15.11.16

אלגוריתמים - הרצאה 5

בעיית תת סדרה הארוכה ביותר: (המשותפת)

__הגדרה:__ יהי $X=X_1,\dots,X_n$ סדרה. $Z=Z_1,\dots,Z_k$ תת סדרה של X

אם קיימים $1\le i_1\le i_2\le\dots\le i_k\le n$ כך ש- $Z_i=X_{i_j}$

דוגמ- $X=abbacac$

$Z=bca$

__הגדרה:__ עבור $X=x_1,\dots,x_n$, $Y=y_1,\dots,y_m$, $Z=z_1,\dots,z_k$ תת סדרה משותפת של X ו-Y אם Z תת סדרה של X וגם תת סדרה של Y.

דוגמ- $X=abacabca$, $Y=baacdabc$, $Z=bacabc$

__בעיה:__

קלט: $X=x_1,\dots,x_n$ ו- $Y=y_1,\dots,y_m$

פלט: תת סדרה המשותפת הארוכה ביותר.

אורך תת סדרה הארוכה ביותר המשותפת של X ו-Y.

__הצעות אלגוריתמים:__

1) פתרון נאיבי -

עבור על כל תתי הסדרות של X ונבדוק האם הן תת סדרה של Y, אם כן נעדכן אורך מקסימלי.

זמן ריצה: מס' תתי הסדרות של סדרה באורך n שווה למס' המחרוזות הבינאריות באורך n: 2^n.

על כל תת מחרוזת כזו נעבור על Y פעם אחת לכן $O(m)$

$\Leftarrow$ $O(m\cdot 2^n)$ או $O(n\cdot 2^m)$

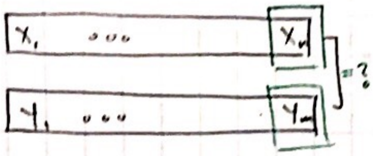
יכלנו גם לעבור על כל תתי סדרות של Y ולבדוק ב- X.

(*) נרצה לעבור על כל המחרוזות של המערך הכי קטן.

לכן אם $m<n$ $\Leftarrow$ $O(n\cdot 2^m)$ ואם $m>n$ $\Leftarrow$ $O(m\cdot 2^n)$

2) פתרון רקורסיבי: נרצה למצוא נוסחת נסיגה ולזו אלגוריתם כדי להשתמש בו בפתרון בתכנות דינמי.

נחפש תכונות שהמחרוזות מקיימות:



(*) אם $y_m = x_n$ אזי בהכרח $z_k = y_m = x_n$

(*) אם $y_m \neq x_n$ אזי מתקיים אחד מהתנאים:

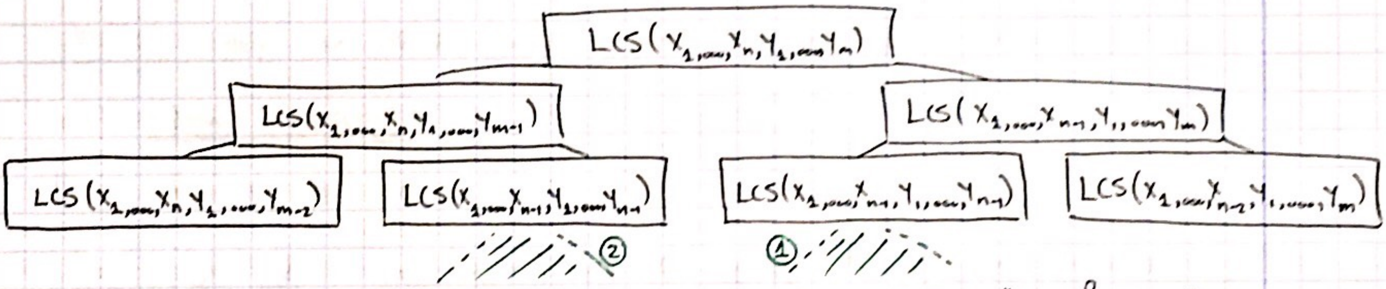
1) $z_k \neq x_n$ או 2) $z_k \neq y_m$

נפתח לבסוף את נוסחת הנסיגה:

Longest Common Subsequence

$$LCS(x_1,\dots,x_n,y_1,\dots,y_m) = \begin{cases} 0 & , \text{אם } n=0 \text{ או } m=0 \\ LCS(x_1,\dots,x_{n-1},y_1,\dots,y_{m-1})+1 & , \text{אם } y_m = x_n \\ \max\{LCS(x_1,\dots,x_{n-1},y_1,\dots,y_m),\ LCS(x_1,\dots,x_n,y_1,\dots,y_{m-1})\} & \text{אם } x_n \neq y_m \end{cases}$$

(*) נניח שכל התווים של X שונים מכל התווים של Y. (worst-case)



נסתכל שהעץ הנ"ל ממשיך עד ש- n או m שווה ל- 0.

