

Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

Dynamisk Programmering
[CLRS 15]

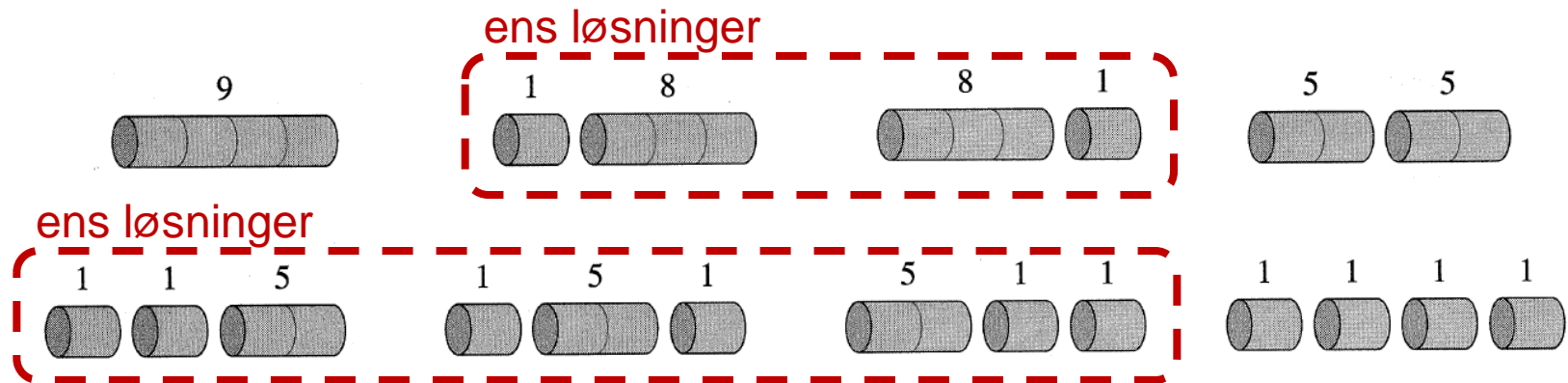


Dynamisk Programmering

- **Generel algoritmisk teknik** – virker for mange (men langt fra alle) problemer
- **Krav:** "Optimal delstruktur" – en løsning til problemet kan konstrueres ud fra optimale løsninger til "**delproblemer**"
- **Rekursive løsning:**
 - Typisk eksponentiel tid
- **Dynamisk Programmering:**
 - Beregn delløsninger **systematisk**
 - Typisk polynomiel tid

Dynamisk Programmering: Optimal opdeling af en stang

Problem: Opdel en stang i dele, hvor hver del har en pris, således at den resulterende sum er maksimereret



En stang af længde 4 kan opdeles på 8 forskellige måder – dog kun 5 forskellige resultater

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

Optimal opdeling af stænger af længde 1..10

Bedste pris

Opdeling

$r_1 = 1$	$1 = 1$ (no cuts),
$r_2 = 5$	$2 = 2$ (no cuts),
$r_3 = 8$	$3 = 3$ (no cuts),
$r_4 = 10$	$4 = 2 + 2$,
$r_5 = 13$	$5 = 2 + 3$,
$r_6 = 17$	$6 = 6$ (no cuts),
$r_7 = 18$	$7 = 1 + 6$ or $7 = 2 + 2 + 3$,
$r_8 = 22$	$8 = 2 + 6$,
$r_9 = 25$	$9 = 3 + 6$,
$r_{10} = 30$	$10 = 10$ (no cuts).

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

Optimal opdeling af en stang: Rekursiv løsning

$$r_n = \begin{cases} 0 & \text{hvis } n = 0 \\ \max_{i=1..n} (p_i + r_{n-i}) & \text{ellers} \end{cases}$$

CUT-ROD(p, n)

```
1  if  $n == 0$ 
2      return 0
3   $q = -\infty$ 
4  for  $i = 1$  to  $n$ 
5       $q = \max(q, p[i] + \text{CUT-ROD}(p, n - i))$ 
6  return  $q$ 
```

