Homework 4

See Blackboard for assignment and due dates

Please enter your code and responses below and save the results to submit through Blackboard. Your code should follow the guidelines laid out in class, including commenting. Partial credit will be given for nonfunctional code that is logical and well commented. **This assignment must be completed on your own**.

Problem 4.1 (50%):

In this problem you will implement aspects of the LCS algorithm discussed in **Lecture 3**. Create a script in the box below. This script must contain the two functions described below, plus any additional code you need to run the script.

- `compute_lcs` : this function takes as input two sequences and `returns` (not prints) the LCS matrix.
- `print_lcs` : this function takes as input the same two sequences as well as the computed LCS matrix (from `compute_lcs`) and prints the LCS Plot. For full credit, `print_lcs` should:
  - print the two sequences one below the other
  - display every row of the LCS matrix on a separate line.
  - **Extra credit**: print the sequence characters next to their associated rows and columns as seen in the example below.

Example output: Sequence A = TACGCTGGA Sequence B = AACTGGCAG A A C T G G C A G T 0 0 0 1 0 0 0 0 0 A 1 1 0 0 0 0 0 1 0 C 0 0 2 0 0 0 1 0 0 G 0 0 0 0 1 1 0 0 1 C 0 0 1 0 0 0 2 0 0 T 0 0 0 2 0 0 0 0 0 G 0 0 0 0 3 1 0 0 1 G 0 0 0 0 1 4 0 0 1 A 1 1 0 0 0 0 0 1 0

In [1]:
```
#write your python code here

seq_A = 'TACGCTGGA'
seq_B = 'AACTGGCAG'

# this function takes as input two sequences and returns (not prints) the LCS matrix.
def compute_lcs(seq_A,seq_B):
    m = len(seq_A)
    n = len(seq_B)

    #create an empty lcs matrix
    lcs_matrix =[]
    row=[] #create an empty list to store rows of lcs matrix

    #filling the lcs matrix with 0s
    for i in range(m):
        for j in range(n):
            row.append(0)
        lcs_matrix.append(row)
        row=[]

    #filling the lcs matrix with values
    for i in range(m):
        for j in range(n):
            if seq_A[i]==seq_B[j]:
                lcs_matrix[i][j]=lcs_matrix[i-1][j-1]+1

    return lcs_matrix
```

In [2]:
```
compute_lcs(seq_A,seq_B)
```

Out[2]:
```
[[0, 0, 0, 1, 0, 0, 0, 0, 0],
 [1, 1, 0, 0, 0, 0, 0, 1, 0],
 [0, 0, 2, 0, 0, 0, 1, 0, 0],
 [0, 0, 0, 0, 1, 1, 0, 0, 1],
 [0, 0, 1, 0, 0, 0, 2, 0, 0],
 [0, 0, 0, 2, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 3, 1, 0, 0, 1],
 [0, 0, 0, 0, 1, 4, 0, 0, 1],
 [2, 1, 0, 0, 0, 0, 0, 1, 0]]
```

In [2]:
```
#this function takes as input the same two sequences as well as the computed LCS matrix
#(from compute_lcs) and prints the LCS Plot
def print_lcs(seq_A,seq_B):
    #first, print two seqences out
    print("Sequence A =",seq_A+'\n'+"Sequence B =",seq_B)

    #get the lcs matrix from the compute_lcs function
    matrix = compute_lcs(seq_A,seq_B)
    m = len(seq_A)
    n = len(seq_B)

    #update the matrix to get a nicer output format
    #print sequence B as the line header of the output
    sep = " "
    headerline=sep*2 #add two space in front
    for ele in seq_B:
        headerline+= ele + sep # add one space between nucleotides in sequence
```

```
    print(headerline)

    #print the rest lines
    rowStr=''
    for i in range(m):
            rowStr+= seq_A[i]+sep #add i-th nucleotide of sequence A at the first place of each line
            for j in range(n):
                rowStr+= str(matrix[i][j])+sep #add matrix values
            print(rowStr)
            rowStr = ''
#easier way: use pandas.DataFrame(matrix,index=list(seq_A), columns=list(seq_B)) to print out the matrix
print_lcs(seq_A,seq_B)
```

```
Sequence A = TACGCTGGA
Sequence B = AACTGGCAG
  A A C T G G C A G
T 0 0 0 1 0 0 0 0 0
A 1 1 0 0 0 0 0 1 0
C 0 0 2 0 0 0 1 0 0
G 0 0 0 0 1 1 0 0 1
C 0 0 1 0 0 0 2 0 0
T 0 0 0 2 0 0 0 0 0
G 0 0 0 0 3 1 0 0 1
G 0 0 0 0 1 4 0 0 1
A 2 1 0 0 0 0 0 1 0
```

Problem 4.2 (40%):

Detecting orthologous genes between two genomes is an important task in bioinformatics. BLAST can be used as a simple ortholog detector. We first perform a BLAST search of a given protein from Genome 1 (protein A) against Genome 2. We then take the best matching protein from Genome 2 (protein B) and perform a BLAST search against Genome 1. If the best matching protein from Genome 1 is protein A, then we conclude that protein A and protein B are orthologous (i.e., the genes that encode them are orthologs). This is called the *reciprocal best hit strategy*.

