Αν. Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

Οικονομικό Πανεπιστήμιο Αθηνών

Ευθυγράμμιση Ακολουθιών

Στη μοριακή βιολογία και στην βιοπληροφορική, μελετούμε, ανάμεσα στα άλλα, αλυσίδες βάσεων που δημιουργούν τις αλυσίδες των νουκλεϊκών οξέων, DNA και RNA. Στην περίπτωση του DNA οι βάσεις αυτές είναι η Αδενίνη (Adenine, A), η Κυτοσίνη (Cytosine, C), η Γουανίνη (Guanine, G) και η Θυμίνη (Thymine, T). Στην περίπτωση του RNA αντί της Θυμίνης έχουμε την Ουρακίλη (Uracil, U). Έχουμε λοιπόν στη διάθεσή μας αλυσίδες από αυτές τις βάσεις (όπως "GATTACA", "GCATGCG") και θέλουμε να δούμε πόσο μοιάζουν ή διαφέρουν μεταξύ τους, ώστε να εντοπιστούν ομοιότητες, μεταλλάξεις, κ.λπ.

Αυτές τις αλυσίδες μπορούμε να τις αντιμετωπίσουμε ως ακολουθίες (sequences), οπότε το πρόβλημά μας είναι να μπορούμε να *ευθυγραμμίσουμε* δύο ακολουθίες: δηλαδή, να δούμε σε ποια σημεία είναι ίδιες, σε ποια σημεία διαφέρουν, και ποια σημεία της μίας ακολουθίας λείπουν από την άλλη. Τις δύο παραπάνω ακολουθίες μπορούμε να τις ευθυγραμμίσουμε ως:

```
G – A T T A C A
G C A – T G C G
```

ή ως:

```
G – A T T A C A
G C A T – G C G
```

ή ως:

```
G – A T T A C A
G C A T G – C G
```

Πώς τις βρίσκουμε όμως τις ευθυγραμμίσεις, δεδομένου ότι αυτές μπορεί να έχουν πολλά στοιχεία; Αν οι δύο ακολουθίες που θέλουμε να ευθυγραμμίσουμε είναι ίδιες, προφανώς η ευθυγράμμιση είναι απευθείας οι ίδιες οι ακολουθίες. Αν όμως διαφέρουν, θα πρέπει με κάποιον τρόπο να ορίσουμε με ποιον τρόπο επιθυμούμε να τις ταιριάξουμε. Μπορούμε να τις ταιριάξουμε διαγράφοντας τα στοιχεία που διαφέρουν, ή αλλάζοντας τα στοιχεία που διαφέρουν, ενώ θέλουμε να κρατήσουμε όσα κοινά στοιχεία είναι δυνατόν.

Σε κάθε ένα διαφορετικό ενδεχόμενο θα προσδώσουμε μία διαφορετική τιμή. Με τον τρόπο αυτό κάθε δυνατή ευθυγράμμιση θα έχει μια βαθμολογία και θα επιλέξουμε την ευθυγράμμιση με τη μεγαλύτερη βαθμολογία (ή τις ευθυγραμμίσεις με τις μεγαλύτερες βαθμολογίες σε περίπτωση ισοβαθμίας).

Όταν ξεκινάμε, οπότε δεν έχουμε βρει ακόμα ευθυγράμμιση, η βαθμολογία είναι 0.

1. Αν δύο στοιχεία είναι τα ίδια, θα προσθέτουμε m (match) στη βαθμολογία της ευθυγράμμισης.
2. Αν δύο στοιχεία είναι διαφορετικά, θα αφαιρούμε d (differ) από τη βαθμολογία της ευθυγράμμισης.
3. Αν διαγράψουμε ένα στοιχείο, θα αφαιρούμε g (gap) από τη βαθμολογία της ευθυγράμμισης.

Φτιάχνουμε έναν πίνακα F με τόσες γραμμές γραμμές όσα τα στοιχεία της πρώτης ακολουθίας συν ένα και τόσες στήλες όσες τα στοιχεία της δεύτερης ακολουθίας συν ένα. Στον πίνακα αυτό γεμίζουμε την πρώτη γραμμή και την πρώτη στήλη με τον τύπο $g \times i$, όπου i είναι ο αριθμός του κελιού στη γραμμή ή στη στήλη. Ας πάρουμε ως παράδειγμα τις ακολουθίες "CTAAC" και "ACTGACG":

