Name, SURNAME and ID ⇒ SOLUTIONS

Middle East Technical University
Department of Computer Engineering

CENG 465

Introduction to Bioinformatics

Spring '2006-2007

Midterm Exam I

- **Duration: 100** minutes.
- **Exam:**
  - This is a **closed book, closed notes** exam. The use of any reference material is strictly forbidden.
  - No attempts of cheating will be tolerated. In case such attempts are observed, the students who took part in the act will be prosecuted. The legal code states that students who are found guilty of cheating shall be expelled from the university for **a minimum of one semester!**
- **About the exam questions:**
  - The points assigned for each question are shown in parenthesis next to the question.
  - For *True-False* type questions, put your results in the boxes provided.
- **This exam consists of *8* pages including this page. Check that you have them all!**
- **GOOD LUCK !**

Question 1	20
Question 2	20
Question 3	25
Question 4	25
Question 5	10
Total ⇒	100

1 (20 pts)

20

For the following 10 statements, indicate whether the statement is *true* or *false* by marking the corresponding box with **T** or **F**, respectively (2 points each).

i. Assume that we have computed the *z-score* of a sequence alignment. A higher *z-score* indicates a more significant alignment, i.e., the sequences are more biologically related. **T**

ii. If we are aligning two DNA sequences of the same length, the dynamic programming algorithm will not introduce any gaps in the resulting alignment. **F**

iii. The $E-value$ of a sequence similarity hit provided by a BLAST search cannot be greater than 1. **F**

iv. The time complexity of *global alignment* by dynamic programming is greater than *local alignment* with dynamic programming. **F**

v. In global sequence alignment, the maximum score (optimum score of the DP alignment) is always at the lower-right corner of the partial scores table. **T** F is accepted aswell due to ambiguous ~~statement~~ statement

vi. A codon (DNA sequence of length 3) is able to encode 64 different amino acids, since $4 \times 4 \times 4 = 64$. However, since there are 20 amino acids in nature, some of the codons are ignored during the transcription of DNA. **F**

vii. An internal node of a suffix tree may have more than two child nodes. **T**

viii. The number of leaf nodes in a suffix tree is equal to the number of suffixes of the string (including the terminal character) for which the tree is built. **T**

ix. The number of columns in a multiple sequence alignment is equal to the length of the longest sequence of the aligned set. **F**

x. Consider the multiple alignment of the same set of sequences by two methods, A and B. If the alignment of sequences with A has a lower entropy score, it means that A better aligns these sequences compared to B. **T**

2 **(20 pts)**

20

(a)(10 pts) Fill out the dynamic programming table for determining the optimum **local alignment** between the sequences MVILL and VGIL. Assume that a match is scored +3 and that mismatches and gaps are penalized -1 each (assume linear gap penalty).

	-	M	V	I	L	L
-	0	0	0	0	0	0
V	0	0	3	2	1	0
G	0	0	2	2	1	0
I	0	0	1	5	4	3
L	0	0	0	4	8	7

(b)(10 pts) What is the optimum local alignment corresponding to the table in part (a) and what is its score? Show the alignment path on the partial scores table and show the resulting alignment.

seq 1: V – I L

seq 2: V G I L

3+(-1)+3+3 = 8

3 (25 pts)

25

Consider the following 4 DNA sequences and the optimum global alignment between all pairs, where match score is 3 and mismatch score is -1. A linear gap model with a gap penalty of -1 is used.

Sequences:
(1) TCTT, **(2)** CTTA, **(3)** GCTT, and **(4)** TCTA

Optimum global pairwise alignments and their corresponding scores:

1. TCTT-
   |||
   -CTTA
   Score: 7
2. TCTT
   |||
   GCTT
   Score: 8
3. TCTT
   |||
   TCTA
   Score: 8
4. -CTTA
   |||
   GCTT-
   Score: 7
5. -CTTA
   || |
   TCT-A
   Score: 7
6. GCTT
   ||
   TCTA
   Score: 4

(a)(10 pts) Suppose that we want a multiple alignment of these four sequences using the **Star Alignment** algorithm. Which one of these sequences would be the *center* sequence? Why? Show your numerical calculations.

Total score of SEQ1 = 7+8+8 = 23
Total score of SEQ2 = 7+7+7 = 21
Total score of SEQ3 = 8+7+4 = 19
Total score of SEQ4 = 8+7+4 = 19

SEQ1 : TCTT has the best total score. Therefore, it is the closest sequence to all the sequences. Hence, it is the center sequence.

(b)(10 pts) Using the progressive star alignment technique and the center sequence you have chosen in part (a), construct a multiple alignment of these four sequences. Show individual steps of construction (3 alignments in total).

Alignment 1:

```
T C T T -
  | | |
- C T T A
```

Alignment (Step) 2:

```
T C T T -
- C T T A
G C T T -
```

Alignment (Step) 3:

```
T C T T -
- C T T A
G C T T -
T C T A -
```

Resulting multiple alignment.

At each step of the alignment, we use the optimum alignment between the center sequence and the sequence added to the MSA at that step.

(c)(5 pts) Which column (or columns) of the alignment gives the lowest entropy? Which column (or columns) gives the highest entropy? You do not need to calculate the exact entropy numerically.

The second and third (all Cs and all Ts) columns have the lowest entropy (0: zero). The highest entropy column is the first column, because it is the most diverse column with two Ts, one G, and a gap.

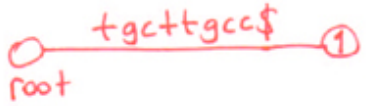
4 (25 pts)

25

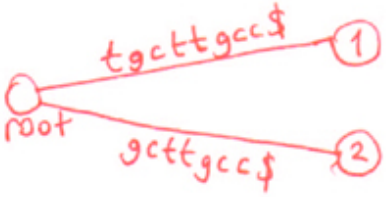
(a)(15 pts) Construct a suffix tree for the following string: **tgcttgcc$**. Show the individual steps of construction (9 steps in total).

