הרצאה 8: תכנות דינמי

(Dynamic Programming)

בהרצאה זו נלמד פרדיגמה חדשה לבניית אלגוריתמים.
הפרדיגמה קרויה **תכנות דינמי** והיא משמשת לפתרון בעיות שאינן ניתנות לפתרון יעיל בשיטת הפרד ופתור.
בתחילת ההרצאה נציג בעיה פשוטה:
בעית התשלום המינימלי ונפתור אותה תוך שימוש בתכנות דינמי. לאחר מכן נציג את עיקרי השיטה באופן כללי.
במחצית השניה של ההרצאה נשתמש בתכנות דינמי כדי להציג אלגוריתם יעיל לפתרון בעיית תת הסדרה המשותפת המכסימלית.

דוגמא - בעית התשלום המינימלי

נתון כביש מהיר ובו n תחנות תשלום.
בכל תחנה התשלום שונה.
מכונית העוברת בכביש צריכה לשלם לפחות בכל שער שלישי. נניח כי יש 10 תחנות, המכונית יכולה לשלם לפי הסדרות הבאות:

3,6,9

1,4,7,8

2,3,5,6,7,10

2,3,4,7,9,10

סדרת התשלום הבאה אינה מותרת:

1,4,8,10

כי לא משלמים בשערים מס' 5,6,7

דוגמא - בעית התשלום המינימלי

יש להציע אלגוריתם לפתרון הבעיה הבאה:

קלט - מערך pay עם n איברים,
$pay[i]$ הוא התשלום בתחנה i.

פלט - התשלום המינימלי הנדרש למעבר הכביש.

בניית אלגוריתם רקורסיבי

ננסה לפתור את הבעיה בעזרת פרוצדורה רקורסיבית m_pay.
לכל $1 \le i \le n$, נגדיר את $m\_pay(i)$ כתשלום המינימלי שעלינו לשלם מתחנה i ועד תחנה n כאשר ידוע לנו כי עלינו לשלם בתחנה i.

החישוב הסופי

כל מכונית שעוברת בכביש חייבת לשלם בתחנה 1 או בתחנה 2 או בתחנה 3.
מכונית שעוברת בתחנה 1 תשלם לפחות $m\_pay(1)$.
מכונית שעוברת בתחנה 2 תשלם לפחות $m\_pay(2)$
מכונית שעוברת בתחנה 3 תשלם לפחות $m\_pay(3)$

התשלום המינימלי יתקבל על ידי:

$$fpay \leftarrow \min\left\{\begin{matrix} m\_pay(1) \\ m\_pay(2) \\ m\_pay(3) \end{matrix}\right\}$$

חישוב רקורסיבי של $m\_pay[i]$

לכל $1 \le i \le n-3$, נתון ש**אנו חייבים לשלם בתחנה** i והסכום הוא $pay[i]$.
לאחר ששילמנו תשלום זה, עלינו לבחור אם לשלם בתחנה $i+1$, בתחנה $i+2$ או בתחנה $i+3$.

אם בחרנו לשלם בתחנה $i+1$ יהיה עלינו לשלם עוד $m\_pay(i+1)$.
אם בחרנו לשלם בתחנה $i+2$ יהיה עלינו לשלם עוד $m\_pay(i+2)$.
אם בחרנו לשלם בתחנה $i+3$ יהיה עלינו לשלם עוד $m\_pay(i+3)$.

בניית אלגוריתם רקורסיבי (המשך)

כדי להחליט מהי התחנה הבאה בה נשלם נבצע 3 קריאות רקורסיביות ונחשב את המינימום בין הערכים המוחזרים:

$$fpay \leftarrow pay[i] + \min\left\{\begin{matrix} m\_pay(i+1) \\ m\_pay(i+2) \\ m\_pay(i+3) \end{matrix}\right\}$$

תנאי הקצה

$$m\_pay(n) = pay[n]$$
$$m\_pay(n-1) = pay[n-1]$$
$$m\_pay(n-2) = pay[n-2]$$

האלגוריתם

$compute(pay)$

$$fpay \leftarrow \min\left\{\begin{matrix} m\_pay(1) \\ m\_pay(2) \\ m\_pay(3) \end{matrix}\right\}$$

$\text{return}(fpay)$

procedure $m\_pay(i)$

if $i = n$ **or** $i = n-1$ **or** $i = n-2$

return$(pay[i])$

$$fpay \leftarrow pay[i] + \min\left\{\begin{matrix} m\_pay(i+1) \\ m\_pay(i+2) \\ m\_pay(i+3) \end{matrix}\right\}$$

return$(fpay)$

הוכחת נכונות

מיידית.

