Εισαγωγή στη Βιοπληροφορική

2024-25

Project 1

*Εξέταση 26/6/2025 μετά τις 11:00*

|  |  |
| --- | --- |
| Δημητρακόπουλος Χρήστος | Μπερτσεκάς Παρασκευάς - Σωτήριος |
| chris.dimitrako@ac.upatras.gr | up1093445@ac.upatras.gr |
| 1079500 | 1093445 |

Περιεχόμενα

[1. Ερώτημα 1 3](#_Toc201522088)

[1.1 Εργαλεία πρόβλεψης πρωτεϊνικών δομών 3](#_Toc201522089)

[1.1.1 AlphaFold3 3](#_Toc201522090)

[1.1.2 ESMFold2 3](#_Toc201522091)

[1.2 Αναζήτηση BlastP 4](#_Toc201522092)

[1.3 Πολλαπλή στοίχιση 5](#_Toc201522093)

[1.3.1 Default παράμετροι 5](#_Toc201522094)

[1.3.2 Προσαρμοσμένες παράμετροι 6](#_Toc201522095)

[1.3.3 Τελική σύγκριση 6](#_Toc201522096)

[1.3.4 Συμπέρασμα 7](#_Toc201522097)

[2. Ερώτημα 2 7](#_Toc201522098)

[2.1 Έλεγχος σχέσης 8](#_Toc201522099)

[2.2 Περιγραφή αλγορίθμου εύρεσης LCS 8](#_Toc201522100)

[3. Ερώτημα 3 8](#_Toc201522101)

[3.1 Μέγιστη κοινή υποακολουθία SARS-CoV-2 & MERS-CoV 8](#_Toc201522102)

[3.2 Σύγκριση δομής πρωτεϊνών με Dali 9](#_Toc201522103)

[4. Ερώτημα 4.α 10](#_Toc201522104)

[4.1 Γενικευμένο δέντρο επιθεμάτων 10](#_Toc201522105)

[4.1.1 Ανάλυση αλγορίθμου 10](#_Toc201522106)

[4.2 Bounded gap 11](#_Toc201522107)

[4.2.1 Ανάλυση αλγορίθμου 11](#_Toc201522108)

[4.3 k-mers 11](#_Toc201522109)

[4.3.1 Ανάλυση αλγορίθμου 12](#_Toc201522110)

[4.4 Quasiperiodicities 12](#_Toc201522111)

[4.4.1 Ανάλυση αλγορίθμου 12](#_Toc201522112)

[4.5 Συμπέρασμα 12](#_Toc201522113)

[5. Ερώτημα 5 13](#_Toc201522114)

[5.1 Ολική στοίχιση 13](#_Toc201522115)

[5.2 Τοπική στοίχιση 14](#_Toc201522116)

[6. Ερώτημα 6 15](#_Toc201522117)

[6.1 Άσκηση 1 15](#_Toc201522118)

[6.2 Άσκηση 2 15](#_Toc201522119)

# Ερώτημα 1

## Εργαλεία πρόβλεψης πρωτεϊνικών δομών

### AlphaFold3

Το AlphaFold3 είναι εργαλείο που προβλέπει την τρισδιάστατη δομή πρωτεϊνών μόνο με βάση την πρωτοταγή αλληλουχία αμινοξέων. Ανήκει στην κατηγορία των μοντέλων βαθιάς μάθησης. Χρησιμοποιεί εξελιγμένες τεχνικές τεχνητής νοημοσύνης προκειμένου να υπολογίζει με υψηλή ακρίβεια τη χωρική διάταξη μιας πρωτεΐνης. Σε αντίθεση με απλούστερες μεθόδους για πρόβλεψη δομής πρωτεϊνών, το AlphaFold3 μπορεί να προβλέψει τη δομή πολλαπλών τύπων βιομορίων, όπως του DNA, του RNA και των συμπλεγμάτων πρωτεϊνών. Αυτό το καθιστά πολύτιμο για τη μελέτη μοριακών αλληλεπιδράσεων και για την κατανόηση πολύπλοκων βιολογικών μηχανισμών.

Το AlphaFold3 χρησιμοποιείται είτε μέσω της πλατφόρμας alphafoldserver.com είτε μέσω του ColabFold, εργαλείου που βασίζεται στο Google Colab. Ο χρήστης δίνει ως είσοδο μια ή περισσότερες αλληλουχίες σε μορφή FASTA και λαμβάνει ως έξοδο μοντέλα δομής σε μορφή Protein Data Bank (PDB), καθώς και ενδείξεις αξιοπιστίας, όπως το pLDDT score. Για βασική web χρήση δεν απαιτούνται ειδικές γνώσεις ή μεγάλη εμπειρία: αρκεί ο χρήστης να γνωρίζει ως ένα βαθμό το προς μελέτη αντικείμενο.

