



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Εργασία Βιοπληροφορικής 2018-2019

Μέλη ομάδας :

Π16097 ΔΙΟΝΥΣΗΣ ΝΙΚΑΣ,
Π16195 ΝΗΣΙΩΤΗΣ ΜΑΡΙΝΟΣ

Περιεχόμενα

ΠΕΡΙΕΧΟΜΕΝΑ.....	2
ΑΣΚΗΣΗ 6.13.....	4
1.1 ΕΚΦΩΝΗΣΗ.....	4
1.2 ΨΕΥΔΟΚΩΔΙΚΑΣ	4
1.3 ΥΛΟΠΟΙΗΣΗ	8
1.4 ΑΠΟΤΕΛΕΣΜΑΤΑ	9
ΑΣΚΗΣΗ 6.15.....	11
2.1 ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ	11
2.2 ΕΥΡΕΣΗ ΑΚΟΛΟΥΘΙΩΝ ΝΟΥΚΛΕΟΤΙΔΙΩΝ.....	11
2.3 ΨΕΥΔΟΚΩΔΙΚΑΣ	11
2.4 ΥΛΟΠΟΙΗΣΗ	12
2.5 ΑΠΟΤΕΛΕΣΜΑΤΑ	14
ΑΣΚΗΣΗ 6.22.....	15
3.1 ΕΚΦΩΝΗΣΗ.....	15
3.2 ΕΥΡΕΣΗ ΑΚΟΛΟΥΘΙΩΝ ΝΟΥΚΛΕΟΤΙΔΙΩΝ.....	15
3.3 ΨΕΔΟΚΩΔΙΚΑΣ.....	15
3.4 ΥΛΟΠΟΙΗΣΗ	17
3.5 ΑΠΟΤΕΛΕΣΜΑΤΑ	19
ΑΣΚΗΣΗ 6.23.....	20
4.1 ΕΚΦΩΝΗΣΗ.....	20
4.2 ΨΕΥΔΟΚΩΔΙΚΑΣ	20
4.3 ΥΛΟΠΟΙΗΣΗ	22
4.4 ΑΠΟΤΕΛΕΣΜΑΤΑ	24
ΑΣΚΗΣΗ 6.37.....	25
5.1 ΕΚΦΩΝΗΣΗ.....	25
5.2 ΨΕΥΔΟΚΩΔΙΚΑΣ	25
5.3 ΕΥΡΕΣΗ ΑΚΟΛΟΥΘΙΩΝ ΑΜΙΝΟΞΕΩΝ ΚΑΙ ΝΟΥΚΛΕΟΤΙΔΙΩΝ	27
5.4 ΥΛΟΠΟΙΗΣΗ	27
5.5 ΑΠΟΤΕΛΕΣΜΑΤΑ	29
ΑΣΚΗΣΗ 11.4.....	30
6.1 ΕΚΦΩΝΗΣΗ.....	30
6.2 ΨΕΥΔΟΚΩΔΙΚΑΣ	30
6.3 ΥΛΟΠΟΙΗΣΗ	32
6.4 ΑΠΟΤΕΛΕΣΜΑΤΑ	33
ΑΣΚΗΣΗ 11.6.....	34
7.1 ΕΚΦΩΝΗΣΗ.....	34
7.2 ΨΕΥΔΟΚΩΔΙΚΑΣ	34
7.3 ΥΛΟΠΟΙΗΣΗ	37
7.4 ΑΠΟΤΕΛΕΣΜΑΤΑ	39
7.5 	39
ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΒΙΒΛΙΟΘΗΚΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ.....	40

ΒΙΒΛΙΟΘΗΚΕΣ ΜΑΤΛΑΒ	40
8.2 ΒΙΒΛΙΟΓΡΑΦΙΑ	40
8.3 ΒΙΒΛΙΟΓΡΑΦΙΑ ΔΙΑΔΙΚΤΥΟΥ	41

Άσκηση 6.13

1.1 Εκφώνηση

«Δύο παίκτες παίζουν το ακόλουθο παιχνίδι με δύο αλληλουχίες που έχουν μήκος n και m νουκλεοτίδια αντίστοιχα. Σε κάθε γύρο του παιχνιδιού, ένας παίκτης μπορεί να αφαιρέσει έναν τυχαίο αριθμό νουκλεοτιδίων από τη μια αλληλουχία ή τον ίδιο (αλλά και πάλι τυχαίο) αριθμό νουκλεοτιδίων και από τις δύο αλληλουχίες. Ο παίκτης που αφαιρεί το τελευταίο νουκλεοτίδιο κερδίζει. Ποιος θα κερδίσει; Περιγράψτε τη νικηφόρα στρατηγική για όλες τις τιμές των n και m .»

1.2 Ψευδοκώδικας

seq1 = ακολουθία_νουκλεοτιδίων
seq2 = ακολουθία_νουκλεοτιδίων

n = seq1_σε_μορφή_string
 m = seq2_σε_μορφή_string

Επιλογή παίκτη που θα παίξει πρώτος //παραδοχή δικιά μας είναι ότι ο παίκτης 1 θα παίζει τυχαία, ενώ ο 2 θα παίζει με τακτική//

Όσο $m \neq 0$ ΚΑΙ $n \neq 0$

Αν επιλογή == 1

[seq1, seq2] = συνάρτηση Player1(seq1, seq2, m , n)

Εμφάνιση της νέας ακολουθίας

επιλογή 2 //αλλαγή στον παίκτη 2

Αλλιώς

[seq1, seq2] = συνάρτηση Player2(seq1, seq2, m , n)

Εμφάνιση της νέας ακολουθίας

```

        επιλογή 1 // αλλαγή στον παίκτη 1
    Τέλος_αν
    m = string (seq1)
    n = string (seq2)
Τέλος_επανάληψης

Συνάρτηση [new_seq1, new_seq2] = Player1(seq1, seq2, m, n)
    Αν m==n
        Για i από 1 μέχρι m
            seq1 = seq1 (από 2 μέχρι το τέλος της)
            seq2 = seq2 ( από 2 μέχρι το τέλος της)
        Τέλος_επανάληψης

    Αλλιώς_αν n == 0
        Για i από 1 μέχρι n
            seq2 = seq2( από 2 μέχρι το τέλος της)
        Τέλος_επανάληψης

    Αλλιώς_αν m == 0
        Για i από 1 μέχρι m
            seq1 = seq1 (από 2 μέχρι το τέλος της)
        Τέλος_επανάληψης
    Αλλιώς
        c = γεννήτρια_τυχαίων_αριθμών [από 1 μέχρι 3]
        Αν c = 1
            Αριθμός = γεννήτρια_τυχαίων_αριθμών[από 1
            μέχρι το min(m, n)
            Για i από 1 μέχρι Αριθμός
                seq1 = seq1(από 2 μέχρι το τέλος της)
                seq2 = seq2 (από 2 μέχρι το τέλος της)
            Τέλος_επανάληψης
        Αλλιώς_αν c = 2
            Αριθμός = γεννητρία_τυχαίων_αιρθμών[από 1
            μέχρι το m
            Για i από 1 μεχρι Αριθμός
                seq1 = seq2( από 2 μέχρι το τέλος της)

```

```

        Τέλος_επανάληψης
    Αλλιώς
        Αριθμός = γεννήτρια_τυχαίων_αριθμών(από 1
                                                μέχρι n
        Για i από 1 μέχρι num
            seq1 = seq2(από 2 μέχρι το τέλος της)
        Τέλος_επανάληψης
    Τέλος_αν
Τέλος_συνάρτησης Player1

```

Επιστροφή των νέων ακολουθιών μετά την επεξεργασία

Συνάρτηση [new_seq1, new_seq2] = Player2(seq1, seq2, m, n)

```

Αν m==n
|   Για i από 1 μέχρι m
|       seq1 = seq1 (από 2 μέχρι το τέλος της)
|       seq2 = seq2 ( από 2 μέχρι το τέλος της)
|   Τέλος επανάληψης

```

```

Αλλιώς αν n == 0
|   Για i από 1 μέχρι n
|       seq2 = seq2( από 2 μέχρι το τέλος της)
|   Τέλος επανάληψης

```

```

Αλλιώς αν m == 0
|   Για i από 1 μέχρι m
|       seq1 = seq1 (από 2 μέχρι το τέλος της)
|   Τέλος επανάληψης