נניח m=n ונקבל n רמות מלאות (עד 2^n עלים) ⇐ מס' הקודקודים $\geq 2^n$.

נראה שעלה 1 ועלה 2 אנו בעצם מחשבים פעמיים את אותו קדקוד!

3) פתרון תכנות דינמי:

מבנה הנתונים: מערך דו מימדי. טבלה $n \times m$.

LCS	0	1	...	j	...	m
0	0	0	...			0
1	0					
⋮	⋮					
i						
⋮						
n	0					

$\square \rightarrow LCS(x_1,\dots,x_i,y_1,\dots,y_j)$

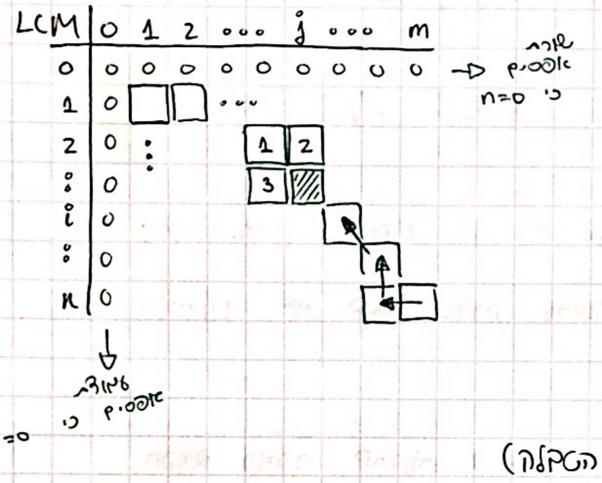
הפתרון שאנו מחפשים יהיה פה.

$LCS(x_1,\dots,x_n,y_1,\dots,y_m)$

15.11.16

אופן המימוש: כאשר $i=0$, $j=0$ נמלא את השורות האלה באפסים.

לאחר מכן נלך לנקודה ב- $i=1$ ו- $j=1$ ונמלא את השורה והעמודה



(*) נרצה לחשב את ▨ כאשר כבר חישבנו את שלושת המשבצות שלידה.

דרך החישוב תהיה כך:

אם $Y_j = X_i \Leftarrow$ ▨ $= \boxed{1} + 1$

אם $Y_j \neq X_i \Leftarrow$ ▨ $= \max\{\boxed{2}, \boxed{3}\}$

זמן החישוב יהיה $O(m \cdot n)$ (כגודל הטבלה)

סיבוכיות מקום $O(m \cdot n)$

[redacted]

פסאודו קוד של האלגוריתם:

LCS

```
for i=0 to n  c[i,0] ← 0
for j=0 to m  c[0,j] ← 0
for i=1 to n
    for j=1 to m
        if x_i = y_j
            c[i,j] ← c[i-1, j-1] + 1
            d[i,j] ← "↖"
        else
            if c[i,j-1] > c[i-1,j]
                c[i,j] ← c[i,j-1]
                d[i,j] ← "←"
            else
                c[i,j] ← c[i-1,j]
                d[i,j] ← "↑"
```

(*) למה צריך את טבלה d ?

← בסוף התהליך התשובה שלנו נמצאת ב- $C[n,m]$ אך זו רק התשובה של מה האורך של המחרוזת.

d שומרת לנו את המחרוזת עצמה. כך נוכל לשחזר את התשובה.

(*) תהליך השחזור עם החצים לוקח $O(n+m)$ זמן ו- $O(m \cdot n)$ מקום.

Scanned by CamScanner

האם אנחנו צריכים את כל הטבלה?

← אם מחשבים לפי שורות, מספיק לשמור את השורה שמחשבים ואת השורה מעליה ⇐ $O(m)$ מקום.