הסיבוכיות

נתבונן בעץ הקריאות הרקורסיביות.
כל קריאה עם $0 \le i \le n-3$ מניבה עץ **טרנרי** מלא (לכל צומת פנימי יש 3 בנים).
בעץ זה, אורך המסלולים מן השורש אל העלים הוא בין $n/3$ לבין n.
אם **נגזום** את העץ בעומק $n/3$, נקבל עץ טרנרי **מאוזן** ובו $3^{n/3}$ עלים.
מספר הקריאות הרקורסיביות גדול מ $3^{n/3}$ כלומר **אקספוננציאלי**.

בעיה

איך אפשר להוריד את הסיבוכיות הזו לרמה נסבלת?

כדי לענות על השאלה, עלינו לברר מהו המקור של הסיבוכיות הגבוהה הזו.

שימו לב:

יש **בדיוק** $n+1$ פרמטרים שונים לקריאות רקורסיביות:

$$m\_pay(1)$$
$$m\_pay(2)$$
$$\vdots$$
$$m\_pay(n)$$

כל אחת מן הקריאות האלה מתבצעת מספר גדול ביותר של פעמים.

מה הפתרון?

נבצע כל קריאה פעם אחת בלבד ונשמור את תוצאת הקריאה ב **טבלת מעקב (Look-up Table)**

נשתמש מערך עזר tm_pay ובו n איברים.

למעשה, נבטל לחלוטין את הקריאות הרקורסיביות ונחשב אך ורק את **הערכים המוחזרים** של קריאות אלה.

בכל פעם שיהיה בידינו הערך המוחזר על ידי $m\_pay(i)$ נכניס אותו לאיבר $tm\_pay[i]$.

מילוי המערך

אנו שואפים למלא את המערך בדרך היעילה ביותר. לכל i, נחשב את $tm\_pay[i]$, ברגע שיהיו בידינו כל הארגומנטים הנדרשים. נפתח במילוי האיברים התואמים לתנאי הקצה:

$$tm\_pay[n] \leftarrow pay[n]$$
$$tm\_pay[n-1] \leftarrow pay[n-1]$$
$$tm\_pay[n-2] \leftarrow pay[n-2]$$

כעת נחשב את $tm\_pay[n-3]$ על ידי

$$tm\_pay[n-3] \leftarrow$$
$$pay[n-3] + \min\left\{\begin{array}{l} tm\_pay[n-2] \\ tm\_pay[n-1] \\ tm\_pay[n] \end{array}\right\}$$

מילוי המערך (המשך)

באותו אופן לכל $1 \le i \le n-4$ בכל פעם שיהיו בידינו האיברים

$$tm\_pay[i+1],\ tm\_pay[i+2]$$
$$tm\_pay[i+3]$$

נוכל לחשב את $tm\_pay[i]$ על ידי

$$pay[i] + \min\left\{\begin{array}{l} tm\_pay[i+1] \\ tm\_pay[i+2] \\ tm\_pay[i+3] \end{array}\right\}$$

לאחר מילוי המערך, התוצאה הסופית תוחזר על ידי ביצוע

$$\min\left\{\begin{array}{l} tm\_pay[3] \\ tm\_pay[2] \\ tm\_pay[1] \end{array}\right\}$$

<u>הקוד</u>

$DP\_compute(pay)$

$tm\_pay[n] \leftarrow pay[n]$

$tm\_pay[n-1] \leftarrow pay[n-1]$

$tm\_pay[n-2] \leftarrow pay[n-2]$

for $i \leftarrow n-3$ **downto** 1 **do**

$tm\_pay[i] \leftarrow$

$$pay[i] + \min\begin{Bmatrix} tm\_pay[i+1] \\ tm\_pay[i+2] \\ tm\_pay[i+3] \end{Bmatrix}$$

end for

$$fpay \leftarrow \min\begin{Bmatrix} tm\_pay[3] \\ tm\_pay[2] \\ tm\_pay[1] \end{Bmatrix}$$

return$(fpay)$

<u>פיתוח אלגוריתמים בשיטת התכנות הדינמי</u>

שיטת ה**תכנות דינמי** לבניית אלגוריתמים משמשת לפתרון בעיות שאינן ניתנות לפתרון יעיל בשיטת הפרד ופתור.

