ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «Βιοπληροφορική»

Τίτλος	***Απαλλακτική Εργασία Εαρινού Εξαμήνου 2025***
Όνομα φοιτητή – Αρ. Μητρώου	Λαζαρίδης Χρήστος-Λάζαρος - Π22083
	Καλογερόπουλος Αθανάσιος - Π22223
Ημερομηνία παράδοσης	11/6/2025

Εργασία Βιοπληροφορικής, εαρινό εξάμηνο ακαδημαϊκού έτους 2024-2025

Η εργασία είναι απαλλακτική και εκπονείται σε ομάδες 1-2 φοιτητών. Μπορεί επίσης να παραδοθεί στην εξεταστική Σεπτεμβρίου. Δεν θα γίνουν δεκτές εκπρόθεσμες εργασίες ή εργασίες που δεν θα παραδοθούν μέσω e-class.

Παραδοτέα μέσω της ενότητας εργασιών στο e-class:

a. η τεκμηρίωση της εργασίας σε ένα αρχείο documentation.pdf, στην πρώτη σελίδα του οποίου αναγράφονται τα ονοματεπώνυμα των φοιτητών και οι ΑΜ. Δεν θα βαθμολογηθούν εργασίες που δεν περιέχουν τεκμηρίωση ή που δεν αναφέρουν τα ονόματα των μελών της ομάδας στην τεκμηρίωση
b. τα αρχεία source code σε ένα συμπιεσμένο αρχείο με όνομα source2025.zip (ή .rar ή άλλη σχετική κατάληξη)
c. οποιαδήποτε άλλα συνοδευτικά αρχεία η ομάδα κρίνει απαραίτητα σε ένα συμπιεσμένο αρχείο με το όνομα auxiliary2025.zip (ή .rar ή άλλη σχετική κατάληξη).

Θέμα

I. Δίνονται τα παρακάτω patterns από το αλφάβητο A, C, G, T: pattern1=ATTAGA, pattern2=ACGCATTT, pattern3=AGGACTCAA και pattern4=ATTTCAGT. Σχεδιάστε και υλοποιήστε μία μέθοδο σύνθεσης συμβολοσειράς, η οποία λειτουργεί ως εξής: α) επιλέγει ένα έως τρία σύμβολα με τυχαίο τρόπο και τα τοποθετεί στην αρχή της συμβολοσειράς. β) Επιλέγει κάθε ένα από τα παραπάνω patterns μία φορά, με τη σειρά που αναγράφονται και αντικαθιστά το πολύ δύο σύμβολα σε τυχαίες θέσεις, είτε με ένα άλλο τυχαία επιλεγμένο σύμβολο (για κάθε θέση ξεχωριστά) είτε με την κενή συμβολοσειρά (διαγραφή συμβόλου). Ότι προκύπτει συνενώνεται με την υφιστάμενη έως εκείνη τη στιγμή συμβολοσειρά. γ) Προσθέτει ένα έως δύο τυχαία σύμβολα στο τέλος της συμβολοσειράς. Δημιουργήστε συνολικά 100 συμβολοσειρές με τον αλγόριθμο σύνθεσης. Διαλέξτε με τυχαίο τρόπο 10 συμβολοσειρές και τοποθετήστε τις σε ένα σύνολο datasetA, 70 σε ένα σύνολο datasetB και τις υπόλοιπες 20 σε ένα σύνολο datasetC.

II. Σχεδιάστε και υλοποιήστε αλγόριθμο πολλαπλής στοίχισης για τις συμβολοσειρές του συνόλου datasetA. Για τη σύγκριση δύο συμβολοσειρών, υιοθετήστε αλγόριθμο καθολικής στοίχισης ο οποίος ορίζει ότι: η οριζόντια και η κάθετη μετάβαση προσθέτουν gap penalty -α, η τοπική ομοιότητα προσθέτει score +1 και η τοπική ανομοιότητα προσθέτει penalty -α/2. Επίσης, δυνατοί πρόγονοι του κόμβου (i,j) είναι οι (i-1,j-1), (i,j-1), (i-1,j). Εκτυπώστε το αποτέλεσμα της πολλαπλής στοίχισης. Αν όλα τα ΑΜ των μελών της ομάδας καταλήγουν σε περιττό ψηφίο τότε α=2. Σε κάθε άλλη περίπτωση α=1.

III. Με βάση το αποτέλεσμα της πολλαπλής στοίχισης κατασκευάστε HMM profile και ρυθμίστε τους σχετικούς πίνακες πιθανοτήτων. Εκπαιδεύστε το με τις συμβολοσειρές του συνόλου datasetB.

IV. Υπολογίστε τα alignment scores και alignment paths για τις ακολουθίες του datasetC. Επίσης υπολογίστε τα alignment scores για 20 τυχαίες ακολουθίες και σχολιάστε τα αποτελέσματα.

Αποδεκτές γλώσσες υλοποίησης είναι οι Python και Matlab. Κάθε ερώτημα πρέπει να συνοδεύεται από τεκμηρίωση στο αρχείο documentation.pdf. Η λογοκλοπή οποιουδήποτε είδους οδηγεί σε μηδενισμό.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΠΙΝΑΚΩΝ

1. Θέμα 1

a. Υλοποίηση

Στο θέμα 1 μας ζητείτε να δημιουργήσουμε 3 dataset με συγκεκριμένο τρόπο.
Μας δώθηκαν 4 συγκεκριμένα patterns του αλφάβητου

- Pattern 1 = *"ATTAGA"*
- Pattern 2 = *"ACGCATTT"*
- Pattern 3 =*"AGGACTCAA"*
- Pattern 4 =*"ATTTCAGT"*

Από τα οποία πρέπει να δημιουργήσουμε 100 συμβολοσειρές , τις οποίες τις δημιουργούμε μεταλλάσοντας και επαυξάνοντας τα παραπάνω patterns με τον εξής τρόπο

- Αρχικά, για όλη την εργασία ως αλφάβητο θεωρούμε το νουκλεοτιδικό αλφάβητο **(A, C, G, T)**
- Κάθε sequence αρχίζει με 1 έως 3 τυχαία σύμβολα του αλφάβητου
- Για κάθε καινούργιο sequence που θα δημιουργηθεί επιλέγουμε τα 4 patterns με τη σειρά από το πρώτο στο τέταρτο, μια φορά το καθένα και αλλάζουμε από κανένα μέχρι δύο σύμβολα σε τυχαίες θέσεις, είτε με ένα τυχαίο επιλεγμένα σύμβολο του αλφάβητου (μπορεί να επιλεχθεί ακόμα και το ίδιο σύμβολο, στην οποία περίπτωση δεν θα παρατηρηθεί αλλαγή), είτε με το κενό, διαγράφοντας το σύμβολο (φυσικά το αν θα προβούμε σε substitution ή deletion για μια δεδομένη θέση αποφασίζεται και αυτό με τυχαίο τρόπο) και τελικά προσθέτουμε το αποτέλεσμα της μετάλλαξης στη μέχρι τώρα συμβολοσειρά, μέχρι να έχουμε προσθέσει το αποτέλεσμα κάποιας τυχαίας μετάλλαξης και των 4 patterns.
- Τέλος κάθε sequence θα τελειώνει με 1 έως 2 τυχαία σύμβολα του αλφάβητου.