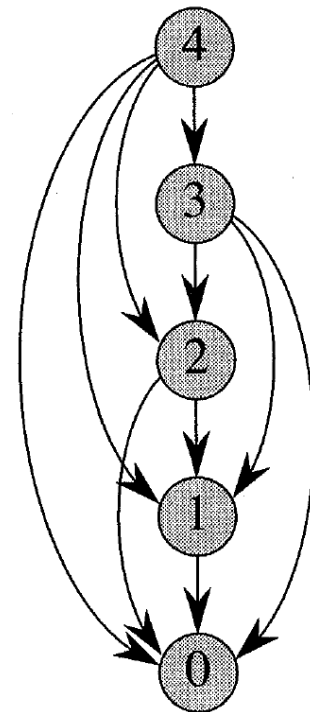

Optimal opdeling af en stang: Rekursiv løsning + Memoization

MEMOIZED-CUT-ROD(p, n)

```
1 let  $r[0..n]$  be a new array
2 for  $i = 0$  to  $n$ 
3    $r[i] = -\infty$ 
4 return MEMOIZED-CUT-ROD-AUX( $p, n, r$ )
```

MEMOIZED-CUT-ROD-AUX(p, n, r)

```
1 if  $r[n] \geq 0$ 
2   return  $r[n]$ 
3 if  $n == 0$ 
4    $q = 0$ 
5 else  $q = -\infty$ 
6   for  $i = 1$  to  $n$ 
7      $q = \max(q, p[i] + \text{MEMOIZED-CUT-ROD-AUX}(p, n - i, r))$ 
8  $r[n] = q$  ← husk resultatet !
9 return  $q$ 
```

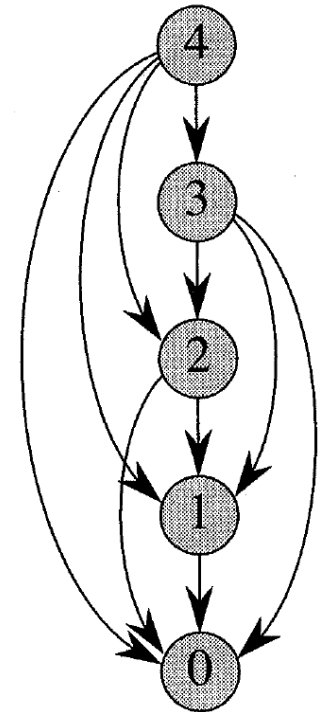


Tid $O(n^2)$

Optimal opdeling af en stang: Systematisk udfyldning

BOTTOM-UP-CUT-ROD(p, n)

```
1  let  $r[0..n]$  be a new array
2   $r[0] = 0$ 
3  for  $j = 1$  to  $n$  ← rækkefølgen vigtig !
4       $q = -\infty$ 
5      for  $i = 1$  to  $j$ 
6           $q = \max(q, p[i] + r[j - i])$ 
7       $r[j] = q$ 
8  return  $r[n]$ 
```



Tid $O(n^2)$

Optimal opdeling af en stang: Udskrivning af løsningen

EXTENDED-BOTTOM-UP-CUT-ROD(p, n)

```
1  let  $r[0..n]$  and  $s[0..n]$  be new arrays
2   $r[0] = 0$ 
3  for  $j = 1$  to  $n$ 
4       $q = -\infty$ 
5      for  $i = 1$  to  $j$ 
6          if  $q < p[i] + r[j - i]$ 
7               $q = p[i] + r[j - i]$ 
8               $s[j] = i$ 
9       $r[j] = q$ 
10 return  $r$  and  $s$ 
```

i	0	1	2	3	4	5	6	7	8	9	10
$r[i]$	0	1	5	8	10	13	17	18	22	25	30
$s[i]$	0	1	2	3	2	2	6	1	2	3	10

PRINT-CUT-ROD-SOLUTION(p, n)

```
1  ( $r, s$ ) = EXTENDED-BOTTOM-UP-CUT-ROD( $p, n$ )
2  while  $n > 0$ 
3      print  $s[n]$ 
4       $n = n - s[n]$ 
```

Tid $O(n^2+n)$

Matrix Multiplikation

MATRIX-MULTIPLY(A, B)

```
1  if  $A.columns \neq B.rows$ 
2      error “incompatible dimensions”
3  else let  $C$  be a new  $A.rows \times B.columns$  matrix
4      for  $i = 1$  to  $A.rows$ 
5          for  $j = 1$  to  $B.columns$ 
6               $c_{ij} = 0$ 
7              for  $k = 1$  to  $A.columns$ 
8                   $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ 
9      return  $C$ 
```

Multiplikation af to matricer A og B af størrelse

$p_1 \times p_2$ og $p_2 \times p_3$ tager tid $O(p_1 \cdot p_2 \cdot p_3)$

Matrix-kæde Multiplikation

$(A \cdot B) \cdot C$ eller $A \cdot (B \cdot C)$?