Το AlphaFold3 είναι κατάλληλο για έρευνα όπου η *ακρίβεια* και η *λεπτομέρεια* είναι κρίσιμες, όπως στον σχεδιασμό φαρμάκων ή στη μελέτη λειτουργικών περιοχών σε πρωτεΐνες.

### ESMFold2

Το ESMFold2 είναι κι αυτό ένα μοντέλο πρόβλεψης πρωτεϊνικών δομών, το οποίο ανέπτυξε η Meta AI. Βασίζεται σε transformer αρχιτεκτονική και αξιοποιεί τεχνικές self-suprvised learning για να προβλέψει τη δομή μιας πρωτεΐνης απευθείας από την αλληλουχία, χωρίς να χρησιμοποιεί MSA (multiple sequence alignment) ή templates. Το βασικότερο πλεονέκτημα έναντι άλλων εργαλείων είναι η ταχύτητα, αφού δεν χρησιμοποιεί τη χρονοβόρα διαδικασία της πολλαπλής στοίχισης. Αυτό καθιστά το ESMFold2 κατάλληλο για μαζική ανάλυση πρωτεϊνών, όπως σε μεταγονιδιωματικά δεδομένα, όπου πρέπει να αναλυθούν χιλιάδες (ή και ακόμα περισσότερες) ακολουθίες χωρίς ομόλογα.

Το ESMFold2 είναι διαθέσιμο στο GitHub και μπορεί να εγκατασταθεί τοπικά, ενώ παράλληλα υπάρχει και online demo λειτουργίας στα πλαίσια του ESM Metagenomic Atlas. Η εγκατάστασή του προϋποθέτει μερικές βασικές γνώσεις Python και Linux.

Συνοπτικά, είναι ένα μοντέλο που εστιάζει στην ταχύτητα, σε μαζικές αναλύσεις πολλών πρωτεϊνών, με σκοπό τη γενίκευση της έρευνας. Δεν εστιάζει στη λεπτομερή μοντελοποίηση κάθε πρωτεΐνης ξεχωριστά, αλλά λειτουργεί ως εργαλείο αρχικής διαλογής ώστε να εντοπιστούν αυτές που αξίζουν περαιτέρω ανάλυση και σε επόμενο βήμα να αναλυθούν με πιο ακριβή μοντέλα, όπως το AlphaFold3.

## Αναζήτηση BlastP[[1]](#footnote-1)

A screenshot of a computer program

AI-generated content may be incorrect.

Εικόνα : Αναζήτηση

A screenshot of a computer

AI-generated content may be incorrect.

Εικόνα : Αποτελέσματα

## Πολλαπλή στοίχιση

Ένας από τους βασικότερους τρόπους αξιολόγησης του αποτελέσματος μιας πολλαπλής στοίχισης είναι η ανάλυση των συντηρημένων περιοχών. Αυτές οι περιοχές, σε αντίθεση με τις N-terminal και C-terminal περιοχές, οι οποίες εμφανίζουν διαφοροποιήσεις, είναι κατά βάση ίδιες, αφού αντιπροσωπεύουν δομικές και λειτουργικές περιοχές της πρωτεΐνης. Στο παράδειγμα της άσκησής μας, επιλέξαμε 10 συγγενείς πρωτεΐνες για στοίχιση, αναμένοντας έτσι εκτενείς συντηρημένες περιοχές. Άρα, ως πρώτο κριτήριο ορίζουμε το κατά πόσο κάθε εργαλείο μπορεί να διατηρήσει αυτές τις περιοχές ευθυγραμμισμένες με όσο το δυνατόν λιγότερα κενά. Τα εργαλεία από το EBI μας δίνουν τη δυνατότητα να εμφανίσουμε τα αποτελέσματα μέσω του MView, το οποίο μας δίνει αυτό το ποσοστό «υψηλής συντήρησης» ως consensus. Επιπλέον, μας δίνει το pid, το οποίο δείχνει ποσοστό ομοιότητας με τη reference αλληλουχία. Αυτό σημαίνει ότι, όσο πιο υψηλό είναι, τόσο πιο εύκολη και πιο αξιόπιστη θα είναι η στοίχιση.

