



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ: ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ

Επί των φοιτητών: Μόσχου Δημήτριου - Π18209
Σιάτρα Απόστολου – Π18215

Διδάσκων: Πικράκης Άγγελος

Πειραιάς, 2021



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ: ΒΙΟΠΛΗΡΟΦΟΡΙΚΗ

Τελική Εργασία Μαθήματος

Επί των φοιτητών: Μόσχου Δημήτριου - Π18209
Σιάτρα Απόστολου – Π18215

Διδάσκων: Πικράκης Άγγελος

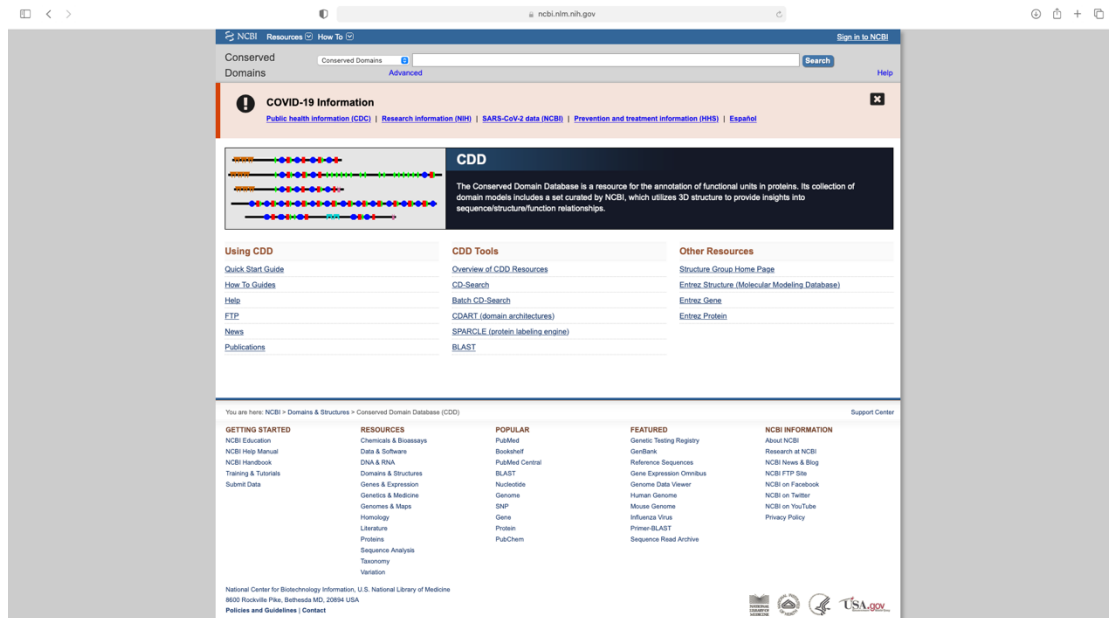
Πειραιάς, 2021

Πίνακας περιεχομένων

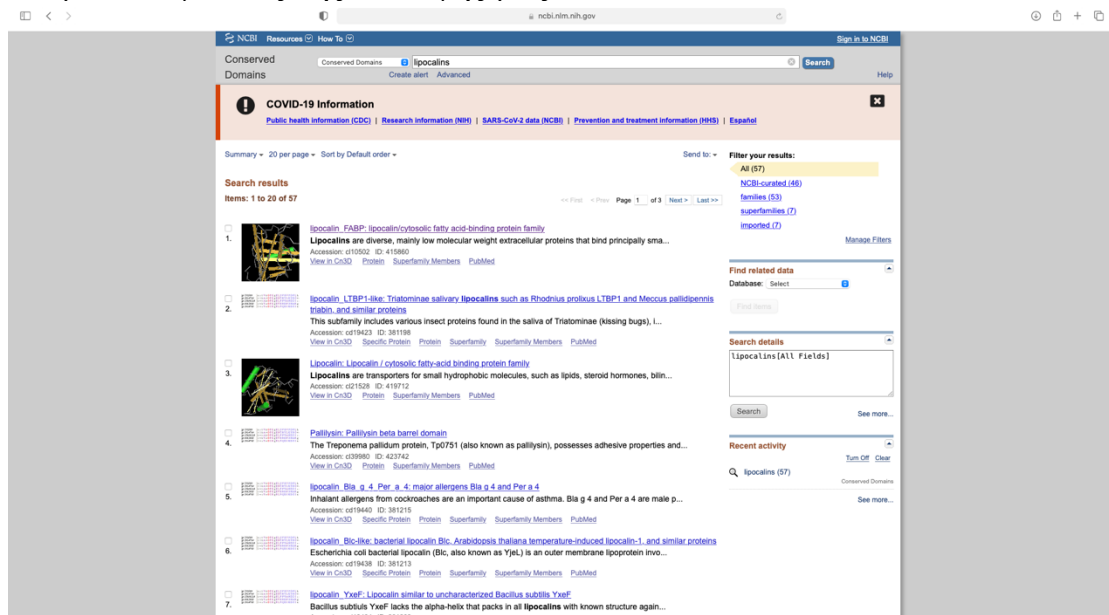
- 1. ΘΕΜΑ 1ο**
- 2. ΘΕΜΑ 2ο**
- 3. ΘΕΜΑ 3ο**
- 4. ΘΕΜΑ 4ο**
 - 4.1 Ερώτημα 1^ο**
 - 4.2 Ερώτημα 2^ο**
 - 4.3 Ερώτημα 3^ο**
 - 4.4 Ερώτημα 4^ο**
 - 4.5 Ερώτημα 5^ο**

1. ΘΕΜΑ 1^ο

Αρχικώς, μεταβαίνουμε στη βάση δεδομένων των συντηρημένων δοκιμών επικρατειών στο NCBI



Στη συνέχεια, εισάγουμε τον όρο λιποκαλίνες (lipocalins), ή κάποιο άλλο όνομα οικογένειας της επιλογής μας.



