

# Βιοπληροφορική

Παναγιώτα Θωμοπούλου | Π14053

Δημήτρης Τζιλιβάκης | Π14179

## Εισαγωγή

Η γλώσσα που χρησιμοποιήθηκε ήταν η python και οι ασκήσεις που επιλέχθηκαν ήταν οι 6.12, 6.14, 6.15 και 6.37. Οι κατάλληλες αλληλουχίες για τις ασκήσεις βρίσκονται στους υποφακέλους 6.12-6.14Seq, 6.15Seq και 6.37Seq αντίστοιχα. Τα προγράμματα εμφανίζουν τον τρόπο νίκης ή λύσης των προβλημάτων με την λογική του δυναμικού προγραμματισμού. Στα παιχνίδια, θεωρούμε ότι ο Παίκτης A είναι ο παίκτης που παίζει πρώτος.

## 6.15

### ΔΥΝΑΜΙΚΗ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

Η άσκηση 6.15 περιγράφει ένα παιχνίδι με 2 παίκτες και μία νουκλεοτιδική αλληλουχία μήκους n. Κάθε παίκτης με την σειρά του αφαιρεί είτε ένα είτε δύο νουκλεοτίδια, και ο παίκτης που αφαιρεί το τελευταίο κερδίζει.

Για να λυθεί αυτό με δυναμικό προγραμματισμό, αρκεί να βρούμε τι θα γίνει αν υπάρχει ένα νουκλεοτίδιο στην ακολουθία. Προφανώς ο παίκτης A, θα αφαιρέσει το ένα νουκλεοτίδιο και θα κερδίσει.

Στην συνέχεια, ελέγχουμε για 2 νουκλεοτίδια. Ο παίκτης A μπορεί να αφαιρέσει και τα δύο και να κερδίσει, ή μπορεί να αφαιρέσει 1. Αν αφαιρέσει 1, τότε ξέρουμε ότι από τον προηγούμενο έλεγχο, ότι ο επόμενος παίκτης κερδίζει. Έτσι λοιπόν, αρκεί ο πρώτος παίκτης να αφαιρέσει 2 και θα κερδίσει.

Ελέγχοντας για 3, μπορούμε να αφαιρέσουμε είτε 1 είτε 2. Άρα η αλληλουχία θα περιέχει 2 ή 1. Από τους προηγούμενους ελέγχους, ξέρουμε ότι ο επόμενος παίκτης με 2 ή 1 θα κερδίσει. Άρα ο πρώτος παίκτης αναγκαστικά θα χάσει.

Για n νουκλεοτίδια, ελέγχουμε για n-1 και n-2. Αν κάποιο από τα δύο χάνει, τότε ο παίκτης A πρέπει να επιλέξει αυτό.

### ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Για την υλοποίηση της λύσης προβλήματος, αρχικά πρέπει να διαβαστεί η αλληλουχία. Η συνάρτηση readFile, διαβάζει την κατάλληλη αλληλουχία και την επιστρέφει ως λίστα.

```
def readFile():
    seqFile = open("./6.15Seq/α-lactalbumin.txt", "r")
    seq = []
    lines = seqFile.readlines()[1:]
    for line in lines:
        for letter in line:
            seq.append(letter)
```

```
seqFile.close()
return seq
```

Επίσης, τα αρχεία περιέχουν στην πρώτη γραμμή τους ένα σχόλιο με πληροφορίες σχετικές με την αλληλουχία, για αυτό πρέπει να αγνοηθεί η πρώτη γραμμή του αρχείου.

```
lines = seqFile.readlines()[1:]
```

Στην συνέχεια χρειαζόμαστε μία λίστα που περιέχει την πληροφορία για το αν κερδίζει η χάνει η κάθε ακολουθία.

```
canWin = []
```

Και μία λίστα για να αποθηκεύει τι αφαιρέθηκε σε κάθε βήμα για να κερδίσει ο κάθε παίκτης.

```
winningRemoval = []
```

