Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

Dynamische Programmierung

Optimale Unterstrukturen

- Ein Problem hat optimale Unterstrukturen, wenn eine optimale Lösung optimale Lösungen für Unterprobleme enthält
- Dies ist oft ein Indikator, dass dynamische Programmierung eingesetzt werden kann

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1, .., x_m)$ und $Y = (y_1, .., y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1, .., i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1, .., n$.

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1,..,x_m)$ und $Y = (y_1,..,y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1,..,i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1,..,n$.

Beispiel

Folge Y

B	C	A	C

Folge X

A	B	A	C	A	B	C

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1, .., x_m)$ und $Y = (y_1, .., y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1, .., i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1, .., n$.

Beispiel

Folge Y

B	C	A	C

Folge X

A	B	A	C	A	B	C

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1,..,x_m)$ und $Y = (y_1,..,y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1,..,i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1,..,n$.

Beispiel

Folge Y

B	C	A	C

Folge X

A	B	A	C	A	B	C

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1, .., x_m)$ und $Y = (y_1, .., y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1, .., i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1, .., n$.

Beispiel

Folge Y

B	C	A	C

Folge X

A	B	A	C	A	B	C

Längste gemeinsame Teilfolge

Definition

- Seien $X = (x_1, .., x_m)$ und $Y = (y_1, .., y_n)$ zwei Folgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A.
- Dann heißt Y Teilfolge von X, wenn es aufsteigend sortierte Indizes $i_1, .., i_n$ gibt mit $x_{i_j} = y_j$ für $j = 1, .., n$.

Beispiel

Folge Y

B	C	A	C

Folge X

A	B	A	C	A	B	C

- Y ist Teilfolge von X
- Wähle $(i_1, i_2, i_3, i_4) = (2,4,5,7)$

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z gemeinsame Teilfolge von X und Y, wenn Z Teilfolge sowohl von X als auch von Y ist.

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z gemeinsame Teilfolge von X und Y, wenn Z Teilfolge sowohl von X als auch von Y ist.

Beispiel

Folge Z

B	C	A	C

Folge X

A	B	A	C	A	B	C

Folge Y

B	A	C	C	A	B	B	C

- Z ist gemeinsame Teilfolge von X und Y

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z gemeinsame Teilfolge von X und Y, wenn Z Teilfolge sowohl von X als auch von Y ist.

Beispiel

Folge Z	B	C	A	C				
Folge X	A	B	A	C	A	B	C	
Folge Y	B	A	C	C	A	B	B	C

- Z ist gemeinsame Teilfolge von X und Y

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z gemeinsame Teilfolge von X und Y, wenn Z Teilfolge sowohl von X als auch von Y ist.

Beispiel

Folge Z	B	C	A	C				
Folge X	A	B	A	C	A	B	C	
Folge Y	B	A	C	C	A	B	B	C

- Z ist gemeinsame Teilfolge von X und Y

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z längste gemeinsame Teilfolge von X und Y, wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z längste gemeinsame Teilfolge von X und Y, wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Beispiel

Folge X

A	B	A	C	A	B	C

Folge Y

B	A	C	C	A	B	B	C

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z längste gemeinsame Teilfolge von X und Y, wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Beispiel

Folge X

A	B	A	C	A	B	C

Folge Y

B	A	C	C	A	B	B	C

Folge Z_1

B	C	A	C

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z längste gemeinsame Teilfolge von X und Y, wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Beispiel

