ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΚΡΟΙΤΟΡ ΚΑΤΑΡΤΖΙΟΥ ΙΩΑΝ Π21077

ΑΛΕΞΙΟΣ ΒΑΣΙΛΕΙΟΥ Π21009

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ ΜΑΘΗΜΑΤΟΣ ΒΙΟΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΕΙΡΑΙΑΣ

Ιούλιος 2024

ΠΡΟΛΟΓΟΣ

Η παρούσα εργασία αναπτύχθηκε ως μέρος του μαθήματος Βιοπληροφορικής, με κύριο στόχο τη δημιουργία προγραμμάτων για εξοικείωση με αλγόριθμους που χρησιμοποιούνται στον τομέα αυτό.

ΕΚΦΩΝΗΣΗ

Θέμα

I. Δίνονται τα παρακάτω patterns από το αλφάβητο A, C, G, T: pattern1=AATTGA, pattern2=CGCTTAT, pattern3=GGACTCAT και pattern4=TTATTCGTA. 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 50 συμβολοσειρές με τον αλγόριθμο σύνθεσης. Διαλέξτε με τυχαίο τρόπο 15 συμβολοσειρές και τοποθετήστε τις σε ένα σύνολο datasetA και τις υπόλοιπες σε ένα σύνολο datasetB.

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=1. Σε κάθε άλλη περίπτωση α=2.

III. Με βάση το αποτέλεσμα της πολλαπλής στοίχισης κατασκευάστε HMM profile και ρυθμίστε τους σχετικούς πίνακες πιθανοτήτων. Υπολογίστε τα alignment scores και alignment paths για τις ακολουθίες του datasetB.

Αποδεκτές γλώσσες υλοποίησης είναι οι Python και Matlab. Κάθε ερώτημα πρέπει να συνοδεύεται από τεκμηρίωση της λύσης.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 ΣΤΟΧΟΙ ΕΡΓΑΣΙΑΣ

Βασικός στόχος της εργασίας είναι η υλοποίηση αλγοριθμικών προγραμμάτων για την εξοικείωση με διαδικασίες στον τομέα της βιοπληροφορικής.

2. ΠΕΡΙΓΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

2.1 ΠΑΡΟΥΣΙΑΣΗ ΑΡΧΙΚΗΣ ΣΚΕΨΗΣ

2.1.1 ΕΡΩΤΗΜΑ i

Το πρώτο ερώτημα του προβλήματος αφορούσε την παραγωγή συνθετικών δεδομένων, συγκεκριμένα DNA αλληλουχιών. Η διαδικασία περιλαμβάνει 4 patterns από το αλφάβητο *A, C, G, T* τα οποία είναι τα *pattern1=AATTGA*, *pattern2=CGCTTAT*, *pattern3=GGACTCAT* και *pattern4=TTATTCGTA*. Ακολουθώντας την μέθοδο που αναφέρεται την εκφώνηση, παράγονται τα δύο σύνολα δεδομένων *datasetA* και *datasetB*, τα οποία αποθηκεύονται σε δύο διαφορετικά αρχεία.

