ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ

Πρώτο Σύνολο Ασκήσεων

Κρατημένου Χριστίνα

1067495

Δ' έτος

2021-2022

Ερώτημα 1

Για την υλοποίηση αυτού του ερωτήματος, αρχικά, έγινε εγκατάσταση των κατάλληλων πακέτων για την χρήση της Biopython. Στη συνέχεια, μελετήθηκε το documentation και έγινε εξοικείωση με τις λειτουργίες της μέσω παραδειγμάτων (δημιουργία sequence, χρήση συναρτήσεων complement κ.α.). Προχωρώντας στα προβλήματα της Rosalind, έγινε η παρακάτω καταγραφή αποτελεσμάτων:

Introduction to the Bioinformatics Armory

Αφορά την καταμέτρηση αριθμού βάσεων σε μία ακολουθία. Χρησιμοποιώντας την συνάρτηση count() της Biopython, η καταμέτρηση γίνεται με ευκολία και ταχύτητα.
Ενδεικτικός κώδικας:

```
from Bio.Seq import Seq
seqq = Seq(lines)
seqq.count("A")
seqq.count("C")
seqq.count("G")
seqq.count("T")
```

GenBank Introduction

Αφορά την αναζήτηση σε βάση δεδομένων αλληλουχιών (GenBank). Η αναζήτηση γίνεται μέσω της Entrez, η οποία αποτελεί βάση δεδομένων μοριακής βιολογίας.
Ενδεικτικός κώδικας:

```
from Bio import Entrez

def prob(genus, dtmin, dtmax):
    Entrez.email = "xriskt@gmail.com"
    term = '%s[Organism] AND (%s[Publication Date] : %s[Publication
Date])'
    handle = Entrez.esearch(db="nucleotide", term=term)
    record = Entrez.read(handle)
    return record["Count"]

print (prob("Limnocottus", "2001/03/14", "2003/01/15"))
```

Data Formats

Αφορά την εύρεση ακολουθίας συσχετισμένη με δοσμένο ID, ενώ γίνεται αναφορά στο format δεδομένων του GenBank (FASTA).
Ενδεικτικός κώδικας:

```
from Bio import Entrez
Entrez.email = "xriskt@gmail.com"
handle = Entrez.efetch(db="nucleotide", id=["BT149866, NM_204821,
JX308817, NM_001194889, NM_001003102, JQ867090, JF927157,
NM_001135551, JN698988"], rettype="fasta")
records = handle.read()

print (records)
```

FASTQ format introduction

Αφορά την μετατροπή format δεδομένων από FASTQ σε FASTA, με την χρήση του bio.seqIO σε συνδυασμό με την διαχείριση txt αρχείων.

Ενδεικτικός κώδικας:

```
from Bio import SeqIO

def conv(fastq):
        SeqIO.convert(fastq, "fastq", "t_fasta.txt", "fasta")
conv('rosalind_tfsq')
```

Read Quality Distribution

Αφορά την εύρεση κάτω του μετρίου reads (κάτω από το μέσο Phred Quality Score) με την χρήση του bio.seqIO και των letter_annotations.
Σημείωση: Για να εισαχθεί το αρχείο που μας δίνεται στην κατασκευασμένη συνάρτηση και να διαβαστεί, πρέπει να αφαιρεθεί το κατώφλι ποιότητας που αναγράφεται στην αρχή του αρχείου και να αποθηκευτεί εκ νέου.

Ενδεικτικός κώδικας:

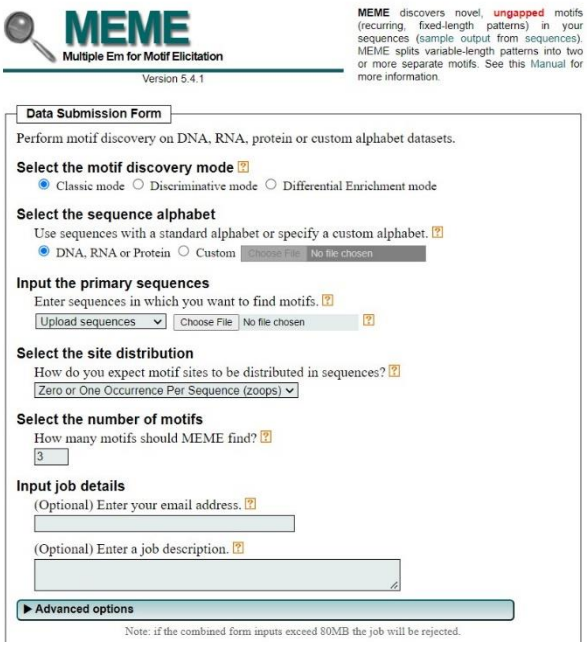
```
from Bio import SeqIO

def average(n):
        return sum(n) / float(len(n))
def ftq_thr(threshold, fastq):
        handle = SeqIO.parse(fastq, "fastq")
        belowthreshold = 0
        for record in handle:
                if
average(record.letter_annotations["phred_quality"]) < threshold:
                        belowthreshold += 1
        return belowthreshold
print(ftq_thr(26, "rosalind_phre.txt"))
```