Η διαδικασία επαναλαμβάνεται 100 φορές και τα 100 sequences που δημιουργούνται θα εισαχθούν με τυχαίο τρόπο σε 3 datasets

- Dataset A: 10 sequences
- Dataset B: 70 sequences
- Dataset C: 20 sequences

Για τις ανάγκες του θέματος κάνουμε χρήση των συναρτήσεων

- **mutate_pattern**: Εφαρμόζει έως *max_mutations* τυχαίες αλλαγές (substitution ή deletion) στο δοθέν pattern.
  - Arguments: *pattern* (σε μορφή string) και *max_mutations,* δηλαδή ένας ακέραιος με τον αριθμό των μεταλλάξεων (default 2)
  - Output: Το μεταλλαγμένο pattern σε μορφή string
- **create_random_sequence:** δημιουργεί τυχαία ακολουθία από *min_len* μέχρι *max_len*
  - Arguments: 2 ακέραιοι *min_len* και *max_len* που δηλώνουν το ελάχιστο και το μέγιστο επιτρεπτό μήκος της συμβολοσειράς
  - Output: Η τυχαία παραγώμενη συμβολοσειρά σε μορφή string

Ακόμη χρησιμοποιούνται και 2 μέθοδοι **save_fasta** που σώζουν τα παραγώμενο datasets σε αρχεία τύπου fasta. Για το θέμα δημιουργήθηκε το script **thema_1.py**

b. Εκτέλεση

Η εκτέλεση του προγράμματος μας δίνει αυτά τα αποτελέσματα με αυτά τα στατιστικά:

- Dataset A:
  - Αριθμός Ακολουθιών: 10
  - Μέσο μήκος ακολουθίας: 33.0
  - Πρώτη ακολουθία: *"ACGATTAGGACCCTTTACGACTAAAATTTCACTCC" (μήκος = 35)*
  - Τελευταία ακολουθία *"GAATTAGAACGCATTTAGGACTCAAATTTCAGTC" (μήκος = 34)*
- Dataset B:
  - Αριθμός Ακολουθιών: 70
  - Μέσο μήκος ακολουθίας: 32.79
  - Πρώτη ακολουθία: *"TAATTAGAACCATCTAAGACTCAAATTTCGTG" (μήκος = 32)*
  - Τελευταία ακολουθία *"TAAGGTAGAACGCATATAGGTCTCACATTTCATCG" (μήκος = 35)*
- Dataset C:
  - Αριθμός Ακολουθιών: 20
  - Μέσο μήκος ακολουθίας: 32.10
  - Πρώτη ακολουθία: *"CCTATTAGACGCATTTAGGACGCAAATTTCATGC" (μήκος = 34)*
  - Τελευταία ακολουθία *"AAATGAGACGCTTTTAGGACTCAATTTCAGC" (μήκος = 31)*

2. Θέμα 2

a. Σύσταση του αλγόριθμου Πολλαπλής Στοίχισης

Και τα 2 ΑΜ της ομάδας καταλήγουν σε περριτό αριθμό, επομένως για τις ανάγκες του θέματος α =2.
Θα πρέπει να δημιουργηθεί αλγόριθμος πολλαπλής στοίχισης βασισμένος σε έναν αλγόριθμο global alignment (καθολικής στοίχισης) ο οποίος σύμφωνα με την εκφώνηση θα οριστεί ως εξής

- Η οριζόντια και η κάθετη μετάβαση στο πλέγμα δυναμικού προγραμματισμού θα τιμωρείτε με gap penalty -2
- Η τοπική ομοιότητα θα προσθέτει score +1 (αρχικά, θα μιλήσουμε για το lookup table αργότερα)
- Η τοπική ανομοιότητα θα επιβάλει penalty -1

- Δυνατοί πρόγονοι του κόμβου (i,j) θα είναι οι κόμβοι (i-1, j-1), (i, j-1) και (i-1, j) δηλαδή, ο από κάτω του, ο αριστερά του και ο κάτω και αριστέρα διαγώνιος
- Προφανώς ο αλγόριθμος αποτελεί αλγόριθμο μεγιστοποίησης με βάση την ομοιότητα 2 συμβολοσειρών, οπότε επιλέγουμε πάντα τη μετάβαση που μεγιστοποιεί το σκορ
- Η στοίχιση των 2 προφίλ γίνεται μέσο backtracking με τον εξής τρόπο:
    - Διαγραφή για κάθετη κίνηση (προσθήκη "_" στη δεύτερη ακολουθία)
    - Εισαγωγή για οριζόντια κίνηση (προσθήκη "_" στη πρώτη ακολουθία)
    - Όταν εφαρμώσαμε διαγώνια κίνηση τότε μιλάμε για αντιστοίχιση (δεν μας ενδιαφέρει αν τα σύμβολα είναι ίδια)

Το παραπάνω υλοποίει η συνάρτηση *"align_seq"*

Για τη διαδικασία της πολλαπλής στοίχισης θα υιοθετηθεί η διαδικασία hierarchical clustering (βασική ρουτίνα "*hierarchical_msa*") η οποία παραδόθηκε στην 7η διάλεξη του μαθήματος και:

- Για κάθε εποχή (μέχρι να μείνουν με 2 sequences)
    - Δημιουργούμε ένα πίνακα ανομοιοτήτων (συνάρτηση *"compute_distance_matrix"*
    - Επιλέγουμε το το λιγότερο ανόμοιο ζευγάρι
    - Εφαρμόζουμε τον αλγόριθμο καθολικής στοίχισης στο ζευγάρι
    - Φτιάχνουμε την ακολουθία συναίνεσης (όχι στη τελευταία εποχή)
    - Αντικαθιστούμε το ζευγάρι με την ακολουθία συναίνεσης (όχι στη τελευταία εποχή)
- Κάνουμε unfold τα αποτελέσματα