Έπειτα, αφότου κάνουμε την επιλογή μας, θα επιλέξουμε το mFasta ως μορφή αρχείου και στην συνέχεια θα εκτελέσουμε το Reformat. Το αποτέλεσμα που θα εμφανιστεί θα είναι μία πολλαπλή στοίχιση αλληλουχιών.

```

>gi|609411913|pdb|3WJ8|B
--mwshpqfknq1qqlnpgesspvphfVAPLSYLLGTWRGQGEYEPTIPFSRYGEEIRFSH-SGKPVIAYTQKTWKL
ESgA-PLLAESGYFRprPDGS--IEVVIACSTGLVEVQKGTYNv--dEqsIKLKS---DLV---GNASKVKEISREFELV
DGK--LSYVVRLSTTTNPLQPLKAILDkl-----
>gi|504799881|ref|WP_014986983.1|
-----mvesqppiaphpdIAPLAALLGTWRGNGHGEYPTIQPFDYLEEVRFGH-LGRPFLTYRQRTAA
DD-GrPMHAETGYLR--CPRPdrVELILAHPTGITEICEGALTvdgAlhLEFDS---TSIgrsSTAKLVTALGRTFQV-
KGDt--IDYTVRMAAVGEPLQHHLAATLIrae-----
>gi|218551765|sp|A1T297.1|Y449_MYCVP
-----madvspalhpdlVAALAPLLGTWVGEYSGEYPTIEPFYTEEITFGH-VGKPFLTYAQRTAA
DD-GrPLHAETGYLR--ASAPdrIEWILAHPTGITEIQEQQLTadgdGlrMELVS---SSIgrsGSAKEVTDVGRSIEL-
RGDt--LTYTLRMAAVGQPLQHHLSAVLRrvr-----
>gi|333487064|gb|AEF36456.1|
-----mELAPLLGTWSGRGRGVYPTIASFDYLEEVTFSH-VGKPLVYVGQKTKSA
AD-GIPLHAETGYLR--VPQPgrIEWVLHAPSGITEIEVGSYRvtadGieLEMSA---PTIglPTAKEVTALSRRYRL-
ARDe--LSYTLDMGAVGEPQNHILTAALRrtg-----
>gi|218551734|sp|B2HLY1.1|Y1995_MYCM
-----mpadlhpdlDALAPLLGTWAGQGSGEYPTIEPFYLEEVRFSH-VGKPFLVYAQKTRAV
AD-GaPLHAETGYLR--VPKPgqVELVLHAPSGITEIEVGTYSasggVieMEMVT---TAIgmtPTAKEVTALSRSFRM-
VGDe--LSYRLRMGAVGLPLQHHLGARLRrks-----
>gi|81413567|sp|Q73W27.1|Y2833_MYCPA
-----mptdlhpdlAALAPLLGTWGRSGKYPTIQPFDYLEEVTFSH-VGKPFLAYAQKTRAA
AD-GkPLHAETGYLR--VPQPgrLELVLHAPSGITEIEVGSYAvtgGlieMRMST---TSIglTPSAKEVTALARWFI-
DGDe--LSYSVQMGAVGQPLQDHLAAVLRhr-----
>gi|88193107|pdb|2FR2|A
-----mtrdlapalQALSPLLSWAGRGAGKYPTIRPFYLEEVRFAH-VGKPFLTYTQOTRAV
AD-GkPLHSETGYLR--VCRPgcVELVLHAPSGITEIEVGTYSVtgdvieLELSTradGSIglPTAKEVTALDRSYRI-
DGDe--LSYSLQMRVAVGQPLQDHLAAVLRhrqrshhhhh
>gi|81537071|sp|Q9CCB8.1|Y1006_MYCLE
-----mpsdlcpdlQALAPLLGSWVGRGMGYPTIQPFYLEEVRFSH-LDRPFLTYTQKTRAI
TD-GkPLHAETGYLR--VPQPghIELVLAHHSIDIAIEVGTYSVtgdlieVELVT---TTIglvPTAKQVTALGRFFRI-
DGDe--FAYSVMGAVGQPLQDHLVAVLRhrq-----
>gi|256008631|gb|ACU54198.1|
-----mtireMLEGTWTGSGIGSYPEVAEFSYQERLRFES-NGRPFLRMEQHT-TG
PD-GsPLHTEVGYLR--FVANgrLELVVAQPTGIVEVLEGGAEaeggvISLTS---LVVattPSAKRVDAVQRRFTL-
RGDe--LMTLEWMDAMERGMMQHLESRLRrg-----
>gi|81750622|sp|Q8FTT5|Q8FTT5_COREF
mgarvermIfdarsapilprstgmdlhpnlAPYVGLTGTWTKGKHGFYPTIEDFSYEETLNFSTIPGKPFPRYEQKTMGL
Dg---PLHTELGFLR--ILDDgrAEFILAQPMGQTELEGMVSeegTlvFDFTF---STVansGSAKRVGTARTYTF-
TADrtgLSAQFAMGAVGQPLQQHLESELTkqs-----