New Motif Discovery

Αφορά την εύρεση μοτίβων σε ακολουθίες πρωτεϊνών. Προτείνεται ένα διαδικτυακό εργαλείο για την ανίχνευσή τους, το οποίο ονομάζεται Multiple Em for Motif Elicitation.

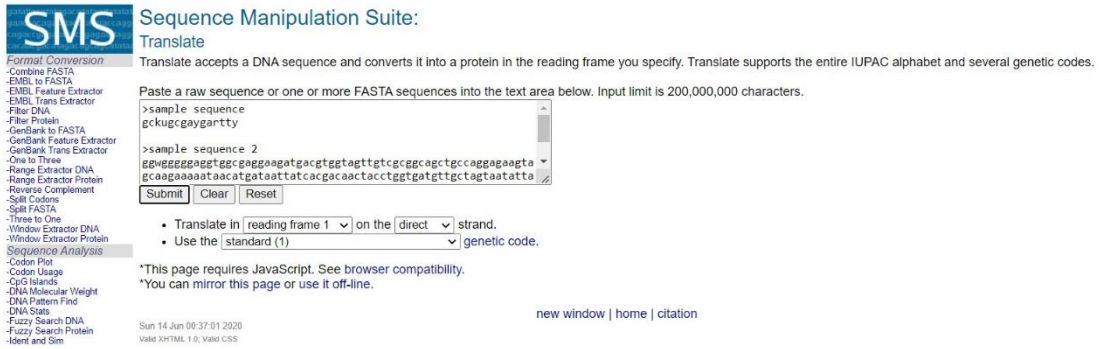
Ενδεικτικό screenshot:



Protein Translation

Αφορά την μετάφραση του DNA σε πρωτεΐνη, μια θεμελιώδη διαδικασία για το γενετικό υλικό. Το εργαλείο που προτείνεται για την επίλυση αυτού του προβλήματος είναι το Sequence Manipulation Suite (SMS).

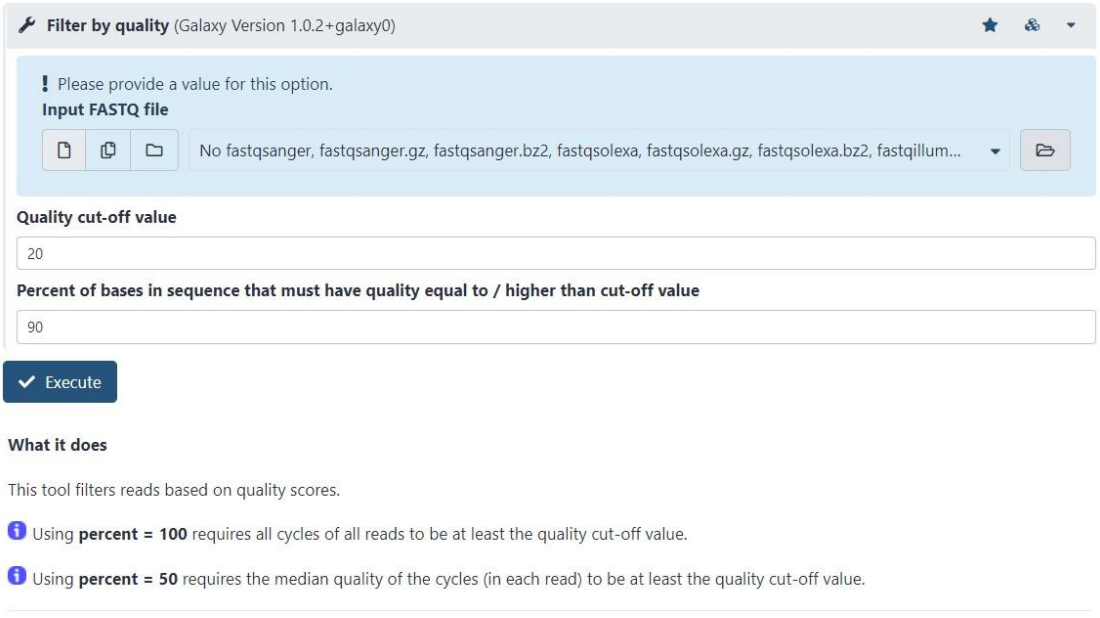
Ενδεικτικό screenshot:



Read Filtration by Quality

Αφορά το φιλτράρισμα των κακής ποιότητας reads με την χρήση του FASTQ Quality Filter. Το online εργαλείο για την χρήση του φίλτρου αυτού μπορεί να βρεθεί στην πλατφόρμα Galaxy.