		A	C	T	G	A	C	G
	0	−2	−4	−6	−8	−10	−12	−14
C	−2							
T	−4							
A	−6							
A	−8							
C	−10							

Αν ονομάσουμε την πρώτη ακολουθία A και τη δεύτερη B, το κάθε κελί (i, j) του πίνακα θα μας δείχνει το κόστος της ευθυγράμμισης της ακολουθίας $\mathrm{A}[0 \dots i+1]$ και $B[0 \dots j+1]$. Έτσι, το κελί $(0, 2)$ μας δείχνει το κόστος της ευθυγράμμισης της ακολουθίας "" από την "CTAAC" με την ακολουθία "AC" από την "ACTGACG". Για να ευθυγραμμιστούν οι δύο αυτές ακολουθίες θα πρέπει να διαγραφούν τα δύο στοιχεία της "AC" από την "ACTGACG" με κόστος 2×-2. Ομοίως, το κελί $(3, 0)$ μας δείχνει το κόστος της ευθυγράμμισης της ακολουθίας "CTA" από την "CTAAC" και "" από τη "ACTGACG". Για να ευθυγραμμιστούν οι δύο αυτές ακολουθίες θα πρέπει να διαγραφούν τα τρία στοιχεία της "CTA" από την "CTAAC με κόστος 3×-2.

Προχωράμε τώρα να γεμίσουμε τον υπόλοιπο πίνακα, από πάνω προς τα κάτω και από αριστερά προς τα δεξιά. Στο κελί (i, j) ελέγχουμε τις τρεις περιπτώσεις που αναφέραμε παραπάνω:

1. Αν δύο στοιχεία είναι τα ίδια, θα προσθέτουμε m (match) στη βαθμολογία της ευθυγράμμισης, επομένως η βαθμολογία της ευθυγράμμισης θα είναι $F[i-1, j-1] + m$.
2. Αν δύο στοιχεία είναι διαφορετικά, θα αφαιρούμε d (differ) από τη βαθμολογία της ευθυγράμμισης, επομένως η βαθμολογία της ευθυγράμμισης θα είναι $F[i-1, j-1] - d$.
3. Αν διαγράψουμε ένα στοιχείο, θα αφαιρούμε g (gap) από τη βαθμολογία της ευθυγράμμισης. Αν διαγράψουμε στοιχείο από την πρώτη ακολουθία, η βαθμολογία της ευθυγράμμισης θα είναι $F[i-1, j] - g$. Αν διαγράψουμε

στοιχείο από τη δεύτερη ακολουθία, η βαθμολογία της ευθυγράμμισης θα είναι $F[i, j-1] - g$.

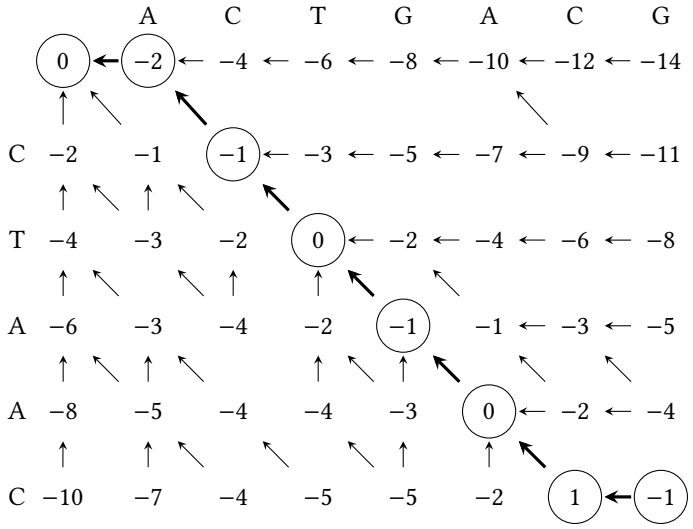
Εφόσον θέλουμε την ευθυγράμμιση με τη μεγαλύτερη βαθμολογία, στο κελί (i, j) θα βάλουμε την μεγαλύτερη τιμή που προκύπτει από τις τρεις παραπάνω περιπτώσεις. Βλέπουμε λοιπόν ότι πράγματι ο πίνακας μπορεί να γεμίσει εξετάζοντας, για κάθε στοιχείο (i, j) το στοιχείο διαγώνια επάνω, $(i-1, j-1)$, επάνω, $(i, j-1)$, αριστερά, $(i-1, j)$, δηλαδή πηγαίνοντας από πάνω προς το κάτω και από τα αριστερά προς τα δεξιά. Συνοψίζοντας, οι τιμές του πίνακα F προκύπτουν ως εξής:

$$F[i,j] = \begin{cases} 0, \text{αν } i = 0 \text{ και } j = 0 \\ (-g) \times i, \text{αν } j = 0 \\ (-g) \times j, \text{αν } i = 0 \\ \max \begin{cases} F[i-1, j] - g \\ F[i, j-1] - g \\ F[i-1, j-1] + m, \text{αν } A[i] = B[j] \\ F[i-1, j-1] - d, \text{αν } A[i] \neq B[j] \end{cases} \end{cases}$$

