Exercise set 1 Solution

1. **What are the main possible advantages of DNA-based computing? How about disadvantages ?**

Advantages:

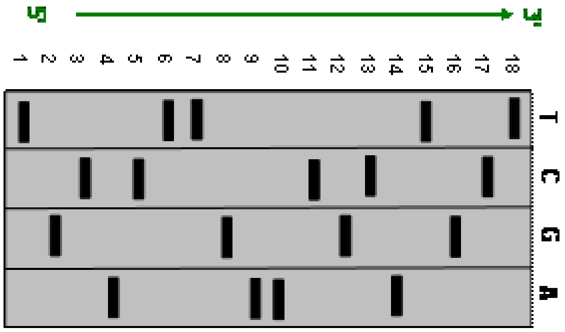
- More parallel computation
- Able to store tremendous amount of information in very small space
- Various fields could be combined together to reach a desirable solution
- Allows for computing in “wet” environments
- Solution for nano-scale computing and engineering

Disadvantage:

- Takes much time to solve simple problems.
- Very difficult to handle errors
- Sometimes there may be an error in the pairing of nucleotides present in the DNA strands.
- Very few “smart” algorithmic solutions exist at the moment, most based on exhaustive search (using the parallelism of biocomputations)

More details: http://www.bvicam.ac.in/news/NRSC%202007/pdfs/papers/st_222_02_02_07.pdf

2. **What is the DNA sequence corresponding to the Sanger plot below?**



The sequence is read from 5’ to 3’.
TGCACTTGAACGCATGCT

3. **Given two sequences, which value is larger: their local similarity or their global similarity? Why? How does their semi-global similarity compare with the other two values?**

<u>Solution:</u>

Global similarity checks what is the similarity between two given sequences and local similarity checks what is the maximum similarity between two given sequences. In semi global similarity we seek a global alignment where we do not penalize for gaps at one or another end of the string.

<u>By definition:</u>

Any global alignment is also a semi global alignment, but there could be better semi-global alignments (each semi-global alignment can be seen as a global alignment between a prefix of one string and a suffix of the other string). Thus, the semi-global best score could be larger than the global score. Also, any semi-global alignment is at the same time a local alignment, but there could be better local alignments (any local alignment could be seen as a semi-global alignment between a prefix of one string and a suffix of the other string; note that it can also be seen as a global alignment between two factors of the two strings). Thus, the local alignment score could be larger than the semi-global one.

The three scores are in general in the following relationship:

Global score ≤ semi-global score ≤ local score

4. **Find all best global alignments between sequences AAAC and AGC, where the scoring scheme is +1 for match, -1 for mismatch and -2 for an alignment with a gap.**

Solution:

Two sequences are given:
s : AAAC
t : AGC

For finding best alignments between *s* and *t*, first create a score matrix D filled with maximum alignments score and right most cell in the last raw gives the best alignment score.

This is calculated using following formula.

D(i,j) = Max { $D(i, j-1) + g$, $D(i-1, j) + g$, $D(i-1, j-1) + f(s[i], t[j])$ }

Here $f(s[i], t[j])$ gives the mismatch/match score for characters s[i] and t[j].

$f(s[i], t[j])$ = 1 , if s[i] = t[j]
= -1 otherwise.

t

D =

s

			A	G	C
		0	1	2	3
	0	0	← -2	← -4	← -6
A	1	↑-2	↖ 1		
A	2	↑-4			
A	3	↑-6			
C	4	↑-8			

D(1,1) = Max { $D(1,0) + (-2)$, $D(0,1) + (-2)$, $D(0,0) + f(s[1], t[1])$ }
= Max { $-2-2$, $-2-2$, $0+1$ }
= 1 (It is calculated from the cell D(0,0). Cell D(1,1) would have diagonal arrow pointing D(0,0).)

Arrows in the table indicate from which cell the maximum score is calculated. We continue filling entries and tracing arrows.

t

D =

s

			A	G	C
		0	1	2	3
	0	0	← -2	← -4	← -6
A	1	↑-2	↖ 1	←-1	← -3
A	2	↑-4	↖ ↑ -1	↖ 0	↖ ← -2
A	3	↑-6	↖ ↑ -3	↖ ↑ -2	↖ -1
C	4	↑-8	↑ -5	↖ ↑ -4	↖ -1

Best alignment score is -1.

Optimal alignments could be found by walking on the traced arrow path from D(m,n) to D(0,0).

3 possible moves:

- Diag: the letters from two sequences are aligned
- Left: gap is introduced into the left sequence
- Up: a gap is introduced into the top sequence

Resulting alignments are as follows:

Alignments	AAAC _ AGC	A A A C A G _ C	A A A C A _ G C
Score	(-2)+1+(-1)+1 = -1	1+(-1)+(-2)+1 = -1	1+ (-2)+(-1)+1 = -1
Solution	Best Alignment	Best Alignment	Best Alignment

Alignments listed above are the best alignments.