*Η λίστα canWin περιέχει True ή False – True σε περίπτωση νίκης και False σε περίπτωση ήττας. Η winningRemoval περιέχει τους αριθμούς 1 ή 2.*

Τοποθετούμε στην θέση 0 της λίστας canWin την τιμή False. Στην συνέχεια ξεκινώντας από 1 μέχρι το n, ελέγχουμε αν το στοιχείο στην canWin είναι False. Αν είναι False, σημαίνει ότι επιλέγοντας το θα χάσει ο επόμενος παίκτης, άρα θα κερδίσουμε εμείς. Το επιλέγουμε και προσθέτουμε το αποτέλεσμα στην canWin. Επίσης ενημερώνουμε ότι για να κερδίσουμε πρέπει να αφαιρέσουμε 1.

```
if not(canWin[i-1]):
    canWin.append(True)
    winningRemoval.append(1)
```

Με τον ίδιο τρόπο ελέγχουμε αν για 2 στοιχεία μπορούμε να κερδίσουμε.

```
elif not(canWin[i-2]):
    canWin.append(True)
    winningRemoval.append(2)
```

Αν τελικά αφαιρώντας είτε 1 είτε 2 στοιχεία δεν μπορούμε να κερδίσουμε, αφαιρούμε ένα στοιχείο από την αλληλουχία για να συνεχίσει το παιχνίδι.

```
else:
    canWin.append(False)
    winningRemoval.append(1)
```

Για να διαπιστώσουμε αν ο Παίκτης Α κέρδισε, αρκεί να ελέγξουμε το τελευταίο στοιχείο της λίστας canWin. Αν είναι True, ο παίκτης έχει κερδίσει, αλλιώς έχει χάσει

```

if (canWin[-1]):
    print("Player A won.")
else:
    print("Player A lost.")

```

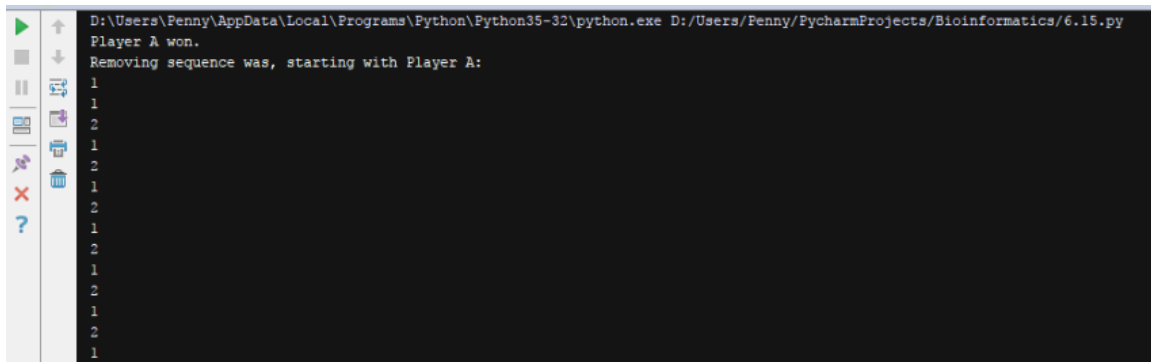
Τέλος, εμφανίζουμε τον τρόπο που πρέπει να αφαιρεθούν τα νουκλεοτίδια για να κερδίσει ο παίκτης.

```

print("Removing sequence was, starting with Player A:")
i = len(winningRemoval)-1
while(i>=0):
    print(winningRemoval[i])
    i -= winningRemoval[i]