Για τη δημιουργία του πίνακα ανομοιοτήτων θα χρησιμοποιήσουμε τον αλγόριθμο k-mer ο οποίος διδάχθηκε στην 6η διάλεξη του μαθήματος και ορίζεται ως εξής

- Ορίζουμε κάποιο k
- Με τεχνική συρρώμενου παραθύρου βρίσκουμε όλες τις πιθανές k-αδες και μετράμε τη συχνότητα τους, φτιάχνοντας ένα πίνακα συχνοτήτων ο οποίος θα μπορούσε να μοιάζει ως εξής:
  Παράδειγμα: 2 συμβολοσειρές
    - Seq1: "ATGCTAGC"
    - Seq2: "ATGCGC"

  Ο πίνακας συχνοτήτων θα ήταν

	Seq1	Seq2
ATG	1	1
TGC	1	1
GCT	1	0
CTA	1	0
TAG	1	0
AGC	1	0
GCG	0	1
CGC	0	1

Table 1: Παράδειγμα πίνακα συχνοτήτων

Το παραπάνω υλοποίει η συνάρτηση *"compute_kmer_freq"*
Από εκεί και πέρα μπορούμε να διανυσματοποιήσουμε το παραπάνω πίνακα για κάθε sequence και να υπολογίσουμε της απόστασης συνημιτόνου

$$Έστω\ \underline{S_1} = \begin{bmatrix}1\\1\\1\\1\\1\\1\\0\\0\end{bmatrix} και\ \underline{S_2} = \begin{bmatrix}1\\1\\0\\0\\0\\0\\1\\1\end{bmatrix}$$

Ορίζουμε την απόσταση συνημιτόνου (cosine distance, με το αντίστοιχο function στο κώδικα) ως εξής

$$dot(\underline{a},\underline{b}) = \left|\left|\underline{a}\right|\right| \cdot \left|\left|\underline{b}\right|\right| \cdot cos(\theta) <=>$$
$$cos(\theta) = \frac{dot(\underline{a},\underline{b})}{\left|\left|\underline{a}\right|\right| \cdot \left|\left|\underline{b}\right|\right|}$$
$$D(\underline{a},\underline{b}) = cosinde\ distance = 1 - cos(\theta)$$

όπου:
$dot(\underline{a},\underline{b})$ το εσωτερικό γινόμενο των a και b
$||\underline{a}||$το ευκλείδιο μήκος ενός διανύσματος
Τελικά θα καταλήξω με ένα άνω τριγωνικό πίνακα, του οποίου το κελί (i , j) αποτελεί το $D(\underline{S_i},\underline{S_j})$

- Για την ακολουθία συναίνεσης θα χρησιμοποιήσουμε την εξής διαδικασία στις συμβολοσειρές μετά το alignment (το αποτέλεσμα δηλαδή του backtracking στο πλέγμα δυναμικού προγραμματισμού)
    - Αν $S_i[\kappa] = S_j[\kappa]$ τότε $C_{ij}[\kappa] = S_i[\kappa]$
    - Αν $S_i[\kappa] = \emptyset\ και\ S_i[\kappa] \neq \emptyset$ τότε $C_{ij}[\kappa] = S_j[\kappa]$(και αντίστοιχα)
    - Αν $S_i[\kappa] \neq S_j[\kappa]$ τότε εισάγω ένα καινούργιο σύμβολο (πχ X) το οποίο ορίζεται ως: $C_{ij}[\kappa] = X = \begin{cases} S_i[\kappa],, score = 1/2 \\ S_j[\kappa], score = 1/2 \end{cases}$

Τα scores αυτά ανανεώνουν το lookup table,έτσω ώστε το X θα προσθέσει score = 0.5 αν βρεθεί με τα $S_i[\kappa]$ ή$S_j[\kappa]$ στον αλγόριθμο στοίχισης. Προφανώς στο lookup table οι βαθμολογίες αρχικά είναι 1 για όμοια σύμβολα και 0 για ανόμοια σύμβολα, δηλαδή

	A	C	G	T
A	1	0	0	0
C	0	1	0	0

G	0	0	1	0
T	0	0	0	1

Table 2: Αρχικό lookup table

Τα scores για κάθε σύμβολο προφανώς και μπορεί να αλλάξουν, αν ένα σύμβολο χρειάζεται να αντιπροσωπεύει περισσότερα σύμβολα, με τα scores να αλλάζουν με βάση το frequency
Τα παραπάνω υλοποίει η συνάρτηση *"consensus_sequence"*

- Τέλος απαιτείτε να κάνουμε unfold το αποτέλεσμα της πολλαπλής στοίχισης. Ουσιαστικά αυτό είναι το αποτέλεσμα όλων των στοιχισμένων συμβολοσειρών, δηλαδή κάθε στοιχισμένη ακολουθία από την οποία φτιάχνουμε κάποια ακολουθία συναίνεσης, θεωρείτε στοιχισμένη, οπότε αν πχ στη πρώτη εποχή στοιχίσω 2 ακολουθίες *"ACCGTTACCATAC" και ""AGTTTACATC"* και λάβω τη στοίχιση *"ACCGTT_ACCATAC" και "A__GTTTAC_AT_C"* , προφανώς και θα φτιάξουμε την ακολουθία συναίνεσης *"ACCGTTTACCATAC"* και οι 2 στοιχισμένες ακολουθίες που θα αφαιρεθούν από τη διαδικασία θα προστεθούν στο αποτέλεσμα της πολλαπλής στοίχισης και θα θεωρηθούν πλέον πλήρως στοιχισμένες. Όποια από τις αρχικές ακολουθίες βγαίνει από τη διαδικασία θεωρείτε στοιχισμένη. Το unfold είναι στη περίπτωση μας η εκτύπωση της στοίχισης
- **Clustering:** Κατά τη δημιουργία του αλγόριθμου σε python βρήκαμε ένα θεμελιώδες πρόβλημα της παραπάνω στρατηγικής το οποίο οφείλεται στο γεγονός ότι σε κάποια επανάληψη μπορεί να επιλεχτούν προς στοίχιση 2 ακολουθίες συναίνεσης. Αν σε αυτές εισαχθεί κάποιο κενό, τότε προστίθεται πληροφορία σε ακολουθία συναίνεσης που δεν προήλθε από το dataset και αυτό οδηγεί σε ένα αποσυγχρονισμό των στηλών των ακολουθιών που συνέβαλαν στη δημιουργία της αντίστοιχης ακολουθίας συναίνεσης. Τελικά κάποιες ακολουθίες είχαν μικρότερο μήκος από κάποιες άλλες. Αυτό λύνετε με 2 τρόπους: Ο ένας θα ήταν να χρησιμοποιούμε πάντα έστω μια ακολουθία από τις αρχικές, και ο δεύτερος (και αυτός που επιλέχθηκε) θα ήταν να διατηρούμε πληροφορία, ομαδοποιόντας μαζί (clustering) όλες τις ακολουθίες που συνέβαλαν στη δημιουργία μιας ακολουθίας συναίνεσης. Αν σε αυτή εισαχθεί κάποιο κενό το εισάγουμε μέσο μιας διαδικασίας propagation σε όλες τις ακολουθίες του αντίστοιχου cluster

b. Σύγκριση της μεθόδου με τη μέθοδο clustal (bonus)