```

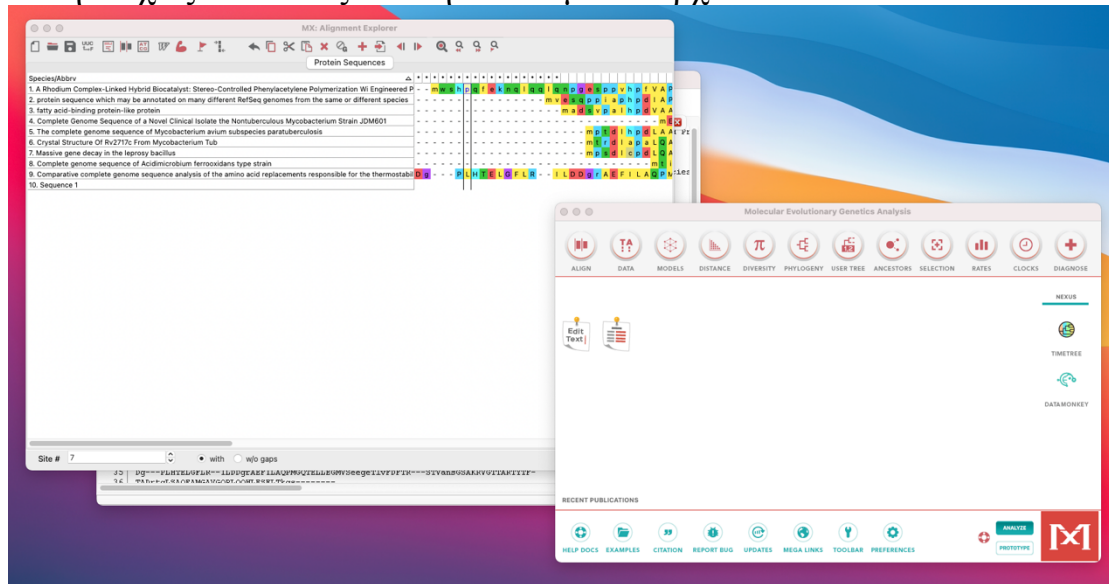
Μετά, θα αντιγράψουμε το αποτέλεσμα αυτό σε έναν text editor της επιλογής μας (στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε το VS Code και το TextEdit της Apple), έτσι ώστε να απλοποιήσουμε τα ονόματα των αλληλουχιών.

```

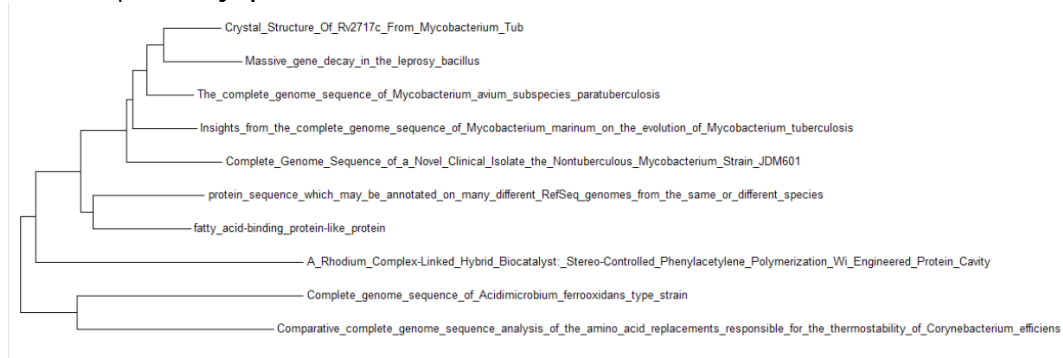
1  2  4  6  8  10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
>A Rhodium Complex-Linked Hybrid Biocatalyst: Stereo-Controlled Phenylacetylene Polymerization Wi Engineered Protein Cavity
--mwshpqfknq1qqlnpgesspvphfVAPLSYLLGTWRGQGEYEPTIPFSRYGEEIRFSH-SGKPVIAYTQKTWKL
ESgA-PLLAESGYFRprPDGS--IEVVIACSTGLVEVQKGTYNv--dEqsIKLKS---DLV---GNASKVKEISREFELV
DGK--LSYVVRLSTTTNPLQPLKAILDkl-----
> protein sequence which may be annotated on many different RefSeq genomes from the same, or different, species
-----mvesqppiaphpdIAPLAALLGTWRGNGHGEYPTIQPFDYLEEVRFGH-LGRPFLTYRQRTAA
DD-GrPMHAETGYLR--CPRPdrVELILAHPTGITEICEGALTvdgAlhLEFDS---TSIgrsSTAKLVTALGRTFQV-
KGDt--IDYTVRMAAVGEPLQHHLAATLIrae-----
>fatty acid-binding protein-like protein
-----madvspalhpdlVAALAPLLGTWVGEYSGEYPTIEPFYTEEITFGH-VGKPFLTYAQRTAA
DD-GrPLHAETGYLR--ASAPdrIEWILAHPTGITEIQEQQLTadgdGlrMELVS---SSIgrsGSAKEVTDVGRSIEL-
RGDt--LTYTLRMAAVGQPLQHHLSAVLRrvr-----
>Complete Genome Sequence of a Novel Clinical Isolate, the Nontuberculous Mycobacterium Strain JDM601
-----mELAPLLGTWSGRGRGVYPTIASFDYLEEVTFSH-VGKPLVYVGQKTKSA
AD-GIPLHAETGYLR--VPQPgrIEWVLHAPSGITEIEVGSYRvtadGieLEMSA---PTIglPTAKEVTALSRRYRL-
ARDe--LSYTLDMGAVGEPQNHILTAALRrtg-----