Matrix multiplikation er associativ (kan sætte paranteser som man vil) men ikke kommutative (kan ikke bytte rundt på rækkefølgen af matricerne)

Matrix-kæde Multiplikation

Problem: Find den bedste rækkefølge (paranteser) for at gange n matricer sammen

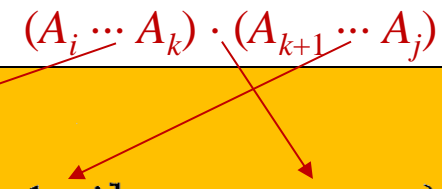
$$A_1 \cdot A_2 \cdot \dots \cdot A_n$$

hvor A_i er en $p_{i-1} \times p_i$ matrix

NB: Der er $\Omega(4^n/n^{3/2})$ mulige måder for paranteserne

Matrix-kæde Multiplikation

$m[i, j]$ = minimale antal (primitive) multiplikationer for at beregne $A_i \cdot \dots \cdot A_j$

$$m[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j} \{m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j\} & \text{if } i < j \end{cases}$$


RECURSIVE-MATRIX-CHAIN(p, i, j)

```
1  if  $i == j$ 
2      return 0
3   $m[i, j] = \infty$ 
4  for  $k = i$  to  $j - 1$ 
5       $q =$  RECURSIVE-MATRIX-CHAIN( $p, i, k$ )
          + RECURSIVE-MATRIX-CHAIN( $p, k + 1, j$ )
          +  $p_{i-1} p_k p_j$ 
6      if  $q < m[i, j]$ 
7           $m[i, j] = q$ 
8  return  $m[i, j]$ 
```

Tid $\Omega(4^n/n^{3/2})$

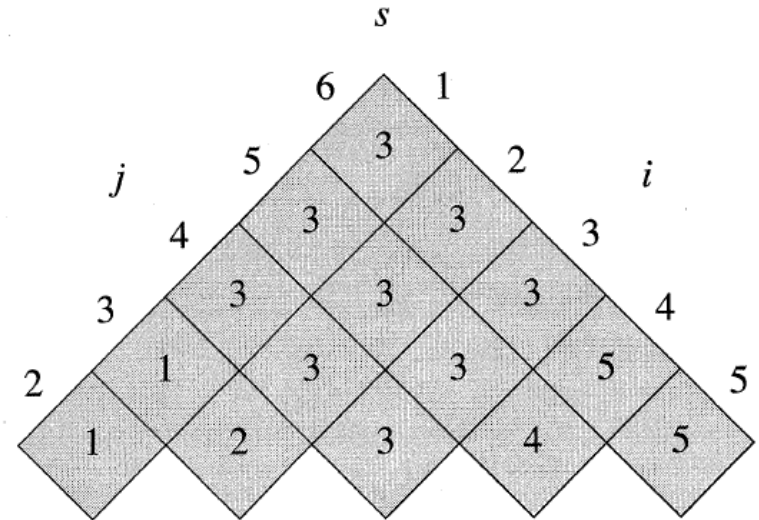
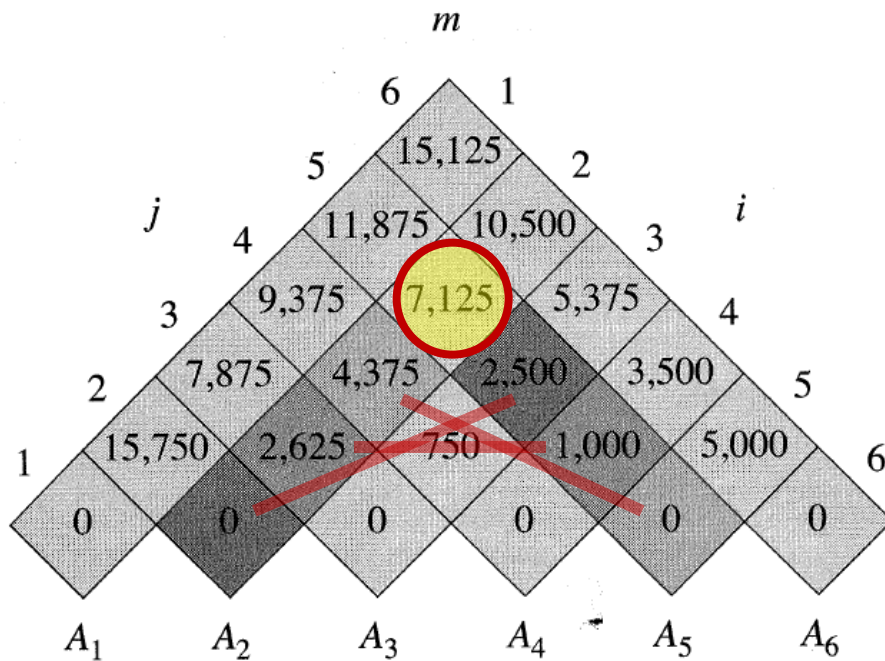
Matrix-kæde Multiplikation