Στο τέλος του θέματος κρίναμε σκόπιμο να συγκρίνουμε τη τελική μέθοδο μας με κάποια γνωστή μέθοδο Multi Sequence Alignment η οποία χρησιμοποιείτε ευρέως στο πεδίο της Βιοπληροφορικής. Συγκεκριμένα επιλέξαμε τη μέθοδο clustalW της GenomeNet, με την οποία κάναμε align το dataset A. Τα αποτελέσματα τα κατεβάσαμε σε ένα αρχείο *clustalw.aln* και με βάση αυτό συγκρίναμε τα αποτελέσματα της δικιάς μας custom στοίχισης. Για τη διαδικασία του benchmarking κάναμε:

- **Άθροισμα Ζευγαριών (SP) (32.5%)**
  Για κάθε Στοίχισηκακολουθιών k και L στηλών η μετρική ορίζεται ως:

$$SP = \sum_{1 \le i < j \le k} \sum_{c=1}^{L} p(M_{i,c}, M_{j,c})$$

Όπου $p(a,b) = \begin{cases} 1, M_{i,c} = M_{j,c} \text{ και } M_{i,c} \neq gap \\ 0, M_{i,c} \neq M_{j,c} \text{ και } M_{i,c} \neq gap \text{ και } M_{j,c} \neq gap \\ -2, M_{i,c} = gap \text{ ή } M_{j,c} = gap \end{cases}$

Το παραπάνω κανονικοποείτε ως:

$$SP_{max} = \frac{k(k-1)}{2} \cdot L, SP_{norm} = \frac{SP}{SP_{max}}$$

- **Σκορ Στύλης (CS) (16.7%)**
  Ο αριθμός των στηλών στον οποίο όλες οι ακολουθίες έχουν τον ίδιο χαρακτήρα

$$CS = \frac{|c \mid \text{στήλη } c \text{ ίδια}|}{L}$$

  Τα παραπάνω υλοποιούνται για τις 2 αντιστοιχήσεις, ακολουθία προς ακολουθία και στήλη προς στήλη για τις ακολουθίες με ίδιο id
- **Ποσοστό ταυτότητας στήλης στις τελικές ακολουθίες συναίνεσης (86.11%)**
  Με αυτό μετράμε κατά πόσον η τελική ακολουθία συναίνεσης των 2 στοιχήσεων ήταν ίδια

$$PID = \frac{1}{L} \sum_{c=1}^{L} 1(a_c = b_c) \cdot 100\%$$

Όπως παρατηρούμε η μέθοδος μας, αν και βασισμένη στη λογική, παράγει αρκετά ληψά αποτελέσματα σε σχέση με μια industry standard μέθοδο πολλαπλής στοίχισης, αποτέλεσμα που το περιμέναμε. Για το θέμα 3 θα χρησιμοποιήσουμε τα αποτελέσματα της δικής μας μεθόδου στοίχισης.

c. Εκτέλεση

ACGATTAGGACCCTTTACGACTAAAATTTCACTCC
__ACAAGTAAACGATTAGG_CTCAAATTTCAG__C
_CGATTAGAACGATCTAGGACTCAAATTTCAGTAT
ACTATAGCACCCATTTAGGATTCAAATTT_GGTCG
GCCATAGAACGCA_TTAGGACTCAAATTTCAGTGG
AAATTAG__CGC_ATTAGGACTCCAATTTCAGTTC
T_TAATTAGAAGCTTTAGGACTCAAATTT_GAGTG
__AAAAGAAGG_ATTTAGGACTCAAGTTTCAGTAG
__GATTAGAACGCTTTAGGACT_AAATTCCAGTAG
GAATTAGAACGCATTTAGGACTCAAATTTCAGT_C

Table 3: Αποτέλεσμα πολλαπλής στοίχισης για το dataset A

3. Θέμα 3

Στο θέμα 3 μας ζητείται η αρχικοποίηση και η εκπαίδευση ενός Hidden Markov Model (HMM) με τα στοιχισμένα προφίλ του θέματος 2 και να το εκπαιδεύσουμε με τις συμβολοσειρές του dataset B. Η εργασία ζητάει μια δική μας υλοποίηση ενός HMM profile οπότε και αυτό θα πράξουμε.

a. Ισχυρές Καταστάσεις

Πρώτη μας δουλειά είναι, με αυτόματο τρόπο να κατηγοριοποιήσουμε τις στήλες σε δυνατές και αδύναμες. Αυτό μπορεί να γίνει με έτοιμα μέσα πάνω στα νουκλεοτίδια, παρόλα αυτά οι ακολουθίες με τις οποίες ξεκινήσαμε είναι εντελώς τυχαίες. Μια ιδέα θα ήταν να υπολογίσω το ποσοστό ταυτότητας στοίχισης, και οι καταστάσεις των οποίων το σκορ θα ξεπερνάει ένα κατόφλι θα θεωρούνται ισχυρές. Μπορούμε αυθαίρετα να πούμε ότι το 75% των τιμών είναι ισχυρές. Αυτό είναι μια αθώα (naive) ιδέα. Στη βιολοπληροφορική έχουν παρατηρηθεί προσπάθειες εύρεσης τέτοιων καταστάσεων με χρήση της θεωρίας πληροφορίας του Shannon. Η μέθοδος αυτή θα υλοποιούταν με τον εξής τρόπο

