

Curso Bacharelado em Ciência da Computação

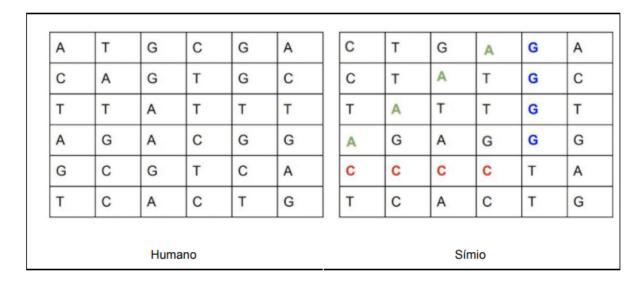
Disciplina: Algoritmos Aluno: Matheus Possenti Docente: Manassés Ribeiro

RELATÓRIO DE DESENVOLVIMENTO DO TRABALHO FINAL SÍMIOS

1. INTRODUÇÃO

O presente relatório tem como objetivo apresentar a linha de raciocínio seguida para implementar em C um algoritmo que detectas uma sequência de caracteres de um DNA e verifica se o DNA pertence a um humano ou a um símio.

O algortimo recebe uma sequência de DNA, que é representado por uma matriz, e deve fazer a verificação para encontrar uma sequência de 4 caracteres nas direções horizontais, verticais e diagonais, como é mostrado na imagem abaixo.



2. METODOLOGIA

Para iniciar a implementação do algoritmo, é necessário criar uma função que receba a sequência do DNA e carregue a matriz. Para isso o algoritmo pede ao usuário quantas linhas e colunas a matriz terá e assim alocando memória para a matriz nesse tamanho.

```
1 // Preenche a matriz linha por linha
2 void carregarMatriz(char** mat, int tam) {
3   int i, j, k;
4   for (i = 0; i < tam; i++) {
5     char* sequencia = alocaMemoriaVetor(tam);
6     carregaVetor(sequencia, i, tam);
7     k = 0;
8     for(j = 0; j < tam; j++, k++) {
9         mat[i][j] = sequencia[k];
10     }
11     free(sequencia);
12   }
13 }</pre>
```

```
char* carregaVetor(char* sequencia, int i, int tam) {
  printf("Informe a sequencia de DNA da linha " GRN "%d/%d \n" RESET, i+1 , tam);
  scanf("%s", sequencia);
}
```

Logo, a função chama outra função que pergunta ao usuário a sequência do DNA, linha por linha. O exemplo a seguir mostra como o carregamento da matriz funciona e deve ser feito para o programa funcionar.

```
Informe a quantidade de linhas/colunas da matriz: 6

Preencha a matriz com os caracteres (A, T, G, C)
Informe a sequencia de DNA da linha 1/6
AACCAA
Informe a sequencia de DNA da linha 2/6
AACCAA
```

Para verificar a sequência de 4 caracteres na matriz do DNA, podemos seguir a lógica de percorrer a matriz verificando apenas uma área 4x4, até que chegue ao final da matriz.

Isso pode ser feito utilizando 2 estruturas de repetição, que passam como parâmetro para as funções de verificação, qual a posição da área 4x4 em cima da matriz que deve ser feita a verificação. Como deve ser analisada apenas uma área 4x4, o for irá repetir até o tamanho - 3, logo que não terá mais uma área 4x4 se passar disso.

```
int isSimian(char** mat, int tam) {
  int is imian = 0;
  int simian = 0;

// Como deve ser analisado uma area de 4x4, o for vai somente até coluna/linha -3
  for (i = 0; i < tam - 3; i++) {
    for (j = 0; j < tam - 3; j++) {
        simian = analisalinha(mat, i, j);
        if (simian == 1) return 1;

        simian = analisaColuna(mat, i, j);
        if (simian == 1) return 1;

        simian = analisaDiagonalP(mat, i, j);
        if (simian == 1) return 1;

        simian = analisaDiagonalS(mat, i, j);
        if (simian == 1) return 1;

        return simian;
}
</pre>
```

Para verificação da linha, um for itera sobre as linhas da área 4x4 comparando se os elementos são iguais. Para verificação da coluna, é basicamente a mesma coisa, porém o for itera sobre as colunas, comparando os elementos. As funções de verificação das diagonais, recebem a linha e coluna atual, e verificam os elementos da diagonal principal e secundária.

3. CONCLUSÃO

Com o desenvolver do projeto foi possível colocar em prática e estudar grande parte dos conteúdos abordados em sala de aula. Foi necessário entender o problema e analisar uma forma otimizada e eficiente de resolvê-lo. No decorrer do desenvolvimento vários outros problemas apareceram, sendo assim necessário consulta da internet, documentações e etc. Ao executar o programa recebemos como resultados 2 possibilidades, Símio ou Humano. Como mostra a imagem abaixo:

```
Matriz de DNA:

A| C| T| G|
F| T| G| A|
G| A| C| T|
A| A| G| T|

O usu|írio | humano
Fim do Programa!
```