Lab Session 6

16.09.2024

BT 3051 - DSA Biology Lab

Problem1: Optimal DNA Fragment Selection

In DNA sequencing, you often need to assemble a sequence from smaller overlapping fragments. You are given several DNA fragments and your goal is to select a subset of non-overlapping fragments that covers the longest part of the DNA sequence.

Each fragment has a start and end position on the DNA strand. Your task is to select the maximum number of non-overlapping fragments.

Input:

- n (number of DNA fragments)
- An array fragments[] of length n, where each element is a tuple (start, end) representing the start and end positions of a DNA fragment.

Output:

- The maximum number of non-overlapping fragments you can select.

Hint (Greedy strategy): Sort the fragments by their end positions, and then use a greedy approach to select the first fragment that ends the earliest and doesn't overlap with previously selected fragments.

Problem 2:

You are working on optimizing gene sequencing. A gene can be represented as a string made up of characters 'A', 'C', 'G', and 'T'. Given a collection of genes (strings) and their importance values, your task is to select the most important genes such that the total length of selected genes does not exceed a given limit L. The goal is to maximize the total importance of the selected genes.

Input:

- n (number of genes)
- An array genes[] of length n, where each element is a tuple (gene_string, importance).
- An integer L (the maximum allowed length of the selected genes).

Output:

- The maximum total importance of the selected genes.

Hint (Greedy strategy): Select the genes with the highest importance-to-length ratio, as this will give you the most importance per unit length.

Problem 3: In protein folding, the energy required to fold a chain of proteins can be reduced if certain specific subchains are folded together. Each subchain has an associated folding cost. Given a sequence of protein folding costs, determine the minimum cost required to fold the entire protein chain.

Input:

- n (the number of segments in the protein chain).
- An array cost[] of length n representing the folding cost of each segment.

Output:

- The minimum cost required to fold the entire chain.

Hint (Dynamic Programming strategy): Use a DP array where dp[i] represents the minimum cost to fold the chain up to segment i. At each step, decide whether to fold a segment alone or combine it with previous ones to minimize the total folding cost.