Ενδεικτικό screenshot:



Complementing a Strand of DNA

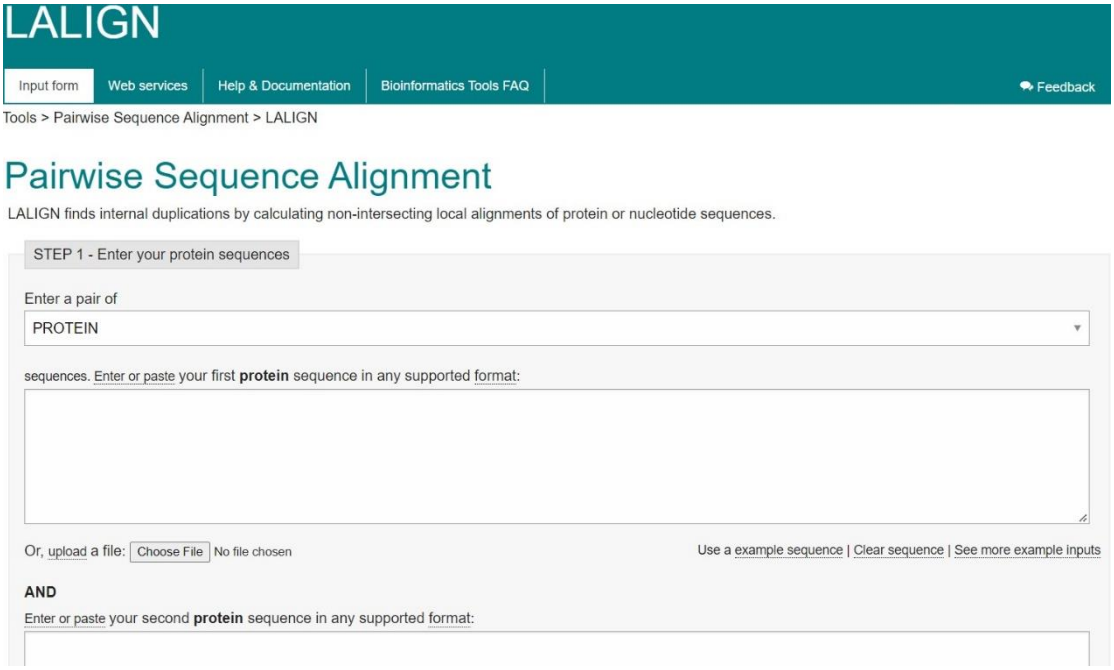
Αφορά την επίσης θεμελιώδη διαδικασία για το γενετικό υλικό, την συμπλήρωση μίας έλικας DNA. Για το πρόβλημα αυτό χρησιμοποιείται ξανά το Sequence Manipulation Suite.

Ενδεικτικό screenshot:

Suboptimal Local Alignment

Αφορά την εύρεση πολλαπλών εναλλακτικών ταιριασμάτων μεταξύ δύο ακολουθιών, μέσω του suboptimal alignment. Το εργαλείο που χρησιμοποιείται είναι το Lalign.

Ενδεικτικό screenshot:



