Ειδικά Θέματα Βιοπληροφορικής

Διάλεξη 8η :

Perl (6) : Ένα παράδειγμα ...

Το πρόβλημα

Δίδεται ένα αρχείο το οποίο περιέχει αλληλουχίες σε FASTA format. Γράψτε ένα πρόγραμμα σε perl το οποίο θα βρίσκει και θα τυπώνει στην καθιερωμένη έξοδο την μεγαλυτέρου μήκους κοινή υπακολουθία αυτών των αλληλουχιών καθώς και τη θέση (σε κατάλοιπα) αυτής της υπακολουθίας σε κάθε μία από τις αλληλουχίες.

Τα δεδομένα εισόδου

```
# cat anchor.dat
>gi|442541|pdb|102L|  Lysozyme Insertion Mutant With Ala Inserted After ...
N40-A), C54t,C97a)
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGIL
RNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWD
AYKNL
>gi|442542|pdb|103L|  Phage T4 Lysozyme Insertion Mutant With Ser, Leu, ...
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNSLDAAKSELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRG
ILRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGT
WDAYKNL
>gi|442544|pdb|104L|B Chain B, Lysozyme Insertion Mutant With Ala, Ala ...
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKSAAELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGI
LRNAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTW
DAYKNL
>gi|442545|pdb|107L|  Lysozyme (E.C.3.2.1.17) Mutant With Ser 44 Replaced By ...
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKGELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGILR
NAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWDA
YKNL
>gi|442546|pdb|108L|  Lysozyme (E.C.3.2.1.17) Mutant With Ser 44 Replaced By ...
MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKIELDKAIGRNTNGVITKDEAEKLFNQDVDAAVRGILR
NAKLKPVYDSLDAVRRAALINMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVITTFRTGTWDA
YKNL
>gi|442547|pdb|109L|  Lysozyme (E.C.3.2.1.17) Mutant With Ser 44 Replaced By ...
....
```

Παράδειγμα χρήσης

```
# ./anchor.pl anchor.dat
Longest match was LINMVFQMGETGVAGF found at :
100      gi|442541|pdb|102L|  Lysozyme Insertion Mutant Wit...
102      gi|442542|pdb|103L|  Phage T4 Lysozyme Insertion M...
101      gi|442544|pdb|104L|B Chain B, Lysozyme Insertion M...
99       gi|442545|pdb|107L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442546|pdb|108L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442547|pdb|109L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442548|pdb|110L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442551|pdb|111L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442554|pdb|112L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442557|pdb|113L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442560|pdb|114L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442563|pdb|115L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442568|pdb|118L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442571|pdb|119L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442572|pdb|120L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442575|pdb|122L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442578|pdb|123L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442581|pdb|125L|  Lysozyme (E.C.3.2.1.17) Mutan...
99       gi|442584|pdb|126L|  Lysozyme (E.C.3.2.1.17) Mutan...
....
```

Σχεδιασμός δομής δεδομένων

Θα μπορούσαμε να χρησιμοποιήσουμε hashes όπου τα 'κλειδιά' θα ήταν οι κωδικοί των αλληλουχιών, ενώ οι 'τιμές' θα ήταν οι αλληλουχίες. Θα το κρατήσουμε απλούστερο: το πρόγραμμα μας θα χρησιμοποιεί δύο διακριτά arrays. Το array με το όνομα @seqs θα περιέχει τις διαδοχικές αλληλουχίες ενώ το array με το όνομα @names θα περιέχει τα ονόματα (κωδικούς) αυτών των αλληλουχιών. Διαδοχικές θέσεις (ξεκινώντας από τη θέση με δείκτη μηδέν) αυτών των arrays θα περιέχουν διαδοχικές αλληλουχίες με τη σειρά που υπάρχουν στο αρχείο εισόδου. Έτσι :

Prelude ...

```
#!/usr/bin/perl -w

(@ARGV == 1) or die "Usage: anchor.pl <FASTA file>\n";

open( SEQS, $ARGV[0] ) or die "Can not open $ARGV[0]\n";
```