- Για κάθε στήλη c:

$$H(c) = -\sum_{i=1}^{5} p_i log_2(p_i)$$

  Όπου p_i είναι η συχνότητα του αντίστοιχου συμβόλου στη στήλη (θεωρούμε αυθαίρετα, A = 1, C = 2, G = 3, T = 4, gap = 5)
- Η μέγιστη εντροπία για αλφάβητο 4 χαρακτήρων θα ήταν $\log_2(5)$, οπότε για να βρουμε τη λεγόμενη περιεκτικότητα σε πληροφορία (IC) θα πρέπει να υπολογίσουμε

$$IC(c) = \log2(5) - |H(c)|$$

  Αν IC(c) > 1 τότε θα λέμε ότι η κατάσταση είναι ισχυρή, αλλίως θα λέμε ότι είναι ανίσχυρη

Βρίσκουμε ότι με αυτά τα κριτήρια 26 από τις 35 στήλες θεωρούνται ισχυρές, ενώ η μέση τιμή της μετρικής IC στο σύνολο των στοιχισμένων ακολουθιών είναι 1.44. Η απόδοση μας καλύπτει οπότε θα θεωρήσουμε το παραπάνω ως τον αλγόριθμο εύρεσης ισχυρών καταστάσεων.

b. Ζητήματα Markovιανής Μοντελοποίησης

Το HMM το οποίο θα αρχικοποιήσουμε λοιπόν θα έχει **26 + 2** (start , end) καταστάσεις αντιστοίχισης (**match states**), οι οποίες πιθανόν να συνοδεύονται από τις αντίστοιχες καταστάσεις διαγραφής (**delete states**) και **35 – 26 = 9** καταστάσεις εισαγωγής (**insert states**) (προφανώς αυτοί οι αριθμοί βρίσκονται αλγοριθμικά για οποιαδήποτε είσοδο και οι καταστάσεις δημιουργούνται με πιο γενικευμένο τρόπο, εξηγούμε παρακάτω). Η βασικοί κανόνες είναι πως:

- Κάθε **match state** εκπέμπει τα σύμβολα της αντίστοιχης ισχυρής στήλης (όχι το gap) με την αντίστοιχη συχνοτική πιθανότητα (στη πράξη

αρχικοποιούμε όλα τα σύμβολα, αυτά τα οποία δεν εμφανίζονται στη στήλη με κάποια οσοδήποτε μικρή πιθανότητα (Dirichlet Prior))

$$\forall M_i (i \in 1, \dots, 28)$$

$$e(M_i, a) = \frac{1}{\sum_{c=1}^{10} 1\{x_{c,i} \neq \_\}} \sum_{s=1}^{10} 1\{x_{s,i} = a\}, a \in \{A, C, G, T\}$$

- Ακόμη μοντελοποιούμε τις ανίσχυρες στήλες με **insert state**, διατηρώντας πάλι τον ανάλογο πίνακα πιθανοτήτων. Αξίζει να σημειωθεί πως μπορούμε να μοντελοποιήσουμε πολλαπλά συνεχόμενα insert state με αυτοαναφορά. Για ανάγκες απλότητας κώδικα αρχικοποιούμε τόσες **insert states** όσες **match states** με οσοδήποτε μικρές πιθανότητες

$$\forall I_j : (j \in (1, \dots, 28)$$

$$e(I_j, \alpha) = \frac{1}{10} \sum_{s=1}^{10} 1\{x_{s,j} = \alpha\}, a \in \{A, C, G, T\}, \alpha\nu\ \eta\ j - \text{οστή στήλη είναι αδύναμη κατάσταση}$$

αλλιώς

$$e(I_j, \alpha) = 10^{-6} \Pi_\alpha$$

Και αντίστοιχα για την πιθανότητα μετάβασης (Dirichlet Prior με βάση τα κενά) περισσότερα για αυτό παρακάτω)

- Από κάθε **match state** μπορούμε να μεταβούμε στην επόμενη **match state** με κάποια πιθανότητα, ίση με τον αριθμό των συμβόλων διάφορων του κενού στην επόμενη ισχυρή κατάσταση. Ακόμη μπορούμε να μεταβούμε στην μεθεπόμενη κατάσταση (πρακτικά σε οποιαδήποτε κατάσταση πέραν της επόμενης) με κάποιο **delete state**, η οποία πιθανότητα μετάβασης ρυθμίζετε με βάση τον αριθμό των κενών στη στήλη που αντιπροσωπεύει το επόμενο **match state**. Επίσης για κάθε **match state** θα μοντελοποιήσουμε και ένα αντίστοιχο **delete state** με οσοδήποτε μικρή πιθανότητα

$$\alpha(M_i \to M_{i+1}) = \frac{\sum_{c=1}^{10} 1\{x_{c,i+1} \neq \_\}}{10} - 10^{-6},$$

$$\alpha(M_i \to D_{i+1}) = \begin{cases} \frac{\sum_{c=1}^{10} 1\{x_{c,i+1} = \_\}}{10}, \alpha\nu\ \frac{\sum_{c=1}^{10} 1\{x_{c,i+1} = \_\}}{10} > 0 \\ 10^{-6}, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases},$$

αν δε υπάρχει **insert state**

$$\alpha(M_i \to I_j) = \frac{\sum_{s=1}^{10} 1\{x_{s,j} = \alpha\}}{10} - 10^{-6},$$

$$\alpha(M_i \to M_{i+1}) = \begin{cases} \frac{\sum_{s=1}^{10} 1\{x_{s,j} = \alpha\}}{10}, \alpha\nu \ \frac{\sum_{s=1}^{10} 1\{x_{s,j} = \alpha\}}{10} > 0 \\ 10^{-6}, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Για τους σκοπούς της μοντελοποίησης θα δημιουργηθούν:

- $\underline{A}$ = πίνακας πιθανοτήτων για τη μετάβαση καταστάσεων (πίνακας 2 διαστάσεων της numpy)
- $\underline{B}$ = πίνακας πιθανοτήτων συμβόλων για όλες τις καταστάσεις (υλοποιήθηκε με εμφολευμένο λεξικό της python, όπου κάθε κατάσταση (εξωτερικό κλειδί) περιέχει ένα εσωτερικό λεξικό στο οποίο υπάρχει η πληροφορία των πιθανοτήτων εκπομπής ανά σύμβολο)
- $\underline{\Theta}$ = ένα λεξικό με τις αρχικές πιθανότητες των συμβόλων (το οποίο σύμφωνα με τη θεωρία δεν αλλάζει κατά τη διάρκεια της εκπαίδευσης) (χρησιμοποιείτε κυρίως για την αρχικοποίηση των μηδενικών συμβόλων στο πίνακα B, δηλαδή αρχικοποιούμε το B με *Dirichlet Prior*)
- Τα indices στους πίνακες πάνε με τη σειρά: Start, M_1, …, M_{26}, D_1, …, D_{26}, I_1, …,I_{26}, End δηλαδή όλα τα match states, μετά όλα τα delete states, μετά όλα τα insert states, ενώ υπάρχουν και οι καταστάσεις αρχής και τέλους.
  Note: Συνήθως στα δίκτυα HMM αρχικοποιούμαι και ένα πίνακα π ο οποίος περιέχει την αρχική κατανομή καταστάσεων της πολλαπλής στοίχησης με βάση την οποία αρχικοποιούμαι το αυτόματο. Αυτό στη περίπτωση μας δεν είναι απαραίτητο καθώς η εν λόγω κατανομή αποτελεί εν γένη χαρακτηρηστικό των δεδομένων εισόδου (από το θέμα 1 κιόλας) και ως εκ τούτου δεν μας ενδιαφέρει η αλλαγή της κατά την εκπαίδευση, άρα ο πίνακας π δεν θα μας απασχολήσει για τη συγκεκριμένη υλοποίηση.

