Problem 1 (15%): Suppose we are given a very long DNA sequence where the occurrence probabilities of nucleotides A (adenine), C (cytosine), G (guanine), T (thymine) are 0.1, 0.3, 0.4, and 0.2, respectively.

(a) (10%): Construct a Huffman code for them. You should work out the binary tree construction as well as the code assignment.

(b) (5%): By the above Huffman coding scheme, what is the binary string for a 10-nucleotide DNA sequence “GGGCTTCACG.”

Problem 2 (15%): In class, we introduced an \( O(n \log n) \)-time algorithm for finding a longest increasing subsequence. Use \( ⟨8, 2, 6, 4, 5, 7, 3, 1, 12, 9, 10⟩ \) to explain how the algorithm works.

Problem 3 (10%): Given a sequence of real numbers \( A = ⟨a_1, a_2, \ldots, a_n⟩ \), the maximum-sum segment problem is to find a consecutive subsequence, i.e., a substring or segment, in \( A \) with the maximum sum. Let \( \text{prefix sum} \ P[i] = \sum_{j=1}^{i} a_j \) be the sum of the first \( i \) elements. Explain how to use the prefix sum to deliver the maximum-sum segment in \( O(n) \) time.

Problem 4 (25%): In this problem, we employ a simple scoring scheme where each gap symbol is penalized by a nonnegative constant \( \beta \). Let \( S[i, j] \) denote the score of an optimal alignment between \( ⟨a_1, a_2, \ldots, a_i⟩ \) and \( ⟨b_1, b_2, \ldots, b_j⟩ \). With proper initializations, \( S[i, j] \) can be computed by the following recurrences:

\[
S[i, j] = \max \left\{ 
\begin{array}{ll}
S[i-1, j] - \beta \\
S[i, j-1] - \beta \\
S[i-1, j-1] + \sigma(a_i, b_j)
\end{array} \right.
\]

(a) (15%): Write down a complete pseudo-code for computing \( S[m, n] \) in \( O(mn) \) time and \( O(m+n) \) space. All initializations should be included in the pseudo-code.

(b) (10%): Assume that we allow at most three gaps in an alignment. Give a method (as efficient as possible) for computing the score of an optimal alignment.

Problem 5 (20%): In affine gap penalties, a gap of length \( k \) is penalized by \( \alpha + k \times \beta \), where \( \alpha \) and \( \beta \) are both nonnegative constants.

(a) (10%): Give the recurrence relations for computing the score of an optimal (global) alignment between \( A \) and \( B \). Justify your recurrence relations and include all initializations.

(b) (10%): Give the recurrence relations for computing the score of an optimal local alignment between \( A \) and \( B \). Explain your recurrence relations and include all initializations.

Problem 6 (15%): Consider the problem of computing all \( \Delta \)-points of two sequences of lengths \( m \) and \( n \), where \( m \ll n \). Describe a method for computing all \( \Delta \)-points that works in \( O(mn) \) time and \( O(m^{\frac{3}{2}} + n) \) working space.