Universidade Federal de Viçosa - *Campus* Florestal
Bacharelado em Ciência da Computação
CCF 330 - Projeto e Análise de Algoritmos
Prof. Daniel Mendes Barbosa

Trabalho Prático 04

Processamento de Caracteres

Samuel Jhonata S. Tavares 2282
Wandella Maia de Oliveira 2292

Florestal - MG
2018

Sumário

1 Introdução

Este trabalho tem por objetivo implementar um processamento de caracteres, fazendo o casamento exato de padrões, de acordo com dois algoritmos vistos em sala: *Força Bruta* e *Boyer-Moore-Horspool (BMH)*, onde ambos são comparados através de diversas entradas de tamanhos diferentes.

2 Desenvolvimento

2.1 Considerações Gerais

Neste trabalho, foram utilizadas como ferramentas a *IDE Netbeans* para a implementação, o Sistema Operacional *Linux (Distribuições Ubuntu e Mint)*, além do *MS Excel* para geração dos gráficos.

Ambos os algoritmos foram retirados do material de aula, uma vez que o objetivo maior deste trabalho é a comparação entre eles (ZIVIANI et al., 2010).

2.2 Parte A - Implementação do Casamento Exato de Caracteres

2.2.1 Implementação

A implementação foi feita de forma modular, com os arquivos: ***casamentoCaracteres.c***, ***casamentoCaracteres.h*** e ***main.c***.

2.2.1.1 *CasamentoCaracteres*

O cabeçalho foi implementado no arquivo ***casamentoCaracteres.h***, que é mostrado na Figura 1, o menu, no arquivo ***main.c*** e toda a lógica foi implementada no arquivo ***casamentoCaracteres.c***.

Figura 1 – Arquivo ***casamentoCaracteres.h***

```
1  #ifndef CASAMENTOCARACTERES_H
2  #define CASAMENTOCARACTERES_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  #define MAXTAMTEXTO 1000000
8  #define MAXTAMPADRAO 100
9  #define MAXCHAR 256
10
11 typedef char TipoTexto[MAXTAMTEXTO] ;
12 typedef char TipoPadrao[MAXTAMPADRAO] ;
13
14 int carregaArquivo(char *nomeArq, char* texto);
15 void ForcaBruta(TipoTexto T, long n, TipoPadrao P, long m, int mostrarPosicao);
16 void BMH(TipoTexto T, long n, TipoPadrao P, long m, int mostrarPosicao);
17
18 #endif /* CASAMENTOCARACTERES_H */
```

Foram redefinidos dois vetores de caracteres para guardar o texto e o padrão a ser pesquisado, respectivamente *TipoTexto*, com tamanho de *MAXTAMTEXTO*, definido em 1000000, e *TipoPadrao*, com tamanho de *MAXTAMPADRAO*, definido em 100. Também foi definido *MAXCHAR* em 256, que é usado no algoritmo *BMH*.

Foram implementadas as funções *carregaArquivo()*, *ForcaBruta()* e *BMH()*.

A função *carregaArquivo()*, carrega o arquivo com o texto a ser analisado posteriormente (C PROGRESSIVO.NET, 2015), salvando-o em um vetor de caracteres do tamanho máximo permitido (*TAMMAXTEXTO*), retornando 1 caso tenha sido carregado corretamente, e 0 caso contrário.

A função *ForcaBruta()*, foi retirada dos slides de aula (ZIVIANI et al., 2010; ZIVIANI et al., 2004), e funciona procurando casamentos a partir de cada caractere do texto, um a um, do primeiro ao último possível (até a última posição onde o padrão caiba), com testes exaustivos.

Já a função *BMH()*, também retirada dos slides de aula , é uma simplificação do *Boyer-Moore (BM)* feita por *Horspool*, executando de forma mais rápida (ZIVIANI et al., 2010; ZIVIANI et al., 2004). Primeiramente, é feito um pré-processamento, com um vetor criado do tamanho de *MAXCHAR+1*, onde, nas primeiras posições (até o tamanho do padrão), é salvo seu deslocamento, e nas outras posições, o tamanho do padrão. Depois, a pesquisa é feita tendo por base essa tabela, que irá definir o deslocamento das posições a serem comparadas.