```

## OUTPUT ΠΡΟΓΡΑΜΜΑΤΟΣ



```

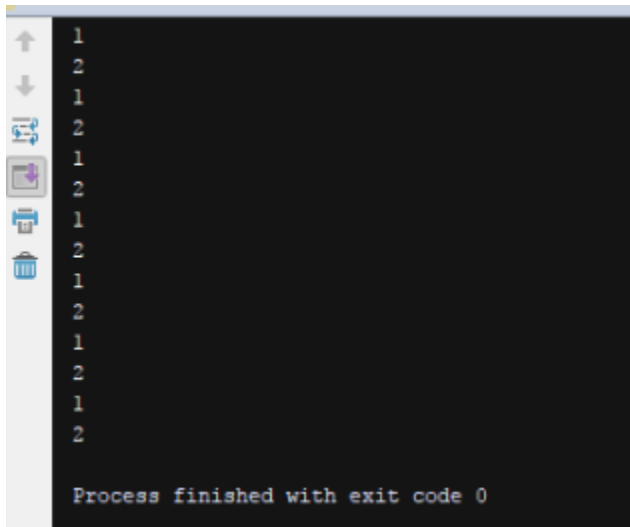
D:\Users\Penny\AppData\Local\Programs\Python\Python35-32\python.exe D:/Users/Penny/PycharmProjects/Bioinformatics/6.15.py
Player A won.
Removing sequence was, starting with Player A:
1
1
2
1
2
1
2
1
2
1
2
1
2
1
2
1
1

```

```

Length of sequence: 1 Can win: True
Length of sequence: 2 Can win: True
Length of sequence: 3 Can win: False
Length of sequence: 4 Can win: True
Length of sequence: 5 Can win: True
Length of sequence: 6 Can win: False
Length of sequence: 7 Can win: True
Length of sequence: 8 Can win: True
Length of sequence: 9 Can win: False
Length of sequence: 10 Can win: True
Length of sequence: 11 Can win: True
Length of sequence: 12 Can win: False
Length of sequence: 13 Can win: True
Length of sequence: 14 Can win: True
Length of sequence: 15 Can win: False
Length of sequence: 16 Can win: True
Length of sequence: 17 Can win: True

```



Για μήκος 4, ο παίκτης Α έχει την επιλογή να χωρίσει σε 1 και 3 ή 2 και 2. Αν χωρίσει σε 1 και 3, από τα προηγούμενα συμπεράσματα ξέρουμε ότι θα χάσει, ενώ αν επιλέξει το 2 και 2 θα κερδίσει. Ο παίκτης Α θα επιλέξει το 2 και 2 άρα θα κερδίσει.

Αρκεί να συνεχίσουμε την διαδικασία για την αλληλουχία με μεγαλύτερο μήκος.

## ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Αρχικά, πρέπει να διαβάσουμε και να αποθηκεύσουμε τις δύο αλληλουχίες της επιλογής μας.

```
def fillSeq(selectedSeqName):
    seqFile = open("./6.12-6.14Seq/" + selectedSeqName + ".txt", "r")
    seq = []
    lines = seqFile.readlines()[1:]
    for line in lines:
        for letter in line:
            seq.append(letter)
    seqFile.close()
    return seq

seq1 = fillSeq(input("Select first sequence to use (brain, liver or muscle)"))
seq2 = fillSeq(input("Select second sequence to use (brain, liver or muscle)"))
```

Ο τρόπος που διαβάζονται οι αλληλουχίες είναι παρόμοιος με την άσκηση 6.15.

Τοποθετούμε την μεγαλύτερη αλληλουχία στην λίστα seq1 και την μικρότερη στην seq2:

```
if len(seq2) > len(seq1):
    temp = seq1
    seq1 = seq2
    seq2 = temp
```

Αρχικοποιούμε τις λίστες canWin και winningRemoval για να αποθηκεύσουμε ποια αλληλουχία κερδίζει και πως πρέπει να χωριστούν για να κερδίσουν.

```
canWin = []
winningRemoval = []
canWin.append(False)
canWin.append(True)
```

Τοποθετήσαμε στην θέση 0 το False, γιατί αν η αλληλουχία έχει 0 στοιχεία έχουμε ήδη χάσει και στην θέση 1 το True γιατί ο σκοπός του παιχνιδιού είναι να αφαιρέσουμε το τελευταίο στοιχείο.