```

```

Αλλιώς
|   c = γεννήτρια τυχαίων αριθμών [από 1 μέχρι 3]
|   Αν c = 1

```

```

|      Αριθμός = γεννήτρια τυχαίων αριθμών[από 1  

|      μέχρι το min(m, n)  

|      Για i από 1 μέχρι Αριθμός  

|      seq1 = seq1(από 2 μέχρι το τέλος της)  

|      seq2 = seq2(από 2 μέχρι το τέλος της)  

|      Τέλος επανάληψης  

|      Αλλιώς αν c = 2  

|      Αριθμός = γεννήτρια τυχαίων αριθμών[από 1  

|      μέχρι το m  

|      Για i από 1 μέχρι Αριθμός  

|      seq1 = seq2(από 2 μέχρι το τέλος της)  

|      Τέλος επανάληψης  

|      Αλλιώς  

|      Αν loseProb(m-1, n)  

|      seq1 = seq1(από το 2 μέχρι το τέλος της)  

|      Αλλιώς_αν loseProb(m, n-1)  

|      seq2 = seq2(από το 2 μέχρι το τέλος της)  

|      Αλλιώς  

|      seq1 = seq1(από το 2 μέχρι το τέλος της)  

|      seq2 = seq2(από το 2 μέχρι το τέλος της)  

|      Τέλος_αν  

|      Τέλος_αν  

|      Τέλος_αν  

|      Επιστροφή νέων ακολουθιών μετά την επεξεργασία  

Τέλος_συνάρτησης Player2

```

```

Συνάρτηση loser = loseProb(m, n)
  Αν m==n
    χαμένος = 0
  Αλλιώς_αν m == 0 ΚΑΙ n>0
    χαμένος = 0
  Αλλιώς n == 0 ΚΑΙ m>0
    χαμένος = 0
  Αλλιώς
    χαμένος = 1

```

Τέλος_αν
Τέλος_συνάρτησης loser

/ Στην συνάρτηση Player2 ο υπογραμμισμένος κώδικας είναι ίδιας λογικής με της Player1. Για λόγους πληρότητας προσθέσαμε και αυτή */*

1.3 Υλοποίηση

move1 = διέγραψε κάποιον αριθμό (όχι τυχαίο) από την μια ακολουθία
move2 = διέγραψε κάποιον αριθμό (όχι τυχαίο) και από τις δύο ακολουθίες

Το παιχνίδι ξεκινάει και δίνει την επιλογή στον χρήστη να διαλέξει ποιος παίκτης θα παίξει πρώτος. Στη συγκεκριμένη άσκηση έχουμε κάνει την παραδοχή ότι ο παίκτης που θα επιλεγθεί να παίξει 1^{ος} από τον χρήστη, θα παίζει με τυχαίο τρόπο. Αφού επιλεγθεί δείχνει στην έξοδο τις ακολουθίες και ξεκινάει η διαδικασία παιχνιδιού. Όσο οι ακολουθίες m & n είναι διάφορες του μηδενός, δηλαδή δεν έχουν φτάσει στο τέλος του, ο παίκτης1 και ο παίκτης2 συνεχίζουν να παίζουν. Όταν τελειώσει με την κίνηση του ένας από τους δύο παίκτες, τότε η μεταβλητή choice αλλάζει αντίστοιχα για να παίξει ο αντίπαλος παίκτης. Όταν η μεταβλητή choice είναι 1 παίζει ο παίκτης1 και καλείται η συνάρτηση Player1. Αντίστοιχα για τον παίκτης2 καλείται η συνάρτηση Player2.

Στην συνάρτηση Player1, αν οι ακολουθίες m & n ισούνται τότε ο παίκτης1 βρίσκεται σε νικηφόρα θέση διαγράφοντας έναν κάποιον(όχι τυχαίο) αριθμό και από τις δύο ακολουθίες(move2). Αλλιώς αν η ακολουθία n ισούται με το 0, τότε ο παίκτης1 έχει κερδίσει αφού αφαιρώντας έναν κάποιον(όχι τυχαίο) αριθμό από την ακολουθία n μηδενίστηκε(move1). Επομένως αφαίρεσε και το τελευταίο νουκλεοτίδιο και νίκησε. Αντίστοιχα

ισχύει και για την ακολουθία m όταν αυτή ισούται με 0, για τον παίκτη1.

Για την επιλογή του τυχαίου αριθμού, ο παίκτης1 θα επιλέξει τυχαία ανάμεσα στις κινήσεις $move1$ & $move2$, για να αφαιρέσει οποιονδήποτε τυχαίο αριθμό από ολόκληρη/ρες την ακολουθία/ες. Στο τέλος της συνάρτησης επιστρέφονται οι νέες ακολουθίες μετά την αφαίρεση νουκλεοτιδίων.

Για την συνάρτηση $Player2$, όπου σε αυτή την περίπτωση ο παίκτης δεν παίζει τυχαία, ακολουθεί η ίδια λογική στην περίπτωση της αφαίρεσης ενός (όχι τυχαίου) αριθμού. Στην περίπτωση όπου ο παίκτης1 παίζει τυχαία, εδώ δημιουργήσαμε τη συνάρτηση $losePosi$ όπου ελέγχει αν η θέση που βρίσκεται ο παίκτης2 είναι μειονεκτική (*losing position*). Αν οι ακολουθίες έχουν το ίδιο μήκος, τότε επιστρέφει τιμή 0. Αν η ακολουθία m ισούται με 0 και η n είναι μεγαλύτερη από το 0 παλι θα επιστραφεί η τιμή 0. Αυτό γίνεται διότι από δω και πέρα θα πρέπει να κάνει κινήσεις που αφορούν τη μια ακολουθία, όχι και τις 2. Το ανάποδο, αν η n ισούται με 0 και η m μεγαλύτερη του μηδενός, η επιστρεφόμενη τιμή θα είναι 0. Τελικά επιστρέφει 1 μόνο όταν οι ακολουθίες είναι διάφορες του μηδενός και όχι ίσες.

1.4 Αποτελέσματα

Αρχικές ακολουθίες ενωμένες και οι δύο μαζί. Από κάτω και σε κάθε loop αναγράφονται οι νέες ακολουθίες που προκύπτουν.

```
Initial sequences :
GCAGTGCCGGGCGCCAGAGCAGCGGCGCCAGAGCAGCTGCACCATCCCGGCGTTCGCGTGTGCCGCCGCT
New sequences:
TTAA AACACAATT TTAAGGCCAGTGTGGTGGCTCACCCTGTAAATCCCAGTGATTTGGGAGGCTGAGGCCT
AGAGCTTAAAAATAAAGTGTCAATTTCCAAGGGCTA
```

```
Player 1 made a move
New sequences:
GTCGTGGCCAGCCCTGCAGCTTCAGCACCTGCCCCACCCAGAGTG
AGCTTAAAAATAAAGTGTCAATTTCCAAGGGCTA
```

```
Player 2 made a move
New sequences:
TCGTGGCCAGCCCTGCAGCTTCAGCACCTGCCCCACCCAGAGTGG
AGCTTAAAAATAAAGTGTCAATTTCCAAGGGCTA
```

```
Player 1 made a move
New sequences:
TGCAGCTTCAGCACCTGCCCCACCCAGAGTGGGAGTCAGGT
AGTGTCAATTTCCAAGGGCTA
```

```
Player 2 made a move
New sequences:
CTTTGTGGCTGTGGTTGGCTGA
AGTGTCAATTTCCAAGGGCTA
```

```
Player 1 made a move
New sequences:
CTGA
CTA
```

Στις 2 παραπάνω φωτογραφίες παρατηρούμε πως ο παίκτης2 δεν άλλαξε ιδιαίτερα τον τρόπο με τον οποίο παίζει γιατί, η μια ακολουθία είναι μεγαλύτερη από την δεύτερη. Ωστόσο όταν άρχισαν να έρχονται πιο κοντά, και να μειώνονται, τότε και ο παίκτης2 έπαιξε βάση της στρατηγικής καλώντας την σύναρτηση losePosi.

Άσκηση 6.15

2.1 Εκφώνηση άσκησης

«Δύο παίκτες παίζουν το ακόλουθο παιχνίδι με δυο αλληλουχίες που έχουν μήκος n και m νουκλεοτίδια αντίστοιχα. Σε κάθε γύρο του παιχνιδιού, ένας παίκτης μπορεί να αφαιρέσει δύο νουκλεοτίδια από τη μία αλληλουχία (είτε την πρώτη είτε την δεύτερη) και ένα νουκλεοτίδιο από την άλλη. Ο παίκτης που δεν μπορεί να κάνει κίνηση κερδίζει. Ποιος θα κερδίσει; Περιγράψτε τη νικηφόρα στρατηγική για όλες τις τιμές των n και m .»

2.2 Εύρεση ακολουθιών νουκλεοτιδίων

Ακολουθώντας το link στο [gunet](#) για την α -lactalbumin θα βρεθούμε στην ιστοσελίδα της ncbi όπου θα κατεβάσουμε και την ακολουθία FASTA και θα τη μετατρέψουμε σε αρχείο txt. Έπειτα στο matlab χρησιμοποιώντας την εντολή `fastaread` θα δημιουργηθεί ένα αντικείμενο με τον header και την ακολουθία από το αρχείο και εμείς θα επιλέξουμε την ακολουθία για να υλοποιήσουμε την άσκηση.