2.2.2 Main

No arquivo ***main.c***, foi implementado o menu principal, com as opções de "Carregar Arquivo", "Mostrar Texto", "Pesquisar Padrão"e "Sair".

Ao selecionar "Carregar Arquivo", é solicitada a entrada de um arquivo, que será carregado e salvo o seu texto.

Ao selecionar "Mostrar Texto", o texto que foi carregado do arquivo é mostrado na tela, ou, caso não tenha sido carregado nenhum, é mostrado uma mensagem de erro.

Ao selecionar "Pesquisar Padrão", o usuário deve entrar o padrão a ser pesquisado e escolher se as posições que forem encontradas devem ou não aparecer. Então, os dois algoritmos são executados, são mostradas na tela as posições em que o padrão foi encontrado, caso aplicável, e o tempo de execução de cada algoritmo, caso esteja no modo *Debug*. É considerado apenas o tempo a partir em que o algoritmo é invocado, até o momento em que ele retorna.

Ao selecionar "Modo de Apresentação", o usuário pode escolher entre o modo *Normal*, onde não é mostrado o tempo de execução dos algoritmos, e o modo *Debug*, onde é mostrado.

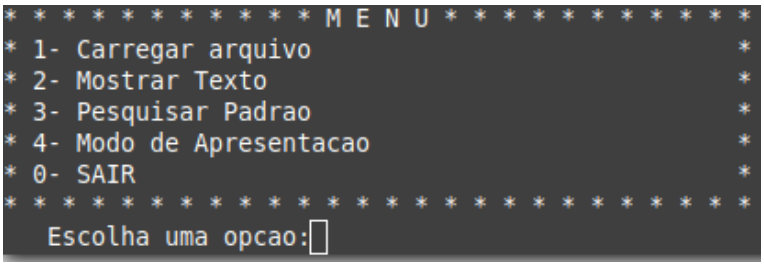
O usuário pode carregar um arquivo por vez, pesquisar por diversos padrões, carregar outros textos, e então, sair do programa.

2.2.3 Execução

A seguir, é mostrada uma breve execução do programa, com os pontos principais:

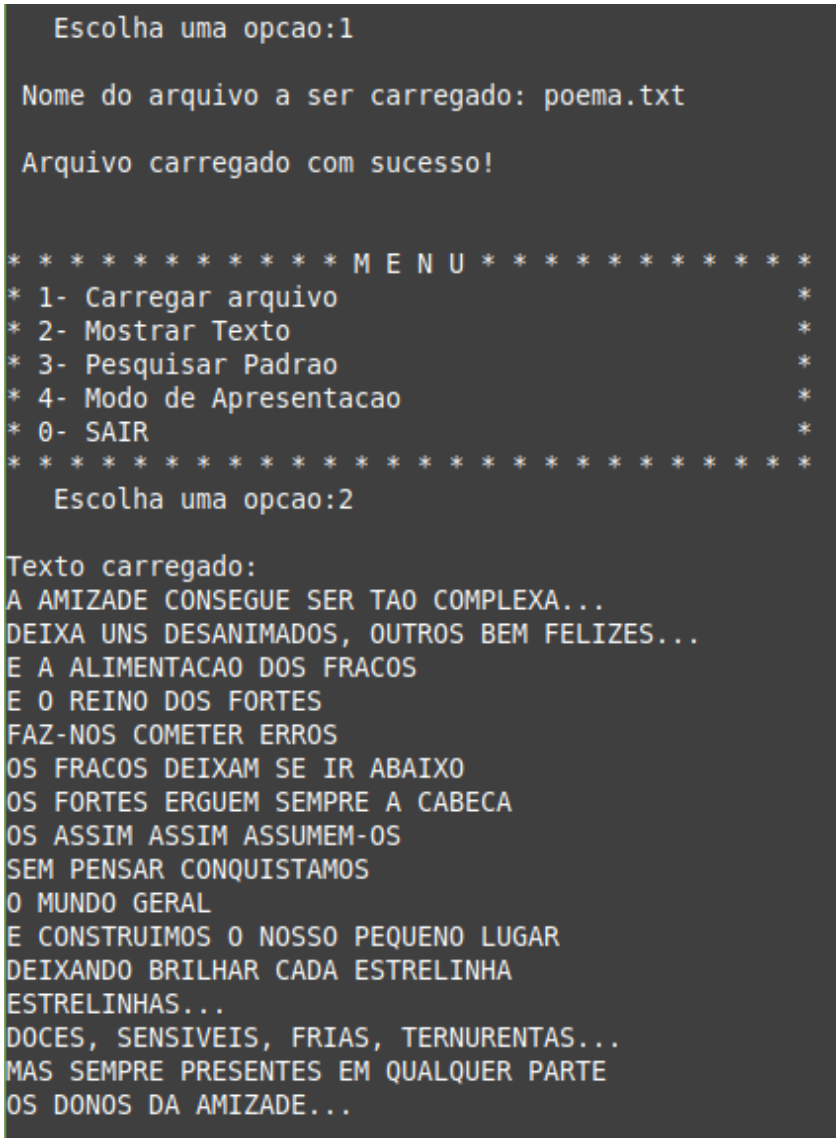
Na Figura 2, é mostrado o menu do programa.

Figura 2 – Menu



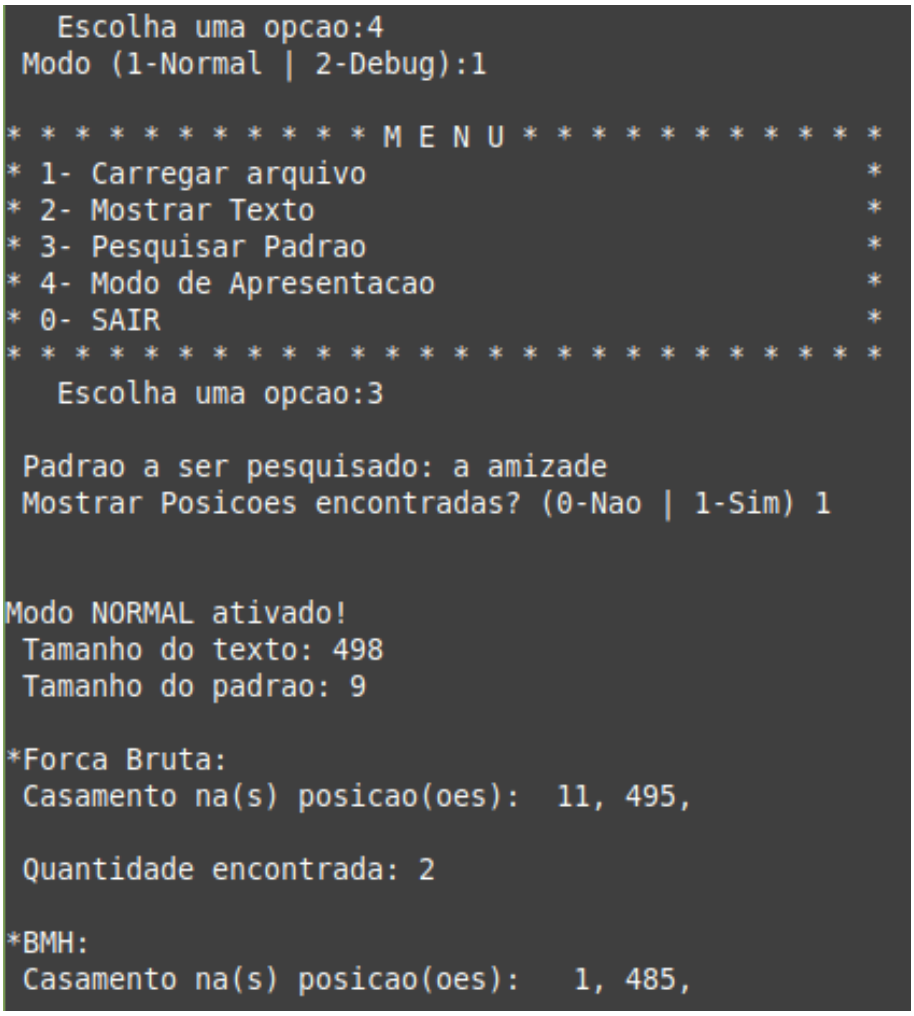
Na Figura 3, é mostrado o carregamento e exibição do texto.

