Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Ciência da Computação

# PROJETO E ANÁLISE DE ALGORITMOS

## Terceiro Trabalho. disponível em:

`http://www.dcc.ufmg.br/~horacio/PaaTP3`

Horácio Antonio Braga Fernandes de Oliveira

Professor - Nivio Ziviani

Belo Horizonte
31 de maio de 2004

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Códigos

# 1 Introdução

O problema do *Casamento de Padrão* consiste em encontrar todas as ocorrências de um dado padrão $P = p_1p_2 \cdots p_m$ em um texto $T = t_1t_2 \cdots t_n$, onde tanto $T$ e $P$ são seqüências de caracteres de um conjunto finito de caracteres $\Sigma$ [6]. Tais seqüências ou cadeias de caracteres podem aparecer no processamento de textos em linguagem natural, códigos, dicionários, seqüenciamento de DNA em biologia computacional, representação de imagens por meio de bitmaps, dentre outros [8].

Esta busca por ocorrências pode ser feita de forma *Exata* (ver Seção 2) ou *Aproximada* (ver Seção 3). Um valor $k$ define esta aproximação, sendo que para $k = 0$ tem-se um casamento exato, e para $0 < k < m$ tem-se um casamento aproximado.

Diversos algoritmos existem para resolver estes dois tipos de problema. Os mais antigos e famosos são o *Knuth-Morris-Pratt* e o *Boyer -Moore*, que datam da época de 1977. A partir desta data, vários estudos têm sido realizados para encontrar algoritmos mais simples, melhores, que procurassem por padrões mais complexos, que tivessem complexidade linear de espaço e etc.

## 1.1 Histórico do Problema

Um algoritmo básico de pesquisa utilizando *força-bruta* é mostrado no Código 1. A complexidade deste algoritmo é, no pior caso, proporcional a $N * M$, apesar de na prática sua complexidade média é proporcional a $M + N$.

**Código 1** Algoritmo básico de pesquisa utilizando *força-bruta*.

```
Function bruteSearch: integer;
  var i, j: integer;
  begin
  i:=1; j:=1;
  repeat
  if (T[i] = P[j])  then begin i:=i+1; j:=j+1   end
           else begin i:=i-j+2; j:=1   end
   until (j>M)  or (i>N);
  if (j>M)  then bruteSearch:=i-<  else bruteSearch:=i
End
```

Em 1970, Cook provou um resultado teórico sobre um tipo particular de máquina abstrata mostrando que existia um algoritmo que resolveria o problema de casamento de padrão em um tempo proporcional a $M + N$ no pior caso. D. E. Knuth e V. R. Pratt começaram a estudar o modelo teórico de Cook para prover uma implementação prática desta teoria resultando em um algoritmo melhor de busca. Porém, ao mesmo tempo, foi descoberto que J. H. Morris descobriu o mesmo algoritmo ao tentar resolver um outro problema.

Knuth, Morrins e Pratt não publicaram seus algoritmos até 1976, e neste meio tempo, R. S. Boyer e J. S. Moore descobriram um algoritmo bem mais rápido.

## 1.2 Objetivos do Trabalho

O objetivo deste trabalho é realizar experimentos em um conjunto de programas para recuperar ocorrências de padrões em arquivos constituídos de documentos, utilizando algoritmos lineares de busca sequencial.

Os algoritmos estudados são:

- *Algoritmo de Boyer-Moore-Horspool para Casamento Exato de Padrões*
- *Alg. de Boyer-Moore-Horspool-Sunday para Casamento Exato de Padrões*
- *Algoritmo Shift-And para Casamento Exato de Padrões*
- *Algoritmo Shift-And para Casamento Aproximado de Padrões*

Além disso, será apresentado, analisado e comparado a implementação de um algoritmo para casamento aproximado usando programação dinâmica:

- *Algoritmo de Programação Dinâmica*[1] *para casamento Aproximado de Padrões*

As implementações, em especial as de casamento aproximado, serão comparadas com o programa *AGrep* (ver Seção 3.3), uma implementação do casamento aproximado de padrões feita por Sun Wu e Udi Manber, do Departamento de Ciência da Computação da Universidade do Arizona.

[1] Este algoritmo será referenciado simplesmente como "Programação Dinâmica", mas fica subentendido que estamos nos referindo ao algoritmo que resolve o problema do casamento aproximado de padrões utilizando um método de Programação Dinâmica.

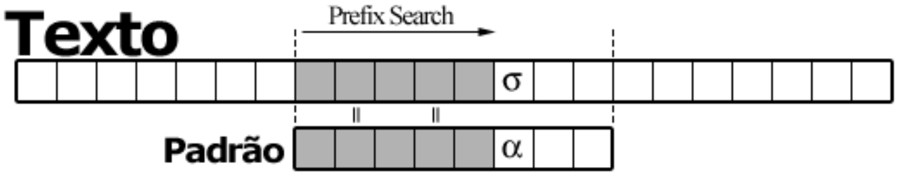



Figura 1: Método de busca utilizando o *Prefix Searching*.

# 2 Casamento Exato de Padrões

Nesta parte estamos interessados no casamento exato de padrões, ou seja, na pesquisa de padrões $P$ exatos no texto $T$ sem considerar erros ($k = 0$). Tais algoritmos são:

- *Algoritmo de Boyer-Moore-Horspool (BMH):* Seção 2.2
- *Algoritmo de Boyer-Moore-Horspool-Sunday (BMHS):* Seção 2.3
- *Algoritmo Shift-And:* Seção 2.4

## 2.1 Metodologias de Procura

Geralmente, três metodologias de procura são utilizados para o casamento de padrões em cadeias de caracteres [6]:

- *Prefix Searching*
- *Suffix Searching*
- *Factor Searching*

**Prefix Searching**: A procura é feita seguindo em frente na janela de pesquisa, lendo todos os caracteres do texto um após o outro. Para cada posição da janela, procuramos palo maior prefixo[2] da janela que é também um prefixo do padrão. Esta forma de procura é ilusrada na Figura 1. Exemplos de algoritmos que utilizam esta metodologia é o KNUTH-MORRIS-PRATT e o SHIFT-AND.

**Suffix Searching**: A procura é feita seguindo de trás pra frente na janela de pesquisa, lendo o mais longo sufixo da janela que é também um sufixo do padrão, como mostra a Figura 2. Este método nos permite na média evitar ler alguns caracteres do texto, e por consequência, resulta em um algoritmo sublinear no caso médio. O mais famoso algoritmo que utiliza esta metodologia é o BOYER-MOORE e suas variantes HORSPOOL e SUNDAY.

**Factor Searching**: A procura é feita seguindo de trás pra frente na janela de pesquisa, procurando pelo mais longo sufixo que é também um fator do padrão. Ver Figura 3. Como no Suffix Searching, este método resulta em um tempo esperado sublinear do algoritmo, e até mesmo a algoritmos ótimos. O principal ponto é que este método requer um modo de reconhecer um conjunto de fatores do padrão, e isso é bastante complexo.

[2]Dado strings $x$, $y$ e $z$, dizemos que $x$ é prefixo de $xy$, um sufixo de $yx$ e um fator de $yxz$.

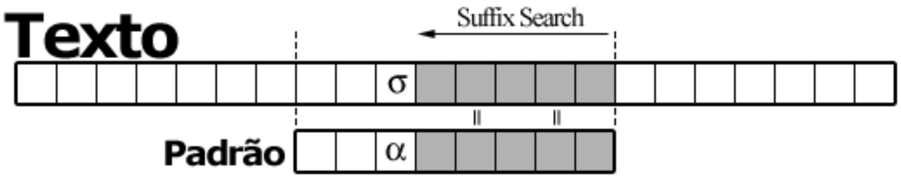



Figura 2: Método de busca utilizando o *Suffix Searching*.

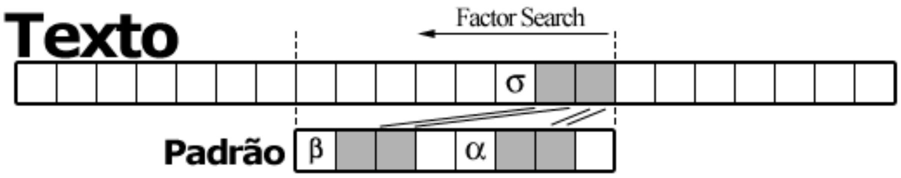



Figura 3: Método de busca utilizando o *Factor Searching*.

## 2.2 Algoritmo de Boyer-Moore-Horspool (BMH)

Conforme mencionado na Seção 1.1, no ano de 1977, R. S. Boyer e J. S. Moore descobriram um algoritmo mais rápido do que o Knuth-Morrins-Pratt para o problema do casamento de padrão. Este algoritmo ficou conhecido como Boyer-Moore (BM). Em 1980, Horspool apresentou uma simplificação importante no algoritmo BM ficando este algoritmo conhecido como Boyer-Moore-Horspool (BMH). Esta algoritmo executa uma busca sequencial no texto da coleção a ser pesquisada através de comparações realizadas da direita para a esquerda com relação ao padrão, o que torna este muito rápido.

O enfoque do algoritmo BMH consiste em pesquisar o padrão $P$ em uma janela que desliza ao longo do texto $T$. Para cada posição desta janela, o algoritmo faz uma pesquisa por um sufixo da janela que casa com um sufixo de $P$ por meio de comparações realizadas no sentido da direita para a esquerda. Se não ocorrer uma desigualdade, então uma ocorrência de $P$ em $T$ foi encontrada. Senão, o algoritmo calcula um deslocamento em que o padrão deve ser deslizado para a direita antes que uma nova tentativa de casamento se inicia [8].

O algoritmo Boyer-Moore-Horspool funciona da seguinte forma (ver Figura 4)[6]: para cada posição da janela de procura, comparamos o seu último caractere ($\beta$ na figura) com o último caractere do padrão. Se eles casarem, verificamos a janela de procura voltando para o penúltimo caractere do padrão assim até que seja encontrado o padrão ou falhe no caractere do teste ($\alpha$ na figura). Então, independente se teve um casamento ou não, fazemos um deslocamento na janela de acordo com a próxima ocorrência da letra $\beta$ no padrão.

Basicamente o que diferencia o algoritmo BMH do BM é que Hospool assumiu que, para um alfabeto relativamente grande, a função de deslocamento utilizada sempre possuirá um deslocamento maior. Hospool apenas considerou uma pequena modificação nesta função que é fácil de computar e que resulta em deslocamentos maiores. Resultando em um algoritmo mais rápido.

O **algoritmo** para o Boyer-Moore-Horspool é mostrado no Código 2. O

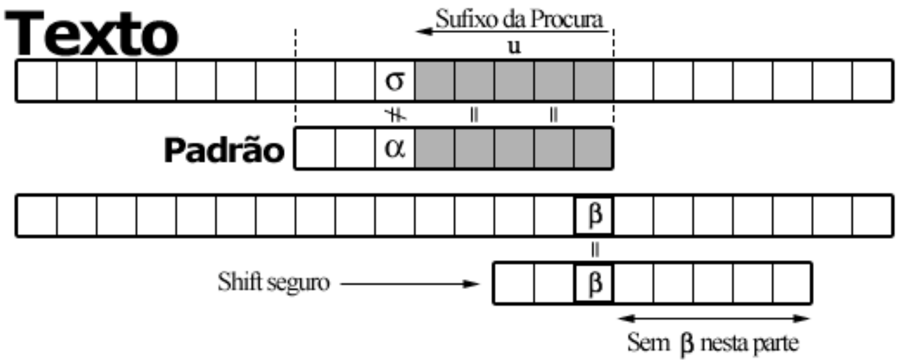



Figura 4: Algoritmo Boyer-Moore-Horspool. O padrão é deslocado de acordo com o último caractere da janela de procura.

pré-processamento do padrão para obter a tabela de deslocamento $d$ ocorre nas linhas 2 e 3. A fase de pesquisa é constituída de um anel em que $pos$ varia de 0 até $n - m$ com incrementos de $d[t_{pos+m}]$, o que equivale ao endereço da tabela $d$ do caractere que está na $i$ -ésima posição no texto, o qual correponde à posição do último caractere de P.

**Código 2** Algoritmo Boyer-Moore-Horspool

```
1   Function Boyer-Moore-Horspool (P = p1p2 ... pm, T = t1t2 ... tn);
2     // Pré-processamento
3         For c ∈ Σ  Do d[c] ← m
4         For j ∈ 1...m-1  Do d[pj] ← m - j
5     // Procura
6         pos ← 0
7         While pos ≤ n - m  Do
8             j ← m
9             While j > 0 and tpos+j = pj  Do j ← j - 1
10            if j = 0  Then Mostra ocorrência em pos + 1
11            pos ← pos + d[tpos+m]
12        End of While
13  End
```

Uma **análise de complexidade** deste algoritmo mostra que na fase de pré - processamento, quando os deslocamentos são computados utilizando a heurística *ocorrência* e com base apenas no padrão e no alfabeto, a complexidade de tempo e de espaço é de $O(c)$. Para a fase de pesquisa, o pior caso do algoritmo é $O(mn)$, o melhor caso é $O(n/m)$ e o caso esperado ou médio é $O(n/m)$, se $c$ não é pequeno e $m$ não é muito grande [8].

## 2.3 Algoritmo de Boyer-Moore-Horspool-Sunday (BMHS)

"Ao invés de realizar deslocamentos na janela usando o seu último caractere, nós podemos user o próximo caractere depois da janela, o que resulta em maiores deslocamentos, em geral."

Esta simples modificação do algoritmo de Boyer-Moore-Horspool foi proposta por Sunday, em 1990, e passou a ser conhecida como Algoritmo de Boyer-Moore-Horspool-Sunday (BMHS). Através de deslocamentos maiores, e o baixo número de referências à memória deste novo algoritmo, o faz executar mais rapidamente no geral.

Desta forma, para pré-computar o padrão, o valor inicial de todas as entradas na tabela de deslocamentos é feito igual a $m+1$. A seguir, os $m$ primeiros caracteres do padrão são usados para obter os outros valores da tabela.

O **algoritmo** para o Boyer-Moore-Horspool-Sunday é mostrado no Código 3. E como no BMH, o pré-processamento do padrão para obter a tabela de deslocamento $d$ ocorre nas linhas 2 e 3. A fase de pesquisa é constituída de um anel em que $pos$ varia de 0 até $n-m$ com incrementos de $d[t_{pos+m+1}]$, o que equivale ao endereço da tabela $d$ do caractere que está na $i+1$-ésima posição no texto, o qual correponde à posição do último caractere de P.

**Código 3** Algoritmo Boyer-Moore-Horspool-Sunday

```
1   Function Boyer-Moore-Horspool-Sunday (P = p1p2 ... pm, T = t1t2 ... tn);
2     // Pré-processamento
3         For c ∈ Σ  Do d[c] ← m + 1
4         For j ∈ 1...m-1  Do d[pj] ← m + 1 - j
5     // Procura
6         pos ← 0
7         While pos ≤ n - m  Do
8            j ← m
9            While j > 0 and t_{pos+j} = pj  Do j ← j - 1
10           if j = 0  Then Mostra ocorrência em pos + 1
11           pos ← pos + d[t_{pos+m+1}]
12        End of While
13  End
```

A **análise de complexidade** deste algoritmo é a mesma do algoritmo BMH. Entretanto os deslocamentos são mais longos, levando a saltos relativamente maiores para padrões curtos e resultando em execuções mais rápidas.

## 2.4 Algoritmo Shift-And (SA)

Este algoritmo foi proposto por Baeza-Yates e Gonet em 1989. A idéia por trás deste algoritmo é bastante simples, e seu tempo de execução é bem melhor que diversos algoritmos de busca como por exemplo o proposto por Knuth-Morris-Pratt.

Este consiste em manter um conjunto de todos os prefixos de $P$ que casem com um sufixo do texto lido. Os algoritmos utilizam paralelismo de bits[3] para atualizar este conjunto a cada novo caractere. Este conjunto é representado por uma máscara $D = d_m \ldots d_1$.

Colocamos um 1 na $j$-ésima posição de $d$ (esta posição é então dita estar ativa) se e somente se $p_1 \ldots p_j$ é um sufixo de $t_1 \ldots t_i$. Se o tamanho de $p$ é menor que $w$, então este vetor poderá ser armazenado em um registrador. Reportamos um casamento sempre que $d_m$ ficar ativo.

Ao ler o próximo caractere $t_{i+1}$, temos que computar o novo conjunto $D'$. Uma posição $j+1$ neste conjunto será ativada se e somente se a posição $j$ estava ativada em

[3]Uma técnica que tira proveito do paralelismo intrínseco das operações sobre bits dentro de uma palavra de computador. Neste caso, é possível empacotar muitos valores em uma única palavra e atualizar todos eles em uma única operação.

$D$, isto é, $p_1 \dots p_j$ era um sufixo de $t_1 \dots t_i$ e $t_{i+1}$ casa com $p_{j+1}$. Este novo conjunto é fácil de computar em tempo constante usando opreações com paralelismo de bits.

O algoritmo primeiro monta uma tabela B, que armazena a máscara de bits $b_m \dots b_1$ para cada caractere. A máscara em B[c] tem o $j$-ésimo bit ligado se $p_j = c$.

Inicialmente setamos $D = 0^m$, e para cada novo caractere $t_{i+1}$ atualizamos $D$ usando a fórmula

$$\boxed{\text{D'} \leftarrow ((\text{D} << 1 \mid 0^{m-1}1) \ \& \ \text{B}[t_{i+1}]} \qquad (1)$$

Intuitivamente, o "<<" desloca a posição para a esquerda para marcar no passo $i + 1$ quais posições de $p$ eram sufixos no passo $i$. Também marcamos uma string vazia $\epsilon$ como um sufixo, para que possamos fazer um OR da nova máscara com $0^{m-1}1$. Agora podemos manter até esta posição apenas aquelas que casam $t_{i+1}$ com $p_{j+1}$, fazendo um AND deste conjunto de posições com o conjunto B[$t_{i+1}$] de posições de $t_{i+1}$ em $p$.

Apesar de parecer um pouco complicado, este **algorimo** pode ser implementado em poucas linhas, como mostra o Código 4.

**Código 4** Algoritmo Shift-And

```
Function Shift-And (P = p1p2 ... pm , T = t1t2 ... tn);
  // Pré-processamento
     For c ∈ Σ  Do B[c] ← 0^m
     For j ∈ 1...m  Do B[pj] ← B[pj] | 0^(m-j)10^(j-1)
  // Procura
     D ← 0^m
     For pos ∈ 1...n  Do
        D ← ((D << 1) | 0^(m-1)1) & B[t_pos]
        if D & 10^(m-1) ≠ 0^m  Then Mostra ocorrência em pos - m + 1
     End of For
End
```

O **custo deste algoritmo** é $O(n)$, assumindo que as operações na Fórmula (1) podem ser realizadas em tempo constante. O custo de espaço é $O(c)$.

# 3 Casamento Aproximado de Padrões

Nesta parte estamos interessados no casamento aproximado de padrões, ou seja, na pesquisa de padrões $P$ no texto $T$ considerando um certo número $k$ de erros $(0 < k < m)$.

Tais algoritmos são:

- *Algoritmo Shift-And Aproximado (SAA):* Seção 3.1
- *Algoritmo Utilizando Programação Dinâmica:* Seção 3.2
- *Implementação do AGrep:* Seção 3.3

Este problema é também conhecido como "casamento de strings permitindo erros" [6]. E consiste em encontrar um padrão $P$ em um texto $T$ quando um limitado número $k$ de diferenças é permitido entre o padrão e o texto encontrado.

Por "erros", entendemos o que é definido pelo modelo denominado *Levenshtein Distance* ou *Edit Distance*. Neste caso, uma diferença ou erro corresponde a uma operação de edição, ou seja, uma *inserção*, *deleção* ou *substituição* de um caractere.

## 3.1 Algoritmo Shift-And Aproximado (SAA)

Como o próprio nome indica, este algoritmo é uma modificação do Algoritmo Shift-And original que permite o casamento aproximado dos padrões. Este algoritmo foi proposto por Wu e Manber em 1992.

A implementação deste algoritmo é bastante parecida com a implementação do Shift-And original. Ele também representa o autômato como uma seqüência de bits e, através de operações lógicas bit a bit, realiza o caminhamento no autômato. Uma máscara também é definida para cada um dos caracteres do alfabeto.

Este algoritmo empacota cada linha $j(0 < j < k)$ do autômato não-determinista em uma palarvra $R_j$ diferente do computador. A cada novo caractere lido do texto, todas as transições do autômato são simuladas usando operações entre as $k+1$ máscaras de bits. Todas as $k+1$ máscaras de bits têm a mesma estrutura e assim o mesmo bit é alinhado com a mesma posição no texto[8].

Na posição $i$ do texto, os novos valores $R'_j$, $o < j < k$, são obtidos a partir dos valores correntes de $R_j$, a saber:

$$R'_0 = ((R_0 >> 1) \mid 10^{m-1}) \ \& \ \mathrm{M[T[i]]}$$
$$R'_j = ((R_j >> 1) \ \& \ \mathrm{M[T[i]]}) \mid R_{j-1} \mid (R_{j-1} >> 1) \mid (R'_{j-1} >> 1)$$

onde $M$ é a tabela do algoritmo Shift-And conforme mostrado anteriormente. A princípio estas operações em $R_j$ parecem complicadas, mas explicando elas por partes, ficam bem mais fácil de entender:

- *$((R_j >> 1)$ & M[T[i]]):* faz com que os estados ativos acumulados pelo registrados $R_i$ continuem caminhando em direção ao estado final do autômato em conformidade com a máscara do caractere atual do texto;
- *$R_{j-1}$:* simula a inserção de um caractere no padrão, ou que um caractere do texto foi ignorado;

- *($R_{j-1}$ >> 1):* simula a substituição de um caractere do padrão;
- *($R'_{j-1}$ >> 1):* simula a remoção do próximo caractere do padrão.

Quando o registrador $R_j$ alcança o seu estado final, indica que ocorreu um casamento aproximado de padrão.

Apesar de parecer um pouco complicado, este **algorimo** pode ser **implementado** em poucas linhas, como mostra o Código 5. O funcionamento do mesmo é similar ao mostrado no Código 4.

**Código 5** Algoritmo Shift-And Aproximado

```
Function Shift-And-Aproximado (P = p1p2 ... pm, T = t1t2 ... tn);
  // Pré-processamento
     For c ∈ Σ  Do M[c] ← 0^m
     For j ∈ 1...m  Do M[p_j] ← M[p_j] | 0^(j-1)10^(m-j)
  // Procura
     For j ∈ 0...k  Do R_j ← 1^j0^(m-j)
     For i ∈ 1...n  Do
        Rant  ← R0
        Rnovo ← ((Rant >> 1) | 10^(m-1)) & M[T[i]];
        R0 ← R0 Rnovo
        For j ∈ 1...k  Do
           Rnovo ← ((R_j >> 1 & M[T[i]]) | Rant | ((Rant|Rnovo) >> 1)
           Rant  ← R_j
           R_j ← Rnovo | 10^(m-1)
        if Rnovo & 0^(m-1)1 ≠ 0^m  Then Mostra ocorrência em pos - m + 1
     End of For
End
```

A **complexidade** deste algoritmo é de $O(k\lceil m/w\rceil n)$ no pior caso e no caso médio, o que equivale a $O(kn)$ para padrões típicos [8].

## 3.2 Algoritmo usando Programação Dinâmica

A solução utilizando Programação Dinâmica é considerada uma das mais antigas formas de resolver este problema. O algoritmo final é atribuído a Sellers, em 1980, apesar de algoritmos deste tipo terem sido desenvolvidos várias vezes desde 1960.

O algoritmo funciona da seguinte forma: uma matriz $C_{0\ldots m}$ é inicializada com valores $C_i = i$ e então o texto $T$ é processado caractere a caractere. A cada novo caractere $t_j$, esta matriz é atualizada para $C'_{0\ldots m}$ de acordo com a seguinte fórmula:

$$C'_i \leftarrow \begin{cases} C_{i-1} & \text{se } p_i = t_j, \\ 1 + min(C_{i-1}, Ci - 1', C_i) & \text{caso contrário} \end{cases} \tag{2}$$

e as posições do texto onde $C_m \leq k$ são reportadas como posições finais das ocorrências.

Uma pequena modificação neste algoritmo reduz o seu tempo de processamento. A idéia é que, já que um determinado padrão não ocorre normalmente em um texto, os valores de cada coluna lidos da do início para o fim rapidamente alcança $k+1$ (i.e., não-casamento), e que se uma célula tem valor maior que $k+1$, o resultado da procura não depende de seu valor exato. Uma célula é dita ativa se seu valor é pelo menos $k$. O algoritmo mantém o contador da última célula ativa e evita trabalhar nas células subsequentes.

O Código 6 motra o **algoritmo** explicado acima.

**Código 6** Algoritmo de Busca utilizando Programação Dinâmica

```
Function ProgramaçãoDinâmica (P = p1p2 ... pm, T = t1t2 ... tn);
  // Pré-processamento
     For i ∈ 0...m  Do Ci ← i
     lact ← k + 1 //  Última célula ativa
  // Procura
     For pos ∈ 1...n  Do
        pC ← 0
        nC ← 0
        For i ∈ 1...lact  Do
           If pi = tpos  Then nC ← pC
           Else
              If pC < nC  Then nC ← pC
              If Ci < nC  Then nC ← Ci
              nC ← nC + 1
           End of If
           pC ← Ci
           Ci ← nC
        End of For
        While Clact > k  Do lact ← lact - 1
        If lact = m  Then Mostra ocorrência em pos
        Else lact ← lact + 1
     End of For
End
```

A **complexidade do algoritmo** sem a modificação mencionada é $O(mn)$. Com a modificação, esta complexidade passa a ser de $O(kn)$. Em ambos os casos tem-se uma complexidade de espaço de $O(m)$.

## 3.3 Implementação AGrep

O AGrep é um utilitário desenvolvido por Sun Wu e Udi Manber, do Departamento de Ciência da Computação da Universidade do Arizona. Este utilitário procura em um arquivo por uma string ou expressão regular e tem a capacidade de casamento aproximado.

Seu desenvolvimento inicial ocorreu em fevereiro de 1991 e a versão utilizada neste trabalho corresponde à última versão encontrada, o AGrep 2.04 que foi publicada em março de 1992. Este programa foi compilado a partir do código fonte em um sistema Linux Kernel 2.4. O AGrep está disponível através de FTP anônimo para cs.arizona.edu com o nome de agrep/agrep-2.04.tar.Z.

Esta implementação do AGrep inclui os seguintes algoritmos (com excessão do casamento exato de padrões simples, o qual é utilizado uma simples variação do algoritimo Boyer-Moore):

- *Bitap:* O algoritmo mais geral dentro do AGrep. Suporta muitas extensões como expressões regulares aproximadas, casamento de padrão, custos não-uniformes, casamentos simultâneos de múltiplos padrões, mistura de casamento exato/aproximado, etc.

- *Mgrep:* Um algoritmo de tempo médio sublinear para casamento de um conjunto de padrões. Assume que o conjunto de padrões possui $k$ padrões e que o menor padrão é de tamanho $m$.
- *Amonkey:* Um Algoritmo estilo BOYER-MOORE para casamento aproximado de padrão.
- *Mmonkey:* Combinando o algoritmo Mgrep com uma técnica de partição, tem-se um algoritmo com a mesma complexidade de tempo que o Amonkey, porém para textos e padrões ASCII, este algoritmo é mais rápido do que o Amonkey.

# 4 Implementação dos Algoritmos

A implementação dos algoritmos explicados até o momento para a solução do *Problema de Busca Exato e Aproximado* foi realizada na *linguagem C*, utilizando o compilador *GCC*. Todo o código-fonte encontra-se listado no Anexo B e disponível também em meio digital na página deste trabalho (ver Seção 6).

Para facilitar tanto a implementação como também o entendimento do código, este foi separado em diversos arquivos, cada qual com seus conteúdos relacionados:

- *Module.h:* Contém as definições dos Tipos de Dados e as declarações de todas as Funções da Implementação.
- *Main.c:* Código inicial da Implementação. Executa as funções dependendo do método escolhido para resolver o *Problema da Busca*, inicializa algumas variáveis globais e se encarrega de calcular o tempo de execução dos métodos.
- *Geral.c:* Funções auxiliares que são utilizadas pelos diversos métodos de solução do problema, como por exemplo funções para imprimir ocorrências, para subtrair tempos, uma função que imprime a ajuda do programa e etc.
- *BoyerMooreHorspool.c:* Contém a implementação da solução do *Problema de Busca Exata* utilizando o algoritmo Boyer-Moore-Horspool.
- *BoyerMooreHorspoolSunday.c:* Contém a implementação da solução do *Problema de Busca Exata* utilizando o algoritmo Boyer-Moore-Horspool-Sunday.
- *ShiftAnd.c:* Contém a implementação da solução do *Problema de Busca Exata* utilizando o algoritmo Shift-And.
- *ShiftAndAprox.c:* Contém a implementação da solução do *Problema de Busca Aproximada* utilizando o algoritmo Shif-And Aproximado.
- *ProgDin.c:* Contém a implementação da solução do *Problema de Busca Aproximada* utilizando um algoritmo de Programação Dinâmica.
- *makefile:* Diretrizes para o comando make para compilar, limpar e executar a implementação. Compatível com o make do Unix.

A utilização da implementação é parecida com a interface do AGrep e tem o seguinte formato:

```
Utilizacao: ./HGrep <-a Algoritmo> <-f ArquivoEntrada> <-p Padrao> [-k Aproximacao]

            Algoritmos: [BMH]  Boyer-Moore-Horspool
                        [BMHS] Boyer-Moore-Horspool-Sunday
                        [SA]   Shift-And
                        [SAA]  Shift-And-Aproximado
                        [PD]   Programação Dinâmica
```

## 4.1 Tipos de Dados Utilizados

Basicamente as estruturas de dados utilizadas para a implementação foram matrizes para a representação de *Textos* e *Padroes*. Uma constante chamada `MAX_PARAM` foi definida como sendo o tamanho máximo de um parâmetro passado para o programa em sua linha de comando. Essa constante foi utilizada também para limitar o tamanho destas duas matrizes (Textos e Padrões).