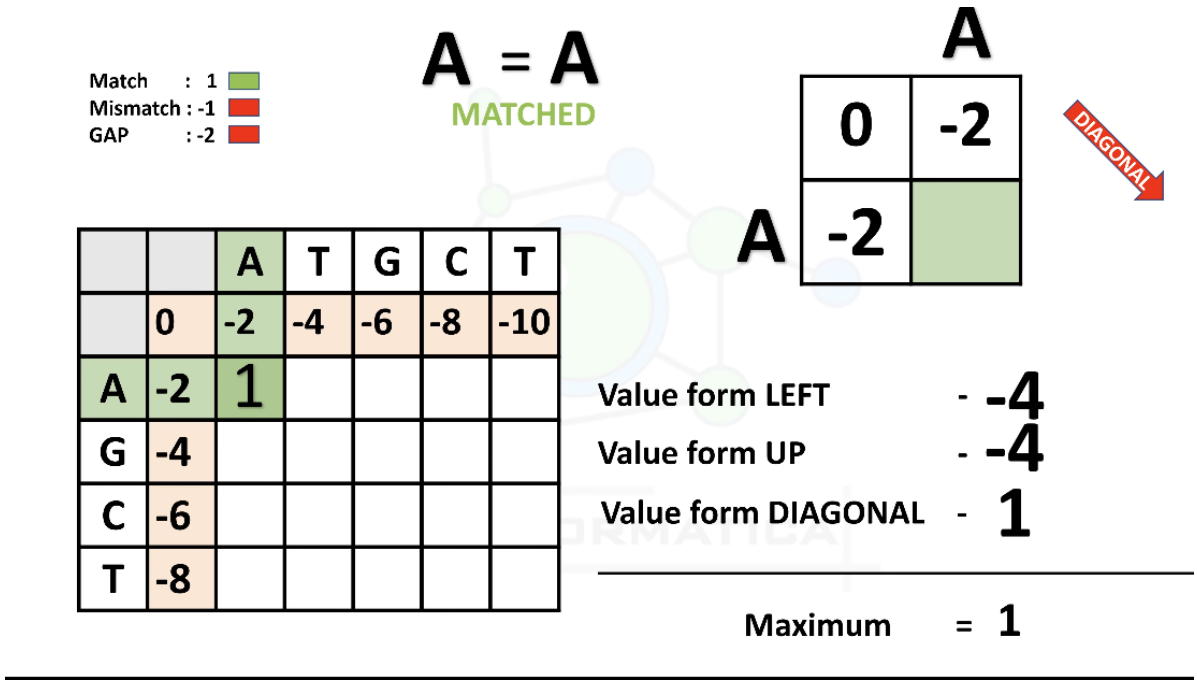
2.1.2 ΕΡΩΤΗΜΑ ii

Το δεύτερο ερώτημα του προβλήματος αφορούσε την υλοποίηση του MSA (Multiple Sequence Alignment/Πολλαπλή Στοίχιση Ακολουθιών) αλγόριθμου, για τα δεδομένα του *datasetA,* χρησιμοποιώντας αλγόριθμο καθολικής στοίχισης (global alignment). Συγκεριμένα αναφερόταν 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*. Έτσι, υιοθετήθηκε η μέθοδος του Needleman-Wunsch η οποία χρησιμοποιεί δυναμικό προγραμματισμό διασπώντας το πρόβλημα σε υπό-προβλήματα και λύνοντάς τα επιμέρους θα βρεθεί η βέλτιστη λύση. Έτσι, με τη βοήθεια των συγγραμμάτων του μαθήματος και πηγών που αναφέρονται στις *Βιβλιογραφικές Αναφορές,* υλοποιήθηκε μία μέθοδος που αξιοποιεί την μήτρα βαθμολόγησης για την εύρεση της λύσης. Δηλαδή, αρχικά αρχικοποιείται με τιμές του gap penalty στην πρώτη στήλη και πρώτη γραμμή. Στην συνέχεια, ξεκινώντας από πάνω δεξιά, για να γεμίσει κάθε κελί της μήτρας (να λάβει μια βαθμολογία δηλαδή), χρησιμοποιείται η μέγιστη τιμή εκ των τριών περίγυρων κελιών (πάνω, αριστερά και διαγώνια). Δηλαδή για κάθε κελί του πίνακα υπάρχουν τρεις περιπτώσεις βαθμολόγησης:

1. Τοπική ομοιότητα/ανομοιότητα: Διαγώνια μετακίνηση από το *(i-1,j-1)* στο *(i,j)*. Δηλαδή αν οι χαρακτήρες είναι ίδιοι και στις δύο αλληλουχίες, στο δεδομένο σημείο *(i,j)*.
2. Κενό στην αλληλουχία 1: Κάθετη μετακίνηση από το *(i-1,j) στο (i,j)*
3. Κενό στην αλληλουχία 2: Οριζόντια μετακίνηση από το *(i,j-1) στο (i,j)*

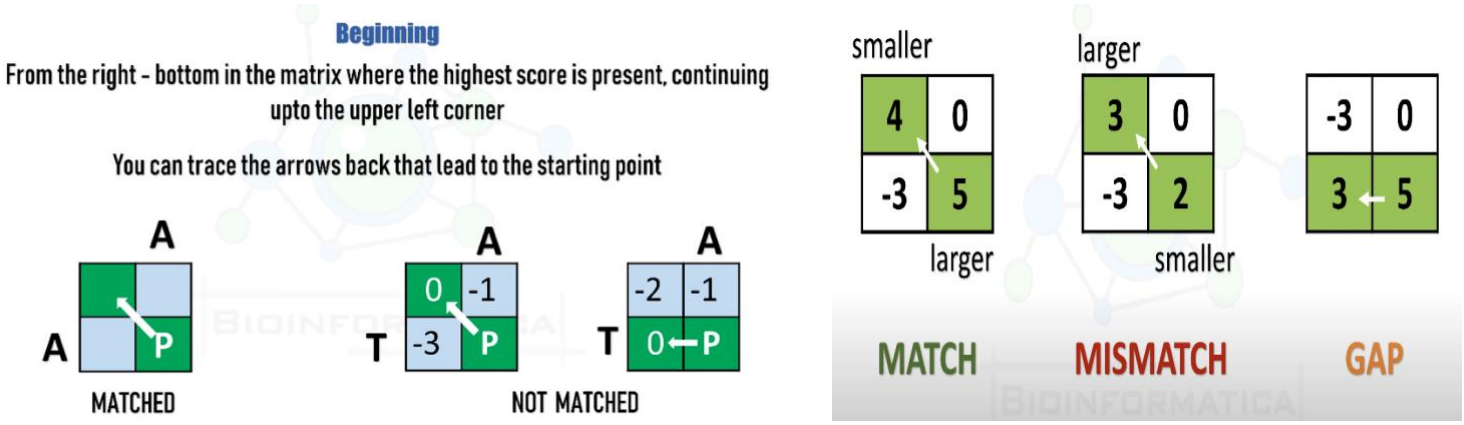
Κάθε κελί, λοιπόν, του πίνακα αυτού αντιπροσωπεύει την βαθμολογία της έως τότε καλύτερης στοίχισης μεταξύ των πρώτων *i* χαρακτήρων της μιας συμβολοσειράς, και των πρώτων *j* χαρακτήρων της άλλης συμβολοσειράς.

Παρακάτω φαίνεται ένα στιγμιότυπο που οπτικοποιεί την διαδικασία:



Εικόνα 1. Υπολογισμός των scores

Ύστερα, ξεκινώντας από κάτω αριστερά, διασχίζεται η μήτρα βαθμολόγησης που δημιουργήθηκε ώστε να ληφθεί η βέλτιστη στοίχιση ακολουθιών. Επομένως αποτελεί έναν τρόπο ανακατασκευής της στοιχισμένης ακολουθίας. Οι λεπτομέρειες λειτουργίας αναφέρονται παρακάτω, στην αντίστοιχη επεξήγηση κώδικα. Επιπροσθέτως η παρακάτω εικόνα οπτικοποιεί την διαδικασία.



Εικόνα 2. Ανακατασκευή της καλύτερης στοίχισης των δύο αλληλουχιών

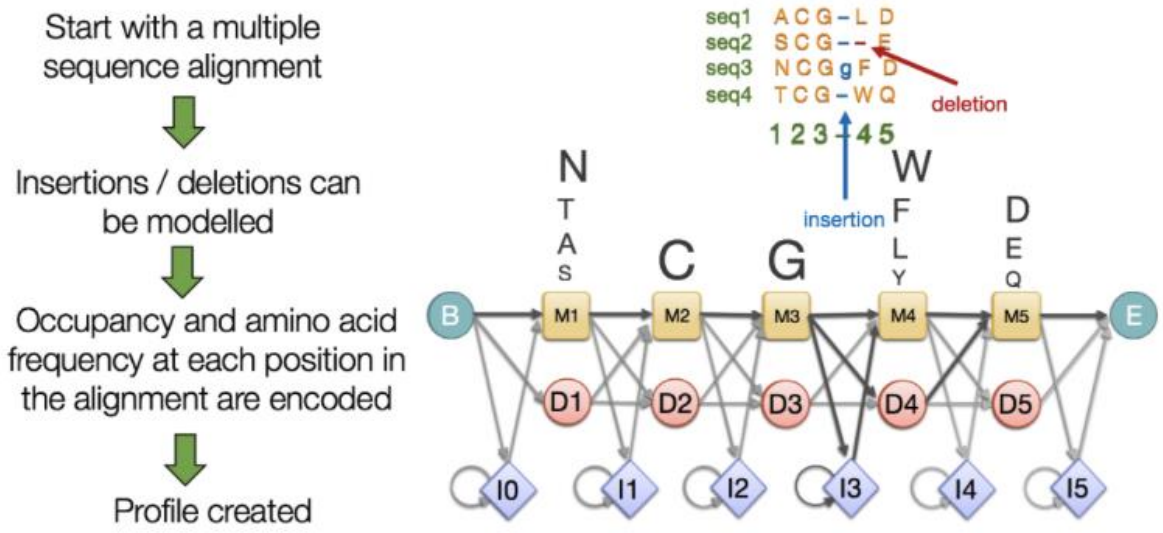
Τέλος, ο αλγόριθμος της πολλαπλής στοίχισης, παίρνει με τη σειρά της δεδομένες αλληλουχίες και της στοιχίζει κατάλληλα χρησιμοποιώντας τις προαναφερόμενες συναρτήσεις. Αρχικά, παίρνει τις πρώτες δύο και της στοιχίζει. Μετά, στην επόμενη επανάληψη θα πάρει την επόμενη προς στοίχιση αλληλουχία και την τελευταία από τις ήδη