### Default παράμετροι[[2]](#footnote-2)

Με βάση αυτά τα κριτήρια, καταλαβαίνουμε ότι και στις 3 περιπτώσεις εργαλείων τα αποτελέσματα θα είναι αρκετά όμοια, όμως μπορούμε να δούμε ότι υπάρχουν μικρές διαφορές στην τοποθέτηση κενών αλλά και στη διατήρηση των συντηρημένων περιοχών. Πιο συγκεκριμένα:

* MAFFT: Η στοίχιση που δίνει έχει τη μεγαλύτερη συνοχή στις συντηρημένες περιοχές, με ελάχιστα κενά και μεγάλα blocks υψηλού consensus.
* MUSCLE: Ενώ συνολικά δίνει καλή στοίχιση, εισάγει περισσότερα κενά σε ενδιάμεσες περιοχές, κάτι που οδηγεί σε πιο σύνθετη στοίχιση σε σχέση με το MAFFT.
* T-Coffee: Παρόλο που είναι το πιο ευαίσθητο από τα 3 εργαλεία, εισάγει περισσότερα κενά ακόμα και εντός περιοχών υψηλής συντήρησης, κάτι που υποδηλώνει ότι ακόμα και σε σημεία υψηλής ομοιότητας η στοίχιση δεν θα είναι καλή.

Τέλος, παρατηρώντας το pid επιβεβαιώνεται ότι οι περισσότερες αλληλουχίες έχουν υψηλή ομοιότητα με τη reference αλληλουχία (πάνω από 90%). Αυτό σημαίνει ότι οι μικρές αποκλείσεις των εργαλείων δεν επηρεάζουν ουσιαστικά τη βιολογική αξιοπιστία.

### Προσαρμοσμένες παράμετροι

Οι παράμετροι–ρυθμίσεις που παρέχονται αφορούν κυρίως το MAFFT, μιας και αυτό είναι το μόνο στο οποίο μπορούμε να αλλάξουμε gap penalty, max iterations κ.ά. Γι’ αυτό θα ασχοληθούμε μόνο με την παραμετροποίηση του MAFFT, και συγκεκριμένα θα θέσουμε gap open penalty 1.8, gap extension penalty 0.2, max iterate 100 και θα ενεργοποιήσουμε FFT σε local pair.

Με αυτές τις παραμέτρους μπορούμε να δούμε αισθητή βελτίωση στην ποιότητα της στοίχισης. Οι συντηρημένες περιοχές είναι μεγαλύτερες και παραμένουν ευθυγραμμισμένες χωρίς προσθήκη κενών.

### Τελική σύγκριση

* MAFFT (default)

Η στοίχιση είναι σε γενικές γραμμές καλή, με ικανοποιητική διατήρηση των συντηρημένων περιοχών και με λίγα μόνο κενά. Ωστόσο υπάρχουν ενδιάμεσες περιοχές όπου εισάγονται κενά τα οποία διασπούν μικρά blocks με υψηλό consensus, κάτι που δείχνει ελαττωμένη ακρίβεια στη στοίχιση.

* MAFFT (modified)

Με την ενεργοποίηση του FFT σε local pair και όλες τις υπόλοιπες αλλαγές που αναφέραμε προηγουμένως, βλέπουμε ως αποτέλεσμα τη βέλτιστη δυνατή στοίχιση. Οι συντηρημένες περιοχές είναι πλήρως στοιχισμένες, με ελάχιστα κενά, και τα consensus blocks είναι μεγαλύτερα και συνεχή.

* MUSCLE (default)

Η στοίχιση είναι αποδεκτή με υψηλά pid και καλό επίπεδο στοίχισης, όμως εισάγονται περισσότερα κενά σε ενδιάμεσες περιοχές σε σχέση με το MAFFT (και στις 2 περιπτώσεις του). Μερικά consensus blocks διασπώνται χωρίς προφανή λόγο, με αποτέλεσμα η ποιότητα της στοίχισης να πέφτει.

* T-Coffee (default)

Είναι το πιο επιθετικό στην εισαγωγή κενών. Ακόμα και σε περιοχές «υψηλής συντήρησης» εμφανίζονται περιττά κενά τα οποία μειώνουν τη στοίχιση δομικά κρίσιμων περιοχών. Η συνολική του ποιότητα είναι μειωμένη, κάτι που δεχνει πως δεν είναι ιδανικό εργαλείο για περιπτώσεις στοίχισης με υψηλή ομοιότητα, όπως έχουμε εν προκειμένω.

### Συμπέρασμα

Η modified στοίχιση του MAFFT παρέχει την πιο συνεπή και ακριβή στοίχιση. Η χρήση των default παραμέτρων είναι επαρκής για μια αρχική αξιολόγηση των εργαλείων, αλλά ανάλογα με το ζητούμενο και τις αλληλουχίες ο MAFFT μπορεί να αποδώσει καλύτερα.

