

**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
Comparação de tempos de execução	10
Modo DEBUG	11
Desenvolvimento	11
Conclusão	14

## 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

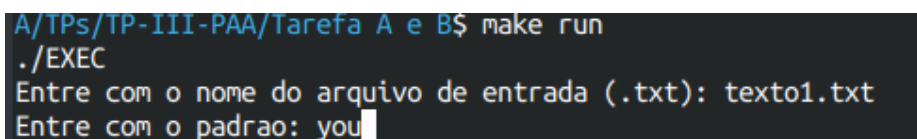
E para executar:

\$ 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



```
A/TPs/TP-III-PAA/Tarefa A e B$ make run
./EXEC
Entre com o nome do arquivo de entrada (.txt): texto1.txt
Entre com o padrao: you
```

Fonte: Terminal Linux

Após a escolha desejada e execução da mesma, o programa irá sempre finalizar em seguida e apresentar os resultados. Foram utilizados 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

```
Algoritmo Shift-And:
--> Casamento na posicao: 18 <--
--> Casamento na posicao: 108 <--
--> Casamento na posicao: 164 <--
--> Casamento na posicao: 211 <--
--> Casamento na posicao: 267 <--
--> Casamento na posicao: 300 <--
--> Casamento na posicao: 335 <--
--> Casamento na posicao: 384 <--
--> Casamento na posicao: 538 <--
--> Casamento na posicao: 584 <--
--> Casamento na posicao: 640 <--
--> Casamento na posicao: 673 <--
--> Casamento na posicao: 708 <--
--> Casamento na posicao: 755 <--
--> Casamento na posicao: 788 <--
--> Casamento na posicao: 823 <--
--> Casamento na posicao: 881 <--
--> Casamento na posicao: 913 <--
--> Casamento na posicao: 945 <--
--> Casamento na posicao: 991 <--
--> Casamento na posicao: 1024 <--
--> Casamento na posicao: 1059 <--
--> Casamento na posicao: 1106 <--
--> Casamento na posicao: 1139 <--
--> Casamento na posicao: 1174 <--

-> O tempo de execucao do algoritmo Shift-And foi de 0.000199 segundos
```

Fonte: Terminal Linux

Figura 4

Algoritmo BMH:

```
--> Casamento na posicao: 18 <--  
--> Casamento na posicao: 108 <--  
--> Casamento na posicao: 164 <--  
--> Casamento na posicao: 211 <--  
--> Casamento na posicao: 267 <--  
--> Casamento na posicao: 300 <--  
--> Casamento na posicao: 335 <--  
--> Casamento na posicao: 384 <--  
--> Casamento na posicao: 538 <--  
--> Casamento na posicao: 584 <--  
--> Casamento na posicao: 640 <--  
--> Casamento na posicao: 673 <--  
--> Casamento na posicao: 708 <--  
--> Casamento na posicao: 755 <--  
--> Casamento na posicao: 788 <--  
--> Casamento na posicao: 823 <--  
--> Casamento na posicao: 881 <--  
--> Casamento na posicao: 913 <--  
--> Casamento na posicao: 945 <--  
--> Casamento na posicao: 991 <--  
--> Casamento na posicao: 1024 <--  
--> Casamento na posicao: 1059 <--  
--> Casamento na posicao: 1106 <--  
--> Casamento na posicao: 1139 <--  
--> Casamento na posicao: 1174 <--
```

-> O tempo de execucao do algoritmo BMH foi de 0.000168 segundos

Fonte: Terminal Linux

Figura 5

```
--> Casamento na posicao: 8875 <--  
--> Casamento na posicao: 10286 <--  
--> Casamento na posicao: 10376 <--  
--> Casamento na posicao: 10432 <--  
--> Casamento na posicao: 10479 <--  
--> Casamento na posicao: 10535 <--  
--> Casamento na posicao: 10568 <--  
--> Casamento na posicao: 10603 <--  
--> Casamento na posicao: 10652 <--  
--> Casamento na posicao: 10806 <--  
--> Casamento na posicao: 10852 <--  
--> Casamento na posicao: 10908 <--  
--> Casamento na posicao: 10941 <--  
--> Casamento na posicao: 10976 <--  
--> Casamento na posicao: 11023 <--  
--> Casamento na posicao: 11056 <--  
--> Casamento na posicao: 11091 <--  
--> Casamento na posicao: 11149 <--  
--> Casamento na posicao: 11181 <--  
--> Casamento na posicao: 11213 <--  
--> Casamento na posicao: 11259 <--  
--> Casamento na posicao: 11292 <--  
--> Casamento na posicao: 11327 <--  
--> Casamento na posicao: 11374 <--  
--> Casamento na posicao: 11407 <--  
--> Casamento na posicao: 11442 <--  
  
-> O tempo de execucao do algoritmo BMH foi de 0.000832 segundos
```

Fonte: Terminal Linux

Figura 6

```
--> Casamento na posicao: 8840 <--  
--> Casamento na posicao: 8875 <--  
--> Casamento na posicao: 10286 <--  
--> Casamento na posicao: 10376 <--  
--> Casamento na posicao: 10432 <--  
--> Casamento na posicao: 10479 <--  
--> Casamento na posicao: 10535 <--  
--> Casamento na posicao: 10568 <--  
--> Casamento na posicao: 10603 <--  
--> Casamento na posicao: 10652 <--  
--> Casamento na posicao: 10806 <--  
--> Casamento na posicao: 10852 <--  
--> Casamento na posicao: 10908 <--  
--> Casamento na posicao: 10941 <--  
--> Casamento na posicao: 10976 <--  
--> Casamento na posicao: 11023 <--  
--> Casamento na posicao: 11056 <--  
--> Casamento na posicao: 11091 <--  
--> Casamento na posicao: 11149 <--  
--> Casamento na posicao: 11181 <--  
--> Casamento na posicao: 11213 <--  
--> Casamento na posicao: 11259 <--  
--> Casamento na posicao: 11292 <--  
--> Casamento na posicao: 11327 <--  
--> Casamento na posicao: 11374 <--  
--> Casamento na posicao: 11407 <--  
--> Casamento na posicao: 11442 <--  
  
-> 0 tempo de execucao do algoritmo Shift-And foi de 0.000832 segundos
```

Fonte: Terminal Linux

Figura 7

```
--> Casamento na posicao: 124390 <--  
--> Casamento na posicao: 125801 <--  
--> Casamento na posicao: 125891 <--  
--> Casamento na posicao: 125947 <--  
--> Casamento na posicao: 125994 <--  
--> Casamento na posicao: 126050 <--  
--> Casamento na posicao: 126083 <--  
--> Casamento na posicao: 126118 <--  
--> Casamento na posicao: 126167 <--  
--> Casamento na posicao: 126321 <--  
--> Casamento na posicao: 126367 <--  
--> Casamento na posicao: 126423 <--  
--> Casamento na posicao: 126456 <--  
--> Casamento na posicao: 126491 <--  
--> Casamento na posicao: 126538 <--  
--> Casamento na posicao: 126571 <--  
--> Casamento na posicao: 126606 <--  
--> Casamento na posicao: 126664 <--  
--> Casamento na posicao: 126696 <--  
--> Casamento na posicao: 126728 <--  
--> Casamento na posicao: 126774 <--  
--> Casamento na posicao: 126807 <--  
--> Casamento na posicao: 126842 <--  
--> Casamento na posicao: 126889 <--  
--> Casamento na posicao: 126922 <--  
--> Casamento na posicao: 126957 <--  
  
-> O tempo de execucao do algoritmo BMH foi de 0.004991 segundos
```