MATRIX-CHAIN-ORDER(p)

```
1   $n = p.length - 1$ 
2  let  $m[1..n, 1..n]$  and  $s[1..n - 1, 2..n]$  be new tables
3  for  $i = 1$  to  $n$ 
4       $m[i, i] = 0$ 
5  for  $l = 2$  to  $n$            //  $l$  is the chain length
6      for  $i = 1$  to  $n - l + 1$ 
7           $j = i + l - 1$ 
8           $m[i, j] = \infty$ 
9          for  $k = i$  to  $j - 1$ 
10              $q = m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j$ 
11             if  $q < m[i, j]$ 
12                  $m[i, j] = q$ 
13                  $s[i, j] = k$ 
14  return  $m$  and  $s$ 
```

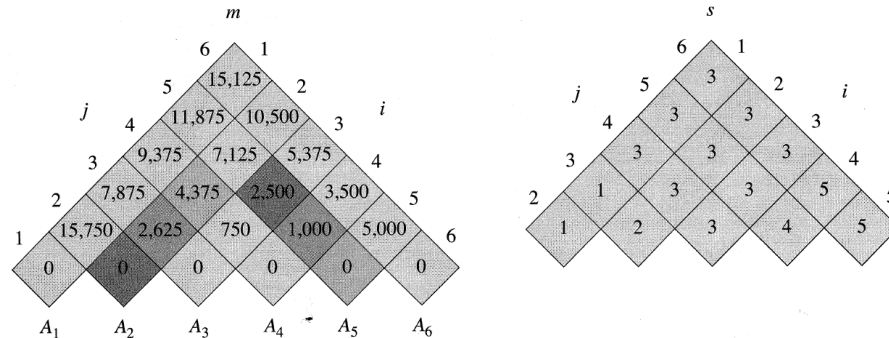
Tid $O(n^3)$

Matrix-kæde Multiplikation



matrix	A_1	A_2	A_3	A_4	A_5	A_6
dimension	30×35	35×15	15×5	5×10	10×20	20×25

Matrix-kæde Multiplikation



PRINT-OPTIMAL-PARENS (s, i, j)

- 1 **if** $i == j$
- 2 print “ A ” $_i$
- 3 **else** print “(”
- 4 PRINT-OPTIMAL-PARENS ($s, i, s[i, j]$)
- 5 PRINT-OPTIMAL-PARENS ($s, s[i, j] + 1, j$)
- 6 print “)”

Tid $O(n)$

”Memoized” Matrix-kæde Multiplikation

MEMOIZED-MATRIX-CHAIN(p)