>Insights from the complete genome sequence of Mycobacterium marinum on the evolution of Mycobacterium tuberculosis
-----mpadlhpdlDALAPLLGTWAGQGSGEYPTIEPFYLEEVRFSH-
VGKPLVYAQKTRAV AD-GaPLHAETGYLR--VPKPgqVELVLHAPSGITEIEVGTYSasggVieMEMVT---TAIgmtPTAKEVTALSRSFRM- VGDe--LSYRLRMGAVGLPLQHHLGARLRrks-----
>The complete genome sequence of Mycobacterium avium subspecies paratuberculosis
-----mptdlhpdlAALAPLLGTWGRSGKYPTIQPFDYLEEVTFSH-VGKPFLAYAQKTRAA
AD-GkPLHAETGYLR--VPQPghIELVLHAPSGITEIEVGSYAvtgGlieMRMST---TSIghTPSAKEVTALARWFI-
DGDe--LSYSVQMGAVGQPLQDHLAAVLRhr-----
>Crystal Structure Of Ry2717c From Mycobacterium Tub
-----mtrdlapalQALSPLLSWAGRGAGKYPTIRPFYLEEVRFAH-VGKPFLTYTQOTRAV
AD-GkPLHSETGYLR--VCRPgcVELVLHAPSGITEIEVGTYSVtgdvieLELSTradGSIglPTAKEVTALDRSYRI-
DGDe--LSYSLQMRVAVGQPLQDHLAAVLRhrqrshhhhh
>Massive gene decay in the leprosy bacillus
-----mpsdlcpdlQALAPLLGSWVGRGMGYPTIQPFYLEEVRFSH-LDRPFLTYTQKTRAI
TD-GkPLHAETGYLR--VPQPghIELVLAHHSIDIAIEVGTYSVtgdlieVELVT---TTIglvPTAKQVTALGRFFRI-
DGDe--FAYSVMGAVGQPLQDHLVAVLRhrq-----
>Complete genome sequence of Acidimicrobium ferrooxidans type strain
-----mtireMLEGTWTGSGIGSYPEVAEFSYQERLRFES-NGRPFLRMEQHT-TG
PD-GsPLHTEVGYLR--FVANgrLELVVAQPTGIVEVLEGGAEaeggvISLTS---LVVattPSAKRVDAVQRRFTL-
RGDe--LMTLEWMDAMERGMMQHLESRLRrg-----
>Comparative complete genome sequence analysis of the amino acid replacements responsible for the thermostability of Corynebacterium efficiens
mgarvermIfdarsapilprstgmdlhpnlAPYVGLTGTWTKGKHGFYPTIEDFSYEETLNFSTIPGKPFPRYEQKTMGL
Dg---PLHTELGFLR--ILDDgrAEFILAQPMGQTELEGMVSeegTlvFDFTF---STVansGSAKRVGTARTYTF-
TADrtgLSAQFAMGAVGQPLQQHLESELTkqs-----