2.3 Ψευδοκώδικας

Για i από 3 μέχρι το μέγεθος της ακολουθίας:

 Αν `player_win` αφαιρέσει 2 νουκλεοτίδια

`player_win` = αληθής_παίξιμο

`player_moves` = συμπληρώνεται ο πίνακας με την κίνηση

 Αλλιώς_αν `player_win` αφαιρέσει 1 νουκλεοτίδιο

```
player_win= αληθής_παίξιμο  
player_moves = συμπληρώνεται ο πίνακας με την  
κίνηση
```

Αλλιώς

```
player_win= ψευδής_παίξιμο //δεν υπάρχουν άλλα  
νουκλεοτίδια
```

Τέλος_αν

Τέλος_επανάληψης

Για i από 10 μέχρι 20:

Εμφάνιση μηνυμάτων για το ποιος παίκτης κέρδισε και
ποιες ήταν οι κινήσεις που έκανε.

Τέλος_επανάληψης

2.4 Υλοποίηση

Πρώτα φορτώνουμε την ακολουθία η οποία θα χρησιμοποιηθεί για την εκτέλεση του προγράμματος. Δημιουργούμε ένα πίνακα player_win όπου στην περίπτωση που είναι κενός, έχουμε κάνει την παραδοχή πως ο παίκτης1 θα κερδίσει. Αν δεν είναι κενός ο πίνακας αρχικοποιείται με την ακολουθία και τους 2 παίκτες να έχουν τιμή Boolean true για το αν έχουν παίξει ή όχι. Στο loop, στην πρώτη συνθήκη αφαιρούνται από την ακολουθία 2 νουκλεοτίδια, σημειώνεται η κίνηση και ενημερώνεται με true(=1) ή false(=0) η σειρά του κάθε παίκτη. Στην δεύτερη συνθήκη (elseif) αφαιρείται 1 νουκλεοτίδιο, σημειώνεται η κίνηση και ενημερώνεται με true(=1) ή false(=0) η σειρά του κάθε παίκτη, όπως και πριν. Στην τρίτη συνθήκη, αν δεν υπάρχουν άλλα νουκλεοτίδια η σειρά του παίκτη ενημερώνεται με 0(false) που σημαίνει και το τέλος του loop.

Παρακάτω υπάρχουν τα μηνύματα εμφάνισης τα οποία δείχνουν τη σειρά αφαίρεσης νουκλεοτιδίων την οποία

ακολουθήσε ο κάθε παίκτης για να νικήσει, καθώς και τον νικητή.

Η στρατηγική παιξίματος κάθε φορά έχει να κάνει με το ποιος παίκτης θα φτάσει πρώτος στα 3 νουκλεοτίδια πχ: αν φτάσει ο παίκτης1 πρώτος στα 3 νουκλεοτίδια και είναι η σειρά του παίκτης2, τότε όποιο αριθμό νουκλεοτίδιων αφαιρέσει, ο παίκτης1 θα κερδίσει. Υποθετικά βγάζει 2 νουκλεοτίδια ο παίκτης2. Τώρα ο παίκτης1 μπορεί να αφαιρέσει το 1 νουκλεοτίδιο και να κερδίσει. Αν από την άλλη ο παίκτης2 αφαιρέσει 1 νουκλεοτίδιο, τότε ο παίκτης2 πάλι μπορεί να κερδίσει αφαιρώντας και τα 2 νουκλεοτίδια.

Βέβαια αυτή η στρατηγική μπορεί να επεκταθεί για η αριθμό νουκλεοτιδίων. Ας πάρουμε για παράδειγμα 10 νουκλεοτίδια. Τώρα σημασία έχει ποιος θα φτάσει πρώτος στα 9 νουκλεοτίδια, για να μπορεί να διαχειριστεί το παιχνίδι μέχρι το 3 όπου και είναι ο απώτερος σκοπός για να κερδίσει. Ο παίκτης1 παίζει και αφαιρεί 1, άρα έχουμε 9 νουκλεοτίδια και είναι η σειρά του παίκτης2. Αν και εκείνος αφαιρέσει 2 τότε θα πάμε στα 7. Η επόμενη κίνηση του παίκτη1 είναι να αφαιρέσει 1 νουκλεοτίδιο για να πάει στα 6. Τώρα ο παίκτης2 αν αφαιρέσει 2, θα πάμε στα 4 νουκλεοτίδια και ο παίκτης1 θα αφαιρέσει 1 νουκλεοτίδιο οπότε θα είναι και αυτός που θα κερδίσει γιατί έφτασε πρώτος στα 3. Αν πάλι ο παίκτης2 αφαιρέσει 1, θα πάμε στα 5 νουκλεοτίδια και ο παίκτης1 θα αφαιρέσει 2 αυτή τη φορά για να φτάσει πρώτος στα 3 και θα κερδίσει το παιχνίδι. Βλέπουμε δηλαδή πως η στρατηγική επεκτείνεται στους περιττούς αριθμούς κάθε φορά. Επομένως σημασία έχει ποιος θα φτάσει στο 9, μετά στο 7, 5 και τελικά στο 3. Αν γίνει κάποιο λάθος και φτάσει ο παίκτης2 πρώτος στο 7 ή στο 9 (όπως στο προηγούμενο παράδειγμα) τότε ο παίκτης1 θα πρέπει να κάνει κίνηση η οποία δεν θα επιτρέψει στον παίκτης2 να φτάσει πρώτος στον ζητούμενο περιττό αριθμό, 3. Στην περίπτωση όπου ο αριθμός των νουκλεοτιδίων είναι περιττός πχ: $n=11$ τότε ο παίκτης ο οποίος παίζει πρώτος, έχει από την αρχή το

προβάδισμα διότι ο αριθμός είναι ήδη περιττός. Άρα θα πρέπει να κρατήσει αυτή τη λογική και να στέλνει στον αντίπαλο παίκτη άρτιους.

Οποιοσδήποτε και να είναι ο αριθμός του n , αρκεί κάθε φορά ο παίκτης που θα επιχειρήσει την νικηφόρα στρατηγική να έχει κατά νου ότι πρέπει να φτάσει πρώτος στους περιττούς αριθμούς και να στέλνει στον αντίπαλο τους άρτιους.

2.5 Αποτελέσματα

Εικόνες από τις κινήσεις των 2 παικτών:

Για μέγεθος ακολουθίας 11 νικάει ο πρώτος παίκτης
Για μέγεθος ακολουθίας 12 νικάει ο δεύτερος παίκτης

```
For length 11:
Player 1 wins!! with moves:
    2    2    2    2    2    1

For length 12:
Player 2 wins!! with moves:
    2    2    2    2    2    2
```

Για το τελικό μήκος της ακολουθίας (μήκος = 20) νικάει ο πρώτος παίκτης

```
For length 20:
Player 1 wins!! with moves:
    2    2    2    2    2    2    2    2    2    2
```

Άσκηση 6.22

3.1 Εκφώνηση

«Διατυπώστε έναν αλγόριθμο που υπολογίζει τη βέλτιστη στοίχιση επικάλυψης και εκτελείται σε χρόνο $O(nm)$.»

3.2 Εύρεση ακολουθιών νουκλεοτιδίων

Χρησιμοποιώντας την προηγούμενη ακολουθία του προβλήματος 6.15, ακολουθούμε παρόμοια διαδικασία για το link στο gunet για την εύρεση της ακολουθίας νουκλεοτιδίων της Λυσοζύμης-c.