בעיות אלה מתאפיינות על ידי הצורך לפתור אותה תת-בעיה הרבה מאוד פעמים.

אם נפתור את תת הבעיה כל פעם מחדש נגיע לסיבוכיות גבוהה ביותר.

במקום זאת אנו נפתור, באופן שיטתי, את כל תת הבעיות, אחת אחר השניה.

<u>פיתוח אלגוריתמים בשיטת התכנות הדינמי (המשך)</u>

בכל פעם שנסיים פתרון תת בעיה כלשהי, נאחסן את הפתרון כך שיהיה זמין לשימוש חוזר ככל שיידרש.

מקורם של אלגוריתמים בשיטת התכנות הדינמי הוא בבעיות שיש להן פתרון רקורסיבי, אך הפתרון הרקורסיבי אינו יעיל.

אם מספר צרופי הפרמטרים האפשריים הוא "קטן יחשית", כלומר **פולינומי** אפשר לעתים לפתור את הבעיה תוך שימוש בתכנות דינמי. קווי פעולה כלליים למציאת אלגוריתם כזה מוצגים בשקף הבא.

<u>קווי פעולה למציאת אלגוריתם בשיטת התכנות הדינמי</u>

1. מציגים אלגוריתם רקורסיבי לפתרון הבעיה.
2. מגלים כי האלגוריתם הרקורסיבי הוא אקספוננציאלי.
3. מגלים כי מספר צרופי הפרמטרים האפשריים הוא פולינומי וכי הסיבוכיות האקספוננציאלית נובעת ממספר רב של קריאות חוזרות עם וקטור פרמטרים זהה.
4. פותרים את הבעיה על ידי חישוב שיטתי של ערך השגרה הרקורסיבית עם כל צרופי הפרמטרים המותרים.

<u>תת הסדרה המשותפת הארוכה ביותר</u>

בהמשך ההרצאה, נדגים את שיטת ה**תכנות הדינמי** כדי לפתור את פתרון בעיית **תת הסדרה המשותפת המכסימלית**

(Longest Common Subsequence)

או **LCS**.

בבעיה זו הקלט הוא שתי סדרות של תווים והפלט הוא תת הסדרה הארוכה ביותר המשותפת לשתי סדרות הקלט.

<u>דוגמא</u>

$$X=(a,c,d,b,a,e,d,b,c,a,e,b,f)$$
$$Y=(c,a,b,c,e,a,b,f,d,a,c,d,f)$$

$$LCS=(c,a,b,c,e,b,f)$$

<u>תת הסדרה המכסימלית - הגדרת הבעיה</u>

קלט: שתי **סדרות**

$$X=(x_1,x_2,...,x_n)$$
$$Y=(y_1,y_2,...,y_m)$$

פלט: תת סדרה משותפת שארכה מכסימלי. להלן נקרא לפלט בשם **תת הסדרה המכסימלית של X ו Y.** נסמן את הפלט הנדרש

$$LCS(X,Y)$$

שימו לב: יתכנו מספר תת סדרות מכסימליות .

<u>תכונות תת הסדרה המשותפת</u>

להלן נוכיח מספרמשפטים אשר יאפשרו לנו לבנות אלגוריתם רקורסיבי לחישוב הפלט

<u>משפט פשוט – תנאי קצה</u>

יהיו X ו Y שתי סדרות. אם אחת הסדרות ריקה אז תת הסדרה המשותפת המכסימלית גם היא ריקה. נסמן את הסדרה הריקה ב Λ ונקבל: לכל X ולכל Y

$$LCS(X,\Lambda)=LCS(\Lambda,Y)=\Lambda$$

<u>הסבר</u>

האורך של תת הסדרה המכסימלית **קטן או שוה** לאורך כל אחת מן הסדרות.

