Universidade Federal do Rio Grande do Sul
Instituto de Informática
Prof. Dr. Márcio Dorn
Lista 1 de Biología Computacional

Aluno: Vicente Merlo
Matrícula: 244950

Questão 1.A.

Criei uma função chamada *distance* que retorna um inteiro correspondente à distância entre a sequência de 20 caracteres à subsequência XXX. Caso a distância relativa seja maior que 1 em algum dos nucleotídeos, ele já para o cálculo como uma otimização, e retorna o número 2 (que deve ser a distância relativa naquele momento).
Depois, abri o arquivo *sequence.fasta* e realizei uma busca por todas as subsequências de tamanho 20, indo até o final do arquivo, e aplicando a função distance em cada subsequência, e imprimo na tela a subsequência e a posição dela no arquivo.
Assim, consigo listar todas as mutações possíveis de 1 nucleotídeo (subsequências com distância == 1) no arquivo.

Resultado:

```
python e1-1a.py squence.fasta
Sequencia CAGGAGATCATCGTGGCCAC em 44139800
```

Questão 1.B.

Criei duas funções, a *complementar* e a *is_palindromo*.
A função *complementar* retorna a subsequência complementar à subsequência dada, seguindo as regras descritas pelo professor em aula.
A função *is_palindromo* verifica se uma string invertida é igual a outra.
Percorro o arquivo de 10 em 10 subsequências, e executo um teste condicional *is_palindromo(complementar(subsequencia), sequencia)*. Sempre que for verdadeiro, adiciono em uma lista para posteriormente imprimir.

Resultado:

```
python e1-1b.py sequence.fasta
CGGTCGACCG 1
CAGCATGCTG 94
TATGCGCATA 7
ATGTCGACAT 10
GTTATATAAC 67
AGGATATCCT 32
GCCAATTGGC 20
GGTCCGGACC 3
```

(resultado completo em e1-1b.out)

Questão 1.C.

Criei um hash chamado *contador* e passei por toda a sequência em subsequências de 37 nucleotídeos.

Sempre que encontrava uma subsequência nova, a adicionava na hash. Quando a subsequência já existia, incrementava o seu valor na hash.

Assim, passei por todas subsequências de tamanho 37 e obtive quantas vezes elas se repetiam.

Como estou usando um laptop com recursos computacionais limitados, precisei rodar o algoritmo em um servidor com 16 gb de RAM, devido ao tamanho do hash resultante.

Resultado:

GGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGC: 507
GGCCTCCCAAAGTGCTGGGATTACAGGCGTGAGCCAC: 531
GTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGCC: 531
TATATATATATATATATATATATATATATATATATAT: 611
ATATATATATATATATATATATATATATATATATATA: 624
GTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTG: 765
TGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGT: 837
CACACACACACACACACACACACACACACACACACAC: 869
ACACACACACACACACACACACACACACACACACACA: 949
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN: 2360

(resultado em e1-1c.out)

Questão 1.D.

Criei um hash com entradas para cada um dos 4 nucleotídeos conhecidos, que serve como contador.

Passei por todos os nucleotídeos da sequência e sempre que encontrava um nucleotídeo presente no contador, o incrementava. Caso contrário, o adicionava à uma lista de nucleotídeos desconhecidos.

Resultado:

python e1-1d.py sequence.fasta
Contadores: {'A': 17146584, 'C': 11918693, 'T': 17132531, 'G': 11909449}
Sim, caracteres diferentes encontrados: ['N']

Questão 1.E.

Criei a função *gerar_matricula*, que gera uma sequência para uma dada matrícula, seguindo a tabela de mapeamento da questão, e a função *mutar*, que retorna uma lista com todas as mutações possíveis para o nucleotídeo na posição 5 de uma dada sequência.

Então, faço uma busca por todas as subsequências de tamanho 8 na sequência do cromossomo 7, procurando por alguma das subsequências que gerei.

Resultado:

```
python e1-1e.py sequence.fasta 00244950
Sequencias geradas: ['TTCGGGCT', 'AAGCACGA', 'AAGCGCGA', 'AAGCTCGA', 'AAGCCCGA']
AAGCCCGA 108
AAGCACGA 165
TTCGGGCT 108
AAGCGCGA 20
AAGCTCGA 124
```