CMSC423 - Midterm 10/7/2008

Name _____

Honor Pledge

The University of Maryland Code of Academic Integrity requests that you write **by hand** and sign the following statement pledging your commitment to academic integrity. Please do so in the blank space below the text of the honor pledge.

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Signature _____

Please write your name on all additional sheets of paper you use.

NOTE: Please budget your time carefully! You only have 75 minutes available for this exam. There is a very good chance you will not be able to answer all the questions in the allotted time (though it is definitely possible to do that).

1. (20 points) The basics!
a) Define the term "synonymous mutation"
b) What is the "central dogma" of molecular biology?
c) Reverse complement the following DNA string

ATGCGGGCAC GGAAGCCGTG TACGCGAGCG CGCTTGAGGG

d) Identify the longest open reading frame in the following DNA sequence and translate it into an amino-acid sequence (note: translation table provided at the end of the exam)

TGCGTATGTATGTCAGACGGTGAGACGCTTGCGGGCTAAGCGACG

2. (20 points) Given the following string, construct a suffix tree, including the suffix links.

GATATAGTAG

(Note: this will be messy - draw carefully. Also, no need to draw suffix links that point to the root of the tree)

3. (20 points) Exact string matching.

a) The steps outlined below represent an execution of the KMP algorithm (text on top, pattern on bottom)

```
(i)  ABCABCDABABCDABDABDE        (ii)  ABCABCDABABCDABDABDE
     |||X                              |||||X
     ABCDABD                           ABCDABD

(iii)  ABCABCDABABCDABDABDE      (iv)  ABCABCDABABCDABDABDE
                 X                                |||||||
               ABCDABD                            ABCDABD
```

In step (i), the pattern is matched until a mismatch occurs at the 4th character. Since no suffix of the matched portion matches a prefix of the whole pattern (no prefix and suffix of ABC match each other), the pattern is shifted beyond the aligned region and the matching continues from the beginning of the pattern. In (ii) a mismatch is found at position 7 in the pattern and the pattern is shifted (iii) so that the common prefix and suffix of the matched regions are aligned (the first AB in the pattern is placed where the second AB matched at step (ii)). The third position in the pattern is compared to the text and results in a mismatch. The pattern is then shifted past this position and a match is found.

Using this execution as an example, provide an argument why the overall running time of the algorithm proportional to the sum of the lengths of pattern and text.

b) During the execution of the algorithm described in class for computing Z values, when computing the value Z[i] we relied on the Z-value for a position $j < i$ such that
(i) Z[j] extends the farthest in the string ($j + Z[j]$ is maximum over all choices of $j < i$) and
(ii) $j + Z[j] > i$

Would the algorithm still work efficiently (linear time) if only the second condition were satisfied? Which part of the reasoning would fall apart?

Hint: this is related to part a) of this question.

4. (20 points) Remember that a suffix tree is a compressed representation of all suffixes in a string, such that each suffix is represented by a different leaf in the tree. The least common ancestor of two nodes in a tree is the lowest node shared by the paths from the two nodes to the root. Assume n is the least common ancestor of leaves i and j in a suffix tree for string S.

a. What does this node represent?
b. Describe an algorithm that will compute the Z values for S in O(n) time, using the suffix tree assuming you are given function that allows you to compute the least common ancestor of any two nodes in constant time.

Reminder: for any location i in S, Z[i] is the longest prefix of S[i..n] that matches a prefix S.

5. (20 points) Suppose we have sequences $v = v_1, \ldots, v_n$ and $w = w_1, \ldots, w_m$, where v is longer than w. We wish to find a substring of v which best matches all of w. Global alignment won't work because it will try to align all of v. Local alignment won't work because it may not align all of w. The problem (called the fitting problem) can be formulated as the problem of finding a substring v' of v that maximizes the score of alignment s(v', w) among all substrings of v. Give an algorithm which computes the optimal fitting alignment in O(nm) time. Describe both how to compute the score of this alignment and how to compute the actual alignment.

Note: You don't need to spell out all the details of the algorithm. When using traditional dynamic programming approach, it is sufficient to specify the initial conditions (what values are written in the first row/column of the matrix), where will the score for this best alignment be located in the matrix, and whether you use the global alignment recurrence in the algorithm (max value between left, diagonal, and above), or the local alignment recurrence that allows the alignment to restart at any location by taking the maximum of four values: 0 and the three values mentioned above.

6. (BONUS: 20 points). In class we discussed two approaches to sequence alignment - global and local alignment. A global alignment requires the two sequences to be aligned end-to-end while a local alignment allows one to ignore any mismatches occurring at the end of the sequences. There is, however, a middle ground - the **semiglobal** alignment. In a semi-global alignment all characters in the two sequences must be aligned, however only gaps internal to the alignment are counted, while gaps at either end of the alignment are "free".

For example, the following sequences aligned optimally using global alignment:

```
CAGCACTTGGATTCTCCGG
CAGC-----G-T-----GG
```

can be aligned optimally in a semiglobal fashion as follows:

```
CAGCA-CTTGGATTCTCGG
---CAGCGTGG--------
```

Describe a dynamic programming algorithm that computes the semiglobal alignment of two strings in time O(mn). The note in problem 5 applies here as well.

Translation table

Ter - stop codon

TTT F Phe	TCT S Ser	TAT Y Tyr	TGT C Cys
TTC F Phe	TCC S Ser	TAC Y Tyr	TGC C Cys
TTA L Leu	TCA S Ser	TAA * Ter	TGA * Ter
TTG L Leu	TCG S Ser	TAG * Ter	TGG W Trp
CTT L Leu	CCT P Pro	CAT H His	CGT R Arg
CTC L Leu	CCC P Pro	CAC H His	CGC R Arg
CTA L Leu	CCA P Pro	CAA Q Gln	CGA R Arg
CTG L Leu	CCG P Pro	CAG Q Gln	CGG R Arg
ATT I Ile	ACT T Thr	AAT N Asn	AGT S Ser
ATC I Ile	ACC T Thr	AAC N Asn	AGC S Ser
ATA I Ile	ACA T Thr	AAA K Lys	AGA R Arg
ATG M Met	ACG T Thr	AAG K Lys	AGG R Arg
GTT V Val	GCT A Ala	GAT D Asp	GGT G Gly
GTC V Val	GCC A Ala	GAC D Asp	GGC G Gly
GTA V Val	GCA A Ala	GAA E Glu	GGA G Gly
GTG V Val	GCG A Ala	GAG E Glu	GGG G Gly