στοιχισμένες και θα την στοιχίσει με βάση την ακολουθία αυτή. Έτσι, στο τέλος, προκύπτει το σύνολο με τις στοιχισμένες ακολουθίες, το οποίο αποθηκεύεται σε ένα αρχείο. Επιπλέον, ορίζεται η παράμετρος alpha ίση με 1 εφόσον οι ΑΜ όλων των μελών της ομάδας καταλήγουν σε περιττό αριθμό.

2.1.2 ΕΡΩΤΗΜΑ iii

Το τρίτο ερώτημα του προβλήματος αφορούσε την κατασκευή HMM profile και ρύθμιση των σχετικών πινάκων πιθανοτήτων των στοιχισμένων αλληλουχιών του *datasetA*, καθώς και υπολογισμό των alignment scores και alignment path των αλληλουχιών του *datasetB.* Μετά από εκτενή ανάγνωση, βάση του συγγράμματος και διαδικτυακών πηγών (βλέπε βιβλιογραφικές αναφορές), συμπεράθηκε ότι το HMM profile αποτελείται από 3 είδη καταστάσεων, δηλαδή

1. Καταστάσεις ταιριάσματος *(M),* όπου αντιπροσωπεύουν τις θέσεις στην ακολουθία όπου υπάρχουν χαρακτήρες που έχουν συντηρηθεί (δεν έχουν αλλάξει/διαγραφεί). Στις καταστάσεις αυτές η πιθανοτική κατανομή είναι η συχνότητα των νουκλεοτιδίων σε εκείνο το σημείο.
2. Καταστάσεις προσθήκης *(I),* χρησιμοποιούνται για να μοντελοποιήσουν εξαιρετικά μεταβαλλόμενες περιοχές στη στοιχισμένη αλληλουχία.
3. Καταστάσεις αφαίρεσης *(D),* ή αλλιώς 'σιωπηρές' καταστάσεις, αφού δεν αντιστοιχούν σε κανένα 'απομεινάρι', και υπάρχουν απλώς για να είναι δυνατή η μετάβαση σε μία ή περισσότερες στήλες της στοίχισης.



Εικόνα 3. Μοντελοποίηση HMM profile για πολλαπλή στοίχιση ακολουθίας

Επιπλέον, οι καταστάσεις ταιριάσματος και προσθήκης εκπέμπουν σύμβολα, ενώ οι καταστάσεις αφαίρεσης όχι. Αρχικά, λοιπόν, θα πρέπει να υπολογιστούν οι πιθανότητες εκπομπής (δηλαδή η πιθανότητα να παρατηρηθεί ο κάθε χαρακτήρας δεδομένης μίας κατάστασης) και μεταβολής κατάστασης ώστε να χρησιμοποιηθούν στον αλγόριθμο *Viterbi,* αλγόριθμος ο οποίος κάνει χρήση του δυναμικού προγραμματισμού για την εύρεση της πιο

πιθανής αλληλουχίας των 'κρυμμένων' καταστάσεων σε ένα κρυφό μοντέλο Markov (HMM), δεδομένων των προαναφερόμενων πιθανοτήτων. Μέσω αυτού, θα βρεθεί το πιο πιθανό alignment path μιας ακολουθίας και τα alignment scores. Ο αλγόριθμος λειτουργεί μέσω της επαναληπτικής διαδικασίας υπολογισμού τα μεγαλύτερα probability path σε κάθε κατάσταση σε κάθε βήμα, αποθηκεύοντας της πιθανότητες και κάνοντας backtracking για να προσδιορίσει την πιο πιθανή ακολουθία των κρυφών καταστάσεων. Οι λεπτομέρειες αναφέρονται παρακάτω στην επεξήγηση του κώδικα.

2.2 ΑΝΑΛΥΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για την ανάπτυξη των προγραμμάτων, χρησιμοποιήθηκε η Pyhton3. Αρχικά θα δοθούν κάποιες βασικές πληροφορίες και στη συνέχεια θα αναλυθούν τα κυρίως προγράμματα.

2.2.1 ΓΕΝΙΚΗ ΠΕΡΙΓΡΑΦΗ

ΟΙ βασικές βιβλιοθήκες που χρησιμοποιήθηκαν είναι η *random* για παραγωγή ψευδοτυχαίων τιμών και η *numpy* για διευκόλυνση των αριθμητικών πράξεων.

