Computational Biology: Assignment #1

Due on Monday, Mar 10, 2014

Jianyang Zeng 1:30pm

Weiyi Chen

Problem 1

Pairwise Sequence Alignment

(a)

$$M[i,j] = \begin{cases} 0, i = 0 \text{ and } j = 0 \\ M[i-1,j] - 3, i > 0 \text{ and } j = 0 \\ M[i,j-1] - 3, i = 0 \text{ and } j > 0 \\ M[i-1,j-1] + 1, i,j > 0 \text{ and } A[i] = B[j] \\ max\{M[i,j-1] - 3, M[i-1,j] - 3, M[i-1,j-1] - 1\}, \text{ otherwise} \end{cases}$$

(b)

		A	G	A	T	T
	0	← -3	← -6	← -9	← -12	← -15
A	↑ -3	↖ 1	← -2	← or ↖ -5	← -8	← -11
G	↑ -6	↑ -2	↖ 2	← -1	← -4	← -7
T	↑ -9	↑ -5	↑ -1	↖ 1	↖ 0	← or ↖ -3
T	↑ -12	↑ -8	↑ -4	↑ or ↖ -2	↖ 2	↖ 1

(c)

We can see the largest number is 2, the its trace-back path can be easily seen on the table with a diagonal line. So the optimal alignment is as follows.

$$\begin{array}{ccccc} A & G & A & T & T \\ \downarrow & \downarrow & \downarrow & \downarrow & \\ A & G & T & T & \end{array}$$

Problem 2

I have written the function in python and tested it. Following is my code and I will also attach to make it available for you to check.

```python
def find_longest_increasing_subsequence(ls_sequance):
        '''
        @summary: The longest increasing subsequence problem for a set of numbers
        is to find a subsequence of a given set of numbers, in which elements are
        in sorted order, from lowest to highest.
        @param ls_sequance: list of the original set of numbers
        @return: list of longest subsequence
        '''
        # list of list of longest subsequence for each entry
        ll_longest_subseq = []
        for i, i_number in enumerate(ls_sequance):
                # list of longest subsequence for each entry and initialization
                ls_longest_subseq = [i_number]
                if i == 0:
```

Problem 2 continued on next page. . .

```
                        ll_longest_subseq.append(ls_longest_subseq)
                else:
                        i_longest_length = 0
                        for j in range(i):
                                if len(ll_longest_subseq[i-j-1])>=i_longest_length \
                                and ls_sequance[i-j-1]<i_number:
                                        ls_longest_subseq = ll_longest_subseq[i-j-1]+[i_number]
                                        i_longest_length = len(ls_longest_subseq)-1
                        ll_longest_subseq.append(ls_longest_subseq)
        return ll_longest_subseq[-1]

print find_longest_increasing_subsequence([9,5,8,7,15])
# the return result is 5, 8, 15
```