Στο παράδειγμά μας, αν δουλέψουμε με αυτόν τον τρόπο και έχουμε $g = 2$, $m = 1$, $d = 1$, ο πίνακας F θα γεμίσει ως εξής:

		A	C	T	G	A	C	G
	0 ←	−2 ←	−4 ←	−6 ←	−8 ←	−10 ←	−12 ←	−14
	↑ ↖	↖				↖		
C	−2	−1	−1 ←	−3 ←	−5 ←	−7 ←	−9 ←	−11
	↑ ↖	↑ ↖	↖					
T	−4	−3	−2	0 ←	−2 ←	−4 ←	−6 ←	−8
	↑ ↖	↖	↑	↑ ↖	↖			
A	−6	−3	−4	−2	−1	−1 ←	−3 ←	−5
	↑ ↖	↑ ↖		↑ ↖	↑ ↖	↖	↖	
A	−8	−5	−4	−4	−3	0 ←	−2 ←	−4
	↑	↑ ↖	↖	↖	↑	↑ ↖		
C	−10	−7	−4	−5	−5	−2	1 ←	−1

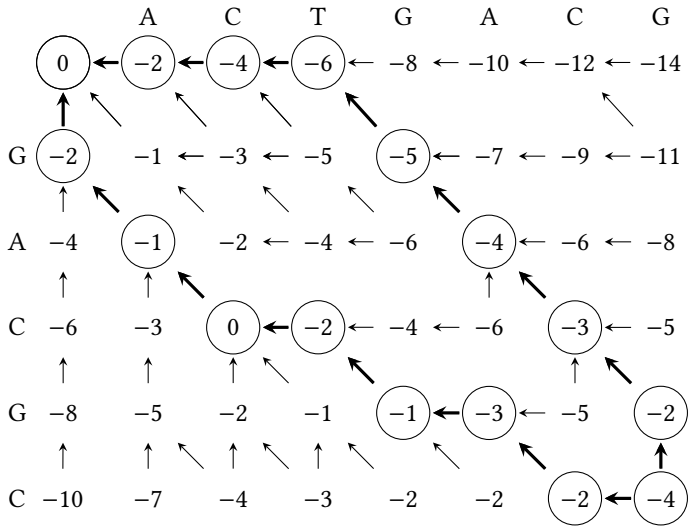
Τα βέλη σε κάθε κελί δείχνουν πώς προκύπτει η τιμή του· προσέξτε ότι μπορεί η τιμή να προκύπτει με παραπάνω από έναν τρόπο. Αφού κατασκευάσουμε τον πίνακα, τότε οι δυνατές ευθυγραμμίσεις των ακολουθιών προκύπτουν από τις δυνατές διαδρομές από την κάτω δεξιά γωνία του πίνακα στην πάνω αριστερή. Στο παράδειγμά μας, υπάρχει μία τέτοια διαδρομή, η οποία φαίνεται παρακάτω:



Η διαδρομή αυτή αντιστοιχεί στην ευθυγράμμιση:

–	C	T	A	A	C	–
A	C	T	G	A	C	G

Ας πάρουμε ένα άλλο παράδειγμα, τις ακολουθίες “GACGC” και “ACTGACG”. Ο πίνακας αυτή τη φορά θα είναι ο παρακάτω:



Όπως βλέπουμε, υπάρχουν δύο μονοπάτια από την κάτω δεξιά στην άνω αριστερή γωνία, τα οποία αντιστοιχούν στην ευθυγράμμιση:

–	–	–	G	A	C	G	C
A	C	T	G	A	C	G	–

και:

G	A	C	–	G	–	C	–
–	A	C	T	G	A	C	G

Για να βρούμε λοιπόν τις ευθυγραμμίσεις, μπορούμε να κατασκευάσουμε τον πίνακα όπως περιγράψαμε και καθώς τον φτιάχνουμε να δημιουργήσουμε ένα γράφο με κόμβους τα κελιά του πίνακα και συνδέσμους τα βέλη που αντιστοιχούν στον τρόπο που προκύπτει η τιμή του κάθε κελιού. Αφού κατασκευάσουμε τον πίνακα, η απαρίθμηση όλων των μονοπατιών από κάτω δεξιά μέχρι και πάνω αριστερά θα μας δώσει τις ευθυγραμμίσεις.