2.3.1 ΑΝΑΛΥΣΗ ΒΑΣΙΚΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

2.3.1.1 i.py

Το αρχείο αυτό περιέχει τον κώδικα για την υλοποίηση των ζητούμενων του 1ου ερωτήματος. Για την εισαγωγή της τυχαιότητας χρησιμοποιούνται συναρτήσεις της *random.*

def string_composition(strings: list):
Η συνάρτηση αυτή πραγματοποιεί την σύνθεση μιας αλληλουχίας, που παράγεται σύμφωνα με το αλφάβητο A, C, G, T. Αρχίζοντας, επιλέγονται από το αλφάβητο ένα έως τρία σύμβολα ώστε να τοποθετηθούν στην αρχή της συμβολοσειράς. Στην συνέχεια, για κάθε ένα από τα patterns:

1. Εξάγεται ο αριθμός των αλλαγών που θα πραγματοποιηθούν (από καμία έως δύο)
2. Επαναλαμβάνεται για τον αριθμό που εξάχθηκε φορές
    a. Λαμβάνεται τυχαία ο δείκτης του συμβόλου που θα αντικατασταθεί/διαγραφεί και ισοπίθανα μέσω της *randint(0,1),* το σύμβολο που βρίσκεται στην θέση που επιδεικνύει ο δείκτης, είτε θα αντικατσταθεί με ένα άλλο σύμβολο του ίδιου pattern (όχι όμως το ίδιο που επιλέχθηκε), είτε θα διαγραφεί, αντικαθιστώντας το ουσιαστικά με την κενή συμβολοσειρά. Σημειώνεται ότι στην περίπτωση που οι αντικαταστάσεις είναι δύο, τότε τη δεύτερη φορά ο δείκτης που θα αντικατασταθεί θα είναι διαφορετικός από αυτόν που αντικαταστάθηκε προηγουμένως
3. Ύστερα προστίθεται στο τέλος της συμβολοσειράς αυτής ένα ή δύο σύμβολα από το αλφάβητο (μπορεί το ίδιο σύμβολο να προστεθεί δύο φορές)

def generate_datasets():

Η συνάρτηση αυτή επαναλαμβάνει την διαδικασία της συνάρτησης *string_composition()* 50 φορές, καθώς επίσης διασπά τα δεδομένα σε δύο σύνολα, στο *datasetA* που περιλαμβάνει 15 αλληλουχίες και στο *datasetB* που περιλαμβάνει 35 αλληλουχίες. Τέλος τα σύνολα αυτά αποθηκεύονται σε δύο *.txt* αρχεία, τα *datasetA* και *datasetB.*

2.3.1.2 ii.py

Το αρχείο αυτό περιέχει τον κώδικα για την υλοποίηση του 2ου ερωτήματος. Ο πίνακας *aligned_sequences* που προκύπτει και περιέχει τις στοιχισμένες ακολουθίες, αποθηκεύεται σε ένα αρχείο *aligned_sequences.txt.*

def initialize_matrix(n: int, m: int, gap_penalty: int):

Η συνάρτηση αυτή πραγματοποιεί την αρχικοποίηση της μήτρας βαθμολόγησης, με 0 σε όλα τα κελία της, στην οποία θα κρατούνται τα scores των στοιχίσεων. Λαμβάνει ως ορίσματα τα μήκη των δύο ακολουθιών που πρόκειται να στοιχηθούν *n, m* και την τιμή του gap penalty. Η μήτρα είναι διαστάσεων *(n+1, m+1)*, όπου n το μήκος της πρώτης ακολουθίας για την οποία θα εκτελεστεί η στοίχιση και m το αντίστοιχο μήκος για την δεύτερη ακολουθία. Στην συνέχεια, ενημερώνονται τα κελία της πρώτης στήλης και της πρώτης γραμμής προσθέτοντας το gap_penalty σε κάθε προηγούμενη τιμή του αντίστοιχου κελιού.

def calculate_scores(sequence1: list, sequence2: list, gap_penalty: int, match_reward: int, mismatch_penalty: int):

Η συνάρτηση αυτή πραγματοποιεί τον υπολογισμό των scores της μήτρας βαθμολόγησης και λαμβάνει ως ορίσματα τις δύο αλληλουχίες προς στοίχιση *sequence1, sequence2,* καθώς και τις τιμές των *gap_penalty*, *match_reward* και *mismatch_penalty*. Ξεκινώντας, αρχικοποιείται η μήτρα βαθμολόγησης μέσω της *initialize_matrix()* και ύστερα εντός ενός διπλού βρόχου που επαναλαμβάνεται κατά μήκος των δύο αλληλουχιών:

1. Σύμφωνα με την διαδικασία που παρουσιάστηκε παραπάνω, στην Παρουσίαση Αρχικής Σκέψης, για να ληφθεί η τιμή του διαγώνιου score ελέγχεται εάν ο χαρακτήρες της μίας αλληλουχίας είναι ίδιος με τον αντίστοιχο χαρακτήρα της άλλης και
    a. Αν ναι, τότε προστίθεται το *match_reward* στην ήδη υπάρχουσα τιμή του διαγώνιου προς τα πάνω κελιού
    b. Αν όχι, τότε προστίθεται το mismatch_penalty στην ήδη υπάρχουσα τιμή του διαγώνιου προς τα πάνω κελιού
2. Το *score_up,* αντιπροσωπεύει το score που προέρχεται από το πάνω κελί προσθέτοντας το *gap_penalty*
3. Το *score_left,* αντιπροσωπεύει το score που προέρχεται από το αριστερό κελί προσθέτοντας το *gap_penalty*

Τέλος επιστρέφεται η μήτρα βαθμολόγησης.

def traceback(matrix: list, sequence1: list, sequence2: list, gap_penalty: int, match_reward: int, mismatch_penalty: int):

Η συνάρτηση αυτή πραγματοποιεί την ανακατασκευή των στοιχισμένων ακολουθιών βάση της μήτρας βαθμολόγησης και λαμβάνει ως ορίσματα τις δύο αλληλουχίες προς στοίχιση *sequence1, sequence2,* καθώς και τις τιμές των *gap_penalty*, *match_reward* και *mismatch_penalty*. Ξεκινώντας, από το τελευταίο κελί της μήτρας (κάτω δεξιά),

επαναλαμβάνεται για όσο *i>0* και *j>0*, δηλαδή για όσο δεν έχει φτάσει στην αρχή της μήτρας βαθμολόγησης:

1. Λαμβάνεται το score του τρέχοντος κελιού, το score του διαγώνια αριστερά κελιού και το score του από πάνω κελιού
2. Ελέγχεται εάν ο χαρακτήρες της μίας αλληλουχίας είναι ίδιος με τον αντίστοιχο χαρακτήρα της άλλης και
    a. Αν ναι, τότε θα προστεθεί το *match_reward* στο επόμενο βήμα
    b. Αν όχι, τότε θα προστεθεί το *mismatch_penalty* στο επόμενο βήμα
4. Ελέγχεται εάν το score του τρέχοντος κελιού είναι ίσο με αυτό της διαγώνιου αφού προστεθεί και το *match_reward* ή το *mismatch_penalty,* ανάλογα τι προέκυψε από το προηγούμενο βήμα.
    a. Αν ναι, τότε οι χαρακτήρες των αλληλουχιών *sequence1* και *sequence2* των θέσεων *i-1* και *j-1* προστίθενται στις στοιχισμένες αλληλουχίες *aligned_sequence1* και *aligned_sequence2,* αντίστοιχα. Επειδή η ανακατασκευή γίνεται αντίστροφα (από το τέλος προς την αρχή), πρώτα προστίθεται ο χαρακτήρας και ύστερα το υπόλοιπο μέρος της
    b. Αν το score ισούται με το score του από πάνω κελιού αφού προστεθεί και το *gap_penalty,* τότε ο χαρακτήρας της θέσης *i-1* της αλληλουχία *sequence1* θα προστεθεί στο *aligned_sequence1,* ενώ στο *aligned_sequence2* θα προστεθεί το κενό '-'
    c. Σε κάθε άλλη περίπτωση συμβαίνει το ακριβώς αντίστροφο που περιεγράφηκε στο *b*

Ύστερα, για να διαχειριστούν οι ακραίες περιπτώσεις (πρώτη γραμμή και πρώτη στήλη), υπάρχουν δύο βρόχοι που εκτελούνται όταν *i>0, j>0*, αντίστοιχα. Όταν *i>0*, τότε συμβαίνει ότι περιεγράφηκε στο 3b ενώ στην περίπτωση του βρόχου *j>0*, τότε συμβαίνει ότι περιεγράφηκε στο 3c. Τέλος, επιστρέφονται οι δύο στοιχισμένες ακολουθίες.

def multiple_sequence_alignment(sequences: list, alpha: int):

Η συνάρτηση αυτή πραγματοποιεί την πολλαπλή στοίχιση, υιοθετώντας καθολική στοίχιση για όλες της αλληλουχίες του *datasetA,* και λαμβάνει ως ορίσματα το σύνολο των αλληλουχιών προς στοίχιση και την τιμή α, που χρησιμοποιείται στον ορισμό τον παραμέτρων. Έτσι, αρχικά ορίζουμε το *gap_penalty = -a, match_reward=1, mismatch_penalty = -a/2,* όπως αναγράφεται στην εκφώνηση. Ύστερα, ορίζουμε το σύνολο *aligned_sequences* όπου θα περιέχει της στοιχισμένες αλληλουχίες, με πρώτη την μη στοιχισμένη αλληλουχία 1. Στην συνέχεια, εντός ενός βρόχου που επαναλαμβάνεται έως ότου να μην υπάρχουν άλλες αλληλουχίες προς στοίχιση:

1. Ως αλληλουχία 1 ορίζεται η τελευταία αλληλουχία του πίνακα *aligned_sequences* και ως αλληλουχία 2 ορίζεται η επόμενη μη στοιχισμένη ακολουθία του πίνακα *sequences*
2. Σε αυτές τις δύο αλληλουχίες εφαρμόζεται η συνάρτηση calculate_scores(), που θα επιστρέψει τον πίνακα βαθμολόγησης ο οποίος θα περαστεί ως όρισμα στην συνάρτηση traceback() που επιστρέφει τις δύο στοιχισμένες αλληλουχίες
3. Ενημερώνεται ο πίνακας *aligned_sequences*

Τέλος επιστρέφεται ο πίνακας *aligned_sequences*.

2.3.1.3 iii.py

Το αρχείο αυτό περιέχει την υλοποίηση για το ερώτημα iii. Επιπλέον χρησιμοποιούνται οι βιβλιοθήκες Counter της Collections για διευκόλυνση καταμέτρησης και pandas για οπτικοποίηση των πιθανοτήτων.

def emission_probabilities(aligned_sequences, chars):
Η συνάρτηση αυτή πραγματοποιεί τον υπολογισμό των πιθανοτήτων εκπομπής του HMM, και λαμβάνει ως όρισμα τις στοιχισμένες αλληλουχίες, και τους επιτρεπόμενους χαρακτήρες στην αλληλουχία. Οι πιθανότητες εκπομπής υπολογίζονται σύμφωνα με τον τύπο,

$$Pemission(i) = freq(i) / sum(freq)$$

Πρώτα, αρχικοποιείται ένα λεξικό το οποίο θα αποθηκεύει της πιθανότητες εκπομπής για κάθε θέση της στοιχισμένης αλληλουχίας. Στη συνέχεια, για κάθε μια θέση (ανά στήλη), μετράει την συχνότητα του κάθε χαρακτήρα, από τον πίνακα *chars,* στη δεδομένη θέση, για κάθε αλληλουχία, χρησιμοποιώντας την *Count* και ένα εμφωλευμένο βρόχο. Ύστερα, πραγματοποιεί τον υπολογισμό της πιθανότητας, διαιρώντας την συχνότητα του χαρακτήρα σε εκείνη την θέση, με τον συνολικό αριθμό των χαρακτήρων σε εκείνη την θέση. Επιπλέον, σε περίπτωση που κανένας χαρακτήρας δεν είναι παρών σε εκείνη την θέση σε καμία από τις στήλες, τότε η πιθανότητα ορίζεται ίση με 0.

def transition_probabilities(aligned_sequences):
Η συνάρτηση αυτή πραγματοποιεί τον υπολογισμό των πιθανοτήτων μετάβασης από μία κατάσταση σε μία άλλη, του HMM, και λαμβάνει ως όρισμα τις στοιχισμένες αλληλουχίες. Στην αρχή, ορίζονται τα δύο δυσδιάστατα λεξικά *transitions* και *state_count* στα οποία θα αποθηκευτούν ο αριθμός μεταβάσεων μεταξύ των καταστάσεων και ο αριθμός των μεταβάσεων από κάθε κατάσταση αντίστοιχα. Στη συνέχεια, για κάθε στοιχισμένη αλληλουχία, ορίζεται η προηγούμενη κατάσταση ως η αρχική *S,* και αν αυτή δεν υπάρχει (πρώτη επανάληψη), αρχικοποιούνται τα λεξικά. Μετά, εντός βρόχου που επαναλαμβάνεται για κάθε χαρακτήρα των αλληλουχιών, γίνεται

1. Ελέγχεται εάν η κατάσταση είναι αφαίρεσης, δηλαδή ο χαρακτήρας είναι ‘-’, ή ταιριάσματος, δηλαδή κάθε άλλη περίπτωση και ορίζεται η τρέχουσα κατάσταση ως *‘D’* ή *‘M’,* ανάλογα.
2. Εάν η τρέχουσα κατάσταση δεν είναι στις μεταβάσεις, τότε ο counter μεταβάσεων για την τρέχουσα κατάσταση ορίζεται ίση με 0 και ύστερα αυξάνονται κατά 1, τόσο το λεξικό μεταβάσεων και καταστάσεων, ενώ ορίζεται η τρέχουσα κατάσταση ως προηγούμενη. Ύστερα ελέγχεται αν η προηγούμενη κατάσταση που μόλις ορίστηκε έχει ήδη οριστεί στα λεξικά, ώστε να ενημερωθούν ανάλογα.