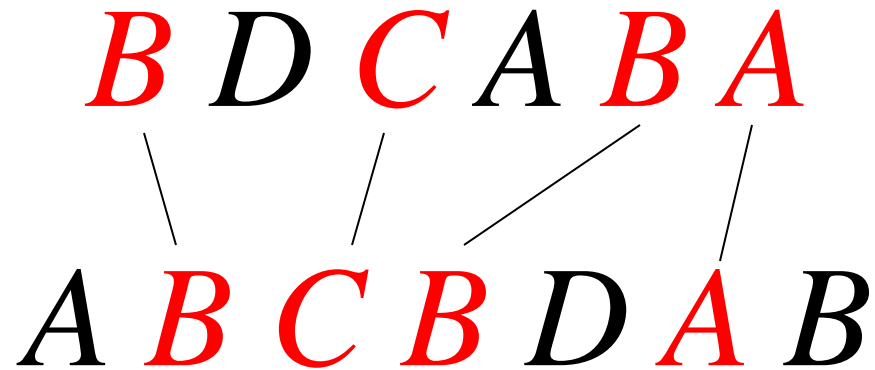
```
1   $n = p.length - 1$ 
2  let  $m[1..n, 1..n]$  be a new table
3  for  $i = 1$  to  $n$ 
4      for  $j = i$  to  $n$ 
5           $m[i, j] = \infty$ 
6  return LOOKUP-CHAIN( $m, p, 1, n$ )
```

LOOKUP-CHAIN(m, p, i, j)

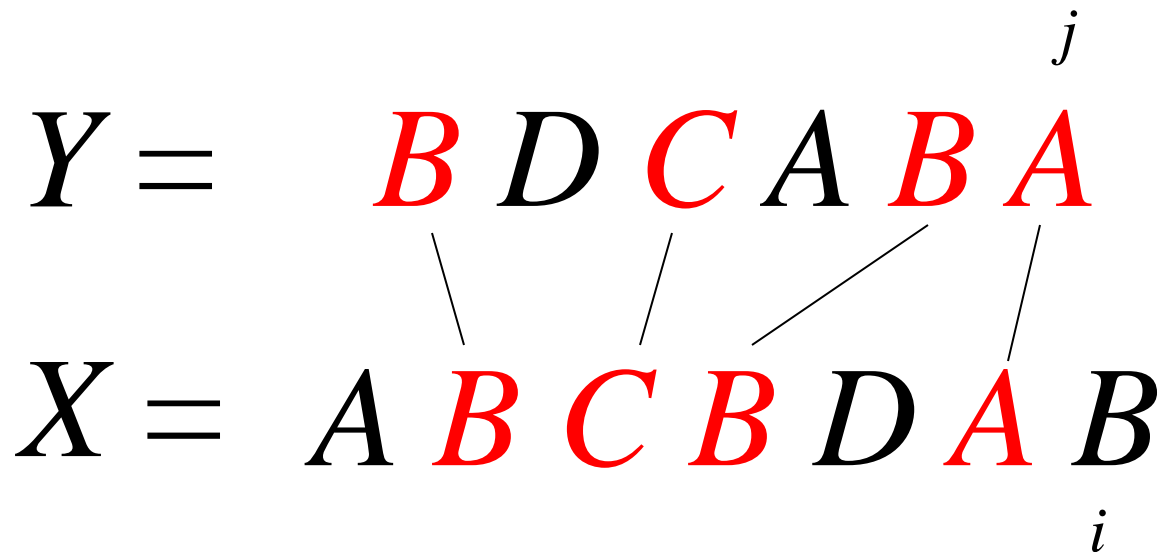
```
1  if  $m[i, j] < \infty$ 
2      return  $m[i, j]$ 
3  if  $i == j$ 
4       $m[i, j] = 0$ 
5  else for  $k = i$  to  $j - 1$ 
6       $q = \text{LOOKUP-CHAIN}(m, p, i, k)$ 
            $+ \text{LOOKUP-CHAIN}(m, p, k + 1, j) + p_{i-1}p_kp_j$ 
7      if  $q < m[i, j]$ 
8           $m[i, j] = q$ 
9  return  $m[i, j]$ 
```

Tid $O(n^3)$

Længste Fælles Delsekvens



Længste Fælles Delsekvens



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

længden af en længste fælles delsekvens af $x_1x_2 \cdots x_i$ og $y_1y_2 \cdots y_j$

Længste Fælles Delsekvens

j	0	1	2	3	4	5	6
i	y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0
1	A	0	↑	↑	↑	↖1	↖1
2	B	0	↖1	↖1	↖1	↑1	↖2
3	C	0	↑1	↑1	↖2	↖2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3
5	D	0	↑1	↖2	↑2	↑2	↖3
6	A	0	↑1	↑2	↑2	↖3	↑3
7	B	0	↖1	↑2	↑2	↑3	↖4