Υπάρχει και ένας άλλος τρόπος να απαριθμήσουμε τις δυνατές ευθυγραμμίσεις, χωρίς να κατασκευάσουμε τον γράφο. Από τον πίνακα που φτιάξαμε, μπορούμε να ξεκινήσουμε από το κάτω δεξί του άκρο και να εξετάσουμε με ποιον τρόπο μπορεί να προκύψει η τιμή του από το κελί που βρίσκεται από πάνω του, διαγώνια και από πάνω του, ή αριστερά του. Προσαρμόζουμε αντίστοιχα τις δυνατές ευθυγραμμίσεις και συνεχίζουμε αναδρομικά προχωρόντας προς το πάνω αριστερό άκρο του πίνακα. Σε μορφή ψευδοκώδικα ο αλγόριθμος μπορεί να περιγραφεί ως εξής:

`EnumerateAlignments`(A, B, F, W, Z)
Input: A, the first sequence to be aligned
B, the second sequence to be aligned
F, the alignment matrix
W, the alignment of A being constructed
Z, the alignment of B being constructed
Data: WW, a list
ZZ, a list
Result: WW and ZZ contain all the alignments of A and B

1 $i \leftarrow |A|$
2 $j \leftarrow |B|$
3 **if** $i = 0$ **and** $j = 0$ **then**
4 `AppendToList`(WW, W)
5 `AppendToList`(ZZ, Z)
6 **return**
7 **if** $i > 0$ **and** $j > 0$ **then**
8 $m \leftarrow$ `Compare`$(A[i-1], B[j-1])$
9 **if** $F[i, j] = F[i-1, j-1] + m$ **then**
10 `EnumerateAlignments`$(A[0 \ldots i-1], B[0 \ldots j-1], F, A[i-1] + W, B[j-1] + Z)$
11 **if** $i > 0$ **and** $F[i, j] = F[i-1, j] + g$ **then**
12 `EnumerateAlignments`$(A[0 \ldots i-1], B, F, A[i-1] + W, \text{“}-\text{”} + Z)$
13 **if** $j > 0$ **and** $F[i, j] = F[i, j-1] + g$ **then**
14 `EnumerateAlignments`$(A, B[0 \ldots j-1], F, \text{“}-\text{”} + W, B[j-1] + Z)$

Ο αλγόριθμος απαριθμεί μία-μία τις δυνατές ευθυγραμμίσεις, τις οποίες αποθηκεύει σε δύο λίστες *WW* και *ZZ*. Κατά την εξέλιξη της αναδρομής, η κάθε ευθυγράμμιση χτίζεται στις παραμέτρους W και Z, που αρχικά είναι κενές.

Η κατασκευή του πίνακα και η απαρίθμηση των ευθυγραμμίσεων αποτελούν τον αλγόριθμο Needleman-Wunsch, ο οποίος ονομάστηκε από τους δύο δημιουργούς του Saul B. Needleman και Christian D. Wunsch. Ο αλγόριθμος Needleman-Wunsch τρέχει σε χρόνο $\mathrm{O}(mn)$, όπου m είναι το μήκος της ακολουθίας A και B είναι το μήκος της ακολουθίας B, άρα είναι αποτελεσματικός όσον αφορά τον χρόνο. Για την εκτέλεσή του απαιτεί χώρο mn, για την κατασκευή του πίνακα F. Αυτό όμως αποτελεί ένα πρόβλημα. Αν θέλουμε να ευθυγραμμίσουμε ακολουθίες με πολλά στοιχεία, ο πίνακας F θα είναι πάρα πολύ μεγάλος. Θα θέλαμε λοιπόν έναν τρόπο ώστε από τη μία να βελτιώσουμε τις απαιτήσεις σε χώρο, από την άλλη να μη θυσιάσουμε σημαντικά τις επιδόσεις σε ταχύτητα.