O Código 7 lista estes tipos de dados implementados na liguagem C bem como a definição das funções da implementação.

**Código 7** Tipos de Dados utilizados ao longo de toda a Implementação.

```c
// Constantes
#define MAX_PARAM  127   // Tamanho Máximo de um Parametro da Linha de Comando
#define MAX_LINHA  1000  // Tamanho Maximo de uma Linha no Arquivo de Entrada
#define MAX_CHAR   256   // Inteiro Maximo que representa um Caractere
#define MAX_ERRORS 10    // Maximo de Erros na Aproximacao
#define DEBUG      0     // Nivel de Debug ( 0=Minimo )

// Definicao das Funcoes Principais
//   --> BoyerMooreHorspool.c
int BoyerMooreHorspool(char *linha, char *padrao, int numLinha, int compat);
//   --> BoyerMooreHorspoolSunday.c
int BoyerMooreHorspoolSunday(char *linha, char *padrao, int numLinha, int compat);
//   --> ShiftAnd.c
int ShiftAnd(char *linha, char *padrao, int numLinha, int compat);
//   --> ShiftAndAprox.c
int ShiftAndAprox(char *linha, char *padrao, int numLinha, int aprox);
//   --> ProgDin.c
int ProgDin(char *linha, char *padrao, int numLinha, int aprox);

// Definicao das Funcoes Auxiliares --> Geral.c
void printCasamento(char *linha, char *padrao, int numLinha, int posicao);
void printHelp();
void chomp(char *string);
int timevalSubtract(struct timeval *result, struct timeval *x,struct timeval *y);
```

## 4.2 Boyer-Moore-Horspool

A implementação do Boyer-Moore-Horspool é chamada pelo programa principal (`Main.c`) quando este é executado na linha de comando com a opção `-a BMH`. Esta função tem a seguinte declaração:

**int** `BoyerMooreHorspool(`*char *linha, char *padrao, int numLinha, int compat*`);`

onde `char *linha` correponde à linha (texto) onde está sendo feita a pesquisa. Já `char *padrao` contém o padrão que está sendo pesquisado no texto. O parâmetro `int numLinha` contém o número da linha do arquivo que está sendo processada atualmente, para que esta possa ser impressa junto com o resultado de um eventual casamento. Já `int compat` não é utilizada nesta implementação e, como o próprio nome diz, está aí somente por compatibilidade, para que todas as funções de busca possam ter a mesma definição. Todas as informações necessárias para a execução da implementação estão contidas nestes parâmetros e nenhuma variável global é considerada.

A implementação deste algoritmo, que foi mostrado no Código 2, é mostrada no Código 8.

**Código 8** Implementação do Algoritmo de Boyer-Moore-Horspool.

```
int BoyerMooreHorspool(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long i, j, k;
  long d[MAX_CHAR+1];
  int comp=0;

  for (j=0; j<=MAX_CHAR; j++) d[j] = m;
  for (j=1; j<m; j++) d[(int)padrao[j-1]] = m-j;
  i = m;
  while (i<=n) {
    k=i;
    j=m;
    while ((linha[k-1] == padrao[j-1] && comp++) && j>0)  k--; j--;
    if (j==0) printCasamento(linha, padrao, numLinha, k+1);
    i+=d[(int)linha[i-1]];
  }
  return comp;
}
```

## 4.3 Boyer-Moore-Horspool-Sunday

A implementação do Boyer-Moore-Horspool-Sunday é chamada pelo programa principal (`Main.c`) quando este é executado na linha de comando com a opção `-a BMHS`. Esta função tem a seguinte declaração:

```
int BoyerMooreHorspoolSunday(char *linha, char *padrao, int numLinha, int compat);
```

onde `char *linha` correponde à linha (texto) onde está sendo feita a pesquisa. Já `char *padrao` contém o padrão que está sendo pesquisado no texto. O parâmetro `int numLinha` contém o número da linha do arquivo que está sendo processada atualmente, para que esta possa ser impressa junto com o resultado de um eventual casamento. Já `int compat` não é utilizada nesta implementação e, como o próprio nome diz, está aí somente por compatibilidade, para que todas as funções de busca possam ter a mesma definição. Todas as informações necessárias para a execução da implementação estão contidas nestes parâmetros e nenhuma variável global é considerada.

A implementação deste algoritmo, que foi mostrado no Código 3, é mostrada no Código 9.

## 4.4 Shift-And

A implementação do Shift-And é chamada pelo programa principal (`Main.c`) quando este é executado na linha de comando com a opção `-a SA`. Esta função tem a seguinte declaração:

**Código 9** Implementação do Algoritmo de Boyer-Moore-Horspool-Sunday.

```
int BoyerMooreHorspoolSunday(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long i, j, k;
  long d[MAX_CHAR+1];
  int comp=0;

  for (j=0; j<=MAX_CHAR; j++) d[j] = m + 1;
  for (j=1; j<m; j++) d[(int)padrao[j-1]] = m - j + 1;
  i = m;
  while (i<=n) {
    k=i;
    j=m;
    while ((linha[k-1] == padrao[j-1] && comp++) && j>0)  k--; j--;
    if (j==0) printCasamento(linha, padrao, numLinha, k+1);
    i+=d[(int)linha[i-1]];
  }
  return comp;
}
```

**int** `ShiftAnd(`*char *linha, char *padrao, int numLinha, int compat*`)`;

onde `char *linha` correponde à linha (texto) onde está sendo feita a pesquisa. Já `char *padrao` contém o padrão que está sendo pesquisado no texto. O parâmetro `int numLinha` contém o número da linha do arquivo que está sendo processada atualmente, para que esta possa ser impressa junto com o resultado de um eventual casamento. Já `int compat` não é utilizada nesta implementação e, como o próprio nome diz, está aí somente por compatibilidade, para que todas as funções de busca possam ter a mesma definição. Todas as informações necessárias para a execução da implementação estão contidas nestes parâmetros e nenhuma variável global é considerada.

A implementação deste algoritmo, que foi mostrado no Código 4, é mostrada no Código 10.

**Código 10** Implementação do Algoritmo Shift-And.

```
int ShiftAnd(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long Masc[MAX_CHAR];
  long i;
  long R = 0;

  if (DEBUG>8) printf("[ShiftAnd] Procurando por %s em %s \n", padrao, linha);

  for (i=0; i<MAX_CHAR; i++) Masc[i] = 0;
  for (i=1; i<=m; i++) Masc[padrao[i-1] + 127] |= 1 << (m-i);
  for (i=0; i<n; i++) {
    R = ((((unsigned long)R) >> 1) | (1 << (m-1))) & Masc[linha[i]+127];
    if ((R&1) != 0)
    printCasamento(linha, padrao, numLinha, i - m + 2);
  }
  return 0;
}
```

## 4.5 Shift-And Aproximado

A implementação do Shift-And Aproximado é chamada pelo programa principal (`Main.c`) quando este é executado na linha de comando com a opção `-a SA`. Esta função tem a seguinte declaração:

**int** `ShiftAndAprox`(*char *linha, char *padrao, int numLinha, int aprox*);

onde `char *linha` correponde à linha (texto) onde está sendo feita a pesquisa. Já `char *padrao` contém o padrão que está sendo pesquisado no texto. O parâmetro `int numLinha` contém o número da linha do arquivo que está sendo processada atualmente, para que esta possa ser impressa junto com o resultado de um eventual casamento. Já `int aprox` corresponde ao número de erros aceitáveis ($k$). Todas as informações necessárias para a execução da implementação estão contidas nestes parâmetros e nenhuma variável global é considerada.

A implementação deste algoritmo, que foi mostrado no Código 5, é mostrada no Código 11.

**Código 11** Implementação do Algoritmo Shift-And Aproximado.

```
int ShiftAndAprox(char *linha, char *padrao, int numLinha, int aprox) {
int n = strlen(linha);
int m = strlen(padrao);
long Masc[MAX_CHAR];
long i, j, Ri, Rant, Rnovo;
long R[MAX_ERRORS + 1];

if (DEBUG>8) printf("[ShiftAndAprox] Procurando por %s em %s\n", padrao, linha);

for (i=0; i<MAX_CHAR; i++) Masc[i] = 0;
for (i=1; i<=m; i++) Masc[padrao[i-1] + 127] |= 1 << (m - i);
R[0] = 0;
Ri = 1 << (m - 1);
for (j = 1; j<=aprox; j++) R[j] = (1 << (m-j)) | R[j-1];
for (i=0; i<n; i++) {
    Rant = R[0];
    Rnovo = ((((unsigned long)Rant) >> 1) | Ri) & Masc[linha[i] + 127];
    R[0] = Rnovo;
    for (j = 1; j<=aprox; j++) {
      Rnovo = ((((unsigned long)R[j]) >> 1) & Masc[linha[i] + 127]) | Rant |
              (((unsigned long)(Rant | Rnovo)) >> 1);
      Rant = R[j];
      R[j] = Rnovo | Ri;
    }
    if ((Rnovo & 1) != 0) printCasamento(linha, padrao, numLinha, i-(strlen(padrao))+2);
  }
  return 0;
  }
```

## 4.6 Programação Dinâmica

A implementação utilizando Programação Dinâmica é chamada pelo programa principal (`Main.c`) quando este é executado na linha de comando com a opção `-a PD`. Esta função tem a seguinte declaração:

```
int ProgDin(char *linha, char *padrao, int numLinha, int aprox);
```

onde `char *linha` correponde à linha (texto) onde está sendo feita a pesquisa. Já `char *padrao` contém o padrão que está sendo pesquisado no texto. O parâmetro `int numLinha` contém o número da linha do arquivo que está sendo processada atualmente, para que esta possa ser impressa junto com o resultado de um eventual casamento. Já `int aprox` corresponde ao número de erros aceitáveis ($k$). Todas as informações necessárias para a execução da implementação estão contidas nestes parâmetros e nenhuma variável global é considerada.

A implementação deste algoritmo, que foi mostrado no Código 6, é mostrada no Código 12.

**Código 12** Implementação da Busca utilizando Programação Dinâmica.

```
int ProgDin(char *linha, char *padrao, int numLinha, int aprox) {
  int n = strlen(linha);
  int m = strlen(padrao);

  int C[MAX_LINHA];
  int i, lact, pos, pC, nC;

  if (DEBUG>8) printf("[ProgDin] Procurando por %s em %s\n", padrao, linha);

  for (i=0; i<=m; i++) C[i]=i;
  lact = aprox + 1;

  for (pos=1; pos<=n; pos++) {
    pC = 0;
    nC = 0;
    for (i=1; i<=lact; i++) {
      if (padrao[i-1] == linha[pos-1]) nC = pC;
      else {
        if (pC < nC) nC = pC;
        if (C[i] < nC) nC = C[i];
        nC = nC + 1;
      }
      pC = C[i];
      C[i] = nC;
    }
    while (C[lact] > aprox) lact = lact-1;
    if (lact == m) printCasamento(linha, padrao, numLinha, pos-m+1);
    else lact = lact + 1;
  }
  return 0;
}
```

| Coleção | Tam(MB) | Linhas |
|---|---|---|
| `wsj89` | 2.8 | 43495 |
| `wsj88_10` | 10 | 140540 |
| `wsj88_20` | 20 | 278934 |
| `wsj88` | 109 | 1523894 |

Tabela 1: Características dos arquivos da coleção TREC.

# 5 Testes e Resultados

O objetivo dos experimentos realizados foi de executar um conjunto de programas para recuperar ocorrências de padrões em arquivos constituídos de documentos.

Os programas utilizados foram os citados neste trabalho:

- *Boyer-Moore-Horspool*
- *Boyer-Moore-Horspool-Sunday*
- *Shift-And*
- *Shift-And Aproximado*
- *Programação Dinâmica (Aproximado)*
- *AGrep (Aproximado)*

Os textos utilizados nos experimentos foram os arquivos de documentos da coleção TREC disponíveis na rede do DCC/UFMG. Estes arquivos possuem as características citadas na Tabela 1.

Um exemplo de registro deste arquivo encontra-se a seguir:

```
<DOC>
<DOCNO> WSJ890802-0125 </DOCNO>
<DD> = 890802 </DD>
<AN> 890802-0125. </AN>
<HL> Inside Track:
@  NCNB Director Sold Big Holding in July
@  Worth $7.4 Million More 4 Weeks Later
@  ----
@  By Alexandra Peers and John R. Dorfman
@  Staff Reporters of The Wall Street Journal </HL>
<DD> 08/02/89 </DD>
<SO> WALL STREET JOURNAL (J) </SO>
<CO> NCB BPCO TRN EGGS LABOR </CO>
<IN> STOCK MARKET, OFFERINGS (STK)
SECURITIES INDUSTRY (SCR) </IN>
<TEXT>
   Even insiders make mistakes.
Sometimes, big, fat, $7 million-dollar mistakes.

(...)

</TEXT>
</DOC>
```

| Num | Abrev. | Padrão Completo |
|---|---|---|
| *1* | DCC .. | `DCC` |
| *2* | That.. | `That` |
| *3* | Stoc.. | `Stock` |
| *4* | Doll.. | `Dollar` |
| *5* | Uber.. | `Uberaba` |
| *6* | Exch.. | `Exchange` |
| *7* | Mana.. | `Manacapuru` |
| *8* | Pric.. | `Price Index` |
| *9* | Admi.. | `Administration` |
| *10* | DCC.. | `DCC UFMG dollar` |
| *11* | Bra.. | `Brazilian coffee` |
| *12* | New.. | `New York Stock Exchange` |
| *13* | Mac.. | `Macunaima administration` |
| *14* | The.. | `The agency sees widespread use` |
| *15* | man.. | `many investors now riding the Tokyo rocket are so` |
| *16* | Low.. | `Low wages and a large work force attract the investors, while suprisingly robust agricultural` |
| *17* | Mr... | `Mr. Jordan said the provision allowing non-lawyers to be partners reflects shifts in providing legal services that have` |

Tabela 2: Padrões utilizados nos experimentos bem como a identificação do número e da abreviação de cada um.

Já os padrões utilizados para serem buscados nos textos foram os listados na Tabela 2. Para facilitar a criação das tabelas e dos gráficos mostrados nesta seção e no Anexo A, cada um dos padrões utilizados poderá ser identificado pelo seu número e/ou pela sua abreviação, mostrada também na Tabela 2. Em outros casos abreviações maiores diferentes da mostrada poderão ser utilizadas, mas sempre acompanhado do número do padrão.

Um dos fatores que influenciam bastante no tempo de execução bem como no número de comparações realizados pelos algoritmos é o tamanho do padrão utilizado na busca e também o número de ocorrências do padrão no texto pesquisado. Desta forma, a Tabela 3 mostra estas informações de tamanho e de ocorrências em cada uma das coleções utilizadas.

As implementações dos algoritmos foram executadas em uma máquina AMD Athlon XP 2400+ (2.00 GHz) com 512MB de Memória. O sistema operacional utilizado em todos os testes foi o Sistema Linux Kernel 2.4 em um ambiente totalmente modo texto, ou seja, sem o gerenciador de janelas X rodando, o que liberou mais memória para a implementação. Todos os serviços foram terminados de modo ainda a liberar mais memória.

## 5.1 Validação da Implementação

Para **validar** a Implementação bem como ilustrar os resultados obtidos, foi realizada a pesquisa do padrão `Brazilian` utilizando os cinco algoritmos implementados e a

| **Padrão** | **Tam.** | **Oc. Wsj89** | **Oc. Wsj88_10** | **Oc. Wsj88_20** | **Oc. Wsj88** |
|---|---|---|---|---|---|
| *1. DCC* | 3 | 7 | 0 | 5 | 90 |
| *2. That* | 4 | 8972 | 794 | 1728 | 232 |
| *3. Stock* | 5 | 9504 | 776 | 1629 | 208 |
| *4. Dollar* | 6 | 788 | 75 | 150 | 15 |
| *5. Uberaba* | 7 | 0 | 0 | 0 | 2 |
| *6. Exchange* | 8 | 10279 | 833 | 1680 | 210 |
| *7. Manacapuru* | 10 | 0 | 0 | 0 | 67 |
| *8. Price Index* | 11 | 43 | 8 | 7 | 5 |
| *9. Administration* | 14 | 1336 | 106 | 244 | 39 |
| *10. DCC UFMG dollar* | 15 | 0 | 0 | 0 | 17 |
| *11. Brazilian coffee* | 16 | 14 | 0 | 1 | 1 |
| *12. New York Stock ..* | 23 | 4674 | 353 | 784 | 91 |
| *13. Macunaima admin ..* | 24 | 0 | 0 | 0 | 4 |
| *14. The agency sees ..* | 30 | 1 | 1 | 0 | 0 |
| *15. many investors ..* | 49 | 1 | 1 | 0 | 0 |
| *16. Low wages and a ..* | 93 | 1 | 0 | 1 | 0 |
| *17. Mr. Jordan said ..* | 119 | 1 | 0 | 1 | 0 |

Tabela 3: Tamanho e número de ocorrências dos padrões utilizados.

coleção `wsj89`.

### Implementação do Boyer-Moore-Horspool

```
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                     ^^^^^
^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                             ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


              ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                  ^^^^^^^^
^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                        ^^^^^^^^^
```

### Implementação do Boyer-Moore-Horspool-Sunday

```
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                     ^^^^^
^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                             ^^^^^^^^^
```

```
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


              ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                  ^^^^^^^^
^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                        ^^^^^^^^^
```

## Implementação do Shift-And

```
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                     ^^^^^
^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                             ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


              ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                  ^^^^^^^^
^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                        ^^^^^^^^^
```

## Implementação do Shift-And Aproximado com k=1

```
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                    ^^^^^^
^^^
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                     ^^^^^
^^^^
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                      ^^^^
^^^^^
Linha  2527: Another dealer said great unhappiness about the severe drop in prices was exp
ressed at a meeting last weekend in Colombia attended by representatives from that nation,
 Brazil and Mexico.


 ^^^^^^^^^
Linha  2529: As a result, the dealer said, rumors circulated early yesterday that Brazil a
nd Colombia might close registrations of coffee sales.
                                                                                  ^^^^^^^^
^
Linha  2531: Brazil and Colombia later denied the rumors, the dealer said.
             ^^^^^^^^^
```

```
Linha  3432:    Until the information was disclosed Monday at a congressional hearing on t
he so-called export enhancement program, the administration had planned to subsidize the s
ale of 100,000 tons of wheat to Yemen, 100,000 tons of barley malt to Brazil and 50,000 to
ns of barley to Iraq.


                                                                      ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                            ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                             ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                              ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


             ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


              ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


               ^^^^^^^^^
Linha 14328: Representatives from Colombia, Brazil and Mexico met in Cartagena, Colombia,
during the weekend, but there was no indication of whether the talks made any progress, th
e dealer said.
                                            ^^^^^^^^^
Linha 25254:    The decision to renew the steel quotas, and the earlier Super-301 actions
against Japan, Brazil and India, underline the pressures that the Bush administration face
s from Congress to move trade policy away from the multilateralism embodied in the General
 Agreement on Tariffs and Trade.

               ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                 ^^^^^^^^^

Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                  ^^^^^^^^
^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                   ^^^^^^^
^^
Linha 28822:    Traders were apparently ignoring a meeting on the current market situation
 scheduled for this weekend in Colombia and involving that nation as well as Brazil and Me
xico, according to Arthur Stevenson, analyst for Prudential-Bache Securities Inc., New Yor
k.

                                                                             ^^^^^^^^^
Linha 28837:    "But the weather in Brazil has been very stable and mild, and the lack of
threatening cold weather has spurred selling by Brazil and others," he said.

                                                ^^^^^^^^^
```

```
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                       ^^^^^^^^^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                        ^^^^^^^^^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                         ^^^^^^^^^
Linha 42662:    And a similar pattern is evident elsewhere: The private-sector debts of Br
azil and Argentina have also fallen sharply while the two countries' governments have beco
me further mired.
                                                                                        ^^
^^^^^^^
```

## Implementação do Programação Dinâmica(Aproximado) com k=1

```
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                    ^^^^^^
^^^
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                     ^^^^^
^^^^
Linha  2486: U.S. farmers have stepped up soybean sales recently, and beans from the Brazi
lian harvest have flowed onto world markets as well.
                                                                                      ^^^^
^^^^^
Linha  2527: Another dealer said great unhappiness about the severe drop in prices was exp
ressed at a meeting last weekend in Colombia attended by representatives from that nation,
 Brazil and Mexico.


 ^^^^^^^^^
Linha  2529: As a result, the dealer said, rumors circulated early yesterday that Brazil a
nd Colombia might close registrations of coffee sales.
                                                                                  ^^^^^^^^
^
Linha  2531: Brazil and Colombia later denied the rumors, the dealer said.
             ^^^^^^^^^
Linha  3432:    Until the information was disclosed Monday at a congressional hearing on t
he so-called export enhancement program, the administration had planned to subsidize the s
ale of 100,000 tons of wheat to Yemen, 100,000 tons of barley malt to Brazil and 50,000 to
ns of barley to Iraq.


                                                                      ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                            ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                             ^^^^^^^^^
Linha 11583: @  To Japanese, Brazilian Firms </HL>
                              ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


             ^^^^^^^^^
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


              ^^^^^^^^^
```

```
Linha 11590:    One month after it spent US$350 million to acquire Quebec Cartier Mining C
o., Dofasco Inc. sold half of its interest in the big iron ore mine to a Japanese trading
company and a Brazilian industrial concern.


               ^^^^^^^^^
Linha 14328: Representatives from Colombia, Brazil and Mexico met in Cartagena, Colombia,
during the weekend, but there was no indication of whether the talks made any progress, th
e dealer said.
                                            ^^^^^^^^^
Linha 25254:    The decision to renew the steel quotas, and the earlier Super-301 actions
against Japan, Brazil and India, underline the pressures that the Bush administration face
s from Congress to move trade policy away from the multilateralism embodied in the General
 Agreement on Tariffs and Trade.

               ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                  ^^^^^^^^^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                   ^^^^^^^^
^
Linha 25450: Investors who bought the Brazil Fund in April when VBI Corp. announced a 9.9%
 stake would have lost a bundle; shortly after the news, the fund was rocked by a Brazilia
n market crisis, and the fund's shares plunged about 25%.

                                                                                    ^^^^^^^
^^
Linha 28822:    Traders were apparently ignoring a meeting on the current market situation
 scheduled for this weekend in Colombia and involving that nation as well as Brazil and Me
xico, according to Arthur Stevenson, analyst for Prudential-Bache Securities Inc., New Yor
k.

                                                                              ^^^^^^^^^
Linha 28837:    "But the weather in Brazil has been very stable and mild, and the lack of
threatening cold weather has spurred selling by Brazil and others," he said.

                                                ^^^^^^^^^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                        ^^^^^^^^^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                         ^^^^^^^^^
Linha 28852:    As an example of what's available now, the dealer said, Brazilian coffee c
an be purchased for 73.50 cents a pound and Colombian at 86 cents.
                                                                          ^^^^^^^^^
Linha 42662:    And a similar pattern is evident elsewhere: The private-sector debts of Br
azil and Argentina have also fallen sharply while the two countries' governments have beco
me further mired.
                                                                                         ^^
^^^^^^^
```

## 5.2 Experimentos Realizados

Basicamente três tipos de experimentos foram realizados:

- *Tempo de Busca dos Algoritmos:* Ver 5.2.1.
- *Tempo de Busca com Variação da Aproximação:* Ver 5.2.2.
- *Comparações de Caracteres para BMH e BMHS:* Ver 5.2.3.

### 5.2.1 Tempo de Busca dos Algoritmos

O objetivo principal deste tipo de experimento é tentar entender como é que ocorre a variação do tempo de execução de cada um dos algoritmos em conjunto diante de modificações do padrão utilizado na busca e também diante de modificações no tamanho do texto. Queremos responder a perguntas do tipo:

- *Qual(is) o(s) algoritmo(s) mais eficiente(s)?*
- *Em quais casos os algoritmos são mais eficientes?* Lembre-se que estamos lidando aqui com variações nos padrões e nos texto. Os padrões, em específico possuem características como por exemplo o seu tamanho e o seu número de ocorrências, que influenciam e muito a execução dos algoritmos. Já para os textos, o que mais influencia é o seu tamanho.
- *Qual o comportamento dos algoritmos diante de padrões diferentes (tamanho e ocorrências)?*
- *Qual o comportamento dos algoritmos diante de textos diferentes (tamanho)?*

Para a realização deste tipo de experimento, cada um dos padrões foi buscado em cada uma das coleções (textos) utilizando cada um dos algoritmos. Cada busca foi realizada 10 (dez) vezes para cada um dos padrões e a média do tempo de execução, bem como o desvio padrão, foram registrados. Ao todo, foram realizadas $17 * 4 * 6 * 10 = 4080$ execuções.

Os algoritmos que fazem busca aproximada também foram incluídos neste tipo de teste. Nestes casos, a aproximação $k = 0$ foi utilizada para que uma busca exata fosse realizada com esses algoritmos e uma comparação mais realista fosse possível. Apesar destes algoritmos terem sido incluídos, fica claro que o objetivo principal deste tipo de teste é mesmo fazer a comparação entre os algoritmos exatos. Os algoritmos aproximados são analisados no próximo tipo de experimento.

### 5.2.2 Tempo de Busca com Variação da Aproximação

O objetivo principal deste tipo de experimento é procurar entender como é o comportamento (i.e., tempo de execução) dos algoritmos de busca aproximada estudados diante de variações na aproximação (número de erros permitidos - $k$) utilizada.

Uma vez que os algoritmos são estudados e analisados em conjunto, este experimento permite também uma comparação mais realista do que a realizada no tipo de experimento anterior, no que diz respeito aos algoritmos de busca aproximada.

Sendo assim, levando em consideração estes algoritmos, procura-se responder as seguintes questões:

- *Qual(is) o(s) algoritmo(s) aproximados mais eficiente(s)?*
- *Em quais casos os algoritmos são mais eficientes?* Neste tipo, além das variações de texto e de padrões, deve-se levar em conta principalmente o número de erros permitidos (aproximação - $k$).
- *Qual o comportamento dos algoritmos diante de aproximações diferentes?*

- *Qual o comportamento dos algoritmos diante de aproximações e padrões diferentes (tamanho e ocorrências)?*
- *Qual o comportamento dos algoritmos diante de aproximações e textos diferentes (tamanho)?*

Para a realização deste tipo de experimento, cada um dos padrões foi buscado em cada uma das coleções (textos) utilizando cada um dos algoritmos sendo estes executados com aproximação de 0 a 5. Cada busca foi realizada 10 (dez) vezes e a média do tempo de execução, bem como o desvio padrão, foram registrados. Ao todo foram realizadas $17 * 4 * 3 * 6 * 10 = 12240$ execuções.

Os algoritmos de busca exata não foram considerados. Logo os experimentos só foram executados para os algoritmos Shift-And Aproximado, Programação Dinâmica e AGrep.

### 5.2.3 Comparações de Caracteres para BMH e BMHS

Conforme mencionado na Seção 2.3, o algoritmo Boyer-Moore-Horspool-Sunday é uma variante do algoritmo Boyer-Moore-Horspool que procura fazer um menor número de comparações entre caracteres através de deslocamentos maiores.

A idéia principal para este tipo de experimento é verificar o quanto a simples modificação de Sunday melhora o algoritmo original no que diz respeito ao número de comparações entre caracteres. A análise no que diz respeito ao tempo de execução foi realizada nos experimentos do primeiro tipo, explicado anteriormente.

Desta forma, uma vez realizadas tais experimentos será possível de se responder a perguntas do tipo:

- *O BMHS realmente faz um número menor de comparações do que o BMH?*
- *Quanto que o BMHS é melhor que o BMH?*
- *Quando que o BMHS é melhor que o BMH?*
- *Como varia o número de comparações realizadas pelos dois algoritmos ao variar o padrão utilizado na busca?*
- *Como varia o número de comparações realizadas pelos dois algoritmos ao variar o texto utilizado?*

Para a realização deste tipo de experimento, cada um dos padrões foi buscado em cada uma das coleções (textos) utilizando cada um dos dois algoritmos da família BMH. Cada busca foi realizada 10 (dez) vezes e o número de comparações entre caracteres foi registrado. Ao todo foram realizadas $17 * 4 * 2 * 10 = 1360$ execuções.

Como pode-se perceber, nestes experimentos apenas os algoritmos Boyer-Moore-Horspool e Boyer-Moore-Horspool-Sunday foram considerados.

Em todos os experimentos realizados, as informações relativas à execução do AGrep para os padrões 16 e 17 foram ignoradas ou colocadas em 0 (zero), pois este programa não aceitou um padrão com os tamanhos destes citados.

## 5.3 Resultados Obtidos

Conforme mencionado, uma quantidade razoável de dados foi gerada como resultados dos diversos experimentos. Tais dados serão mostrados através de gráficos e tabelas. Nesta seção, será trabalhado principalmente com os gráficos. Todos os dados que geraram tais gráficos desta seção estão tabelados no Anexo A e serão referenciados sempre que possível.

Além dos dados dos gráficos desta Seção, no Anexo A encontram-se também as informações de desvio padrão das médias geradas bem como apresenta alguns dados que não foram mostrados em gráficos. Logo, uma consulta aos dados lá armazenados pode ser bastante útil.

Os resultados dos experimentos serão agrupados por tipo, ou seja:

- *Tempo de Busca dos Algoritmos:* Ver 5.3.1.
- *Tempo de Busca com Variação da Aproximação:* Ver 5.3.2.
- *Comparações de Caracteres para BMH e BMHS:* Ver 5.3.3.

### 5.3.1 Tempo de Busca dos Algoritmos

Conforme mencionado, o objetivo principal deste tipo de experimento é tentar entender como é que ocorre a variação do tempo de execução de cada um dos algoritmos em conjunto diante de modificações do padrão utilizado na busca e também diante de modificações no tamanho do texto.

