B363: Bioinformatics algorithms

HW3 (Due: **Oct. 14 Wed 11:59pm**)

(You do not need to write computer programs for the following questions.)

1. (5 pts) Show the complete theoretical mass spectrum generated from the cyclopeptide NLYV.

2. (10 pts) Mr. Study devises the following algorithm to solve the spectrum convolution problem (page 203 of the textbook). Is the algorithm correct? What is the run time of the algorithm? Can you revise the algorithm to make its run time only dependent on the size of *spectrum* and independent on *MaxMass*?

```
Input: A collection of integers Spectrum
NovelSpectrumConvolution(Spectrum)
MaxMass ← maximum integer in Spectrum
Initialize an array Count of size MaxMass with all values = 0
Sort Spectrum in increasing order
for each M1 in Spectrum
    for each M2 > M1 in Spectrum
        Diff ← M2 – M1
        Count[Diff] ← Count[Diff] + 1
ConvolutionSpectrum ← empty set
for i ← 1 to MaxMass
    if Count[i] > 0
        add (i, Count[i]) into ConvolutionSpectrum   //(mass, multiplicity)
Sort ConvolutionSpectrum in decreasing order of the multiplicity
Output ConvolutionSpectrum
```

3. (10 pts) Mr. Fuzzy claims that the theoretical spectrum of a cyclopeptide is a superset of the theoretical spectrum of any corresponding linear peptide (that resulting from the break of the circular peptide at an arbitrary location), and thus in order to reconstruct a cyclopeptide from a given mass spectrum, one can always first reconstruct a linear peptide. Is Mr. Fuzzy correct? If yes, explain your answer; otherwise, give a counterexample.

4. (10 pts) Devise a dynamic programming algorithm to solve the *Counting peptides with Given Mass Problem* (page 193): given an integer, count the number of linear peptides having the integer mass m.

5. (10 pts) In a noisy mass spectrum of a cyclopeptide, the maximum value in the spectrum may not be the parent mass of the cyclopeptide. Devise an algorithm to compute the parent mass of a cyclopeptide from a given noisy mass spectrum.

6. (15 pts) Construct the alignment graph of ACGTTAA and AGTTTA (using the score = 3 for matches, and = -2 for mismatches and gaps). Show the optimal alignment, and its corresponding path in the alignment graph.

7. (10 pts) Modify the dynamic programming algorithm for the global alignment of two DNA sequences to solve the Overlap Alignment Problem (page 266): given two strings *u* and *v*, and a scoring matrix *Score*, find a highest-scoring *overlap alignment*, i.e. a global alignment of a suffix of *u* and a prefix of *v*.

8. (5 pts) Does an optimal multiple alignment always induce optimal pairwise alignments? If yes, explain your answer; otherwise, give a counterexample.

9. (15 pts) Construct the alignment graph of ACTACA and ACAA (using the score = 2 for matches = -1 for mismatches, -7 for gap opening penalty and -1 for gap extension penalty). Show the optimal alignment, and its corresponding path in the alignment graph.

10. (10 pts) The divide-and-conquer algorithm presents a simple approach to parallelize the pairwise alignment of very long DNA sequences (e.g., genomic sequence of millions of nucleotides), because 1) the FromSource(i) and ToSink(i) algorithms can be carried out independently on two separate CPUs without exchanging intermediate results; and 2) the alignment of the prefix and suffix strings broken at the midNode can be carried out separately on separate CPUs. However, the two subproblems after the divide-and-conquer may take much different time to complete (note that the areas under these two subproblems may not be the same if the midNode locates far below n/2 in rows even though it always locates on m/2 in columns). Modify the algorithm of finding the midNode so that the run time of the two subproblems after the divide-and-conquer are approximately equal, assuming the two input sequences have about the same length (m ~ n). (Hint: you can try to find a midNode not on the column of m/2, but on a main diagonal of the alignment grid).