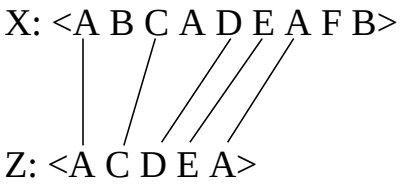
Μέγιστη κοινή ακολουθία

Ορισμός Με δεδομένη μία ακολουθία $X =< x_1, x_2, \cdots, x_m >$, μία άλλη ακολουθία $Z =< z_1, z_2, \cdots, z_k >$ είναι υπό-ακολουθία της X εάν υπάρχει μία γνησίως αύξουσα ακολουθία $< i_1, i_2, \cdots, i_k >$ από δείκτες της X τέτοια ώστε για όλα τα $j = 1 \cdots k$ ισχύει $x_{i_j} = z_j$.

Παράδειγμα $X =<$ A, B, C, A, D, E, A, F, B $>$

Υπό-ακολουθία της X	Ακολουθία δεικτών της X
$Z =<$ A, C, D, E, A $>$	$<$ 1, 3, 5, 6, 7 $>$
$Z =<$ A, B $>$	$<$ 1, 2 $>$, $<$ 1, 9 $>$, $<$ 4, 9 $>$

- Μια καλύτερη αναπαράσταση:

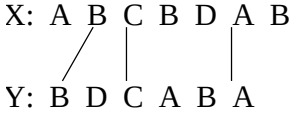


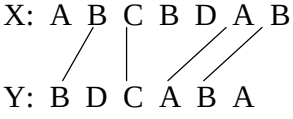
Το πρόβλημα της μέγιστης κοινής υπό-ακολουθίας (ΜΚΥ)
(The longest common subsequence (LCS) problem)

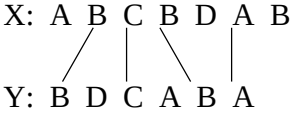
Είσοδος: Δυο ακολουθίες X και Y.

Έξοδος: Η μέγιστη κοινή υπό-ακολουθία.

Παραδείγματα







Γιατί μελετάμε το πρόβλημα της ΜΚΥ;

- Λύνεται με χρήση δυναμικού προγραμματισμού.
- Έχει αρκετές εφαρμογές.

Εφαρμογή

Με δεδομένες 2 διαφορετικές εκδόσεις (versions) ενός προγράμματος, να προσδιοριστούν οι αλλαγές που έγιναν από την πρώτη έκδοση έως τη δεύτερη.

Λύση

Βρείτε την ΜΚΥ των δύο προγραμμάτων. Το κείμενο που δεν είναι μέλος της ΜΚΥ προέρχεται από προσθέσεις ή αφαιρέσεις που έγιναν κατά την μετάβαση από την μία έκδοση προς την άλλη.

Μια απλοϊκή προσέγγιση

- Έστω S το σύνολο όλων των δυνατών υπό-ακολουθιών των X και Y.
- Έλεγξε κάθε υπό-ακολουθία του S και προσδιόρισε την ΜΚΥ.

Ερώτηση Ποιο είναι το μέγεθος του συνόλου S;

Απάντηση Έστω $k = \min(legth(X), length(Y))$. Τότε, $|S| = 2^k$.

Αυτό καθιστά την απλοϊκή προσέγγιση ανεδαφική!

Μία αναδρομική λύση

Έστω $X =< x_1, x_2, \cdots, x_m >$ και $Y =< y_1, y_2, \cdots, y_n >$.

Συμβόλισε με X_i την υπό-ακολουθία $X_i =< x_1, x_2, \cdots, x_i >$
και με Y_i την υπό-ακολουθία $Y_i =< y_1, y_2, \cdots, y_i >$.

Έστω $c[i, j]$ το μέγεθος της ΜΚΥ των X_i και Y_j.

Μας ενδιαφέρει να υπολογίσουμε το $c[m, n]$.

Μπορούμε να γράψουμε την αναδρομική σχέση:

$$c[i,j] = \begin{cases} 0 & \text{εάν } i = 0 \text{ ή } j = 0 \\ c[i-1, j-1] + 1 & \text{εάν } i, j > 0 \text{ και } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & \text{εάν } i, j > 0 \text{ και } x_i \neq y_j \end{cases}$$

Ερώτηση ι) Είναι η παραπάνω σχέση σωστή;
ιι) Πως καταλήξαμε σε αυτή;

Απάντηση ι) ΝΑΙ

ιι) $$c[i,j] = \begin{cases} 0 & \text{εάν } i = 0 \text{ ή } j = 0 \\ c[i-1, j-1] + 1 & \text{εάν } i, j > 0 \text{ και } x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & \text{εάν } i, j > 0 \text{ και } x_i \neq y_j \end{cases}$$

- εάν $i = 0$ ή $j = 0$:
  Τουλάχιστον μία από τις ακολουθίες είναι κενή. Άρα, η μόνη υπό-ακολουθία είναι η κενή υπό-ακολουθία (συμβολίζεται με το ϵ).
- εάν $i, j > 0$ και $x_i = y_j$:

$$\begin{array}{ll} & LCS(X_i, Y_j) = LCS(X_{i-1}, Y_{j-1}) \oplus x_i \\ < x_1, x_2, \cdots, x_{i-1}, x_i > & \Longrightarrow c[i,j] = c[i-1, j-1] + 1 \\ = & \\ < y_1, y_2, \cdots, y_{j-1}, y_j > & (\oplus \text{ συμβολίζει τη συνένωση συμβολοσειρών / ακολουθιών}) \end{array}$$