<u>משפט נוסף – קיצור הקלט</u>

יהיו XוY סדרות. אם $x_n=y_m$, אזי

$$LCS(X,Y)=LCS(X_{n-1},Y_{m-1})\circ x_n$$

כלומר: כדי לחשב את תת הסדרה המשותפת המכסימלית של X וY יש:

1. לקצץ את התו המשותף מקצה X ומקצה Y.
2. לחשב את תת הסדרה המשותפת המכסימלית של שתי הסדרות המקוצצות X_{n-1} ו Y_{m-1}.
3. לשרשר את התו המשותף בקצה תת הסדרה המשותפת המכסימלית שהתקבלה בקריאה הרקורסיבית.

דוגמא

$$X=(a,c,b,d,f)$$

$$Y=(b,c,d,a,f)$$

במקרה זה:

$$LCS(X,Y)=$$
$$=LCS(X_4,Y_4)\circ f=(c,d,f)$$

הסבר

התו האחרון בשתי הסדרות חייב להיות התו האחרון בLCS. אחרת, נשרשר תו זה בקצה הLCS ונקבל תת סדרה משותפת ארוכה יותר.

עוד משפט

יהיו X וY סדרות. אם $x_n \neq y_m$, אזי

$$LCS(X,Y)=$$
$$LCS(X,Y_{m-1}) \mid LCS(X_n,Y)$$

כלומר אפשר להוריד תו אחד מן הקצה הימני של אחת הסדרות, בלי לפגוע ב LCS.

דוגמא

$$X=(c,b,d,g,f)$$

$$Y=(b,c,d,g)$$

במקרה זה אפשר להוריד את התו האחרון מן הסדרה X ולקבל

$$LCS(X,Y)=$$
$$LCS(X_3,Y_4)=(c,d,g)$$

הסבר

אם התו האחרון באחת הסדרות משתתף בLCS, הוא חייב להיות במקום האחרון של הLCS. במקום זה יכול להיות תו אחד בלבד. לכן, אפשר להוריד את התו האחרון בסדרה השניה.

אם שני התווים האחרונים אינם משתתפים בLCS, בוודאי אפשר להוריד כל אחד מהם.

שימו לב: ניסוח פורמלי והוכחה של משפט זה מופיעים בנספח להרצאה.

אלגוריתם רקורסיבי לחישוב אורך *LCS*

בהנתן שתי סדרות X ו Y, אפשר לחשב ישירות את תת הסדרה המכסימלית אולם, מסתבר כי קל יותר לחשב תחילה את האורך של תת הסדרה הזו ורק לאחר מכן לחשב את תת הסדרה המכסימלית עצמה.

להלן נציג אלגוריתם הרקורסיבי לחישוב אורך הLCS תוך שימוש במשפטים שלמדנו.

סימונים

בהנתן X ו Y, נסמן ב $L(X,Y)$ את אורך תת הסדרה המכסימלית של X וY.

נניח כי $X=(x_1,...,x_n)$

ו $Y=(y_1,...,y_m)$

איברי הסדרות X ו Y יקרא **תוים.**

הרישא ה i של X, $0\le i\le n$, תסומן על ידי $X_i=(x_1,...,x_i)$

הרישא ה j של Y, $0\le j\le m$, תסומן על ידי $Y_j=(y_1,...,y_j)$.

לכ ל i וj, $0\le i\le n$, $0\le j\le m$,

נסמן ב$L(X_i,Y_j)$ את אורך תת הסדרה המכסימלית של X_i וY_j.

נוסחת נסיגה לחישוב אורך LCS

תנאי קצה

אם אורכה של אחת הסדרות הוא 0, אז אורך הLCS גם הוא 0. כלומר:

לכל X ו Y ולכל i וj $0\le i\le n$ ו $0\le j\le m$,:

$$L(X_i,\Lambda)=L(\Lambda,Y_j)=0$$

נוסחת נסיגה

מהמשפטים שלמדנו נובע כי

לכל X ו Y ולכל i וj $1\le i\le n$ ו $1\le j\le m$, אם $x_i=y_j$, אז

$$L(X_i,Y_j)=L(X_{i-1},Y_{j-1})+1$$

אם $x_i\ne y_j$**, אז**

$$L(X_i,Y_j)=\max\{L(X_{i-1},Y_j)L(X_i,Y_{j-1})\}$$

אלגוריתם רקורסיבי עבור $L(X_i,Y_j)$

$L(X_i,Y_j)$

if $i=0$ **return** 0

if $j=0$ **return** 0

if $x_i=y_j$

return $L(X_{i-1},Y_{j-1})+1$

else

return

$\max\{L(X_{i-1},Y_j),L(X_i,Y_{j-1})\}$

end

נכונות וסיבוכיות

נכונות האלגוריתם נובעת מיידית מן המשפט שהוכחנו.

