $Z' = z_1 z_2 z_3 \dots z_{k-1}$ , som må være en fælles delstreng af  $X_{i-1}$  og  $Y_{j-1}$  (tegnene i  $Z$  skal komme i samme rækkefølge som i  $X$  og  $Y$ , så kun sidste tegn i  $Z$  har mulighed for at være  $x_i$  og  $y_j$ ).

# Optimale delproblemer I

Formel for  $\text{lcs}(i, j)$ :

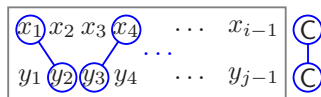
Case I:  $x_i = y_j$

Observation: en fælles delstreng  $Z$  for  $X_i$  og  $Y_j$  består af

- ▶ Et sidste tegn  $z_k$ .
- ▶ En streng  $Z' = z_1 z_2 z_3 \dots z_{k-1}$ , som må være en fælles delstreng af  $X_{i-1}$  og  $Y_{j-1}$  (tegnene i  $Z$  skal komme i samme rækkefølge som i  $X$  og  $Y$ , så kun sidste tegn i  $Z$  har mulighed for at være  $x_i$  og  $y_j$ ).

Den essentielle egenskab (optimale delproblemer) for Case I:

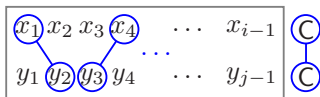
Hvis  $Z$  er en *længste* fælles delstreng for for  $X_i$  og  $Y_j$ , må  $Z'$  være en længste fælles delstreng af  $X_{i-1}$  og  $Y_{j-1}$ . For hvis der fandtes en længere fælles delstreng for  $X_{i-1}$  og  $Y_{j-1}$ , kunne den tilføjes tegnet  $x_i$  ( $= y_j$ ) og blive en længere fælles delstreng for  $X_i$  og  $Y_j$ .



# Optimale delproblemer I

Af den essentielle egenskab haves i Case I ( $x_i = y_j$ ):

- ▶  $\text{lcs}(i, j) = \text{lcs}(i - 1, j - 1) + 1$
- ▶ En længste fælles delstreng for  $X_{i-1}$  og  $Y_{j-1}$  tilføjet tegnet  $x_i (= y_j)$  er en længste fælles delstreng for  $X_i$  og  $Y_j$ .



## Optimale delproblemer II

Formel for  $\text{lcs}(i, j)$ :

Case II:  $x_i \neq y_j$

Observation: en fælles delstreng  $Z = z_1 z_2 z_3 \dots z_k$  for  $X_i$  og  $Y_j$  kan ikke have  $z_k$  værende en parring af  $x_i$  og  $y_j$  (da disse jo er forskellige).

Så  $Z$  må være en fælles delstreng for *enten*  $X_{i-1}$  og  $Y_j$  *eller* for  $X_i$  og  $Y_{j-1}$  (eller evt. begge).

## Optimale delproblemer II

Formel for  $\text{lcs}(i, j)$ :

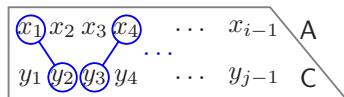
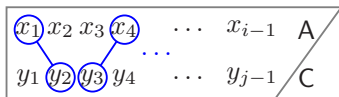
Case II:  $x_i \neq y_j$

Observation: en fælles delstreng  $Z = z_1 z_2 z_3 \dots z_k$  for  $X_i$  og  $Y_j$  kan ikke have  $z_k$  værende en parring af  $x_i$  og  $y_j$  (da disse jo er forskellige).

Så  $Z$  må være en fælles delstreng for *enten*  $X_{i-1}$  og  $Y_j$  *eller* for  $X_i$  og  $Y_{j-1}$  (eller evt. begge).

Den essentielle egenskab (optimale delproblemer) for Case II:

Hvis  $Z$  er en *længste* fælles delstreng for  $X_i$  og  $Y_j$ , må den være en længste fælles delstreng for enten  $X_{i-1}$  og  $Y_j$  eller for  $X_i$  og  $Y_{j-1}$  (eller evt. begge). For hvis der fandtes en længere fælles delstreng for enten  $X_{i-1}$  og  $Y_j$  eller for  $X_i$  og  $Y_{j-1}$ , ville denne også være en længere fælles delstreng for  $X_i$  og  $Y_j$ .





## Optimale delproblemer II

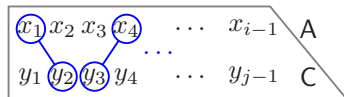
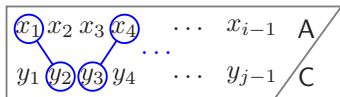
Lad  $T_1$  være en længste fælles delstreng for  $X_{i-1}$  og  $Y_j$ , og lad  $T_2$  være en længste fælles delstreng for  $X_i$  og  $Y_{j-1}$ .