Apesar dos algoritmos aproximados terem sido incluídos nas tabelas e nos gráficos, eles não serão levados muito em consideração por terem um funcionamento diferente dos algoritmos de busca exata. Além disso, estes algoritmos serão melhor estudados no segundo tipo de experimentos (ver Seção 5.3.2).

O *primeiro experimento* realizado neste tipo foi comparar o *tempo de execução dos algoritmos para cada padrão* citado na Tabela 2.

Os gráficos das Figuras 5, 6, 7 e 8 correspondem à execução deste experimento para as Coleções `wsj89`, `wsj88_10`, `wsj88_20` e `wsj88`, respectivamente. A idéia destes gráficos é comparar a execução de cada um dos algoritmos mostrando o tempo de execução entre eles. Permite também mostrar como cada padrão (que possui sua característica própria de número de caracteres e de número de ocorrências) se comporta em cada um dos algoritmos.

Já os gráficos da Figura 9(a, b, c e d), correspondem à execução deste mesmo experimento para as coleções `wsj89`, `wsj88_10`, `wsj88_20` e `wsj88`, porém não considerando os padrões em si, e sim os seus tamanhos. Através destes gráficos, é possível ter uma idéia de como o tempo de execução dos algoritmos variam com o tamanho do padrão.

Uma análise destes gráficos mostra que na média de todos os casos, considerando os algoritmos de busca exata, o algoritmo Boyer-Moore-Horpool-Sunday obteve os melhores tempos, seguido bem perto do Boyer-Moore-Horpool e por último, com uma certa distância, o Shift-And.

Agora considerando rapidamente os algoritmos com busca aproximada (com excessão da implementação do AGrep), tem-se que estes apresentaram tempos maiores, com excessão do algoritmo utilizando Programação Dinâmica que conseguiu

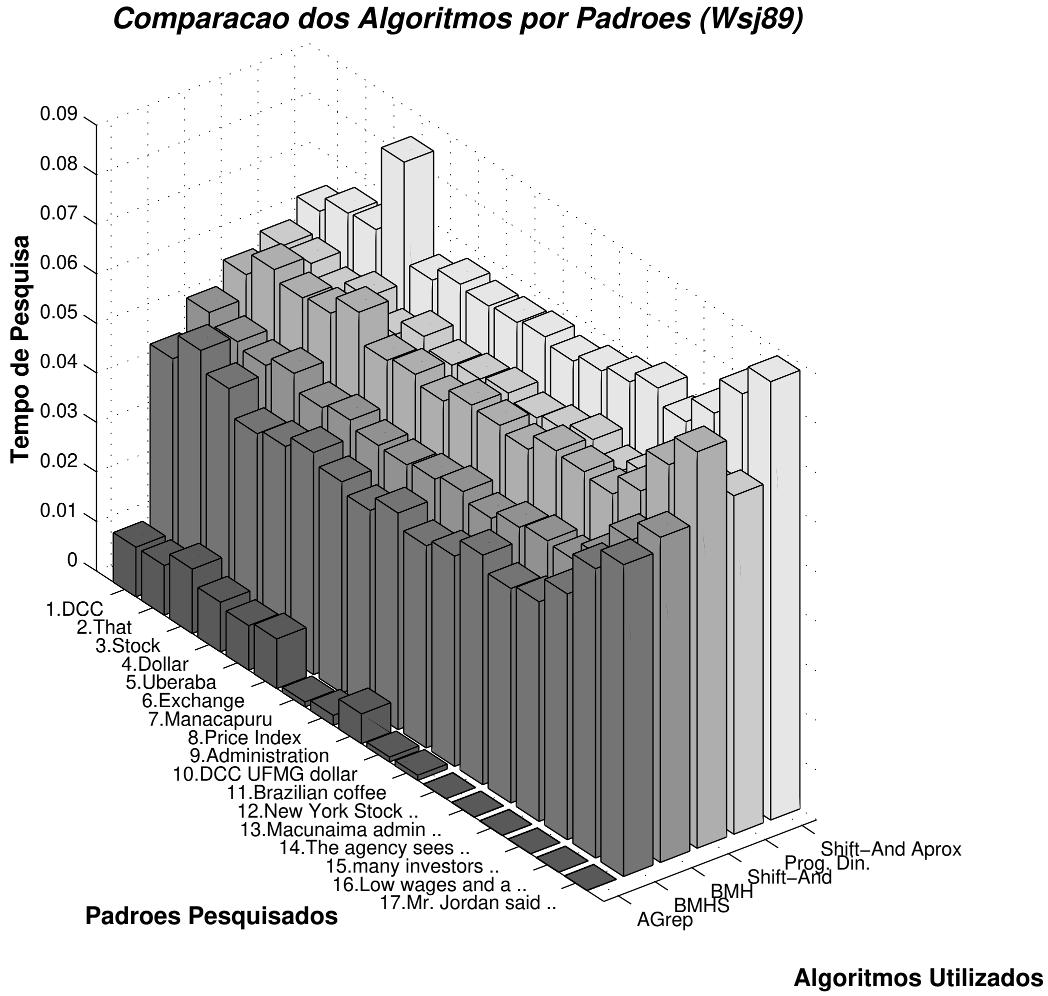



Figura 5: Comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj89`.

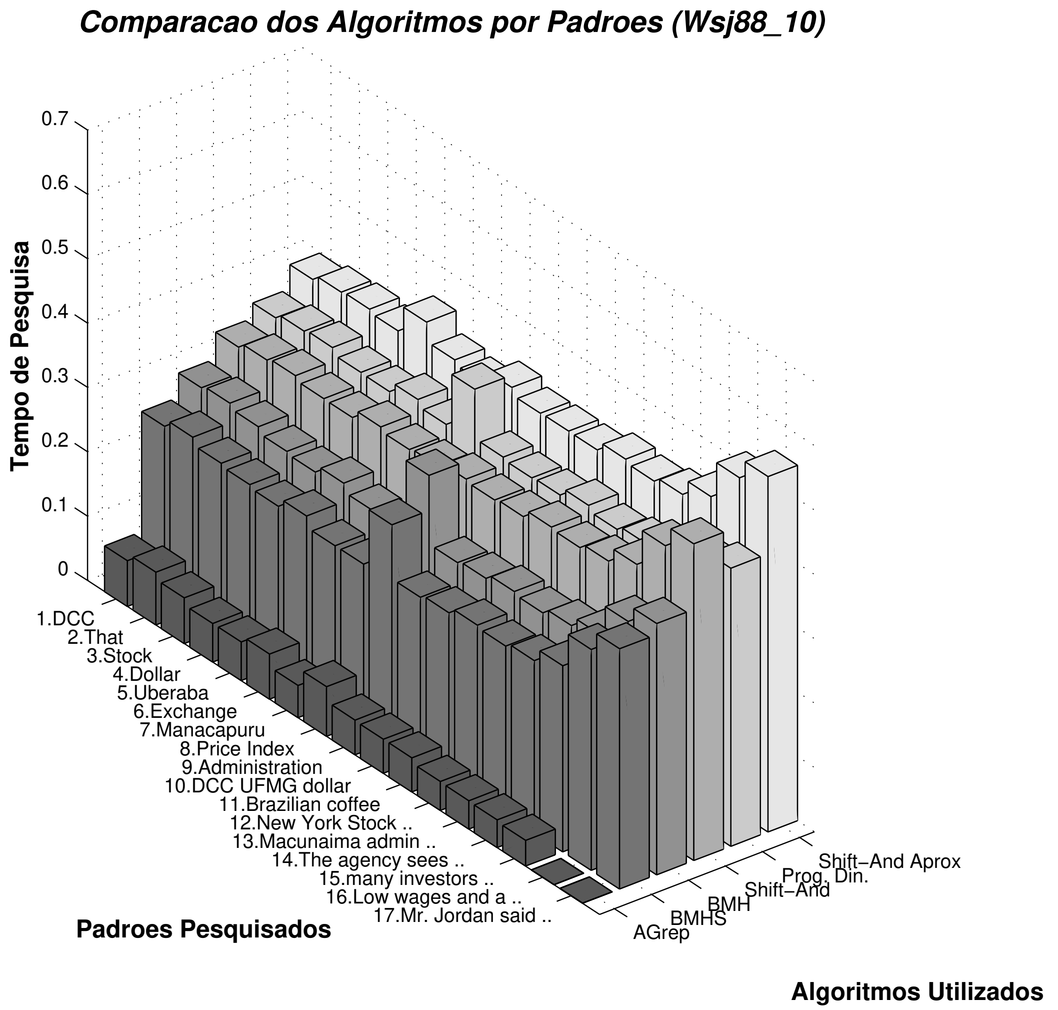



Figura 6: Comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj88_10`.

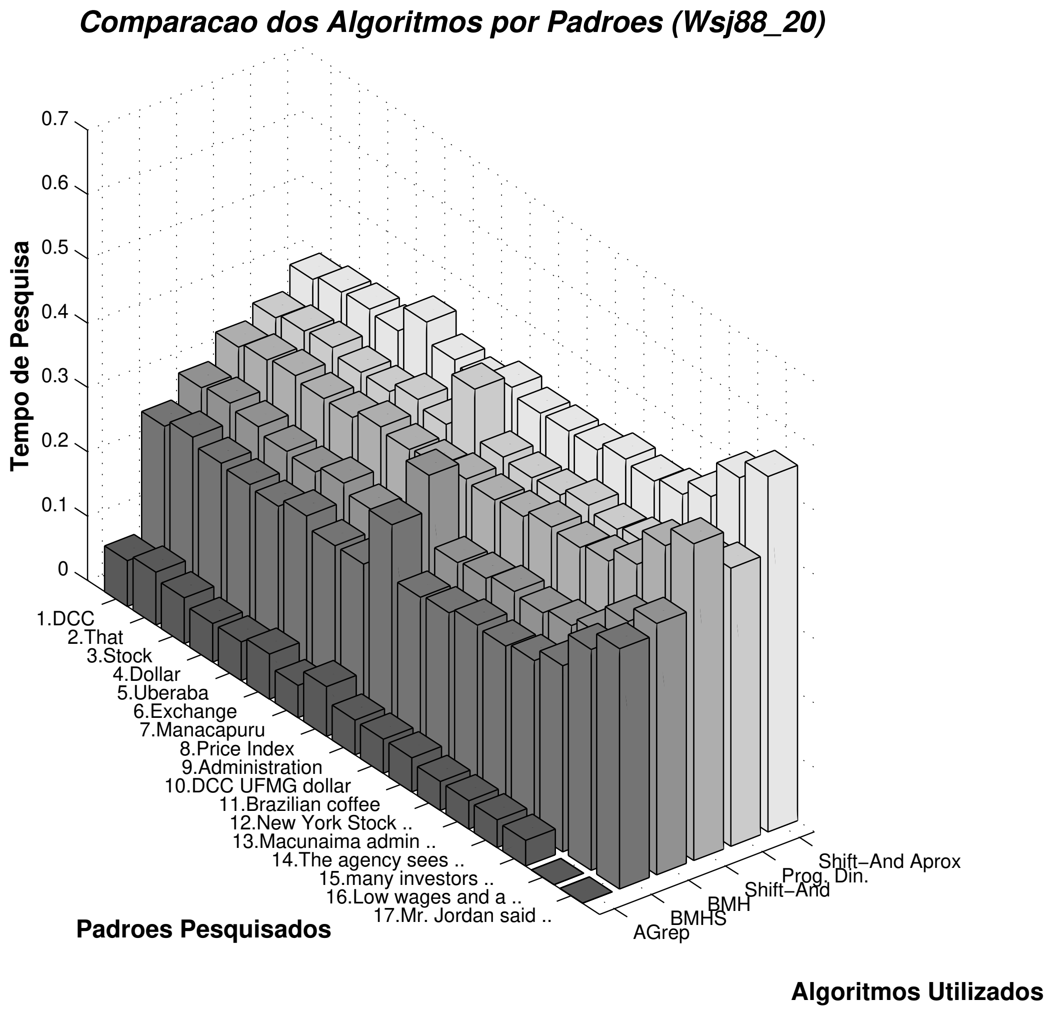



Figura 7: Comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj89_20`.

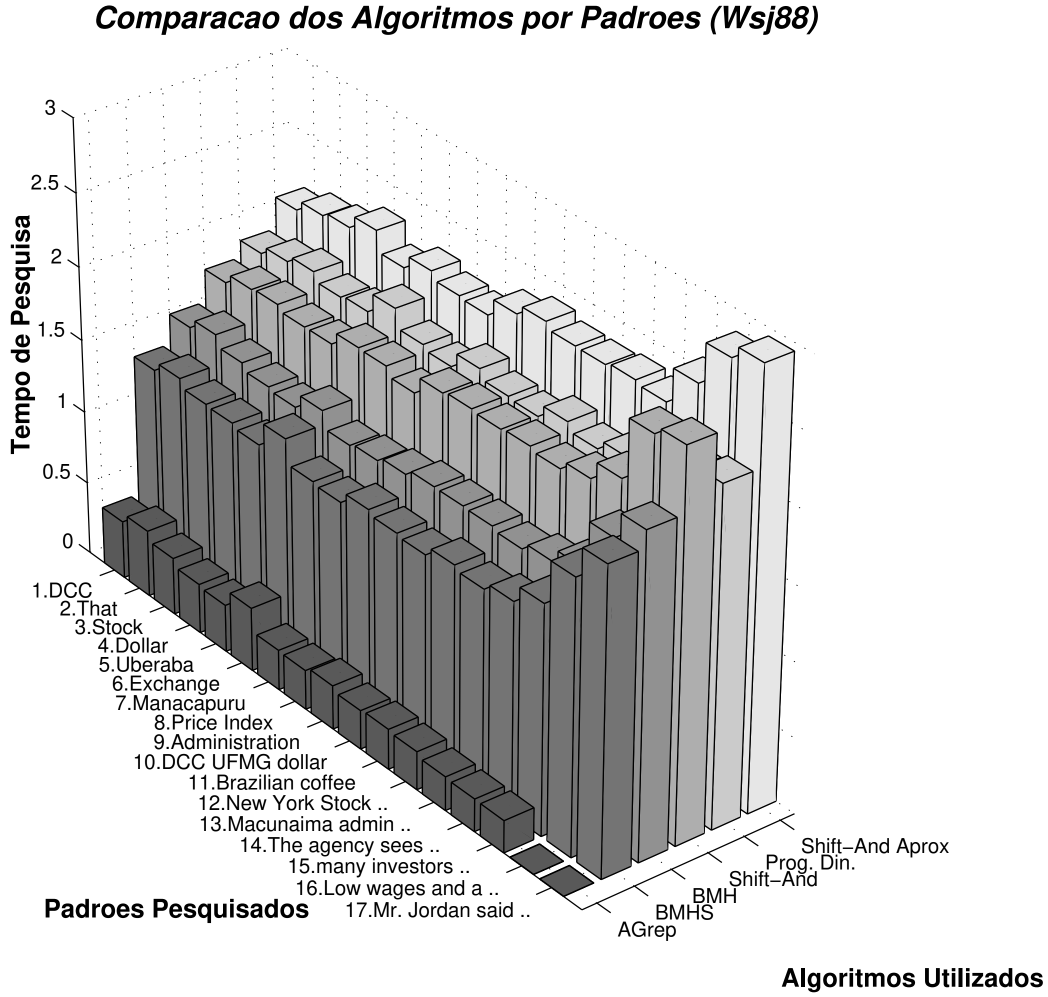



Figura 8: Comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj88`.

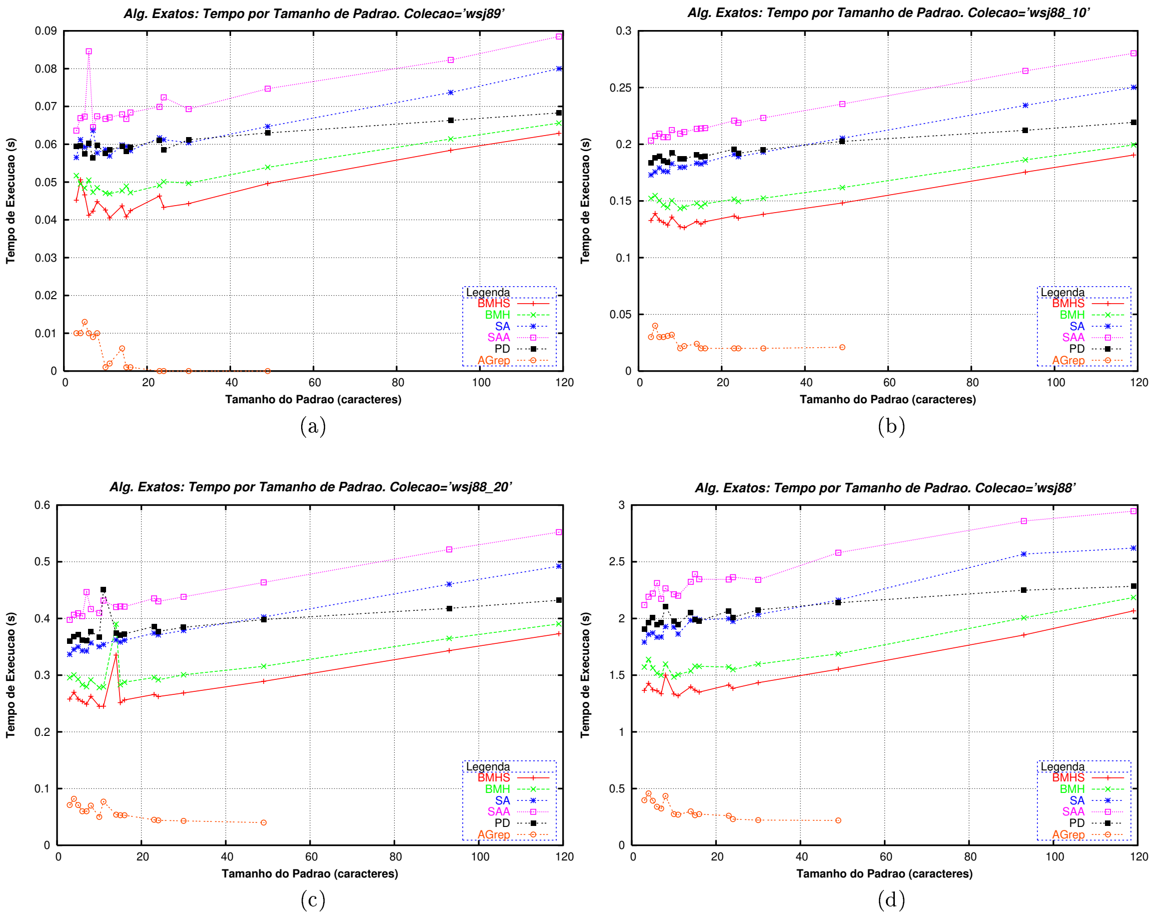



Figura 9: Comparação do tempo de execução dos algoritmos pelo tamanho do padrão para cada coleção.

ficar abaixo do Shift-And para tamanhos grandes de padrão. Para $k = 0$, o algoritmo utilizando Programação Dinâmica conseguiu resultados melhores do que o algoritmo Shift-And Aproximado. Já a implementação do AGrep, foi o que conseguiu os melhores resultados em todos os casos.

Os dados gerados pelos experimentos são mostrados nas tabelas do Anexo A. As informações de desvio padrão também são incluídas. Desta forma, as Tabelas 4, 5, 6 e 7, correspondem aos dados de comparação do tempo de execução de cada padrão em cada um dos algoritmos para as coleções `Wsj89`, `Wsj88_10`, `Wsj88_20` e `Wsj88`, respectivamente.

Um *segundo experimento* foi realizado que procurou comparar o *tempo de execução dos algoritmos com a variação no tamanho do texto.* Como cada padrão tem um comportamento um pouco diferente, para cada um destes padrões foi procurado em coleções de tamanhos diferentes para cada um dos algoritmos e o tempo foi coletado para gerar os gráficos mostrados nas Figuras 10, 11 e 12.

Conforme esperado, o que pode-se observar nestes gráficos é que o tempo de execução dos algoritmos aumentam conforme o aumento do tamanho do texto. Para alguns padrões este aumento no tempo é maior do que para outros, dependendo do seu tamanho, do número de ocorrências e do algoritmo que está sendo utilizado.

A Tabela 8, corresponde aos dados de comparação do tempo de execução e desvio padrão dos algoritmos com a variação no tamanho do texto para cada um dos padrões estudados.

### 5.3.2 Tempo de Busca com Variação da Aproximação

Neste tipo de experimentos, enfocado principalmente nos algoritmos de busca aproximada, será analisado e comparado o comportamento destes algoritmos enquanto é variado o número permitido de erros ($k$).

Em todos os casos, a nível de comparações, o número de erros $k$ foi considerado como sendo de 0 (zero) até o menor número entre a metade do tamanho do padrão e 5 (cinco).

Um *primeiro experimento* foi realizado considerando os diferentes padrões em conjunto com a variação da aproximação de um determinado algoritmo para os arquivos de coleções. A idéia é ver como a variação do padrão e da aproximação influenciam um determinado algoritmo de busca. Desta forma, o gráficos das Figuras 13, 14 e 15 mostram o comportamento dos algoritmos Shift-And Aproximado, Programação Dinâmica e a implementação AGrep ao variar o padrão e a aproximação para a coleção `Wsj88`.

A análise dos gráficos mostra claramente que com o aumento da aproximação, há o aumento do tempo de execução dos três algoritmos para todos os padrões estudados.

As Tabelas 9, 10 e 11 correspondem aos dados de comparação do tempo de execução dos algoritmos SAA, PD e AGrep ao variar o padrão e a aproximação para a coleção `Wsj88` que foram plotados nos gráficos já citados. Apesar de não ter sido gerado outros gráficos, por não se achar necessário, os dados dos algoritmos citados referentes às coleções `Wsj88_20`, `Wsj88_10` e `Wsj89` encontram-se no Anexo A.

Para fazer uma comparação no tempo de execução entre os algoritmos com busca aproximada, um *segundo experimento* foi realizado. A idéia aqui é verificar o quanto

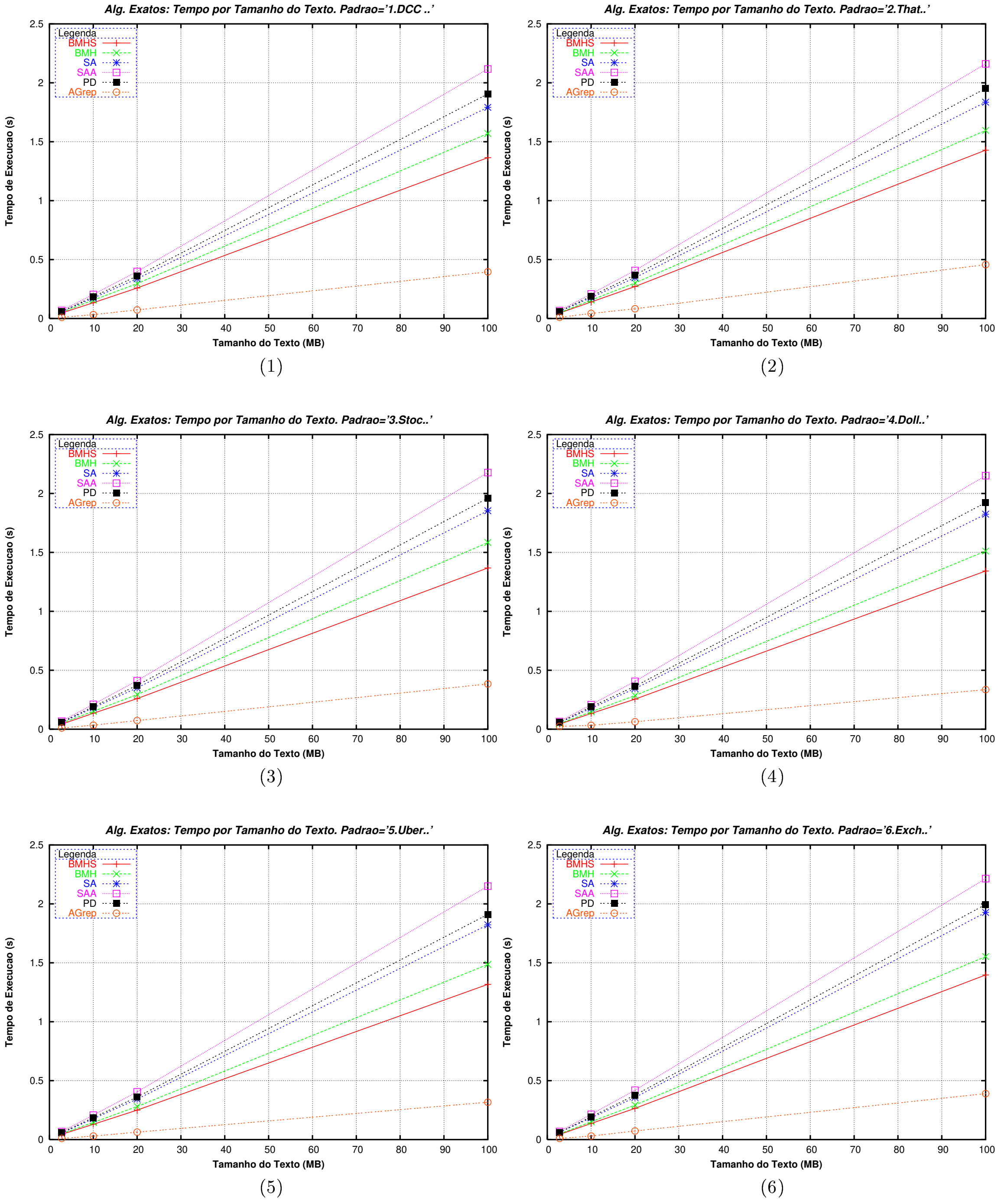
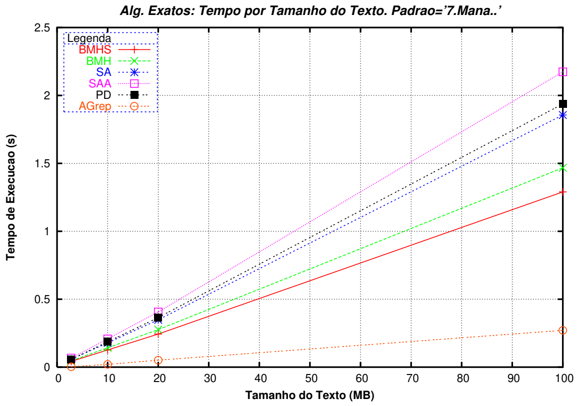
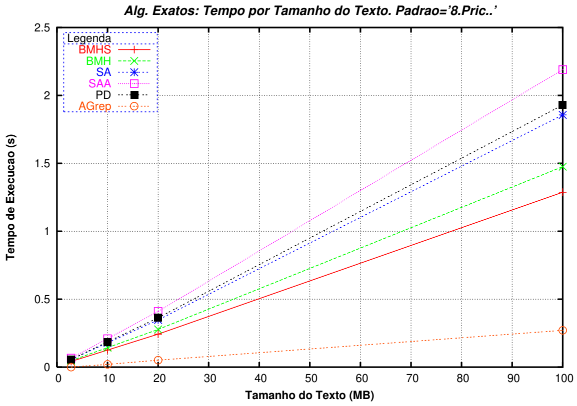


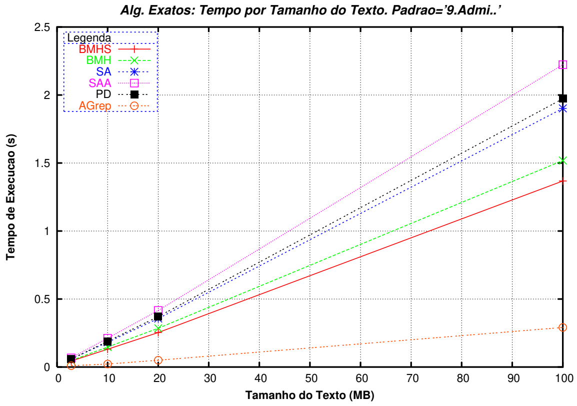

Figura 10: Comparação do tempo de execução dos algoritmos pelo tamanho do texto para cada padrão (Parte 1).



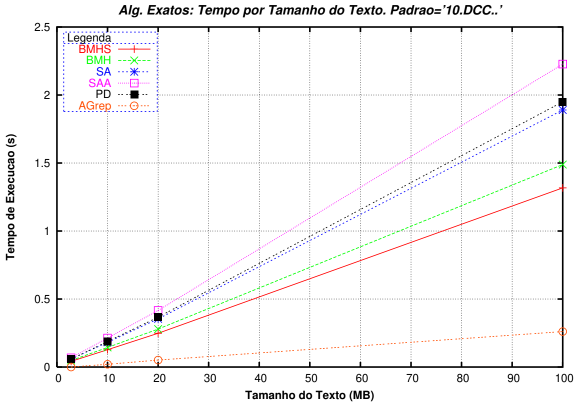

(7)



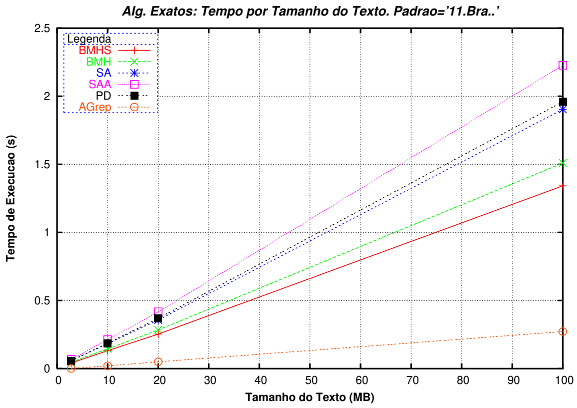

(8)



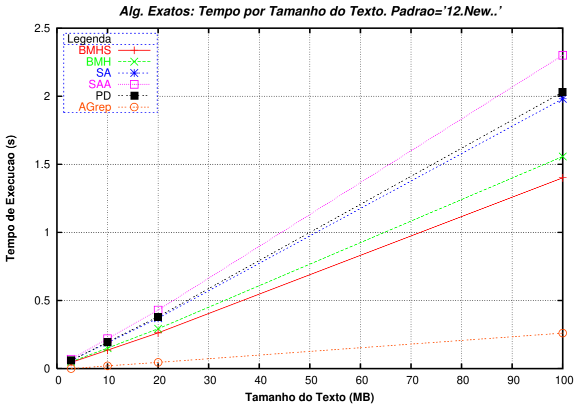

(9)




(10)

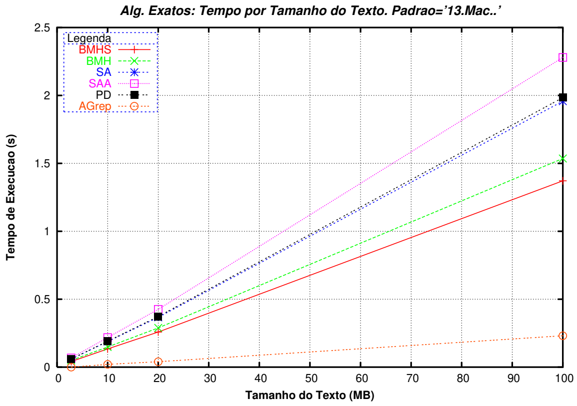



(11)

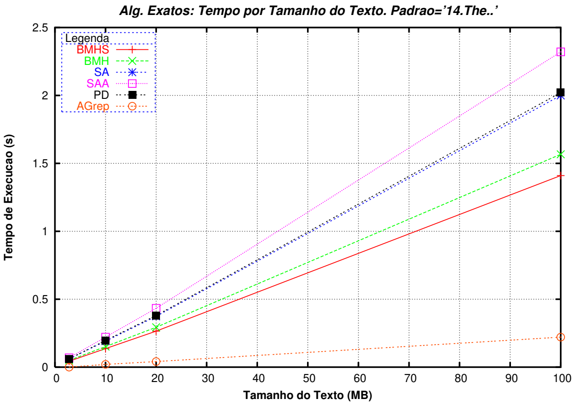



(12)

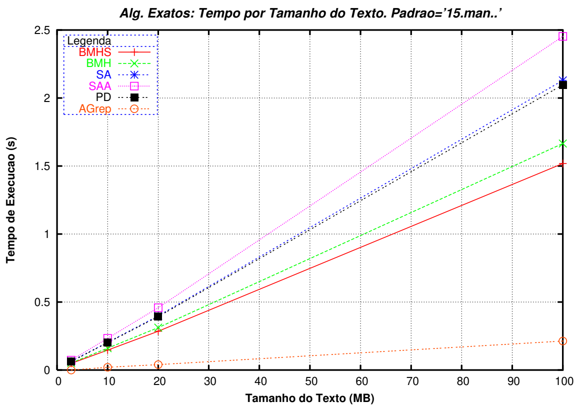
Figura 11: Comparação do tempo de execução dos algoritmos pelo tamanho do texto para cada padrão (Parte 2).



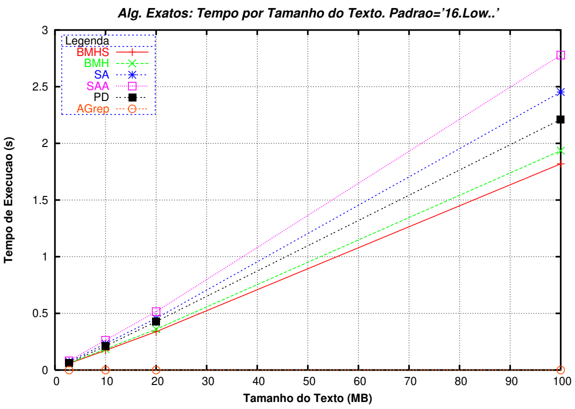

(13)



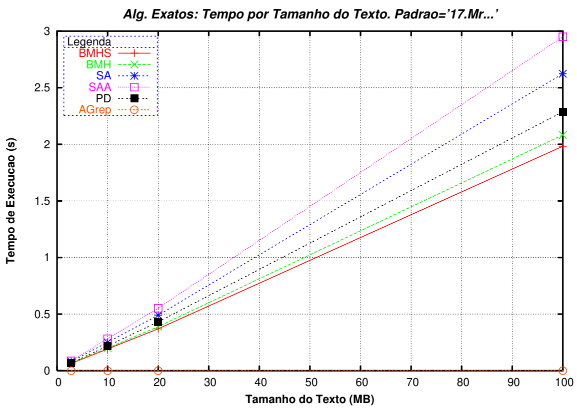

(14)




(15)




(16)




(17)

Figura 12: Comparação do tempo de execução dos algoritmos pelo tamanho do texto para cada padrão (Parte 3).

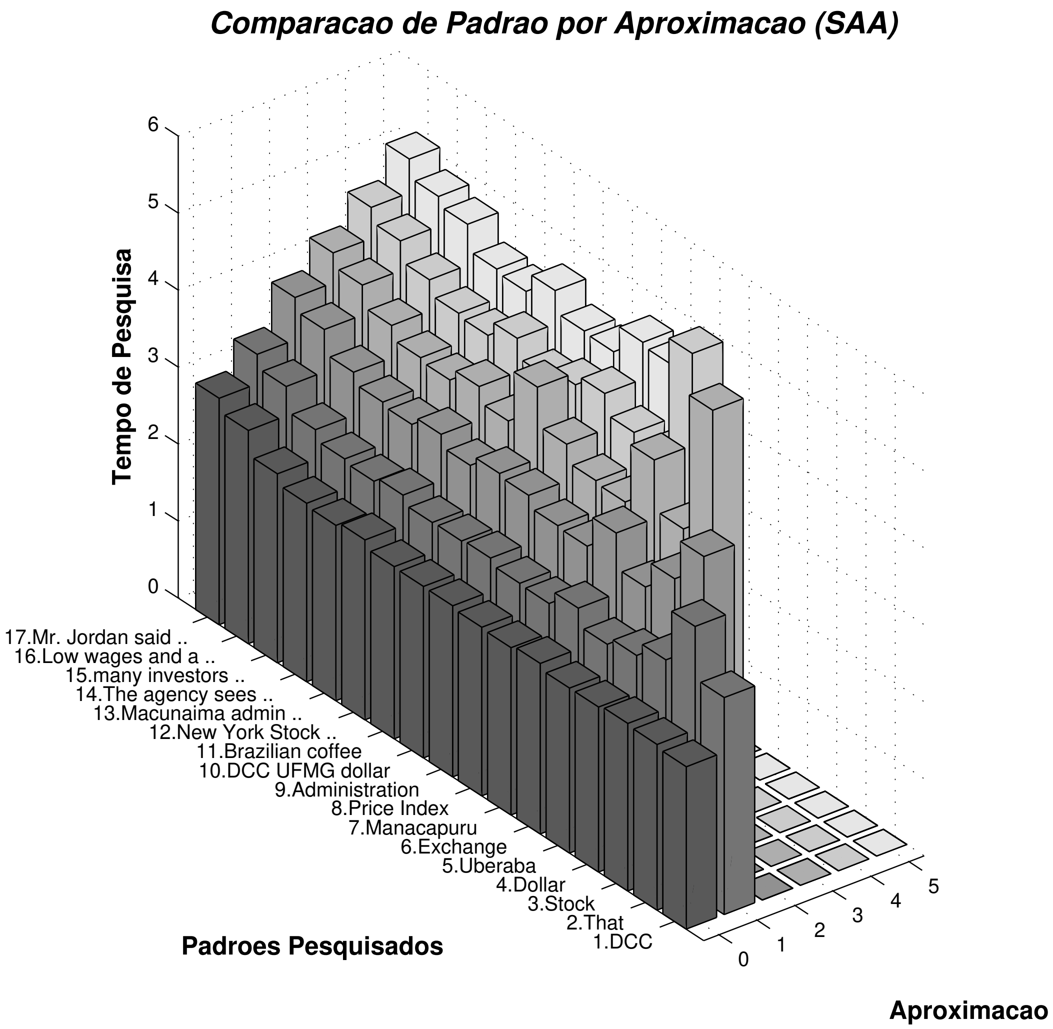



Figura 13: Comparação do tempo de execução do algoritmo Shift-And Aproximado ao variar o padrão e a aproximação para a coleção `Wsj88`.

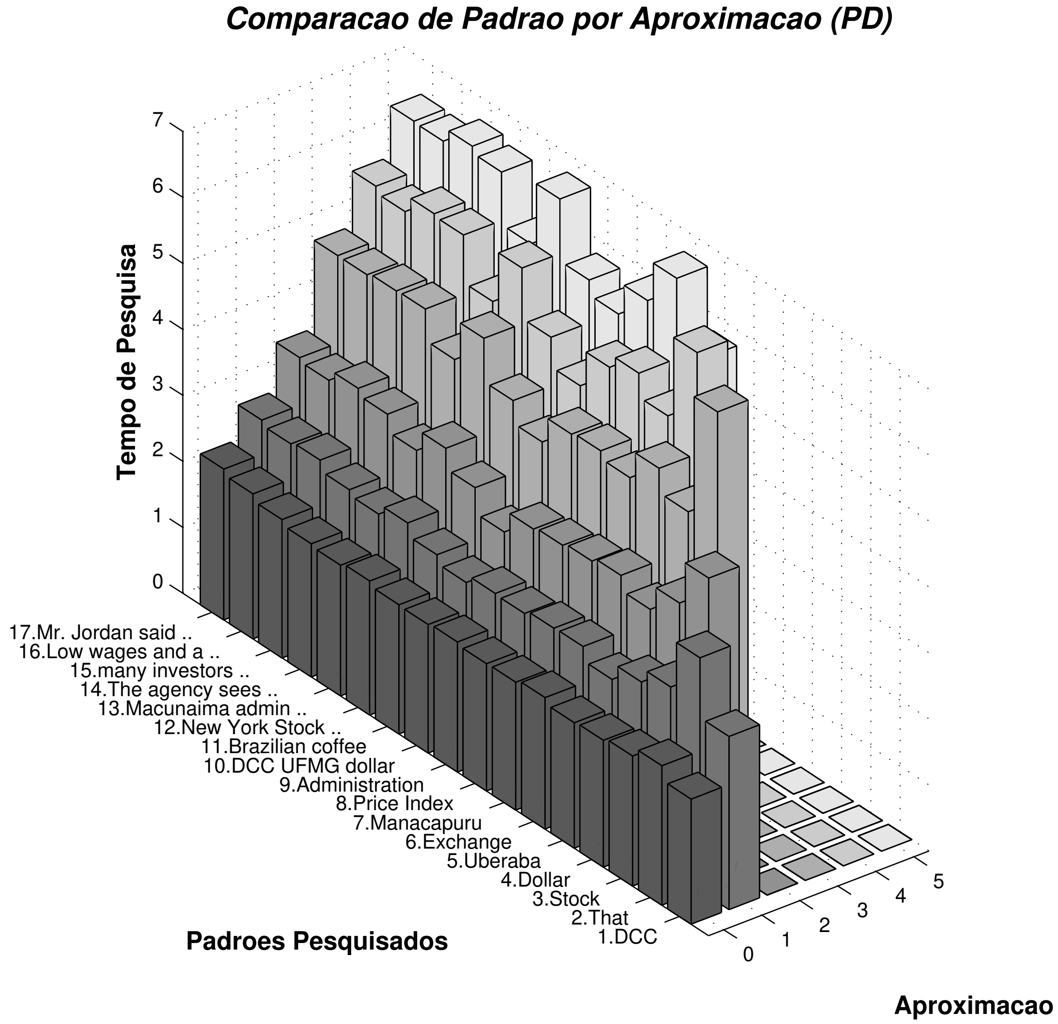



Figura 14: Comparação do tempo de execução do algoritmo de Programação Dinâmica ao variar o padrão e a aproximação para a coleção `Wsj88`.

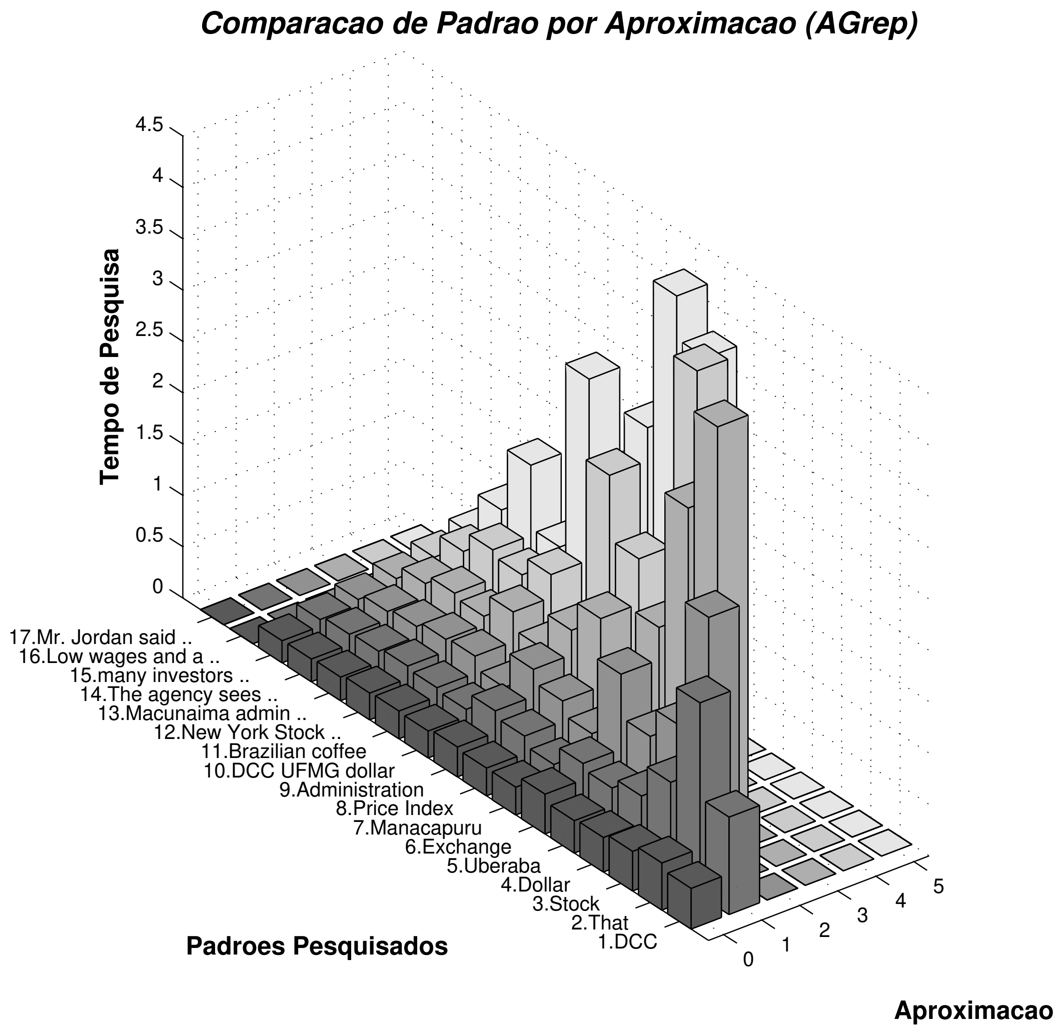



Figura 15: Comparação do tempo de execução da implementação AGrep ao variar o padrão e a aproximação para a coleção `Wsj88`.

um algoritmo é melhor ou pior do que o outro em todas as aproximações estudadas (de 0 a 5) e para cada um dos padrões estudados. Desta forma, para cada padrão um gráfico foi gerado mostrando o comportamento dos algoritmos ao buscar este padrão com diferentes aproximações (erros - $k$). Tais gráficos são ilustados nas Figuras 16, 17 e 18.

Analisando-se os diversos gráficos, pode-se concluir que com o aumento da aproximação $k$, o tempo de execução da implementação tente a crescer. Observa-se também que para $k$ menor ou perto de 2, a implementação de Programação Dinâmica apresentou tempos menores que o Shift-And, enquanto que se aumentarmos mais um pouco $k$, esta observação se inverte. Já o AGrep, obteve resultados muito inferiores em todos os casos, porém observa-se que com $k$ muito maiores, esta implementação obteve um aumento substancial do tempo de execução.

A Tabela 21 mostra os dados ilustrados nestes gráficos e que corresponde aos dados de comparação do tempo de execução e desvio padrão da execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88` (referente aos gráficos gerados). A coleção `Wsj88` foi escolhida por ser a maior coleção e proporcionar números mais confiáveis. Porém, apesar de não terem sido plotados os gráficos, os dados para as coleções `Wsj88_20`, `Wsj88_10` e `Wsj89` encontram-se no Anexo A.

### 5.3.3 Comparações de Caracteres para BMH e BMHS

Conforme mencionado, a idéia principal para este tipo de experimento é verificar o quanto a simples modificação de Sunday melhora o algoritmo original no que diz respeito ao número de comparações entre caracteres.

Para isso, novamente dois experimentos foram realizados. No *primeiro experimento*, a idéia é analisar como o número de comparações varia nos dois algoritmos com a utilização de padrões diferentes. Desta forma, os gráficos das Figuras 19, 20, 21 e 22 ilustram o resultado deste experimento para as coleções `Wsj89`, `Wsj88_10`, `Wsj88_20` e `Wsj88`, respectivamente.

Já os gráficos da Figura 23(a, b, c e d), correspondem à execução deste mesmo experimento para as coleções `wsj89`, `wsj88_10`, `wsj88_20` e `wsj88`, porém não considerando os padrões em si, e sim os seus tamanhos. Através destes gráficos, é possível ter uma idéia de como o número de comparações dos algoritmos variam com o tamanho do padrão.

Por estes gráficos é fácil observar dois pontos principais. O primeiro é que nem sempre o BMHS é melhor do que o BMH no que diz respeito ao número de comparações de caracteres. Observa-se que para padrões pequenos (com até 5 ou 6 caracteres) o ganho em comparações é bem considerado. Para padrões médios, pelo menos até o padrão de número 10 (que possui 15 caracteres), houve ainda um ganho, mas muito pouco. Já para padrões maiores nota-se que na maioria dos casos o BMH chega a fazer um número menor de comparações que o BMHS.

Porém, isso não quer dizer que nestes últimos casos o BMH chegue a ser melhor do que o BMHS. Note que nos gráficos mostrados anteriormente na comparação dos tempos, o BMHS obteve sempre tempos de execução menores do que o BMH. Isso deve-se ao menor número de acesso à memória que o BMHS faz, o que o deixa mais rápido mesmo diante de um maior número de comparações (que acontece com padrões grandes).

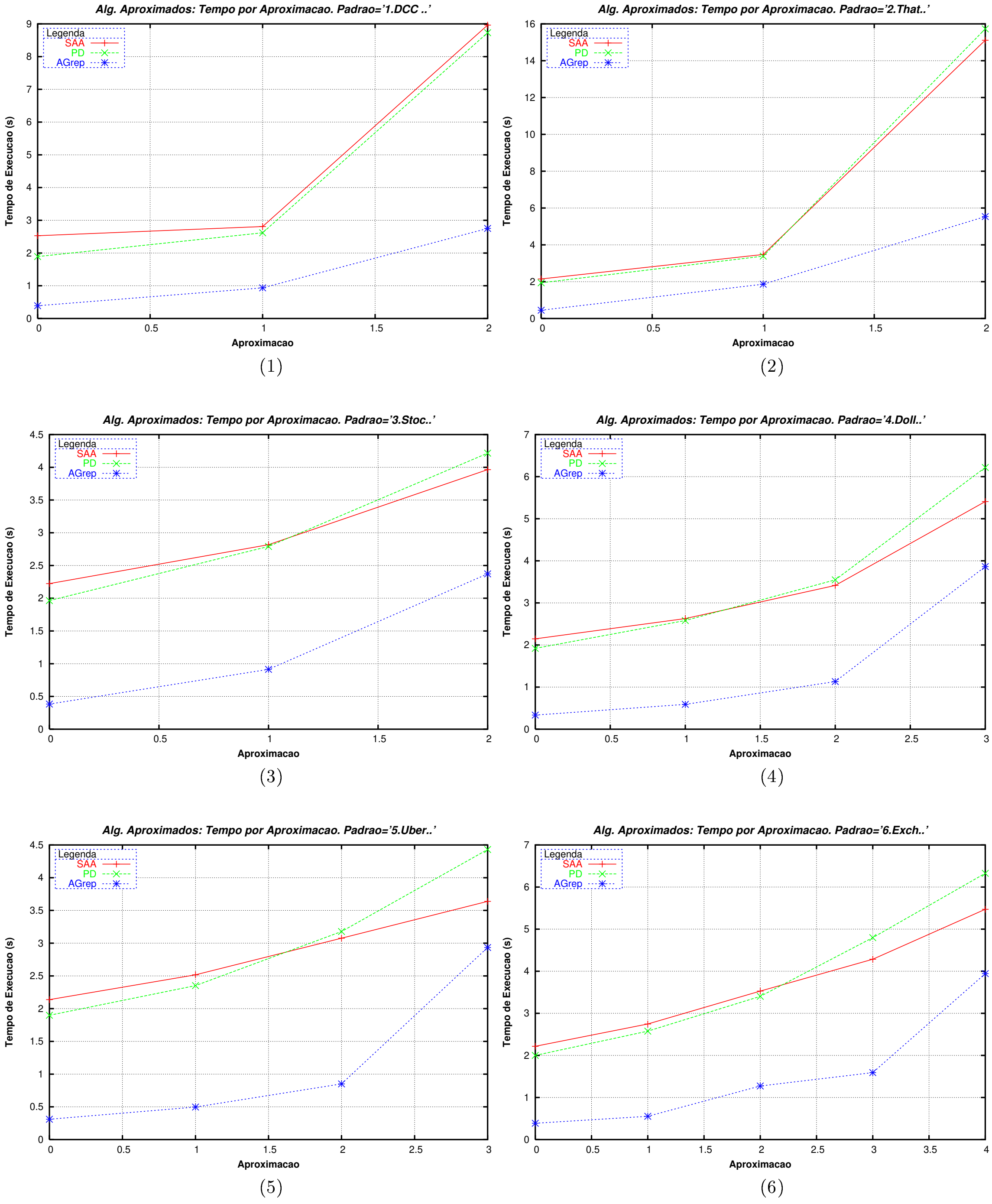
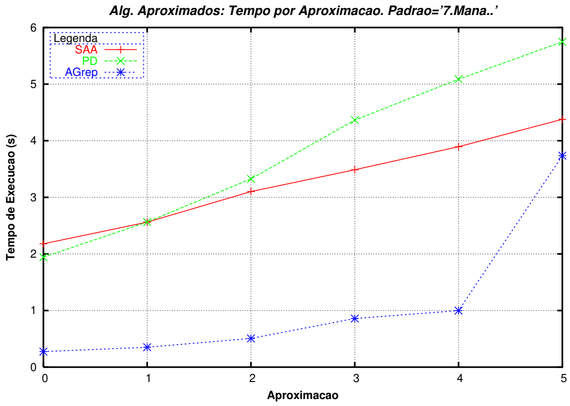
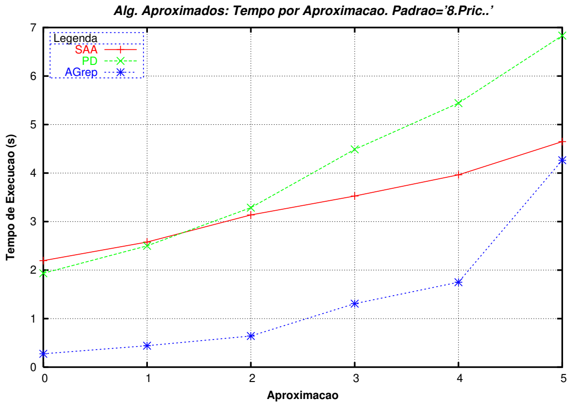


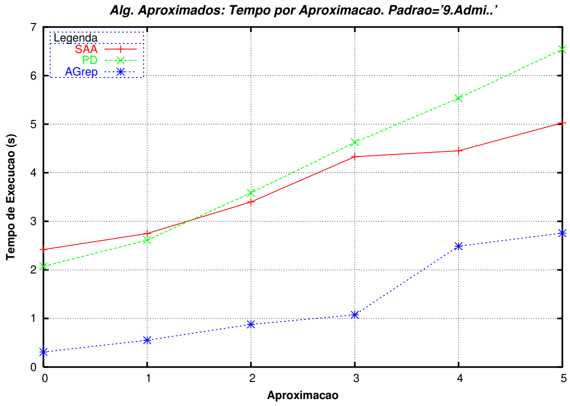

Figura 16: Comparação dos tempos de execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88` (Parte 1).



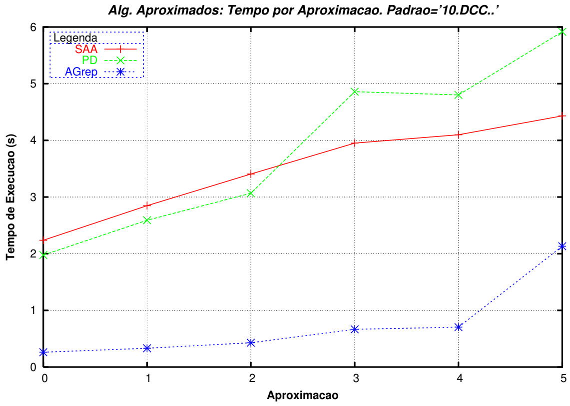

(7)



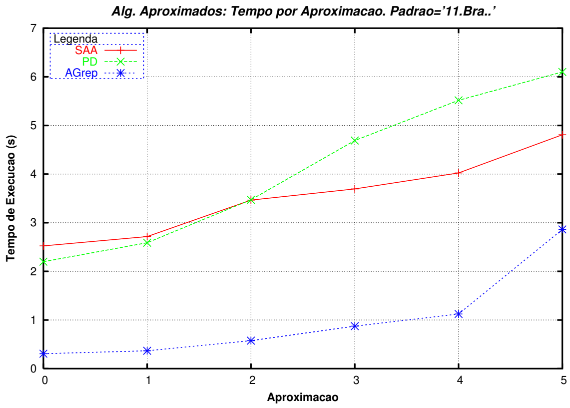

(8)



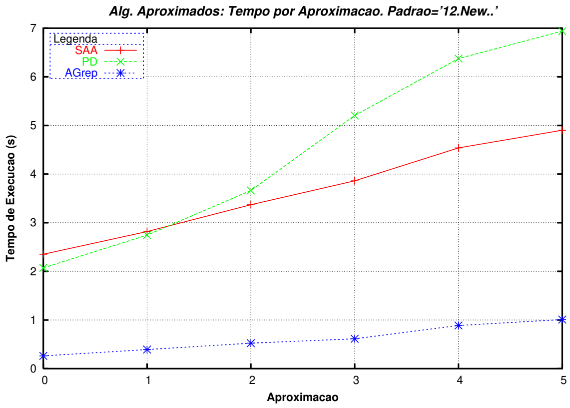

(9)




(10)

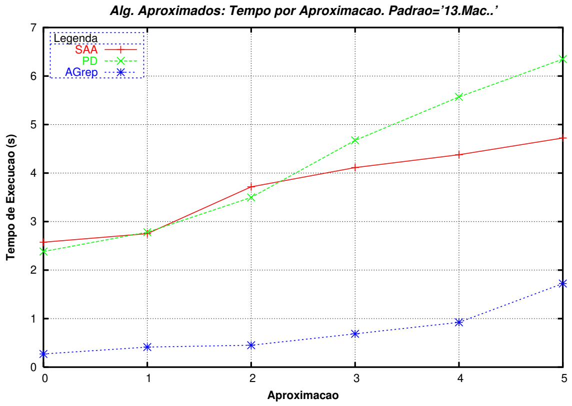



(11)

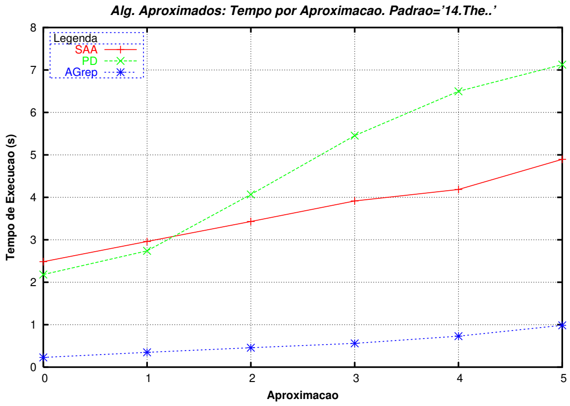



(12)

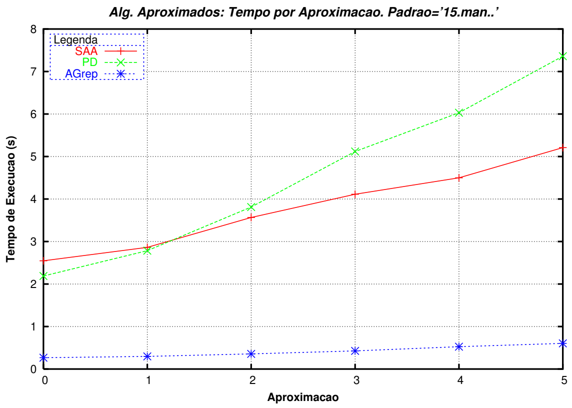
Figura 17: Comparação dos tempos de execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88` (Parte 2).



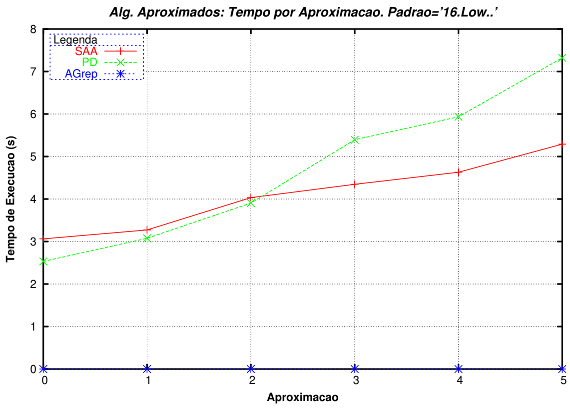

(13)



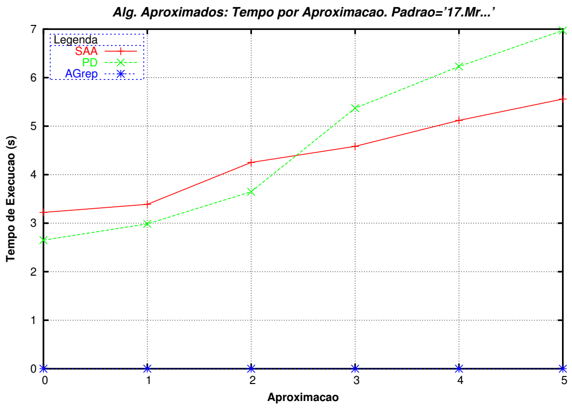

(14)




(15)




(16)




(17)

Figura 18: Comparação dos tempos de execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88` (Parte 3).

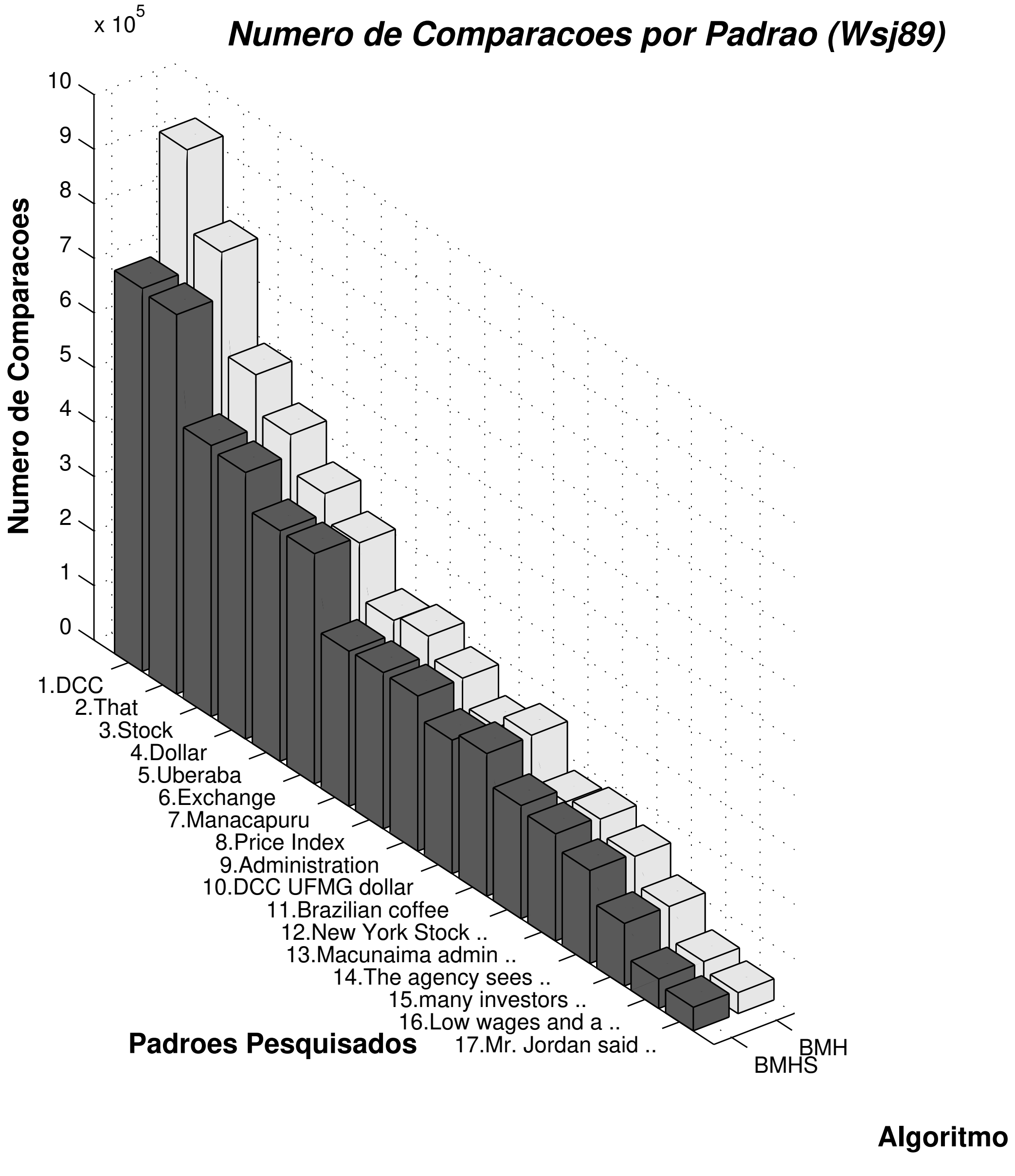



Figura 19: Gráfico de número de comparações por padrão para os algoritmos BMH e BMHS e para a coleção `Wsj89`.

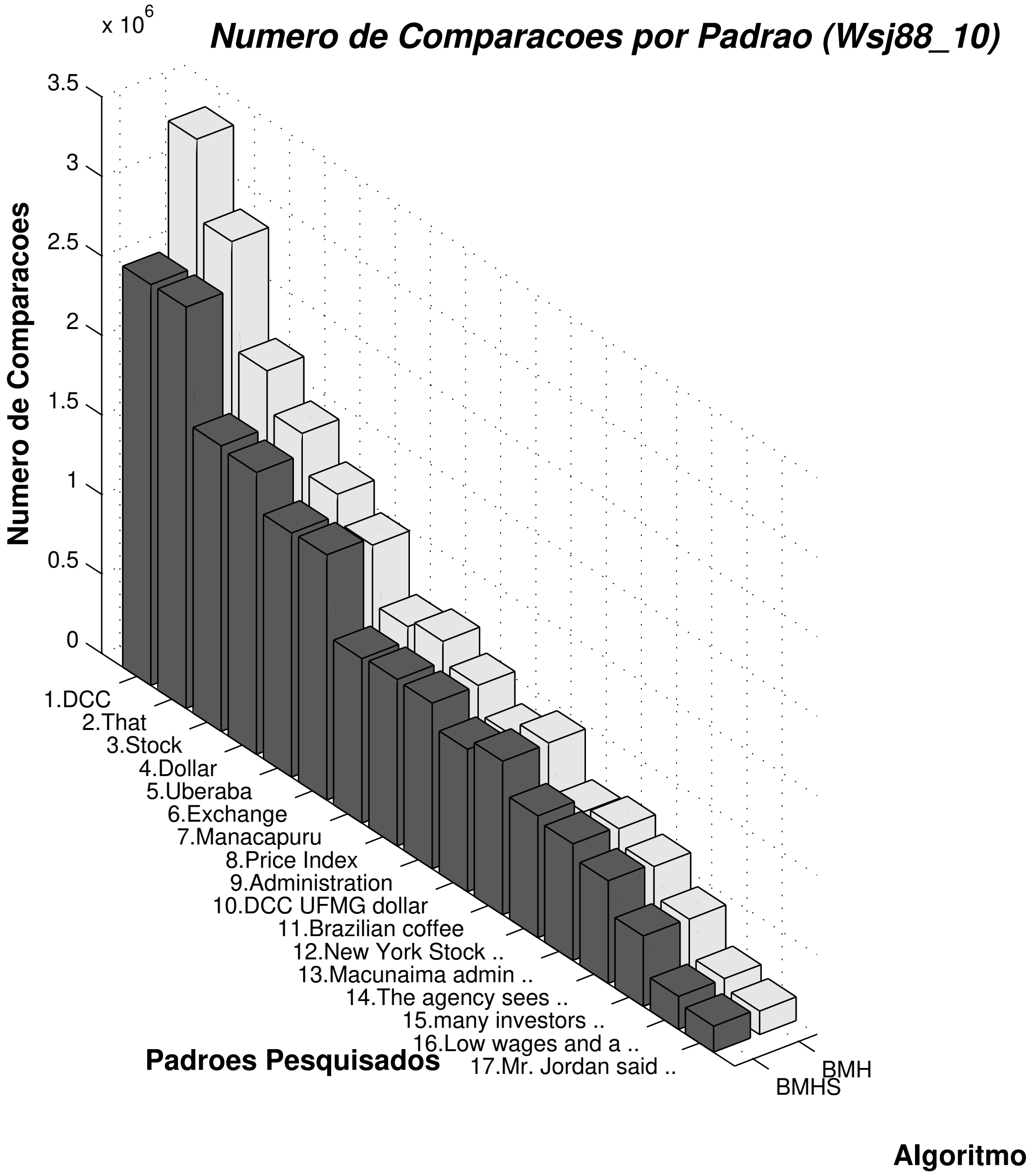



Figura 20: Gráfico de número de comparações por padrão para os algoritmos BMH e BMHS e para a coleção `Wsj88_10`.

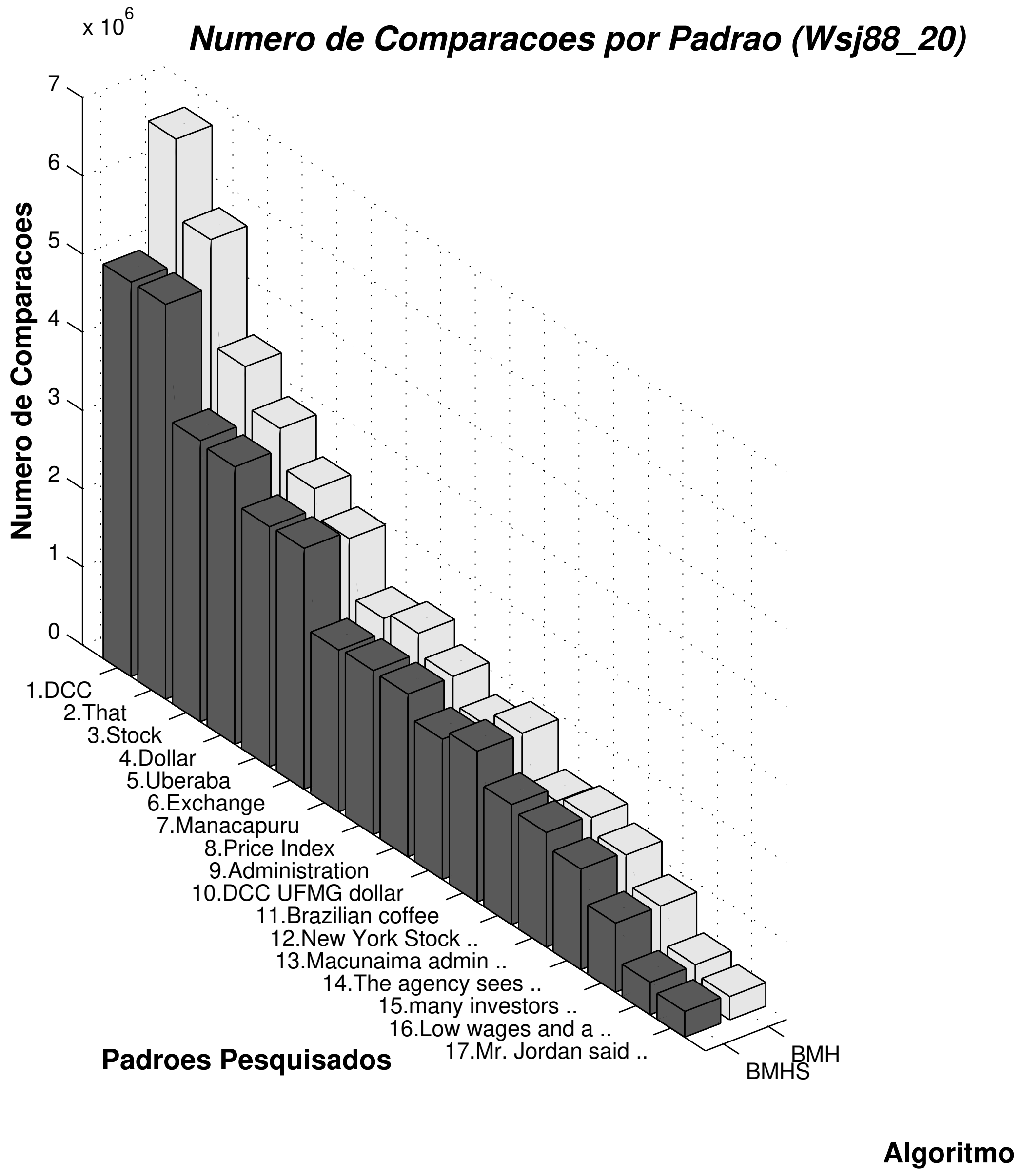



Figura 21: Gráfico de número de comparações por padrão para os algoritmos BMH e BMHS e para a coleção `Wsj88_20`.

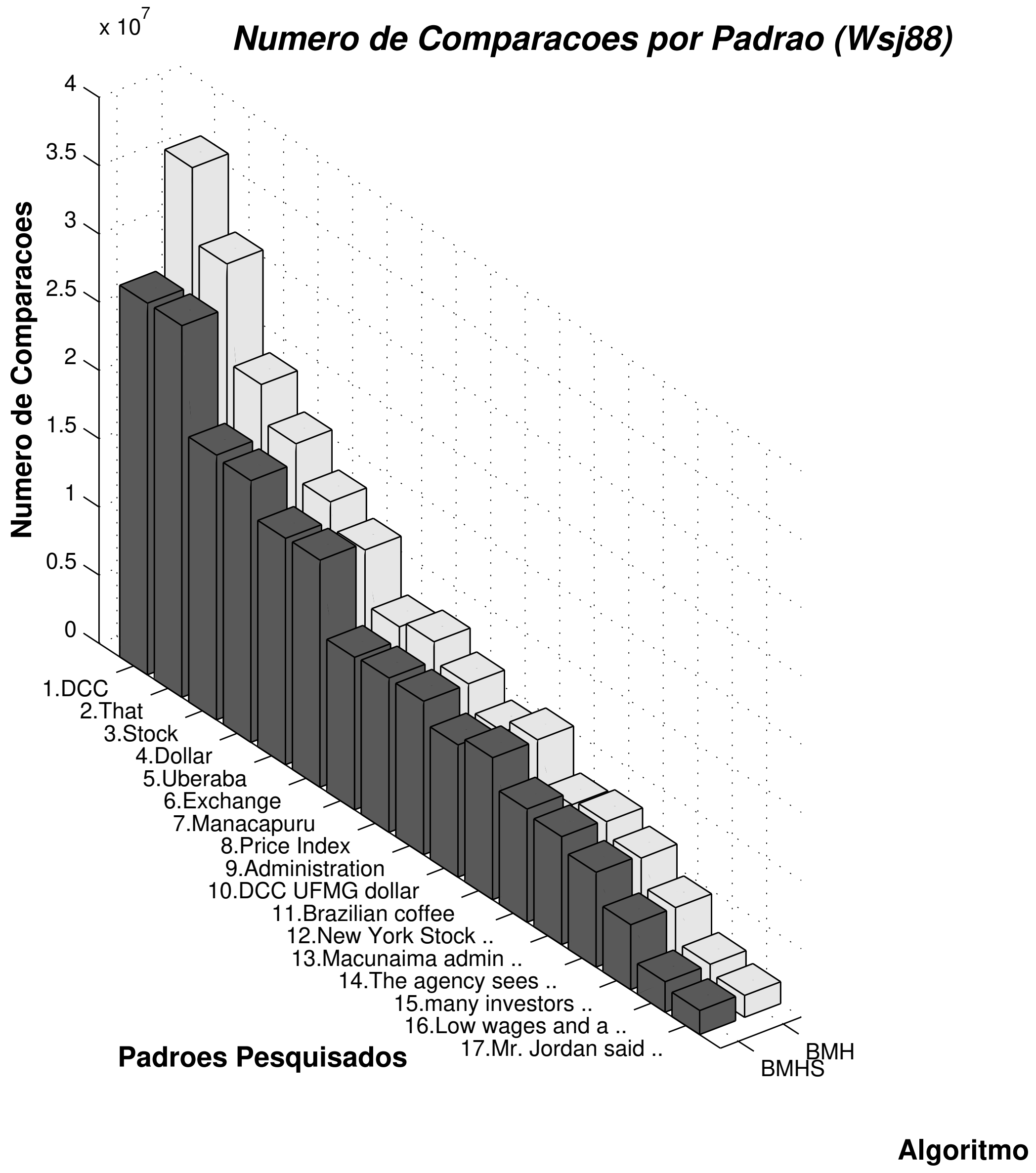
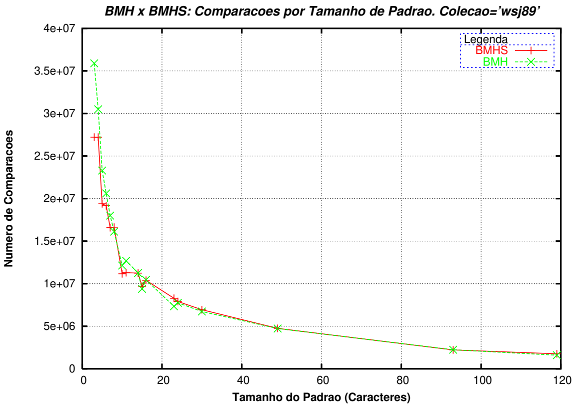
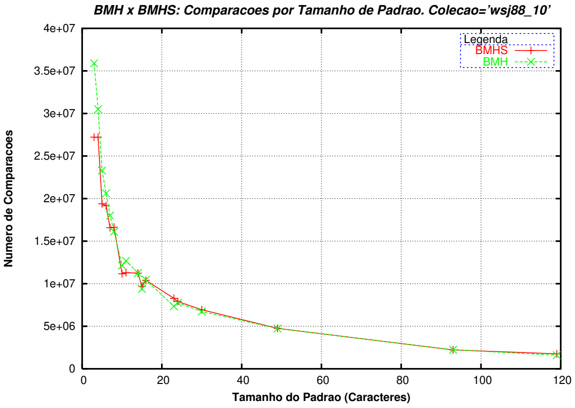


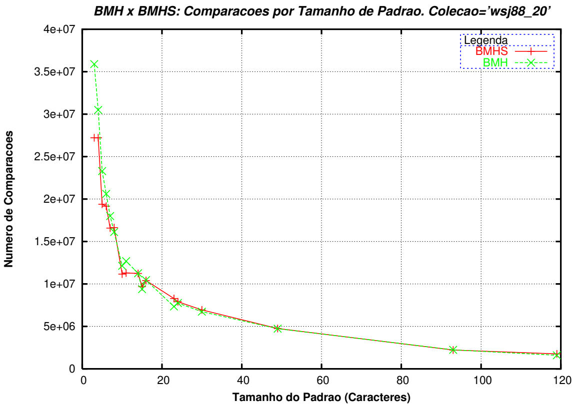

Figura 22: Gráfico de número de comparações por padrão para os algoritmos BMH e BMHS e para a coleção `Wsj88`.



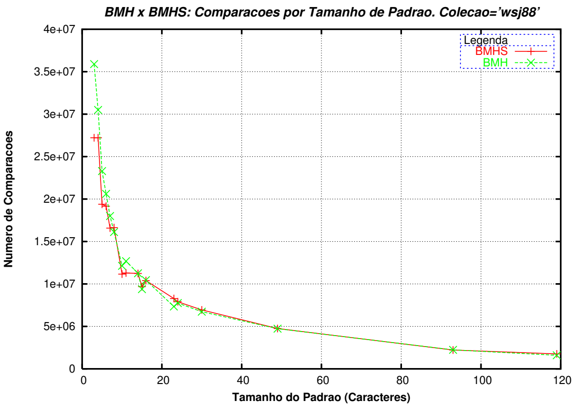

(a)




(b)




(c)




(d)

Figura 23: Número de comparações pelo tamanho do padrão para os algoritmos BMH e BMHS e para cada coleção.

Um segundo ponto a ser observado é que com o aumento do tamanho do padrão, há uma diminuição no número de comparações realizadas tanto pelo BMH como pelo BMHS.

Os dados gerados pelos experimentos são mostrados nas tabelas do Anexo A. Desta forma, a Tabela 25 corresponde aos dados de número de comparações pelo tamanho do padrão para os algoritmos BMH e BMHS e para as Coleções `Wsj89`, `Wsj88_10`, `Wsj88_20` e `Wsj88`.

No *segundo experimento* realizado, procurou-se mostrar como o número de comparações dos dois algoritmos se comporta ao variar o tamanho do texto. Desta forma, cada um padrões foi executado com tamanhos de textos diferentes para os dois algoritmos em estudo e os gráficos das Figuras 24, 25 e 26 foram gerados.

O principal ponto a ser observado é bem claro, que é o aumento do número de comparações com o aumento do texto. Estes gráficos confirmam as observações realizadas no que diz respeito ao tamanho do padrão.

A Tabela 26 mostra os dados ilustrados nestes gráficos e que corresponde aos dados de número de comparações dos algoritmos pelo tamanho do texto para cada padrão.

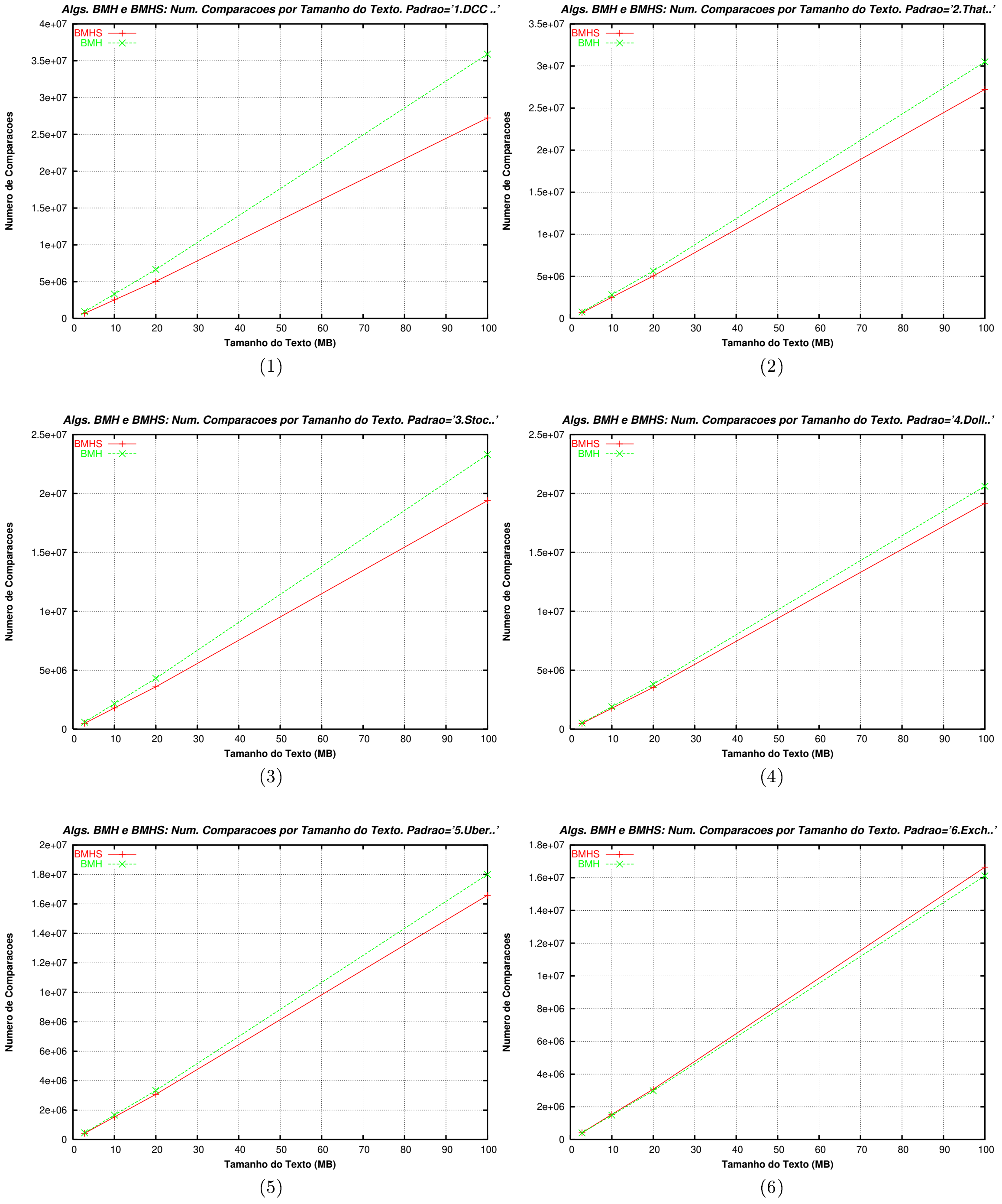
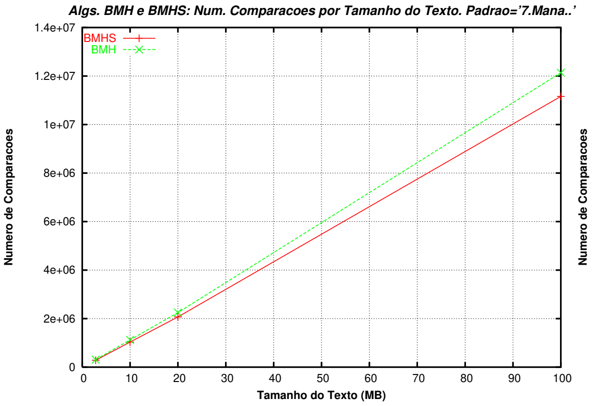
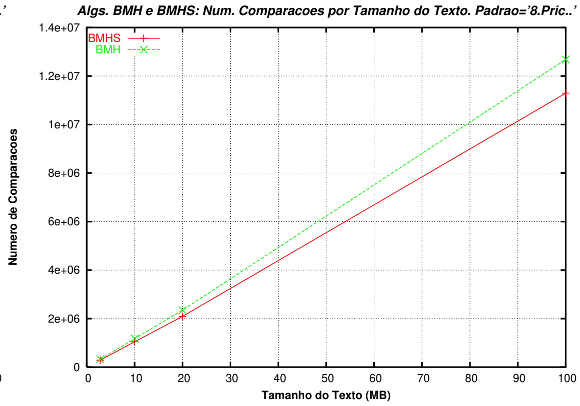


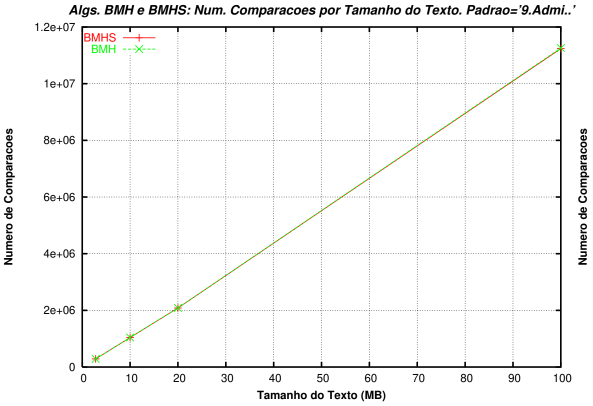

Figura 24: Número de comparações dos algoritmos pelo tamanho do texto para cada padrão (Parte 1).



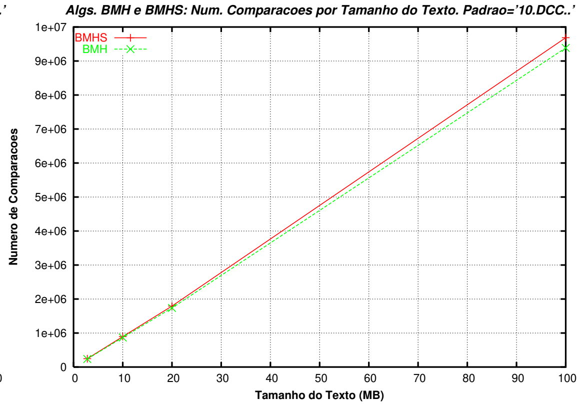

(7)



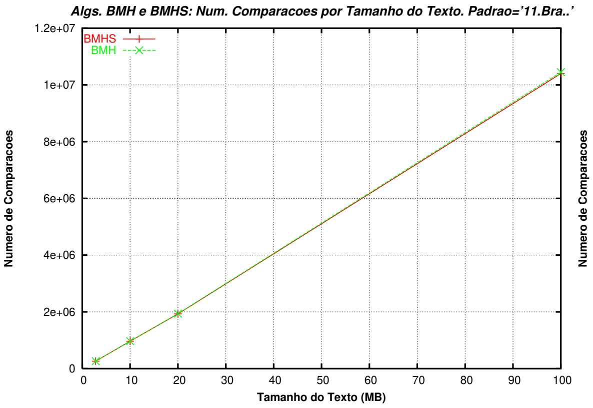

(8)



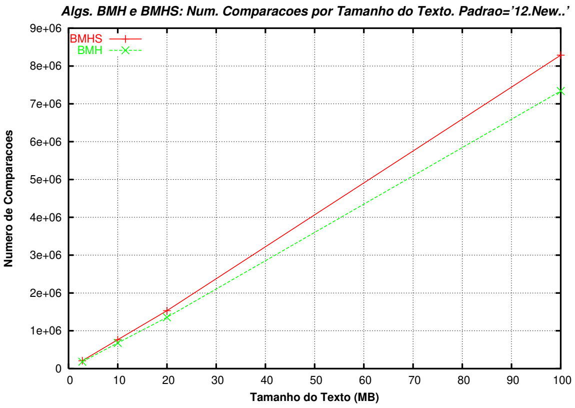

(9)




(10)

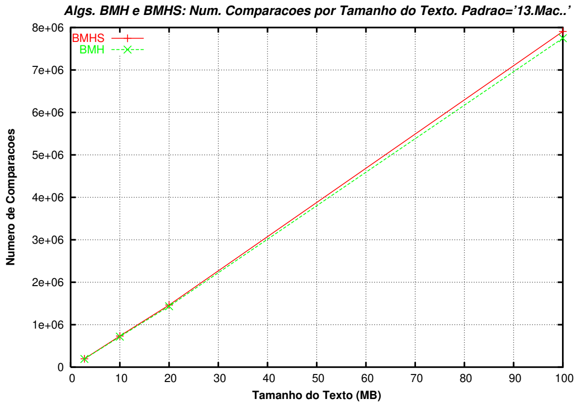



(11)

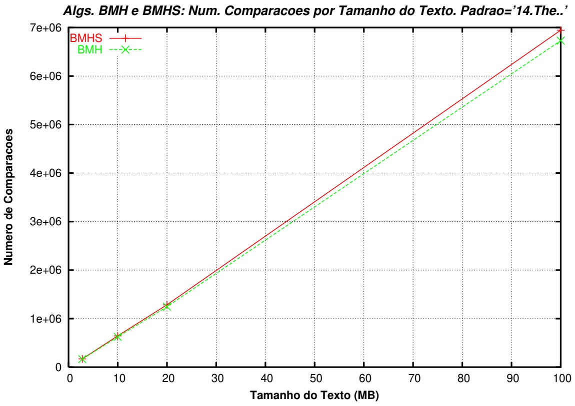



(12)

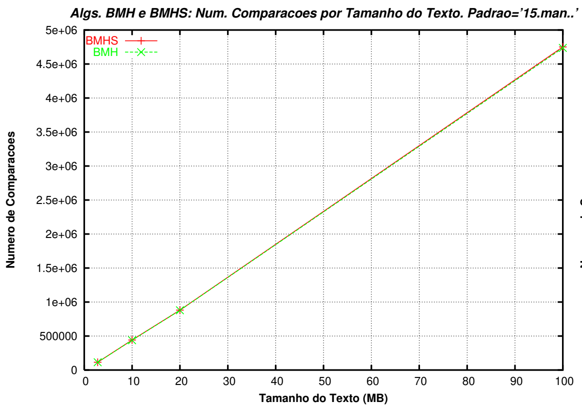
Figura 25: Número de comparações dos algoritmos pelo tamanho do texto para cada padrão (Parte 2).



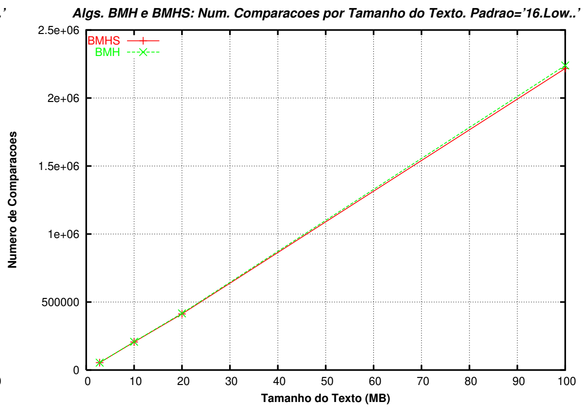

(13)



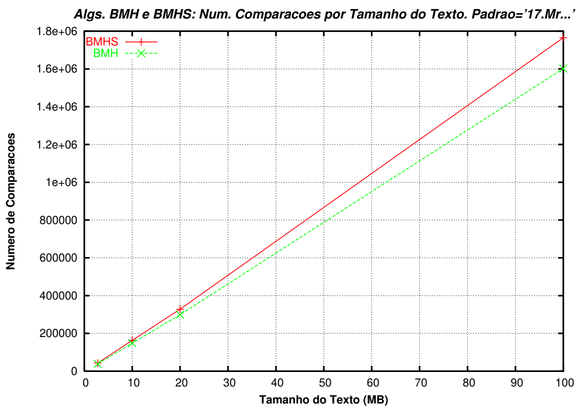

(14)




(15)




(16)




(17)

Figura 26: Número de comparações dos algoritmos pelo tamanho do texto para cada padrão (Parte 3).

# 6 Conclusão

Este trabalho focalizou cinco importantes algoritmos para a busca de padrões em texto. Sendo que três deles, o Boyer-Moore-Horspool, o Boyer-Moore-Horspool-Sunday, e o Shift-And são de busca exata e os outros dois, o Shift-And Aproximado e o Programação Dinâmica são de busca aproximada. Além disso, a implementação AGrep foi também estudada em conjunto para que fosse possível fazer uma comparação com uma implementação considerada rápida.

Para cada um dos cinco algoritmos, uma introdução foi feita ao método utilizado, um algoritmo foi mostrado e implementado e diversos experimentos foram realizados em cima desta implementação. Os resultados, muito interessantes na maioria dos casos, foram mostrados em tabelas e ilustrados em forma de gráficos.

# 7 Sobre o Documento

Este documento foi feito completamente utilizando o sistema LaTeX[4][5]. Os gráficos foram gerados utilizando o MatLab e o GNUPlot. As implementações, com o GCC e *Bash Shell Script*.

Uma versão on-line da mesma, incluindo os códigos-fonte em C e em Bash, todos os dados gerados (inclusive os dados que não puderam ser colocado neste documento), e os gráficos, pode ser encontrada em:

`http://www.dcc.ufmg.br/~horacio/PaaTP3`

# Referências


[1] CORMEN, Thomas H. LEISERSON, Charles E. RIVEST, Ronald L. STEIN, Clifford. *Algoritmos: Teoria e Prática*. Editora Campus, 2 edição, 2002.

[2] CROCHEMORE, Maxime. RYTTER, Wojciech *Text Algorithms*. Oxford, 1994.

[3] GONNET, G. BAEZA-YATES, R. *Handbook of Algorithms and Data Structures*. Addison-Wesley, 2 edição, 1991.

[4] GOOSSENS, Michel. MITTELBACH, Frank. SAMARIN, Alexander. *The LaTeX Companion*. Editora Addison-Wesley, 1993.

[5] LAMPORT, Leslie. *The LaTeX: A Document Preparation System*. Editora Addison-Wesley, 1986.

[6] NAVARRO, G. RAFFINOT, M. *Flexible Pattern Matching in Strings*. Cambridge University Press, 2002.

[7] SEDGEWICK, Robert. *Algorithms*. Addison-Wesley, 1983.

[8] ZIVIANI, Nivio. *Projeto de Algoritmos: Com implementações em Pascal e C*. Pioneira Thomson Learning, 2 edição, 2004.

[9] *CiteSeer: Scientific Literature Digital Library*. http://citeseer.nj.nec.com/cs, acessado em abril de 2004.

# A Tabelas dos Experimentos

## A.1 Algoritmos Exatos

### Comparação dos Algoritmos por Padrão

### Coleção Wsj89

| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0452 0.0033 | 0.0517 0.0047 | 0.0565 0.0034 | 0.0636 0.0006 | 0.0595 0.0086 | 0.0100 0.0000 |
| *2.That..* | 0.0506 0.0130 | 0.0495 0.0008 | 0.0612 0.0168 | 0.0669 0.0089 | 0.0596 0.0089 | 0.0100 0.0000 |
| *3.Stoc..* | 0.0466 0.0114 | 0.0484 0.0007 | 0.0592 0.0093 | 0.0673 0.0092 | 0.0574 0.0004 | 0.0130 0.0094 |
| *4.Doll..* | 0.0412 0.0020 | 0.0505 0.0101 | 0.0598 0.0090 | 0.0846 0.0393 | 0.0602 0.0061 | 0.0100 0.0000 |
| *5.Uber..* | 0.0423 0.0013 | 0.0473 0.0049 | 0.0637 0.0160 | 0.0645 0.0008 | 0.0564 0.0007 | 0.0090 0.0031 |
| *6.Exch..* | 0.0448 0.0041 | 0.0485 0.0008 | 0.0577 0.0009 | 0.0674 0.0043 | 0.0597 0.0039 | 0.0100 0.0000 |
| *7.Mana..* | 0.0426 0.0039 | 0.0471 0.0009 | 0.0586 0.0040 | 0.0667 0.0036 | 0.0576 0.0012 | 0.0010 0.0031 |
| *8.Pric..* | 0.0405 0.0021 | 0.0469 0.0011 | 0.0569 0.0007 | 0.0671 0.0043 | 0.0585 0.0035 | 0.0020 0.0042 |
| *9.Admi..* | 0.0437 0.0041 | 0.0477 0.0005 | 0.0598 0.0042 | 0.0679 0.0041 | 0.0594 0.0041 | 0.0060 0.0051 |
| *10.DCC..* | 0.0408 0.0026 | 0.0489 0.0042 | 0.0594 0.0042 | 0.0667 0.0007 | 0.0581 0.0005 | 0.0010 0.0031 |
| *11.Bra..* | 0.0424 0.0021 | 0.0472 0.0022 | 0.0583 0.0007 | 0.0684 0.0048 | 0.0592 0.0041 | 0.0010 0.0031 |
| *12.New..* | 0.0463 0.0042 | 0.0491 0.0005 | 0.0617 0.0049 | 0.0699 0.0043 | 0.0611 0.0047 | 0.0000 0.0000 |
| *13.Mac..* | 0.0433 0.0022 | 0.0501 0.0044 | 0.0612 0.0047 | 0.0724 0.0100 | 0.0585 0.0005 | 0.0000 0.0000 |
| *14.The..* | 0.0443 0.0023 | 0.0497 0.0059 | 0.0604 0.0018 | 0.0693 0.0007 | 0.0612 0.0046 | 0.0000 0.0000 |
| *15.man..* | 0.0496 0.0043 | 0.0539 0.0046 | 0.0647 0.0005 | 0.0747 0.0049 | 0.0630 0.0042 | 0.0000 0.0000 |
| *16.Low..* | 0.0584 0.0041 | 0.0614 0.0042 | 0.0737 0.0008 | 0.0823 0.0005 | 0.0663 0.0050 | 0.0000 0.0000 |
| *17.Mr...* | 0.0629 0.0046 | 0.0656 0.0041 | 0.0800 0.0039 | 0.0885 0.0044 | 0.0683 0.0038 | 0.0000 0.0000 |

Tabela 4: Dados de comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj89`.

### Coleção Wsj88_10

| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.1328 0.0038 | 0.1522 0.0045 | 0.1729 0.0039 | 0.2032 0.0038 | 0.1836 0.0037 | 0.0300 0.0000 |
| *2.That..* | 0.1389 0.0044 | 0.1547 0.0041 | 0.1756 0.0012 | 0.2072 0.0043 | 0.1880 0.0041 | 0.0400 0.0000 |
| *3.Stoc..* | 0.1331 0.0044 | 0.1503 0.0040 | 0.1791 0.0042 | 0.2093 0.0045 | 0.1895 0.0048 | 0.0300 0.0000 |
| *4.Doll..* | 0.1311 0.0048 | 0.1464 0.0043 | 0.1762 0.0042 | 0.2062 0.0041 | 0.1855 0.0044 | 0.0300 0.0000 |
| *5.Uber..* | 0.1287 0.0041 | 0.1442 0.0041 | 0.1758 0.0042 | 0.2062 0.0039 | 0.1841 0.0038 | 0.0310 0.0031 |
| *6.Exch..* | 0.1358 0.0047 | 0.1503 0.0041 | 0.1828 0.0035 | 0.2125 0.0039 | 0.1926 0.0040 | 0.0320 0.0063 |
| *7.Mana..* | 0.1272 0.0042 | 0.1433 0.0038 | 0.1796 0.0040 | 0.2093 0.0038 | 0.1872 0.0041 | 0.0200 0.0000 |
| *8.Pric..* | 0.1264 0.0039 | 0.1444 0.0041 | 0.1798 0.0044 | 0.2108 0.0046 | 0.1869 0.0038 | 0.0220 0.0063 |
| *9.Admi..* | 0.1319 0.0055 | 0.1479 0.0043 | 0.1833 0.0047 | 0.2136 0.0048 | 0.1906 0.0047 | 0.0240 0.0051 |
| *10.DCC..* | 0.1296 0.0051 | 0.1450 0.0047 | 0.1826 0.0046 | 0.2139 0.0047 | 0.1887 0.0045 | 0.0200 0.0000 |
| *11.Bra..* | 0.1317 0.0046 | 0.1475 0.0039 | 0.1841 0.0041 | 0.2142 0.0041 | 0.1895 0.0038 | 0.0200 0.0000 |
| *12.New..* | 0.1367 0.0043 | 0.1515 0.0042 | 0.1911 0.0042 | 0.2208 0.0039 | 0.1957 0.0040 | 0.0200 0.0000 |
| *13.Mac..* | 0.1348 0.0044 | 0.1496 0.0043 | 0.1890 0.0045 | 0.2189 0.0041 | 0.1919 0.0042 | 0.0200 0.0000 |
| *14.The..* | 0.1382 0.0046 | 0.1524 0.0045 | 0.1931 0.0042 | 0.2232 0.0044 | 0.1953 0.0042 | 0.0200 0.0000 |

*Continua na próxima página ..*

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
| *15.man..* | 0.1482 0.0039 | 0.1618 0.0046 | 0.2053 0.0039 | 0.2355 0.0045 | 0.2025 0.0038 | 0.0210 0.0031 |
| *16.Low..* | 0.1754 0.0048 | 0.1862 0.0039 | 0.2343 0.0036 | 0.2647 0.0038 | 0.2123 0.0038 | 0.0000 0.0000 |
| *17.Mr...* | 0.1905 0.0039 | 0.1994 0.0038 | 0.2502 0.0038 | 0.2802 0.0041 | 0.2193 0.0039 | 0.0000 0.0000 |

Tabela 5: Dados de comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj88_10`.

## Coleção Wsj88_20

| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.2578 0.0049 | 0.2956 0.0048 | 0.3367 0.0037 | 0.3976 0.0045 | 0.3601 0.0048 | 0.0710 0.0031 |
| *2.That..* | 0.2698 0.0057 | 0.3004 0.0048 | 0.3454 0.0046 | 0.4066 0.0040 | 0.3682 0.0050 | 0.0820 0.0063 |
| *3.Stoc..* | 0.2579 0.0041 | 0.2928 0.0039 | 0.3500 0.0043 | 0.4091 0.0011 | 0.3716 0.0035 | 0.0710 0.0031 |
| *4.Doll..* | 0.2536 0.0053 | 0.2832 0.0009 | 0.3432 0.0041 | 0.4041 0.0039 | 0.3618 0.0009 | 0.0600 0.0000 |
| *5.Uber..* | 0.2488 0.0041 | 0.2797 0.0035 | 0.3425 0.0035 | 0.4466 0.0984 | 0.3607 0.0037 | 0.0600 0.0000 |
| *6.Exch..* | 0.2627 0.0041 | 0.2916 0.0041 | 0.3568 0.0047 | 0.4166 0.0042 | 0.3773 0.0044 | 0.0700 0.0000 |
| *7.Mana..* | 0.2452 0.0051 | 0.2781 0.0043 | 0.3503 0.0043 | 0.4096 0.0046 | 0.3672 0.0045 | 0.0500 0.0000 |
| *8.Pric..* | 0.2453 0.0045 | 0.2797 0.0038 | 0.3540 0.0074 | 0.4316 0.0507 | 0.4508 0.0906 | 0.0770 0.0200 |
| *9.Admi..* | 0.3354 0.0909 | 0.3900 0.0499 | 0.3621 0.0122 | 0.4202 0.0111 | 0.3743 0.0107 | 0.0540 0.0126 |
| *10.DCC..* | 0.2516 0.0115 | 0.2833 0.0104 | 0.3584 0.0105 | 0.4213 0.0105 | 0.3707 0.0106 | 0.0530 0.0094 |
| *11.Bra..* | 0.2563 0.0111 | 0.2876 0.0108 | 0.3616 0.0106 | 0.4209 0.0110 | 0.3729 0.0105 | 0.0530 0.0094 |
| *12.New..* | 0.2660 0.0120 | 0.2960 0.0119 | 0.3742 0.0117 | 0.4355 0.0116 | 0.3858 0.0117 | 0.0450 0.0052 |
| *13.Mac..* | 0.2623 0.0115 | 0.2918 0.0113 | 0.3711 0.0114 | 0.4302 0.0112 | 0.3771 0.0112 | 0.0440 0.0126 |
| *14.The..* | 0.2685 0.0118 | 0.3007 0.0134 | 0.3788 0.0108 | 0.4382 0.0108 | 0.3844 0.0108 | 0.0430 0.0094 |
| *15.man..* | 0.2891 0.0116 | 0.3157 0.0111 | 0.4025 0.0107 | 0.4636 0.0101 | 0.3982 0.0099 | 0.0400 0.0000 |
| *16.Low..* | 0.3433 0.0107 | 0.3648 0.0103 | 0.4604 0.0110 | 0.5217 0.0109 | 0.4179 0.0115 | 0.0000 0.0000 |
| *17.Mr...* | 0.3732 0.0116 | 0.3904 0.0111 | 0.4921 0.0109 | 0.5523 0.0115 | 0.4321 0.0116 | 0.0000 0.0000 |

Tabela 6: Dados de comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj88_20`.

## Coleção Wsj88

| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 1.3659 0.0085 | 1.5712 0.0085 | 1.7917 0.0076 | 2.1184 0.0091 | 1.9055 0.0084 | 0.3970 0.0125 |
| *2.That..* | 1.4280 0.0098 | 1.6383 0.0947 | 1.8594 0.0699 | 2.1912 0.0459 | 1.9620 0.0044 | 0.4580 0.0063 |
| *3.Stoc..* | 1.3715 0.0048 | 1.5635 0.0042 | 1.8724 0.0063 | 2.2201 0.0573 | 2.0080 0.0154 | 0.3930 0.0067 |
| *4.Doll..* | 1.3630 0.0291 | 1.5181 0.0072 | 1.8342 0.0045 | 2.3120 0.2620 | 1.9481 0.0098 | 0.3400 0.0047 |
| *5.Uber..* | 1.3361 0.0114 | 1.4984 0.0089 | 1.8366 0.0105 | 2.1733 0.0112 | 1.9652 0.0514 | 0.3240 0.0069 |
| *6.Exch..* | 1.5005 0.1040 | 1.5982 0.0292 | 1.9289 0.0102 | 2.2653 0.0215 | 2.1028 0.0953 | 0.4350 0.0519 |
| *7.Mana..* | 1.3343 0.0183 | 1.4838 0.0090 | 1.9246 0.0596 | 2.2133 0.0119 | 1.9778 0.0118 | 0.2750 0.0052 |
| *8.Pric..* | 1.3188 0.0133 | 1.5050 0.0553 | 1.8632 0.0082 | 2.2007 0.0125 | 1.9477 0.0271 | 0.2700 0.0000 |
| *9.Admi..* | 1.3974 0.0782 | 1.5363 0.0489 | 1.9838 0.0613 | 2.3227 0.0705 | 2.0533 0.0551 | 0.2990 0.0172 |
| *10.DCC..* | 1.3678 0.0508 | 1.5793 0.0683 | 2.0001 0.0950 | 2.3906 0.0524 | 1.9906 0.0468 | 0.2650 0.0126 |
| | | | | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Padrão* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
| *11.Bra..* | 1.3506 0.0092 | 1.5779 0.0966 | 1.9871 0.0415 | 2.3464 0.1113 | 1.9780 0.0036 | 0.2750 0.0070 |
| *12.New..* | 1.4137 0.0047 | 1.5719 0.0036 | 1.9987 0.0036 | 2.3435 0.0650 | 2.0658 0.0621 | 0.2610 0.0031 |
| *13.Mac..* | 1.3837 0.0036 | 1.5496 0.0037 | 1.9740 0.0035 | 2.3644 0.0744 | 2.0093 0.0118 | 0.2310 0.0031 |
| *14.The..* | 1.4338 0.0240 | 1.5981 0.0239 | 2.0356 0.0117 | 2.3410 0.0050 | 2.0754 0.0232 | 0.2220 0.0042 |
| *15.man..* | 1.5526 0.0146 | 1.6883 0.0111 | 2.1614 0.0288 | 2.5802 0.0745 | 2.1383 0.0110 | 0.2190 0.0056 |
| *16.Low..* | 1.8541 0.0532 | 2.0057 0.0681 | 2.5679 0.0683 | 2.8596 0.0656 | 2.2491 0.0145 | 0.0000 0.0000 |
| *17.Mr...* | 2.0673 0.0979 | 2.1850 0.0956 | 2.6209 0.0088 | 2.9464 0.0083 | 2.2839 0.0084 | 0.0000 0.0000 |

Tabela 7: Dados de comparação do tempo de execução de cada padrão em cada um dos algoritmos para a coleção `Wsj88`.

## Comparação dos Algoritmos por Tamanho de Texto

| *Tam(M)* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| → **Padrão 1: "DCC"** | | | | | | |
| *2.8* | 0.0470 0.0102 | 0.0541 0.0139 | 0.0597 0.0055 | 0.0683 0.0061 | 0.0588 0.0040 | 0.0100 0.0000 |
| *10.0* | 0.1340 0.0049 | 0.1565 0.0110 | 0.1738 0.0082 | 0.2041 0.0081 | 0.1850 0.0099 | 0.0330 0.0094 |
| *20.0* | 0.2589 0.0085 | 0.2964 0.0079 | 0.3375 0.0083 | 0.3987 0.0078 | 0.3605 0.0076 | 0.0730 0.0094 |
| *100.0* | 1.3650 0.0067 | 1.5699 0.0065 | 1.7917 0.0064 | 2.1174 0.0064 | 1.9052 0.0072 | 0.3960 0.0096 |
| → **Padrão 2: "That"** | | | | | | |
| *2.8* | 0.0472 0.0092 | 0.0491 0.0009 | 0.0588 0.0096 | 0.0670 0.0092 | 0.0599 0.0093 | 0.0100 0.0000 |
| *10.0* | 0.1396 0.0095 | 0.1561 0.0096 | 0.1781 0.0100 | 0.2082 0.0097 | 0.1892 0.0097 | 0.0430 0.0094 |
| *20.0* | 0.2716 0.0111 | 0.3020 0.0102 | 0.3466 0.0103 | 0.4075 0.0096 | 0.3699 0.0109 | 0.0830 0.0094 |
| *100.0* | 1.4284 0.0108 | 1.5960 0.0069 | 1.8356 0.0096 | 2.1607 0.0077 | 1.9524 0.0102 | 0.4570 0.0125 |
| → **Padrão 3: "Stock"** | | | | | | |
| *2.8* | 0.0431 0.0006 | 0.0518 0.0117 | 0.0563 0.0009 | 0.0685 0.0119 | 0.0609 0.0109 | 0.0100 0.0000 |
| *10.0* | 0.1350 0.0111 | 0.1515 0.0112 | 0.1810 0.0108 | 0.2110 0.0112 | 0.1910 0.0107 | 0.0340 0.0126 |
| *20.0* | 0.2598 0.0118 | 0.2943 0.0112 | 0.3505 0.0121 | 0.4116 0.0117 | 0.3725 0.0115 | 0.0730 0.0094 |
| *100.0* | 1.3678 0.0093 | 1.5840 0.0908 | 1.8534 0.0056 | 2.1773 0.0048 | 1.9599 0.0034 | 0.3840 0.0069 |
| → **Padrão 4: "Dollar"** | | | | | | |
| *2.8* | 0.0434 0.0049 | 0.0477 0.0055 | 0.0552 0.0003 | 0.0653 0.0054 | 0.0578 0.0048 | 0.0240 0.0442 |
| *10.0* | 0.1331 0.0122 | 0.1477 0.0109 | 0.1777 0.0116 | 0.2085 0.0113 | 0.1897 0.0155 | 0.0330 0.0094 |
| *20.0* | 0.2545 0.0125 | 0.2860 0.0115 | 0.3448 0.0105 | 0.4057 0.0113 | 0.3641 0.0107 | 0.0630 0.0094 |
| *100.0* | 1.3420 0.0100 | 1.5113 0.0100 | 1.8245 0.0088 | 2.1509 0.0095 | 1.9221 0.0087 | 0.3360 0.0126 |
| → **Padrão 5: "Uberaba"** | | | | | | |
| *2.8* | 0.0444 0.0119 | 0.0500 0.0113 | 0.0552 0.0005 | 0.0677 0.0116 | 0.0596 0.0106 | 0.0100 0.0000 |
| *10.0* | 0.1299 0.0112 | 0.1461 0.0110 | 0.1767 0.0112 | 0.2078 0.0112 | 0.1861 0.0106 | 0.0300 0.0000 |
| *20.0* | 0.2499 0.0116 | 0.2811 0.0107 | 0.3437 0.0106 | 0.4053 0.0107 | 0.3616 0.0111 | 0.0630 0.0094 |
| *100.0* | 1.3172 0.0088 | 1.4864 0.0085 | 1.8228 0.0087 | 2.1504 0.0088 | 1.9110 0.0082 | 0.3170 0.0125 |
| → **Padrão 6: "Exchange"** | | | | | | |
| *2.8* | 0.0434 0.0006 | 0.0476 0.0009 | 0.0611 0.0115 | 0.0689 0.0109 | 0.0617 0.0107 | 0.0100 0.0000 |
| | | | | | | *Continua na próxima página ..* |

*Continuação da página anterior ..*

| *Tam(M)* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
|---|---|---|---|---|---|---|
| *10.0* | 0.1370 0.0110 | 0.1514 0.0106 | 0.1847 0.0108 | 0.2141 0.0110 | 0.1934 0.0111 | 0.0300 0.0000 |
| *20.0* | 0.2647 0.0118 | 0.2929 0.0109 | 0.3576 0.0112 | 0.4189 0.0107 | 0.3775 0.0108 | 0.0730 0.0094 |
| *100.0* | 1.3970 0.0094 | 1.5537 0.0081 | 1.9276 0.0913 | 2.2143 0.0039 | 1.9951 0.0051 | 0.3900 0.0047 |
| → **Padrão 7: "Manacapuru"** | | | | | | |
| *2.8* | 0.0424 0.0040 | 0.0467 0.0007 | 0.0582 0.0049 | 0.0665 0.0043 | 0.0571 0.0003 | 0.0030 0.0048 |
| *10.0* | 0.1261 0.0047 | 0.1420 0.0045 | 0.1783 0.0039 | 0.2079 0.0040 | 0.1862 0.0040 | 0.0200 0.0000 |
| *20.0* | 0.2436 0.0045 | 0.2759 0.0045 | 0.3480 0.0045 | 0.4068 0.0047 | 0.3647 0.0048 | 0.0510 0.0031 |
| *100.0* | 1.2896 0.0037 | 1.4675 0.0032 | 1.8561 0.0051 | 2.1738 0.0048 | 1.9376 0.0045 | 0.2710 0.0031 |
| → **Padrão 8: "Price Index"** | | | | | | |
| *2.8* | 0.0414 0.0054 | 0.0475 0.0041 | 0.0567 0.0006 | 0.0668 0.0046 | 0.0568 0.0006 | 0.0000 0.0000 |
| *10.0* | 0.1257 0.0043 | 0.1434 0.0041 | 0.1786 0.0048 | 0.2095 0.0047 | 0.1860 0.0048 | 0.0200 0.0000 |
| *20.0* | 0.2430 0.0054 | 0.2774 0.0046 | 0.3477 0.0046 | 0.4102 0.0045 | 0.3628 0.0050 | 0.0510 0.0031 |
| *100.0* | 1.2874 0.0048 | 1.4756 0.0042 | 1.8565 0.0039 | 2.1907 0.0034 | 1.9296 0.0035 | 0.2710 0.0031 |
| → **Padrão 9: "Administration"** | | | | | | |
| *2.8* | 0.0438 0.0038 | 0.0474 0.0011 | 0.0593 0.0048 | 0.0674 0.0040 | 0.0577 0.0004 | 0.0100 0.0047 |
| *10.0* | 0.1311 0.0049 | 0.1470 0.0041 | 0.1824 0.0043 | 0.2122 0.0045 | 0.1892 0.0042 | 0.0220 0.0042 |
| *20.0* | 0.2536 0.0046 | 0.2851 0.0037 | 0.3563 0.0044 | 0.4159 0.0038 | 0.3711 0.0038 | 0.0500 0.0000 |
| *100.0* | 1.3677 0.0526 | 1.5184 0.0047 | 1.9019 0.0048 | 2.2233 0.0036 | 1.9726 0.0037 | 0.2910 0.0031 |
| → **Padrão 10: "DCC UFMG dollar"** | | | | | | |
| *2.8* | 0.0415 0.0020 | 0.0482 0.0043 | 0.0591 0.0046 | 0.0678 0.0039 | 0.0577 0.0004 | 0.0000 0.0000 |
| *10.0* | 0.1272 0.0005 | 0.1442 0.0040 | 0.1816 0.0037 | 0.2128 0.0036 | 0.1874 0.0037 | 0.0200 0.0000 |
| *20.0* | 0.2485 0.0036 | 0.2793 0.0036 | 0.3540 0.0042 | 0.4163 0.0037 | 0.3667 0.0041 | 0.0510 0.0031 |
| *100.0* | 1.3171 0.0038 | 1.4889 0.0046 | 1.8899 0.0046 | 2.2272 0.0033 | 1.9499 0.0040 | 0.2610 0.0031 |
| → **Padrão 11: "Brazilian coffee"** | | | | | | |
| *2.8* | 0.0413 0.0025 | 0.0470 0.0021 | 0.0595 0.0048 | 0.0676 0.0039 | 0.0576 0.0007 | 0.0010 0.0031 |
| *10.0* | 0.1313 0.0046 | 0.1451 0.0012 | 0.1830 0.0043 | 0.2129 0.0042 | 0.1872 0.0008 | 0.0200 0.0000 |
| *20.0* | 0.2534 0.0045 | 0.2841 0.0042 | 0.3570 0.0041 | 0.4173 0.0045 | 0.3689 0.0040 | 0.0500 0.0000 |
| *100.0* | 1.3421 0.0040 | 1.5101 0.0037 | 1.9042 0.0039 | 2.2276 0.0042 | 1.9600 0.0039 | 0.2720 0.0063 |
| → **Padrão 12: "New York Stock .."** | | | | | | |
| *2.8* | 0.0455 0.0046 | 0.0501 0.0041 | 0.0600 0.0007 | 0.0698 0.0046 | 0.0592 0.0009 | 0.0000 0.0000 |
| *10.0* | 0.1359 0.0041 | 0.1505 0.0044 | 0.1897 0.0040 | 0.2195 0.0045 | 0.1945 0.0046 | 0.0200 0.0000 |
| *20.0* | 0.2624 0.0016 | 0.2926 0.0045 | 0.3705 0.0050 | 0.4301 0.0047 | 0.3816 0.0043 | 0.0460 0.0051 |
| *100.0* | 1.4011 0.0036 | 1.5578 0.0039 | 1.9813 0.0036 | 2.3013 0.0041 | 2.0309 0.0039 | 0.2610 0.0031 |
| → **Padrão 13: "Macunaima admin .."** | | | | | | |
| *2.8* | 0.0411 0.0029 | 0.0494 0.0046 | 0.0595 0.0008 | 0.0677 0.0005 | 0.0598 0.0045 | 0.0000 0.0000 |
| *10.0* | 0.1339 0.0047 | 0.1484 0.0046 | 0.1879 0.0046 | 0.2178 0.0046 | 0.1903 0.0044 | 0.0200 0.0000 |
| *20.0* | 0.2583 0.0050 | 0.2884 0.0041 | 0.3665 0.0037 | 0.4257 0.0025 | 0.3725 0.0034 | 0.0400 0.0000 |
| *100.0* | 1.3713 0.0031 | 1.5355 0.0031 | 1.9576 0.0038 | 2.2798 0.0120 | 1.9841 0.0022 | 0.2310 0.0031 |
| → **Padrão 14: "The agency sees .."** | | | | | | |
| *2.8* | 0.0437 0.0019 | 0.0495 0.0039 | 0.0615 0.0028 | 0.0688 0.0006 | 0.0602 0.0031 | 0.0000 0.0000 |
| *10.0* | 0.1366 0.0028 | 0.1513 0.0026 | 0.1913 0.0028 | 0.2210 0.0025 | 0.1967 0.0100 | 0.0200 0.0000 |
| *20.0* | 0.2649 0.0028 | 0.2936 0.0023 | 0.3741 0.0028 | 0.4336 0.0025 | 0.3796 0.0029 | 0.0410 0.0031 |