# Ερώτημα 2

Το πρόβλημα της μέγιστης κοινής υπακολουθίας (LCS) μεταξύ δύο συμβολοσειρών S1 και S2 μπορεί όντως να θεωρηθεί ειδική περίπτωση του προβλήματος της εύρεσης της ελάχιστης απόστασης μετασχηματισμού (edit distance) με δοσμένα βάρη. Το edit distance D(n, m) είναι το ελάχιστον συνολικό κόστος για να μετατραπεί η S1 στην S2 επιτρέποντας οποιονδήποτε συνδυασμό πράξεων από τις εξής:

|  |  |  |
| --- | --- | --- |
| **Πράξη** |  | **Κόστος** |
| Ένθεση/διαγραφή | d | 1 |
| Αντικατάσταση | r | 2 |
| Ταίριασμα | e | 0 |

Η μέγιστη κοινή υπακολουθία είναι η μεγαλύτερη ακολουθία χαρακτήρων που εμφανίζεται και στις δυο συμβολοσειρές, με την ίδια σειρά, αλλά όχι απαραίτητα σε συνεχόμενες θέσεις. Η εκφώνηση ζητάει να ελέγξουμε την ορθότητα της σχέσης όπου u το μήκος της LCS.

ή ισοδύναμα

Η σχέση δείχνει ότι το ελάχιστο κόστος μετασχηματισμού μεταξύ δύο συμβολοσειρών μπορεί να εκφραστεί απευθείας ως συνάρτηση των μηκών τους και του μήκους της LCS. Πρακτικά μας λέει ότι κάθε χαρακτήρας που δεν ανήκει στην LCS πρέπει να διαγραφεί από την S1 ή να εισαχθεί στην S2.

## Έλεγχος σχέσης

Για τα συγκεκριμένα βάρη η παραπάνω σχέση δεν ισχύει πάντα. Συγκεκριμένα, η χρήση αντικατάστασης (με κόστος 2) μπορεί να δώσει εναλλακτικά ελάχιστα μονοπάτια μετασχηματισμού τα οποία δεν συμφωνούν πάντα την σχέση που δίνεται. Αντιπαράδειγμα είναι οι συμβολοσειρές και με και αντίστοιχα. Η μέγιστη κοινή υποακολουθία είναι “AT” άρα . Από τον πίνακα υπολογίζουμε , ενώ η σχέση μας δίνει . Προφανώς, καταλήγουμε σε άτοπο συμπέρασμα, άρα η σχέση είναι ψευδής.

Για να ισχύει αυτή η σχέση με αυτά τα βάρη, θα πρέπει το ελάχιστο κόστος να προκύπτει μόνο με ενθέσεις και διαγραφές, δηλαδή να μη χρειαστεί ποτέ να κάνουμε αντικαταστάσεις.

## Περιγραφή αλγορίθμου εύρεσης LCS

Το συγκεκριμένο ζητούμενο υλοποιήθηκε με τη χρήση python. Πιο συγκεκριμένα, υλοποιήθηκε χρησιμοποιώντας δυναμικό προγραμματισμό σε πίνακα. Αρχικά, αποθηκεύεται το μήκος των συμβολοσειρών και το ελάχιστο κόστος μετατροπής του πρώτου i γράμματος της μίας στο πρώτο j της άλλης. Αν μία συμβολοσειρά είναι κενή, χρειάζεται i διαγραφές ή j εισαγωγές, ενώ ελέγχεται κάθε σενάριο (διαγραφή ή εισαγωγή ή ταίριασμα) για να φτάσουμε στο μικρότερο κόστος, το οποίο εισάγεται σε πίνακα. Η τελική απόσταση του μετασχηματισμού προκύπτει ως το κάτω δεξί στοιχείο του πίνακα που κατασκευάσαμε. Τέλος, για την LCS ο αλγόριθμος ξεκινά από το τέλος του πίνακα και ακολουθεί το μονοπάτι των ελαχίστων κόστους για να ανακατασκευάσει την LCS, αν οι χαρακτήρες ταιριάζουν, προστίθενται στη LCS και στο τέλος γίνεται αναστροφή για να προκύψει η σωστή σειρά. Εν κατακλείδι, επιστρέφονται το συνολικό κόστος μετασχηματισμού, το μήκος της μέγιστης κοινής υποακολουθίας και η ίδια η υποακολουθία (LCS). Μπορείτε να βρείτε τον κώδικα στο [GitHub](https://github.com/xrhstosdim1/ceid_bioInformatics/blob/main/Project%201/exc2/testc.py).