Folge X

A	B	A	C	A	B	C

Folge Y

B	A	C	C	A	B	B	C

Folge Z_1

B	C	A	C

Folge Z_2

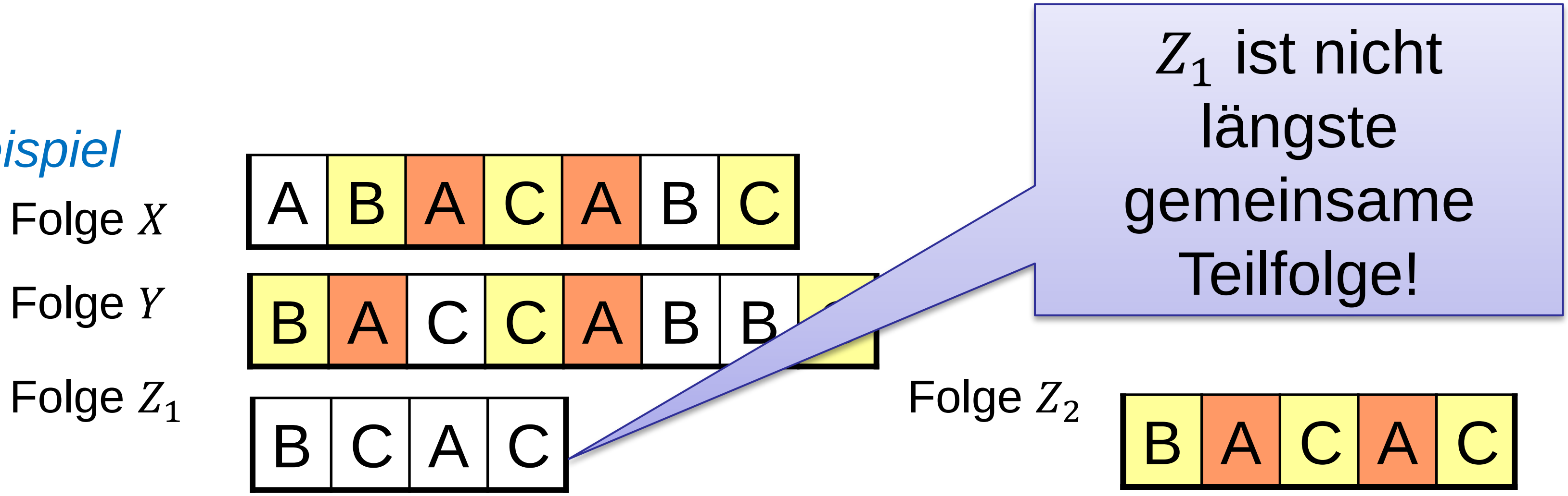
B	A	C	A	C

Längste gemeinsame Teilfolge

Definition

- Seien X, Y, Z Folgen über A.
- Dann heißt Z längste gemeinsame Teilfolge von X und Y, wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Beispiel



Problem LCS

Eingabe

- Folge $X = (x_1, \dots, x_m)$
- Folge $Y = (y_1, \dots, y_n)$

Ausgabe

- Längste gemeinsame Teilfolge Z
  (**L**ongest **C**ommon **S**ubsequence)

Problem LCS

Eingabe

- Folge $X = (x_1, \dots, x_m)$
- Folge $Y = (y_1, \dots, y_n)$

Ausgabe

- Längste gemeinsame Teilfolge Z
(**L**ongest **C**ommon **S**ubsequence)

Beispiel

Folge X

A	B	C	B	D	A	B

Folge Y

B	D	C	A	B	A

Einfacher Ansatz

Algorithmus

- Erzeuge alle möglichen Teilfolgen von X
- Teste für jede Teilfolge von X, ob auch Teilfolge von Y
- Merke zu jedem Zeitpunkt bisher längste gemeinsame Teilfolge

Laufzeit

- 2^m mögliche Teilfolgen
- Exponentielle Laufzeit!

Struktur von LCS

Satz 28

Seien $X = (x_1, \dots, x_m)$ und $Y = (y_1, \dots, y_n)$ beliebige Folgen und sei $Z = (z_1, \dots, z_k)$ eine längste gemeinsame Teilfolge von X und Y. Dann gilt

1. Ist $x_m = y_n$, dann ist $z_k = x_m = y_n$ und $(z_1, \dots, z_{k-1})$ ist eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und $(y_1, \dots, y_{n-1})$.

