BT-3051 Data Structures and Algorithms for Biology
Assignment - 4

Submission : Since this is a coding based assignment, students need to submit their codes in a zipped folder. Your zip file should be named something like BTyyBxxx.zip, based on your roll number. This zip file must contain the codes used for each of the problems in separate **.ipynb** files with proper documentation.

Deadline : 30 September 2024, 23:59 hrs
Bonus marks and penalty will be applied as mentioned in the course plan.

Instructions:

a. Total marks for this assignment - 30
b. No restrictions on usage of functions, unless otherwise specified.
c. Submissions will be evaluated only if they are provided in the required .ipynb format.

Greedy Algorithmic Strategy:

Problem 1: Optimal Gene Splicing (7 marks)

In genomics, researchers often need to assemble DNA fragments into a full genome. Each fragment represents a portion of the genome, and some fragments overlap with others. The goal is to reconstruct the genome with the minimum number of fragments. Each fragment has a start position and an end position.

You are given a list of DNA fragments, where each fragment is represented by its start and end positions. Write a function to find the **minimum number of fragments** required to cover the entire genome from position 1 to n using a **greedy approach**.

Function:

```
def min_dna_fragments(fragments, n):
    """
    Returns the minimum number of DNA fragments required to cover the genome.

     :param fragments: List of tuples (start, end), where each tuple represents the start and end positions
of a fragment
    :param n: int, the length of the genome
    :return: int, minimum number of fragments required to cover the genome
    """
    pass

# Example call
fragments = [(1, 4), (3, 5), (0, 6), (5, 10), (8, 12)]
n = 12
min_dna_fragments(fragments, n)
```

Output Example:

```
3  # Minimum number of fragments (e.g., fragments [(0, 6), (5, 10), (8, 12)])
```

Problem 2: (5+2 marks)

In gene expression studies, researchers need to activate certain genes for specific time intervals during an experiment. Each gene activation represents a critical experiment interval where the gene is expressed. However, due to resource limitations (e.g., using a specific tool to express genes), **only one gene can be activated at a time**.

Given a list of gene expression intervals, each with a start and end time, your task is to select the **maximum number of non-overlapping gene expression intervals** to run during the experiment.

Write a greedy algorithmic (python code) to solve the **Interval Scheduling Problem** .

Also Explain the Approach used in the code .

Input:
A list of n gene expression intervals. Each interval is represented as a tuple (start_time, end_time) where start_time is the time when the gene starts expressing, and end_time is the time when the gene stops expressing.

Output:
Return the maximum number of non-overlapping gene expression intervals that can be scheduled.
Input
intervals = [(1, 3), (2, 5), (4, 7), (6, 8)]
Output
Maximum number of non-overlapping intervals: 2

Explanation : you can select the intervals (1, 3) and (6, 8) as they do not overlap. This allows for two gene expression activations without conflict.

Dynamic Programming:

Problem 3: (6+2 marks)

In genetics, palindromic sequences play a crucial role in biological processes like DNA replication and transcription. A palindromic DNA sequence is one that reads the same forward and backward. Given a DNA sequence, your task is to partition the sequence such that every subsequence in the partition is a palindromic DNA sequence.
Write a python program to return all possible ways to partition the DNA sequence where every partition is a palindrome using Dynamic Programming .
Also Explain the Approach

Input:
A string dna_sequence, consisting of nucleotides ('A', 'T', 'C', 'G').
Output:
Return a list of lists, where each inner list represents a valid partition of palindromic subsequences.

Example:
Input:
dna_sequence = "ATTA"
Output:

```
[["A", "T", "T", "A"],
 ["A", "TT", "A"],
 ["ATTA"]
]
```

Problem 4: DNA Sequence Alignment (8 marks)

In RNA folding, predicting secondary structure involves finding the maximum number of non-overlapping base pairs that form between certain bases (e.g., A-U and G-C). Given an RNA sequence, the task is to find the **maximum number of base pairs** that can be formed, using dynamic programming.

Write a dynamic programming solution to calculate the maximum number of valid base pairs that can be formed in an RNA sequence, ensuring no overlapping pairs.

Function:

```
def max_rna_pairs(rna_sequence):
    """
    Finds the maximum number of valid base pairs that can form in an RNA sequence.

    :param rna_sequence: str, the RNA sequence
    :return: int, the maximum number of valid base pairs
    """
    pass

# Example call
max_rna_pairs("AUGCUAGC")
```

Output Example:

```
3  # Maximum number of base pairs that can be formed (e.g., A-U, G-C, G-C)
```