סיבוכיות האלגוריתם, תלויה במספר הקריאות הרקורסיביות. במקרה הגרוע ביותר, בו כל אותיות הקלט שונות זו מזו. במקרה זה, כל קריאה לאלגוריתם, עם קלטים שאינם ריקים ואשר סכום אורכיהם n , מבצעת שתי קריאות רקורסיביות עם קלטים שסכום אורכיהם הוא $n-1$ ולכן :

$$T(n)=2T(n-1)+c$$

במקרה זה, הסיבוכיות של T היא אקספוננציאלית.

אלגוריתם בתכנות דינמי

נשים לב כי לכל זוג סדרות, X ו Y, מספר צרופי הפרמטרים השונים לפרוצדורה L, כולל סדרות ריקות, הוא בדיוק

$$(m+1)\cdot(n+1)$$

במצב זה, אפשר להחליף את הפונקציה הרקורסיבית $L(,)$ במערך דו-ממדי $TL[,]$ מסדר $(m+1)\times(n+1)$ כאשר בסיום האלגוריתם נקבל $TL(i,j)=L(X_i,Y_j)$, כלומר האיבר ה(i,j) יכיל את **האורך** של תת הסדרה המכסימלית של X_i ושל Y_j.

האלגוריתם שנציע, יחשב את ערכי המערך, תוך שימוש במשפט שהוכחנו.

תאור האלגוריתם בתכנות דינמי

האלגוריתם מאפס תחילה את השורה ה 0 ואת העמודה ה 0 של המטריצה המתאימות לאורך LCS של שתי סדרות שאחת מהן ריקה.

לאחר מכן, האלגוריתם מחשב את איברי המטריצה, שורה אחר שורה, תוך שימוש בנוסחת הנסיגה מן האלגוריתם הרקורסיבי, ובערכי המטריצה שחושבו כבר.

עם סיום האלגוריתם מתקיים :

$$|LCS(X,Y)|=L(X_n,Y_m)=TL[n,m]$$

האלגוריתם - הקוד

$DPL(X,Y)$
for $i\leftarrow 1$ **to** m **do** $TL[i,0]\leftarrow 0$
for $j\leftarrow 1$ **to** n **do** $TL[0,j]\leftarrow 0$
for $i\leftarrow 1$ **to** m **do**
  for $j\leftarrow 1$ **to** n **do**
    if $x_i=y_j$ **then**
      $TL[i,j]\leftarrow TL[i-1,j-1]+1$
    else if $TL[i,j-1]\geq TL[i-1,j]$
      $TL[i,j]\leftarrow TL[i,j-1]$
    else
      $TL[i,j]\leftarrow TL[i-1,j]$
  end
end
return$TL(n,m)$

דוגמא

נתבונן בדוגמא שהצגנו בתחילת ההרצאה:

$$X=(a,c,d,b,a,e,d,b,c,a,e,b,f)$$

$$Y=(c,a,b,c,e,a,b,f,d,a,c,d,f)$$

להלן, נבנה את המערך TL עבור הדוגמא הזו.

דוגמא

	Λ	c	a	b	c	e	a	b	f	d	a	c	d	f
Λ														
a														
b														
d														
b														
b														
e														
d														
b														
c														
a														
e														
b														
f														

דוגמא

	Λ	c	a	b	c	e	a	b	f	d	a	c	d	f
Λ	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	1	1	1	1	1	1	1	1	1	1	1	1
b	0	1	1	1	2	2	2	2	2	2	2	2	2	2
d	0	1	1	1	2	2	2	2	2	3	3	3	3	3
b	0	1	1	2	2	2	2	3	3	3	3	3	3	3
b	0	1	2	2	2	2	3	3	3	3	4	4	4	4
e	0	1	2	2	2	3	3	3	3	3	4	4	4	4
d	0	1	2	2	2	3	3	3	3	4	4	4	5	5
b	0	1	2	3	3	3	3	4	4	4	4	4	5	5
c	0	1	2	3	4	4	4	4	4	4	4	5	5	5
a	0	1	2	3	4	4	5	5	5	5	5	5	5	5
e	0	1	2	3	4	5	5	5	5	5	5	5	5	5
b	0	1	2	3	4	5	5	6	6	6	6	6	6	6
f	0	1	2	3	4	5	5	6	7	7	7	7	7	7

סיכום

בהרצאה זו, למדנו פרדיגמה חדשה לתכנון אלגוריתמים הקרויה: **תכנות דינמי (Dynamic Programming).**

עיקר ההרצאה הוקדש לפתרון בעית **תת הסדרה המכסימלית** תוך שימוש בתכנות דינמי.