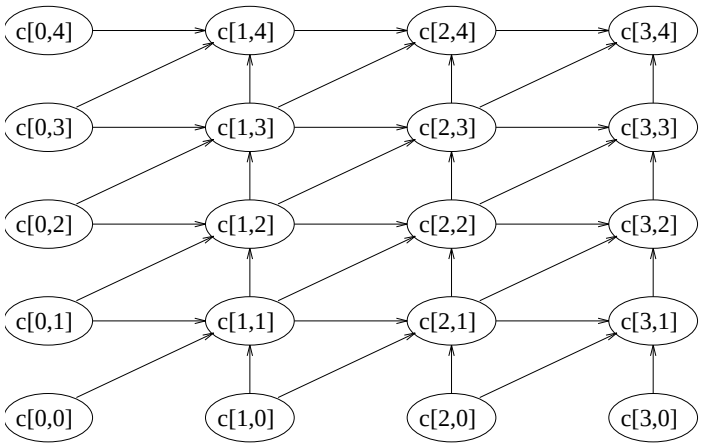
- εάν $i, j > 0$ και $x_i \neq y_j$:

$$\begin{array}{ll} & LCS[X_i, Y_j] = \begin{cases} LCS[X_i, Y_{j-1}] \\ \text{ή} \\ LCS[X_{i-1}, Y_j] \end{cases} \\ < x_1, x_2, \cdots, x_{i-1}, x_i > & \\ \neq & \\ < y_1, y_2, \cdots, y_{j-1}, y_j > & \Longrightarrow c[i,j] = \max(c[i, j-1], c[i-1, j]) \end{array}$$

Μία αναδρομική προσέγγιση

- Στην περίπτωση όπου οι 2 υπό-ακολουθίες δεν έχουν κοινές υπό-ακολουθίες (εκτός της ϵ), ο αλγόριθμος χρειάζεται εκθετικό χρόνο.

Παράδειγμα Σχεδιάστε το 'γράφημα αναδρομής' για το $C[3,4]$.



Άσκηση Να υπολογιστεί ο αριθμός των κλήσεων της διαδικασίας που υπολογίζει το $c[i,j]$ κατά τη διάρκεια του υπολογισμού του $c[m,n]$, όπου $0 \leq i \leq m, \quad 0 \leq j \leq n$.

- Προφανώς, υπάρχουν επικαλυπτόμενα υπό-προβλήματα τα οποία ο αλγόριθμος θα λύσει παραπάνω από μία φορές.

(Ο αναδρομικός αλγόριθμός χρειάζεται τουλάχιστον εκθετικό χρόνο.)

Παρατήρηση Το $c[m,n]$ μπορεί να υπολογιστεί εύκολα ξεκινώντας από τη βάση (bottom-up).

(Να συμπληρωθεί ο πίνακας c γραμμή-γραμμή από τα αριστερά προς τα δεξιά.)

$LCS\_length(X, Y)$

$m = length(X)$
$n = length(Y)$
for $i = 1$ **to** m **do** $c[i, 0] = 0$
for $j = 1$ **to** n **do** $c[0, j] = 0$
for $i = 1$ **to** m **do**
  for $j = 1$ **to** n **do**
    if $x_i = y_j$
      then $c[i, j] = c[i-1, j-1] + 1$
        $b[i, j] = \swarrow$
      else if $c[i-1, j] \geq c[i, j-1]$
        then $c[i, j] = c[i-1, j]$
          $b[i, j] = \downarrow$
        else $c[i, j] = c[i, j-1]$
          $b[i, j] = \leftarrow$
return $c,\ b$

Σημείωση Ο πίνακας b χρησιμοποιείται για την ανάκτηση της ΜΚΥ.

Παράδειγμα X= A B C B D A B
Y= B D C A B A

i	j	0	1 (B)	2 (D)	3 (C)	4 (A)	5 (B)	6 (A)
7	B	0	↙ 1	↓ 2	↓ 2	↓ 3	↙ 4	↓ 4
6	(A)	0	↓ 1	↓ 2	↓ 2	↙ 3	↓ 3	↙ 4
5	D	0	↓ 1	↙ 2	↓ 2	↓ 2	↓ 3	↓ 3
4	(B)	0	↙ 1	↓ 1	↓ 2	↓ 2	↙ 3	← 3
3	(C)	0	↓ 1	↓ 1	↙ 2	← 2	↓ 2	↓ 2
2	(B)	0	↙ 1	← 1	← 1	↓ 1	↙ 2	← 2
1	A	0	↓ 0	↓ 0	↓ 0	↙ 1	← 1	↙ 1
0		0	0	0	0	0	0	0

Ερώτηση 1 Ποια είναι η πολυπλοκότητα της LCS();

Απάντηση: $O(mn)$

Ερώτηση 2 Πόσος χρόνος χρειάζεται για την ανάκτηση της ΜΚΥ από τον πίνακα b;

Απάντηση: $O(m+n)$

Ερώτηση 3 Πόση μνήμη απαιτείται για την εκτέλεση της LCS();

Απάντηση: $O(mn)$

Ερώτηση 4 Είναι δυνατό να βελτιωθεί η πολυπλοκότητα χώρου στην περίπτωση όπου θέλουμε να υπολογίσουμε μόνο το μήκος της ΜΚΥ;

Απάντηση: Ναι! Μόνο 2 γραμμές του πίνακα c είναι απαραίτητοι.
$\Longrightarrow$ $O(\min(m,n))$ μνήμη.

Σημείωση: Στην περίπτωση αυτή δεν μπορεί να ανακτηθεί η ΜΚΥ.