Αφού τερματίσει ο βρόχος, σημαίνει ότι πλέον βρίσκεται σε κατάσταση τερματισμού, επομένως αυτή ορίζεται. Πριν υπολογιστούν οι πιθανότητες, ενημερώνεται τα λεξικά με την κατάσταση προσθήκης *I,* ώστε να συμπεριλάβει μία κατάσταση I για κάθε μία από τις καταστάσεις, ακόμη και να αυτή δεν παρατηρήθηκε στις αλληλουχίες. Το βήμα αυτό γίνεται για λόγους πληρότητας. Φτάνοντας στο τέλος, οι πιθανότητες μετάβασης υπολογίζονται ως εξής

1. Για κάθε προηγούμενη κατάσταση στο λεξικό μεταβάσεων ορίζεται *transition_probabilities[previous_state] = {}*
    a. Για την τρέχουσα κατάσταση στο λεξικό μεταβάσεων ορίζεται η πιθανότητα μετάβασης από την προηγούμενη κατάσταση στην τρέχουσα κατάσταση, ως

(αριθμός μεταβάσεων από προηγούμενη στη τρέχουσα) /
(συνολικός αριθμός μεταβάσεων από προηγούμενη στη τρέχουσα)

Τέλος επιστρέφεται το λεξικό με τις πιθανότητες μετάβασης.

def viterbi_algorithm(sequence, emission_probabilities, transition_probabilities, chars):
Η συνάρτηση αυτή πραγματοποιεί την υλοποίηση του αλγόριθμου *Viterbi*, υπολογίζοντας τα alignment path και alignment scores των αλληλουχιών του *datasetB,* και λαμβάνει ως όρισμα τις στοιχισμένες ακολουθίες, τις πιθανότητες εκπομπής, τις πιθανότητες μετάβασης από μία κατάσταση σε μία άλλη και τους χαρακτήρες που υπάρχουν στις ακολουθίες, δηλαδή τα νουκλεοτίδια *A, C, G, T*. Παρακάτω παρατίθεται ο ψευδοκώδικας του αλγορίθμου.

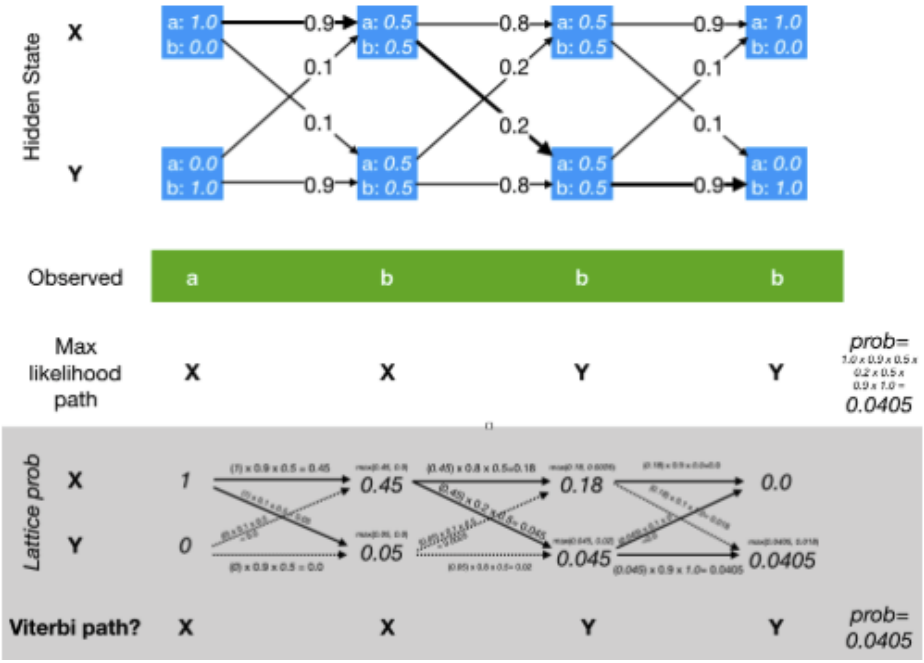
Αρχικοποίηση
- Δημιουργία του πίνακα όπου θα αποθηκευτούν οι πιθανότητες του πιο πιθανού alignment path (*Viterbi matrix*).
- Δημιουργία του πίνακα μονοπατιών όπου θα αποθηκεύονται οι καταστάσεις των πιο πιθανών μονοπατιών (*path matrix*).

Αρχικοποίηση πιθανοτήτων
- Ορίζονται οι αρχικές τιμές του *Viterbi matrix* και του *path matrix*.

Για κάθε αλληλουχία
- Για κάθε χαρακτήρα της αλληλουχίας
    - Υπολογισμός της πιθανότητας κάθε κατάστασης (*M, D, I*).
    - Ενημέρωση του *Viterbi matrix* με την μέγιστη πιθανότητα.
    - Ενημέρωση του *path matrix* με την κατάσταση που αντιστοιχεί στην μέγιστη πιθανότητα.

Traceback για εύρεση της πιο πιθανής ακολουθίας
- Αρχίζοντας από το τέλος της αλληλουχίας.
- Οπισθοδρόμηση μέσα στο *path matrix* για να βρεθεί η πιο πιθανή αλληλουχία των κρυμμένων καταστάσεων.

Παρακάτω φαίνεται στιγμιότυπο από την διαδικασία (βλέπε πηγές).



