**Πρώτο Σύνολο Ασκήσεων 2022-2023**

Προφορική εξέταση 30/06/2023

**Ερώτημα 1:**

**Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα**Αφού επισκεφθήκαμε την ιστοσελίδα του Sequence Manipulation Suite (SMS), υπολογίσαμε ότι η αντίστροφη συμπληρωματική (αντιπαράλληλη) αλληλουχία της GCAATTCCC είναι η εξής:

Για τον υπολογισμό του της αντιπαράλληλης ακολουθίας χρησιμοποιήσαμε το εργαλείο Format Conversion -­­-> Reverse Complement που παρέχεται από τον ιστότοπο του SMS.

Εικόνα που περιέχει κείμενο, γραμμή, γραμματοσειρά, γράφημα

Περιγραφή που δημιουργήθηκε αυτόματα

Σύγκριση των εργαλείων του Sequence Manipulation Suite με τα εργαλεία του EMBOSS:

Ύστερα από εξερεύνηση και στις δύο ιστοσελίδες προέκυψαν οι εξής παρατηρήσεις:

* Το EMBOSS παρέχει πολύ περισσότερα εργαλεία και σε μεγαλύτερη ποικιλία από ότι το SMS.
* Η ιστοσελίδα του SMS είναι πιο εύκολη στην χρήση από ότι του EMBOSS καθώς τα εργαλεία παρέχονται online και μπορούν να χρησιμοποιηθούν εκείνη την στιγμή από τον χρήστη. Αντίθετα, στην αρχική σελίδα του EMBOSS (<https://emboss.sourceforge.net/>) για να χρησιμοποιηθούν τα εργαλεία θα πρέπει να γίνει η εκτέλεση κάποιων συγκεκριμένων εντολών σε command line.
* Στην σελίδα (<https://wwwdev.ebi.ac.uk/Tools/jdispatcher/emboss>) υπάρχουν εργαλεία που μπορεί κάποιος να χρησιμοποιήσει online. Παρόλα αυτά, ο χρήστης στο EMBOSS δεν έχει την δυνατότητα να αλλάξει τις παραμέτρους σε όλα τα εργαλεία όπως στο SMS.
* Τέλος, παρατηρήσαμε ότι τα εργαλεία του EMBOSS απαιτούν κάποιον παραπάνω χρόνο ώστε να εμφανίσουν τα αποτελέσματα από αυτά του SMS.

Παρακάτω έχουμε κάποια παραδείγματα χρήσης των εργαλείων τόσο του SMS όσο και του EMBOSS:

1. Χρήση εργαλείου για μετάφραση του DNA (Translate):

*Εικόνα που περιέχει κείμενο, γραμματοσειρά, γραμμή, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα****Sequence Manipulation Suite***

**Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, λευκό

Περιγραφή που δημιουργήθηκε αυτόματα**

***EMBOSS*** *(*[*https://wwwdev.ebi.ac.uk/Tools/jdispatcher/emboss*](https://wwwdev.ebi.ac.uk/Tools/jdispatcher/emboss)*)*

Εικόνα που περιέχει κείμενο, γραμμή, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Όπως φαίνεται και στις εικόνες παραπάνω παρατηρούμε ότι στο εργαλείο του EMBOSS μπορούμε να προσθέσουμε αρχεία ως είσοδο ενώ στο SMS αυτό δεν είναι εφικτό.

1. Χρήση εργαλείου για ολική στοίχιση (global alignment) 2 ακολουθιών:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα***Sequence Manipulation Suite***

***EMBOSS*** *(*[*https://wwwdev.ebi.ac.uk/Tools/jdispatcher/emboss*](https://wwwdev.ebi.ac.uk/Tools/jdispatcher/emboss)*)*

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Ακόμα, όσον αφορά τα αποτελέσματα τις στοίχισης, και στα δύο εργαλεία φαίνεται η στοίχιση και το σκορ αλλά στο εργαλείο του EMBOSS αναγράφονται περισσότερες λεπτομέρειες όπως πχ. το ποσοστό ομοιότητας των δύο ακολουθιών κ.α.

**Ερώτημα 2:**

1. Για αυτό το ερώτημα πραγματοποιήσαμε την αναζήτηση BLAST που ζητείται από την βάση δεδομένων NCBI χρησιμοποιώντας την αλυσίδα της φεριτίνης (NP\_000137).