Πρέπει να εξετάσουμε όλες τις περιπτώσεις ξεκινώντας από το μήκος 2 μέχρι το μήκος της μεγαλύτερης ακολουθίας. Επιπλέον πρέπει να εξετάσουμε όλους τους πιθανούς χωρισμούς της αλληλουχίας. Πχ για 6, πρέπει να εξεταστούν οι περιπτώσεις [1,5], [2,4], [3,3].

```
for i in range(2, len(seq1)+1):
    for j in range(1, int((i/2)+1)):
```

Στην συνέχεια, ελέγχουμε και οι δύο αλληλουχίες σε κάθε περίπτωση επιστρέφουν False. Αν επιστρέφουν False, σημαίνει ότι ο επόμενος παίκτης θα χάσει ότι και να επιλέξει, άρα εμείς θα κερδίσουμε. Αν, λοιπόν, συμβεί αυτό, αποθηκεύουμε στην λίστα winningRemoval το σημείο που χωρίστηκε. Επιπλέον, ενημερώνουμε στην λίστα canWin ότι στο σημείο αυτό ο παίκτης μπορεί να κερδίσει.

```
if not(canWin[j]) and not(canWin[i-j]):
    canWin.append(True)
    winningSplit = []
    winningSplit.append(j)
    winningSplit.append(i-j)
    winningRemoval.append(winningSplit)
    break
```

Αν ολοκληρωθεί το εμφωλευμένο loop χωρίς να συναντήσει τον «χωρισμό» που κερδίζει, η θα τοποθετηθεί στην λίστα canWin False.

```
else:
    canWin.append(False)
```

*Η Python υποστηρίζει την μορφή for-else. Το συγκεκριμένο else δεν ανήκει αλλά στο if, αλλά στο for. « Loop statements may have an else clause; it is executed when the loop terminates through exhaustion of the list (with for) or when the condition becomes false (with while), but not when the loop is terminated by a break statement. When used with a loop, the else clause has more in common with the else clause of a try statement than it does that of if statements: a try statement's else clause runs when no exception occurs, and a loop's else clause runs when no break occurs.<sup>i</sup> »*

Όταν τελειώσει η επανάληψη, ελέγχουμε τα στοιχεία στις θέσεις μήκους των δύο αλληλουχιών. Αν κάποια από τις δύο είναι True, ο παίκτης A έχει κερδίσει.

```
if (canWin[-1] or canWin[len(seq2)]):
    print("Player A won.")
else:
    print("Player A lost.")
```

---

<sup>i</sup> [Python Documentation](#)

Τέλος, εμφανίζουμε τους χωρισμούς που επιτρέπουν να κερδίσουν οι παίκτες.

```
print("Winning splits were:")
for i in winningRemoval:
    print(i)
```

## OUTPUT ΠΡΟΓΡΑΜΜΑΤΟΣ

```
D:\Users\Fenny\AppData\Local\Programs\Python\Python36\python.exe D:/Users/Fenny/PycharmProjects/Bioinformatics/6.12.py
Select first sequence to use (brain, liver or muscle)brain
Select second sequence to use (brain, liver or muscle)liver
Player A won.
Winning splits were:
[2, 2]
[2, 3]
[3, 3]
[2, 7]
[2, 8]
[3, 8]
[2, 12]
[2, 13]
[3, 13]
[2, 17]
[2, 18]
[3, 18]
```

```
[2, 50633]
[3, 50633]
[2, 50637]
[2, 50638]
[3, 50638]
[2, 50642]
[2, 50643]
[3, 50643]
[2, 50647]
[2, 50648]
[3, 50648]
[2, 50652]
[2, 50653]
[3, 50653]

Process finished with exit code 0
```

## ΝΙΚΗΦΟΡΑ ΣΤΡΑΤΗΓΙΚΗ

Από το output συμπεραίνουμε ότι για να κερδίσει ένας παίκτης αρκεί να χωρίζει σε την αλληλουχία με τρόπο έτσι ώστε ένα από τα δύο μέρη να έχει μήκος 2 ή 3.