Εικόνα 4. Αλγόριθμος Viterbi

Όσον αφορά τον κώδικα αυτόν καθαυτών, γίνεται η αρχικοποίηση των δύο δυσδιάστατων πινάκων, *Viterbi* και *path*, μέσω της *np.zeros(),* και ύστερα αρχικοποιείται η πρώτη στήλη και η πρώτη γραμμή με τον λογάριθμο των πιθανοτήτων μετάβασης και μία πολύ μικρή τιμή αντίστοιχα (-10e-8). Γενικά, όπως ζητήθηκε χρησιμοποιήθηκαν λογάριθμοι των πιθανοτήτων μεταβάσεις. Συνεχίζοντας, εντός βρόχου, αρχίζοντας από τον τελευταίο χαρακτήρα της ακολουθίας ενημερώνουμε την θέση *i,j* του *viterbi_matrix*, παίρνοντας την μέγιστη εκ των τιμών

1. Ταίριασμα, που ισούται με την προηγούμενη διαγώνια τιμή και τον λογάριθμο των πιθανοτήτων μετάβασης (από κατάσταση ταιριάσματος σε κατάσταση ταιριάσματος) και εκπομπής
2. Αφαίρεση, που ισούται με την προηγούμενη διαγώνια τιμή και τον λογάριθμο των πιθανοτήτων μετάβασης (από κατάσταση ταιριάσματος σε κατάσταση ταιριάσματος). Εδώ η πιθανότητα εκπομπής είναι αποόυσα.
3. Εισαγωγή, που ισούται με την προηγούμενη διαγώνια τιμή και τον λογάριθμο των πιθανοτήτων μετάβασης (από κατάσταση ταιριάσματος σε κατάσταση εισαγωγής) και εκπομπής

Σημειώνεται ότι στους λογάριθμους προστίθεται μια πολύ μικρή τιμή 1e-8, ώστε να αντιμετωπιστούν οι περιπτώσεις που η πιθανότητα μπορεί να είναι 0. Επιπλέον χρησιμοποιείται η μέθοδος *log()* της *numpy*.
Μετά, ενημερώνουμε το *path_matrix* με τις τιμές 1 ή 2 ή 3 ανάλογα αν η τιμή αντιστοιχεί σε ταίριασμα ή διαγραφή ή εισαγωγή. Ως τελευταίο βήμα, στην οπισθοδρόμηση, ελέγχουμε τις τιμές του *path_matrix* ώστε να κινηθούμε στην αντίστοιχη κατεύθυνση στον πίνακα για να σχηματίσουμε την στοιχισμένη αλληλουχία, όπως και στην οπισθοδρόμηση του ερωτήματος ii. Τέλος επιστρέφεται ο πίνακας *alignment_score* που στην ουσία είναι το *viterbi_matrix,* και η στοιχισμένη αλληλουχία.

Για να τρέξει το πρόγραμμα, στην *main*, φορτώνεται το αρχείο *aligned_sequences* με τις στοιχισμένες αλληλουχίες του *datasetA*, και μέσω της *DataFrame*, εκτυπώνονται οι πιθανότητες εκπομπής και μετάβασης. Στη συνέχεια χρησιμοποιείται ο αλγόριθμος *Viterbi* για τις αλληλουχίες του *datasetB,* για να βρεθούν τα alignment paths (οι στοιχισμένες αλληλουχίες) και τα alignment scores. Και στις δύο περιπτώσεις οι αλληλουχίες γίνονται padded, προσθέτοντας ‘-’ στο τέλος, ώστε όλες να έχουν το ίδιο μήκος.

3. ΕΠΙΔΕΙΞΗ ΤΗΣ ΛΥΣΗΣ

Για να αναδείξουμε την λειτουργία των προγραμμάτων θα τρέξουν με τη σειρά τα προγράμματα *i.py, ii.py*, *iii.py*.

3.1.1 i

datasetA

```
1   ACATTGCGCTTGGACTCATTTATTCTTAT
2   ACGAATTGACGCTTAGGCTCATTTATTCGTGT
3   ACAATTGACGCTTATGACTCATTTATCGAC
4   AAATTGACGCTTATTGACTATTTAATAGTAA
5   AAATTGATGCTTAGGACTCATTTTTCGTAC
6   ACGAATTGACGCTTATGGACTCATTTAATCGTAA
7   ACGAATTGAGGCTTATGGACACATGTAGTCGTAT
8   AATTTACTCTTATGGACTATTATTCGTACA
9   AAATTGACGCTTATGGACTCATTTATTCGTAC
10  ACAATGAGCTTATGACTCATTATTCGTACA
11  ACTATAGACGCTATGACTAATTTATTCGTAAG
12  ACAATTGACCTTAGGAGTCATTTTTTGTAAG
13  AAATTGAAGCTTATGGACTCATTATTCGTAA
14  ACGATTGGTTTATGGACTCATTTATCTAC
15  ACGAATTGACGCTTATGGGCTAATTATTCGTAGA
16
```

datasetB.txt

```
1   ACATGACGCTTATGGCTCATTTATTCGTAT
2   ACGAATTGCGATTATGGACTCATTTATTCCTAGA
3   ATATTGACGCTTATGACTCATTTATTCTAT
4   ACAATTGACGCTTAGGACTCATTTATTCTAC
5   ACGGATGACTCTTAGGACTCATTATTCGTAAC
6   ACATTGACGTTAGGGACTCATTTATTCGTAT
7   ACAATAGAGGCTTATGTACTCATTTATTCGTATT
8   AATTAACGCTTATGGACTCATATATTCGTAG
9   AAATTGACGCCTATGGACTCTTTATTCGTAT
10  AATTGACGCTTATGGACTCATTTATTCGTAA
11  ACAATTTACGCTTATGGCTCATTATTCGTAGG
12  ACAATTGACGCTTATGGACTATGTATTCGTAG
13  AAATTGACGCTTATGACTCATTTATTCGTAT
14  AAATTTCGCTTATTGACTCATTTATTCGCAC
15  ACGAATTGACGCTTATGGACTCTTTATTCGTAT
16  ACGAATTGACGCCTATGGCTCATTTATTCGTAAG
17  ACGATTGACGCTTGTTGACTCACTTTTCGCAA
18  AAATGACGCTTATGGACTCATTTATTGTAGC
19  ACATTGACGCTTATGGCCTCTTTATTGTATA
20  ACAATTTAGCTTATGGACTGATTTATTCGTTCG
21  AAAGTGACGTTATGGACTCTTTATTTGTATT
22  ACAATGCGCTTCGGAGTCATTTATTCGTAT
23  ACGAATTGACATTATGGACCATTTATTCGTAG
24  AAATGACGCTATGGACTCATTTATTCGTG
25  AAATTGACGTTTTGGATCTTTTATTCGTAGA
26  ACAATAGACGTTATGACTCATTTATTCGTAC
27  ACATTGACGCTTATGGACTCATTATTCGTAT
28  ACAATTGACGCTTATGCTCATTTATTCTAC
29  ACGATTGACTCTATGGACTCATTTATCGTATA
30  ACAATTGACGCTTTGGCTCATTTATTCCTATT
31  ACAATTACGCTTATGGCTCATTTATTCGTAGC
32  ACGAATTGACGCTTATGGGCTCATTTATTGTAT
33  AAATTGGCGCTATGGACTTATTTAGTCGTAG
34  ACATTGAGCTATGGACTCATTATTCGTAT
35  AAATTGACGCTTATGGACTCTTTATCGTAGC
36
```

