## The most occurring section B3 task

## Specification:

Input:

 $dna \in T$ 

Output:  $maxSequence \in T$ ,  $highestCount \in \mathbb{N}$ 

Precondition: length(dna)  $\geq 255$ 

 $A: N \rightarrow L; A(isMatch) = (isMatch = true)$ 

 $\forall c \in \text{dna}, c \in \{A, C, G, T\}$  (every character in dna is one of A, C, G, T) mostFrequentSequence = "" (mostFrequentSequence is initialized as an empty string)

highestCount = 0 (highestCount is initialized to zero).

Postcondition:  $1 \le \text{length}(\text{mostFrequentSequence}) \le \text{length}(\text{dna})$ and  $\text{length}(\text{mostFrequentSequence}) \ge 3$ highestCount =  $\text{max}(\text{count}(s) \mid s \subseteq \text{dna}, \text{length}(s) \ge 3)$ .  $\forall s \ (s \subseteq \text{dna}, \text{length}(s) \ge 3) \Rightarrow \text{count}(s) \le \text{highestCount}$ .

$$count = \sum_{i=1}^{length(dna)} 1$$

$$A(isMatch)$$

General Postcondition(maximum item selection + multiple item selection)

$$cnt = \sum_{\substack{i=1\\A(X[i])}}^{length(X)} 1 \text{ and } \forall i (1 \le i \le cnt) : A(X[Y[i]]) \text{ and } Y \subseteq (1, 2, ..., length(X))$$

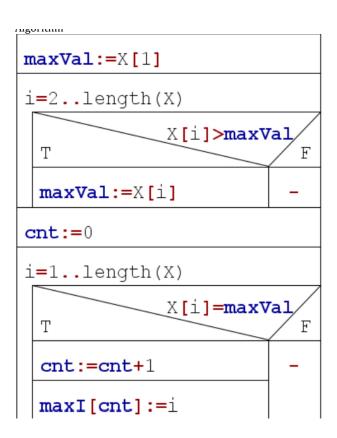
$$cnt = \sum_{i=1}^{length(X)} 1$$

$$A(X i)$$

Pattern: Maximum selection and Counting

Algorithm

Pattern	Task
X	dna
i	start (loop index for substring start)
i ≤ length(X)	start < dna.Length
i := i + 1	start++
exists	highestCount > 1
exists := (i ≤ length(X))	highestCount > 1 (after loop completion)



exists	highestCount > 1
exists := (i ≤ length(X))	highestCount > 1 (after loop completion)
maxl	mostFrequentSequence (tracks the sequence with the highest count)
A(X[j])	countOfCurrentSequence > 1
X[j] > X[maxl]	countOfCurrentSequence > highestCount
maxl := j	mostFrequentSequence = currentSequence (when a new max is found)

art = s	art.,Length(dna) - 2	
lengt	= (start + length)Length(dna)	
٥	punt = 0	
c	urrentSequence := "	
k	= klength	
	currentSequence := currentSequence + dna[start + k]	
	indexLength(dna)-length + 1	
	isMatch := true	
	matchIndex = matchIndexJength	_
	dna[start + matchindex] ≠ dna[index + matchindex]	
	Т	
	isMatch := false	T
	isMatch	_
	Т	_
	count++	
	index := index + length - 1	
	count > highestCount or (count == highestCount and currentSequence.Length > mostFrequentSequence	.Ler
	Т	_/
	mostFrequentSequence := currentSequence	

```
Code
```

```
namespace tasknewnewnew3
  internal class Program
  {
    static void Main(string[] args)
        string dna = Console.ReadLine();
        string mostFrequentSequence = "";
        int highestCount = 0;
        for (int start = 0; start <= dna.Length - 3; start++)
           for (int length = 3; start + length <= dna.Length; length++)
             int count = 0;
            string currentSequence = "";
             // Build the current sequence manually
             for (int k = 0; k < length; k++)
             {
               currentSequence += dna[start + k];
             for (int index = 0; index <= dna.Length - length; index++)
               bool isMatch = true;
               for (int matchIndex = 0; matchIndex < length; matchIndex++)
                 if (dna[start + matchIndex] != dna[index + matchIndex])
                   isMatch = false;
                   break;
                 }
               }
               if (isMatch)
                 count++;
                 index += length - 1;
            }
             if (count > highestCount || (count == highestCount && currentSequence.Length >
mostFrequentSequence.Length))
               mostFrequentSequence = currentSequence;
               highestCount = count;
        Console.WriteLine(highestCount > 1 ? $'{mostFrequentSequence} {highetCount}': "-1")
   }
```

## Initialization

- 1. Input Handling:
  - dna: Read the input string from the console. Trim any leading or trailing whitespace and convert to uppercase for uniform processing.
- 2. Variable Initialization:
  - mostFrequentSequence: Initialize as an empty string. It will eventually store the most frequent DNA sequence.
  - highestCount: Set to 0, to be used for tracking the highest frequency of any sequence.
  - sequenceCounts: A dictionary to keep track of the frequency of all sequences encountered.
  - minSequenceLength: Set to 3, specifying the minimum length for sequences to be considered in the analysis.
- 3. Pre-Processing:
  - Validate dna to ensure it contains only valid DNA bases (A, C, G, T). If invalid, set an error flag or message.
- 4. Initial State Logging:
  - Optionally, log the initial state of variables for debugging or tracking purposes