← אם מחשבים לפי עמודות ⇐ $O(n)$ מקום

⇐ בסה"כ נוכל להשתמש ב- $O(\min(m,n))$ מקום.

ניתן לחסוך במקום רק אם אנחנו רוצים לקבל את אורך תת הסדרה, אך אם אנחנו מעוניינים גם בתת סדרה עצמה נצטרך גם את הטבלה b וכאן לא ניתן לחסוך.

הקושי העיקרי בתכנות דינמי הוא, כמובן, הגדרה נכונה של נוסחת הנסיגה אך בעיקר השמירה של הנתונים בצורה שנוכל להשתמש בהם בחישובים הבאים.

אלגוריתמים חמדניים:

ניקח את הדבר הכי טוב באופן מקומי וכך נגיע לדבר הכי טוב באופן גלובלי. כלומר זו שיטה שפותרת בעיה גלובלית ע"י ביצוע פתרונות מקומיים אופטימליים ומזה נסיק פתרון כללי אופטימלי.

← לא בכל פעם אלגוריתם חמדני יעבוד ולכן צריך כל פעם להוכיח נכונות.

דוגמא באלגוריתם חמדני שלא עובדת:

נסתכל על מכפלת המטריצות משיעור שעבר - $M_1 M_2 \cdots (M_i M_{i+1}) \cdots M_n$

$P_{i-1} \times P_i$ $\quad$ $P_i \times P_{i+1}$

$P_{i-1} \times P_{i+1}$

(*) נמצא כל פעם את החישוב המינימלי הנקודתי, נשמור אותו ונמשיך הלאה.
הפתרון הזה לא עובד כי בכל פעם שמנו סוגריים במקום מסויים אך יש סיכוי שבהסתכלות יותר רחבה הסוגריים במיקום שונה ייתנו תוצאה יותר טובה.

קידוד - דחיסה:

1 2 3 4 5 6 7 8 9 10 11 12 13
X= aaabacabadaab

איך נוכל לשמור את X בצורה שלא תאבד מידע עם מינימום מקום?

- בצורה של ASCII: 8 ביטים לכל אות ← $8\times13=104$
- בצורת קידוד כזו: נבנה טבלה:

a	00
b	01
c	10
d	11

← $2\times13=26$

- בצורת קידוד כזו: נבנה טבלה שונה כך:

a	0
b	10
c	110
d	111

נספור כמה פעמים כל אות מופיעה ונקבל:
$8\cdot1+3\cdot2+1\cdot3+1\cdot3=20$

בס"ד.

15.11.16

כלומר, ניתן לראות שבצורת קידוד שונה, אותה מחרוזת תתפוס מקום שונה.

בנוסף למינימום מקום שמירה נרצה פיענוח יחיד, ופיענוח מיידי.

כלומר, נרצה לדעת מיד, בהינתן מחרוזת של ביטים מה הייתה המחרוזת.

לדוג' - נסתכל על אופציית הקידוד הזאת עם הטבלה

a	00
b	001
c	101
d	01

יכולה להתקבל מחרוזת כזו:

$$\overbrace{0\,0}^{a}\,\overbrace{1\,0\,1}^{c}$$
$$\underbrace{0\,0\,1}_{b}\,\underbrace{0\,1}_{d}$$

וכך לא נוכל לדעת איזו מחרוזת קיבלנו.

לכן נגדיר:

קוד: פונקציה מהא"ב Σ לתוך $\{0,1\}^*$ $\qquad c:\Sigma\to\{0,1\}^*$

קוד רישות: קוד c המקיים לכל $a,b\in\Sigma$ $c(a)$ אינו רישא של $c(b)$

<u>הערה</u>: קוד רישות הוא בעל פיענוח יחיד ומיידי.

בעיה:

קלט: א"ב עם תדירויות $\Leftarrow$ $\Sigma=\{a_1,\dots,a_n\}$ כאשר לכל i: f_i זו התדירות ש- a_i מופיעה במילה.
$f_1,\dots,f_n$

דוג' לקלט: אם $x=aaabacabadaab$ אזי הקלט יהיה: a,b,c,d
$8,3,1,1$

פלט: קוד רישות "אופטימלי", c כלומר, $\sum_{i=1}^{n}|c(a_i)|\cdot f_i$ מינימלי.

כאשר $|c(a_i)|$ = אורך הקוד של a_i