3.3 Ψεδοκώδικας

Δημιουργία πίνακας_ταιριάσματος αλφαβήτου νουκελοτιδίων

Δημιουργία πίνακα F με μηδενικά (διαστάσεων $m \times n$ όπου m = μήκος πρώτης ακολουθίας + 1 , και n = μήκος δεύτερης ακολουθίας + 1)

Για i από 2 μέχρι m

 Για j από 2 μέχρι n

 F(i,j) =

 πίνακας_ταιριάσματος(κωδικοποιημένη(πρώτη_ακολουθία(i-1)), κωδικοποιημένη(δεύτερη_ακολουθία(j-1)))

 Τέλος_επανάληψης

Τέλος_επανάληψης

```
σκορ = []
seqs1 = []
seqs2 = []
k = 2
p = 1
```

Για i από m μέχρι 2 με βήμα -1

```
σκορ = 0
```

```
l = i
```

```
γράμματα1 = []
```

```
γράμματα2 = []
```

Για j από 2 μέχρι k

```
σκορ = σκορ + F(l, j)
```

```
γράμματα1 = [γράμματα1, seq1(l-1)]
```

```
γράμματα2 = [γράμματα2, seq2(j-1)]
```

```
l = l+1
```

Τέλος_επανάληψης

```
k = k+1
```

```
σκορς = [σκορς, σκορ]
```

```
seqs1 = [seqs1 μετατροπή σε string τον πίνακα γράμματα1]
```

```
seqs2 = [seqs2 μετατροπή σε string τον πίνακα γράμματα2]
```

Τέλος_επανάληψης

```
μέγιστο = max(σκορς)
```

```
δείκτες = (σκορς όπου σκορ == μέγιστο)
```

```
τελικό1 = []
```

```
τελικό2 = []
```

Για q από 1 μέχρι μέγεθος(δείκτες)

```
τελικό1 = [τελικό1 seqs1(δείκτες(q))]
```

```
τελικό2 = [τελικό2 seqs2(δείκτες(q))]
```

Τέλος_επανάληψης

Εμφάνιση βέλτιστων στοιχίσεων επικάλυψης με το σκόρ

3.4 Υλοποίηση

Φορτώνουμε τις ακολουθίες και σχηματίζουμε τον πίνακα ομοιότητας όπου έχει 1 μόνο στην κύρια διαγώνιο τού (για τα γράμματα που αντιστοιχούνται μεταξύ τους είναι +1 δηλαδή). Όλα τα υπόλοιπα στοιχεία ισούνται με -1. Δημιουργούμε μία μήτρα μηδενικών στον οποίο θα αποθηκεύσουμε τα ταιριάσματα που θα βρούμε για κάθε γράμμα της μία ακολουθίας για όλα τα γράμματα της άλλης με την βοήθεια του πίνακα ομοιότητας. Αυτό γίνεται στο πρώτο διπλό loop όπου εξετάζουμε το ταιρίασμα για κάθε στοιχείο.

Δημιουργούμε τους πίνακες scores, seqs1, seqs2 όπου θα αποθηκεύσουμε όλα τα score όλων των επικαλύψεων που βρήκαμε, καθώς και όλες τις επικαλύψεις που βρήκαμε.

Η λογική για την εύρεση όλων των επικαλύψεων είναι η εξής:

Δημιουργώντας την μήτρα F με τις βαθμολογίες στοίχισης κάθε γράμματος των δύο ακολουθιών παρατηρούμε ότι για να υπολογίσουμε τα σκόρ στοίχισης όλων των προθεμάτων της πρώτης ακολουθίας και όλων των επιθεμάτων της δεύτερης αρκεί να πάρουμε τα στοιχεία της αριστερής διαγώνιου της μήτρας με συγκεκριμένη σειρά ώστε να μην προσπελάσουμε το ίδιο στοιχείο δεύτερη φορά (κάτι σαν ζίγκ-ζάγκ δηλαδή).

Επειδή όμως για να γίνει αυτό πρέπει ο πίνακας να είναι τετράγωνος, δηλαδή $n = m$ πρέπει να ισχύει και για $n > m$.

Άρα αυτό που κάνουμε είναι στην αρχή να θέτουμε το n ως $n + (\text{διαφορά του } m \text{ με το } n)$, π.χ αν $m = 9$ και $n = 6$ τότε το n για την αρχικοποίησης της μήτρας F θα γίνει $6 + 3 = 9$. Οι επιπλέον στήλες που θα προστεθούν θα γεμίσουν με -1 καθώς θα σημαίνει ότι δεν θα υπάρχει αντιστοιχία με κανένα γράμμα αφού θα είναι ο χαρακτήρας του κενού «-».

Αυτό γίνεται με την βοήθεια της μήτρας F, όπου μέσα στο loop, υπολογίζει όλα τα score στοίχισης από τα νουκλεοτίδια των ακολουθιών. Στις μεταβλητές letters1 και letters2 αποθηκεύονται αναδρομικά τα γράμματα της κάθε επικάλυψης κάθε φορά για να αποθηκευτούν στο τέλος της επανάληψης στον αντίστοιχο πίνακα.

Μόλις τελειώσει το loop το score που υπολογίστηκε ακριβώς από πάνω, αποθηκεύεται στον πίνακα scores, και τα letters1 και letters2 μετατρέπονται σε strings για να μπορέσουμε να τα αποθηκεύσουμε ως ένα στοιχείο και όχι ως αντικείμενο χαρακτήρων.

Τέλος πριν την εμφάνιση, βρίσκουμε το μέγιστο score όλων των scores και το αποθηκεύουμε στην μεταβλητή maximum. Αυτό γίνεται για να βρούμε τη καλύτερη στοίχιση επικάλυψης των δύο ακολουθιών. Χρησιμοποιούμε το μέγιστο σκόρ που βρήκαμε, για να βρούμε αν υπάρχουν και άλλες επικαλύψεις με το ίδιο σκόρ στο πίνακα τον σκόρ, και αν υπάρχουν αποθηκεύουμε την θέση τους σε ένα πίνακα indexes. Στο τελικό loop χρησιμοποιούμε όλα τα indexes των επικαλύψεων που πέτυχαν το μεγαλύτερο σκορ επικάλυψης για να τις απεικονίσουμε. Τέλος εμφανίζουμε τις καλύτερες στοίχισεις επικάλυψης μεταξύ των δύο ακολουθιών μαζί με το σκόρ.

3.5 Αποτελέσματα

Τα αποτελέσματα από την βέλτιστη στοίχιση επικάλυψης των ακολουθιών λυσοζύμης και λακταλβουμίνης

```
Command Window

Elapsed time is 1154.816280 seconds.

The best sequences with the highest score of global-alignment are:
    "C"      "ACGCC"
    "A"      "ATTTC"

With maximum score of: -1
fx >>
```

Άσκηση 6.23

4.1 Εκφώνηση

«Διατυπώστε έναν αλγόριθμο που υπολογίζει τη βέλτιστη στοίχιση προσαρμογής. Εξηγήστε πώς συμπληρώνεται η πρώτη γραμμή και η πρώτη στήλη του πίνακα δυναμικού προγραμματισμού και γράψτε μια σχέση επανάληψης για τη συμπλήρωση του υπόλοιπου πίνακα. Παρουσιάστε μια μέθοδο που βρίσκει την καλύτερη στοίχιση συμπληρωθεί ο πίνακας. Ο αλγόριθμος θα πρέπει να εκτελείται σε χρόνο $O(nm)$.»

4.2 Ψευδοκώδικας

```
seq1 = lysozyme.Sequence  
seq2 = nucleo.Sequence
```

```
Δημιουργία του πίνακα_ταιριάσματος  
Δημιουργία πίνακα F γεμάτος με 0
```

```
Για στήλη j από 0 μέχρι n-1:  
    F(πρώτη_γραμμή, j) = -j // για όλη την πρώτη γραμμή  
Τέλος_επανάληψης
```

```
Για γραμμή i από 0 μέχρι m-1  
    F(i, πρώτη_στήλη) = -i // για όλη την πρώτη στήλη  
Τέλος_επανάληψης  
Για i από 2 μέχρι τέλος_στήλης  
    Για j από 2 μέχρι τέλος_γραμμής  
        ταίριασμα = F(i-1, j-1) +  
        πίνακα_ταιριάσματος(nt2int(seq1(i-1)), nt2int(seq2(j-1)));
```

```

        διαγραφή = F(i-1, j) -1;
        εισαγωγή = F(i, j-1) -1;
        μέγιστο_μπλοκ= [ταίριασμα, διαγραφή, εισαγωγή]
        F(i, j) = μέγιστο(μέγιστο_μπλοκ)
    Τέλος_επανάληψης
Τέλος_επανάληψης

```

```

ΔήλωσηπίνακαAlignmentA = []
ΔήλωσηπίνακαAlignmentB = []
ΔήλωσηπίνακαAlignment = []

```

```

σκορ = F(i, j)
(σκορ, δείκτης) = μέγιστο (F(:, n))
i =μέγιστο((F(:, n)==σκορ))
σκορ = 0
Όσο j>1 επανέλαβε
    Αν(i>1 ΚΑΙ j>1 ΚΑΙ F(i,j) == F(i-1, j-1) +ταίριασμα
        Alignment = [seq1(i-1) AlignmentA]
        Alignment = [seq1(j-1) AlignmentB]
        Αν (seq1(i-1) == seq2(j-1))
            Καταχώρησε '|'
        Αλλιώς
            Καταχώρησε ':'
        Τέλος_αν
        σκορ = σκορ + F(i, j)
        i = i-1
        j = j-1