Αποθηκεύσαμε 10 πρωτεΐνες από τα αποτελέσματα που προέκυψαν σε μορφή FASTA στο αρχείο με όνομα: seqdump.fasta.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

1. Για την υλοποίηση του συγκεκριμένου ερωτήματος επισκεφθήκαμε την ιστοσελίδα του EBI που περιέχει τα προγράμματα πολλαπλών στοιχίσεων MAFFT, MUSCLE και T-COFFEE.

Να σημειωθεί ότι κάθε αποτέλεσμα είναι αποθηκευμένο στην FASTA μορφή του και στα αρχεία που παραθέτουμε έχουμε τις default παραμέτρους για κάθε εργαλείο.

***MAFFT***

Εικόνα που περιέχει κείμενο, γραμμή, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματαΓια την χρήση του MAFFT χρησιμοποιήσαμε τις default παραμέτρους που φαίνονται παρακάτω:

Τα αποτελέσματα της στοίχισης τα έχουμε αποθηκεύσει στο αρχείο MAFFT.fasta.

Από τα αποτελέσματα που προκύπτουν μπορούμε να δούμε ότι οι πρωτεΐνες εμφανίζονται με συγκεκριμένη σειρά με τις μεγαλύτερες ακολουθίες πρωτεϊνών να βρίσκονται προς το τέλος του αρχείου.

Δοκιμάσαμε να εκτελέσουμε την πολλαπλή στοίχιση στο MAFFT με διαφορετικές παραμέτρους. Για πίνακες βαθμολόγησης BLOSUM η σειρά των πρωτεϊνών παραμένει ίδια και αλλάζει σε μικρό ποσοστό η στοίχιση στις πιο μεγάλες ακολουθίες πρωτεϊνών (όπως >VCX39883.1 unnamed protein product [Gulo gulo]).

Εικόνα που περιέχει κείμενο, γραμμή, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, γραμμή, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματαΑντίθετα, για πίνακες βαθμολόγησης JTT PAM αλλάζει η σειρά των πρωτεϊνών και οι ακολουθίες παρουσιάζονται καλύτερα και πιο ομοιόμορφα στοιχισμένες συγκριτικά με πριν.

Παρατηρούμε ότι αυξάνοντας το κόστος δημιουργίας και επέκτασης κενών η στοίχιση γίνεται περισσότερο ομοιόμορφη και οι ακολουθίες είναι στοιχισμένες προς το ίδιο σημείο προς τα δεξιά. Το ίδιο ισχύει και αλλάζοντας τους πίνακες βαθμολόγησης από BLOSUM σε JTT PAM.

***MUSCLE***

Τα αποτελέσματα της στοίχισης τα έχουμε αποθηκεύσει στο αρχείο MUSCLE.fasta.

Από τα αποτελέσματα που προκύπτουν μπορούμε να δούμε ότι οι πρωτεΐνες εμφανίζονται με διαφορετική σειρά από εκείνη στο MAFFT.

Οι πρωτεΐνες με μεγαλύτερη ακολουθία είναι πρώτες σε σειρά ενώ οι μικρότερες είναι προς το τέλος. Το συγκεκριμένο εργαλείο δεν διαθέτει παραμέτρους για τα κενά ή για τους πίνακες βαθμολόγησης.

***T-COFFEE***

Τα αποτελέσματα της στοίχισης τα έχουμε αποθηκεύσει στο αρχείο T-COFFEE.fasta.

Από τα αποτελέσματα που προκύπτουν μπορούμε να δούμε ότι οι πρωτεΐνες εμφανίζονται με διαφορετική σειρά από εκείνη στο MAFFT και MUSCLE. Συγκεκριμένα, το T-COFFEE φαίνεται να εμφανίζει τις πρωτεΐνες και να τις στοιχίζει με αλφαβητική σειρά σύμφωνα με το όνομα τους.

Το συγκεκριμένο εργαλείο δεν διαθέτει παραμέτρους για τα κενά. Έχει παραμέτρους μόνο για τους πίνακες (BLOSUM, PAM και None). Δοκιμάσαμε να αλλάξουμε τους πίνακες βαθμολόγησης αλλά δεν υπήρξε κάποια σημαντική αλλαγή στην στοίχιση.