## 6.14

### ΔΥΝΑΜΙΚΗ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

Το συγκεκριμένο παιχνίδι παίζεται με 2 παίκτες και 2 αλληλουχίες νουκλεοτιδίων. Κάθε παίκτης διαγράφει ένα στοιχείο από την μία αλληλουχία και 2 από την άλλη. Νικητής είναι αυτός που δεν μπορεί να κάνει άλλη κίνηση.



---

Για την επίλυση αυτού του προβλήματος, όπως και στα δύο προηγούμενα προβλήματα, αρκεί να ξεκινήσουμε από την κατάσταση [0,0] και να βρίσκουμε τις επόμενες σύμφωνα με τις προηγούμενες.

Αν ο παίκτης A ξεκινήσει με 2 αλληλουχίες μήκους 0, δεν μπορεί να κάνει κάποια κίνηση άρα κερδίζει. Επομένως, για [0,0] ο παίκτης κερδίζει.

Για τις αλληλουχίες [0,1] επίσης κερδίζει.

Για τις αλληλουχίες [1,3] μπορεί να αφαιρέσει ένα από την πρώτη και 2 από την δεύτερη. Άρα ο επόμενος παίκτης θα έχει την ακολουθία [0,1], οπου ξέρουμε ότι κερδίζει από το προηγούμενο συμπέρασμα, άρα ο παίκτης A χάνει.

Υπολογίζουμε με τον ίδιο τρόπο για [n, m].

## ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Για αυτό το πρόβλημα, χρησιμοποιήθηκαν arrays της βιβλιοθήκης NumPy

```
import numpy as np
```

Αποθηκεύτηκαν οι αλληλουχίες όπως στα προηγούμενα προβλήματα.

```
def fillSeq(selectedSeqName):
    seqFile = open("./6.12-6.14Seq/" + selectedSeqName + ".txt", "r")
    seq = []
    lines = seqFile.readlines()[1:]
    for line in lines:
        for letter in line:
            seq.append(letter)
    seqFile.close()
    return seq

seq1 = fillSeq(input("Select first sequence to use (brain, liver or muscle)"))
seq2 = fillSeq(input("Select second sequence to use (brain, liver or muscle)"))
```

Δημιουργήθηκε το array canWin που περιέχει True αν ο παίκτης κερδίζει με τα συγκεκριμένα μήκη αλληλουχιών και False για όταν δεν κερδίζει. Αρχικοποιήθηκε με True.

Επιπλέον τα array wasRemovedFromI και wasRemovedFromJ για να αποθηκεύουμε αν από την πρώτη ή την δεύτερη αντίστοιχα αλληλουχία αφαιρέσαμε το 1 ή το 2.

```
canWin = np.ones((len(seq1)+1, len(seq2)+1), dtype=bool)
wasRemovedFromI = np.zeros((len(seq1) + 1, len(seq2) + 1), dtype=int)
wasRemovedFromJ = np.zeros((len(seq1) + 1, len(seq2) + 1), dtype=int)
```

---

Ξεκινώντας από το 0 μέχρι το μήκος των αλληλουχιών ελέγχουμε αν μπορούμε να αφαιρέσουμε στοιχεία.

```
for i in range(len(seq1)+1):
    for j in range(len(seq2)+1):
        if (i-1>=0 and j-2>=0) or (i>=2 and j>=1):
```

Αν δεν μπορούμε να αφαιρέσουμε σημαίνει ότι ο παίκτης κέρδισε.

```
else:
    canWin[i][j] = True
```