```

```

Αλλιώς_αν (j>1 ΚΑΙ F(i,j) == F(i, j-1)-1)
    AlignmentA = [seq1(i-1) AlignmentA]
    AlignmentB = καταχώρησε '-'
    Alignment = καταχώρησε ' '
    σκορ = σκορ + F(i, j)
    i = i-1

```

```

    Αλλιώς_αν (j>1 ΚΑΙ (F(i, j) == F(i, j-1) -1))
        AlignmentA = καταχώρησε '-'
        AlignmentB = [seq2(j-1) AlignmentB]
        Alignment = καταχώρησε ' '
        σκορ = σκορ + F(i, j)
    Τέλος_αν
Τέλος_επανάληψης

Όσο AlignmentB(i) == '-'
    πήγαινε στο επόμενο
Τέλος_επανάληψης

Εμφάνιση της καλύτερης βαθμολογίας στοίχισης

```

4.3 Υλοποίηση

Φορτώνουμε τις ακολουθίες και σχηματίζουμε τον πίνακα ομοιότητας όπου έχει 1 μόνο στην κύρια διαγώνιο τού (για τα γράμματα που αντιστοιχούνται μεταξύ τους είναι +1 δηλαδή). Όλα τα υπόλοιπα στοιχεία ισούνται με -1. Δημιουργούμε μία μήτρα μηδενικών F (διαστάσεων mXn όπου m = μήκος πρώτης ακολουθίας +1 , και n= μήκος δεύτερης ακολουθίας + 1).

Για την υλοποίηση της άσκησης θα πρέπει στην ουσία να υλοποιήσουμε τον αλγόριθμο **Needleman–Wunsch** με κάποιες αλλαγές ώστε να κάνει την στοίχιση χωρίς όμως να αφαιρεί στοιχεία της δεύτερης ακολουθίας από το τελικό αποτέλεσμα.

- Γέμισμα μήτρας F

Σε όλη την πρώτη γραμμή της F θα υπάρχουντο σκόρ της πρώτης ακολουθίας με το «κενό» (άρα για i από 1 μέχρι N θα

είναι -i), το ίδιο και για τα στοιχεία της πρώτης στήλης για τη δεύτερη ακολουθία.

Στο παρακάτω διπλό βρόχο επανάληψης για κάθε κελί θα συγκρίνουμε τη βαθμολογία των γειτονικών κελιών του (δηλαδή πάνω, πάνω-αριστερά και αριστερά κελιά) παίρνοντας κάθε φορά το μέγιστο από αυτά και θέτοντας το στο κελί που είμαστε. Να σημειωθεί εδώ ότι η σύγκριση των βαθμολογιών των γειτονικών κελιών γίνεται ως εξής: α) Αν το πάνω-αριστερά κελί έχει αντιστοιχία χαρακτήρων (δηλαδή ο πίνακα ομοιότητας επιστρέφει 1), τότε η βαθμολογία του κελιού αυτού θα είναι η ήδη υπάρχουσα + 1 για τη σύγκριση, αλλιώς -1, β) Για τις περιπτώσεις των αριστερών και πάνω κελιών, που σημαίνουν διαγραφή ή προσθήκη τότε η βαθμολογία του εκάστοτε κελιού θα είναι η ήδη υπάρχουσα -1 για τη σύγκριση.

Έχοντας τη μέγιστη τιμή από αυτά τα τρία κελιά γεμίζουμε κάθε φορά και ένα κελί και μέσω αυτής της διαδικασίας γεμίζει ολόκληρη η μήτρα F.

- Εύρεση καλύτερης στοίχισης προσαρμογής

Δημιουργούμε 3 πίνακες AlignmentA, AlignmentB, Alignment όπου θα χρησιμοποιηθούν για την εμφάνιση των τελικών ακολουθιών προσαρμογής. (Ο πίνακας Alignment είναι για την εμφάνιση των συμβόλων «|, :» που είναι για την αντιστοίχιση συμβόλων.

Τώρα για την εύρεση της καλύτερης στοίχισης προσαρμογής από τη μήτρα F θα ακολουθήσουμε την εξής ιδέα :

Αρχικά θα βρούμε τη καλύτερη βαθμολογία σε όλη την τελευταία στήλη. Αυτό θα το κάνουμε ώστε να ξεκινήσουμε την διαδικασία του backtracking από την αντιστοιχία που δίνει τη μέγιστη συνολική βαθμολογία μεταξύ του τελευταίου γράμματος της δεύτερης ακολουθίας (ακολουθίας προσαρμογής) με το εκάστοτε γράμμα της πρώτης ακολουθίας. Άρα ξεκινώντας από αυτή τη θέση στη μήτρα F κάνουμε backtracking εξασφαλίζοντας ότι η ακολουθία προσαρμογής θα μείνει όπως

είναι και απλά θα έχουμε πια την πιο όμοια υπο-ακολουθία της πρώτης ακολουθίας για να τη συγκρίνουμε.

Τέλος στη διαδικασία του backtracking, ελέγχουμε κάθε φορά αν με βάση τους κανόνες σύγκρισης των κελιών που αναφέραμε πριν, τα κελία έχουν τις αντίστοιχες τιμές και ανάλογα συγκρίνουμε αν τα γράμματα των ακολουθιών που συναντάμε στο «μονοπάτι» είναι ίδια ή διαφορετικά, και ανάλογα αν κάνουμε αντικατάσταση, προσθήκη ή αφαίρεση προσθέτουμε τους κατάλληλους χαρακτήρες ή σύμβολα στους αντίστοιχους πίνακες Alignment που αναφέραμε πριν.

Σημαντικό είναι να αναφέρουμε ότι λόγω της φύσης του αλγόριθμου Needleman-Wunch μπορεί να υπάρχουν παραπάνω από μία βέλτιστες στοιχίσεις ακολουθιών προσαρμογής επειδή στο backtracking γίνεται διακλάδωση σε διαφορετικά μονοπάτια αν υπάρχει ισότητα στις τιμές των γειτονικών κελιών.

4.4 Αποτελέσματα

Μετά την εκτέλεση του προγράμματος για τις ακολουθίες λυσοζύμη και λακταλβουμίνη παίρνουμε τις εξής στοιχίσεις προσαρμογής:

```
The best sequence alignment is:
--TCCCG---CTGTGTGTACG-ACACTGGCAACATGAGGT-CTTTG-CTAATCTTGGTGCT-
|:|:|  || || |:|:| | ||  |||:||||:| | |||| || :|| ||:|:|
ATTTTCAGAAATCT-TG-GGGGGTA-AC---CAAAATGATGTCCTTTGTCT-CTC-TGCTCCTG
With score of: 206974
x >> |
```

Εκτιμώμενος χρόνος: ~5 λεπτά

Άσκηση 6.37

5.1 Εκφώνηση

«Επινοήστε έναν αποδοτικό αλγόριθμο για το πρόβλημα της Χιμαιρικής Στοιχίσης»

5.2 Ψευδοκώδικας

Για από 1 μέχρι μέγεθος(ακολουθίας αμινοξέων)
seq = μετέτρεψε σε ακολουθίανουκλεοτιδίων

Δημιουργία πίνακα newSeq []

Δημιουργία πίνακα counts[με 4 ορίσματα για τα στοιχεία]

παρόν_στοιχείο= αρχικοποίηση με το 1^ο στοιχείο της
ακολουθίας seq(1)

Για από 2 μέχρι μέγεθος_ακολουθίας

επόμενο_στοιχείο= seq(i)

Αν το επόμενο_στοιχείο<>παρόν_στοιχείο

newSeq = newSeq, παρόν_στοιχείο

μετρητής = resetCounts(επόμενο_στοιχείο,
μετρητής)

παρόν_στοιχείο = επόμενο_στοιχείο

Αλλιώς

[μετρητής, μέγιστο,newSeq]= checkNext()

Αν είναι <>μέγιστο ΚΑΙ (i == length(seq))

newSeq = [newSeq, παρόν_στοιχείο]

Τέλος_αν

Τέλος_αν

Τέλος_επανάληψης

Τέλος_επανάληψης

Συνάρτηση μετρητήςAGCT (resetCounts(επόμενο_στοιχείο,
μετρητής))

πίνακα_γραμμάτων= ['A', 'C', 'G', 'T']

Για k από 1 μέχρι μέγεθος πίνακα_γραμμάτων

Αν επόμενο_στοιχείο == πίνακα_γραμμάτων(k)

μετρητής(k) = 1

Αλλιώς

μετρητής(k) = 0

Τέλος_αν

Τέλος_επανάληψης

Επιστροφή μέτρησηAGCT= [μετρήσεις για τα στοιχεία(
1, 2, 3, 4)]

