

Gene Folding

| | | |
|--------------|--------------|-------|
| R.varshini | (21B01A05F3) | CSE-C |
| k.Harshini | (21B01A0587) | CSE-B |
| ch chaitanya | (21B01A0535) | CSE-A |
| K.Akhila | (21B01A1287) | IT-B |
| K.Sushma | (21B01A1288) | IT-B |

Date:10-02-2023

Introduction

- Find a point between two adjacent nucleotides in the nucleotide sequence.
- such that the sequence reads the same from that point in both directions, up until the nearer end of the sequence.
- Then fold the genetic sequence at that point, merging the identical nucleotides. Through repeated process of GOOF, a nucleotide can potentially be made much shorter.
- finally retrieve the length of the shorter sequence.

Approach

- In our approach we have taken three functions. 1.valid 2.goof 3.checkfold
the valid function we are checking the acceptance list the characters is present or not .Because nucleotides should be there in the given input. In the checkfold function we are finding the mid value for checking the foldings are available or not .And after it will check either mid is equal to -1 or not
- In goof function we are assigning the initial sequence into sequence and calling the checkfold(sequence) function

Learnings

- optimize the process of finding the folding points in a given input sequence and choosing them so as to obtain the shortest possible genetic sequence
- Develop skills in string manipulation, pattern recognition, and algorithm optimization.
- As a variant of the Longest Palindromic Substring problem, this problem is a good example of how a small variant can lead to a new problem with different applications.

Challenges

- Understanding the problem.
- Finding the appropriate folding points in the sequence was time-consuming and complex
- Choosing the folding point that results in the shortest possible genetic sequence.
- Automating the process of finding the folding points and choosing them.

Statistics

- There are 36 lines of code.

- Number of Functions

To finish the code we use 3 functions.

There are:

1.valid()

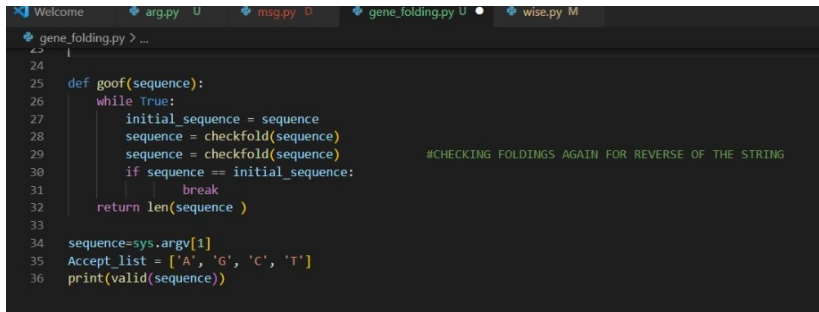
2.checkfold()

3.goof()

Demo/ Screen Shots

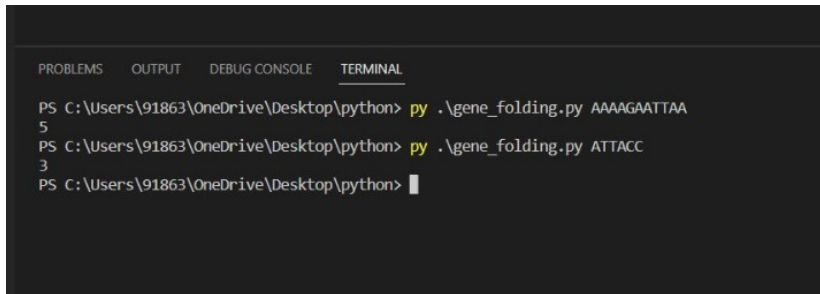
```
gene_folding.py > ...
1  import sys
2  def valid(sequence):
3      flag = 1
4      for ele in sequence:
5          if ele not in Accept_list:
6              flag = 0
7              break
8      if flag == 0:
9          return -1
10     else:
11         return goof(sequence)
12
13
14 def checkfold(sequence):
15     mid = len( sequence ) // 2
16     while mid != -1 :
17         if sequence[:mid][::-1] == sequence[mid: 2 * mid] :
18             sequence = sequence [mid: ]
19             mid = -1
20         else:
21             mid -= 1
22     return sequence[::-1]
23
24
```

Demo/ Screen Shots



```
24
25 def goof(sequence):
26     while True:
27         initial_sequence = sequence
28         sequence = checkfold(sequence)
29         sequence = checkfold(sequence)           #CHECKING FOLDINGS AGAIN FOR REVERSE OF THE STRING
30         if sequence == initial_sequence:
31             break
32     return len(sequence )
33
34 sequence=sys.argv[1]
35 Accept_list = ['A', 'G', 'C', 'T']
36 print(valid(sequence))
```


Demo/ Screen Shots



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL being the active tab. The terminal shows the following commands and outputs:

```
PS C:\Users\91863\OneDrive\Desktop\python> py .\gene_folding.py AAAAGAATTAA
5
PS C:\Users\91863\OneDrive\Desktop\python> py .\gene_folding.py ATTACC
3
PS C:\Users\91863\OneDrive\Desktop\python> 
```

THANK YOU