Finding Genes with ORFs

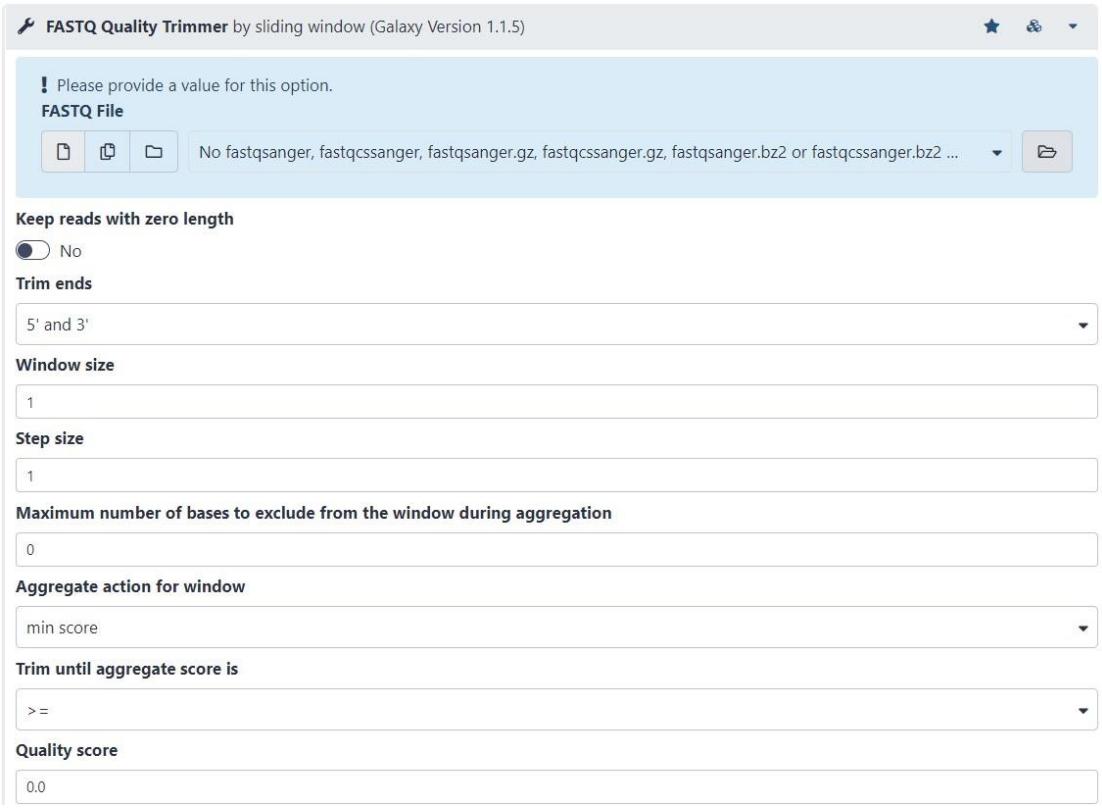
Αφορά την εύρεση πρωτεϊνών και γονιδίων από ακολουθία DNA με την χρήση ORFs (Open Reading Frame, αρχίζει με κωδικόνιο έναρξης τελειώνει με κωδικόνιο λήξης). Χρησιμοποιείται εργαλείο από το Sequence Manipulation Suite.

Ενδεικτικό screenshot:

Base Filtration by Quality

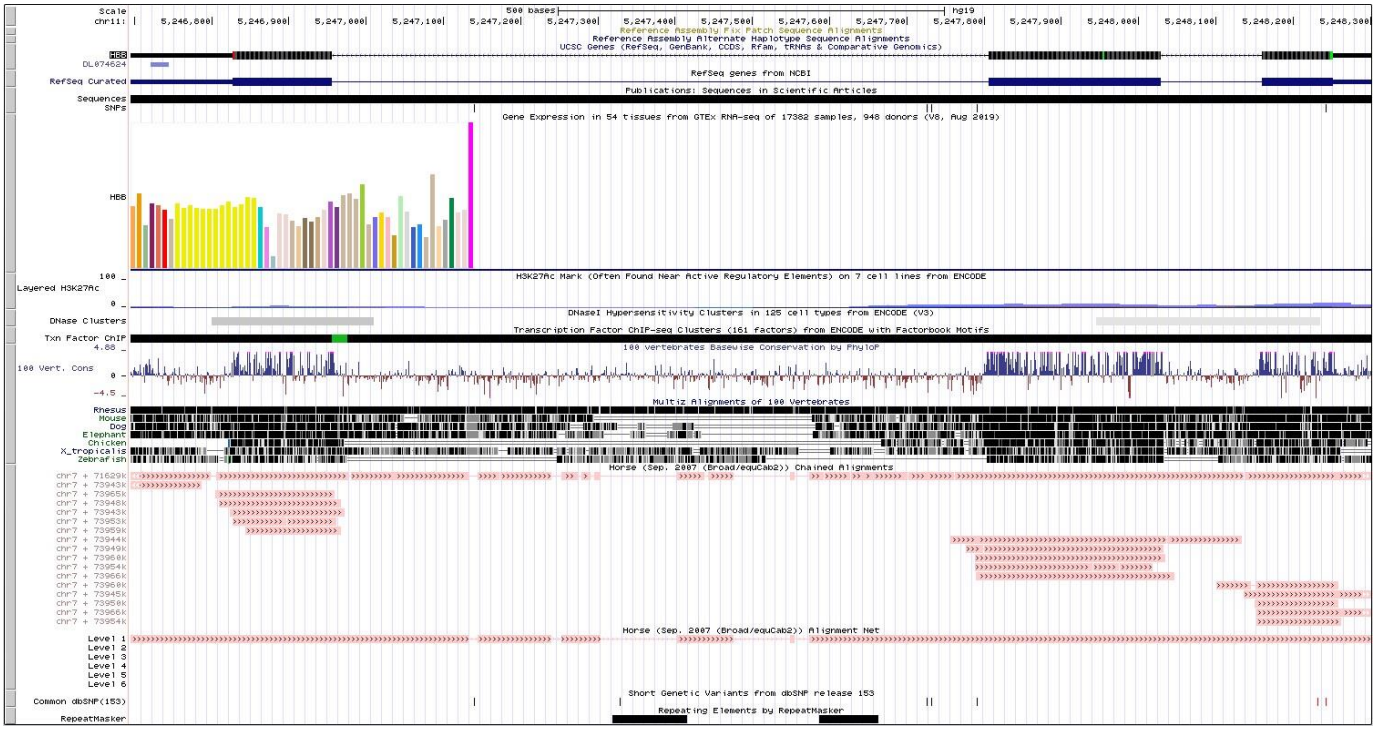
Αφορά την ανίχνευση και την απομάκρυνση βάσεων κακής ποιότητας από κάποια ακολουθία. Χρησιμοποιείται εργαλείο από την πλατφόρμα Galaxy.