Struktur von LCS

Satz 28

Seien $X = (x_1, \dots, x_m)$ und $Y = (y_1, \dots, y_n)$ beliebige Folgen und sei $Z = (z_1, \dots, z_k)$ eine längste gemeinsame Teilfolge von X und Y. Dann gilt

1. Ist $x_m = y_n$, dann ist $z_k = x_m = y_n$ und $(z_1, \dots, z_{k-1})$ ist eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und $(y_1, \dots, y_{n-1})$.
2. Ist $x_m \neq y_n$ und $z_k \neq x_m$, dann ist Z eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und Y.

Struktur von LCS

Satz 28

Seien $X = (x_1, \dots, x_m)$ und $Y = (y_1, \dots, y_n)$ beliebige Folgen und sei $Z = (z_1, \dots, z_k)$ eine längste gemeinsame Teilfolge von X und Y. Dann gilt

1. Ist $x_m = y_n$, dann ist $z_k = x_m = y_n$ und $(z_1, \dots, z_{k-1})$ ist eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und $(y_1, \dots, y_{n-1})$.
2. Ist $x_m \neq y_n$ und $z_k \neq x_m$, dann ist Z eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und Y.
3. Ist $x_m \neq y_n$ und $z_k \neq y_n$, dann ist Z eine längste gemeinsame Teilfolge von X und $(y_1, \dots, y_{n-1})$.

Struktur von LCS

Satz 28

Seien $X = (x_1, \dots, x_m)$ und $Y = (y_1, \dots, y_n)$ beliebige Folgen und sei $Z = (z_1, \dots, z_k)$ eine längste gemeinsame Teilfolge von X und Y. Dann gilt

1. Ist $x_m = y_n$, dann ist $z_k = x_m = y_n$ und $(z_1, \dots, z_{k-1})$ ist eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und $(y_1, \dots, y_{n-1})$.
2. Ist $x_m \neq y_n$ und $z_k \neq x_m$, dann ist Z eine längste gemeinsame Teilfolge von $(x_1, \dots, x_{m-1})$ und Y.
3. Ist $x_m \neq y_n$ und $z_k \neq y_n$, dann ist Z eine längste gemeinsame Teilfolge von X und $(y_1, \dots, y_{n-1})$.

Optimale Unterstrukturen

Struktur von LCS

Beweis

(1) **Annahme:** $Z = (z_1, \dots, z_k)$ ist längste gemeinsame Teilfolge, $x_m = y_n$ und $z_k \neq x_m$

Dann können wir $z_{k+1} = x_m$ setzen, um eine gemeinsame Teilfolge von X und Y der Länge $k + 1$ zu erhalten. Widerspruch: Z ist eine *längste* gemeinsame Teilfolge von X und Y.

$\Rightarrow z_k = x_m = y_n$

$\Rightarrow (z_1, z_2, \dots\ z_{k-1})$ ist eine gemeinsame Teilfolge der Länge $k - 1$ von

$(x_1, x_2, \dots\ x_{m-1})$ und $(y_1, y_2, \dots\ y_{n-1})$.

Noch z.z.: $(z_1, z_2, \dots\ z_{k-1})$ ist längste gemeinsame Teilfolge von $(x_1, x_2, \dots, x_{m-1})$ und $(y_1, y_2, \dots, y_{n-1})$

Struktur von LCS

Beweis

(1) Noch z.z.: $(z_1, z_2, \ldots\, z_{k-1})$ ist längste gemeinsame Teilfolge von $(x_1, x_2, \ldots, x_{m-1})$ und $(y_1, y_2, \ldots, y_{n-1})$

Annahme: Es gibt eine gemeinsame Teilfolge W von $(x_1, x_2, \ldots, x_{m-1})$ und $(y_1, y_2, \ldots, y_{n-1})$, die mindestens Länge k hat. Dann erzeugt das Anhängen von $z_k = x_m$ an W eine gemeinsame Teilfolge von X und Y, deren Länge mindestens $k + 1$ ist. Widerspruch zur Optimalität von Z.

(2) Falls $z_k \neq x_m$ dann ist Z eine gemeinsame Teilfolge von $(x_1, x_2, \ldots, x_{m-1})$ und Y.