Το πρώτο πράγμα που μπορούμε να προσέξουμε είναι ότι για να κατασκευάσουμε τον πίνακα F χρειαζόμαστε κάθε φορά μόνο δύο γραμμές: την τρέχουσα και την προηγούμενη. Έτσι, αν θέλουμε να υπολογίσουμε τη βαθμολογία της ευθυγράμμισης, αρκεί να χρησιμοποιήσουμε τον παρακάτω αλγόριθμο:

`ComputeAlignmentScore`(A, B, `Compare`, g) $\rightarrow L$

Input: A, the first sequence to be aligned
B, the second sequence to be aligned
`Compare`, a function calculating the match or difference score between sequence items
g, the negative gap score

Output: L, the last line of the alignment matrix

```
1  L ← CreateArray(|B| + 1)
2  for j ← 0 to |L| do
3      L[j] ← j × g
4  K ← CreateArray(|B| + 1)
5  for i ← 1 to |A| + 1 do
6      Swap(L, K)
7      L[0] ← i × g
8      for j ← 1 to |B| + 1 do
9          md ← Compare(A[i − 1], B[j − 1])
10         L[j] ← Max(L[j − 1] + g, K[j] + g, K[j − 1] + md)
11 return L
```

Ο αλγόριθμος μας επιστρέφει μια σειρά αριθμών L, την τελευταία γραμμή του πίνακα F. Κάθε στοιχείο $L[j]$ μας δίνει τη βαθμολογία της ευθυγράμμισης της ακολουθίας A με το πρόθεμα $\mathrm{B}[0 \ldots j]$· το τελευταίο στοιχείο της L μας δίνει τη βαθμολογία της ευθυγράμμισης της ακολουθίας A με όλη την ακολουθία B.

Με τον τρόπο αυτό υπολογίζουμε μεν τη βαθμολογία της ευθυγράμμισης, αλλά

δεν έχουμε τρόπο να βρούμε τις ευθυγραμμίσεις, καθώς δεν διατηρούμε κάπου τον πίνακα F και δεν μπορούμε να τον διατρέξουμε. Και ακριβώς αυτό που θέλουμε να αποφύγουμε είναι η κατασκευή του πίνακα F.

Στο σημείο αυτό λοιπόν σκεφτόμαστε με ποιον τρόπο θα μπορούσαμε να σπάσουμε το πρόβλημά μας σε μικρότερα προβλήματα, ώστε να μην χρειάζεται ποτέ να κατασκευαστεί ένας πίνακας μεγέθους $m \times n$. Διασπάμε την ακολουθία A στα μέση, παίρνοντας δύο ακολουθίες, την A_l και την A_r ώστε $A = A_l + A_r$ και $|\mathrm{A}_l| = \lfloor |A|/2 \rfloor$, $|A_r| = |A| - |A_l|$. Θα εξετάσουμε αν αντί να λύσουμε το αρχικό πρόβλημε με όλη την ακολουθία A, μπορούμε να λύσουμε δύο υποπροβλήματα, ένα με την A_l και ένα με την A_r.

Η ακολουθία A_l μπορεί να ευθυγραμμιστεί με την αρχή της B. Ομοίως, η ακολουθία A_r μπορεί να ευθυγραμμιστεί με το τέλος της B. Υπολογίζουμε τη βαθμολογία της ευθυγράμμισης της A_l με όλη τη B με τον παραπάνω αλγόριθμο. Ως αποτέλεσμα θα πάρουμε μία γραμμή, ή ακολουθία, ας την ονομάσουμε S_l. Όπως και πριν, κάθε στοιχείο $S_l[j]$ μας δίνει τη βαθμολογία της ευθυγράμμισης της ακολουθίας A_l με το πρόθεμα $B[0 \ldots j]$. Για μία ακολουθία X, ας συμβολίσουμε με $\tilde{X}$ την αντίστροφή της. Τότε μπορούμε να υπολογίσουμε τη βαθμολογία της ευθυγράμμισης της αντίστροφης της A_r, δηλαδή της $\tilde{A}_r$ με την αντίστροφη της B, δηλαδή τη $\tilde{B}$. Ως αποτέλεσμα θα πάρουμε μια γραμμή, ας την ονομάσουμε S_r. Κάθε στοιχείο $S_r[j]$ μας δίνει τη βαθμολογία της ευθυγράμμισης της ακολουθίας $\tilde{A}_r$ με το πρόθεμα $\tilde{B}[0 \ldots j]$. Ή, με άλλα λόγια, κάθε στοιχείο $\tilde{S}_r[j]$ μας δίνει τη βαθμολογία της ευθυγράμμισης της ακολουθίας A_r με την κατάληξη $B[j \ldots |B|]$. Αν τότε κάνουμε την πρόσθεση $S = S_l + \tilde{S}_r$ κάθε $S[j]$ μας δίνει τη βαθμολογία της ευθυγράμμισης της A_l με το πρόθεμα $B[0 \ldots j]$ και της ευθυγράμμισης της A_r με την κατάληξη $B[j \ldots |B|]$. Επομένως, το μέγιστο στοιχείο του S μας δίνει το καλύτερο αποτέλεσμα που μπορούμε να πετύχουμε διασπώντας τη B. Ας ονομάσουμε τη θέση αυτού του στοιχείου $j_{\max}$. Αυτό μας δίνει την απάντηση στο ερώτημά μας: θα διασπάσουμε την A στη μέση, και τη B στο σημείο $j_{\max}$. Για να ευθυγραμμίσουμε την A με τη B θα ευθυγραμμίσουμε την A_l με τη $B[0 \ldots j_{\max}]$ και την A_r με την $B[j_{\max} \ldots |B|]$.