# Ερώτημα 3

## Μέγιστη κοινή υποακολουθία SARS-CoV-2 & MERS-CoV

Όπως εξηγήσαμε νωρίτερα, η μέγιστη κοινή υποακολουθία είναι η μεγαλύτερη ακολουθία χαρακτήρων που εμφανίζεται και στις δυο συμβολοσειρές, με την ίδια σειρά, αλλά όχι απαραίτητα σε συνεχόμενες θέσεις. Αυτό σημαίνει ότι ο αλγόριθμος που παρουσιάσαμε στο προηγούμενο ερώτημα θα πρέπει να έχει προτεραιότητα την ένθεση/διαγραφή και όχι την αντικατάσταση. Άρα θα πρέπει το d να είναι μικρότερο από το r. Με τιμές d = 1, r = 2, e = 0 το αποτέλεσμα είναι:

Edit Distance D(n, m): 1450

LCS length u: 592

LCS: MVFLLLLVSSCVQTFTRPDVSDPSNTIHVSGTGTPLFNDVFANGFGTTSTSATIKPFLGVNSMRVYSNCTFYPDGNFKNLRNFYTNDLQGFSVDLGNFQALYTPSSSAAAYVYLQPTFLLGIADCDLSLSFVEGYSFVQEVFPLPFVFNTFSVNISNCYSLYFSYPLDLSVAGINYKFTCILVNLLKSNKRSTEQASPCVSPTVGYYRLSPLGLGTVTEQQFGITTDVPLEDTSGVSVGNTQVAYQVNCVPVIHALTTRSTGQTGCLGVNSEDPGCASTTPRRSVASIALNSSSIPTNFSVTEITKTVDCYCGCLLYGFCNALGANQVFAVKQPIFGGFNLPSSRSIEDLLFKVTADGYGDCGAARDLICAQGVLPPLDMAYTSLLGSGWTAGAAIPFAQYRNGVTQVLENQKLIANFNAGQTGKQDVNNAQALKLSSNFGAISSDIRLDVEAQIDRLIGRLTTQQLRESALAAKECVQSKRFCGGHSFAPGLHVYPNICDAPVNGFTNFYEPITSNVVTYQPLSFELDFKNVSINSQIDLEVKLNESIDLELGYYKWPWYIWLGFIAGLVVILCCTCCLKCCCDEDEPKVH

## Σύγκριση δομής πρωτεϊνών με Dali

Η spike protein του SARS-CoV-2 αντιστοιχεί στην P0DTC2 του UniProt, από όπου την πήραμε <https://www.uniprot.org/uniprotkb/P0DTC2/entry>. Η εισαγωγή των δύο ακίδων στο swiss model δίνει πολλαπλές απεικονίσεις και για τις δύο ακίδες, γι’ αυτό διαλέξαμε αυτές με το μεγαλύτερο sequence identity. Τα αρχεία .pdb θα τα βρείτε στο GitHub [sarscov2-model\_05.pdb](https://github.com/xrhstosdim1/ceid_bioInformatics/blob/main/Project%201/exc3/sarscov2-model_05.pdb) (100% sequence coverage) και [merscov-model\_03.pdb](https://github.com/xrhstosdim1/ceid_bioInformatics/blob/main/Project%201/exc3/merscov-model_03.pdb) (99.69% sequence coverage).

Το Dali είναι εργαλείο σύγκρισης πρωτεϊνικών δομών. Παρέχει μετρικές, με σημαντικότερες τις εξής: Z-score (μετράει την αντίστροφη πιθανότητα η ομοιότητα της δομής να προκύψει τυχαία, άρα, όσο μεγαλύτερο αποτέλεσμα, τόσο λιγότερη «τυχαιότητα»), RMSD και lali (length of alignment). Τα αποτελέσματα αυτών των μετρικών είναι:

|  |  |  |
| --- | --- | --- |
| Z-score | RMSD | lali |
| 25.6 | 5.5 | 751 |

Παρά τη μέτρια ομοιότητα μεταξύ των αλληλουχιών (RMSD = 5.5), οι δύο πρωτεΐνες διατηρούν πολύ παρόμοια τρισδιάστατη δομή (Z-score = 25.6), κάτι που υποδηλώνει εξελικτική συγγένεια και πιθανώς διατήρηση βασικών λειτουργικών ρόλων. Επίσης, ένα σημαντικό κριτήριο είναι η στοίχιση από τα 1207 αμινοξέα, τα 751 στοιχήθηκαν, που σημαίνει περίπου το 60%, το οποίο είναι ένα πολύ μεγάλο τμήμα των πρωτεϊνών. Αναλυτικά τα [αποτελέσματα του Dali](https://github.com/xrhstosdim1/ceid_bioInformatics/blob/main/Project%201/exc3/dali-res.pdf).

# Ερώτημα 4.α

Το πρόβλημα που καλούμαστε να λύσουμε είναι η εύρεση συμβολοσειρών που εμφανίζονται τουλάχιστον δύο φορές σε κάθε ακολουθία ενός συνόλου k ακολουθιών (k>2), με επιπλέον απαίτηση η συμβολοσειρά να είναι ίδια σε όλες τις ακολουθίες. Το πρόβλημα δυσκολεύει όταν υπάρχει περιορισμός στο μέγιστο επιτρεπτό κενό (gap) ανάμεσα στις εμφανίσεις της επανάληψης κάθε ακολουθίας. Παρακάτω αναλύονται οι κύριες αλγοριθμικές τεχνικές μαζί με ψευδοκώδικα του βασικού αλγορίθμου και οι δυσκολίες από τους περιορισμούς στα gaps. Τα παρακάτω βασίζονται στη βιβλιογραφία που δίνεται στην εκφώνηση.

## Γενικευμένο δέντρο επιθεμάτων

Η εν συνόλω καλύτερη αντιμετώπιση του προβλήματος βασίζεται στη δομή του γενικευμένου δέντρου επιθεμάτων (GST), δηλαδή της επέκτασης του απλού suffix tree για πολλά strings. Για την ακρίβεια, το GST κατασκευάζεται από όλα τα επιθέματα όλων των ακολουθιών: κάθε φύλλο σημειώνεται με το αναγνωριστικό της ακολουθίας και με τη θέση έναρξης του επιθέματος, και κάθε εσωτερικός κόμβος του GST αντιστοιχεί σε μια υποσυμβολοσειρά.

Η κεντρική ιδέα είναι ότι κάθε εσωτερικός κόμβος τού GST αντιστοιχεί σε μια υποσυμβολοσειρά, και το υποδέντρο του περιέχει όλες τις εμφανίσεις αυτής της υποσυμβολοσειράς, με φύλλα που ανήκουν σε διαφορετικές ακολουθίες.

### Ανάλυση αλγορίθμου

* Βήμα 1: Κατασκευή GST

Ενώνουμε όλες τις ακολουθίες μεταξύ τους, διαχωρίζοντάς τες με χαρακτήρα εκτός του αλφαβήτου των ακολουθιών. Κάθε φύλλο του δέντρου αντιπροσωπεύει ένα επίθεμα και φέρει πληροφορία για το ποια ακολουθία ανήκει σε ποια θέση.

* Βήμα 2

Αφού κατασκευάσουμε το GST, το διατρέχουμε postorder. Σε κάθε εσωτερικό κόμβο του δέντρου παίρνουμε πληροφορίες για το πόσες φορές (και σε ποιες θέσεις) εμφανίζεται η υποσυμβολοσειρά που αντιστοιχεί σε αυτόν τον κόμβο σε κάθε ακολουθία. Γι’ αυτό δημιουργούμε έναν πίνακα k θέσεων, που λειτουργεί ως counter, σε κάθε κόμβο.

* Βήμα 3: Εντοπισμός επαναλήψεων

Αφού συμπληρωθούν όλοι οι πίνακες (άρα έχουμε διατρέξει ολόκληρο το GST), για να ελέγξουμε ποιες υποσυμβολοσειρές εμφανίζονται τουλάχιστον δύο φορές αρκεί να αναζητήσουμε στους πίνακες αποτέλεσμα μεγαλύτερο ή ίσο του δύο.

## Bounded gap

Η μέθοδος του Bounded gap πρακτικά είναι επέκταση του GST, το οποίο εξηγήσαμε παραπάνω. Το σκεπτικό είναι ίδιο, με μια βασική διαφορά: εκτός από τις επαναλήψεις των υποσυμβολοσειρών αποθηκεύεται και η θέση τους. Αυτό μας επιτρέπει να αναζητήσουμε επαναλήψεις εντός μιας απόστασης G που θέτουμε.

### Ανάλυση αλγορίθμου

* Βήμα 1: Κατασκευή GST

Όπως προηγουμένως, κατασκευάζουμε το GST για όλες τις ακολουθίες.

* Βήμα 2: Έλεγχος για bounded gap σε κάθε κόμβο

Διατρέχουμε το GST, όπως και πριν. Όμως, για να αποθηκεύσουμε επανάληψη στους πίνακες λύσεων, ελέγχουμε αν το κριτήριο απόστασης ικανοποιείται, δηλαδή αν υπάρχουν τουλάχιστον δύο εμφανίσεις της συμβολοσειράς με διαφορά θέσεων μικρότερη ή ίση από το G.

* Βήμα 3: Εντοπισμός επαναλήψεων

Όπως και πριν, κοιτώντας τον πίνακα έχουμε τα αποτελέσματα.

## k-mers

Μια πιο απλή και πιο συχνά χρησιμοποιούμενη τεχνική για τον εντοπισμό επαναλήψεων είναι αυτή των k-mers. Τα k-mers είναι όλες οι υποσυμβολοσειρές σταθερού μήκους k που μπορούν να εξαχθούν από κάθε ακολουθία. Για παράδειγμα, η συμβολοσειρά ACGTA έχει k-mers για k=3 τα “ACG”, “CGT”, “GTA”. Το βασικό μειονέκτημα είναι ότι θυσιάζεται η γενικότητα της αναζήτησης προς όφελος του χρόνου.

### Ανάλυση αλγορίθμου

* Βήμα 1

Για κάθε ακολουθία εξάγουμε όλα τα δυνατά k-mers.

* Βήμα 2

Μετράμε πόσες φορές εμφανίζεται κάθε k-mer σε κάθε ακολουθία, χρησιμοποιώντας πίνακα.

* Βήμα 3

Όσα k-mers έχουν counter μεγαλύτερο ή ίσο του δύο στον πίνακα αποτελεσμάτων είναι τα «έγκυρα» αποτελέσματα.

## Quasiperiodicities

Μια σημαντική επέκταση του προβλήματος με το οποίο ασχολούμαστε είναι ο εντοπισμός των quasiperiodicities, δηλαδή μοτίβων που εμφανίζονται περιοδικά μέσα σε μια ακολουθία, με επικαλύψεις ή μικρές διαφορές μεταξύ τους. Το άρθρο των Brodal και Pedersen που δίνεται στην εκφώνηση προτείνει έναν αλγόριθμο που εντοπίζει όλα τα μέγιστα αυτά μοτίβα (maximal quasiperiodicities) σε χρόνο O(n), χρησιμοποιώντας δέντρο επιθεμάτων.

### Ανάλυση αλγορίθμου

* Κάθε μοτίβο ορίζεται μέσω επαναλήψεων με επικαλύψεις ή gaps εντός ορισμένου εύρους
* Κατά την κατασκευή του suffix tree εντοπίζονται όλα τα πιθανά σημεία για επικάλυψη, είτε με exact match είτε με «quasi», δηλαδή ελαφρώς μετατοπισμένα matches.
* Ο αλγόριθμος επιστρέφει όλα τα μέγιστα μοτίβα από αυτά διασφαλίζοντας ότι δεν παραλείπεται κανένα μεγαλύτερο μοτίβο με ίδιο property.

## Συμπέρασμα

Για την ανίχνευση επαναλήψεων σε ένα σύνολο k ακολουθιών, ο πιο αποδοτικός τρόπος είναι η κατασκευή γενικευμένου δέντρου επιθεμάτων, το οποίο επιτρέπει τον εντοπισμό συμβολοσειρών που εμφανίζονται τουλάχιστον δύο φορές σε κάθε ακολουθία και είναι ίδιες σε όλες. Σε πιο πρακτικές, μη θεωρητικές περιπτώσεις χρησιμοποιείται ο περιορισμός του bounded gap, μιας και προσφέρει δυνατότητα ομαδοποίησης εντός προκαθορισμένου διαστήματος, καταγράφοντας τις θέσεις εμφάνισης. Η ανάλυση με k-mers (υποσυμβολοσειρές σταθερού μήκους) είναι ταχύτερη από τις άλλες, όμως δεν καλύπτει το γενικό πρόβλημα, αφού περιορίζει την ανίχνευση σε συγκεκριμένου μήκους συμβολοσειρές. Τέλος, το ζήτημα των quasiperiodicites επεκτείνει αυτό το πρόβλημα πέραν των πανομοιότυπων επαναλήψεων και εστιάζει στην εύρεση μοτίβων που επαναλαμβάνονται περιοδικά.

# Ερώτημα 5

Καθαρά για εμφανισιακούς λόγους χρησιμοποιήθηκαν διαθέσιμα online εργαλεία για την οπτικοποίηση των δύο πινάκων που ζητούνται. Για την ολική στοίχιση χρησιμοποιήθηκε το [Global Alignment App](https://bioboot.github.io/bimm143_W20/class-material/nw/) και για την τοπική στοίχιση ένα [εργαλείο εκπαίδευσης του αλγορίθμου Smith-Waterman](https://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman).

## Ολική στοίχιση

Στον παρακάτω πίνακα φαίνεται το αποτέλεσμα της ολικής στοίχισης. Η πρώτη γραμμή και στήλη του πίνακα αρχικοποιούνται με τιμές κόστους στοίχισης με κενό, υποθέτοντας ότι μέχρι εκείνο το σημείο προσθέτουμε διαρκώς κενά. Για κάθε επόμενο κελί υπολογίζονται οι τιμές για στοίχιση, για ασυμφωνία και για στοίχιση με κενό, και αποθηκεύεται η μέγιστη από αυτές. Κάθε φορά που συμπληρώνεται μια τιμή, σημειώνεται και το κελί προέλευσης, ώστε αργότερα να εκτελεστεί το traceback (κόκκινο μονοπάτι), το οποίο ξεκινάει από το τελευταίο κελί του πίνακα (κάτω δεξιά) μέχρι να φτάσει στην αρχή του πίνακα (πάνω αριστερά). Το μονοπάτι που θα δημιουργηθεί είναι η βέλτιστη στοίχιση.

A screenshot of a game

AI-generated content may be incorrect.

Εικόνα : Πίνακας δυναμικού προγραμματισμού ολικής στοίχισης

## Τοπική στοίχιση

Στον παρακάτω πίνακα φαίνεται το αποτέλεσμα της τοπικής στοίχισης. Η βασική διαφορά της τοπικής στοίχισης από την ολική είναι ότι δεν προσπαθεί να ταιριάξει ολόκληρες τις ακολουθίες αλλά μόνο το ή τα κομμάτια που ταιριάζουν καλύτερα. Πρακτικά αυτό σημαίνει ότι μας ενδιαφέρουν κομμάτια της ακολουθίας, όχι ολόκληρη η ακολουθία. Για αυτόν τον λόγο η πρώτη γραμμή και η πρώτη στήλη του πίνακα αρχικοποιούνται με 0, και επιτρέπεται να εισαχθεί μηδενική τιμή στα ενδιάμεσα κελιά. Όπως και στην ολική στοίχιση, κάθε κελί υπολογίζεται με βάση τις πιθανές κινήσεις, όμως το Traceback ξεκινά από τη μεγαλύτερη τιμή του πίνακα και τελειώνει μόλις συναντήσει μηδενική τιμή. Ακριβώς για αυτόν τον λόγο μπορεί να υπάρχουν πολλές βέλτιστες εναλλακτικές στοιχίσεις, όπως φαίνεται δεξιά του πίνακα.

A screenshot of a computer

AI-generated content may be incorrect.

Εικόνα : Πίνακας δυναμικού προγραμματισμού τοπικής στοίχισης

# Ερώτημα 6

## Άσκηση 1

Επιλέξαμε την κυτοχρωμική πρωτεΐνη C (Cytochrome C), που αντιστοιχεί στον UniProt κωδικό P99999, για να βρούμε τις 8 ομόλογες ακολουθίες που ζητούνται. Αναλυτικότερα, αφού εισαγάγουμε το αρχείο της πρωτεΐνης σε μορφή FASTA, φτιάχνουμε query αναζήτησης με BLASTp με hitlist size = 8 ώστε να πάρουμε 8 αποτελέσματα. Αφού πάρουμε τις 8 ομόλογες ακολουθίες τις αποθηκεύουμε μαζί με την ανθρώπινη κυτοχρωμική C σε ένα αρχείο FASTA. Στη συνέχεια εκτελείται πολλαπλή στοίχιση με το Clustal Omega και αποθηκεύεται σε αρχείο aln5. Τέλος, ο κώδικας πραγματοποιεί ζευγαρωτές στοιχίσεις για κάθε δυνατή δυάδα ακολουθιών και δημιουργεί ξεχωριστά αρχεία aln5 για καθεμία, που περιλαμβάνουν τη στοίχιση και το score. Ο κώδικας και τα εξαγόμενα αποτελέσματα βρίσκονται στο [GitHub](https://github.com/xrhstosdim1/ceid_bioInformatics/tree/main/Project%201/exc6).

## Άσκηση 2

Ο κώδικας που γράφτηκε για το ερώτημα είναι απλός στη δομή του. Αρχικά, εισάγουμε τη mystery\_sequence ως string και την καθαρίζουμε από new lines και κενά, και σβήνουμε το “mystery\_sequence”. Έπειτα δημιουργούμε query για αναζήτηση στην NCBI βάση για αναζήτηση με BLASTp. Αποθηκεύουμε τα αποτελέσματα σε ένα αρχείο XML και έπειτα το φιλτράρουμε με το δεδομένο της εκφώνησης ότι το όνομα του οργανισμού που ψάχνουμε ξεκινάει με «Tyr». Στο παρακάτω screenshot φαίνεται το αποτέλεσμα της αναζήτησης. Στο [GitHub](https://github.com/xrhstosdim1/ceid_bioInformatics/tree/main/Project%201/exc6) μπορείτε να βρείτε ολόκληρο τον κώδικα και το XML της αναζήτησης.

A black screen with white text

AI-generated content may be incorrect.

1. Αναλυτικά τα 10 αποτελέσματα της αναζήτησης θα τα βρείτε στο [GitHub:](https://github.com/xrhstosdim1/ceid_bioInformatics/tree/main/Project%201) [RES\_blastp\_NP-000137s](https://github.com/xrhstosdim1/ceid_bioInformatics/blob/main/Project%201/exc1/RES_blastp_NP-000137s.txt) [↑](#footnote-ref-1)
2. Αναλυτικά τα [αποτελέσματα της στοίχισης](https://github.com/xrhstosdim1/ceid_bioInformatics/tree/main/Project%201/exc1) και με τα 3 εργαλεία θα τα βρείτε στο GitHub. [↑](#footnote-ref-2)