Τέλος_συνάρτησης

Συνάρτηση [μετρητήςAGCT, μέγιστο, νεα_ακολουθ] =

checkNext(επόμενο_στοιχείο, μετρητής, ακολουθία,

παρόν_στοιχείο)

πίνακα_γραμμάτων= ['A', 'C', 'G', 'T']

Για k από 1 μέχρι μέγεθος πίνακα_γραμμάτων

όρια = [6, 11, χωρίς_προσδιορισμό, χωρίς_προσδιορισμό]

Αν επόμενο_στοιχείο == πίνακα_γραμμάτων(k)

μετρητής(k) = μετρητής(k) + 1

Αν μετρητής(k) = όρια(k)

ακολουθία = [ακολουθία, παρόν_στοιχείο]

μετρητής(k) = 0 //μηδενισμός μετρητή για νέα
μέτρηση

μέγιστο = αληθής

Αλλιώς

μέγιστο = ψευδής

Τέλος_αν

Τέλος_αν

Τέλος_επανάληψης

νεα_ακολουθ = ακολουθία

επιστροφή μετρητηAGCT(μετρήσεις για τα στοιχεία(1, 2, 3, 4)]

Τέλος_συνάρτησης

5.3 Εύρεση ακολουθιών αμινοξέων και νουκλεοτιδίων

Ψάχνοντας τον κωδικό 1BBT για το καψίδιο βρίσκουμε στην ιστοσελίδα του RCSB τις 4 ακολουθίες αμινοξέων για τις 4 μορφές του FOOT AND MOUTH DISEASE VIRUS. Έχοντας τις 4 ακολουθίες αμινοξέων για κάθε μία χρησιμοποιούμε την εντολή του matlab aa2nt όπου μετατρέπει ακολουθίες αμινοξέων σε ακολουθίες νουκλεοτιδίων. Έτσι αποτυπώνουμε και την αμινοξική ακολουθία του ιού και την νουκλεοτιδική ακολουθία του ιού.

5.4 Υλοποίηση

Αρχικά φορτώνουμε την ακολουθία που είναι προς επεξεργασία στη μεταβλητή seq. Δημιουργούμε τους πίνακες newSeq όπου θα αποθηκεύσει την νέα ακολουθία, και το πίνακα counts όπου είναι σαν «μετρητής» για τις συνεχόμενες εμφανίσεις του κάθε γράμματος στην ακολουθία. Στο δεύτερο loop ελέγχεται αν το επόμενο στοιχείο της ακολουθίας είναι ίδιο με το αυτό στο οποίο βρίσκεται τώρα, όπου στην αρχή αρχικοποιούμε ως πρώτο στοιχείο το πρώτο γράμμα της ακολουθίας. Αν δεν είναι τότε προσθέτει στο πίνακα newSeq το παρόν στοιχείο. Καλεί την συνάρτηση resetCounts όπου θα επαναφέρει το μετρητή (που κρατάει τις συνεχόμενες εμφανίσεις για τα στοιχεία). Το επόμενο στοιχείο γίνεται το παρόν και το loop συνεχίζει. Αν είναι ίδια τα γράμματα τότε καλείτε η συνάρτηση checkNext όπου ελέγχει πιο είναι το επόμενο στοιχείο μετά από αυτό που βρισκόμαστε τώρα, λαμβάνοντας υπόψιν και τους περιορισμούς που έχει το κάθε

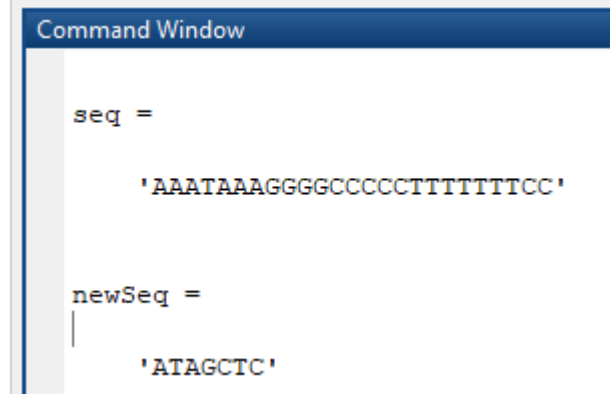
στοιχείο. Αν δεν έχει ξεπεραστεί το μέγιστο πλήθος του περιορισμού ή η ακολουθία έχει φτάσει στο τέλος, τότε η νέα ακολουθία (`newSeq`) είναι η νέα ακολουθία συν το παρόν στοιχείο.

Η συνάρτηση `resetCounts` δέχεται ως όρισμα το επόμενο στοιχείο της ακολουθίας, τα πλήθη των στοιχείων την ακολουθία και το τωρινό στοιχείο της ακολουθίας. Μετά επαναφέρει τον αριθμό εμφανίσεων όλων των στοιχείων σε 0 και απλά προσθέτει 1 στο στοιχείο που είναι το επόμενο στην ακολουθία. Έπειτα επιστρέφει τον αριθμό εμφανίσεων του κάθε στοιχείου.

Η συνάρτηση `checkNext` δέχεται ως ορίσματα το επόμενο στοιχείο, το πόσες φορές εμφανίστηκε στην ακολουθία, την ακολουθία και το παρόν στοιχείο. Μετράει το πλήθος των εμφανίσεων του κάθε στοιχείου, λαμβάνοντας υπόψιν και τα όρια, τα οποία δίνονται από την εκφώνηση του βιβλίου (στα όρια προσθέτουμε +1 καθώς στην εκφώνηση λέει ότι γίνεται προσθήκη ενός νουκλεοτιδίου Πολύ-C, και ούτω καθεξής). Έτσι το γράμμα A μπορεί να έχει μέχρι και 6 συνεχόμενα, το γράμμα C μέχρι και 10, ενώ για τα γράμμα G και T, δεν δίνεται τερματικός αριθμός εμφανίσεων. Η συνάρτηση επιστρέφει την νέα ακολουθία, την λογική τιμή αν έχει υπερβεί κάποιο όριο και τον αριθμό των εμφανίσεων του κάθε γράμματος.

5.5 Αποτελέσματα

Όπως και στα προηγούμενα παραδείγματα, χρησιμοποιήθηκε το παράδειγμα του βιβλίου σελ 255 επειδή το μήκος της ακολουθίας από το gunet2 είναι πολύ μεγάλο. Στο κώδικα υπάρχουν και τα παραδείγματα από το gunet2, απλώς πρέπει να γίνει η επιλογή τους. Βρίσκονται σαν σχόλια.



```
Command Window

seq =

    'AAATAAAGGGGCCCCCTTTTTTCC'

newSeq =
|
    'ATAGCTC'
```

Άσκηση 11.4

6.1 Εκφώνηση

«Στο σχήμα 11.7 φαίνεται ένα HMM με δύο καταστάσεις α και β . Όταν το HMM βρίσκεται στην κατάσταση α , έχει την μεγαλύτερη πιθανότητα να εκπέμψει πουρίνες (Α και G). Όταν βρίσκεται στην κατάσταση β , έχει μεγαλύτερη πιθανότητα να εκπέμψει πυριμιδίνες (C και T). Αποκωδικοποιήστε την πιο πιθανή ακολουθία των καταστάσεων (α/β) για την αλληλουχία GGCT. Χρησιμοποιήστε λογαριθμικές βαθμολογίες για κανονικές βαθμολογίες πιθανοτήτων.»

6.2 Ψευδοκώδικας

Μήτρα πιθανοτήτων μεταβολής καταστάσεων A
Μήτρα πιθανοτήτων εκπομπής B

x = Δοθέν_ακολουθία ('GGCT')

Αρχικοποίηση πιθανοτήτων $[\frac{1}{2}, \frac{1}{2}]$ για την κάθε κατάσταση
ακολουθία = x

Αλλαγή πινάκων(ανάστροφος A, αντίστροφος B)

[σκορ, καλύτερο_μονοπάτι] = συνάρτηση

viterbi4(αρχικοποίηση_πιθανοτήτων, A, B, ακολουθία)

Εμφάνιση σκορ

Ψευδοκώδικας συναρτησης viterbi4

Συνάρτηση [μέγιστο, καλύτερο_μονοπάτι,
δεύτερο_μονοπάτι]=**viterbi4**(αρχικοποίηση_πιθανοτή
των, A, B, ακολουθία)

N = μέγεθος (A, 1)

M = μέγεθος (B, 1)

T = μέγεθος(ακολουθίας)

C = πίνακας πιθανοτήτων viterbi (N, T)

πρόβλεψη= μέγεθος(N, T)

L = μέγεθος(A, 2)

αρχικοποίηση_ πιθανοτήτων= $\log_2(\text{αρχικοποίηση_πιθανοτήτων})$

A = $\log(A)$

B = $\log(B)$

Για i από 1 μέχρι N

πίνακας_μηδενικών = αρχικοποίηση_πιθανότητας(i)