Ενδεικτικό screenshot:

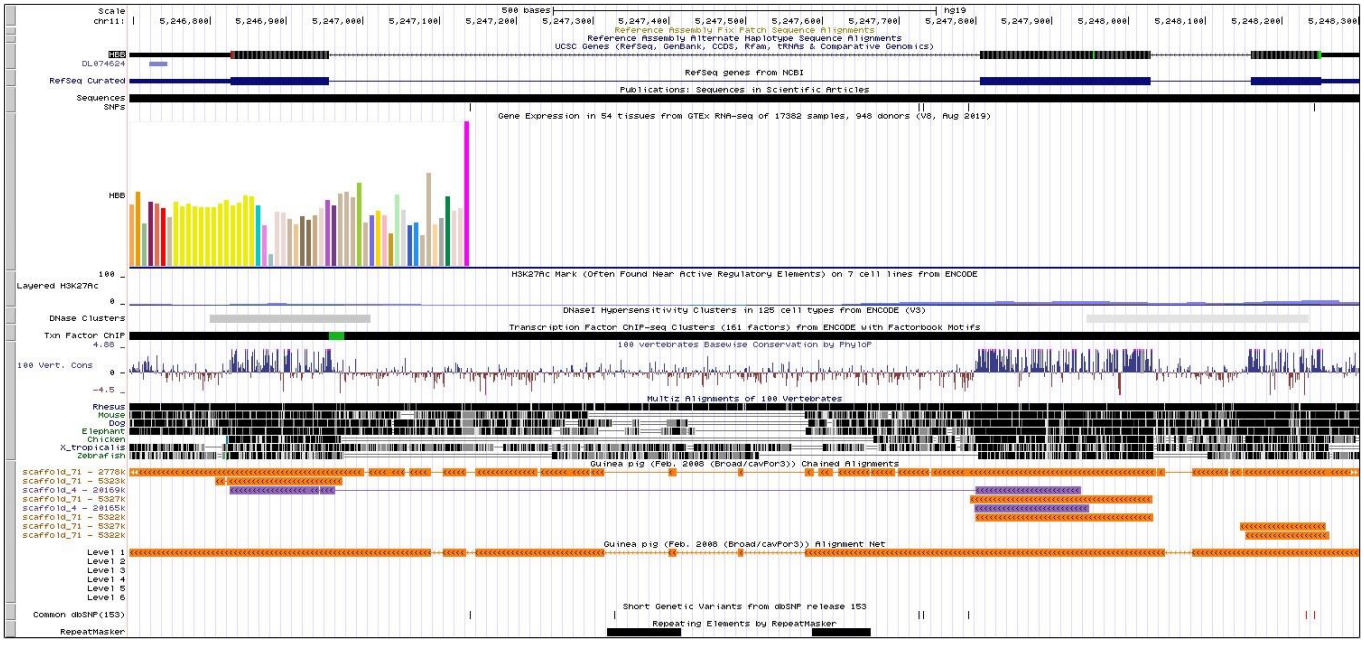


Ερώτημα 2

Human/Horse Chained Alignments – Alignment Nets



Human/Guinea Pig Chained Alignments – Alignment Nets



Human/Pig Chained Alignments – Alignment Nets

Ερώτημα 2

Στο πρώτο μέρος του ερωτήματος, ζητείται να αναπτυχθεί αλγόριθμός με χρήση Suffix Tree που ελέγχει, έπειτα από προεπεξεργασία *O(n)* χρόνου, εάν μία συμβολοσειρά *P* μήκους *m* εμφανίζεται σε μία συμβολοσειρά *T* μήκους *n*, πριν από την τιμή *k* σε χρόνο *O(m)*. Στο δεύτερο μέρος του ερωτήματος, ζητείται η κατασκευή αλγορίθμου που ανακαλύπτει την ελάχιστη συμβολοσειρά που εμφανίζεται μία μόνο φορά σε ένα κείμενο.

α) Ο παρακάτω αλγόριθμος λαμβάνει ως εισόδους τις συμβολοσειρές *T* και *P*, καθώς και τον αριθμό *k* που συμβολίζει τη θέση στην οποία σταματάει η σύγκριση των δυο συμβολοσειρών.

- Εισαγωγή *T*, *P* και *k*.
- Δημιούργησε Suffix Tree *ST* για την συμβολοσειρά *T*.
- Ξεκίνα από τον πρώτο χαρακτήρα της *P* και τη ρίζα του *ST* και συνέκρινε κάθε χαρακτήρα των συμβολοσειρών έναν προς ένα.
    - Για τον τρέχοντα χαρακτήρα της *P*, αν υπάρχει ακμή από τον τρέχοντα κόμβο του *ST*, τότε ακολούθησε την ακμή.
    - Αν δεν υπάρχει ακμή, δεν εμφανίζεται η συμβολοσειρά *P* στην *T*.
- Αν έχουν ελεγχθεί όλοι οι χαρακτήρες της *P* στο *ST* και ο αριθμός του φύλλου που εμφανίζεται η συμβολοσειρά *P* είναι μικρότερος του *k*, τότε έχουμε πλήρες ταίριασμα πριν τη θέση *k*.

Ο αλγόριθμος για το ταίριασμα των δύο συμβολοσειρών χρειάζεται χρόνο *O(m)* (*m* μήκος συμβολοσειράς *P*). Η δημιουργία του Suffix Tree χρειάζεται χρόνο *O(n)* (*n* μήκος συμβολοσειράς *T*).