5. Find all best global alignments between sequences ATAG and TTCG, where the scoring scheme is +1 for match, -1 for mismatch and -1 for an alignment with a gap.

Solution:

t

D =

s

			T	T	C	G
		0	1	2	3	4
	0	0	← -1	← -2	← -3	← -4
A	1	↑-1	↖← -1	↖← -2	↖← -3	↖← -4
T	2	↑-2	↖ 0	↖ 0	← -1	← -2
A	3	↑-3	↑ -1	↖↑-1	↖ -1	↖ -2
G	4	↑-4	↑ -2	↖↑-2	↖↑ -2	↖ 0

Start tracing a path from D(4,4) to D(0,0).

Alignments	A T A G T T C G	A _ T A G _ T T C G
Score	(-1)+1+(-1)+1 = 0	(-1)+(-1)+1-1+1 = -1

6. **Find all best local alignments between sequences ATACTGGG and TGACTGAG, using the same scoring scheme as in exercise 2.**

Solution:

Two sequences are given

s : ATACTGGG
t: TGACTGAG

Method : Smith Waterman method

Entries in the table are calculated using following formula.

$$L(i,j) = Max\{0, L(i-1,j-1) + f(s[i],t[j]), L(i-1,j) + g, L(i,j-1) + g\}$$

t

L =

s

			T	G	A	C	T	G	A	G
		0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0	0
A	1	0	0	0	↖1	0	0	0	↖1	0
T	2	0	↖1	0	0	↖1	↖1	0	0	↖0
A	3	0	0	↖0	↖1	0	↖0	↖0	↖1	0
C	4	0	0	0	0	↖2	←0	0	0	↖0
T	5	0	↖1	0	0	0	↖3	←1	0	0
G	6	0	0	↖2	←0	0	↑1	↖4	←2	↖1
G	7	0	0	↑0	0	0	0	↖2	↖3	↖3
G	8	0	0	↖1	0	0	0	↖1	↖ 1↑	↖4

From the above matrix we find the highest score and trace the path until we come to a cell with score zero. This cell is not included in the alignment.

Alignments	ACTGGG ACTGAG	ACTG ACTG
Score	1+1+1+1+(-1)+1 = 4	1+1+1+1 = 4

Above alignments are the best local alignments.

7. What scoring schemes should you use to determine the longest common substring and the longest common subsequence for two given strings, using the algorithm for best global alignment?

Solution: For the longest common substring and longest common subsequence we do not want to penalize the gaps in the beginning and in the end. To exclude mismatches or gaps, we penalize drastically, say by scoring them with -N, where N is larger than the length of either string. For example, N = 1 + (length(s) + length(t)) (or any other number larger than both m and n).

- The scoring scheme for longest common substring for two given strings *s* and *t* where length of s is m and length of t is n. We reward matches with 1 and exclude mismatches and gaps by scoring them with –N.

  L(0,j) = L(i,0) = 0, where i=0,1,…m and j = 0,1,…..,n.

  L(i,j) = L(i-1,j-1) + 1 if (s[i] = t[j])
   - N,   otherwise.
- The scoring scheme for longest common sequence would be as follows.
  We reward matches with 1 and exclude mismatches by scoring them with –N. For longest common subsequence we allow insertion of gaps.

L(0,j) = L(i,0) = 0, i=0,1,…m, j = 0,1,…..,n.

L(i,j) = Max (L(i,j-1), L(i-1,j), L(i-1,j-1)+ f(s[i],t[j])) where f(s[i],t[j]) = 1, if s[i] = t[j]
-N , otherwise.

- We fill scoring matrix according to above scoring scheme and with tracing arrows. The arrow points the cell from where the current cell value is coming from.

- Start with the lower right corner cell and trace back until it reaches to the upper-left corner.

- Extract from this sequence of numbers the longest non-increasing sequence. This sequence, without its last number, will indicate the longest common substring/subsequence

- Another way in which this matrix can be used is as follows. Following the ideas from the local alignment algorithm: start from the highest value in the matrix and go as much as possible on a path of non-increasing sequence of values

8. Apply the scoring scheme you indicated in exercise 7 to find all longest common substrings for strings ATACTGGG and TGACTGGT.

Solution:
Assume we score the insertion of a gap and a mismatch with -9 (any number larger than the length of either string is ok).

The longest common substring is **ACTGG**, (Path colored in red shows the non increasing part).

A T ACTGGG
T GACTGGT

			A	T	A	C	T	G	G	G
		0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0	0
T	1	0	-9	1	-9	-9	1	-9	-9	-9
G	2	0	-9	-9	-9	-9	-9	2	-8	-8
A	3	0	1	-9	-8	-9	-9	-9	-9	-9
C	4	0	-9	-9	-9	-7	-9	-9	-9	-9
T	5	0	-9	-8	-9	-9	-6	-9	-9	-9
G	6	0	-9	-9	-9	-9	-9	-5	-8	-8
G	7	0	-9	-9	-9	-9	-9	-8	-4	-7
T	8	0	-9	-8	-9	-9	-8	-9	-9	-9