The leaf nodes are labeled with the suffix ids.

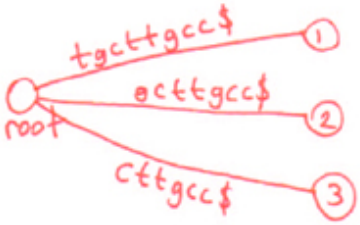
step 1 :
(insert tgcttgcc$)



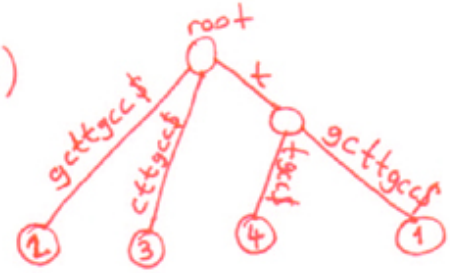
step 2 :
(insert gcttgcc$)



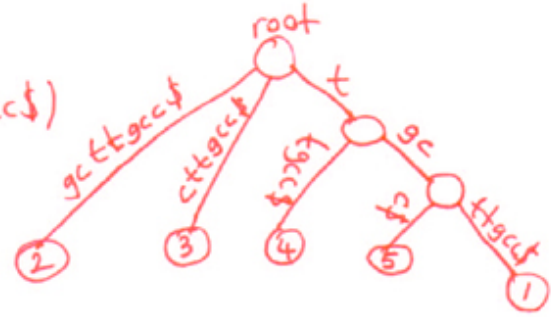
step 3:
(insert cttgcc$)



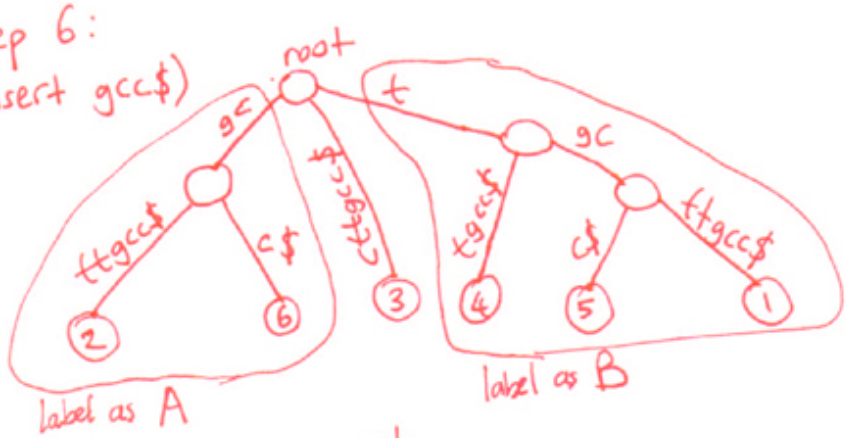
step 4:
(insert ttgcc$)



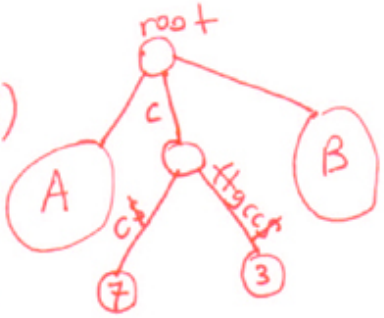
step 5:
(insert tgcc$)



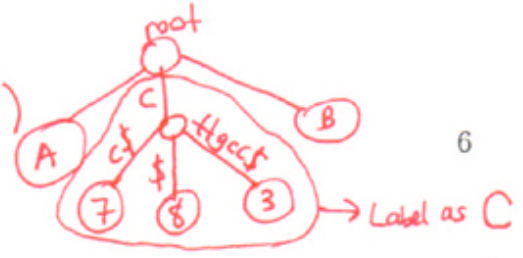
step 6:
(insert gcc$)



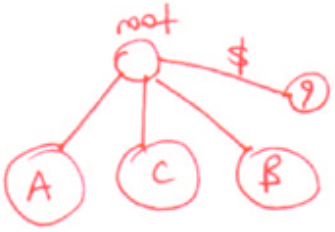
step 7:
(insert cc$)



step 8:
(insert c$)



step 9:
(insert $)



(b)(10 pts) Search the following patterns in the suffix tree you created in part (a). Show your search steps. You may either provide written explanations of the search steps without redrawing the suffix tree you created in part (a), or you may show your steps visually by redrawing the suffix tree. If the pattern is found, indicate the number occurrences of the pattern and show the places of occurrence.

1. (5 pts) **Search pattern: gc**

We first follow the link from the root that has the edge label "gc". Since these are all the characters of the search pattern, we have completed the matching process. There are 2 leaves below; hence, 2 occurences

gcc$ ⟶ starts from the sixth character

gc++gcc$ → starts from the second character

2. (5 pts) **Search pattern: tgt**

We follow the edge with label "t" from the root. We then follow the edge whose label starts with g. But the third character "t" does not match the second character in the label which is "c". So, there is a mismatch and the pattern does not occur in the string.

5 (10 pts)

10

Describe why suffix arrays are considered more space efficient compared to suffix trees, despite the fact that they both have the same space complexity.

Suffix trees and suffix arrays have the same space complexity. However, the constant terms ~~describing~~ in the equations describing the space requirement is different.

In suffix arrays, we only store n suffix ids in an array. The space requirement is exactly n. However in suffix trees, in addition to the n leaves for n suffixes, we have intermediate nodes and pointers in the tree structure. These additional terms increase the space requirement of suffix trees compared to suffix arrays.