להרצאה זו 2 נספחים בהם אנו משלימים את החומר שהובא בהרצאה:

בנספח 1 אנו מוכיחים את נכונות האלגוריתם באופן פורמלי.

בנספח 2 אנו מביאים את האלגוריתם המקבל כקלט את טבלת המעקב TL ומחשב את תת הסדרה המשותפת המכסימלית.

נספח 1: הוכחת המשפט היסודי

המשפט הבא מוכיח באופן פורמלי את רעיון האלגוריתם.

משפט

יהיו $X=(x_1,...,x_n)$ ו $Y=(y_1,...,y_m)$ שתי סדרות ותהי $Z=(z_1,...,z_k)$ אחת מה LCS שלהם. אזי:

1. אם $x_n=y_m$, אזי $z_k=x_n$ ו Z_{k-1} היא LCS של X_{n-1} ו Y_{m-1}.
2. אם $x_n \ne y_m$, ו$x_n \ne z_k$ אזי Z היא LCS של X_{n-1} ו Y.
3. אם $y_m \ne x_n$ ו $y_m \ne z_k$, אזי Z היא LCS של Y_{m-1} וX.

שימו לב: סעיפים 2 ו3 הם סימטריים.

הוכחה

הוכחת 1: אם $z_k \ne x_n$, נתבונן ב $(z_1,...,z_k,x_n)$. זוהי, תת סדרה משותפת לX ול Y, בסתירה לכך שאורכה של Z הוא מכסימלי. נותר לנו להוכיח כי $Z_{k-1}=z_1,...,z_{k-1}$ היא LCS של X_{n-1} ושל Y_{m-1}. ברור כי Z_{k-1} היא תת סדרה משותפת של X_{n-1} ושל Y_{m-1} . נניח בדרך השלילה, כי ל X_{n-1} ול Y_{m-1} יש תת סדרה משותפת ארוכה יותר $W=(w_1,...,w_k)$. נתבונן ב$W \circ x_n=(w_1,...,w_k,x_n)$. זוהי תת סדרה משותפת של X ושל Y שארכה $k+1$, בסתירה להנחה שארכה של ה LCSהוא k. **מש"ל**

תרגיל: הוכח את סעיפים 2 ו3.

נספח 2: חישוב תת הסדרה המכסימלית - תכונות המערך TL

לאחר שסיימנו את ביצוע האלגוריתם, אורך תת הסדרה המכסימלית נמצא באיבר $TL[n,m]$.

ברצוננו לפתח אלגוריתם לחישוב תת הסדרה המכסימלית עצמה.

נשים לב כי:

1. הפרש הערכים בין כל שני תאים סמוכים הוא 0 או 1.
2. הערך באיבר ה$l[i,j]$ מחושב כפונקציה של האיברים $l[i,j-1]$, $l[i-1,j]$ ו$l[i-1,j-1]$.

תאור האלגוריתם

להלן מוצגות את חמש התצורות האפשריות לקבוצות בנות ארבעה תאים סמוכים במערך TL.

אנו מניחים כי האיבר הימני בשורה התחתונה הוא $TL[i,j]$ ומתקיים $TL[i,j]=k$, כמתואר בטבלה הבאה:

$TL[i-1,j-1]$	$TL[i-1,j]$
$TL[i,j-1]$	$TL[i,j]=k$

לטבלה זו, אנו קוראים **התצורה של** $TL[i,j]$**.**

חישוב תת הסדרה המכסימלית - תצורות 1-4

בכל אחת מארבעת התצורות הבאות הערך $TL[i,j]=k$, יכול היה להתקבל מאחד האיברים הסמוכים, ללא תוספת תו ל LCS.

k	k
k	k

תצורה 1

$k-1$	k
k	k

תצורה 2

$k-1$	k
$k-1$	k

תצורה 3

$k-1$	$k-1$
k	k

תצורה 4

חישוב תת הסדרה המכסימלית - תצורה 5

בתצורה הבאה הערך $TL[i,j]=k$ התקבל **אך ורק** על ידי זיהוי תו משותף ותוספת 1 לערך $TL[i-1,j-1]$.