Αν μπορούμε να αφαιρέσουμε, ελέγχουμε αν γίνεται να αφαιρέσουμε 1 από την πρώτη και 2 από την δεύτερη,

```
if i-1>=0 and j-2>=0:
```

και ελέγχουμε αν ο επόμενος παίκτης μπορεί να κερδίσει.

```
if canWin[i-1][j-2]:
```

Αν μπορεί να κερδίσει, σημαίνει ότι ο παίκτης Α χάνει, αλλιώς κερδίζει. Επιπλέον ενημερώνουμε τα array wasRemovedFromI και wasRemovedFromJ

```
canWin[i][j] = False
else:
    canWin[i][j] = True
wasRemovedFromI[i][j] = -1
wasRemovedFromJ[i][j] = -2
```

Μετά ελέγχουμε για την αντίθετη περίπτωση. Δηλαδή αν αφαιρέσουμε από την πρώτη 2 και από την δεύτερη ένα.

```
if i>=2 and j>=1:
    if not(canWin[i][j]):
        if canWin[i-2][j-1]:
            canWin[i][j] = False
        else:
            canWin[i][j] = True
            wasRemovedFromI[i][j] = -2
            wasRemovedFromJ[i][j] = -1
```

Αν το τελευταίο στοιχείο του array canWin είναι True, ο παίκτης Α κέρδισε.

```
if canWin[-1][-1]:
    print("Player A won.")
else:
    print("Player A lost.")
```

Τέλος εμφανίζουμε με ποιες αφαιρέσεις κερδίζει ο παίκτης Α.

```

for i in range(len(seq1)+1):
    for j in range(len(seq2)+1):
        print "["+str(i)+", "+str(j)+"]" +str(canWin[i][j]))
        print "["+str(wasRemovedFromI[i][j])+",
"+str(wasRemovedFromJ[i][j])+"]"

```

## OUTPUT ΠΡΟΓΡΑΜΜΑΤΟΣ

### ΝΙΚΗΦΟΡΑ ΣΤΡΑΤΙΓΙΚΗ

Παρατηρούμε ότι για να κερδίσει ένας παίκτης, έχοντας δύο αλληλουχίες μήκους  $n$  και  $m$  αντίστοιχα, για  $m$  μεγαλύτερο του 2, αν το  $n$  είναι περιττός αριθμός πρέπει να αφαιρέσει από το  $n-2$  και από το  $m-1$ , ενώ για άρτιο μήκος  $n$  το αντίστροφο. Αν το  $m$  είναι 0, ο παίκτης Α κερδίζει επειδή δεν μπορεί να κάνει κάποια κίνηση. Για  $m$  ίσο με 2, αν το  $n$  είναι άρτιος κερδίζει με την παραπάνω λογική και για  $m$  ίσο με 1, χάνει πάντα, εκτός από την περίπτωση που το  $n$  είναι μικρότερο του 2, αφού δεν μπορεί να κάνει κάποια άλλη κίνηση.

## 6.37

### ΔΥΝΑΜΙΚΗ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

Το πρόβλημα 6.37 ζητάει να επινοήσουμε ένα αποδοτικό αλγόριθμο για το πρόβλημα της Χιμαιρικής Στοίχισης.

Για να το λύσουμε, πρέπει να ελέγχουμε κάθε φορά με το προηγούμενο νουκλεοτίδιο μέχρι να φτάσουμε το όριο του μήκους ή μέχρι να βρούμε ένα διαφορετικό νουκλεοτίδιο.

Για παράδειγμα, αν έχουμε μια αλληλουχία AAAAAAAG, τότε ξεκινάμε από το πρώτο νουκλεοτίδιο, ελέγχουμε το επόμενο νουκλεοτίδιο το οποίο είναι A, άρα συνεχίζουμε μέχρι το 5<sup>ο</sup>, και επειδή φτάσαμε το όριο μήκους, στην σωστή αλληλουχία βάζουμε ένα A. Μετά συνεχίζουμε την αναζήτηση, και ελέγχουμε το 6<sup>ο</sup> A και μετά το 7<sup>ο</sup>. Όταν πάμε στο G, επειδή το G είναι διαφορετικό από το A, το προσθέτουμε στην σωστή αλληλουχία. Τέλος, προσθέτουμε το G. Άρα η σωστή αλληλουχία είναι AAG.

### ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Αρχικά, διαβάζουμε και αποθηκεύουμε την ακολουθία.

```

seqFile = open("./6.37Seq/nucleotide.txt", "r")
seq = []
lines = seqFile.readlines()[1:]
for line in lines:

```

```
for letter in line:
    seq.append(letter)
seqFile.close()
```

Αρχικοποιούμε τις μεταβλητές που ελέγχουν αν ένα νουκλεοτίδιο ξεπέρασε το όριο μήκους.

```
ACount = 0
CCount = 0
```

*Το όριο για το A είναι 5, ενώ το όριο για C 10*

Για να βρούμε αν πρέπει να προσθέσουμε κάτι στην σωστή αλληλουχία, χρησιμοποιούμε την συνάρτηση `check_seq`:

```
def check_seq(sequence, position, ACount, CCount):
    if sequence[position]=='A':
        ACount+=1
        CCount = 0
        if ACount == 5:
            ACount = 0
            return True
        elif sequence[position] != sequence[position+1]:
            return True
    elif sequence[position]=='C':
        CCount+=1
        ACount = 0
        if CCount == 10:
            CCount = 0
            return True
        elif sequence[position] != sequence[position+1]:
            return True
    else:
        CCount = 0
        ACount = 0
        if sequence[position] != sequence[position+1]:
            return True
    return False
```

Η συνάρτηση αυτή ελέγχει αρχικά, ποιο είναι το νουκλεοτίδιο της θέσης που του δώσαμε και αν ξεπέρασε το όριο. Αν το ξεπέρασε, επιστρέφει True, δηλαδή πρέπει να το προσθέσουμε στην σωστή αλληλουχία. Μετά ελέγχει αν είναι διαφορετικό από το νουκλεοτίδιο της προηγούμενης θέσης. Αν το νουκλεοτίδιο είναι διαφορετικό, επιστρέφει True. Επίσης, κάθε φορά που το νουκλεοτίδιο δεν είναι A, ο μετρητής του A μηδενίζεται, ενώ για C μηδενίζεται όταν το νουκλεοτίδιο δεν είναι C. Αυτό το κάνουμε επειδή το όριο που δεν πρέπει να ξεπεράσουν είναι μόνο όταν τα νουκλεοτίδια είναι συνεχόμενα ίδια.

Προσθέτουμε στην καινούργια αλληλουχία όσα νουκλεοτίδια επιστρέφουν True:

```
for i in range(len(seq)-1):
    if (check_seq(seq,i,ACount,CCount)):
        newSeq.append(seq[i])
```

Και για το τελευταίο νουκλεοτίδιο:

```
if (check_seq(seq, len(seq)-2, ACount, CCount)) :
    newSeq.append(seq[-1])
```

Τέλος, εμφανίζουμε την σωστή αλληλουχία:

```
print(newSeq)
```

## OUTPUT ΠΡΟΓΡΑΜΜΑΤΟΣ

```
D:\Users\Penny\AppData\Local\Programs\Python\Python36\python.exe D:/Users/Penny/PycharmProjects/Bioinformatics/6.37.py
['G', 'A', 'G', 'C', 'A', 'G', 'C', 'A', 'G', 'T', 'C', 'G', 'T', 'C', 'A', 'C', 'A', 'G', 'C', 'A', 'C', 'A', 'G', 'T', 'C', 'T']
Process finished with exit code 0
```

Το output του προγράμματος είναι προφανώς πολύ μεγαλύτερο. Αυτό είναι ενδεικτικά ένα μέρος της αλληλουχίας.