Figura 3 – Carregar arquivo e mostrar texto



Na Figura 4, é mostrado uma pesquisa no modo *Normal*.

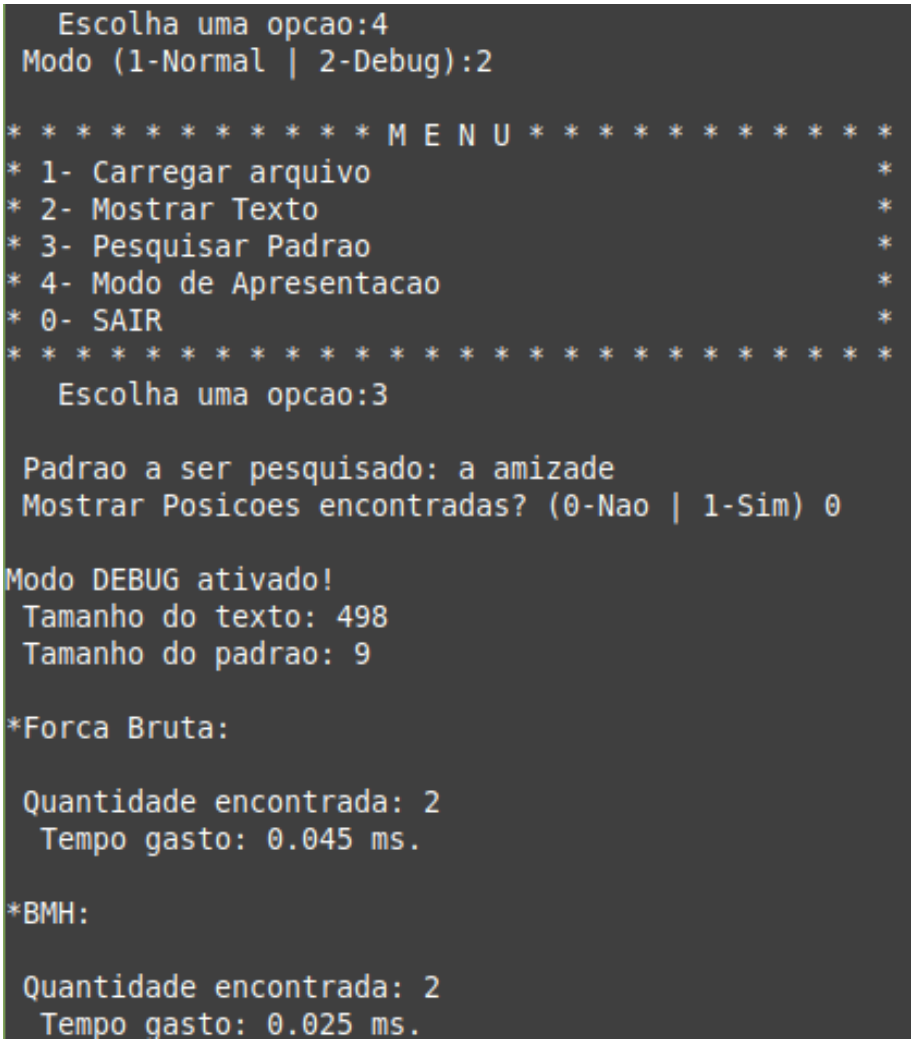
Figura 4 – Menu



É possível notar que o algoritmo *Força Bruta* mostra a última posição do padrão encontrado, uma vez que inicia a busca do início e termina no final. Já o *BMH*, como inicia do final e termina no início do padrão, mostra a primeira posição.

Na Figura 5, é mostrado uma pesquisa no modo *Debug*.

Figura 5 – Menu



2.3 Parte B - Análise

Para analisar e comparar os algoritmos, foram feitas diversas execuções com três textos diferentes, buscando por tamanhos pequeno, médio e grande, onde, para cada um, foi pesquisado padrões que continham todo, ou uma parte, no texto e que não estavam presentes, com tamanhos de 1, 10, 50 e 100 caracteres.

Foram criados 3 arquivos de textos: ***poema.txt***, com um poema, com 498 caracteres, ***nomes.txt***, com milhares de nomes, muitos repetidos, com 7.145 caracteres, e ***mesclado.txt***, contendo partes do poema mesclado com nomes do outro texto, com 654.720 caracteres.

Na Figura 6, são mostrados os padrões que continham todo ou parte presente nos textos.

Figura 6 – Padrões semelhantes ao texto utilizados para comparação

Texto	Tamanho Padrão	Padrão
poema.txt	1	a
	10	consegue s
	50	A amizade consegue ser tao complexa... Deixa uns
	100	A amizade consegue ser tao complexa... Deixa uns desanimados, outros bem felizes... E a alimentaca
mesclado	1	a
	10	consegue s
	50	A amizade consegue ser tao complexa... Deixa uns
	100	A amizade consegue ser tao complexa... Deixa uns desanimados, outros bem felizes... E a alimentaca
nomes	1	s
	10	Samuel Wan
	50	Samuel Wandella Adriana Pedro Daniel Arthur Artur
	100	Samuel Wandella Adriana Pedro Daniel Arthur Artur Alvaro Carlos Jose Maria Clara Augusta Andre Hamil

Já na Figura 7, são mostrados os padrões que não estavam presente nos textos.

Figura 7 – Padrões diferentes ao texto utilizados para comparação

Texto	Tamanho Padrão	Padrão
poema.txt	1	y
	10	leodkcnsdr
	50	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr
	100	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr
mesclado	1	y
	10	leodkcnsdr
	50	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr
	100	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr
nomes	1	y
	10	leodkcnsdr
	50	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr
	100	leodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdrleodkcnsdr

Assim, buscou-se por verificar dois casos importantes: quando os padrões casam, fazendo com que o algoritmo execute mais comparações, e quando os padrões não casa, cessando as comparações já no início do padrão.

A seguir, serão apresentados os gráficos gerados:

Nos gráficos das Figuras 8, 9 e 10, são mostradas as comparações dos dois algoritmos, em cada um dos textos, para padrões semelhantes e diferentes ao conteúdo de cada texto, para os tamanhos de padrão 1, 10, 50 e 100 caracteres. Cada gráfico, é dado em tempo de execução por tamanho do padrão.

Figura 8 – Gráfico do tempo de execução de cada algoritmo, com padrões semelhantes e diferentes ao texto '***poema.txt***'

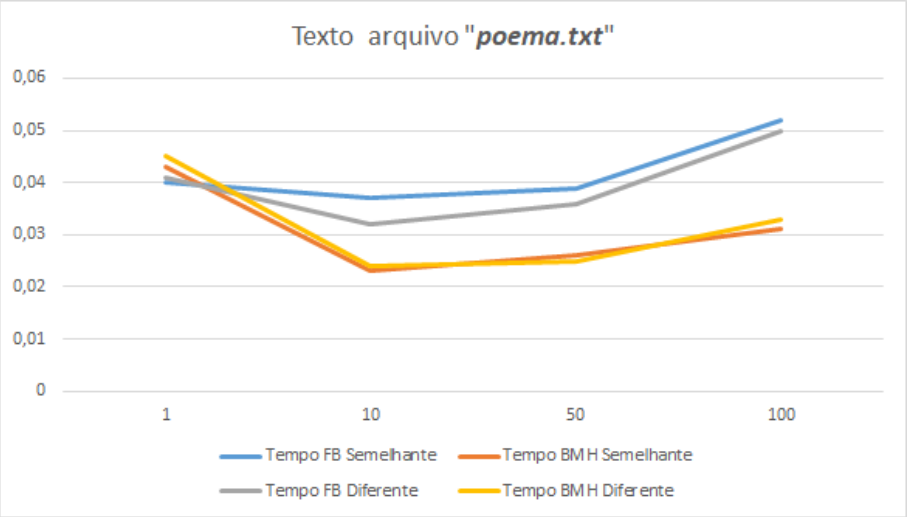


Figura 9 – Gráfico do tempo de execução de cada algoritmo, com padrões semelhantes e diferentes ao texto '***mesclado.txt***'

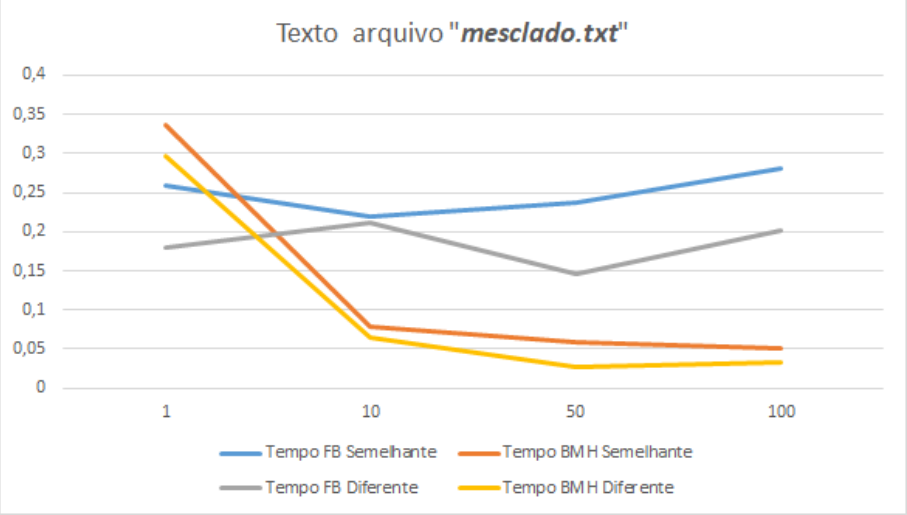
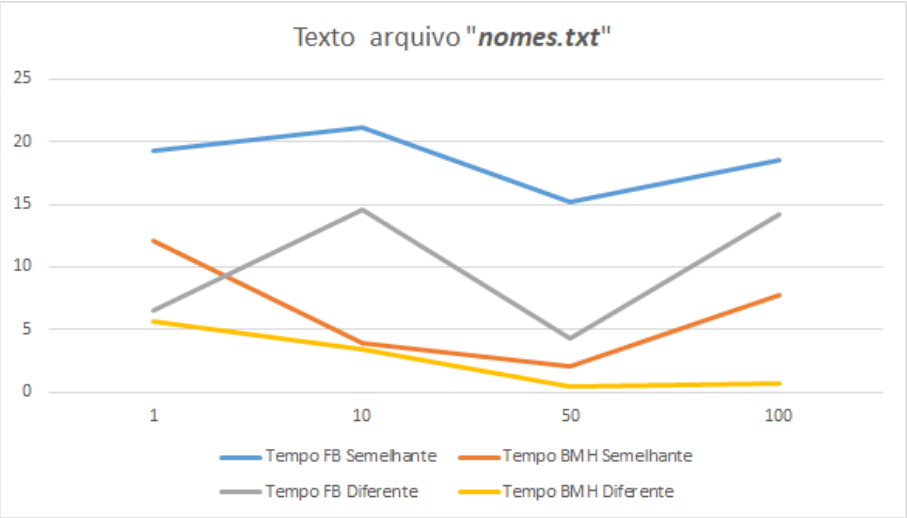


Figura 10 – Gráfico do tempo de execução de cada algoritmo, com padrões semelhantes e diferentes ao texto '***nomes.txt***'



De forma geral, o algoritmo *BMH* é mais eficiente que o *Força Bruta* em todos os casos onde o padrão é maior que 1 caractere. Isso se dá pelo fato de que o seu deslocamento é maior, porém, como há um pré-processamento antes da pesquisa, quando há apenas 1 caractere a ser pesquisado, o *Força Bruta* se mostra um pouco melhor.

Já nos gráficos das Figuras 11 e 12, é feito uma comparação do tempo de execução de cada um dos algoritmos, em cada texto, com padrões semelhantes e diferentes ao texto. Cada gráfico é dado em tempo de execução por tamanho de padrão.

Figura 11 – Gráfico do tempo de execução do algoritmo *Força Bruta*, para padrões semelhantes e diferentes em cada texto

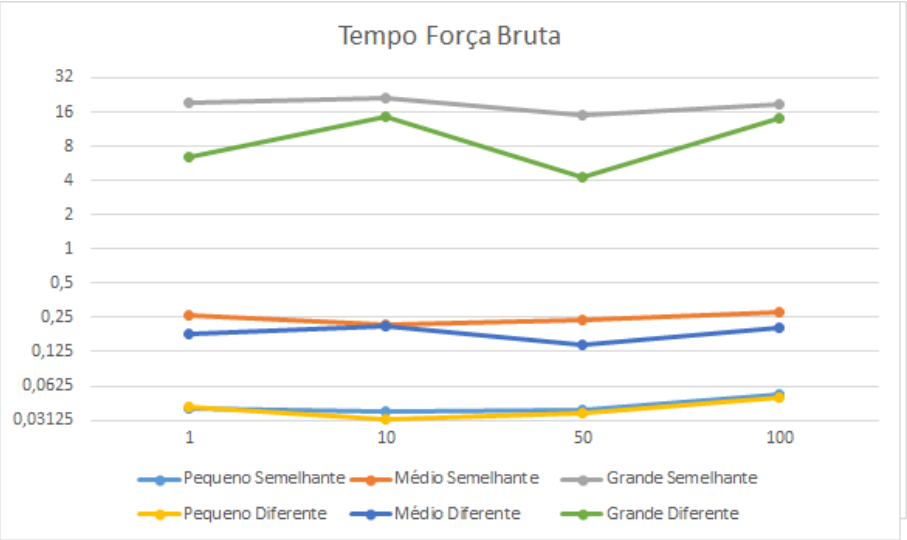
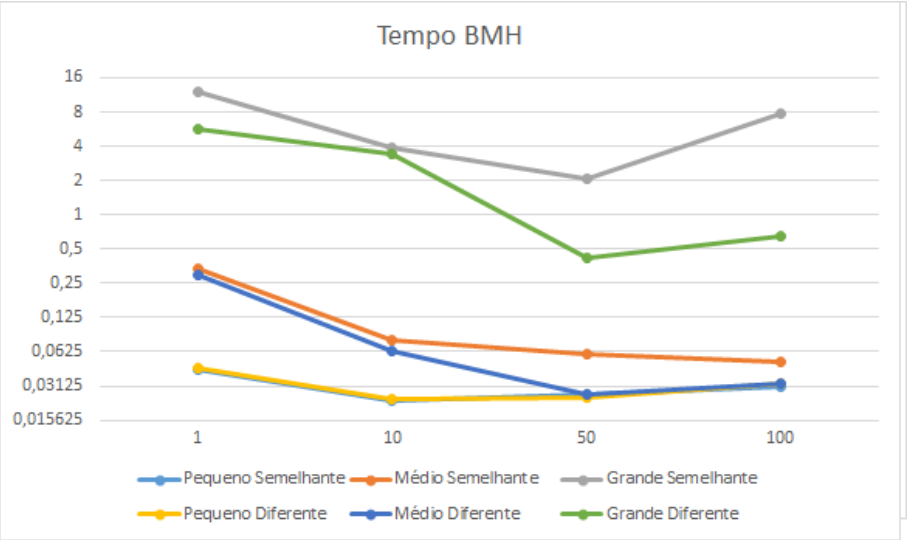


Figura 12 – Gráfico do tempo de execução do algoritmo *BMH*, para padrões semelhantes e diferentes em cada texto



Assim, é possível perceber que, o tamanho do texto influencia, e muito, no tempo de execução dos algoritmos, tendo um impacto ainda maior no algoritmo *Força Bruta*.

3 Conclusão

Com este trabalho, foi possível fazer uma análise entre dois algoritmos de casamento exato de padrões ao processar caracteres, permitindo concluir que o algoritmo *BMH* é mais eficiente que o *Força Bruta*, como era esperado.

Referências

C PROGRESSIVO.NET. *Lendo arquivos em C: As funções fgetc, fscanf e fgets*. 2015. Disponível em: <https://www.cprogressivo.net/2013/11/Como-ler-arquivos-em-C-As-funcoes-fgetc-fscanf-fgets.html>. Acesso em: 18 nov. 2018.

ZIVIANI, N. et al. *Projeto de algoritmos: com implementações em Pascal e C*. [S.l.]: Thomson, 2004. v. 2.

ZIVIANI, N. et al. *Processamento de Cadeias de Caracteres*. 2010. Disponível em: <http://www2.dcc.ufmg.br/livros/algoritmos/cap8/slides/c/completo1/cap8.pdf>. Acesso em: 18 nov. 2018.