Τέλος_επανάληψης

vr = πίνακας_μηδενικών_διαστάσεων(T, N, 2)

τελικο_σκορ = πίνακας_μηδενικών_διαστάσεων(1, N)

Για t από 1 μέχρι T // μέγεθος ακολουθίας

Για i από 1 μέχρι N // πλήθος καταστάσεων

v = πίνακας_μηδενικών_διαστάσεων(1,N)

Για p από 1 μέχρι N

$v(p) = C(p,t)+A(p,i)+B(x(t),p)$ //πιθανότητα

να γίνει εκπομπή του συμβόλου B(x(t),p) βρισκόμενοι στη
κατάσταση με πιθανότητα C(p,t) μεταβάλλοντας την κατάσταση
με πιθανότητα A(p,i)

Τέλος_επανάληψης

[μεγιστη πιθανότητα, δείκτης] max(v)

Προβλεπόμενο βήμα = δείκτης

Τέλος_επανάληψης
Τέλος_επανάληψης

[μέγιστο_σκορ τελευταίων στοιχειών πίνακα viterbi, δείκτης] =
max(C(:,T+1))

μεγιστο= μέγιστο_σκορ
πρόβλεψη(N, T) = δείκτης
t = T
καλύτερο_μονοπάτι = [δείκτης]

Όσο t <> 0
 [σκορ, δείκτης] = μέγιστο(πίνακα_vp(t, :, δείκτης))
 καλύτερο_μονοπάτι = [καλύτερο_μονοπάτι, δείκτης]
 t = t-1
 Αν t == 0
 Break
Τέλος_αν
Τέλος_επανάληψης
σκορ = μέγιστο_σκορ

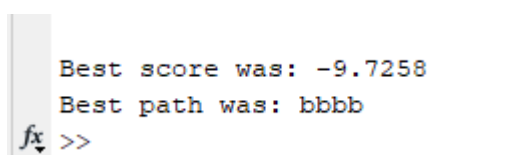
6.3 Υλοποίηση

Ξεκινώντας από το σχήμα του βιβλίου κατασκευάζουμε τους πίνακες μεταβολής καταστάσεων και εκπομπής συμβόλων. Έχοντας την ακολουθία από την εκφώνηση της άσκησης την κωδικοποιούμε σε αριθμούς χρησιμοποιώντας την εντολή του matlab nt2int. Έχοντας πια την κωδικοποιημένη ακολουθία πια την βάζουμε ως όρισμα καλώντας την συνάρτηση viterbi4 όπου θα μας επιστρέψει την μέγιστη πιθανότητα σε λογαριθμική βαθμολογία και την αντίστοιχη ακολουθία καταστάσεων.

- Αλγόριθμος viterbi4

Η υλοποίηση του αλγόριθμου Viterbi σε αυτήν την άσκηση έγινε ως εξής. Πρώτα μετατρέπουμε όλες τις τιμές των πινάκων πιθανοτήτων σε λογαριθμικές τιμές για να μην υπάρχει underflow με τις κανονικές πιθανότητες που θα υπολογίσουμε. Μετά ξεκινάμε βάζοντας στην αρχή του πίνακα Viterbi τις αρχικές πιθανότητες των καταστάσεων μας και συνεχίζουμε για κάθε σύμβολο της ακολουθίας τον υπολογισμό της πιθανότητας να εκπεμφθεί δοσμένης της τωρινής κατάστασης και της πιθανότητας της προηγούμενης έχοντας και «βέλη» οπισθοδρόμησης για την κατάσταση στην οποία καταλήγουμε κάθε φορά. Αφού υπολογίσουμε ολόκληρο το πίνακα Viterbi και βρούμε την μέγιστη πιθανότητα στο τέλος ανάμεσα στις δύο καταστάσεις ξεκινάμε το backtracking με τη βοήθεια των βέλων οπισθοδρόμησης.

6.4 Αποτελέσματα



```
Best score was: -9.7258
Best path was: bbbb
fx >>
```

Άρα εδώ βλέπουμε ότι για την ακολουθία GGCT η καλύτερη ακολουθία καταστάσεων που δίνει μέγιστη πιθανότητα (λογαριθμική) -9.7258 είναι η BBBB, δηλαδή μόνο η δεύτερη κατάσταση.

Άσκηση 11.6

7.1 Εκφώνηση

«Θεωρήστε ένα διαφορετικό παιχνίδι στο οποίο ο κρουπιέρης δεν ρίχνει νόμισμα αλλά ζάρι με τρεις πλευρές που έχουν ετικέτες 1, 2, και 3. (Μην προσπαθήσετε να σκεφτείτε την εμφάνιση ενός τέτοιου ζαριού.) Ο κρουπιέρης έχει δύο στημένα ζάρια $D1$ και $D2$. Για κάθε ζάρι D_i , η πιθανότητα να προκύψει ο αριθμός i είναι ίση με $\frac{1}{2}$, και η πιθανότητα για τα άλλα δυο αποτελέσματα είναι ίση με $\frac{1}{4}$. Σε κάθε γύρο, κρουπιέρης πρέπει να αποφασίσει αν (1) θα κρατήσει το ίδιο ζάρι, (2) θα αλλάξει ζάρι, ή (3) θα σταματήσει το παιχνίδι. Επιλέγει το (1) με πιθανότητα $\frac{1}{2}$ και τα (2) και (3) με πιθανότητα $\frac{1}{4}$. Στην αρχή ο κρουπιέρης επιλέγει ένα από τα δύο ζάρια με την ίδια πιθανότητα»

7.2 Ψευδοκώδικας

Μήτρα πιθανοτήτων μεταβολής καταστάσεων A

Μήτρα πιθανοτήτων εκπομπής B

Αλφάβητο καταστάσεων

$x = \text{Δοθέν_ακολουθία } (1\ 1\ 2\ 1\ 2\ 2)$

Αρχικοποίηση πιθανοτήτων $[\frac{1}{2}, \frac{1}{2}]$ για την κάθε κατάσταση
ακολουθία = x

Αλλαγή πινάκων (ανάστροφος A , αντίστροφος B)

$[\text{σκορ}, \text{καλύτερο_μονοπάτι}, \text{δεύτερο_μονοπάτι}] = \text{συνάρτηση}$
 $\text{viterbi6}(\text{αρχικοποίηση_πιθανοτήτων}, A, B, \text{ακολουθία})$

Εμφάνιση σκορ

Εμφάνιση δυο μονοπατιών

Ψευδοκώδικας συναρτησης viterbi6

Συνάρτηση [μέγιστο, καλύτερο_μονοπάτι,
δεύτερο_μονοπάτι]=**viterbi6**(αρχικοποίηση_πιθανοτή
των, A, B, ακολουθία)

N = μέγεθος (A, 1)

M = μέγεθος (B, 1)

T = μέγεθος(ακολουθίας)

C = πίνακας πιθανοτήτων viterbi (N, T)

πρόβλεψη= μέγεθος(N, T)

L = μέγεθος(A, 2)

αρχικοποίηση_ πιθανοτήτων= log2(αρχικοποίηση_πιθανοτήτων)

A = log2(A)

B = log2(B)

Για i από 1 μέχρι N

πίνακας_viterbi = αρχικοποίηση_πιθανότητας(i)

Τέλος_επανάληψης

vr = πίνακας_μηδενικών(T, N, 2)

τελικο_σκορ = πίνακας_μηδενικών (1, N)

Για t από 1 μέχρι T // μέγεθος ακολουθίας

Για i από 1 μέχρι N // πλήθος καταστάσεων

v = πίνακας_μηδενικών_διαστάσεων(1,N)

Για p από 1 μέχρι N

$v(p) = C(p,t)+A(p,i)+B(x(t),p)$ //πιθανότητα

να γίνει εκπομπή του συμβόλου B(x(t),p) βρισκόμενοι

στη κατάσταση με πιθανότητα C(p,t) μεταβάλλοντας

την κατάσταση με πιθανότητα A(p,i)

$vr(t,p,i) = v(p)$ // N-διάστατος πίνακας που

κρατάει όλες τις πιθανότητες που υπολογίζουμε για

κάθε κατάσταση

```

        Αν t = T //τελευταία κατάσταση end
            v(p) = C(p,t)+A(p,L)+B(x(t),p)
            τελικό_σκόρ(p) = v(p)
        Τέλος_επανάληψης
    Αν t != T
        [μεγιστη πιθανότητα, δείκτης] max(v)
        Προβλεπόμενο βήμα = δείκτης
    Τέλος_επανάληψης
Τέλος_επανάληψης