Εικόνα 5. datasetA και datasetB

3.1.2 ii

aligned_sequences.txt

```
1   AC--ATTG-CGCTTGGACTCATTTATTCTTAT
2   ACGAATTGACGCTTA-GGCTCATTTATTCGTGT
3   AC-AATTGACGCTTA-TGACTCATTTA-TCG-AC
4   A--AATTGACGCTTATTGACT-ATTTAATAGTAA
5   A--AATTGATGCTTA-GGACTCATTT-TTCGTAC
6   ACGAATTGACGCTTATGGACTCATTTAATCGTAA
7   ACGAATTGAGGCTTATGGACACATGTAGTCGTA-T
8   ---AATTTACTCTTATGGAC-TAT-TATTCGTACA
9   --AAATTGACGCTTATGGACTCATTTATTCGTAC-
10  --ACAAT-GA-GCTTAT-GACTCA-TTATTCGTACA
11  --ACTATAGACGC-TAT-GACTAATTTATTCGTAAG
12  --ACAATTGAC-CTTA-GGAGTCATTT-TTTGTAAG
13  --A-AATTGAAGCTTATGGACTCA-TTATTCGTAA-
14  --ACG-ATTG--GTTTATGGACTCATTTA-TC-T-AC
15  --ACGAATTGACGCTTATGGGCT-AATTATTCGTAGA
16
```

Εικόνα 6. Στοιχισμένες αλληλουχίες του datasetA

3.1.3 iii

Transition Probability Table aligned DatasetA

	M	D	I	E
S	0.437500	0.500000	0.062500	0.000001
M	0.897872	0.095745	0.002128	0.004255
D	0.459770	0.379310	0.011494	0.149425

Εικόνα 7. Πιθανότητες μετάβασης των στοιχισμένων αλληλουχιών του datasetA

Emission Probability Table aligned DatasetA

	A	C	G	T
0	0.466667	0.000000	0.000000	0.000000
1	0.000000	0.333333	0.000000	0.000000
2	0.466667	0.000000	0.200000	0.000000
3	0.533333	0.333333	0.000000	0.000000
4	0.800000	0.000000	0.133333	0.066667
5	0.333333	0.000000	0.000000	0.600000
6	0.133333	0.000000	0.000000	0.866667
7	0.066667	0.000000	0.533333	0.333333
8	0.533333	0.000000	0.266667	0.133333
9	0.266667	0.466667	0.200000	0.066667
10	0.133333	0.133333	0.533333	0.066667
11	0.000000	0.666667	0.200000	0.000000
12	0.000000	0.266667	0.133333	0.600000
13	0.000000	0.066667	0.000000	0.866667
14	0.533333	0.000000	0.066667	0.400000
15	0.266667	0.000000	0.066667	0.466667
16	0.200000	0.000000	0.400000	0.333333
17	0.000000	0.066667	0.666667	0.133333
18	0.466667	0.066667	0.400000	0.066667
19	0.266667	0.533333	0.133333	0.066667
20	0.200000	0.266667	0.133333	0.333333
21	0.066667	0.466667	0.000000	0.400000
22	0.533333	0.200000	0.000000	0.266667
23	0.266667	0.066667	0.000000	0.600000
24	0.200000	0.000000	0.066667	0.533333
25	0.133333	0.000000	0.000000	0.866667
26	0.400000	0.000000	0.000000	0.533333
27	0.333333	0.066667	0.066667	0.400000
28	0.133333	0.066667	0.000000	0.800000
29	0.066667	0.400000	0.066667	0.400000
30	0.066667	0.200000	0.466667	0.266667
31	0.000000	0.133333	0.333333	0.466667
32	0.466667	0.000000	0.066667	0.333333
33	0.400000	0.266667	0.000000	0.133333
34	0.333333	0.066667	0.000000	0.066667
35	0.133333	0.000000	0.200000	0.000000
36	0.066667	0.066667	0.000000	0.000000

Εικόνα 8. Πιθανότητες εκπομπής των στοιχισμένων αλληλουχιών του datasetA

Εικόνα 9. Μέρος των alignment scores και alignment paths του datasetB

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] https://www.youtube.com/watch?v=9bCkAsaP_z4 (sequence alignment)

[2] https://www.youtube.com/watch?v=b6xBvl0yPAY (sequence alignment)

[3] https://www.youtube.com/watch?v=um8h3P216Fk (sequence alignment python tutorial)

[4] https://www.geeksforgeeks.org/sequence-alignment-problem/ (sequence alignment python tutorial)

[5] https://www.youtube.com/watch?v=cAzcOh2tI6Q&t=604s (HMM)

[6] https://www.youtube.com/watch?v=Bf55IpJfy-w (Viterbi algorithm)

[7] https://www.youtube.com/watch?v=xumc05Cwtdc (HMM problems)

[8] https://www.ebi.ac.uk/training/online/courses/pfam-creating-protein-families/what-are-profile-hidden-markov-models-hmms/ (HMM in MSA)

[9] https://www.geeksforgeeks.org/viterbi-algorithm-for-hidden-markov-models-hmms/ (Viterbi algorithm python tutoria)

[10] https://stats.stackexchange.com/questions/415670/viterbi-algorithm-for-finding-most-probable-path-with-varying-transition-probabi (HMM figure)

[11] Σύγγραμματα μαθήματος