(A) The following protein is derived from the genome of *Sacchromyces cerevisiae* (Taxonomic ID: 4932) (baker's yeast). Using BLAST, perform a reciprocal best hit search against:

- Candida albicans (5476)
- Haloquadratum walsbyi (293091)
- Homo sapiens (9606)

At each step describe the best hit you discover and comment on its degree of similarity to your query protein. If an orthology relationship is found, state it; if not, suggest a biological explanation.

>gi|170986|gb|AAA34391.1| actin [Saccharomyces cerevisiae]
MDSEVAALVIDNGSGMCKAGFAGDDAPRAVFPSIVGRPRHQGIMVGMGQKDSYVGDEAQSKRGILTLRYPIEHGIVTNWDDMEKIWHHTFYNELRVAPEEHPVLLTEAPMNPKSNREKMTQ
albicans (5476) There is a best matching gene actin [Candida albicans WO-1] Sequence ID: EEQ41895.1 from Candida albicans (5476): with a Score = 751 bits(1938), E-value=0, Identities =355/375(95%), Positives=370/375(98%), and Gaps=0/375(0%). As the E-value≤1e-03, Percent of identity ≥ 25% , it is a significant hit. Using reciprocal best hit strategy, the best matching gene actin [Candida albicans WO-1]from Sacchromyces cerevisiae (Taxonomic ID: 4932) is our query protein,so they are likely to be orthologous. 2.Haloquadratum walsbyi (293091) There are no significant similarity found from Haloquadratum walsbyi (293091) genome. This may because actin is found in essentially all eukaryotic cells and Haloquadratum walsbyi are prokaryotes. 3.Homo sapiens (9606) There best matching of query protein from Homo sapiens is actin, cytoplasmic 2 [Homo sapiens], Sequence ID: NP_001186883.1. Score= 721 bits(1862),Expect=0,Identities=334/375(89%),Positives=360/375(96%),Gaps=0/375(0%) Using reciprocal best hit strategy, the best matching gene is also our query protein. This suggests that these two proteins are orthologous, too.

(B) Describe a BLAST-based procedure (either in a paragraph or with pseudocode) for determining all orthology relationships between a pair of genomes. You may assume that the genomes have been provided in FASTA format with one protein per entry. You DO NOT need to implement your procedure in python.

Readin genomes fasta files, and save them as two txt files genome_A and genome_B, separately. Close original genomes fasta files. Open a file A_against_B to save siganificant hits of sequences from genome_A against genome_B. Open a file B_against_A to save siganificant hits of sequences from genome_B against genome_A. Open a file orthology_AB to store all found orthology relationships between genome A and B. For entry in genome_A, do: set entry as a Query sequence; BLAST Query against genome_B; If there are siganificant hits: save Subject seqences into A_against_B; Find best_matching in A_against_B: BLAST best_matching against genome_A; If there are siganificant hits; save hits sequneces to B_against_A If best_hits in B_against_A is entry(original Query sequence from genome_A): save it to orthology_AB. End If End If End If End for Close file orthology_AB.

Problem 4.3 (10%):

Thought question: how could you use your python script from part 4.1 and python's random library to estimate the probability of observing an LCS of length 4 between two random DNA sequences of length 9? How could you estimate the statistical significance (p-value) of this occurrence?

In [46]:
```
import random

def prob(seq_A,seq_B):

    SampleSize = 1000 # do LCS N=1000 times
    count = 0 #how many LCS of length of 4 would happen in our sample size=1000
    max_s=0 # variable to store current LCS score

    for n in range(SampleSize):
        #store 4 kinds of nucleotides in a list
        ns =['A','T','G','C']
```

```
        #generate two random DNA sequences of length 9
        seq_A = ''.join(random.choices(ns, k=9))
        seq_B = ''.join(random.choices(ns, k=9))

        #excute lcs algorithm
        lcs = compute_lcs(seq_A,seq_B)
        for i in range(len(seq_A)):
            for j in range(len(seq_B)):
                if lcs[i][j] >max_s:
                    max_s = lcs[i][j] # LCS of each case

        if max_s == 4: # if LCS = 4
            count +=1 #add it to the LCS count
            max_s = 0
        else:
            max_s = 0

    p = count/SampleSize # probability of observing an LCS of length 4
                         #between two random DNA sequences of length 9
                             # in a sample size = 1000 data set
    return p
```

In [47]:
```
prob(seq_A,seq_B)
```

Out[47]:
```
0.1174
```

The statistical significance(p-value) of an LCS of length 4 is equivalent to the probability of obtaining a score at least 4, which we can also calculate from the data set of N(sample size) LCS scores using the prob() code above.