c. Fine Tuning (εκπαίδευση)

Ο αλγόριθμος εκπαίδευσης ενός μοντέλου HMM μπορεί να να περιγραφεί ως:

- **Για κάθε εποχή:**
  - Για κάθε ακολουθία στο training set (στη περίπτωση μας το dataset B) κάνουμε Viterbi alignment μεταξύ της ακολουθίας και του μοντέλου HMM.
  - Αθροίζουμε το λογαριθμικό σκορ για όλες τις ακολουθίες του training batch (εδώ χωρίζουμε τυχαία το dataset B σε 7 δεκάδες)
  - Κρατάμε αντίγραφα των A και B και αρχικοποιούμε μετρητές c_A και c_B με βάση το $\underline{\Theta}$
  - Για κάθε ακολουθία:
    - Καλούμε Viterbi και κρατάμε και το μονοπάτι καταστάσεων
    - Αυξάνουμε το μετρητή μετάβασης c_A για κάθε νέα μετάβαση $prev \to st$
    - Αυξάνουμε το μετρητή εκπομπής c_B αν η κατάσταση εκπέμπει σύμβολο (δηλαδή είναι **match state** ή **insert state**)
  - Εφαρμόζουμε μια απλή πράξη κανονικοποίησης και θέτουμε

$$A'[i,j] = \frac{c_A[i,j]}{\sum_j c_A[i,j]}, i,j\ \delta\varepsilon\Omega\kappa\tau\varepsilon\varsigma\ \kappa\alpha\tau\alpha\sigma\tau\acute{\alpha}\sigma\varepsilon\omega\nu$$
$$B'[st,x] = \frac{c_B[st,x]}{\sum_x c_B[st,x]},$$
$$st\ \eta\ \varepsilon\kappa\Phi\sigma\tau\sigma\tau\varepsilon\ \kappa\alpha\tau\acute{\alpha}\sigma\tau\alpha\sigma\eta\ \kappa\alpha\iota\ x\ \tau\alpha\ \sigma\acute{\upsilon}\mu\beta\sigma\lambda\alpha\ \tau\sigma\upsilon\ \alpha\lambda\phi\acute{\alpha}\beta\eta\tau\sigma\upsilon$$

  - Υπολογίζω εκ νέου το λογαριθμικό σκορ με τα A’ και B΄ και αν παρατηρώ βελτίωση, τότε θα κρατήσω τα A’ και B’ αλλιώς θα τα απορρίψω
- **Viterbi Alignment:**
  - **Αρχικοποίηση:** $\forall j \in Q$ (καταστάσεις του μοντέλου Markov)
    $$V[1.j] = \pi_j \cdot e_j(x_1)$$
    π_j = η αρχική πιθανότητα της κατάστασης j
    $e_j(x_1)$ = πιθανότητα εκπομπής x_1
  - **Αναδρομή:** Για t από 2 έως *len(input sequence)*
    - $\forall j \in Q$
      $$V[t,j] = max_{i \in Q}[V[t-1,i] \cdot \alpha_{i->j}] \cdot e_j(x_t)$$
    - Κρατάμε το δείκτη $B[t,j]$που μεγιστοποίει τη παραπάνω σχέση
  - **Backtracking:** Για t από *len(input sequence)* έως 2
    - $s_{t-1} = B[t, s_t]$
    - Το σύνολο $(s_1, s_2, \dots, s_T)$ είναι το σύνολο καταστάσεων

- **Scoring:**
  - Για κάθε ακολουθία που στοιχίζω με το μοντέλο (το μοντέλο με τα A’ και B’) υπολογίζω τη μέγιστη από κοινού πιθανότητα παρατηρήσεων-καταστάσεων, δηλαδή πόσο πιθανή είναι η βέλτιστη διαδρομή
  - Αν το άθροισμα των σκορ της εποχής είναι καλύτερο από το προηγούμενο τότε και μόνο τότε θα ενεργοποιηθεί η διαδικασία fine tuning για όλες τις ακολουθίες της εποχής

d. Υλοποίηση

Για την υλοποίηση έχουμε πάει σε αντικειμενοστρεφή λογική με τη κλάση HMMProfile, τη συνάρτηση αρχικοποίησης, τη συνάρτηση train και τον αλγόριθμο Viterbi. Επίσης παρέχουμε μια συνάρτηση print για τις καταστάσεις του αυτομάτου. Το προφίλ αρχικοποιείτε με τα αποτελέσματα του θέματος 2 ενώ γίνεται training με το dataset B χωρισμένο σε δεκάδες (τυχαία, χωρίς overlap) για να δείξουμε τα πολλαπλά epochs που θα μπορούσε να κάνει ο αλγόριθμος. Το αντικείμενο εν συνεχεία σώζεται σε ένα αρχείο μέσο serialization για χρήση στο θέμα 4 μέσο της βιβλιοθήκης pickle που είναι built-in σε κάθε σχεδόν python distribution. Η αρχικοποίηση γίνεται με το constructor ο οποίος δέχεται μια λίστα αλφαριθμητικών (αποτέλεσμα του Multi Sequence Alignment), ενώ οι πίνακες A και

Β είναι διαθέσιμοι με τα ονόματα Α και Β ως ιδιότητες του αντικειμένου. Στο αρχείο του θέματος έχουν δημιουργηθεί και 2 βοηθητικές συναρτήσεις για την εκτύπωση των πινάκων Α και Β. Η συνάρτηση train μπορεί να εκπαιδεύσει το μοντέλο σύμφωνα με τα παραπάνω ενώ η συνάρτηση Viterbi μπορεί να στοιχήσει το μοντέλο με κάποια ακολουθία

