

The most occurring section B3 task

Specification :

Input:

$\text{dna} \in T$

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

Precondition: $\text{length}(\text{dna}) \geq 255$

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

$\forall c \in \text{dna}, c \in \{A, C, G, T\}$ (every character in dna is one of A, C, G, T)

$\text{mostFrequentSequence} = ""$ ($\text{mostFrequentSequence}$ is initialized as an empty string)

$\text{highestCount} = 0$ (highestCount is initialized to zero).

Postcondition: $1 \leq \text{length}(\text{mostFrequentSequence}) \leq \text{length}(\text{dna})$

and $\text{length}(\text{mostFrequentSequence}) \geq 3$

$\text{highestCount} = \max(\text{count}(s) \mid s \subseteq \text{dna}, \text{length}(s) \geq 3)$.

$\forall s (s \subseteq \text{dna}, \text{length}(s) \geq 3) \Rightarrow \text{count}(s) \leq \text{highestCount}$.

$$\text{count} = \sum_{i=1}^{\text{length}(\text{dna})} 1_{A(\text{isMatch})}$$

General Postcondition(maximum item selection + multiple item selection)

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

$$\text{cnt} = \sum_{i=1}^{\text{length}(X)} 1_{A(X[i])}$$

Pattern: Maximum selection and Counting

Algorithm

```
maxVal := X[1]
```

Pattern	Task
X	dna
i	start (loop index for substring start)
$i \leq \text{length}(X)$	start < dna.Length
$i := i + 1$	start++
exists	highestCount > 1
exists := ($i \leq \text{length}(X)$)	highestCount > 1 (after loop completion)

argument	
maxVal := X[1]	
i = 2 .. length(X)	
T	X[i] > maxVal
maxVal := X[i]	-
cnt := 0	
i = 1 .. length(X)	
T	X[i] = maxVal
cnt := cnt + 1	-
maxI[cnt] := i	

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

The most occurring section	
In := dna	
start = start..Length(dna) - 2	
length = (start + length)..Length(dna)	
count := 0	
currentSequence := ""	
k = k..length	
currentSequence := currentSequence + dna[start + k]	
index ..Length(dna)-length + 1	
isMatch := true	
matchIndex = matchIndex..length	
T	dna[start + matchIndex] ≠ dna[index + matchIndex]
isMatch := false	∅
isMatch	
T	
count ++	∅
index := index + length - 1	
count > highestCount or (count == highestCount and currentSequence.Length > mostFrequentSequence.Length	
T	
mostFrequentSequence := currentSequence	∅
highestCount := count	
OUTPUT := highestCount > 1 ? \$"{mostFrequentSequence} {highestCount}": "-1"	

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