B D C A B A
 \ / / /
A B C B D A B

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

Længste Fælles Delsekvens

LCS-LENGTH(X, Y)

```

1   $m = X.length$ 
2   $n = Y.length$ 
3  let  $b[1..m, 1..n]$  and  $c[0..m, 0..n]$  be new tables
4  for  $i = 1$  to  $m$ 
5       $c[i, 0] = 0$ 
6  for  $j = 0$  to  $n$ 
7       $c[0, j] = 0$ 
8  for  $i = 1$  to  $m$ 
9      for  $j = 1$  to  $n$ 
10         if  $x_i == y_j$ 
11              $c[i, j] = c[i - 1, j - 1] + 1$ 
12              $b[i, j] = \nwarrow$ 
13         elseif  $c[i - 1, j] \geq c[i, j - 1]$ 
14              $c[i, j] = c[i - 1, j]$ 
15              $b[i, j] = \uparrow$ 
16         else  $c[i, j] = c[i, j - 1]$ 
17              $b[i, j] = \leftarrow$ 
18  return  $c$  and  $b$ 

```

		n							
		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
m	0	x_i	0	0	0	0	0	0	0
	1	A	0	↑	↑	↑	↖	←	↖
	2	B	0	↖	↑	←	↑	↖	←
	3	C	0	↑	↑	↖	←	↑	↑
	4	B	0	↖	↑	↑	↑	↖	←
	5	D	0	↑	↖	↑	↑	↑	↑
	6	A	0	↑	↑	↑	↖	↑	↖
	7	B	0	↖	↑	↑	↑	↖	↑

Tid $O(nm)$

Længste Fælles Delsekvens

PRINT-LCS(b, X, i, j)

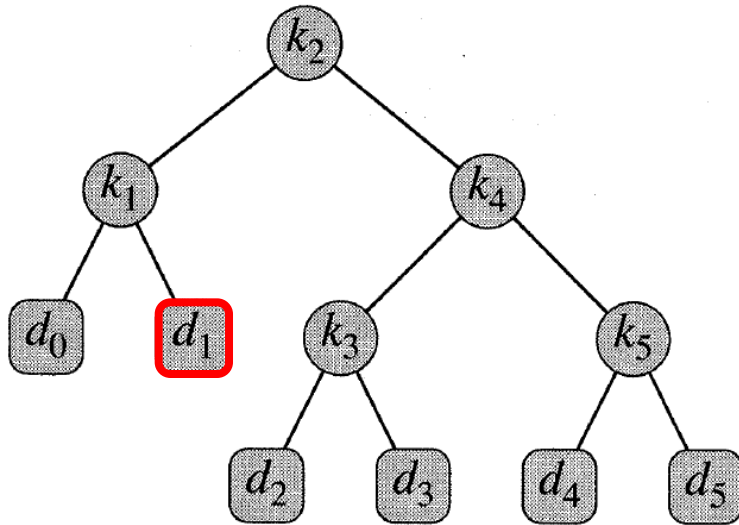
```

1  if  $i == 0$  or  $j == 0$ 
2      return
3  if  $b[i, j] == \swarrow$ 
4      PRINT-LCS( $b, X, i - 1, j - 1$ )
5      print  $x_i$ 
6  elseif  $b[i, j] == \uparrow$ 
7      PRINT-LCS( $b, X, i - 1, j$ )
8  else PRINT-LCS( $b, X, i, j - 1$ )
    
```

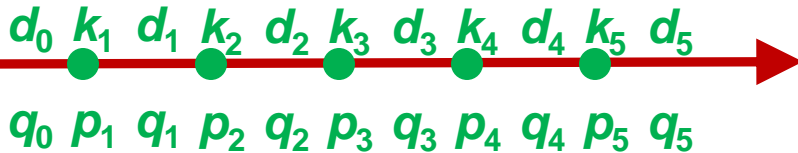
		n							
		j	0	1	2	3	4	5	6
i	y_j		B	D	C	A	B	A	
	0	x_i	0	0	0	0	0	0	0
1	A	0	↑	↑	↑	↖	←	↖	
2	B	0	↖	↑	←	↑	↖	←	
3	C	0	↑	↑	↖	←	↑	↑	
4	B	0	↖	↑	↑	↑	↖	←	
5	D	0	↑	↖	↑	↑	↑	↑	
6	A	0	↑	↑	↑	↖	↑	↖	
m 7	B	0	↖	↑	↑	↑	↖	↑	