```

Εισάγουμε το αρχείο .txt στο πρόγραμμα MEGA (στην προκειμένη περίπτωση χρησιμοποιείται η έκδοση X ή αλλιώς 10), θα στοιχίσουμε τις αλληλουχίες και θα τις αποθηκεύσουμε σε αρχεία τύπου .mas.



Ακόμη, θα επιλέξουμε τη διαδικασία Phylogeny έτσι ώστε να δημιουργήσουμε δέντρα με τις μεθόδους ένωσης γειτόνων, μέγιστης πιθανοφάνειας ή άλλων.



2. ΘΕΜΑ 2^ο

Παρακάτω παρατίθεται ο κώδικας του συγκεκριμένου θέματος:

```
from numpy import log as ln
dual_states=('A','B') #Δυο καταστάσεις (A,B)

possibility_a = [['A','G','T','C'],[0.4,0.4,0.1,0.1]] #Η πιθανότητα
να εκπέμψει πουργίνες και πυριμιδίνες στην κατάσταση A
possibility_b=[['T','C','A','G'],[0.3,0.3,0.2,0.2]] #Η πιθανότητα να
εκπέμψει πουργίνες και πυριμιδίνες στην κατάσταση B

goal=('G','G','C','T') #Η ζητούμενη αλληλουχία

pos_a_to_a_and_b=[0.9,0.1] #Η πιθανότητα να συνεχίσει από την A
κατάσταση στην A και στην B
pos_b_to_a_and_b=[0.1,0.9] #Η πιθανότητα να συνεχίσει από την B
κατάσταση στην B και στην A

initial_pos=[0.5,0.5] ##Η αρχική πιθανότητα να ξεκινήσει από το A
και το B

def
viterbi_algo(possibility_a,pos_a_to_a_and_b,goal,initial_pos,possibil
ity_b,from_b_to_b_and_a,dual_states):
    temp_a=[] #Αποθηκεύονται οι τιμές της A κατάστασης από τον
viterbi αλγόριθμο
    temp_b=[] #Αποθηκεύονται οι τιμές της B κατάστασης από τον
viterbi αλγόριθμο
    flag=True
    temp1=False
    temp2=False
    start=False
    log_num=0
    i=0
    best_path=[]
    while(flag): #Όσο το flag παραμένει true η while θα τρέχει
        for j in range(0,len(goal)): #Μπαίνει σε μια for με
μεγεθος οση είναι η ζητούμενη αλληλουχία(goal)
            if(start==False): #Εάν το start είναι false σημαίνει
οτι βρισκόμαστε στην πρώτη κατάσταση με ζητούμενο το G
                if(possibility_a[0][j]==goal[i]): #Εάν το
possibility_a[0][j] ισούται με το ζητούμενο
                    temp_a.append(initial_pos[0]*possibility_a[1][j])
#τότε η temp_a παίρνει την αρχική πιθανότητα του A και την
πολλαπλασιάζει με την πιθανότητα του ζητούμενου νουκλεοτιδίου
                    temp1=True
                if(possibility_b[0][j]==goal[i]):
                    temp_b.append(initial_pos[1]*possibility_b[1][j])
##τότε η temp_b παίρνει την αρχική πιθανότητα του B και την
πολλαπλασιάζει με την πιθανότητα του ζητούμενου νουκλεοτιδίου
                    temp2=True
```

```

        if(temp1 and temp2): #Εαν το temp1 και το temp2 είναι
true τότε έχουμε βρει το πρώτο ζητούμενο(goal[0])
            start=True #Το start γίνεται True οπότε το
προγραμμα δεν μπαίνει ξανά στην αρχική if
            i+=1
            if(temp_a[0]>temp_b[0]): #Εαν το temp_a που
βρήκαμε είναι μεγαλύτερο από το temp_b που βρήκαμε τότε
                best_path.append(dual_states[0]) #Το
best_path παίρνει την πρώτη κατάσταση
                log_num+= ln(temp_a[0])
#Μετατρέπει τις αριθμητικές τιμές σε λογαριθμικές και τις προσθέτει
σε μια μεταβλητή log_num
            elif(temp_a[0]<temp_b[0]): #Αλλιώς το best_path
παίρνει την δεύτερη κατάσταση
                best_path.append(dual_states[1])
                log_num+= ln(temp_b[0])
#Μετατρέπει τις αριθμητικές τιμές σε λογαριθμικές και τις προσθέτει
σε μια μεταβλητή log_num
            break #Αφού βρει ποιο είναι μεγαλύτερο
κάνει break από την for και ξεκινάει για τις υπολοιπές καταστάσεις
αφού το start γίνεται True στην σειρά 36
            if(start and temp1 and temp2): #Εάν και τα τρία flag
είναι true ψάχνουμε τις υπολοιπές καταστάσεις
                temp1=False #Κάνουμε το temp1 και το temp2 False
ώστε να τα χρησιμοποιήσουμε ξανά στον υπολοιπό κώδικα
                temp2=False
                if(start==True): #Εάν το start είναι True
                    if(possibility_a[0][j]==goal[i] and best_path[i-
1]=='A'): #Εάν το νουκλεοτιδίο της possibility_a ισούται με το
ζητούμενο νουκλεοτιδίο και το best_path ισούται με την πρώτη
κατάσταση (A)

temp_a.append(round(pos_a_to_a_and_b[0]*possibility_a[1][j]*temp_a[i-
1],10)) #Η temp_a παίρνει την πιθανότητα να πάει από το A στο A
πολλαπλασιάζοντας την με την πιθανότητα

#του ζητούμενου νουκλεοτιδίου και το temp_a της προηγούμενης
κατάστασης
                temp1=True
                elif(possibility_a[0][j]==goal[i] and best_path[i-
1]=='B'): #Αλλιώς εάν το νουκλεοτιδίο της possibility_a ισούται
με το ζητούμενο νουκλεοτιδίο και το best_path ισούται με την δεύτερη
κατάσταση (B)

temp_a.append(round(pos_b_to_a_and_b[0]*possibility_a[1][j]*temp_b[i-
1],10)) #Η temp_a παίρνει την πιθανότητα να πάει από το B στο A
πολλαπλασιάζοντας την με την πιθανότητα

#του ζητούμενου νουκλεοτιδίου και το temp_b της προηγούμενης
κατάστασης
                temp1=True
                if(possibility_b[0][j]==goal[i] and best_path[i-
1]=='A'): #Εάν το νουκλεοτιδίο της possibility_b ισούται με το
ζητούμενο νουκλεοτιδίο και το best_path ισούται με την πρώτη
κατάσταση (A)

temp_b.append(round(pos_a_to_a_and_b[1]*possibility_b[1][j]*temp_a[i-