Ανάγνωση δεδομένων

```
$current = "";
$pointer = 0;
while ( $line = <SEQS> )
  {
    if ( $line =~ /^\>(.*)$/ )
      {
        if ( defined( $identifier ) )
          {
            $seqs[ $pointer ] = $current;
            $names[ $pointer ] = $identifier;
            $pointer++;
          }
        $identifier = $1;
        $current = "";
      }
    else
      {
        chomp( $line );
        $current = $current . $line;
      }
  }
```

Ανάγνωση δεδομένων

Ο κώδικας της προηγούμενης διαφάνειας αποτυγχάνει να αποθηκεύσει την τελευταία αλληλουχία (γιατί ;). Με μία μικρή προσθήκη η ανάγνωση ολοκληρώνεται :

```
...
        chomp( $line );
        $current = $current . $line;
      }
  }

if ( defined( $identifier ) )
  {
    $seqs[ $pointer ] = $current;
    $names[ $pointer ] = $identifier;
    $pointer++;
  }
```

Μέθοδος επίλυσης

Για κάθε μήκους υπακολουθία της μικρότερης αλληλουχίας (γιατί της μικρότερης ;), από τη μεγαλύτερη δυνατή (ίση, δηλαδή, με το μήκος της μικρότερης αλληλουχίας) έως και τους δύο χαρακτήρες, ερεύνησε [χρησιμοποιώντας την index()] την παρουσία της υπακολουθίας σε όλες τις αλληλουχίες εισόδου. Εάν η υπακολουθία υφίσταται σε όλες τις αλληλουχίες εισόδου, τύπωσε (στην καθιερωμένη έξοδο) τις θέσεις της.

Εύρεση μικρότερης αλληλουχίας

```
...
# Do I have enough sequences ?
$nofseqs = $pointer;
( $nofseqs > 1 ) or die "No sequences found ?\n";

# Which is the shortest sequence ? (result in $target)
$minlen = length( $seqs[0] ) + 1;
for ( $i=0 ; $i < $nofseqs ; $i++ )
  {
  if ( length( $seqs[$i] ) < $minlen )
    {
      $minlen = length( $seqs[$i] );
      $target = $i;
    }
  }
```

Το κυρίως πρόγραμμα

```
# For each length, position, and sequence ...
for ( $length = length( $seqs[$target] ) ; $length >= 2 ; $length-- )
  {
    for ( $pos = 0 ; $pos <= length( $seqs[$target] ) - $length ; $pos++ )
      {
        $tolook = substr( $seqs[$target], $pos, $length );
        for ( $seq = 0 ; $seq < $nofseqs ; $seq++ )
          {
            if ( index( $seqs[ $seq ], $tolook, 0 ) == -1 )
              {
                last;
              }
          }

        # ... has a match been found ?
        if ( $seq == $nofseqs )
          {
            # MATCH FOUND
          }
      }
  }
```

Έξοδος

```
...
# ... has a match been found ?
if ( $seq == $nofseqs )
 {
   print "Longest match was ", substr( $seqs[$target], $pos, $length );
   print " found at : \n";
   for ( $seq = 0 ; $seq < $nofseqs ; $seq++ )
     {
       print index( $seqs[ $seq ], $tolook, 0 )+1;
       print "\t $names[$seq]\t", "\n";
     }
  print "\n";
  exit( 1 );
 }
```

ΑΣΚΗΣΗ

1. Εάν θέλαμε να βρίσκουμε όχι μόνο τη μεγαλύτερη κοινή υπακολουθία, αλλά όλες τις πιθανές υπακολουθίες, πως θα έπρεπε να τροποποιήσουμε το πρόγραμμα ;

2. Το πρόγραμμα αυτό έχει μία σημαντική αδυναμία: Εάν κάποια ή κάποιες αλληλουχίες περιέχουν την υπακολουθία σε δυο ή περισσότερες θέσεις, το πρόγραμμα μας θα εντοπίσει μόνο την πρώτη από αυτές. Πως θα έπρεπε να τροποποιήσουμε το πρόγραμμα ώστε να βρίσκει όλες τις παρουσίες (σε μία αλληλουχία) κάποιας υπακολουθίας ;