Universidade Federal do Rio Grande do Sul - Instituto de Informática
INF05018 Biologia Computacional - Prof: Márcio Dorn

Rodrigo Okido (252745)

Relatório E1-I1

Porto Alegre
2018

Sumário

1. **Introdução**

Este relatório tem como objetivo demonstrar a resolução dos exercícios da lista I (E1-I1). Será colocado o pseudocódigo de cada exercício assim como o resultado esperado para cada um.

2. **Exercícios lista 1**

Considerando que **todos** os exercícios desta primeira lista envolvem a leitura do arquivo "sequence.fasta", será explicado inicialmente como foi realizado isto, considerando que a mesma se repetirá para todos. A leitura é realizada ignorando a primeira linha e os "\n" da seguinte forma:

```
Open ("sequence.fasta") no arquivo:
    Pula primeira linha
    Para cada linha em arquivo
        Linha = le e remove \n
        Conteudo = conteudo + linha
```

a. **No primeiro exercício foi implementado da seguinte forma:**

```
Para i em sentença
    Tentativa = lista ( sentença )
    Se tentativa [ i ] == T | C | A | G
        Mudar para sua complementar
        Junta a palavra e tenta encontrar essa sentença no arquivo.
        Se Encontrou:
            Imprime Resposta
        Caso contrário:
            Tentativa = Sentença
```

```
Resposta Encontrada:
    Subsequence starts at index:  44139800
    Mutation happened at index:  44139809
    Changed: A -> T
```

b. **O segundo exercício foi implementado da seguinte forma:**

```
Function complement_checkpalindrome ( subsequence ) :
    Original = subsequence
    Modified = list (subsequence)
```

```
        para i em len(modified)
            Se modified [i] ==  A| T | C | G
                Muda para sua complementar

        Se original == reversa(modified)
            Retorna True
        Se não:
            Retorna False

    Function main ( ) :
        Le arquivo
        j = 0
        Para i em arquivo
            Subseq = arquivo [ j:j+8 ]
            Se não contem "N" na subseq e tamanho == 8
                Se complement_checkpalindrome (subseq) == True:
                    Adiciona num dicionario caso não existir
                Caso exista:
                    Adiciona mais um ao valor (Pegando a chave)
            j = j + 1

        Resultado Encontrado:
            {'GCTCGAGC': 143, 'GCTATAGC': 366, 'CTTATAAG': 1182,
'TGATATCA': 1071, 'TCTGCAGA': 1943, 'CCGTACGG': 13, 'GTTGCAAC': 377,
'GCATATGC': 693, 'AGAATTCT': 2246, 'ACACGTGT': 552, 'TATATATA': 16514,
'CTTCGAAG': 150, 'TCACGTGA': 361, 'TGCATGCA': 1550, 'GGAATTCC': 826,
'CCCATGGG': 920, 'CGCCGGCG': 92, 'ACCTAGGT': 614, 'TCGGCCGA': 37,
'TTTATAAA': 5759, 'GTCATGAC': 457, 'ATGTACAT': 1684, 'ATTGCAAT': 1176,
'CTCATGAG': 1231, 'AGGATCCT': 763, 'GAGTACTC': 323, 'CAGGCCTG': 2231,
'GTAATTAC': 765, 'AGCATGCT': 984, …..}
```

c. O terceiro exercício foi implementado da seguinte forma:

```
        Function acha_todos (all_string, substring) :
            Start = 0
            Enquanto True:
                Encontra Substring na string
                Adiciona index
```

```
Function main ( ):
        Lista_subsequencias = { }
        Para cada i em conteudo_arquivo:
                Subseq = conteudo_arquivo [j : j + 37]
                Se não contem “N” e tamanho == 37
                        Acha_todos (conteudo_arquivo, subseq)
                        Se não existe no dicionario
                                Adiciona novo valor no dicionario com sua
                                quantidade
```

Resultado Encontrado:

```
{'ACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAA':                3,
'CCCTAACCCTAACCCTAACCCTAACCCTAACCCTAAC':                 3,
'CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACC':                 2,
'AACCCTAACCCTAACCCTAACCCTAACCCTAACCCTA':                 3,
'CTAACCCTAACCCTAACCCTAACCCTAACCCTAACCC':                 3,
'TAACCCTAACCCTAACCCTAACCCTAACCCTAACCCT':                 3,
'CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACA':                 1,
'CTAACCCTAACCCTAACCCTAACCCTAACCCTAACAA':
1,
}
…… (Muito grande)
```

d. A letra (d) foi implementada da seguinte forma:

```
qtdA, qtdT, qtdC,qtdG, qtdN = 0
Para i em (conteudo_arquivo)
        Se for C | T | A | G | N
                Adiciona sua respectiva quantidade.
```

Resultado Encontrado:

```
        Quantidades:
         C -  11918693
         G -  11909449
         T -  17132531
         A -  17146584
        Caracter Diferente - SIM:
         N -  2396
```

e. O ultimo exercício foi implementado da seguinte forma:

```
Function find_all - Identico ao usado no exercicio C.
Function generate_sequence (registry):
        Sequence = list(registro)
        Para i em range (sequence)
                Se for [0 : 9]
                        Atribui uma letra
                Se não
                        Erro
        Retorna sequence

Function complement (sequence)
        Modified = list(sequencia)
        Para cada i em len(modificada)
                Se modified [i] == A | T | C | G
                        Modified [ i ] = sua complementar

        Join (modified)
        Return modified


Function mutate_reg_complement(sequence)
        Mutated_list = [ ]
        getSequence = list(sequence)

        Se getSequence [4] = A | T | C | G
                Cria 3 modificações em cima da quinta posição.
                Adiciona na mutated_list

        Retorna mutated_list

Function main () :
        Registry = “N CARTAO”
        Var1 = generate_sequence(registry) #Generate sequence of the registry
        Var2 = complement(var1) #Generate the complement of the sequence
        Var3 = mutate_reg_complement(var2) #Generate 3 mutation on 5th position (list)

counter_mutated = {}

Para i em len(var3)
```

```
        Mutated = var3[i] #Take each mutated sequence of the list
        Var4 = list(find_all (conteudo_arquivo, mutated)
        Consulta dicionario
        Se == NONE
                Adiciona nova chave com todas suas possibilidades

return counter_mutated. #Return dictionary
```

Resultado Encontrado:

```
        AAGGGACG
        TTCCCTGC
        ['TTCCATGC', 'TTCCGTGC', 'TTCCTTGC']
        {'TTCCTTGC': 1363, 'TTCCGTGC': 118, 'TTCCATGC': 1198}
```