4. Θέμα 4

Το θέμα αυτό μας ζητάει να υπολογίσουμε τα alignment scores και alignment paths για τις ακολουθίες του dataset C καθώς και για 20 τυχαίες ακολουθίες. Τα καλά νέα εδώ είναι πως μπορούμε να επαναχρησιμοποιήσουμε το κώδικα του θέματος 3 και το αρχείο στο οποίο έχουμε αποθηκεύση το trained HMM. Απλά προσθέσαμε μια συνάρτηση που επιστρέφει το alignment από το best path του αλγόριθμου Viterbi

a. Alignment με το Dataset C

Original Sequences	**Aligned Sequences**	**Alignment Scores**
CCTATTAGACGCATTTAGGACGCAAATTTCATGC	*CCTAT___TAGACGCATTTAG_GACGCA_AATT*	*-38.155*
TTGATAGAACGATTTAGATCAATGTCAGTA	*TTGA___TAGAACGATTT_AGATCA_ATG*	*-69.714*
ACATTAGAACGCATTTGGACTCAAATTTCAGTA	*ACATTAG__AACGCATTTGGA_CTCAAA_TTT*	*-33.097*
CCATTAGAACGCATTGACTCAAATTTGAGCGT	*CCATTAG___AACGCATTG_ACTCAA_ATTT*	*-37.703*
GAGATTAGAACTCATTTAGGGCTCAAATTTAGTCA	*GAGATTAGAAC_TCATTTAGGG_CTCAAA_TTTA*	*- 57.447*
CAGATAGAACGCATTTAGGATGCAAATTTCAGCTG	*CAGATAG__AACGCATTTAGG_ATGCAAA_TTTC*	*-38.991*
TTATTAAACGTATTTAGGACTAAATTCAGTT	*TTAT__TAAACGTATTTAG_GACTAA_ATT*	*-34.234*

TGATTAGAACACATTTAGGCCTCAATTTCAGTA	*TGAT__TAGAACACATTTAGG_CCTCAA_TTT*	*-32.376*
TAATTAGGATGCATTAAGGACTCAAATTTCAGTT	*TAAT___TAGGATGCATTAAG_GACTCAA_ATT*	*-51.733*
CACATTAAAAGCATTAGGACCCAAATTAGTTG	*CACATTAA___AAGCATTA_GGACCC_AAAT*	*-38.076*
TAGTTAGACGCATTTGGATCAAATTTCATA	*TAGTTAG___ACGCATTT_GGATCA_AAT*	*-50.867*
GATTAGAACGATTTGGACTCAAAATCAGTC	*GATTAG___AACGATTTG_GACTCA_AAA*	*-34.023*
TCTATTGAACGCATTTAGACTCAAATTTCAGTTT	*TCTAT__TGAACGCATTTAGA_CTCAAA_TTTC*	*-50.850*
TCAATTAAACGTAGTTAGGACTCAAAGTTCATT	*TCAATTAAACG__TAGTTAGG_ACTCAA_AGT*	*-72.765*
TATTATGCCCATTTAGGACTCAAATTCCGTG	*TATT__ATGCCCATTTAGG_ACTCAA_ATT*	*-32.065*
AATGAGAACGCATTTAGATCAACTTTCAGTT	*AATG___AGAACGCATTTA_GATCAA_CTT*	*-36.372*
CAGATTGAACGCATTTAGGACTCAAATTTCAGAT	*CAGAT__TGAACGCATTTAGGA_CTCAAA_TT_*	*-37.296*
TCATAGAACGCATTTAGGACTCATTTCAGTT	*TCAT___AGAACGCATTTA_GGACTC_ATT*	*-51.423*
CGATGAACGCATTAGACTCAAATTCGTC	*CGAT___GAACGCATT_AGACTC_AAA*	*-35.954*
AAATGAGACGCTTTTAGGACTCAATTTCAGC	*AAAT___GAGACGCTTTTA_GGACTC_AA_*	*-35.261*

Table 4: Αποτελέσματα στοίχισης του HMM με το dataset C

b. Alignment με 20 τυχαία string (A, C, G, T)

Original Sequences	Aligned Sequences	Alignment Scores
AGAAAGGGAAAGGGGCCCTTACCATACCCCACATACGTCA	*AGAAAG___GGAAAGGG_GCCCTTAC_CATACCCCACA_*	*-86.459*
AACAAACGCCACTGGAGACTGGGTTAACCAT	*AACAAA___CGCCACTGG_AGACTGG_GT_*	*-45.463*
ACTGAAGGGCGCTACTTAAGGCGCCGTGATTTCCA	*ACTGAAGGGCGC_TACTTAAGGC_GCCGTG_AT_*	*-48.261*
TTCGATTCGGATGTGACATTTCATTACATT	*TTCG___ATTCGGATGTG_ACATTT_CAT*	*-52.938*
GTAGGTTTGGCGCTCAAAGGAGAAACGCCGGT	*GTAG___GTTTG_GC_GCTCAAAGGA_GAA_*	*-79.295*
CTTAACCCGGTACCTAACCCATCTGATTTTTACACACTC	*CTTAACCCG__GTACCTA_AC_CCATCTG_ATTTTTAC*	*-52.963*
CGGTGCAAAGACACTGAGTAATCTGGAAGGCCGCCAG	*CGGTG__CAAAGACACTG_AG_TAATCTGGAA_GG_*	*-56.701*
CTACTTAACTTTCATGGTGATCGTAAAGCGGAGCCTT	*CTACTTAACTTTCA__TGGTG_AT_CGTAAAG_CGG*	*-85.754*
GTACATACGGATGAACGCACTTACGGCGTTG	*GTAC___ATACGGAT_GA_ACGCAC_TTAC*	*-75.094*
TTTTAGCTTTTCTATTATCCTAAACTTCGCTGT	*TTTT___AGCTT_TT_CTATTATCCTAA_AC_*	*-75.255*
TCGTAGCTCTATAACGTGTGACTTGGATGGC	*TCGTAG___CTCTA_TA_ACGTGTG_ACTT*	*-87.096*
GAGCGAGCTCATGCCGGTTGGCGGATACTAATGT	*GAGCGAGCTCA__TGCCG_GT_TGGCGG_ATAC*	*-68.888*
CGAGCAGATTACATGAATCTGTGTTGGGTGTGCCAGT	*CGAGCAGAT___TACAT_GA_ATCTGTGTTGG_GT_*	*-65.621*