```

[μέγιστο_σکور τελευταίων στοιχειών πίνακα viterbi, δείκτης] = max(C(:,T+1))

```

μεγιστο= μέγιστο_σکور
πρόβλεψη(N, T) = δείκτης
t = T
καλύτερο_μονοπάτι = [δείκτης]
δευτερο_μονοπάτι = [δείκτης]
δεύτερος_δείκτης = δείκτης

```

```

Όσο t<> 0
    [σکور, δείκτης] = μέγιστο(πίνακα_νρ(t, :, δείκτης))
    καλύτερο_μονοπάτι = [καλύτερο_μονοπάτι, δείκτης]
    Αν υπάρχουν δύο ίσες πιθανότητες για μία μεταβολή
        ΤΟΤΕ
            Δευτερο_μονοπάτι = [δεύτερο μονοπάτι (κατασταση)2 ]
        Αλλιως
            [σکور, δεύτερος_δείκτης] = μέγιστο(πίνακα_νρ(t, :,
            δεύτερος_δείκτης))
            δεύτερο_μονοπάτι = [δευτερο_μονοπάτι,
            δεύτερος_δείκτης]

    t = t-1
    Αν t ==0
        Break

```

Τέλος_αν
Τέλος_επανάληψης
σκορ = μέγιστο_σκορ

7.3 Υλοποίηση

Ξεκινώντας από την εκφώνηση του βιβλίου κατασκευάζουμε τους πίνακες μεταβολής καταστάσεων και εκπομπής συμβόλων καθώς και το αλφάβητο. Έχοντας την ακολουθία των νούμερων από τη ρίψη ζαριών από την εκφώνηση της άσκησης. Έχοντας πια την ακολουθία την περνάμε ως όρισμα καλώντας την συνάρτηση `viterbi6` όπου θα μας επιστρέψει την μέγιστη πιθανότητα σε λογαριθμική βαθμολογία και την αντίστοιχες ακολουθίες καταστάσεων.

Η υλοποίηση της άσκησης είναι πολύ παρόμοια με την 11.4 αλλά η ουσιαστική διαφορά μεταξύ τους είναι ότι σε αυτή την άσκηση θέλουμε να βρούμε και τυχόν δεύτερο καλύτερο μονοπάτι (δηλαδή διακλάδωση βέλων οπισθοδρόμησης στο πίνακα `Viterbi`) καθώς και ότι η τελευταία κατάσταση είναι μια κατάσταση `end` σε αντίθεση με τη προηγούμενη άσκηση που ήταν απλώς το μέγιστο δύο καταστάσεων.

- Αλγόριθμος `viterbi6`

Η υλοποίηση του αλγόριθμου `Viterbi` σε αυτήν την άσκηση έγινε ως εξής. Πρώτα μετατρέπουμε όλες τις τιμές των πινάκων πιθανοτήτων σε λογαριθμικές τιμές για να μην υπάρχει `underflow` με τις κανονικές πιθανότητες που θα υπολογίσουμε. Μετά ξεκινάμε βάζοντας στην αρχή του πίνακα `Viterbi` τις αρχικές πιθανότητες των καταστάσεων μας και συνεχίζουμε για

κάθε σύμβολο της ακολουθίας τον υπολογισμό της πιθανότητας να εκπεμφθεί δοσμένης της τωρινής κατάστασης και της πιθανότητας της προηγούμενης έχοντας και «βέλη» οπισθοδρόμησης για την κατάσταση στην οποία καταλήγουμε κάθε φορά. Έχουμε επίσης δημιουργήσει και έναν N -διάστατο πίνακα που περιέχει σε κάθε διάσταση, (για κάθε κατάσταση αντίστοιχα) κάθε πιθανότητα που έχουμε υπολογίσει για τη μεταβολή από μία κατάσταση. Αν φτάσουμε στο τελευταίο σύμβολο όμως που σημαίνει ότι μετά είναι η κατάσταση end τότε αποθηκεύουμε ως τελική πιθανότητα την μέγιστη που θα βρούμε δοθείσης και της πιθανότητας για την μεταβολή στη κατάσταση end. Αφού υπολογίσουμε ολόκληρο το πίνακα Viterbi και βρούμε την μέγιστη πιθανότητα στο τέλος για την κατάσταση end ξεκινάμε το backtracking με τη βοήθεια των βέλων οπισθοδρόμησης. Εδώ σημαντικό είναι να εξηγήσουμε ότι το πρώτο βέλτιστο μονοπάτι υπολογίζεται όπως και στην 11.4 χρησιμοποιώντας απλά τη μέγιστη πιθανότητα που βρίσκουμε με τα βέλη οπισθοδρόμησης. Όμως για να βρούμε το δεύτερο μονοπάτι ελέγχουμε κάθε φορά αν υπάρχουν δυο μέγιστες πιθανότητες (δηλαδή ίσες) και ως αποτέλεσμα έχουμε δύο βέλη οπισθοδρόμησης (διακλάδωση). Αυτό γίνεται με τη βοήθεια του πίνακα vr που αναφέραμε πριν και έχει όλες τις υπολογισμένες πιθανότητες. Τέλος επιστρέφουμε και τα δύο μονοπάτια και τη μέγιστη πιθανότητα που βρήκαμε στη κατάσταση end.

7.4 Αποτελέσματα

Best score was: -16

Best path was: D1 D1 D1 D1 D2 D2

Best second path was: D1 D1 D2 D2 D2 D2

Άρα εδώ βλέπουμε ότι για την ακολουθία 1 1 2 1 2 2 στη ρίψη των ζαριών η καλύτερη πρώτη βέλτιστη ακολουθία καταστάσεων που δίνει μέγιστη πιθανότητα (λογαριθμική) -16 είναι η D1 D1 D1 D1 D2 D2, αλλά υπάρχει και μία δεύτερη βέλτιστη ακολουθία των ζαριών που είναι η . D1 D1 D2 D2 D2 D2 με ίδιο σκόρ εννοείται.

Από το αποτέλεσμα είναι ασφαλές να υποθέσουμε ότι οι ισοπίθανες πιθανότητες που προκάλεσαν τη διακλάδωση ήταν στη τρίτη εκπομπή συμβόλου δηλαδή του 2.

7.5

Βιβλιογραφία και βιβλιοθήκες που χρησιμοποιήθηκαν

Βιβλιοθήκες Matlab

Στη διάρκεια της υλοποίησης της άσκησης χρησιμοποιήθηκαν γνώσεις και μέθοδοι από διάφορες έτοιμες βιβλιοθήκες της Matlab που είναι επιστημονικού περιεχομένου και είναι φτιαγμένες για τέτοιου είδους προβλήματα.

Πιο συγκεκριμένα χρησιμοποιήθηκε το Bioinformatics Toolbox. (<https://www.mathworks.com/products/bioinfo/features.html - sequence-analysis>)

Το toolbox αυτό βοήθησε πολύ στην εύκολη ανάγνωση, μετατροπή και κωδικοποίηση νουκλετιδικών ακολουθιών καθώς και αμινοξέων.

8.2 Βιβλιογραφία

Είναι πολύ σημαντικό να σημειωθεί όμως ότι η έμπνευση και ο σχεδιασμός των λύσεων όλων των ασκήσεων έγινε βάση της θεωρίας και των παραδειγμάτων του βιβλίου καθώς και σημειώσεων κατά τη διάρκεια των διαλέξεων.

Πιο συγκεκριμένα για την υλοποίηση του αλγόριθμου Viterbi χρησιμοποιήθηκαν σημειώσεις από το προτελευταίο μάθημα που έγινε πρακτική ανάλυση του κώδικα σε Matlab.

- Για τις ασκήσεις 6.13 & 6.15 χρησιμοποιήθηκε από το βιβλίο η «Εισαγωγή» όπου βοήθησε στην κατανόηση σκέψης και επίλυση.

- Επίσης χρήσιμες βρήκαμε και τις σελίδες 69-73 του βιβλίου όπου μιλάει για τον δυναμικό προγραμματισμό.
- Για τις υλοποιήσεις των 6.22,6.23 και 6.37 χρησιμοποιήθηκε ψευδοκώδικας από τις σελίδες 204-211 του βιβλίου.

8.3 ΒΙΒΛΙΟΓΡΑΦΙΑ ΔΙΑΔΙΚΤΥΟΥ

Από το διαδίκτυο χρησιμοποιήθηκε η ιστοσελίδα του NCBI και RCSB για την εύρεση των ακολουθιών των νουκλεοτιδίων όπως εξηγείται στις ασκήσεις.

Επίσης χρησιμοποιήθηκε ψευδοκώδικας και ως ιδέα για την υλοποίηση του αλγόριθμου Needleman-Wunch το εξής επιστημονικό άρθρο από την ιστοσελίδα της Wikipedia (https://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm).