**UNIVERSIDADE FEDERAL DE VIÇOSA**

**CAMPUS FLORESTAL**

PABLO FERREIRA - 3480

SAMUEL SENA - 3494

**TRABALHO PRÁTICO III**

**RELATÓRIO TAREFA A e B**

FLORESTAL

2019

Sumário

[Introdução 3](#_Toc24317546)

[Comparação de tempos de execução 10](#_Toc24317547)

[Modo DEBUG 1](#_Toc24317549)1

[Desenvolvimento 11](#_Toc24317550)

[Conclusão 14](#_Toc24317551)

# Introdução

O trabalho apresentado a seguir entrega 2 algoritmos, escolhidos pelo aluno, de casamento exato de padrões e também fazer a análise de tempo dos algoritmos, construir gráficos e descreve-los (Tarefa A e B).

Inicialmente para se executar o programa da Tarefa A e B, é necessário realizar a compilação do código fonte em C. Para isso, em algum terminal Linux execute o “*makefile”* da seguinte forma:

Para compilar:

$ make

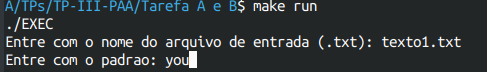
E para executar:

$ make run

O algoritmo foi testado apenas em sistema operacional baseado em Linux, a execução em Windows pode não ser satisfatória.

Primeiramente, entre com o nome do arquivo de entrada desejado, em seguida, caso a abertura seja realizada com sucesso, entre com o padrão desejado, segue abaixo:

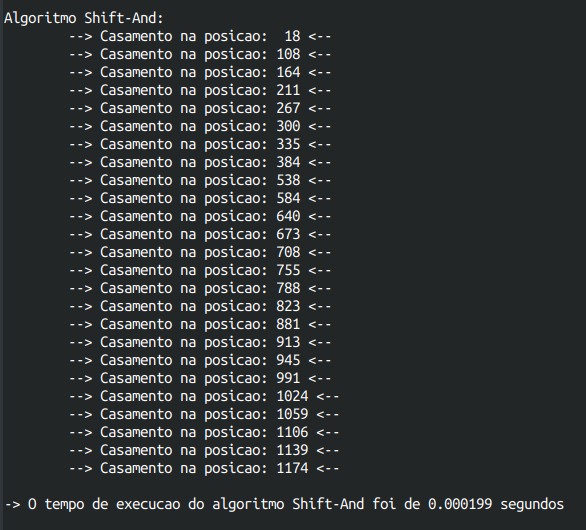
Figura 2



Fonte: Terminal Linux

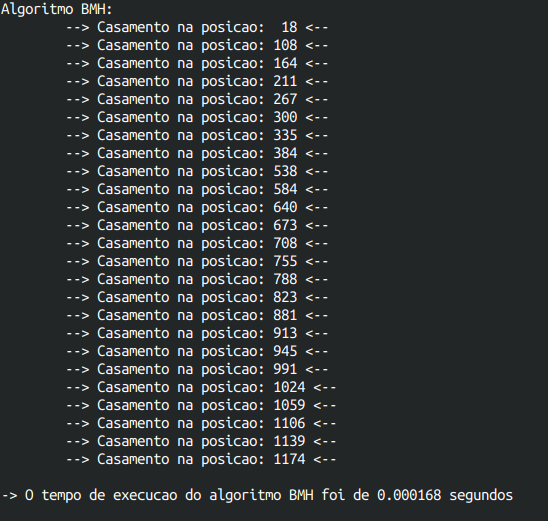
Após a escolha desejada e execução da mesma, o programa irá sempre finalizar em seguida e apresentar os resultados. Foram ultilizados um texto em inglês e procuramos o padrão “you”, sendo 3 testes onde duplicamos o texto base para aumentar o número de ocorrências. Segue abaixo alguns resultados:

Figura 3



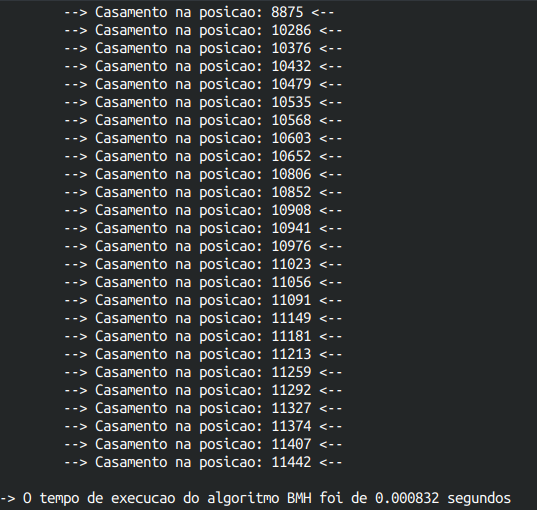
Fonte: Terminal Linux

Figura 4



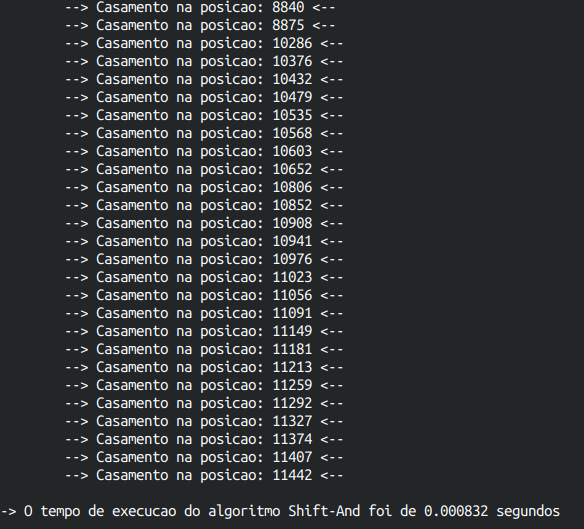
Fonte: Terminal Linux

Figura 5



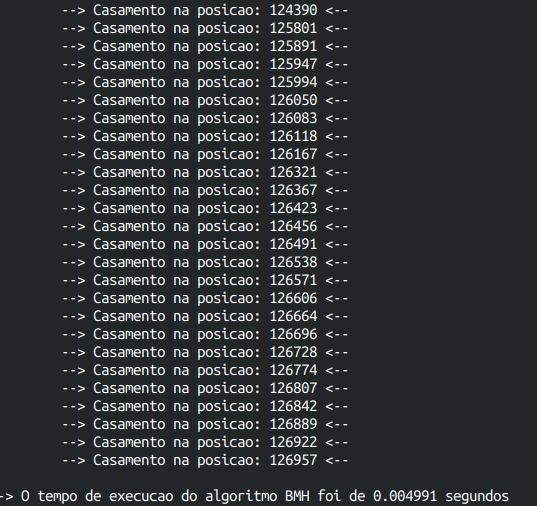
Fonte: Terminal Linux

Figura 6



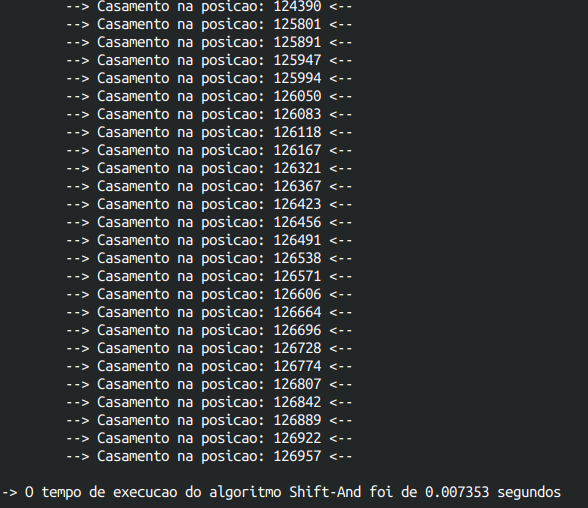
Fonte: Terminal Linux

Figura 7



Fonte: Terminal Linux

Figura 8



Fonte: Terminal Linux

# Comparação de tempos de execução

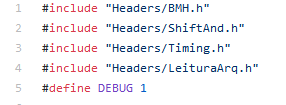
Os gráficos a seguir servem de comparação entre o desempenho dos algoritmos utilizados:

Como podemos perceber, quando começamos a aumentar para um número muito alto de caracteres, percebemos que o Shift And exato tem um desempenho pior em relação ao BMH.