β) Ο παρακάτω αλγόριθμος λαμβάνει είσοδο ένα κείμενο έστω *K*.

- Εισαγωγή *K*.
- Δημιούργησε Suffix Tree *ST* για το κείμενο *K*.
- Για κάθε θέση p του δέντρου, ακολουθεί διαδικασία εύρεσης του *LSUS(p)* (Locational Shortest Unique Substring).
    - Βρες το φύλλο του κόμβου που αντιστοιχεί στο επίθεμα *ST [p,n]*.

- Αν η ετικέτα της ακμής του φύλλου είναι *$*, τότε το *LSUS(p)* δεν υπάρχει και επίστρεψε null. Αλλιώς, συνέχισε τη διαδικασία εύρεσης.
- Έστω *L* το μήκος της ετικέτας της ακμής του φύλλου χωρίς το *$*.
- Τότε το *LSUS(p)* είναι *ST[p,n-L+1]*.
- Αποθήκευσε το *LSUS(p)*.
- Συνέκρινε όλα τα *LSUS* και επίλεξε το μικρότερο.

Ο αλγόριθμος για την εύρεση του *LSUS* για μία ορισμένη θέση χρειάζεται χρόνο *O(n)*, άρα για *m* θέσεις απαιτεί χρόνο $m \cdot O(n)$.

Ερώτημα 6

Οι γενετικές μεταλλάξεις συνήθως προκαλούνται από σφάλματα κυρίως κατά την διαδικασία του αναδιπλασιασμού. Στο πρώτο μέρος του ερωτήματος ζητείται να βρεθεί μία βέλτιστη στοίχιση, με την χρήση μεθόδου βαθμολόγησης με συγγενική ποινή ασυμφωνίας. Επομένως, οι ποινές για συνεχόμενες x ασυμφωνίες είναι $-(\rho+\sigma x)$ για $\rho>0$, το μπόνους για κάθε ταίριασμα είναι *+1* και η ποινή για προσθήκη και αφαίρεση συμβολοσειρών $-\rho$.
Ο σκοπός μιας στοίχισης, είναι να εντοπιστεί η βέλτιστη, όταν η διαδρομή είναι γνωστή.
Για δύο ακολουθίες μήκους *m*, το κόστος εύρεσης της βέλτιστης στοίχισης είναι $(n \cdot m)$.
Στο δεύτερο μέρος του ερωτήματος ζητείται ο σχεδιασμός για την εύρεση του μέγιστου κοινού προθέματος κάθε ζεύγους συμβολοσειρών *k* χαρακτήρων ενός συνόλου *s* συμβολοσειρών.