Annahme: Es gibt eine gemeinsame Teilfolge W von $(x_1, x_2, \ldots, x_{m-1})$ und Y mit einer Länge größer k.

Dann ist W auch eine gemeinsame Teilfolge von X und Y. Widerspruch: Z ist längste gemeinsame Teilfolge von X und Y.

(3) Der Beweis ist analog zu (2)

Aufgabe

Rekursive Formulierung für LCS

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$. Wie sieht eine Rekursion für $C[i][j]$ aus?

Rekursion für Länge von LCS

Korollar 29

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$. Dann gilt:

$$C[i][j] = \begin{cases} 0 & \text{, falls } i = 0 \text{ oder } j = 0 \\ & \end{cases}$$

Rekursion für Länge von LCS

Korollar 29

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von $(x_1, \dots, x_i)$ und $(y_1, \dots, y_j)$. Dann gilt:

$$C[i][j] = \begin{cases} 0 & \text{, falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{, falls } i, j > 0 \text{ und } x_i = y_j \\ & \end{cases}$$

Rekursion für Länge von LCS

Korollar 29

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$. Dann gilt:

$$C[i][j] = \begin{cases} 0 & \text{, falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{, falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{, falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

Rekursion für Länge von LCS

Korollar 29

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von $(x_1, \ldots, x_i)$ und $(y_1, \ldots, y_j)$. Dann gilt:

$$C[i][j] = \begin{cases} 0 & \text{, falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{, falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{, falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

Beobachtung

Rekursive Berechnung der $C[i][j]$ würde zu Berechnung immer wieder derselben Werte führen. Dieses ist ineffizient. Berechne daher die Werte $C[i][j]$ iterativ, nämlich zeilenweise.

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

$m \leftarrow \text{length}[X]$
$n \leftarrow \text{length}[Y]$
new array $C[0..m][0..n]$
for $i \leftarrow 0$ **to** m **do** $C[i][0] \leftarrow 0$
for $j \leftarrow 0$ **to** n **do** $C[0][j] \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
        ➢ Längenberechnung(X, Y, C, i, j)
return C

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

```
m ← length[X]
n ← length[Y]
new array C[0..m][0..n]
for i ← 0 to m do C[i][0] ← 0
for j ← 0 to n do C[0][j] ← 0
for i ← 1 to m do
    for j ← 1 to n do
        ➢ Längenberechnung(X, Y, C, i, j)
return C
```

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

1. $m \leftarrow \text{length}[X]$
2. $n \leftarrow \text{length}[Y]$
3. **new** array $C[0..m][0..n]$
4. **for** $i \leftarrow 0$ **to** m **do** $C[i][0] \leftarrow 0$
5. **for** $j \leftarrow 0$ **to** n **do** $C[0][j] \leftarrow 0$
6. **for** $i \leftarrow 1$ **to** m **do**
7. **for** $j \leftarrow 1$ **to** n **do**
8. ➢ Längenberechnung(X, Y, C, i, j)
9. **return** C

Tabelle für die $C[i][j]$ Werte anlegen.

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

```
m ← length[X]
n ← length[Y]
new array C[0..m][0..n]
for i ← 0 to m do C[i][0] ← 0
for j ← 0 to n do C[0][j] ← 0
for i ← 1 to m do
    for j ← 1 to n do
        ➢ Längenberechnung(X, Y, C, i, j)
return C
```

Erste Spalte der Tabelle auf 0 setzen.

Berechnung der $C[i][j]$ Werte

```
LCS-Länge(X, Y)
    m ← length[X]
    n ← length[Y]
    new array C[0..m][0..n]
    for i ← 0 to m do C[i][0] ← 0
    for j ← 0 to n do C[0][j] ← 0
    for i ← 1 to m do
        for j ← 1 to n do
            ➢ Längenberechnung(X, Y, C, i, j)
    return C
```

Erste Reihe der Tabelle auf 0 setzen.