Πώς όμως θα βρούμε αυτό το πρόθεμα B_1 της B; Για να το βρούμε, θα υπολογίσουμε το κόστος της ευθυγράμμισης της A_1 με τη B και της αντίστροφης της A_2 με την αντίστροφη της B. Γιατί τις αντίστροφες; Γιατί με αυτόν τον τρόπο θα βρίσκουμε ευθυγραμμίσεις των καταλήξεων των A_2 και B, όπως θέλουμε. Η θέση στην οποία θα κόψουμε τη B για να πάρουμε τα B_1 και B_2 θα είναι η θέση με τη μεγαλύτερη δυνατή βαθμολογία ευθυγράμμισης· στην περίπτωση ισοβαθμίας, θα πρέπει να εξετάσουμε όλες τις ισόβαθμες μέγιστες θέσεις. Αφού διασπάμε τη B στη θέση με τη μέγιστη βαθμολογία, θα πάρουμε συνολικά της δύο καλύτερες επιμέρους ευθυγραμμίσεις. Αποδεικνύεται ότι η συνένωσή τους είναι βέλτιστη ευθυγράμμιση όλης της A με όλη τη B.

Η ιδέα αυτή της επίλυσης του προβλήματος με διάσπαση σε ολοένα και μικρότερα μπορεί να υλοποιηθεί με τον παρακάτω αλγόριθμο που επινόησε ο Dan Hirschberg:

```
Hirschberg(A, B)
    Input: A, the first sequence to be aligned
           B, the second sequence to be aligned
    Output: WW, all the alignments of A with B
            ZZ, all the alignments of B with A corresponding to WW
1   if |A| = 0 then
2       WW ← " − " × |B|
3       ZZ ← B
4   else if |B| = 0 then
5       WW ← A
6       ZZ ← " − " × |A|
7   else if |A| = 1 or |B| = 1 then
8       (WW, ZZ) ← NeedlemanWunsch(A, B)
9   else
10      i ← ⌊|A|/2⌋
11      S_l ← ComputeAlignmentScore(A[0 ... i], B)
12      S_r ← ComputeAlignmentScore(Ã[i ... |A|], B̃)
13      S ← S_l + S̃_r
14      J ← Max(S)
15      WW ← CreateList()
16      ZZ ← CreateList()
17      foreach j in J do
18          (WW_l, ZZ_l) ← Hirschberg(A[0 ... i], B[0 ... j])
19          (WW_r, ZZ_r) ← Hirschberg(A[i ... |A|], B[j ... |B|])
20          UpdateAlignments(WW, ZZ, WW_l + WW_r, ZZ_l + ZZ_r)
21  return (WW, ZZ)
```