```



```

1],10)) #Η temp_b παίρνει την πιθανότητα να πάει από το A στο B
πολλαπλασιάζοντας την με την πιθανότητα

#του ζητούμενου νουκλεοτιδίου και το temp_a της προηγούμενης
κατάστασης

temp2=True
elif(possibility_b[0][j]==goal[i] and best_path[i-
1]=='B'): #Αλλιώς εάν το νουκλεοτιδίο της possibility_b ισούται με
το ζητούμενο νουκλεοτιδίο και το best_path ισούται με την δεύτερη
κατάσταση(B)

temp_b.append(round(pos_b_to_a_and_b[1]*possibility_b[1][j]*temp_b[i-
1],10)) #Η temp_b παίρνει την πιθανότητα να πάει από το B στο B
πολλαπλασιάζοντας την με την πιθανότητα

#του ζητούμενου νουκλεοτιδίου και το temp_b της προηγούμενης
κατάστασης

temp2=True
if(temp1==True and temp2==True): #Εάν το temp1 και
το temp2 είναι True
    if(temp_a[i]>temp_b[i]): #Εάν το temp_a
είναι μεγαλύτερο του temp_b
        best_path.append(dual_states[0]) #Τότε το
best_path παίρνει την πρώτη κατάσταση
        log_num+= ln(temp_a[i])
#Μετατρέπει τις αριθμητικές τιμές σε λογαριθμικές και τις προσθέτει
σε μια μεταβλητή log_num
    else:
        best_path.append(dual_states[1]) #Αλλιώς το
best_path παίρνει την δεύτερη κατάσταση
        log_num+= ln(temp_b[i])
#Μετατρέπει τις αριθμητικές τιμές σε λογαριθμικές και τις προσθέτει
σε μια μεταβλητή log_num
    i+=1
    if(i==4 and len(best_path)==4): #Εάν το i έχει γίνει 4
και το best_path έχει 4 τιμές τότε
        flag=False #Το flag γίνεται false και όταν τελειώσει
η while βγαίνει από αυτήν
        for i in range(0,len(best_path)): #Με μια for
εκτυπώνει τις τιμές της A και της B κατάστασης καθώς και το μονοπάτι
με την μεγαλύτερη πιθανότητα
            print('A is: ',temp_a[i],'and B is: ',temp_b[i])
            print('The best path is: ',best_path,'and the rounded
sum of the logarithms for best_path is: ',round(log_num,4))
#Εκτυπώνει το καλύτερο μονοπάτι και το άθροισμα των λογαριθμικών
αθροισμάτων

        break #Αφού έχουν εκτυπωθεί όλα κάνει break, βγαίνει
από την for και αφού το flag έχει γίνει False τελειώνει και η while

viterbi_algo(possibility_a,pos_a_to_a_and_b,goal,initial_pos,possibil
ity_b,pos_b_to_a_and_b,dual_states) #Καλείται η viterbi_algo

```

Παράδειγμα εκτέλεσης:

```
A is: 0.2 and B is: 0.1
A is: 0.072 and B is: 0.004
A is: 0.00648 and B is: 0.00216
A is: 0.0005832 and B is: 0.0001944
The best path is: ['A', 'A', 'A', 'A'] and the rounded sum of the logarithms for best_path is: -16.7265
PS C:\Users\siatr> |
```

Επεξήγηση του συγκεκριμένου θέματος:

Σκοπός της εργασίας είναι η δημιουργία ενός προγράμματος το οποίο έχει ως στόχο να εκτελεί τον αλγόριθμο viterbi και να βρίσκει την πιθανότερη διαδρομή που θα ακολουθήσουν οι δυο καταστάσεις (α , β) για την αλληλουχία GGCT. Το πρόγραμμα παίρνει ως αρχικό όρισμα την πιθανότητα να εκπέμψει πουρίνες και πυριμιδίνες στις δύο καταστάσεις (α , β), την πιθανότητα να συνεχίσει από την κατάσταση (α) στην κατάσταση (β) και αντίστροφα, την αρχική πιθανότητα και την ζητούμενη αλληλουχία. Στην συνέχεια εκτελεί τον αλγόριθμο viterbi και εκτυπώνει το βέλτιστο μονοπάτι.

3. ΘΕΜΑ 3^ο

Παρακάτω παρατίθεται ο κώδικας του συγκεκριμένου θέματος:

```
import random
from Bio import SeqIO
#Φορτώνουμε τις 2 αλληλουχίες
for seq1 in SeqIO.parse("liver.fasta", "fasta"):
    chromo_liver = seq1.seq
for seq2 in SeqIO.parse("brain.fasta", "fasta"):
    chromo_brain = seq2.seq
chromo_liver = len(seq1) #Βρίσκουμε το μέγεθος της πρώτης αλληλουχιάς
chromo_brain = len(seq2) #Βρίσκουμε το μέγεθος της δεύτερης
αλληλουχιάς
player = 1
#Εδώ ελέγχουμε αν κέρδισε κάποιος παίχτης
def check(player):
    if(chromo_brain==0 or chromo_liver==0):
        print("The winner is Player ",player)
while(True): #Το παιχνίδι παίζει μέχρι να νικήσει ένας από τους δύο
παίχτες
    if(chromo_brain!=0 and chromo_liver!=0): #Αν δεν έχει μηδενίσει
καμία αλληλουχία μπαίνει στην if
        print("The player number ",player , " plays now. \nChoose if
you would like to erase from the 1.first(liver), 2.second(brain) or
3.from both sequences. "+
            "\nType 1, 2, or 3: ")
        if(chromo_brain<chromo_liver): #αν η αλληλουχία 1 είναι
μικροτερη απο την 2 τότε ο τυχαίος αριθμός θα είναι απο το 1 μέχρι το
μέγεθος της αλληλουχίας 1
            smaller_seq=chromo_brain
        else:
            smaller_seq=chromo_liver #αν η αλληλουχία 2 είναι
μικροτερη απο την 1 τότε ο τυχαίος αριθμός θα είναι απο το 1 μέχρι το
μέγεθος της αλληλουχίας 2
            randomness = random.randint(1,smaller_seq) #τυχαίος αριθμός
απο το 1 μέχρι τον αριθμό που ορίσαμε απο πάνω
            choice= int(input()) #Ο χρήστης δίνει στο πρόγραμμα το τι
θέλει να κάνει
            while(True): #Τρέχει μέχρι να δεχτεί σωστο τυπο input απο
τον χρηστη
                if(choice==1 or choice==2 or choice==3): #Αν η επιλογή
είναι 1 η 2 η 3 βγαίνει απο την while και συνεχίζει
                    break
                else:
                    print("Available choices: 1, 2 or 3 !!") #Αν η
επιλογή δεν είναι 1 η 2 η 3 εκτυπωνονται οι διαθεσιμες επιλογες
                    choice = int(input()) #Ο χρήστης δίνει στο πρόγραμμα
το τι θέλει να κάνει
            if(choice==1): #Αν επιλεξει την επιλογή 1
                chromo_liver-=randomness #Αφαιρούμε απο την αλληλουχία 1
τον τυχαίο αριθμό randomness
                check(player) #Ελέγχουμε μήπως τελειωσε το παιχνίδι
            elif(choice==2):#Αν επιλεξει την επιλογή 2
```

```

        chromo_brain--randomness #Αφαιρούμε απο την αλληλουχια 2
τον τυχαίο αριθμό randomness
        check(player) #Ελεγχουμε μηπως τελειωσε το παιχνιδι
        elif(choice==3):#Αν επιλεξει την επιλογη 3
            chromo_brain--randomness #Αφαιρούμε απο την αλληλουχια 1
τον τυχαίο αριθμό randomness και
            chromo_liver--randomness #Αφαιρούμε απο την αλληλουχια 2
τον τυχαίο αριθμό randomness
            check(player)

        if(player==1): #Αλλαγή παιχτών
            player=2 #Αν επαιζε ο παιχτης 1 τώρα εχει σειρα ο παιχτης
2
        elif(player==2): #Αλλαγή παιχτών
            player=1 #Αν επαιζε ο παιχτης 2 τώρα εχει σειρα ο παιχτης
1
    else: #Αν δεν μπει στην if
        break; #Βγαίνει απο την while