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

$m \leftarrow$ length$[X]$
$n \leftarrow$ length$[Y]$
new array $C[0..m][0..n]$
for $i \leftarrow 0$ **to** m **do** $C[i][0] \leftarrow 0$
for $j \leftarrow 0$ **to** n **do** $C[0][j] \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
    for $j \leftarrow 1$ **to** n **do**
        ➢ Längenberechnung(X, Y, C, i, j)
return C

Berechnung der $C[i][j]$ Werte

Längenberechnung(X, Y, C, i, j)

```
if x_i = y_j then C[i][j] ← C[i−1][j−1] + 1
else
    if C[i−1][j] ≥ C[i][j−1] then C[i][j] ← C[i−1][j]
    else C[i][j] ← C[i][j−1]
```

$$C[i][j] = \begin{cases} 0 & \text{falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

Berechnung der $C[i][j]$ Werte

Längenberechnung(X, Y, C, i, j)

```
if x_i = y_j then C[i][j] ← C[i-1][j-1] + 1
else
    if C[i-1][j] ≥ C[i][j-1] then C[i][j] ← C[i-1][j]
    else C[i][j] ← C[i][j-1]
```

$$C[i][j] = \begin{cases} 0 & \text{falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

Berechnung der $C[i][j]$ Werte

LCS-Länge(X, Y)

$m \leftarrow \text{length}[X]$
$n \leftarrow \text{length}[Y]$
new array $C[0..m][0..n]$
for $i \leftarrow 0$ **to** m **do** $C[i][0] \leftarrow 0$
for $j \leftarrow 0$ **to** n **do** $C[0][j] \leftarrow 0$
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    ➢ Längenberechnung(X, Y, C, i, j)
return C

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

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

	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						
7	B	0						

	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						
7	B	0						

i	j	0	1	2	3	4	5	6
		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						
7	B	0						

	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	↑ 0					
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

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

	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	↑ 0					
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0				
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0			
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0			
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0			
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1		
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1		
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1		
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0						
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

i		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	$\uparrow$ 0	$\uparrow$ 0	$\uparrow$ 0	$\nwarrow$ 1	$\leftarrow$ 1	$\nwarrow$ 1
2	B		0	$\nwarrow$ 1					
3	C		0						
4	B		0						
5	D		0						
6	A		0						
7	B		0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1				
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1			
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0						
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0						
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0						
6	A	0						
7	B	0						

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

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

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

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

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

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

	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	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Laufzeitanalyse

Lemma 30

Der Algorithmus $\mathrm{LCS\text{-}Länge}$ hat Laufzeit $\mathbf{O}(nm)$, wenn die Folgen X,Y Länge n und m haben.

Beweis

Die Laufzeit wird durch die Initialisierung des Feldes in Zeile 3 sowie die geschachtelten **for**-Schleifen (Zeile 6 bis 8) dominiert. Daraus ergibt sich sofort eine Laufzeit von $\mathbf{O}(nm)$.

Laufzeitanalyse

Lemma 31

Algorithmus LCS-Länge berechnet die Länge einer längsten gemeinsamen Teilfolge.

Beweisskizze

Die Korrektheit folgt per Induktion über die Rekursion aus Korollar 29.

Laufzeitanalyse

Lemma 32

Die Ausgabe der längsten gemeinsamen Teilfolge anhand der Tabelle hat Laufzeit $\mathbf{O}(n+m)$, wenn die Folgen X,Y Länge n und m haben.

Beweis

In jedem Schritt bewegen wir uns entweder eine Zeile nach oben oder eine Spalte nach links. Daher ist die Laufzeit durch die Anzahl Zeilen plus die Anzahl Spalten begrenzt. Dies ist $\mathbf{O}(n+m)$.

Vorgehensweise bei dynamischer Programmierung

1. Bestimme rekursive Struktur einer optimalen Lösung.

2. Entwirf rekursive Methode zur Bestimmung des Wertes einer optimalen Lösung.

3. Transformiere rekursive Methode in eine iterative (bottom-up) Methode zur Bestimmung des Wertes einer optimalen Lösung.

4. Bestimme aus dem Wert einer optimalen Lösung und den in 3. ebenfalls berechneten Zusatzinformationen eine optimale Lösung.