α) Έστω ότι η είσοδος είναι δύο ακολουθίες *S1* και *S2*.

- Συμβολίζεται ως (S_i, S_j) η στοίχιση του στοιχείου *i* με το στοιχείο *j* της δεύτερης ακολουθίας.
- Οι δύο αλληλουχίες τοποθετούνται σε έναν πίνακα $m \cdot n$ διαστάσεων, όπου κάθε στοιχείο του πίνακα είναι η τιμή της βαθμολογίας για την καλύτερη στοίχιση.
- Η γραμμή *0* και στήλη *0* αναπαριστούν το κόστος αν προστίθεντο διαδοχικά σφάλματα και στις δύο ακολουθίες.
- Όσο τα στοιχεία *i* και *j* είναι διαφορετικά του μηδέν, για οριζόντια στοίχιση ισχύει $S_{i,j-1} - \rho$, για κάθετη στοίχιση $S_{i-1,j} - \rho$ και για διαγώνια $S_{i-1,j} + 1$ αν $S_1 = S_2$ $S_{i-1,j} - (-\rho + \sigma x)$ αν $S_1 \neq S_2$.
- Η βαθμολογία προκύπτει ως το μέγιστο ανάμεσα στις οριζόντιες, κάθετες και διαγώνιες στοιχίσεις.
- Αν η βαθμολογία ταυτίζεται με την οριζόντια, τότε το *i* παραμένει ίδιο, ενώ το *j* γίνεται *j-1*.
- Αν η βαθμολογία ταυτίζεται με την κάθετη, τότε το *i* γίνεται *i-1*, ενώ το *j* παραμένει ίδιο.
- Αν η βαθμολογία ταυτίζεται με την διαγώνιο, τότε το *i* γίνεται *i-1*, ενώ το *j* γίνεται *j-1*, όπου υπάρχει ταίριασμα.

Παρατηρείται πως για δύο ακολουθίες μήκους *m*, το κόστος εύρεσης της βέλτιστης στοίχισης είναι $O(n \cdot m)$.

β) Το πρόβλημα εύρεσης μέγιστου κοινού προθέματος ζεύγους συμβολοσειρών ταυτίζεται με το πρόβλημα εύρεσης της μέγιστης κοινής επέκτασης μεταξύ συμβολοσειρών.

- Δημιούργησε Generalized Suffix Tree *ST* για το σύνολο των συμβολοσειρών *T1..TN*.
- Προεπεξεργάσου το *ST* δέντρο ώστε ο χαμηλότερος κοινός πρόγονος των φύλλων του *ST* για κάθε ζεύγος συμβολοσειρών και ένα ζεύγος δεικτών *i, j*, να είναι δυνατό να βρεθεί σε σταθερό χρόνο.

- Βρες τον χαμηλότερο κοινό πρόγονο των φύλλων του δέντρου που αντιστοιχεί στα επιθέματα των συμβολοσειρών που εξετάζεις. Έστω ότι αυτός ο κόμβος είναι ο *v*.
- Η συμβολοσειρά που αποτελεί την ετικέτα του μονοπατιού προς το *v*, αποτελεί την μέγιστη υποσυμβολοσειρά της ακολουθίας *T1* ξεκινώντας από τον δείκτη *i*, η οποία ταιριάζει με την υποσυμβολοσειρά της ακολουθίας *T2* ξεκινώντας από τον δείκτη *j*.
- Όποτε βρίσκεις την μέγιστη κοινή επέκταση, άρα και το μέγιστο κοινό πρόθεμα, μεταξύ ζεύγους συμβολοσειρών, εμφάνισε το *a*, που αποτελεί το πλήθος των μέγιστων επιθεμάτων.

Ο αλγόριθμος δουλεύει για ζεύγη συμβολοσειρών κάθε φορά. Με τον τρόπο αυτό προκύπτει η χρονική πολυπλοκότητα που ζητείται $O(k \cdot n + a)$, καθώς ο αλγόριθμος θα εκτελεστεί k φορές για τα k διαφορετικά ζεύγη συμβολοσειρών των n χαρακτήρων η καθεμία.

Ερώτημα 9

Δίνονται οι ακολουθίες v = GGTTCGTGGA και w = GATCGTGAATT.