Αν μελετήσουμε τον αλγόριθμο βλέπουμε ότι πράγματι διασπάει το πρόβλημα σε ολοένα και μικρότερα προβλήματα, μέχρις ότου η ακολουθία A ή η B να μην έχουν μήκος μεγαλύτερο από ένα, το οποίο σημαίνει ότι δεν θα δημιουργηθεί ποτέ ένας πίνακας $m \times n$. Ο αλγόριθμος είναι αναδρομικός, με την αναδρομή να σταματάει είτε αν έχει εξαντληθεί η ακολουθία A (γραμμές 1-3), είτε όταν έχει εξαντληθεί η ακολουθία B (γραμμές 4-6), είτε αν η A ή και η B έχουν μήκος ένα. Στην τελευταία περίπτωση εκτελείται ο αλγόριθμος Needleman-Wunsch, ο οποίος όμως θα κατασκευάσει έναν πίνακα F που αν $|A| > 1$ θα έχει δύο στήλες και αν $|B| > 1$ θα έχει δύο γραμμές. Οι γραμές 10-21 υλοποιούν τη λογική που περιγράψαμε. Στη γραμμή 13 προσθέτουμε τις βαθμολογίες μία προς μία και στη γραμμή 14 βρίσκουμε τις θέσεις των μέγιστων τιμών του S (μπορεί να είναι περισσότερες από μία λόγω ισοβαθμίας). Οι αναδρομικές κλήσεις γίνονται στις γραμμές 18 και 19 και τα αποτελέσματα μας συνθέτουν τις ευθυγραμμίσεις τις οποίες συγκεντρώνουμε στις λίστες WW και ZZ στη γραμμή 20 εξασφαλίζοντας ότι οι ευθυγραμμίσεις είναι μοναδικές.

Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*
- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2022-2`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `hirschberg.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
hirschberg.py [-t] [-f] [-l] gap match differ a b
```

Η σημασία των παραμέτρων είναι η εξής:

- Η παράμετρος `-t` σημαίνει ότι κατά την εκτέλεση του προγράμματος θα τυπώνει το κάθε (i, j) που προκύπτει μέσα στο βρόχο των γραμμών 17-20 του αλγορίθμου Hirschberg.
- Η παράμετρος `-f` σημαίνει ότι οι παράμετροι `a` και `b` θα είναι ονόματα αρχείων, τα περιεχόμενα των οποίων θα πρέπει να ευθυγραμμιστούν. Διαφορετικά οι παράμετροι `a` και `b` θα είναι συμβολοσειρές που θα πρέπει να ευθυγραμμιστούν.
- Η παράμετρος `-l` σημαίνει ότι στα αρχεία που θα διαβαστούν λόγω της παραμέτρου `-f` τα στοιχεία της κάθε ακολουθίας θα είναι ολόκληρες οι

γραμμές κάθε αρχείου. Έτσι τα περιεχόμενα των `a` και `b` θα ευθυγραμμιστούν γραμμή προς γραμμή και όχι χαρακτήρα προς χαρακτήρα.

- Η παράμετρος `gap` θα δίνει την τιμή $-g$ για την κατασκευή του πίνακα F.
- Η παράμετρος `match` θα δίνει την τιμή m για την κατασκευή του πίνακα F.
- Η παράμετρος `differ` θα δίνει την τιμή $-d$ για την κατασκευή του πίνακα F.

Η έξοδος του προγράμματος θα εμφανίζει όλες τις ευθυγραμμίσεις που μπορούν να βρεθούν, με τον τρόπο που επιδεικνύεται στα παραδείγματα που ακολουθούν.

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -2 +2 -1 AGTACGCA TATGC
```

τότε η έξοδος του προγράμματος θα είναι:

```
AGTACGCA
--TATGC-
```

Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -2 1 -1 GACGC ACTGACG
```

η έξοδος του προγράμματος θα είναι:

```
GAC-G-C-
-ACTGACG

---GACGC
ACTGACG-
```

δηλαδή, οι δύο δυνατές ευθυγραμμίσεις, χωρισμένες με μία κενή γραμμή.

Παράδειγμα 3

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -t -2 1 -1 GACGC ACTGACG
```

η έξοδος του προγράμματος θα είναι:

```
2, 1
1, 1
1, 2
1, 3
1, 2
```

```
1, 1
1, 2
2, 5
1, 4
1, 1
GAC-G-C-
-ACTGACG

---GACGC
ACTGACG-
```

δηλαδή, όπως το προηγούμενο παράδειγμα, αλλά πριν από τις ευθυγραμμίσεις θα εμφανίζονται οι τιμές (i, j) σε κάθε επανάληψη του βρόχου των γραμμών 17-20 του αλγορίθμου Hirschberg.