***Συμπεράσματα***

Καταλήγοντας, μετά από χρήση των τριών εργαλείων μπορούμε να πούμε ότι με το MAFFT μπορούμε να πετύχουμε πιο ακριβείς στοιχίσεις εξαιτίας της δυνατότητας αλλαγής περισσότερων παραμέτρων που προσφέρει συγκριτικά με τα άλλα 2 εργαλεία.

**Ερώτημα 3:**

1. Για το συγκεκριμένο ερώτημα επισκεφθήκαμε τους συνδέσμους που αναφέρονται στην εκφώνηση και αποθηκεύσαμε σε μορφή FASTA τις πρωτεΐνες που ζητούνται: την spike πρωτεΐνη του κορονοϊού SARS-Cov-2 και την spike πρωτεΐνη για τον κορονοϊό Bat-RaTG13.

Τα αρχεία .fasta ονομάζονται CovSpikeProtein.fasta και BatSpikeProtein.fasta αντίστοιχα.

Για την υλοποίηση του κλασσικού αλγορίθμου ολικής στοίχισης δυναμικού προγραμματισμού με κατάλληλα βάρη για τον εντοπισμό της μέγιστης κοινής υποακολουθίας των 2 πρωτεϊνών, χρησιμοποιήσαμε την γλώσσα Python και συγκεκριμένα την βιβλιοθήκη **Biopython**.

Το αρχείο που περιέχει τον ζητούμενο κώδικα είναι το 3i.py και τυπώνει τα παρακάτω αποτελέσματα:

Αρχικά, εκτυπώνουμε την καλύτερη ολική στοίχιση (global alignment) που προκύπτει και στην συνέχεια φαίνεται το σκορ και η μέγιστη κοινή υποακολουθία (Longest Common Subsequence) των 2 ακολουθιών.

Τέλος, εμφανίζουμε τα μήκη για κάθε ακολουθία ώστε να μπορούμε να κάνουμε πιο εύκολα τις συγκρίσεις.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, κατάλογος

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Από το αποτελέσματα παρατηρούμε ότι οι 2 ακολουθίες των πρωτεϊνών έχουν υψηλό σκορ στοίχισης και αρκετά μεγάλη κοινή υποακουθία. Αυτό σημαίνει ότι ταιριάζουν μεταξύ τους και έχουν πολλές ομοιότητες.

1. Βλέποντας την δομή των 2 πρωτεϊνών του SARS-Cov-2 και του Bat-RaTG13 στο εργαλείο swiss-modeler παρατηρούμε ότι προκύπτουν 4 models για κάθε ακολουθία.

Συγκεκριμένα, από τα 4 αυτά μοντέλα τα 2 από αυτά είναι homo-trimers και 2 είναι monomers. Επομένως, τα αρχεία .pdb που προκύπτουν είναι 8 συνολικά, 4 για την κάθε πρωτεΐνη.

Στην συνέχεια, στο εργαλείο Dali κάνουμε τις συγκρίσεις των δομών των πρωτεϊνών με το εργαλείο Pairwise structure comparison όπου και συγκρίνουμε κάθε μοντέλο της μίας πρωτεΐνης με το αντίστοιχό του στην άλλη πρωτεΐνη, ξεχωριστά δηλαδή το καθένα.

Προκύπτουν 8 αρχεία συνολικά: 3 για την σύγκριση μεταξύ των model01 (homo-trimer), 3 για την σύγκριση μεταξύ των models02 (homo-trimer), 1 για σύγκριση των models03 (monomer) και 1 για σύγκριση των models04 (monomer).

Τα αρχεία που περιέχουν τα αποτελέσματα των συγκρίσεων είναι .txt αρχεία. Σύμφωνα με αυτά, υπάρχουν κάποια στοιχεία όπως τα Z, rmsd, lali, nres, %id που μας δίνουν πληροφορίες για τις ομοιότητες και τις διαφορές μεταξύ των δομών.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

***Παρατηρήσεις***

Ειδικότερα, παρατηρούμε ότι για κάθε ζευγάρι models το ποσοστό ταυτότητας των 2 δομών σε όλα τα αρχεία σύγκρισης των μοντέλων είναι %id>=88 που σημαίνει ότι οι δομές των 2 πρωτεϊνών είναι πολύ παρόμοιες μεταξύ τους.