*Continua na próxima página ..*

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Tam(M)* | **BMHS** | **BMH** | **SA** | **SAA** | **PD** | **AGrep** |
| *100.0* | 1.4103 0.0032 | 1.5667 0.0029 | 2.0001 0.0023 | 2.3206 0.0027 | 2.0211 0.0029 | 0.2210 0.0031 |
| → **Padrão 15: "many investors .."** | | | | | | |
| *2.8* | 0.0487 0.0033 | 0.0516 0.0021 | 0.0654 0.0037 | 0.0726 0.0004 | 0.0624 0.0031 | 0.0000 0.0000 |
| *10.0* | 0.1465 0.0033 | 0.1608 0.0034 | 0.2037 0.0033 | 0.2334 0.0028 | 0.2011 0.0033 | 0.0200 0.0000 |
| *20.0* | 0.2849 0.0031 | 0.3117 0.0028 | 0.3981 0.0033 | 0.4582 0.0036 | 0.3929 0.0020 | 0.0400 0.0000 |
| *100.0* | 1.5186 0.0031 | 1.6657 0.0025 | 2.1299 0.0025 | 2.4527 0.0029 | 2.0971 0.0027 | 0.2130 0.0048 |
| → **Padrão 16: "Low wages and a .."** | | | | | | |
| *2.8* | 0.0575 0.0031 | 0.0607 0.0027 | 0.0742 0.0033 | 0.0816 0.0005 | 0.0644 0.0006 | 0.0000 0.0000 |
| *10.0* | 0.1737 0.0040 | 0.1845 0.0034 | 0.2327 0.0034 | 0.2625 0.0034 | 0.2104 0.0034 | 0.0000 0.0000 |
| *20.0* | 0.3385 0.0035 | 0.3598 0.0035 | 0.4558 0.0024 | 0.5158 0.0031 | 0.4279 0.0473 | 0.0000 0.0000 |
| *100.0* | 1.8192 0.0070 | 1.9354 0.0073 | 2.4531 0.0070 | 2.7789 0.0065 | 2.2110 0.0070 | 0.0000 0.0000 |
| → **Padrão 17: "Mr. Jordan said .."** | | | | | | |
| *2.8* | 0.0642 0.0096 | 0.0668 0.0093 | 0.0814 0.0091 | 0.0868 0.0008 | 0.0669 0.0006 | 0.0000 0.0000 |
| *10.0* | 0.1919 0.0095 | 0.2003 0.0094 | 0.2508 0.0094 | 0.2819 0.0095 | 0.2196 0.0097 | 0.0000 0.0000 |
| *20.0* | 0.3699 0.0037 | 0.3895 0.0090 | 0.4910 0.0090 | 0.5521 0.0085 | 0.4311 0.0096 | 0.0000 0.0000 |
| *100.0* | 1.9832 0.0079 | 2.0820 0.0100 | 2.6238 0.0065 | 2.9497 0.0077 | 2.2871 0.0075 | 0.0000 0.0000 |

Tabela 8: Dados de tempo de execução e desvio padrão dos algoritmos com a variação no tamanho do texto para cada um dos padrões estudados.

## A.2 Algoritmos Aproximados

### Comparação dos Algoritmos por Padrão

### Algoritmo Shift-And Aproximado (Coleção Wsj88)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 2.1086 0.0027 | 2.8162 0.0027 | 9.0090 0.0184 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 2.1510 0.0035 | 3.4934 0.0055 | 15.202 0.0260 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 2.1756 0.0034 | 2.8119 0.0031 | 3.9645 0.0039 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 2.1417 0.0029 | 2.6239 0.0025 | 3.4277 0.0107 | 5.4276 0.0066 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 2.1406 0.0033 | 2.5208 0.0033 | 3.0731 0.0024 | 3.6403 0.0025 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 2.2144 0.0032 | 2.7430 0.0019 | 3.5288 0.0059 | 4.2884 0.0041 | 5.4809 0.0065 | 0.0000 0.0000 |
| *7.Mana..* | 2.1719 0.0029 | 2.5572 0.0034 | 3.1096 0.0097 | 3.4936 0.0027 | 3.8887 0.0084 | 4.3764 0.0033 |
| *8.Pric..* | 2.1891 0.0027 | 2.5725 0.0032 | 3.1299 0.0039 | 3.5260 0.0028 | 3.9622 0.0030 | 4.6501 0.0025 |
| *9.Admi..* | 2.2215 0.0030 | 2.6522 0.0024 | 3.2771 0.0032 | 3.7523 0.0098 | 4.2210 0.0033 | 4.6964 0.0046 |
| *10.DCC..* | 2.2264 0.0025 | 2.6282 0.0504 | 3.3136 0.1895 | 4.2450 0.3623 | 4.0851 0.4015 | 4.3267 0.0065 |
| *11.Bra..* | 2.2360 0.0068 | 2.6197 0.0081 | 3.1703 0.0078 | 3.5589 0.0076 | 3.9531 0.0120 | 4.3480 0.0078 |
| *12.New..* | 2.3362 0.0913 | 2.7297 0.0035 | 3.3283 0.0038 | 3.7594 0.0047 | 4.1950 0.0033 | 4.6260 0.0040 |
| *13.Mac..* | 2.2784 0.0038 | 2.6623 0.0041 | 3.2101 0.0035 | 3.5975 0.0033 | 3.9870 0.0093 | 4.3693 0.0037 |
| *14.The..* | 2.3188 0.0030 | 2.7034 0.0043 | 3.2526 0.0036 | 3.6403 0.0031 | 4.0262 0.0036 | 4.4126 0.0033 |
| *15.man..* | 2.4528 0.0034 | 2.8369 0.0042 | 3.3940 0.0036 | 3.8117 0.0659 | 4.2151 0.0040 | 4.7459 0.0119 |
| *16.Low..* | 2.7694 0.0031 | 3.1516 0.0043 | 3.7012 0.0046 | 4.0881 0.0042 | 4.4745 0.0037 | 4.8609 0.0035 |
| | | | | | | *Continua na próxima página ..* |

*Continuação da página anterior ..*

| *Padrão* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *17.Mr...* | 2.9395 0.0035 | 3.3216 0.0035 | 3.8705 0.0031 | 4.2591 0.0034 | 4.6618 0.0093 | 5.0993 0.0037 |

Tabela 9: Dados de tempo de execução e desvio padrão do algoritmo Shift-And Aproximado ao variar o padrão e a aproximação para a coleção `Wsj88`.

## Algoritmo de Programação Dinâmica (Coleção Wsj88)

| *Padrão* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 1.8955 0.0032 | 2.6237 0.0042 | 8.7519 0.0465 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 2.1120 0.2166 | 3.5288 0.2921 | 15.257 0.0280 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 1.9707 0.0133 | 2.7979 0.0081 | 4.2255 0.0092 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 1.9215 0.0070 | 2.5855 0.0968 | 3.5592 0.0055 | 6.2364 0.0050 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 1.9007 0.0031 | 2.3355 0.0037 | 3.1764 0.0033 | 4.4344 0.0030 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 1.9922 0.0028 | 2.5461 0.0044 | 3.4041 0.0049 | 4.8043 0.0086 | 6.3379 0.0068 | 0.0000 0.0000 |
| *7.Mana..* | 1.9373 0.0040 | 2.5351 0.0035 | 3.3326 0.0050 | 4.3746 0.0030 | 5.0892 0.0065 | 5.7525 0.0110 |
| *8.Pric..* | 1.9301 0.0030 | 2.4704 0.0093 | 3.2831 0.0048 | 4.4933 0.0089 | 5.4504 0.0057 | 6.6660 0.0038 |
| *9.Admi..* | 1.9720 0.0029 | 2.4866 0.0033 | 3.2420 0.0029 | 4.4582 0.0032 | 5.2501 0.0055 | 6.0390 0.0087 |
| *10.DCC..* | 1.9474 0.0031 | 2.3648 0.0034 | 2.9060 0.0103 | 4.0497 0.0032 | 4.6805 0.0072 | 5.5481 0.0082 |
| *11.Bra..* | 1.9601 0.0030 | 2.4949 0.0033 | 3.2908 0.0041 | 4.3903 0.0030 | 5.1293 0.0049 | 5.7748 0.0096 |
| *12.New..* | 2.0294 0.0031 | 2.6902 0.0032 | 3.6014 0.0091 | 5.0459 0.0035 | 5.8894 0.0049 | 6.7151 0.0107 |
| *13.Mac..* | 1.9819 0.0029 | 2.5364 0.0031 | 3.2789 0.0022 | 4.4377 0.0043 | 5.0942 0.0069 | 5.7485 0.0102 |
| *14.The..* | 2.0189 0.0034 | 2.6178 0.0038 | 3.5409 0.0028 | 4.9136 0.0045 | 5.8130 0.0065 | 6.5476 0.0082 |
| *15.man..* | 2.0972 0.0030 | 2.7773 0.0090 | 3.6435 0.0039 | 4.8985 0.0038 | 5.8477 0.0103 | 6.6507 0.0661 |
| *16.Low..* | 2.1990 0.0025 | 2.7346 0.0040 | 3.4763 0.0036 | 4.8552 0.0455 | 5.5915 0.0125 | 6.4378 0.0071 |
| *17.Mr...* | 2.2853 0.0081 | 2.8078 0.0076 | 3.5326 0.0071 | 4.8586 0.0116 | 5.6888 0.0987 | 6.4535 0.0088 |

Tabela 10: Dados de tempo de execução e desvio padrão do algoritmo de Programação Dinâmica ao variar o padrão e a aproximação para a coleção `Wsj88`.

## A Implementação AGrep (Coleção Wsj88)

| *Padrão* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.3970 0.0094 | 0.9480 0.0091 | 2.7720 0.0063 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.4580 0.0122 | 1.8760 0.0096 | 5.4450 0.0070 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.3860 0.0096 | 0.9170 0.0094 | 2.3810 0.0073 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.3360 0.0096 | 0.5980 0.0122 | 1.1420 0.0063 | 3.9090 0.0956 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.3140 0.0051 | 0.4910 0.0031 | 0.8510 0.0031 | 2.9330 0.0067 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.3870 0.0048 | 0.5440 0.0096 | 1.2710 0.0031 | 1.5840 0.0051 | 3.9410 0.0031 | 0.0000 0.0000 |
| *7.Mana..* | 0.2710 0.0031 | 0.3500 0.0047 | 0.5070 0.0048 | 0.8610 0.0031 | 0.9890 0.0056 | 3.7400 0.0047 |
| *8.Pric..* | 0.2710 0.0031 | 0.4410 0.0031 | 0.6410 0.0031 | 1.3010 0.0031 | 1.7410 0.0031 | 4.1590 0.0031 |
| *9.Admi..* | 0.2910 0.0031 | 0.5010 0.0031 | 0.7640 0.0051 | 1.0010 0.0031 | 2.3690 0.0031 | 2.6890 0.0031 |
| *10.DCC..* | 0.2610 0.0031 | 0.3310 0.0031 | 0.4020 0.0042 | 0.6430 0.0048 | 0.6790 0.0056 | 2.0010 0.0031 |
| *11.Bra..* | 0.2710 0.0031 | 0.3560 0.0051 | 0.5210 0.0031 | 0.8100 0.0047 | 1.0310 0.0031 | 2.7930 0.0105 |
| *12.New..* | 0.2610 0.0031 | 0.3820 0.0042 | 0.5010 0.0031 | 0.5810 0.0031 | 0.8440 0.0096 | 0.9710 0.0031 |

*Continua na próxima página ..*

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
| *13.Mac..* | 0.2310 0.0031 | 0.3410 0.0031 | 0.4400 0.0047 | 0.6220 0.0042 | 0.9010 0.0031 | 1.5830 0.0048 |
| *14.The..* | 0.2210 0.0031 | 0.3210 0.0031 | 0.4000 0.0047 | 0.5310 0.0031 | 0.7010 0.0031 | 0.9610 0.0031 |
| *15.man..* | 0.2140 0.0051 | 0.2800 0.0000 | 0.3310 0.0031 | 0.3920 0.0042 | 0.4590 0.0031 | 0.5410 0.0031 |
| *16.Low..* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *17.Mr...* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |

Tabela 11: Dados de tempo de execução e desvio padrão do AGrep ao variar o padrão e a aproximação para a coleção `Wsj88`.

**Algoritmo Shift-And Aproximado (Coleção Wsj88_20)**

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.3950 0.0033 | 0.5243 0.0032 | 1.6878 0.0067 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.4030 0.0028 | 0.6533 0.0020 | 2.8517 0.0118 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.4071 0.0028 | 0.5238 0.0026 | 0.7350 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.4009 0.0029 | 0.4879 0.0036 | 0.6336 0.0024 | 0.9977 0.0028 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.4004 0.0032 | 0.4709 0.0037 | 0.5739 0.0031 | 0.6817 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.4131 0.0029 | 0.5083 0.0032 | 0.6502 0.0033 | 0.7882 0.0029 | 1.0105 0.0028 | 0.0000 0.0000 |
| *7.Mana..* | 0.4067 0.0025 | 0.4779 0.0029 | 0.5800 0.0034 | 0.6519 0.0030 | 0.7249 0.0032 | 0.8153 0.0029 |
| *8.Pric..* | 0.4098 0.0035 | 0.4840 0.0099 | 0.5841 0.0034 | 0.6573 0.0039 | 0.7398 0.0033 | 0.8682 0.0031 |
| *9.Admi..* | 0.4156 0.0028 | 0.4952 0.0028 | 0.6112 0.0026 | 0.6986 0.0030 | 0.7858 0.0038 | 0.8728 0.0035 |
| *10.DCC..* | 0.4162 0.0037 | 0.4872 0.0040 | 0.5893 0.0041 | 0.6609 0.0037 | 0.7330 0.0033 | 0.8045 0.0035 |
| *11.Bra..* | 0.4168 0.0043 | 0.4878 0.0039 | 0.5900 0.0044 | 0.6619 0.0040 | 0.7340 0.0035 | 0.8074 0.0034 |
| *12.New..* | 0.4299 0.0037 | 0.5093 0.0033 | 0.6197 0.0037 | 0.6992 0.0040 | 0.7798 0.0035 | 0.8593 0.0031 |
| *13.Mac..* | 0.4262 0.0049 | 0.4971 0.0041 | 0.5990 0.0044 | 0.6709 0.0043 | 0.7425 0.0042 | 0.8141 0.0037 |
| *14.The..* | 0.4337 0.0042 | 0.5046 0.0041 | 0.6072 0.0035 | 0.6787 0.0037 | 0.7546 0.0132 | 0.8394 0.0518 |
| *15.man..* | 0.4581 0.0041 | 0.5298 0.0043 | 0.6333 0.0043 | 0.7074 0.0042 | 0.7876 0.0040 | 0.8903 0.0038 |
| *16.Low..* | 0.5161 0.0044 | 0.5872 0.0043 | 0.6891 0.0043 | 0.7609 0.0041 | 0.8330 0.0038 | 0.9048 0.0037 |
| *17.Mr...* | 0.5476 0.0037 | 0.6185 0.0040 | 0.7206 0.0039 | 0.7927 0.0044 | 0.8674 0.0041 | 0.9498 0.0039 |

Tabela 12: Dados de tempo de execução e desvio padrão do algoritmo Shift-And Aproximado ao variar o padrão e a aproximação para a coleção `Wsj88_20`.

**Algoritmo de Programação Dinâmica (Coleção Wsj88_20)**

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.3570 0.0047 | 0.4905 0.0041 | 1.6410 0.0044 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.3656 0.0048 | 0.6376 0.0042 | 2.8666 0.0103 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.3686 0.0039 | 0.5208 0.0031 | 0.7845 0.0035 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.3602 0.0041 | 0.4759 0.0045 | 0.6597 0.0047 | 1.1510 0.0042 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.3578 0.0044 | 0.4384 0.0046 | 0.5947 0.0044 | 0.8307 0.0051 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.3738 0.0037 | 0.4737 0.0046 | 0.6290 0.0040 | 0.8849 0.0038 | 1.1715 0.0037 | 0.0000 0.0000 |
| *7.Mana..* | 0.3645 0.0036 | 0.4750 0.0046 | 0.6230 0.0045 | 0.8165 0.0040 | 0.9521 0.0128 | 1.0702 0.0039 |
| *8.Pric..* | 0.3631 0.0044 | 0.4629 0.0044 | 0.6145 0.0042 | 0.8376 0.0040 | 1.0166 0.0040 | 1.2438 0.0038 |
| | | | | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
| *9.Admi..* | 0.3708 0.0035 | 0.4662 0.0037 | 0.6057 0.0035 | 0.8314 0.0041 | 0.9781 0.0033 | 1.1238 0.0036 |
| *10.DCC..* | 0.3662 0.0041 | 0.4434 0.0043 | 0.5439 0.0047 | 0.7564 0.0038 | 0.8733 0.0033 | 1.0340 0.0033 |
| *11.Bra..* | 0.3686 0.0036 | 0.4678 0.0041 | 0.6148 0.0034 | 0.8193 0.0036 | 0.9556 0.0035 | 1.0751 0.0039 |
| *12.New..* | 0.3809 0.0038 | 0.5032 0.0034 | 0.6713 0.0046 | 0.9390 0.0032 | 1.0951 0.0039 | 1.2467 0.0043 |
| *13.Mac..* | 0.3725 0.0042 | 0.4741 0.0010 | 0.6131 0.0030 | 0.8278 0.0038 | 0.9494 0.0037 | 1.0735 0.0093 |
| *14.The..* | 0.3792 0.0037 | 0.4900 0.0042 | 0.6619 0.0042 | 0.9162 0.0039 | 1.0829 0.0048 | 1.2194 0.0034 |
| *15.man..* | 0.3939 0.0043 | 0.5193 0.0035 | 0.6801 0.0035 | 0.9132 0.0031 | 1.0896 0.0032 | 1.2338 0.0040 |
| *16.Low..* | 0.4128 0.0037 | 0.5120 0.0034 | 0.6499 0.0038 | 0.9022 0.0033 | 1.0406 0.0039 | 1.1978 0.0037 |
| *17.Mr...* | 0.4266 0.0041 | 0.5236 0.0045 | 0.6580 0.0049 | 0.9038 0.0041 | 1.0505 0.0036 | 1.2001 0.0041 |

Tabela 13: Dados de tempo de execução e desvio padrão do algoritmo de Programação Dinâmica ao variar o padrão e a aproximação para a coleção `Wsj88_20`.

### A Implementação AGrep (Coleção Wsj88_20)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0700 0.0000 | 0.1710 0.0031 | 0.5160 0.0069 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.0810 0.0031 | 0.3490 0.0056 | 1.0110 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.0710 0.0031 | 0.1650 0.0070 | 0.4410 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.0610 0.0031 | 0.1070 0.0067 | 0.2110 0.0031 | 0.7120 0.0042 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.0600 0.0000 | 0.0910 0.0031 | 0.1540 0.0069 | 0.5430 0.0067 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.0710 0.0031 | 0.0950 0.0070 | 0.2310 0.0031 | 0.2910 0.0031 | 0.7310 0.0031 | 0.0000 0.0000 |
| *7.Mana..* | 0.0500 0.0000 | 0.0610 0.0031 | 0.0910 0.0031 | 0.1610 0.0031 | 0.1810 0.0031 | 0.6910 0.0031 |
| *8.Pric..* | 0.0500 0.0000 | 0.0810 0.0031 | 0.1200 0.0000 | 0.2410 0.0031 | 0.3200 0.0000 | 0.7710 0.0031 |
| *9.Admi..* | 0.0500 0.0000 | 0.0910 0.0031 | 0.1410 0.0031 | 0.1810 0.0031 | 0.4400 0.0047 | 0.5010 0.0031 |
| *10.DCC..* | 0.0500 0.0000 | 0.0600 0.0000 | 0.0710 0.0031 | 0.1210 0.0031 | 0.1210 0.0031 | 0.3710 0.0031 |
| *11.Bra..* | 0.0510 0.0031 | 0.0610 0.0031 | 0.0910 0.0031 | 0.1510 0.0031 | 0.1900 0.0047 | 0.5190 0.0031 |
| *12.New..* | 0.0480 0.0063 | 0.0710 0.0031 | 0.0910 0.0031 | 0.1000 0.0000 | 0.1510 0.0031 | 0.1810 0.0031 |
| *13.Mac..* | 0.0400 0.0000 | 0.0610 0.0031 | 0.0810 0.0031 | 0.1120 0.0063 | 0.1710 0.0031 | 0.2910 0.0031 |
| *14.The..* | 0.0400 0.0000 | 0.0610 0.0031 | 0.0710 0.0031 | 0.1010 0.0031 | 0.1310 0.0031 | 0.1780 0.0063 |
| *15.man..* | 0.0410 0.0031 | 0.0510 0.0031 | 0.0600 0.0000 | 0.0710 0.0031 | 0.0810 0.0031 | 0.1010 0.0031 |
| *16.Low..* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *17.Mr...* | 0.0010 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0010 0.0000 | 0.0000 0.0000 |

Tabela 14: Dados de tempo de execução e desvio padrão do AGrep ao variar o padrão e a aproximação para a coleção `Wsj88_20`.

### Algoritmo Shift-And Aproximado (Coleção Wsj88_10)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.2036 0.0073 | 0.2684 0.0063 | 0.8394 0.0061 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.2076 0.0077 | 0.3307 0.0072 | 1.4175 0.0040 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.2100 0.0077 | 0.2665 0.0079 | 0.3694 0.0074 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.2069 0.0083 | 0.2509 0.0079 | 0.3231 0.0075 | 0.5102 0.0091 | 0.0000 0.0000 | 0.0000 0.0000 |
| | | | | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | | | | |
|---|---|---|---|---|---|---|
| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
| *5.Uber..* | 0.2066 0.0083 | 0.2429 0.0085 | 0.2939 0.0078 | 0.3457 0.0087 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.2403 0.0603 | 0.2592 0.0026 | 0.3312 0.0026 | 0.4011 0.0024 | 0.5132 0.0028 | 0.0000 0.0000 |
| *7.Mana..* | 0.2077 0.0032 | 0.2426 0.0024 | 0.2969 0.0099 | 0.3302 0.0025 | 0.3665 0.0026 | 0.4106 0.0024 |
| *8.Pric..* | 0.2091 0.0031 | 0.2447 0.0028 | 0.2965 0.0032 | 0.3338 0.0025 | 0.3746 0.0030 | 0.4396 0.0035 |
| *9.Admi..* | 0.2120 0.0037 | 0.2513 0.0030 | 0.3081 0.0033 | 0.3502 0.0036 | 0.3932 0.0037 | 0.4362 0.0039 |
| *10.DCC..* | 0.2124 0.0038 | 0.2482 0.0038 | 0.2989 0.0038 | 0.3334 0.0012 | 0.3708 0.0037 | 0.4067 0.0036 |
| *11.Bra..* | 0.2128 0.0040 | 0.2484 0.0039 | 0.2993 0.0045 | 0.3352 0.0036 | 0.3715 0.0043 | 0.4088 0.0037 |
| *12.New..* | 0.2192 0.0036 | 0.2586 0.0041 | 0.3134 0.0037 | 0.3528 0.0038 | 0.3925 0.0037 | 0.4321 0.0035 |
| *13.Mac..* | 0.2174 0.0040 | 0.2528 0.0043 | 0.3026 0.0011 | 0.3398 0.0041 | 0.3759 0.0036 | 0.4116 0.0047 |
| *14.The..* | 0.2214 0.0042 | 0.2569 0.0044 | 0.3079 0.0040 | 0.3439 0.0043 | 0.3798 0.0044 | 0.4157 0.0043 |
| *15.man..* | 0.2339 0.0045 | 0.2693 0.0044 | 0.3209 0.0043 | 0.3574 0.0045 | 0.3971 0.0045 | 0.4454 0.0047 |
| *16.Low..* | 0.2629 0.0045 | 0.2984 0.0042 | 0.3492 0.0046 | 0.3850 0.0044 | 0.4211 0.0034 | 0.4571 0.0037 |
| *17.Mr...* | 0.2788 0.0041 | 0.3126 0.0007 | 0.3652 0.0041 | 0.4012 0.0037 | 0.4388 0.0039 | 0.4802 0.0036 |

Tabela 15: Dados de tempo de execução e desvio padrão do algoritmo Shift-And Aproximado ao variar o padrão e a aproximação para a coleção `Wsj88_10`.

## Algoritmo de Programação Dinâmica (Coleção Wsj88_10)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.1827 0.0045 | 0.2488 0.0043 | 0.8116 0.0045 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.1853 0.0010 | 0.3197 0.0040 | 1.4165 0.0042 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.1881 0.0039 | 0.2622 0.0044 | 0.3896 0.0048 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.1839 0.0042 | 0.2419 0.0052 | 0.3338 0.0045 | 0.5820 0.0034 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.1832 0.0045 | 0.2233 0.0046 | 0.3009 0.0045 | 0.4167 0.0048 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.1913 0.0041 | 0.2415 0.0040 | 0.3207 0.0045 | 0.4497 0.0040 | 0.5930 0.0037 | 0.0000 0.0000 |
| *7.Mana..* | 0.1848 0.0003 | 0.2415 0.0038 | 0.3155 0.0038 | 0.4126 0.0036 | 0.4783 0.0037 | 0.5393 0.0049 |
| *8.Pric..* | 0.1855 0.0039 | 0.2355 0.0045 | 0.3117 0.0041 | 0.4231 0.0044 | 0.5129 0.0041 | 0.6278 0.0046 |
| *9.Admi..* | 0.1895 0.0039 | 0.2362 0.0041 | 0.3046 0.0049 | 0.4168 0.0042 | 0.4891 0.0047 | 0.5623 0.0044 |
| *10.DCC..* | 0.1873 0.0042 | 0.2258 0.0047 | 0.2747 0.0012 | 0.3821 0.0040 | 0.4408 0.0036 | 0.5216 0.0041 |
| *11.Bra..* | 0.1885 0.0037 | 0.2379 0.0036 | 0.3116 0.0036 | 0.4133 0.0034 | 0.4817 0.0039 | 0.5419 0.0038 |
| *12.New..* | 0.1942 0.0038 | 0.2549 0.0036 | 0.3390 0.0040 | 0.4727 0.0040 | 0.5532 0.0102 | 0.6260 0.0043 |
| *13.Mac..* | 0.1906 0.0043 | 0.2413 0.0042 | 0.3105 0.0044 | 0.4185 0.0041 | 0.4793 0.0046 | 0.5401 0.0053 |
| *14.The..* | 0.1940 0.0043 | 0.2499 0.0049 | 0.3353 0.0045 | 0.4628 0.0045 | 0.5461 0.0037 | 0.6144 0.0037 |
| *15.man..* | 0.2012 0.0042 | 0.2640 0.0043 | 0.3446 0.0036 | 0.4610 0.0039 | 0.5493 0.0039 | 0.6219 0.0038 |
| *16.Low..* | 0.2106 0.0040 | 0.2603 0.0041 | 0.3292 0.0040 | 0.4549 0.0044 | 0.5247 0.0043 | 0.6031 0.0043 |
| *17.Mr...* | 0.2177 0.0042 | 0.2658 0.0043 | 0.3327 0.0041 | 0.4558 0.0046 | 0.5289 0.0045 | 0.6039 0.0045 |

Tabela 16: Dados de tempo de execução e desvio padrão do algoritmo de Programação Dinâmica ao variar o padrão e a aproximação para a coleção `Wsj88_10`.