Tid $O(n+m)$

Optimale Binære Søgetræer



Forventet søgetid 2.80

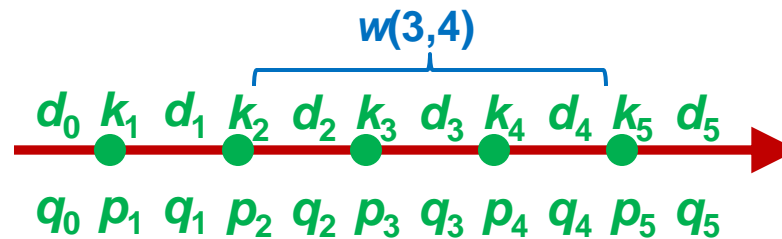


node	depth	probability	contribution
k_1	1	0.15	0.30
k_2	0	0.10	0.10
k_3	2	0.05	0.15
k_4	1	0.10	0.20
k_5	2	0.20	0.60
d_0	2	0.05	0.15
d_1	$(2 + 1) \times$	0.10	$= 0.30$
d_2	3	0.05	0.20
d_3	3	0.05	0.20
d_4	3	0.05	0.20
d_5	3	0.10	0.40
Total			2.80

INPUT		i	0	1	2	3	4	5
	p_i			0.15	0.10	0.05	0.10	0.20
	q_i		0.05	0.10	0.05	0.05	0.05	0.10

Optimale Binære Søgetræer

$$\sum_{i=1}^n p_i + \sum_{i=0}^n q_i = 1 \qquad w(i, j) = \sum_{l=i}^j p_l + \sum_{l=i-1}^j q_l$$



$$e[i, j] = \begin{cases} q_{i-1} & \text{if } j = i - 1, \\ \min_{i \leq r \leq j} \{e[i, r-1] + e[r+1, j] + w(i, j)\} & \text{if } i \leq j. \end{cases}$$

forventet optimal tid for et søgetræ indeholdende k_i, \dots, k_j og d_{i-1}, \dots, d_j

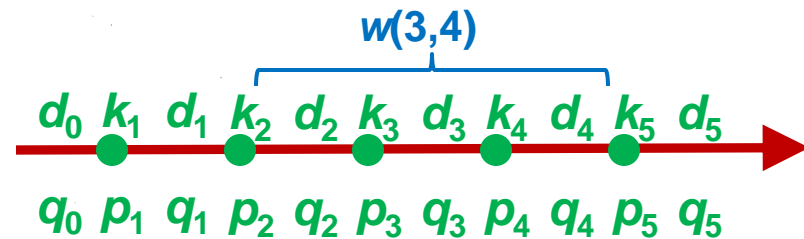
Optimale Binære Søgetræer

$$e[i, j] = \begin{cases} q_{i-1} & \text{if } j = i - 1, \\ \min_{i \leq r \leq j} \{e[i, r - 1] + e[r + 1, j] + w(i, j)\} & \text{if } i \leq j. \end{cases}$$

OPTIMAL-BST(p, q, n)