Επιπλέον, στο ίδιο συμπέρασμα μπορούμε να καταλήξουμε και ελέγχοντας τα Z, rmsd κλπ. καθώς οι τιμές τους δηλώνουν ότι οι δομές έχουν πολλές ομοιότητες μεταξύ τους.

Εικόνα που περιέχει τέχνη

Περιγραφή που δημιουργήθηκε αυτόματα Εικόνα που περιέχει τέχνη

Περιγραφή που δημιουργήθηκε αυτόματα

*CovSpikeProtein Structure BatSpikeProtein Structure*

*(και τα 4 μοντέλα μαζί) (και τα 4 μοντέλα μαζί)*

**Ερώτημα 4:**

1. Αλγόριθμος χρονικής πολυπλοκότητας O(n) που βρίσκει το μέγιστο prefix repeat μιας ακολουθίας Τ μήκους n χαρακτήρων:

Ο αλγόριθμος περιλαμβάνει τα εξής βήματα:

1. Δίνουμε ως είσοδο μία ακολουθία Τ μήκους n χαρακτήρων.
2. Υπολογίζουμε το k , που είναι το μισό του μήκους της ακολουθίας. Αυτός ο περιορισμός εξασφαλίζει ότι ο αλγόριθμος θα εξετάσει μόνο πιθανά prefix που είναι μικρότερα από το μισό του μήκους της συμβολοσειράς.
3. Αρχικοποιούμε τη μεταβλητή max\_prefix\_repeat ως κενή συμβολοσειρά.
4. Ξεκινάμε έναν βρόχο επανάληψης που εκτελείται για κάθε τιμή του i από το 0 μέχρι το k-1.
5. Ελέγχουμε αν οι πρώτοι i+1 χαρακτήρες της ακολουθίας Τ είναι ίσοι με τους επόμενους i+1 χαρακτήρες. Αυτό γίνεται με τη σύγκριση if Τ[:i+1] == Τ[i+1:2\*i+2].
6. Αν η παραπάνω συνθήκη ισχύει, τότε η υποσυμβολοσειρά Τ[:i+1] είναι ένα prefix repeat. Ενημερώνουμε τη μεταβλητή max\_prefix\_repeat με αυτήν την υποσυμβολοσειρά.
7. Ο αλγόριθμος συνεχίζει με την επόμενη τιμή του i και επαναλαμβάνει τα βήματα 4-5.
8. Τελικά, η συνάρτηση επιστρέφει την max\_prefix\_repeat συνενωμένη με τον εαυτό της, παρέχοντας έτσι το μέγιστο prefix repeat της ακολουθίας T.
9. Σε περίπτωση που υπάρχει μέγιστο prefix repeat επιστρέφεται το μήνυμα «Maximum prefix repeat:» ακολουθούμενο από την ζητούμενη υποσυμβολοσειρά.
   1. Αλλιώς, επιστρέφουμε το μήνυμα «There is no prefix repeat.»

Το αρχείο που περιέχει τον κώδικα σε Python για τον παραπάνω αλγόριθμο είναι το 4a.py.

1. Ο αλγόριθμος χρησιμοποιεί απλή εφαρμογή δυναμικού προγραμματισμού, όπως στην ολική στοίχιση δύο συμβολοσειρών.

Ο αλγόριθμος περιλαμβάνει τα εξής βήματα:

1. Αρχικά, γίνεται έλεγχος αν οι δύο συμβολοσειρές έχουν το ίδιο μήκος (σύμφωνα με την εκφώνηση και οι 2 είναι μήκους n χαρακτήρων).
2. Αν οι συμβολοσειρές έχουν ίδιο μήκος:
   1. Δημιουργία ενός πίνακα dp με διαστάσεις (n-k+1) x (n-k+1), όπου κάθε στοιχείο dp[i][j] θα αντιστοιχεί στο σκορ-ταιριάσματος των υποσυμβολοσειρών T1[i:i+k] και T2[j:j+k].
   2. Αρχικοποίηση του πίνακα dp με μηδενικά.
   3. Εκτέλεση δύο εμφωλευμένων βρόχων για να υπολογίσουμε τις τιμές του πίνακα dp:
      1. Ο πρώτος βρόχος θα εξετάσει όλες τις θέσεις i από 0 έως n-k στην πρώτη συμβολοσειρά T1.
      2. Ο δεύτερος βρόχος θα εξετάσει όλες τις θέσεις j από 0 έως n-k στη δεύτερη συμβολοσειρά T2.
      3. Στο εσωτερικό των βρόχων, για κάθε θέση i και j, υπολογίζουμε το σκορ-ταιριάσματος των υποσυμβολοσειρών T1[i:i+k] και T2[j:j+k]. Αυτό γίνεται συγκρίνοντας τους αντίστοιχους χαρακτήρες και αυξάνοντας το σκορ για κάθε ταιριαστό ζευγάρι χαρακτήρων που προκύπτει.
   4. Όταν οι εμφωλευμένοι βρόχοι ολοκληρωθούν, ο πίνακας dp θα περιέχει τις τιμές σκορ-ταιριάσματος για όλα τα ζεύγη υποσυμβολοσειρών.
   5. Επιστροφή του πίνακα dp.
3. Αν δεν έχουν ίδιο μήκος, τυπώνεται αντίστοιχο μήνυμα: «Οι συμβολοσειρές πρέπει να έχουν το ίδιο μήκος.» και θα πρέπει να δοθούν νέες συμβολοσειρές ίδιου μήκους.

Το αρχείο που περιέχει τον κώδικα σε Python είναι το 4b.py.

**Ερώτημα 5:**

Τοπική Στοίχιση (Local Alignment) των 2 ακολουθιών v= ATATCGACGA και w= ATCCGAGAATT με τα παρακάτω βάρη:

Κόστος στοίχισης: +1

Κόστος ασυμφωνίας: -1

Κόστος στοίχισης με κενό: -1

Ο πίνακας δυναμικού προγραμματισμού θα έχει μέγεθος (m+1) x (n+1) = (10+1) x (11+1) = 11 x 12 όπου m το μήκος της ακολουθίας v και n το μήκος της ακολουθίας w.

Υπολογίσαμε τις τιμές των κελίων σύμφωνα με τον παρακάτω τύπο που βρήκαμε μετά από έρευνα στο διαδίκτυο:

D(i, j) = max {

0, // Μηδέν

D(i-1, j-1) + S(v[i], w[j]), // Στοίχιση ή ασυμφωνία

D(i-1, j) - 1, // Στοίχιση με κενό

D(i, j-1) - 1 // Στοίχιση με κενό }

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D(i,j) | - | A | T | A | T | C | G | A | C | G | A |
| - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| T | 0 | 0 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 | 2 | 1 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 2 | 1 | 3 | 2 |
| A | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 4 | 3 | 2 | 4 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 3 | 4 | 3 |
| A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 3 | 5 |
| A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 2 | 4 |
| T | 0 | 0 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 3 |
| T | 0 | 0 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 0 | 2 |

Η τιμή της βέλτιστης τοπικής στοίχισης (Score) είναι το **5**.

Εικόνα που περιέχει κείμενο, γραμματοσειρά, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματαΗ στοίχιση προκύπτει από την σχεδίαση πληροφορίας οπισθοδρόμησης που φαίνεται και στον πίνακα με πράσινο χρώμα και είναι η εξής:

Sequence w:

Sequence v:

**Ερώτημα 6:**

Αλγόριθμος που εντοπίζει επαναλήψεις σε κάθε ακολουθία από μία συλλογή k>2 ακολουθιών όπου η συμβολοσειρά που επαναλαμβάνεται είναι ίδια σε όλες τις ακολουθίες.

Ο αλγόριθμος περιλαμβάνει τα εξής βήματα:

1. Αρχικά, ελέγχουμε αν οι ακολουθίες εισόδου είναι περισσότερες από 2 στο πλήθος.
2. Αν υπάρχουν περισσότερες από δύο ακολουθίες (sequences), τότε ξεκινάμε τη διαδικασία εύρεσης επαναλαμβανόμενων υποσυμβολοσειρών:
   1. Για κάθε ακολουθία (seq) στις sequences:
      1. Υπολογίζουμε το μήκος της ακολουθίας.
      2. Δημιουργούμε ένα κενό σύνολο (substrings) για να αποθηκεύσουμε τις υποσυμβολοσειρές που θα επαναλαμβάνονται.
      3. Επαναληπτικά για όλα τα πιθανά μήκη υποσυμβολοσειρών από 2 έως το μισό του μήκους της ακολουθίας:
         1. Χρησιμοποιούμε ένα παράθυρο μεγέθους length που κυλάει από αριστερά προς τα δεξιά στην ακολουθία.
         2. Παίρνουμε την υποσυμβολοσειρά που αντιστοιχεί στο τρέχον παράθυρο.
         3. Ελέγχουμε αν η υποσυμβολοσειρά εμφανίζεται τουλάχιστον δύο φορές σε όλες τις ακολουθίες.
      4. Αν ισχύει, προσθέτουμε την υποσυμβολοσειρά στο σύνολο substrings.
   2. Ενημερώνουμε το σύνολο επαναλήψεων (repetitions) με όλες τις μοναδικές υποσυμβολοσειρές που βρέθηκαν.
   3. Επιστρέφουμε το σύνολο επαναλήψεων ως μια λίστα με το κατάλληλο μήνυμα «Repeated substrings found».
3. Αν δεν υπάρχουν περισσότερες από δύο ακολουθίες (sequences) επιστρέφουμε κατάλληλο μήνυμα: «Not enough sequences provided.»

1. Επιστρέφουμε το σύνολο επαναλήψεων ως μια λίστα με κατάλληλο μήνυμα «Repeated substrings found: ». Στην περίπτωση αυτή, θα είναι ένα κενό σύνολο.

Το αρχείο που περιέχει τον κώδικα σε Python είναι το 6.py.

***Παρατηρήσεις***

Ο παραπάνω αλγόριθμος που δώσαμε δεν περιλαμβάνει κάποιον περιορισμό στα κενά μεταξύ των επαναλήψεων της συμβολοσειράς σε κάθε ακολουθία.

Αυτό δηλαδή σημαίνει ότι η συμβολοσειρά μπορεί να επαναλαμβάνεται με διαφορετικό πλήθος κενών στην κάθε μία από τις ακολουθίες εισόδου χωρίς να αποτελεί πρόβλημα στην αναζήτηση και τον εντοπισμό της.

Στην περίπτωση όμως, που προσθέταμε περιορισμούς για τα κενά μεταξύ των εμφανίσεων της συμβολοσειράς (όπως για παράδειγμα να υπάρχει συγκεκριμένος αριθμός κενών ανάμεσα στις εμφανίσεις της συμβολοσειράς ή να μην υπήρχαν κενά και οι επαναλήψεις της συμβολοσειράς να ήταν η μία δίπλα στην άλλη) θα υπάρχουν τα εξής προβλήματα:

* Περιορισμένη επεκτασιμότητα: Η επιβολή περιορισμών στα κενά ανάμεσα στις εμφανίσεις της συμβολοσειράς μπορεί να καθιστά δυσκολότερο τον εντοπισμό επαναλήψεων που έχουν διαφορετικό αριθμό κενών ανάμεσά τους, και επομένως, να περιορίζει την ανίχνευση επαναλήψεων γενικότερου μοτίβου.
* Αύξηση πολυπλοκότητας: Η εισαγωγή περιορισμών στα κενά μπορεί να αυξήσει την πολυπλοκότητα του αλγορίθμου. Αναγκαζόμαστε να εξετάζουμε περισσότερες περιπτώσεις και να εφαρμόζουμε περιορισμούς στην αναζήτηση επαναλήψεων, πράγμα που μπορεί να αυξήσει τον αριθμό των υπολογισμών που απαιτούνται και να επηρεάσει την απόδοση του αλγορίθμου.
* Απώλεια γενικότητας: Μπορεί να οδηγήσει σε απώλεια γενικότητας του αλγορίθμου και να γίνει δυσκολότερη η ανίχνευση επαναλήψεων με ποικίλα μοτίβα κενών ανάμεσά τους. Αυτό μπορεί να περιορίσει την ικανότητα του αλγορίθμου να ανιχνεύει επαναλήψεις σε πιο πολύπλοκες και ποικίλες δομές δεδομένων.

Συνολικά, η επιβολή περιορισμών στα κενά μεταξύ των εμφανίσεων της συμβολοσειράς μπορεί να καθιστούσε την αναζήτηση επαναλαμβανόμενης συμβολοσειράς σε δοσμένες ακολουθίες περιορισμένη και περίπλοκη.