```

Παράδειγμα εκτέλεσης:

```

The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The player number 2 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
3
The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The player number 2 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
2
The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
2
The player number 2 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
2
The player number 2 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The player number 2 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
3
The player number 1 plays first.
Choose if you would like to erase from the 1.first(liver), 2.second(brain) or 3.from both sequences.
Type 1, 2, or 3:
1
The winner is Player 1
PS C:\Users\siatr\OneDrive - unipi.gr\ΠΑΝΕΙ\SEPTEMBER_2021\bioinformatics\Bioinformatics-main\Bioinformatics-main\source>

```

Επεξήγηση του προγράμματος:

Το χρωμόσωμα liver έχει αρχικώς 39314 νουκλεοτίδια και το χρωμόσωμα brain έχει αρχικαώς 49943 νουκλεοτίδια.

Σε πρώτο στάδιο, θα παίξει ο παίκτης 1 και θα επιλέγει αν θέλει να διαγράψει από το πρώτο χρωμόσωμα ή από το δεύτερο ή και από τα 2 χρωμοσώματα.

Στη συνέχεια, θα παίξει ο παίκτης 2 και θα επιλέγει αντίστοιχα από πού θέλει να διαγράψει έναν τυχαίο αριθμό νουκλεοτιδίων.

Το παιχνίδι θα τελειώσει όταν ο αριθμός των νουκλεοτιδίων σε ένα από τα 2 χρωμοσώματα θα γίνει ίσος με το 0.

4. ΘΕΜΑ 4^ο

4.1 Ερώτημα 1^ο

α) Η μέθοδος με την οποία έχει προσδιορισθεί η δομή του συμπλόκου είναι η : **X-RAY DIFFRACTION**

β) Η διακριτική ικανότητα της δομής είναι 1.77

γ) Το DOI της δημοσίευσης είναι: 10.1016/j.cell.2021.02.033

4.2 Ερώτημα 2^ο

α) Περιλαμβάνει 3 διακριτές πρωτεϊνικές αλυσίδες

β) COVOX-269 Fab heavy chain : το πλήθος των αμινοξέων είναι 222
COVOX-269 fab light chain : το πλήθος των αμινοξέων είναι 215 Spike
glycoprotein : το πλήθος των αμινοξέων είναι 205

γ) Περιέχει έναν ολιγοσακχαρίτη τον: 2-acetamido-deoxy-beta-D-glucopyranose-(1-4)-[alpha-L-fucopyranose-(1-6)]2-acetamido-2-deoxy-beta-D-glucopyranose

δ) Ανήκει στην αλυσίδα CA

4.3 Ερώτημα 3^ο

α)

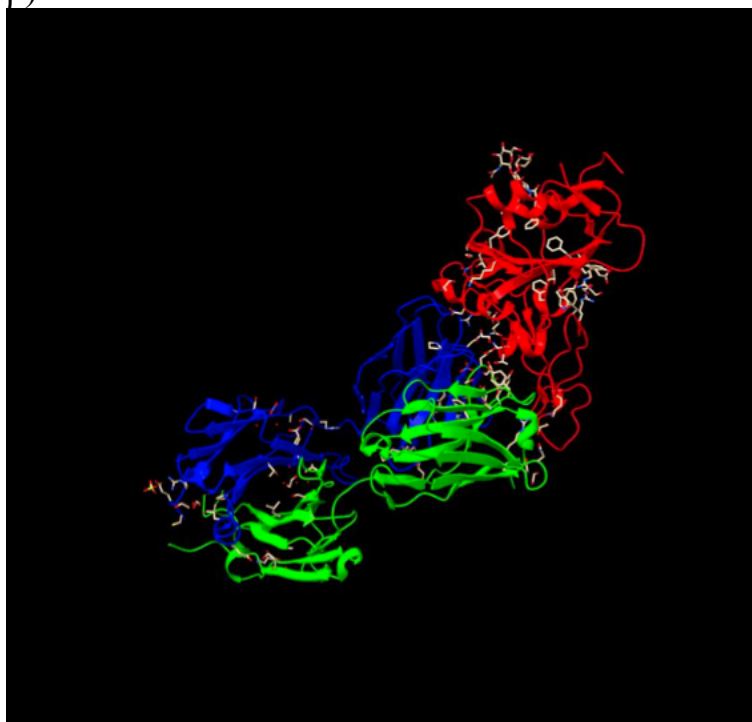
7neh.pdb title:

Crystal structure of the receptor binding domain of sars-cov-2 spike glycoprotein
In complex with covox-269 fab [\[more info...\]](#)

Chain information for 7neh.pdb #1	
Chain	Description
E	spike glycoprotein
H	covox-269 fab heavy chain
L	covox-269 fab light chain

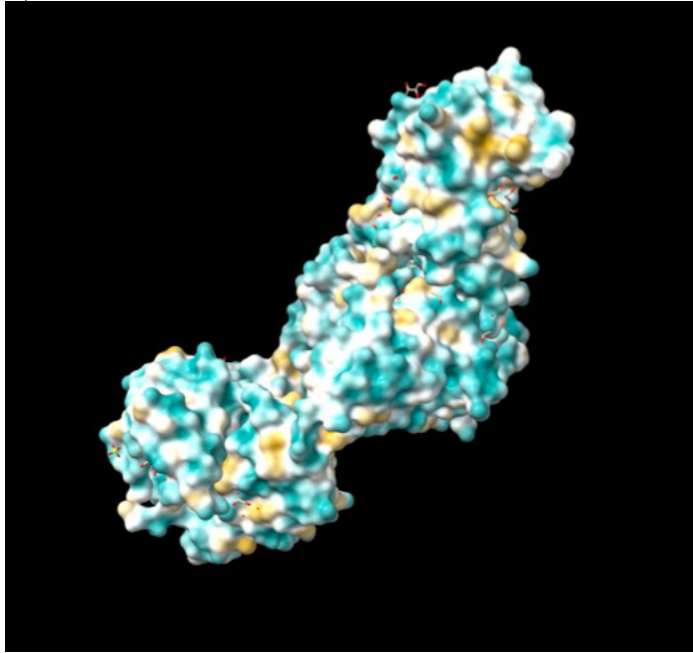
Non-standard residues in 7neh.pdb #1	
CL	chloride ion
EDO	1,2-ethanediol (ethylene glycol)
FUC	α-L-fucopyranose (α-L-fucose; 6-deoxy-α-L-galactopyranose; L-fucose; fucose)
NAG	2-acetamido-2-deoxy-β-D-glucopyranose (N-acetyl-β-D-glucosamine; 2-acetamido-2-deoxy-β-D-glucose; 2-acetamido-2-deoxy-D-glucose; 2-acetamido-2-deoxy-glucose; N-acetyl-D-glucosamine)
NO3	nitrate ion
PEG	di(hydroxyethyl)ether
SO4	sulfate ion

β)



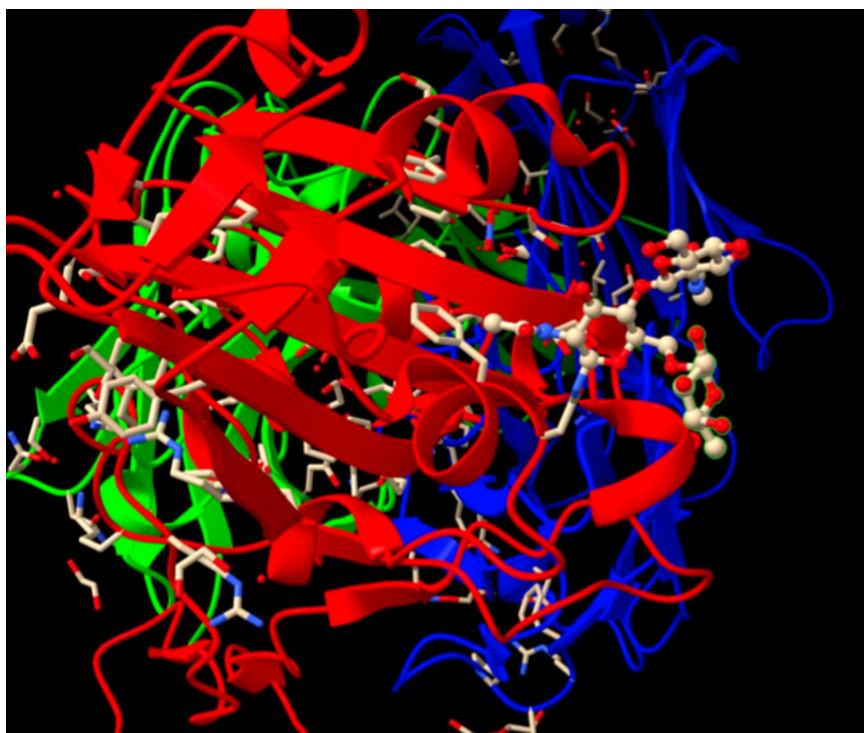
4.4 Ερώτημα 4^ο

α)



4.5 Ερώτημα 5^ο

β)



γ