$k-1$	$k-1$
$k-1$	k

אלגוריתם לחישוב תת הסדרה המכסימלית - איתחול

כדי לבנות את תת הסדרה המשותפת המכסימלית, מבצעים סיור מ $TL(n,m)$ אל $TL(1,1)$. במהלך הסיור בונים את תת הסדרה המשותפת המכסימלית, מן הסוף אל ההתחלה.

לפני תחילת הסיור מאתחלים:

$(j,i)\leftarrow(m,n)$

$LCS\leftarrow\Lambda$

אלגוריתם לחישוב תת הסדרה המכסימלית - צעד

בכל צעד בסיור, עוברים ממיקומנו הנוכחי $TL(i,j)$ אל אחד האיברים הסמוכים.

נניח כי מיקומנו הנוכחי הוא האיבר $TL[i,j]$. המשך הסיור מ$TL[i,j]$, תלוי בתצורה של איבר זה:

אם זוהי אחת מן התצורות 1-3, האיבר הבא הוא $l[i-1,j]$.

שימו לב: בתצורות 1-2, אפשר להתקדם הן אל $l[i-1,j]$ והן אל $l[i,j-1]$. ההחלטה להתקדם אל $l[i-1,j]$ היא שרירותית. התקדמות אל $l[i,j-1]$ תניב תת סדרה משותפת מכסימלית **אחרת**.

אלגוריתם לחישוב תת הסדרה המכסימלית - צעד (המשך)

אם $TL[i,j]$ נמצא תצורה 4, האיבר הבא הוא $TL[i,j-1]$.

בכל המקרים שתוארו עד כה, הערך של $TL[i,j]$, התקבל מאיבר שכן, ללא תוספת תו לתת הסדרה המכסימלית.

אם לעומת זאת $TL[i,j]$ נמצא בתצורה 5, האיבר הבא בסיור הוא $TL[i-1,j-1]$. במקרה זה, חייב להתקיים $x_i = y_j$ ותוך כדי המעבר מ$TL[i,j]$ אל $TL[i-1,j-1]$, אנו מוסיפים לתת הסדרה המשותפת את התו המשותף x_i.

חישוב תת הסדרה המשותפת המכסימלית - צעד

בכל צעד בסיור, עוברים ממיקומנו הנוכחי $TL[i,j]$ אל האיבר שבאמצעותו חישבנו את $TL[i,j]$ כדלהלן:

אם מתקיים השוויון

$$TL(i,j) = TL(i-1,j-1)+1$$

אזי, בשלב זה תת הסדרה המכסימלית הוארכה על ידי שרשור האיבר המשותף $x_i = y_j$ **לפני** האיברים שנמצאו עד כה.במקרה זה, מבצעים:

$LCS \leftarrow x_i \circ LCS$

$(i,j) \leftarrow (i-1,j-1)$

חישוב תת הסדרה המשותפת המכסימלית - צעד (המשך)

אם השוויון הקודם לא מתקיים, אזי מתקיים לפחות אחד מאי השוויונים הבאים:

$TL(i,j) = TL(i-1,j)$

$TL(i,j) = TL(i,j-1)$

תיתכן גם האפשרות או ששניהם מתקיימים.

בכל אחד ממקרים אלה, הסדרה LCS לא משתנה. כדי להמשיך בסיור, עוברים אל איבר המטריצה המקיים את השויון. הקוד הסופי מופיע בשקף הבא:

הקוד

$Extract\_LCS(TL)$

$LCS \leftarrow \Lambda$

$(i,j) \leftarrow (n,m)$

while $(i,j) \neq (1,1)$ **do**

if $TL[i,j] = TL[i-1,j]$

$(i,j) \leftarrow (i-1,j)$

elseif $TL[i,j] = TL[i,j-1]$

$(i,j) \leftarrow (i,j-1)$

else

($TL[i,j] \neq TL[i-1,j]$ **and** $TL[i,j] \neq TL[i,j-1]$**)**

$LCS \leftarrow x_i \circ LCS$

$(i,j) \leftarrow (i-1,j-1)$

endif

endwhile