# Modo DEBUG

O programa conta com a opção de utilização em modo “debug” que (quando ativado) exibe na tela o tempo de execução de cada algoritmo escolhido, para utiliza-lo, o valor de “DEBUG” no arquivo “main.c” deve ser definido para o 1, caso seja definido como 0, o programa ira rodar sem exibir o tempo de execução de cada algoritmo. A figura abaixo demonstra o local da definição do valor de “DEBUG”:

Figura 9



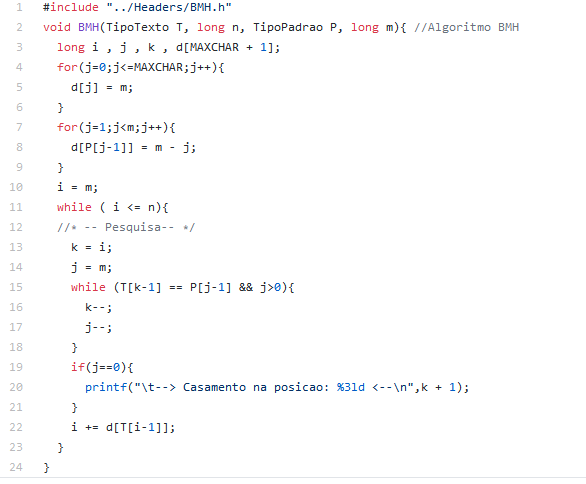
Fonte: main.c

# Desenvolvimento

**Tarefa A:**

Os algoritmos escolhidos pela dupla foram o BMH e o Shift And aproximado, onde o BMH consiste em um programa que pesquisa um sufixo em um texto com o sufixo padrão, fazendo comparações da direita para esquerda, caso não ocorra nenhuma desigualdade entre eles, então foi encontrada uma ocorrência do padrão no texto. Se ocorrer desigualdade, o programa calcula um deslocamento onde o padrão deve ser deslocado para a direita antes que uma nova tentativa de casamento comece. Abaixo o código do BMH:

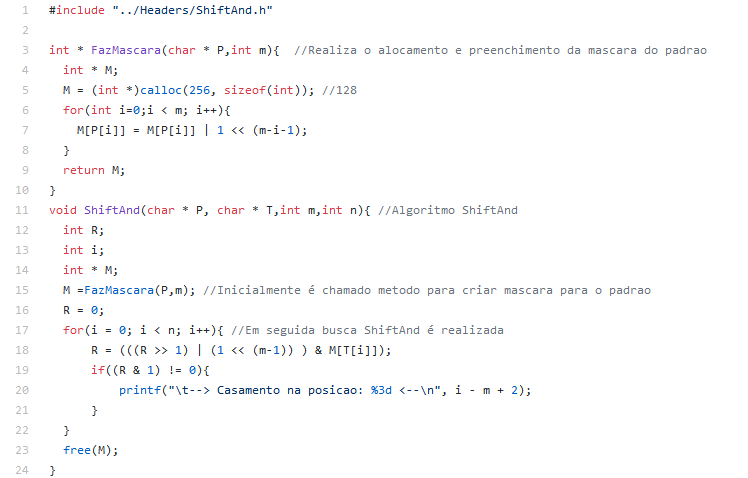
Figura 10



Fonte: Terminal Linux

Já no caso do algoritmo Shift And aproximado, no inicio é criada uma mascara para cada caractere do alfabeto, e então o vetor padrão é percorrido e para cada letra encontrada, é atribuído o valor 1 na posição da máscara correspondente, e na iniciação dos vetores R e R’, é feita da seguinte forma, R é o vetor que receberá R’ deslocado a direita a cada iteração e R’ é o vetor que receberá o & bit a bit entre o vetor R e a máscara correspondente à letra procurada. Esse processo é repetido para as demais letras, até que haja casamento ou o vetor do texto seja totalmente percorrido. O algoritmo segue abaixo:

Figura 11



Fonte: Terminal Linux

# Conclusão

Sem dúvidas, o desenvolvimento desse trabalho foi de importância para o aprendizado da matéria e do funcionamento dos algoritmos apresentados em sala

Agradecimentos ao professor Daniel Mendes pela oportunidade de realização do trabalho e dúvidas sanadas.

Todo o desenvolvimento e distribuição do trabalho encontra-se hospedado na seguinte página do [GitHub](https://github.com/Globson/TP-III-PAA).