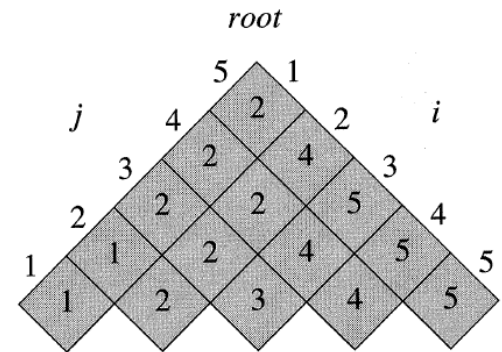
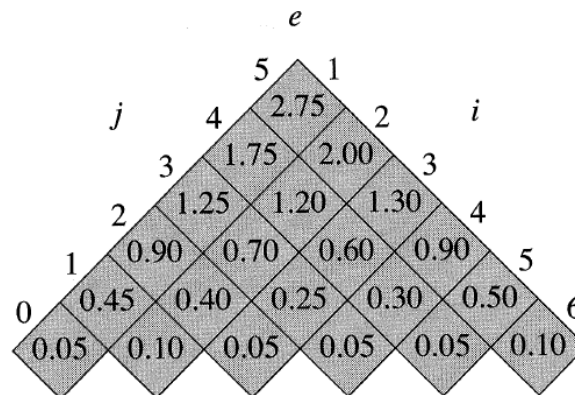
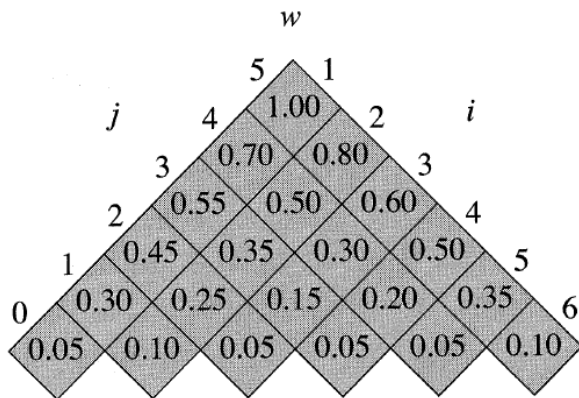
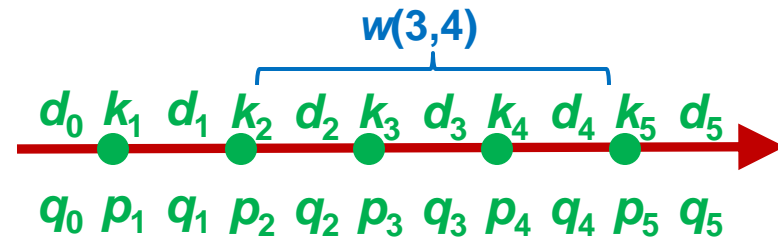
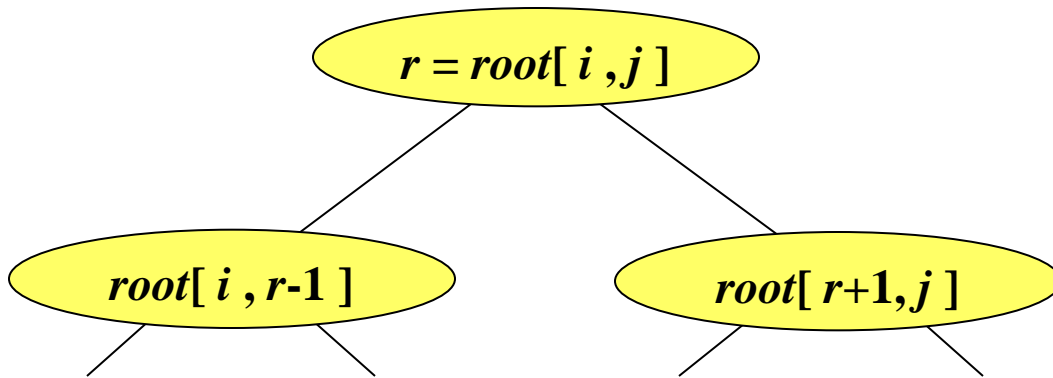
```

1  let  $e[1..n + 1, 0..n]$ ,  $w[1..n + 1, 0..n]$ ,
    and  $root[1..n, 1..n]$  be new tables
2  for  $i = 1$  to  $n + 1$ 
3       $e[i, i - 1] = q_{i-1}$ 
4       $w[i, i - 1] = q_{i-1}$ 
5  for  $l = 1$  to  $n$ 
6      for  $i = 1$  to  $n - l + 1$ 
7           $j = i + l - 1$ 
8           $e[i, j] = \infty$ 
9           $w[i, j] = w[i, j - 1] + p_j + q_j$ 
10         for  $r = i$  to  $j$ 
11              $t = e[i, r - 1] + e[r + 1, j] + w[i, j]$ 
12             if  $t < e[i, j]$ 
13                  $e[i, j] = t$ 
14                  $root[i, j] = r$ 
15  return  $e$  and  $root$ 
    
```



Tid $O(n^3)$

Konstruktion af Optimalt Binært Søgetræ



Dynamisk Programmering

- **Generel algoritmisk teknik**
- *Krav*: "Optimal delstruktur" – en løsning til problemet kan konstrueres ud fra optimale løsninger til "**delproblemer**"

- **Rekursionsligning**

- **Eksempler**

- Stang opdeling
- Matrix-kæde multiplikation
- Længste fælles delsekvens
- Optimale søgetræer