Af den essentielle egenskab i Case II ( $x_i \neq y_j$ ) haves at blandt  $T_1$  og  $T_2$  er der (mindst) en som er en længste fælles delstreng for  $X_i$  og  $Y_j$ .

Ingen af  $T_1$  og  $T_2$  kan være længere end den længste fælles delstreng for  $X_i$  og  $Y_j$  (da de begge er delstrengene af  $X_i$  og  $Y_j$ ).

Så af den essentielle egenskab haves i Case II ( $x_i \neq y_j$ ):

- ▶  $\text{lcs}(i, j) = \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1))$
- ▶ Hvis  $\text{lcs}(i-1, j) \geq \text{lcs}(i, j-1)$ , er en længste fælles delstreng for  $X_{i-1}$  og  $Y_j$  også en længste fælles delstreng for  $X_i$  og  $Y_j$ . Et symmetrisk udsagn gælder for " $\leq$ " og  $X_i$  og  $Y_{j-1}$ .



## Rekursiv formel for lcs(i, j)

Alt i alt har vi fundet flg. rekursive formel for lcs(i, j):

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

## Rekursiv formel for lcs(i, j)

Alt i alt har vi fundet flg. rekursive formel for lcs(i, j):

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Den giver anledning til en naturlig, simpel rekursiv algoritme.

## Rekursiv formel for lcs(i, j)

Alt i alt har vi fundet flg. rekursive formel for lcs(i, j):

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Den giver anledning til en naturlig, simpel rekursiv algoritme.

MEN: det er nemt at se at der er gentagelser blandt delproblemers delproblemer.

Så samme delproblemer bliver gentagne gange beregnet forskellige steder i rekursionstræet, og køretiden bliver meget dårlig.

## Rekursiv formel for $\text{lcs}(i, j)$

Alt i alt har vi fundet flg. rekursive formel for  $\text{lcs}(i, j)$ :

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Den giver anledning til en naturlig, simpel rekursiv algoritme.

MEN: det er nemt at se at der er gentagelser blandt delproblemers delproblemer.

Så samme delproblemer bliver gentagne gange beregnet forskellige steder i rekursionstræet, og køretiden bliver meget dårlig.

Kan evt. løses med memoization: hav en tabel med plads til svaret på alle de mulige delproblemer  $\text{lcs}(i, j)$ , og gem svaret når det er beregnet første gang. Siden, slå det bare op.

## Rekursiv formel for $\text{lcs}(i, j)$

Alt i alt har vi fundet flg. rekursive formel for  $\text{lcs}(i, j)$ :

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Den giver anledning til en naturlig, simpel rekursiv algoritme.

MEN: det er nemt at se at der er gentagelser blandt delproblemers delproblemer.

Så samme delproblemer bliver gentagne gange beregnet forskellige steder i rekursionstræet, og køretiden bliver meget dårlig.

Kan evt. løses med memoization: hav en tabel med plads til svaret på alle de mulige delproblemer  $\text{lcs}(i, j)$ , og gem svaret når det er beregnet første gang. Siden, slå det bare op.

Dynamisk programmering: udfyld i stedet direkte denne tabel bottom-up på struktureret måde.

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0							
1							
2							
·							
·							
$m$							

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						


$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$



# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						



$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i,j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						

$$\text{lcs}(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i,j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						

$$\text{lcs}(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						

The table illustrates the bottom-up construction of the LCS table. The first row (i=0) and the first column (j=0) are shaded blue and contain the value 0. A red arrow points from cell (2,2) to (1,1), indicating a diagonal transition. Blue arrows point from cell (2,2) to (2,1) and (1,2), indicating transitions from the left and top cells.

$$\text{lcs}(i, j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i - 1, j - 1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i - 1, j), \text{lcs}(i, j - 1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Dynamisk programmering

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i,j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						


The table illustrates the bottom-up construction of the LCS table. The top row (i=0) and the first column (j=0) are filled with 0s and shaded with blue diagonal lines. A red arrow points from the cell (2,2) to (1,1), indicating the recursive step. A blue arrow points from the cell (2,2) to (3,3), indicating the next step in the bottom-up process.

$$\text{lcs}(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \text{lcs}(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(\text{lcs}(i-1, j), \text{lcs}(i, j-1)) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

# Køretid

Dynamisk programmering: udfyld tabel over  $\text{lcs}(i, j)$  bottom-up på struktureret måde.