α) Η ολική στοίχιση των δύο συμβολοσειρών επιτυγχάνεται με τον αλγόριθμο δυναμικού προγραμματισμού ολικής στοίχισης, Needleman-Wunsch. Στην πρώτη γραμμή και πρώτη στήλη υπάρχουν οι δείκτες για κάθε αζωτούχα βάση της ακολουθίας.
Για κάθε κελί του πίνακα υπολογίζεται η τιμή του σύμφωνα με τα παρακάτω:

- Το μονοπάτι του πάνω ή αριστερού κελιού αναπαριστά ένα ταίριασμα με κενό, επομένως στις τιμές των κελιών αυτών προστίθεται το κόστος στοίχισης με το κενό (-1).
- Το μονοπάτι του διαγώνιου κελιού αναπαριστά ένα ταίριασμα ή μία ασυμφωνία, επομένως στην τιμή του κελιού προστίθεται το κόστος ταιριάσματος ή ασυμφωνίας αντίστοιχα (+1, -1).

Στο κελί, τελικά, εισάγεται η μέγιστη αυτών των τιμών. Η δεύτερη γραμμή και η δεύτερη στήλη αρχικοποιούνται με το κόστος στοίχισης με το κενό. Κάθε φορά που συμπληρώνεται η τιμή ενός κελιού σημειώνεται από ποιο κελί προήλθε, για την εκτέλεση του backtracking. Το backtracking αρχίζει από το τελευταίο κελί του πίνακα, η τιμή του οποίου είναι και η τιμή της ολικής στοίχισης. Η στοίχιση που θα προκύψει από το backtracking θα είναι η βέλτιστη.

(Ο πίνακας δίπλα παράχθηκε με την χρήση online εργαλείου για global alignment.)

		G	A	T	C	G	T	G	A	A	T	T
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
G	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
G	-2	0	0	-1	-2	-1	-2	-3	-4	-5	-6	-7
T	-3	-1	-1	1	0	-1	0	-1	-2	-3	-4	-5
T	-4	-2	-2	0	0	-1	0	-1	-2	-3	-2	-3
C	-5	-3	-3	-1	1	0	-1	-1	-2	-3	-3	-3
G	-6	-4	-4	-2	0	2	1	0	-1	-2	-3	-4
T	-7	-5	-5	-3	-1	1	3	2	1	0	-1	-2
G	-8	-6	-6	-4	-2	0	2	4	3	2	1	0
G	-9	-7	-7	-5	-3	-1	1	3	3	2	1	0
A	-10	-8	-6	-6	-4	-2	0	2	4	4	3	2

Το τελικό σκορ, όπως φαίνεται και από το τελευταίο κελί του πίνακα, είναι 2.

G	A	T	-	C	G	T	G	A	A	T	T
G	G	T	T	C	G	T	G	G	A	-	-

β) Ο αλγόριθμος δυναμικού προγραμματισμού τοπικής στοίχισης είναι ο Smith-Waterman. Στην τοπική στοίχιση δεν συναντώνται αρνητικές τιμές στα κελιά του πίνακα, ενώ δίνεται έμφαση στις περιοχές με την μεγαλύτερη ομοιότητα. Οι αρνητικές τιμές μετατρέπονται σε 0 και το backtracking δεν αρχίζει από το τελευταίο κελί, αλλά από το κελί με τη μέγιστη ομοιότητα και καταλήγει στο πρώτο κελί με τιμή 0.

(Ο πίνακας δίπλα παράχθηκε με την χρήση online εργαλείου για local alignment.)

S		G_1	G_2	T_3	T_4	C_5	G_6	T_7	G_8	G_9	A_{10}
	0	0	0	0	0	0	0	0	0	0	0
G_1	0	1	1	0	0	0	1	0	1	1	0
A_2	0	0	0	0	0	0	0	0	0	0	2
T_3	0	0	0	1	1	0	0	1	0	0	1
C_4	0	0	0	0	0	2	1	0	0	0	0
G_5	0	1	1	0	0	1	3	2	1	1	0
T_6	0	0	0	2	1	0	2	4	3	2	1
G_7	0	1	1	1	1	0	1	3	5	4	3
A_8	0	0	0	0	0	0	0	2	4	4	5
A_9	0	0	0	0	0	0	0	1	3	3	5
T_{10}	0	0	0	1	1	0	0	1	2	2	4
T_{11}	0	0	0	1	2	1	0	1	1	1	3

Το τελικό σκορ, όπως φαίνεται και από το τελευταίο κελί της στοίχισης, είναι 5.

T	C	G	T	G	-	A
T	C	G	T	G	G	A