CACGACTAACAACCCATCTTATGCTCCTTGGTTCCCCG	*CACGACTAA__CAACCCATCTTATG_CTCCTTG_GT_*	*-52.861*
GTTTGTGGACTACCCCTCTCTCGATAATAAATGACA	*GTTTGTGGAC_TACCCCTCTC_TC_GATAAT_AA_*	*-88.637*
CCCGTGTACGTGTCTAGTAGCGTTGAGGAGACC	*CCCGTG___TACGT_GT_CTAGTA_GCGTTG_*	*-95.386*
TGTGCGTCTGCCGCTTATACGCATAATCTGCATAGC	*TGTG___CGTCTGCCGCTT_AT_ACGCAT_AATC_*	*-79.429*
GGCTAAGCGCGCGCGCCAAAGTAACGTGCAAAA	*GGCTAAG___CGCGC_GC_GCCAAAGTA_AC_*	*-75.415*
GAAACTGTAACCCTCTAACGGATTAAGCGG	*GAAA___CTGTA_AC_CCTCTAA_CGGAT*	*-76.511*
CTCGCGATATGCAAGACGCACCTAACTGTAAAACAATG	*CTCGCGAT___ATGCAAGAC_GCACCT_AACTGTAAA*	*-64.477*

Table 5: Αποτελέσματα στοίχισης του HMM με 20 τυχαίες Συμβολοσειρές

c. Σχόλιο

Παρατηρούμε κατά μέσο όρο χειρότερα alignment scores στις ακολουθίες του dataset C είναι κατά μέσο όρο αλλά και κατά min/max value καλύτερα από τα scores των τελείως τυχαίων συμβολοσειρών. Αυτό είναι αναμενόμενο καθώς το Hidden Markov Model αρχικοποιήθηκε και εκπαιδεύτηκε με τις ακολουθίες των dataset A και B. Οι ακολουθίες όμως και των τριών dataset έχουν προέλθει βασικά από τα ίδια patterns μέσο του θέματος 1. Αν και έχουμε εφαρμόσει αρκετή τυχαιότητα και μετάλλαξη, πράγμα το οποίο φαίνεται καθώς ούτε τα σκορ του alignment με το dataset C είναι ιδιαίτερα καλά, έχει διατηρηθεί αρκετή πληροφορία σε σχέση με τα τελείως τυχαία παραδείγματα, τα οποία έχουν κατά κανόνα χειρότερα σκορ. Δηλαδή ο τρόπος δημιουργίας του dataset C το επιτρέπει να έχουν χαριτολογώντας μια μακρινή “συγγένεια” με τις ακολουθίες που αρχικοποίησαν και εκπαίδευσαν το HMM, ενώ οι τυχαίες ακολουθίες δεν έχουν αυτή τη “συγγένεια” άρα και τα alignment scores τους με το HMM είναι χειρότερα. Προφανώς παίζει ρόλο ότι δημιουργούμε τυχαίες συμβολοσειρές μήκους περίπου ίδιου (30, 40) με τα dataset. Αν θα δημιουργούσαμε συμβολοσειρές με αρκετά αποκλύνων μήκος τότε τα scores θα ήταν αρκετά χειρότερα

5. Σχόλια

Για κάθε θέμα έχει δημιουργηθεί και ένα ξεχωριστό script

- **thema_1.py**

- **thema_2.py**
- **thema_3.py**
- **thema_4.py**

Η κλάση που υλοποίει το Hidden Markov Model για τα θέματα 3 και 4 βρίσεκται σε ξεχωριστό αρχείο **hmm.py**
Όλα αυτά τα αρχεία θα βρίσκονται στο **source2025.rar**
Στο αρχείο **auxiliary2025.rar** έχουμε τοποθετήσει

- Τα αρχεία fasta των 3 dataset που δημιοργήθηκαν στο θέμα 1
- Το αρχείο **clustalw.aln** με το alignment των συμβολοσειρών του dataset A μέσο του εργαλείου clustal (θέμα 2)
- Το αρχείο **custom_msa.fasta** με τα αποτελέσματα της δικής μας ιεραρχικής Στοίχισης του dataset A
- Τα αποτελέσματα της στοίχισης του μοντέλου με το dataset C (**alignment.txt**) και με τις τυχαίες συμβολοσειρές (**random_alignment.txt**), από το θέμα 4
- Το αρχείο **trained_hmm_profile.pkl** που αποτελεί το serialized Hidden Markov Model από το θέμα 3

Γενικά ο πηγαίος κώδικας μπορεί να τρέξει σε οποιδήποτε python environment με:

- **numpy**
- **biopython**

Τα θέματα περιμένουν την ύπαρξη των παραγώμενων αρχείων από τα προηγούμενα τους, στο ίδιο directory με αυτά, οπότε η διάτρεξη καλό είναι να γίνει σειριακά. Εναλλακτικά μπορούν να τοποθετηθούν τα δικά μας είδη δημιουργημένα αρχεία από το αρχείο auxiliary
Τα αποτελέσματα μας μπορείτε να τα αναπαράγεται θέτοντας random.seed(42) στα scripts του θέματος 1, του θέματος 4 και τη κλάση του HMM, παρόλα αυτά ο κώδικας μας είναι σε θέση να τρέξει για οποιαδήποτε είσοδο (εκτός προφανώς από το bonus κομμάτι του θέματος 2 όσον αφορά το clustal, αυτό, δεδομένου ότι το alignment γίνεται εξωτερικά έχει μόνο νόημα για το συγκεκριμένο παράδειγμα, με random παράδειγμα τα αποτελέσματα της σύγκρισης του alignment με το custom alignment είναι τελείως άκυρα. Σε περίπτωση που δεν βρεθεί το αρχείο **clustalw.aln** τότε η διαδικασία σύγκρισης απλά δεν θα γίνει)

6. Βιβλιογραφία

- Multiple Sequence Alignment - CLUSTALW
- Bio.Align package — Biopython 1.85 documentation
- Bio.motifs package — Biopython 1.85 documentation
- Microsoft PowerPoint - 05_msa_clustalw.pptx
- Microsoft PowerPoint - week4-monday.ppt
- BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations - PMC
- EMBOSS: needle manual
- 2308.12103

- H2r: Identification of evolutionary important residues by means of an entropy based analysis of multiple sequence alignments | BMC Bioinformatics | Full Text
- Incorporating background frequency improves entropy-based residue conservation measures - PMC
- machine learning - Baum Welch training of HMM - Cross Validated
- Viterbi algorithm - Wikipedia