$i \backslash j$	0	1	2	·	·	·	$n$
0	0	0	0	0	0	0	0
1	0						
2	0						
·	0						
·	0						
$m$	0						



Tabelstørrelse:  $mn$

Udfyld tabelindgang:  $O(\text{max størrelse af røde graf}) = O(1)$ .

Tid i alt:  $O(\text{produktet af de to}) = O(mn)$ .

## Find en konkret løsning

$lcs(m, n)$  er *længden* af en længste fælles delstreng for  $X = X_m$  og  $Y = Y_n$ .

Hvis vi gerne vil finde en konkret fælles delstreng af denne længde: Gem for hvert felt i tabellen hvilken af de tre røde pile som gav  $lcs(i, j)$ -værdien i dette felt.

		$j$	0	1	2	3	4	5	6
$i$	$y_j$		<b>B</b>	D	<b>C</b>	A	<b>B</b>	<b>A</b>	
	$x_i$		0	0	0	0	0	0	0
0			0	0	0	0	0	0	0
1	A		0	↑	↑	↑	↖	←	↖
2	<b>B</b>		0	↑	↖	←	↑	↖	←
3	<b>C</b>		0	↑	↑	↖	←	↑	↑
4	<b>B</b>		0	↖	↑	↑	↑	↖	←
5	D		0	↑	↖	↑	↑	↑	↑
6	<b>A</b>		0	↑	↑	↑	↖	↑	↖
7	B		0	↖	↑	↑	↑	↖	↑

Følg gemte pile baglæns fra  $lcs(m, n)$ . Når en skrå pil følges er det en Case I, og  $x_i$  ( $=y_j$ ) udskrives. Ellers er den en Case II, og intet udskrives.

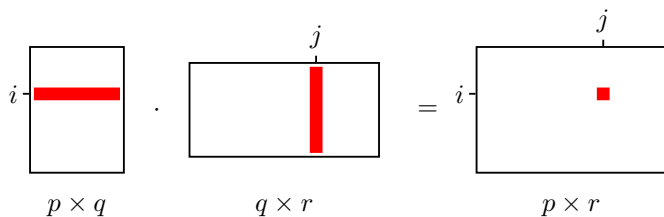
I alt udskrives en længste fælles delstreng for  $X$  og  $Y$  i baglæns orden.



## Eksempel 2: Multi-Matrix-multiplikation

Fra DM527/MM524:

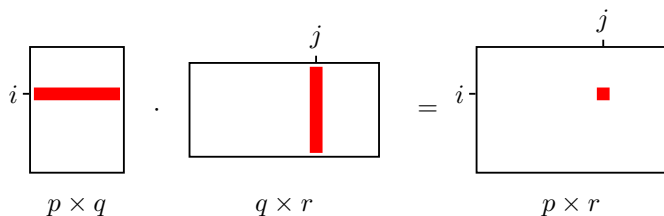
En  $p \times q$  matrix  $A_1$  og en  $q \times r$  matrix  $A_2$  kan multipliceres i tid  $O(pqr)$ .  
Resultatet er en  $p \times r$  matrix.



## Eksempel 2: Multi-Matrix-multiplikation

Fra DM527/MM524:

En  $p \times q$  matrix  $A_1$  og en  $q \times r$  matrix  $A_2$  kan multipliceres i tid  $O(pqr)$ .  
Resultatet er en  $p \times r$  matrix.



Matrix-multiplikation er associativ:

$$A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$$

# Multi-Matrix-multiplikation

Matrix-multiplikation er associativ:

$$A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$$

Men køretiden er IKKE ens.

# Multi-Matrix-multiplikation

Matrix-multiplikation er associativ:

$$A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$$

Men køretiden er IKKE ens. Eksempel:

$$\begin{array}{ccc} A_1 & A_2 & A_3 \\ 10 \times 100 & 100 \times 5 & 5 \times 50 \\ & & 100 \times 50 \\ & 10 \times 5 & \end{array}$$

# Multi-Matrix-multiplikation

Matrix-multiplikation er associativ:

$$A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$$

Men køretiden er IKKE ens. Eksempel:

$$\begin{array}{ccc} A_1 & A_2 & A_3 \\ 10 \times 100 & 100 \times 5 & 5 \times 50 \\ & & 100 \times 50 \\ & & 10 \times 5 \end{array}$$

Tid for  $A_1 \cdot (A_2 \cdot A_3)$ : er  $10 \cdot 100 \cdot 50 + 100 \cdot 5 \cdot 50 = 75.000$

# Multi-Matrix-multiplikation

Matrix-multiplikation er associativ:

$$A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$$

Men køretiden er IKKE ens. Eksempel:

$$\begin{array}{ccc} A_1 & A_2 & A_3 \\ 10 \times 100 & 100 \times 5 & 5 \times 50 \\ & & 100 \times 5 \\ & & 10 \times 5 \end{array}$$

Tid for  $A_1 \cdot (A_2 \cdot A_3)$ : er  $10 \cdot 100 \cdot 50 + 100 \cdot 5 \cdot 50 = 75.000$

Tid for  $(A_1 \cdot A_2) \cdot A_3$ : er  $10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7.500$

# Multi-Matrix-multiplikation

Spørgsmålet:

Givet et produkt af  $n$  matricer

$$A_1, A_2, A_3, \dots, A_n$$

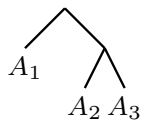
med kompatible dimensioner

$$p_0 \times p_1, p_1 \times p_2, p_2 \times p_3, \dots, p_{n-1} \times p_n$$

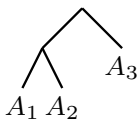
hvad er den billigste rækkefølge at gange dem sammen i?

# Beregningstræer

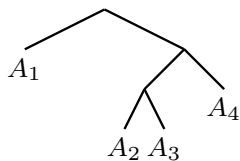
Rækkefølge = parentessætning = binært beregningstræ:



$A_1(A_2A_3)$



$(A_1A_2)A_3$



$A_1((A_2A_3)A_4)$

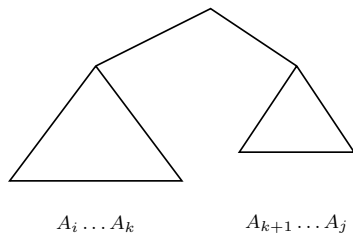


# Optimale delproblemer og rekursiv ligning

Lad  $m(i, j)$  være prisen for bedste måde at gange  $A_i, \dots, A_j$  sammen på.

Observation af den essentielle egenskab:

Undertræerne for roden af et optimalt træ må selv være optimale beregningstræer.



Prøv alle placeringer af rod, dvs. alle split  $A_i, \dots, A_k$  og  $A_{k+1}, \dots, A_j$ :

$$m(i, j) = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j} \{m(i, k) + m(k + 1, j) + p_{i-1}p_kp_j\} & \text{if } i < j \end{cases}$$

# Tabel

Gentagelser blandt delproblemers delproblemer. Lav tabel og udfyld systematisk. Målet er at kende  $m(1, n)$ .

# Tabel

Gentagelser blandt delproblemers delproblemer. Lav tabel og udfyld systematisk. Målet er at kende  $m(1, n)$ .

$i \backslash j$	1	2	3	·	·	·	$n$
1	0						
2		0					
3			0				
·				0			
·					0		
·						0	
$n$							0

# Tabel

Gentagelser blandt delproblemers delproblemer. Lav tabel og udfyld systematisk. Målet er at kende  $m(1, n)$ .

$i \backslash j$	1	2	3	·	·	·	$n$
1	0						
2		0					
3			0				
·				0			
·					0		
·						0	
$n$							0

The diagram shows a table with rows and columns indexed from 1 to n. The diagonal cells (where i=j) contain the value 0. The upper triangular region of the table is shaded with blue diagonal lines. A red path starts at (1,1) and moves to (2,2), then to (3,3), then to (4,4), then to (5,5), then to (6,6), and finally to (n,n). Blue arrows indicate the direction of the path: from (1,1) to (2,2), from (2,2) to (3,3), from (3,3) to (4,4), from (4,4) to (5,5), from (5,5) to (6,6), and from (6,6) to (n,n). There are also blue arrows pointing from (2,2) to (2,3), (2,4), (2,5), and (2,6), and from (3,3) to (3,4), (3,5), and (3,6).

Tabelstørrelse:  $n^2/2$

Udfyld tabelindgang:  $O(\text{max størrelse af røde graf}) = O(n)$ .

Tid i alt:  $O(\text{produktet af de to}) = O(n^3)$ .

# Tabel

Gentagelser blandt delproblemers delproblemer. Lav tabel og udfyld systematisk. Målet er at kende  $m(1, n)$ .

$i \backslash j$	1	2	3	·	·	·	$n$
1	0						
2		0					
3			0				
·				0			
·					0		
·						0	
$n$							0

Tabelstørrelse:  $n^2/2$

Udfyld tabelindgang:  $O(\text{max størrelse af røde graf}) = O(n)$ .

Tid i alt:  $O(\text{produktet af de to}) = O(n^3)$ .

Find konkret løsning: følg de optimale valg baglæns.