Fonte: Terminal Linux

Figura 8

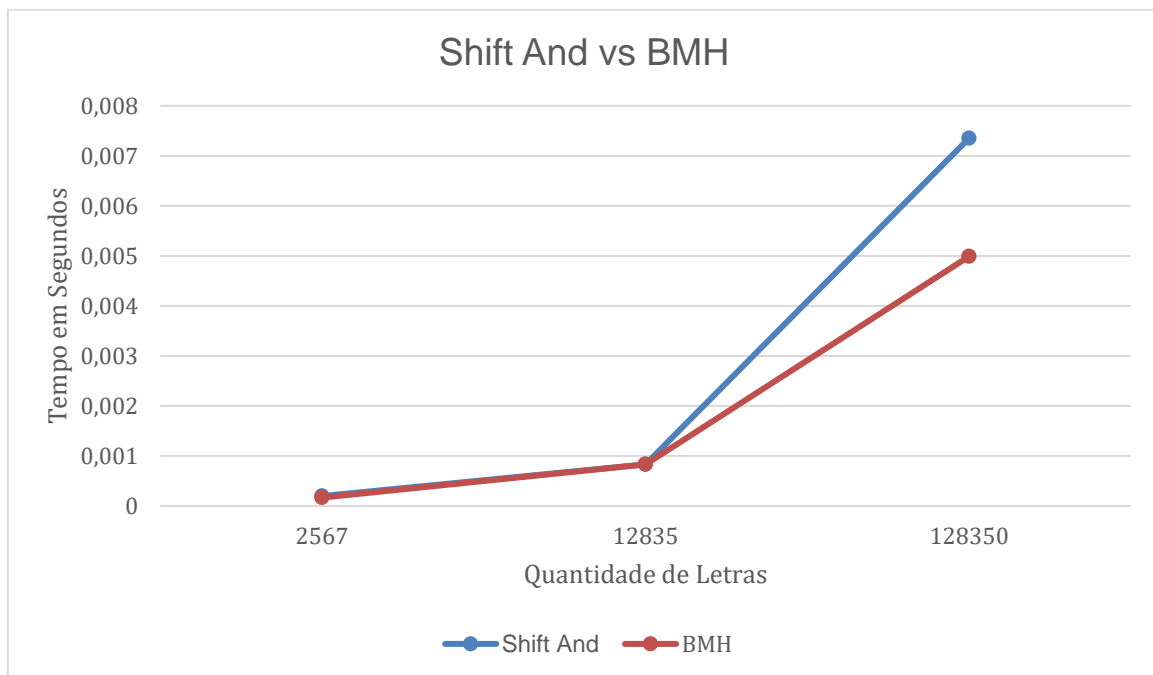


```
--> Casamento na posicao: 124390 <--  
--> Casamento na posicao: 125801 <--  
--> Casamento na posicao: 125891 <--  
--> Casamento na posicao: 125947 <--  
--> Casamento na posicao: 125994 <--  
--> Casamento na posicao: 126050 <--  
--> Casamento na posicao: 126083 <--  
--> Casamento na posicao: 126118 <--  
--> Casamento na posicao: 126167 <--  
--> Casamento na posicao: 126321 <--  
--> Casamento na posicao: 126367 <--  
--> Casamento na posicao: 126423 <--  
--> Casamento na posicao: 126456 <--  
--> Casamento na posicao: 126491 <--  
--> Casamento na posicao: 126538 <--  
--> Casamento na posicao: 126571 <--  
--> Casamento na posicao: 126606 <--  
--> Casamento na posicao: 126664 <--  
--> Casamento na posicao: 126696 <--  
--> Casamento na posicao: 126728 <--  
--> Casamento na posicao: 126774 <--  
--> Casamento na posicao: 126807 <--  
--> Casamento na posicao: 126842 <--  
--> Casamento na posicao: 126889 <--  
--> Casamento na posicao: 126922 <--  
--> Casamento na posicao: 126957 <--  
  
-> O tempo de execucao do algoritmo Shift-And foi de 0.007353 segundos
```

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

```
1  #include "Headers/BMH.h"
2  #include "Headers/ShiftAnd.h"
3  #include "Headers/Timing.h"
4  #include "Headers/LeituraArq.h"
5  #define DEBUG 1
-
```

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

```
1  #include "../Headers/BMH.h"
2  void BMH(TipoTexto T, long n, TipoPadrao P, long m){ //Algoritmo BMH
3      long i , j , k , d[MAXCHAR + 1];
4      for(j=0;j<=MAXCHAR;j++){
5          d[j] = m;
6      }
7      for(j=1;j<m;j++){
8          d[P[j-1]] = m - j;
9      }
10     i = m;
11     while ( i <= n){
12         /* -- Pesquisa-- */
13         k = i;
14         j = m;
15         while (T[k-1] == P[j-1] && j>0){
16             k--;
17             j--;
18         }
19         if(j==0){
20             printf("\t--> Casamento na posicao: %3ld <--\n",k + 1);
21         }
22         i += d[T[i-1]];
23     }
24 }
```

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

```

1  #include "../Headers/ShiftAnd.h"
2
3  int * FazMascara(char * P,int m){ //Realiza o alocamento e preenchimento da mascara do padrao
4      int * M;
5      M = (int *)calloc(256, sizeof(int)); //128
6      for(int i=0;i < m; i++){
7          M[P[i]] = M[P[i]] | 1 << (m-i-1);
8      }
9      return M;
10 }
11 void ShiftAnd(char * P, char * T,int m,int n){ //Algoritmo ShiftAnd
12     int R;
13     int i;
14     int * M;
15     M =FazMascara(P,m); //Inicialmente é chamado metodo para criar mascara para o padrao
16     R = 0;
17     for(i = 0; i < n; i++){ //Em seguida busca ShiftAnd é realizada
18         R = (((R >> 1) | (1 << (m-1)) ) & M[T[i]]);
19         if((R & 1) != 0){
20             printf("\t--> Casamento na posicao: %3d <--\n", i - m + 2);
21         }
22     }
23     free(M);
24 }

```

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](#).