Παράδειγμα 4

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -1 1 -1 GATTACA GCATGCG
```

η έξοδος του προγράμματος θα είναι:

```
G-ATTACA
GCA-TGCG

G-ATTACA
GCAT-GCG

G-ATTACA
GCATG-CG
```

Παράδειγμα 5

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -t -1 +1 -1 "deep end" depend
```

η έξοδος του προγράμματος θα είναι:

```
4, 3
2, 1
1, 1
2, 2
1, 1
2, 1
1, 1
deep end
d-ep-end
```

```
deep end
de-p-end
```

Παράδειγμα 6

Αν ο χρήστης του προγράμματος δώσει:

```
python hirschberg.py -f -l -2 1 -1 a.txt b.txt
```

τότε το πρόγραμμα θα διαβάσει το αρχείο a.txt και το αρχείο b.txt και θα χρησιμοποιήσει ως ακολουθία A τις γραμμές του αρχείου `a.txt` και ως ακολουθία B τις γραμμές του αρχείου `b.txt`. Θα ευθυγραμμίσει τις δύο ακολουθίες και στην έξοδό του θα εμφανίσει τις διαφορές των δύο αρχείων, γραμμή προς γραμμή. Αν διαγράφεται μια γραμμή θα εμφανίζει `-`, αν δύο γραμμές είναι ίδιες θα εμφανίζει την κάθε μία από τις δύο γραμμές πρόθεμα `=`, και αν οι δύο γραμμές είναι διαφορετικές θα εμφανίζει τη γραμμή του αρχείου `a.txt` με το πρόθεμα `<` και τη γραμμή του αρχείου `b.txt` με το πρόθεμα `>`. Για τα δύο παραπάνω αρχεία, η έξοδος θα είναι:

```
< -
> This is an important
< -
> notice! It should
< -
> therefore be located at
< -
> the beginning of this
< -
> document!
< -
>
= This part of the
= This part of the
= document has stayed the
= document has stayed the
= same from version to
= same from version to
= version.  It shouldn't
= version.  It shouldn't
= be shown if it doesn't
= be shown if it doesn't
= change.  Otherwise, that
= change.  Otherwise, that
= would not be helping to
= would not be helping to
= compress the size of the
= compress the size of the
= changes.
```

```
= changes.
<
> -
< This paragraph contains
> -
< text that is outdated.
> -
< It will be deleted in the
> -
< near future.
> -
=
=
= It is important to spell
= It is important to spell
< check this dokument. On
> check this document. On
= the other hand, a
= the other hand, a
= misspelled word isn't
= misspelled word isn't
= the end of the world.
= the end of the world.
= Nothing in the rest of
= Nothing in the rest of
= this paragraph needs to
= this paragraph needs to
= be changed. Things can
= be changed. Things can
= be added after it.
= be added after it.
< -
>
< -
> This paragraph contains
< -
> important new additions
< -
> to this document.
< -
> This is an important
< -
> notice! It should
< -
> therefore be located at
< -
```

```
> the beginning of this
< -
> document!
< -
>
= This part of the
= This part of the
= document has stayed the
= document has stayed the
= same from version to
= same from version to
= version.  It shouldn't
= version.  It shouldn't
= be shown if it doesn't
= be shown if it doesn't
= change.  Otherwise, that
= change.  Otherwise, that
= would not be helping to
= would not be helping to
= compress the size of the
= compress the size of the
= changes.
= changes.
=
=
< This paragraph contains
> -
< text that is outdated.
> -
< It will be deleted in the
> -
< near future.
> -
<
> -
= It is important to spell
= It is important to spell
< check this dokument. On
> check this document. On
= the other hand, a
= the other hand, a
= misspelled word isn't
= misspelled word isn't
= the end of the world.
= the end of the world.
= Nothing in the rest of
```

```
= Nothing in the rest of
= this paragraph needs to
= this paragraph needs to
= be changed. Things can
= be changed. Things can
= be added after it.
= be added after it.
< -
>
< -
> This paragraph contains
< -
> important new additions
< -
> to this document.
```

Περισσότερες Πληροφορίες

Οι Needleman και Wunsch εφηύραν τον επώνυμο αλγόριθμό τους για να εντοπίζουν ομοιότητες στις ακολουθίες αμινοξέων των πρωτεϊνών. Και οι δύο ήταν μοριακοί βιολόγοι· οι αλγόριθμοι δεν είναι μόνο αντικείμενο των επιστημόνων πληροφορικής. Για περισσότερα, δείτε το άρθρο Needleman and Wunsch (1970). Ο αλγόριθμος του Hirschberg δημοσιεύθηκε στο Hirschberg (1975).

Hirschberg, D. S. 1975. "A Linear Space Algorithm for Computing Maximal Common Subsequences." *Communications of the ACM* 18 (6): 341–43. https://doi.org/10.1145/360825.360861.

Needleman, Saul B., and Christian D. Wunsch. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *Journal of Molecular Biology* 48 (3): 443–53. https://doi.org/10.1016/0022-2836(70)90057-4.

Καλή Επιτυχία!