## A Implementação AGrep (Coleção Wsj88_10)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0330 0.0067 | 0.0820 0.0063 | 0.2520 0.0063 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.0400 0.0000 | 0.1710 0.0031 | 0.5030 0.0048 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.0300 0.0000 | 0.0810 0.0031 | 0.2200 0.0047 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.0300 0.0000 | 0.0510 0.0031 | 0.1010 0.0031 | 0.3610 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.0300 0.0000 | 0.0400 0.0000 | 0.0810 0.0031 | 0.2710 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.0310 0.0031 | 0.0510 0.0031 | 0.1180 0.0063 | 0.1440 0.0069 | 0.3640 0.0069 | 0.0000 0.0000 |
| *7.Mana..* | 0.0200 0.0000 | 0.0310 0.0031 | 0.0450 0.0052 | 0.0800 0.0000 | 0.0910 0.0031 | 0.3410 0.0031 |
| *8.Pric..* | 0.0200 0.0000 | 0.0400 0.0000 | 0.0600 0.0000 | 0.1210 0.0031 | 0.1610 0.0031 | 0.3820 0.0042 |
| *9.Admi..* | 0.0220 0.0042 | 0.0400 0.0000 | 0.0710 0.0031 | 0.0910 0.0031 | 0.2210 0.0031 | 0.2490 0.0056 |
| *10.DCC..* | 0.0200 0.0000 | 0.0300 0.0000 | 0.0370 0.0067 | 0.0600 0.0000 | 0.0610 0.0031 | 0.1810 0.0031 |
| *11.Bra..* | 0.0210 0.0031 | 0.0300 0.0000 | 0.0510 0.0031 | 0.0710 0.0031 | 0.0920 0.0063 | 0.2600 0.0047 |
| *12.New..* | 0.0210 0.0031 | 0.0300 0.0000 | 0.0420 0.0063 | 0.0510 0.0031 | 0.0730 0.0048 | 0.0910 0.0031 |
| *13.Mac..* | 0.0200 0.0000 | 0.0310 0.0031 | 0.0400 0.0000 | 0.0600 0.0000 | 0.0810 0.0031 | 0.1500 0.0047 |
| *14.The..* | 0.0210 0.0031 | 0.0300 0.0000 | 0.0330 0.0048 | 0.0500 0.0000 | 0.0610 0.0031 | 0.0910 0.0031 |
| *15.man..* | 0.0210 0.0031 | 0.0200 0.0000 | 0.0310 0.0031 | 0.0320 0.0042 | 0.0400 0.0000 | 0.0500 0.0000 |
| *16.Low..* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0010 0.0031 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *17.Mr...* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |

Tabela 17: Dados de tempo de execução e desvio padrão do algoritmo AGrep ao variar o padrão e a aproximação para a coleção `Wsj88_10`.

### Algoritmo Shift-And Aproximado (Coleção Wsj89)

| *Padrão* | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0638 0.0007 | 0.0824 0.0006 | 0.2709 0.0166 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.0663 0.0045 | 0.1009 0.0139 | 0.3915 0.0349 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.0697 0.0149 | 0.0802 0.0008 | 0.1077 0.0006 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.0695 0.0176 | 0.0753 0.0006 | 0.0975 0.0053 | 0.1417 0.0009 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.0688 0.0113 | 0.0768 0.0082 | 0.0990 0.0178 | 0.1022 0.0010 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.0678 0.0043 | 0.0839 0.0106 | 0.1000 0.0089 | 0.1189 0.0108 | 0.1477 0.0086 | 0.0000 0.0000 |
| *7.Mana..* | 0.0650 0.0005 | 0.0781 0.0089 | 0.0929 0.0087 | 0.1034 0.0087 | 0.1149 0.0089 | 0.1268 0.0083 |
| *8.Pric..* | 0.0680 0.0082 | 0.0779 0.0085 | 0.0922 0.0084 | 0.1030 0.0106 | 0.1138 0.0106 | 0.1312 0.0085 |
| *9.Admi..* | 0.0660 0.0005 | 0.0802 0.0088 | 0.0972 0.0107 | 0.1096 0.0108 | 0.1214 0.0088 | 0.1338 0.0087 |
| *10.DCC..* | 0.0692 0.0088 | 0.0794 0.0089 | 0.0941 0.0090 | 0.1044 0.0093 | 0.1149 0.0092 | 0.1260 0.0095 |
| *11.Bra..* | 0.0694 0.0094 | 0.0761 0.0005 | 0.0935 0.0094 | 0.1035 0.0100 | 0.1135 0.0100 | 0.1236 0.0098 |
| *12.New..* | 0.0716 0.0103 | 0.0822 0.0100 | 0.0976 0.0099 | 0.1083 0.0105 | 0.1169 0.0026 | 0.1304 0.0108 |
| *13.Mac..* | 0.0711 0.0109 | 0.0809 0.0110 | 0.0954 0.0110 | 0.1055 0.0112 | 0.1162 0.0135 | 0.1265 0.0113 |
| *14.The..* | 0.0690 0.0006 | 0.0787 0.0007 | 0.0963 0.0112 | 0.1062 0.0112 | 0.1164 0.0111 | 0.1261 0.0106 |
| *15.man..* | 0.0764 0.0112 | 0.0861 0.0106 | 0.1004 0.0111 | 0.1111 0.0111 | 0.1220 0.0112 | 0.1356 0.0108 |
| *16.Low..* | 0.0817 0.0007 | 0.0952 0.0106 | 0.1092 0.0106 | 0.1194 0.0108 | 0.1255 0.0008 | 0.1401 0.0106 |
| *17.Mr...* | 0.0908 0.0135 | 0.1000 0.0112 | 0.1140 0.0114 | 0.1207 0.0007 | 0.1353 0.0116 | 0.1458 0.0112 |

Tabela 18: Dados de tempo de execução e desvio padrão do algoritmo Shift-And Aproximado ao variar o padrão e a aproximação para a coleção `Wsj89`.

### Algoritmo de Programação Dinâmica (Coleção Wsj89)

| *Padrão* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0567 0.0005 | 0.0793 0.0118 | 0.2596 0.0112 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.0568 0.0004 | 0.0957 0.0115 | 0.3857 0.0114 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.0608 0.0109 | 0.0811 0.0108 | 0.1158 0.0112 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.0597 0.0109 | 0.0753 0.0108 | 0.0998 0.0111 | 0.1651 0.0112 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.0559 0.0005 | 0.0706 0.0116 | 0.0920 0.0110 | 0.1249 0.0112 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.0581 0.0007 | 0.0751 0.0115 | 0.0967 0.0133 | 0.1313 0.0111 | 0.1675 0.0113 | 0.0000 0.0000 |
| *7.Mana..* | 0.0570 0.0004 | 0.0765 0.0116 | 0.0969 0.0103 | 0.1247 0.0107 | 0.1431 0.0112 | 0.1610 0.0112 |
| *8.Pric..* | 0.0570 0.0006 | 0.0740 0.0112 | 0.0948 0.0107 | 0.1227 0.0027 | 0.1493 0.0104 | 0.1807 0.0107 |
| *9.Admi..* | 0.0580 0.0005 | 0.0747 0.0108 | 0.0912 0.0010 | 0.1260 0.0107 | 0.1474 0.0106 | 0.1643 0.0025 |
| *10.DCC..* | 0.0577 0.0005 | 0.0722 0.0111 | 0.0867 0.0107 | 0.1164 0.0108 | 0.1307 0.0029 | 0.1562 0.0112 |
| *11.Bra..* | 0.0613 0.0111 | 0.0713 0.0004 | 0.0950 0.0110 | 0.1230 0.0109 | 0.1429 0.0109 | 0.1589 0.0115 |
| *12.New..* | 0.0592 0.0007 | 0.0796 0.0114 | 0.1024 0.0111 | 0.1402 0.0115 | 0.1617 0.0115 | 0.1816 0.0112 |
| *13.Mac..* | 0.0583 0.0005 | 0.0759 0.0114 | 0.0915 0.0008 | 0.1245 0.0112 | 0.1414 0.0113 | 0.1588 0.0115 |
| *14.The..* | 0.0591 0.0004 | 0.0780 0.0112 | 0.1016 0.0108 | 0.1374 0.0107 | 0.1603 0.0110 | 0.1785 0.0110 |
| *15.man..* | 0.0614 0.0007 | 0.0822 0.0112 | 0.1042 0.0108 | 0.1371 0.0114 | 0.1601 0.0114 | 0.1802 0.0113 |
| *16.Low..* | 0.0644 0.0006 | 0.0817 0.0113 | 0.1011 0.0112 | 0.1362 0.0113 | 0.1546 0.0115 | 0.1771 0.0113 |
| *17.Mr...* | 0.0701 0.0116 | 0.0835 0.0110 | 0.1020 0.0117 | 0.1368 0.0116 | 0.1561 0.0115 | 0.1778 0.0115 |

Tabela 19: Dados de tempo de execução e desvio padrão do algoritmo de PROGRAMAÇÃO DINÂMICA ao variar o padrão e a aproximação para a coleção `Wsj89`.

## A Implementação AGrep (Coleção Wsj89)

| *Padrão* | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| *1.DCC ..* | 0.0100 0.0000 | 0.0240 0.0126 | 0.0730 0.0094 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *2.That..* | 0.0100 0.0000 | 0.0440 0.0126 | 0.1350 0.0126 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *3.Stoc..* | 0.0100 0.0000 | 0.0200 0.0000 | 0.0630 0.0094 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *4.Doll..* | 0.0100 0.0000 | 0.0100 0.0000 | 0.0330 0.0094 | 0.0950 0.0126 | 0.0000 0.0000 | 0.0000 0.0000 |
| *5.Uber..* | 0.0100 0.0000 | 0.0100 0.0000 | 0.0200 0.0000 | 0.0740 0.0126 | 0.0000 0.0000 | 0.0000 0.0000 |
| *6.Exch..* | 0.0100 0.0000 | 0.0130 0.0094 | 0.0300 0.0000 | 0.0400 0.0000 | 0.1000 0.0000 | 0.0000 0.0000 |
| *7.Mana..* | 0.0040 0.0051 | 0.0130 0.0094 | 0.0100 0.0000 | 0.0230 0.0094 | 0.0200 0.0000 | 0.0930 0.0094 |
| *8.Pric..* | 0.0040 0.0126 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0330 0.0094 | 0.0430 0.0094 | 0.1060 0.0096 |
| *9.Admi..* | 0.0100 0.0000 | 0.0100 0.0000 | 0.0200 0.0000 | 0.0230 0.0094 | 0.0600 0.0000 | 0.0690 0.0031 |
| *10.DCC..* | 0.0000 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0180 0.0042 | 0.0530 0.0094 |
| *11.Bra..* | 0.0020 0.0042 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0200 0.0000 | 0.0200 0.0000 | 0.0720 0.0063 |
| *12.New..* | 0.0020 0.0042 | 0.0130 0.0094 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0230 0.0094 | 0.0200 0.0000 |
| *13.Mac..* | 0.0030 0.0094 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0130 0.0094 | 0.0200 0.0000 | 0.0430 0.0094 |
| *14.The..* | 0.0000 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0190 0.0031 | 0.0200 0.0000 |
| *15.man..* | 0.0000 0.0000 | 0.0080 0.0042 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 | 0.0100 0.0000 |
| *16.Low..* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |
| *17.Mr...* | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 | 0.0000 0.0000 |

Tabela 20: Dados de tempo de execução e desvio padrão do AGREP ao variar o padrão e a aproximação para a coleção `Wsj89`.

## Comparação do Tempo dos Algoritmos por Aproximação

## Para a Coleção Wsj88

| *Aproximação* | **SAA** | **PD** | **AGrep** |
|---|---|---|---|
| → **Padrão 1: "DCC"** | | | |
| *0* | 2.5283 01.343 | 1.8927 0.0034 | 0.3910 0.0031 |
| *1* | 2.8091 0.0033 | 2.6184 0.0031 | 0.9370 0.0048 |
| → **Padrão 2: "That"** | | | |
| *0* | 2.1481 0.0025 | 1.9379 0.0026 | 0.4510 0.0031 |
| *1* | 3.4787 0.0049 | 3.3878 0.0039 | 1.8620 0.0042 |
| → **Padrão 3: "Stock"** | | | |
| *0* | 2.2235 0.1013 | 1.9640 0.0064 | 0.3840 0.0096 |
| *1* | 2.8174 0.0061 | 2.7915 0.0078 | 0.9140 0.0096 |
| *2* | 3.9639 0.0073 | 4.2166 0.0069 | 2.3720 0.0063 |
| → **Padrão 4: "Dollar"** | | | |
| *0* | 2.1464 0.0071 | 1.9192 0.0067 | 0.3350 0.0097 |
| *1* | 2.6281 0.0099 | 2.5823 0.0861 | 0.5900 0.0000 |
| *2* | 3.4157 0.0035 | 3.5491 0.0044 | 1.1310 0.0031 |
| *3* | 5.4086 0.0077 | 6.2175 0.0051 | 3.8630 0.0048 |
| → **Padrão 5: "Uberaba"** | | | |
| *0* | 2.1373 0.0027 | 1.8981 0.0023 | 0.3110 0.0031 |
| *1* | 2.5163 0.0027 | 2.3515 0.0407 | 0.4970 0.0094 |
| *2* | 3.0756 0.0062 | 3.1780 0.0065 | 0.8510 0.0073 |
| *3* | 3.6391 0.0072 | 4.4308 0.0086 | 2.9340 0.0069 |
| → **Padrão 6: "Exchange"** | | | |
| *0* | 2.2173 0.0070 | 1.9997 0.0082 | 0.3880 0.0122 |
| *1* | 2.7468 0.0073 | 2.5770 0.0893 | 0.5540 0.0447 |
| *2* | 3.5285 0.0082 | 3.4038 0.0088 | 1.2740 0.0069 |
| *3* | 4.2850 0.0074 | 4.7977 0.0080 | 1.5920 0.0063 |
| *4* | 5.4713 0.0093 | 6.3274 0.0060 | 3.9430 0.0094 |
| → **Padrão 7: "Manacapuru"** | | | |
| *0* | 2.1782 0.0063 | 1.9425 0.0083 | 0.2740 0.0096 |
| *1* | 2.5619 0.0068 | 2.5631 0.0907 | 0.3510 0.0031 |
| *2* | 3.1012 0.0028 | 3.3257 0.0027 | 0.5070 0.0048 |
| *3* | 3.4875 0.0028 | 4.3650 0.0024 | 0.8590 0.0031 |
| *4* | 3.8940 0.0391 | 5.0865 0.0094 | 0.9990 0.0087 |
| *5* | 4.3784 0.0070 | 5.7474 0.0121 | 3.7360 0.0096 |
| → **Padrão 8: "Price Index"** | | | |
| *0* | 2.1954 0.0072 | 1.9357 0.0082 | 0.2750 0.0097 |
| *1* | 2.5778 0.0067 | 2.4997 0.0888 | 0.4410 0.0031 |
| *2* | 3.1368 0.0415 | 3.2865 0.0097 | 0.6420 0.0147 |
| *3* | 3.5266 0.0068 | 4.4887 0.0080 | 1.3080 0.0122 |
| *4* | 3.9626 0.0065 | 5.4424 0.0069 | 1.7500 0.0081 |

*Continua na próxima página ..*

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *5* | 4.6483 0.0075 | 6.8333 0.1713 | 4.2650 0.0594 |
| → **Padrão 9: "Administration"** | | | |
| *0* | 2.4191 0.2348 | 2.0732 0.0378 | 0.3100 0.0141 |
| *1* | 2.7482 0.0625 | 2.6123 0.1054 | 0.5520 0.0431 |
| *2* | 3.3983 0.1171 | 3.5866 0.4032 | 0.8780 0.0664 |
| *3* | 4.3306 0.3067 | 4.6253 0.1154 | 1.0750 0.0356 |
| *4* | 4.4519 0.1079 | 5.5353 0.0990 | 2.4890 0.0460 |
| *5* | 5.0259 0.3242 | 6.5396 0.5911 | 2.7560 0.0501 |
| → **Padrão 10: "DCC UFMG dollar"** | | | |
| *0* | 2.2374 0.0117 | 1.9758 0.0351 | 0.2620 0.0042 |
| *1* | 2.8465 0.3239 | 2.5896 0.3663 | 0.3310 0.0031 |
| *2* | 3.4051 0.2173 | 3.0673 0.0995 | 0.4280 0.0244 |
| *3* | 3.9521 0.3488 | 4.8580 0.7062 | 0.6660 0.0471 |
| *4* | 4.0991 0.1284 | 4.8015 0.1075 | 0.7050 0.0212 |
| *5* | 4.4321 0.0689 | 5.9161 0.2598 | 2.1320 0.1153 |
| → **Padrão 11: "Brazilian coffee"** | | | |
| *0* | 2.5238 0.1583 | 2.1986 0.2465 | 0.3070 0.0302 |
| *1* | 2.7149 0.0846 | 2.5883 0.0865 | 0.3660 0.0177 |
| *2* | 3.4655 0.4738 | 3.4723 0.1642 | 0.5750 0.0575 |
| *3* | 3.6950 0.1521 | 4.6902 0.4701 | 0.8740 0.0505 |
| *4* | 4.0234 0.0837 | 5.5163 0.3147 | 1.1230 0.0714 |
| *5* | 4.8089 0.2271 | 6.0997 0.2379 | 2.8620 0.0702 |
| → **Padrão 12: "New York Stock .."** | | | |
| *0* | 2.3522 0.0303 | 2.0720 0.0282 | 0.2620 0.0042 |
| *1* | 2.8179 0.0454 | 2.7450 0.0242 | 0.3910 0.0031 |
| *2* | 3.3695 0.0059 | 3.6635 0.0429 | 0.5230 0.0402 |
| *3* | 3.8624 0.0550 | 5.2048 0.0798 | 0.6130 0.0518 |
| *4* | 4.5391 0.1827 | 6.3743 0.4561 | 0.8870 0.0577 |
| *5* | 4.9018 0.2142 | 6.9395 0.1988 | 1.0080 0.0565 |
| → **Padrão 13: "Macunaima admin .."** | | | |
| *0* | 2.5757 0.2405 | 2.3831 0.0670 | 0.2710 0.0268 |
| *1* | 2.7517 0.1075 | 2.7815 0.1690 | 0.4130 0.1238 |
| *2* | 3.7174 0.3634 | 3.4975 0.1910 | 0.4510 0.0110 |
| *3* | 4.1127 0.4869 | 4.6752 0.1956 | 0.6870 0.0746 |
| *4* | 4.3786 0.2674 | 5.5697 0.6251 | 0.9240 0.0084 |
| *5* | 4.7233 0.2616 | 6.3492 0.6002 | 1.7230 0.1826 |
| → **Padrão 14: "The agency sees .."** | | | |
| *0* | 2.4842 0.1349 | 2.1803 0.1397 | 0.2280 0.0103 |
| *1* | 2.9604 0.1743 | 2.7389 0.1429 | 0.3480 0.0209 |
| *2* | 3.4309 0.1896 | 4.0670 0.3518 | 0.4570 0.0974 |
| *3* | 3.9146 0.2542 | 5.4533 0.2362 | 0.5600 0.0141 |
| *4* | 4.1847 0.0749 | 6.4968 0.2450 | 0.7290 0.0268 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *5* | 4.8950 0.2598 | 7.1256 0.4457 | 0.9840 0.0157 |
| → **Padrão 15: "many investors .."** | | | |
| *0* | 2.5461 0.0548 | 2.1877 0.0665 | 0.2670 0.0605 |
| *1* | 2.8637 0.0499 | 2.7852 0.0100 | 0.2970 0.0082 |
| *2* | 3.5654 0.1577 | 3.8117 0.1621 | 0.3570 0.0105 |
| *3* | 4.1116 0.2044 | 5.1188 0.2581 | 0.4260 0.0157 |
| *4* | 4.4986 0.1432 | 6.0300 0.0545 | 0.5260 0.1220 |
| *5* | 5.2097 0.2504 | 7.3609 0.4031 | 0.6040 0.0791 |
| → **Padrão 16: "Low wages and a .."** | | | |
| *0* | 3.0621 0.2549 | 2.5315 0.2504 | 0.0000 0.0000 |
| *1* | 3.2729 0.1301 | 3.0758 0.2364 | 0.0000 0.0000 |
| *2* | 4.0318 0.2693 | 3.9047 0.2814 | 0.0000 0.0000 |
| *3* | 4.3469 0.3720 | 5.3939 0.6062 | 0.0000 0.0000 |
| *4* | 4.6301 0.1445 | 5.9350 0.2920 | 0.0000 0.0000 |
| *5* | 5.2905 0.2957 | 7.3217 0.3071 | 0.0000 0.0000 |
| → **Padrão 17: "Mr. Jordan said .."** | | | |
| *0* | 3.2210 0.2225 | 2.6503 0.3174 | 0.0000 0.0000 |
| *1* | 3.3884 0.0695 | 2.9852 0.1367 | 0.0000 0.0000 |
| *2* | 4.2516 0.2946 | 3.6478 0.1730 | 0.0000 0.0000 |
| *3* | 4.5841 0.1915 | 5.3700 0.3094 | 0.0000 0.0000 |
| *4* | 5.1187 0.3470 | 6.2296 0.4746 | 0.0000 0.0000 |
| *5* | 5.5577 0.3395 | 6.9712 0.5525 | 0.0000 0.0000 |

Tabela 21: Dados de tempo de execução e desvio padrão da execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88`.

### Para a Coleção Wsj88_20

| *Aproximação* | **SAA** | **PD** | **AGrep** |
|---|---|---|---|
| → **Padrão 1: "DCC"** | | | |
| *0* | 0.4974 0.3301 | 0.3561 0.0039 | 0.0710 0.0031 |
| *1* | 0.5235 0.0037 | 0.4892 0.0044 | 0.1710 0.0031 |
| *2* | 1.6794 0.0052 | 1.6322 0.0044 | 0.5130 0.0067 |
| → **Padrão 2: "That"** | | | |
| *0* | 0.4021 0.0040 | 0.3644 0.0037 | 0.0810 0.0031 |
| *1* | 0.6516 0.0038 | 0.6364 0.0042 | 0.3440 0.0069 |
| *2* | 2.8428 0.0125 | 2.8521 0.0061 | 1.0010 0.0031 |
| → **Padrão 3: "Stock"** | | | |
| *0* | 0.4064 0.0047 | 0.3667 0.0013 | 0.0710 0.0031 |
| *1* | 0.5227 0.0040 | 0.5197 0.0042 | 0.1670 0.0067 |
| *2* | 0.7333 0.0042 | 0.7812 0.0036 | 0.4410 0.0031 |
| → **Padrão 4: "Dollar"** | | | |
| *0* | 0.4001 0.0039 | 0.3593 0.0035 | 0.0610 0.0031 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *1* | 0.4871 0.0036 | 0.4747 0.0041 | 0.1070 0.0067 |
| *2* | 0.6322 0.0043 | 0.6582 0.0034 | 0.2110 0.0031 |
| *3* | 0.9944 0.0036 | 1.1459 0.0029 | 0.7110 0.0031 |
| → **Padrão 5: "Uberaba"** | | | |
| *0* | 0.3998 0.0045 | 0.3570 0.0045 | 0.0600 0.0000 |
| *1* | 0.4703 0.0040 | 0.4380 0.0042 | 0.0910 0.0031 |
| *2* | 0.5728 0.0042 | 0.5931 0.0033 | 0.1540 0.0069 |
| *3* | 0.6801 0.0037 | 0.8282 0.0039 | 0.5410 0.0031 |
| → **Padrão 6: "Exchange"** | | | |
| *0* | 0.4124 0.0044 | 0.3731 0.0038 | 0.0710 0.0031 |
| *1* | 0.5073 0.0041 | 0.4722 0.0037 | 0.0940 0.0069 |
| *2* | 0.6491 0.0042 | 0.6270 0.0040 | 0.2310 0.0031 |
| *3* | 0.7859 0.0039 | 0.8835 0.0040 | 0.2910 0.0031 |
| *4* | 1.0074 0.0039 | 1.1673 0.0037 | 0.7310 0.0031 |
| → **Padrão 7: "Manacapuru"** | | | |
| *0* | 0.4061 0.0038 | 0.3634 0.0042 | 0.0500 0.0000 |
| *1* | 0.4769 0.0037 | 0.4739 0.0040 | 0.0600 0.0000 |
| *2* | 0.5789 0.0043 | 0.6213 0.0040 | 0.0900 0.0000 |
| *3* | 0.6504 0.0035 | 0.8151 0.0043 | 0.1610 0.0031 |
| *4* | 0.7231 0.0043 | 0.9461 0.0037 | 0.1790 0.0031 |
| *5* | 0.8134 0.0037 | 1.0674 0.0031 | 0.6930 0.0067 |
| → **Padrão 8: "Price Index"** | | | |
| *0* | 0.4092 0.0040 | 0.3626 0.0034 | 0.0510 0.0031 |
| *1* | 0.4802 0.0036 | 0.4616 0.0044 | 0.0810 0.0031 |
| *2* | 0.5825 0.0038 | 0.6129 0.0047 | 0.1210 0.0031 |
| *3* | 0.6559 0.0036 | 0.8360 0.0037 | 0.2400 0.0000 |
| *4* | 0.7377 0.0040 | 1.0148 0.0033 | 0.3210 0.0031 |
| *5* | 0.8659 0.0034 | 1.2405 0.0039 | 0.7710 0.0031 |
| → **Padrão 9: "Administration"** | | | |
| *0* | 0.4149 0.0039 | 0.3698 0.0036 | 0.0500 0.0000 |
| *1* | 0.4946 0.0041 | 0.4652 0.0037 | 0.0910 0.0031 |
| *2* | 0.6098 0.0035 | 0.6040 0.0044 | 0.1410 0.0031 |
| *3* | 0.6973 0.0043 | 0.8296 0.0038 | 0.1820 0.0063 |
| *4* | 0.7839 0.0038 | 0.9763 0.0037 | 0.4370 0.0067 |
| *5* | 0.8708 0.0041 | 1.1212 0.0035 | 0.4950 0.0070 |
| → **Padrão 10: "DCC UFMG dollar"** | | | |
| *0* | 0.4156 0.0036 | 0.3658 0.0040 | 0.0500 0.0000 |
| *1* | 0.4864 0.0038 | 0.4427 0.0037 | 0.0600 0.0000 |
| *2* | 0.5883 0.0041 | 0.5424 0.0035 | 0.0710 0.0031 |
| *3* | 0.6597 0.0039 | 0.7550 0.0034 | 0.1210 0.0031 |
| *4* | 0.7311 0.0035 | 0.8714 0.0040 | 0.1200 0.0000 |
| *5* | 0.8028 0.0044 | 1.0323 0.0036 | 0.3710 0.0031 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| → **Padrão 11: "Brazilian coffee"** | | | |
| *0* | 0.4160 0.0043 | 0.3680 0.0037 | 0.0500 0.0000 |
| *1* | 0.4868 0.0037 | 0.4663 0.0036 | 0.0600 0.0000 |
| *2* | 0.5887 0.0039 | 0.6138 0.0041 | 0.0920 0.0063 |
| *3* | 0.6602 0.0039 | 0.8173 0.0035 | 0.1510 0.0031 |
| *4* | 0.7322 0.0038 | 0.9539 0.0037 | 0.1890 0.0056 |
| *5* | 0.8052 0.0041 | 1.0714 0.0041 | 0.5150 0.0084 |
| → **Padrão 12: "New York Stock .."** | | | |
| *0* | 0.4279 0.0013 | 0.3802 0.0038 | 0.0460 0.0051 |
| *1* | 0.5080 0.0046 | 0.5018 0.0041 | 0.0700 0.0000 |
| *2* | 0.6179 0.0039 | 0.6698 0.0042 | 0.0910 0.0031 |
| *3* | 0.6971 0.0044 | 0.9366 0.0032 | 0.1020 0.0063 |
| *4* | 0.7774 0.0033 | 1.0923 0.0040 | 0.1510 0.0031 |
| *5* | 0.8568 0.0043 | 1.2440 0.0036 | 0.1810 0.0031 |
| → **Padrão 13: "Macunaima admin .."** | | | |
| *0* | 0.4248 0.0041 | 0.3722 0.0043 | 0.0400 0.0000 |
| *1* | 0.4958 0.0035 | 0.4737 0.0028 | 0.0600 0.0000 |
| *2* | 0.5972 0.0027 | 0.6112 0.0027 | 0.0810 0.0031 |
| *3* | 0.6687 0.0024 | 0.8256 0.0033 | 0.1120 0.0042 |
| *4* | 0.7404 0.0032 | 0.9463 0.0027 | 0.1640 0.0051 |
| *5* | 0.8118 0.0025 | 1.0674 0.0033 | 0.2910 0.0031 |
| → **Padrão 14: "The agency sees .."** | | | |
| *0* | 0.4326 0.0034 | 0.3783 0.0031 | 0.0400 0.0000 |
| *1* | 0.5031 0.0034 | 0.4886 0.0028 | 0.0610 0.0031 |
| *2* | 0.6049 0.0030 | 0.6598 0.0026 | 0.0710 0.0031 |
| *3* | 0.6764 0.0029 | 0.9134 0.0023 | 0.1000 0.0000 |
| *4* | 0.7480 0.0029 | 1.0794 0.0024 | 0.1300 0.0000 |
| *5* | 0.8194 0.0026 | 1.2156 0.0029 | 0.1760 0.0051 |
| → **Padrão 15: "many investors .."** | | | |
| *0* | 0.4567 0.0032 | 0.3927 0.0033 | 0.0400 0.0000 |
| *1* | 0.5280 0.0034 | 0.5180 0.0027 | 0.0500 0.0000 |
| *2* | 0.6316 0.0029 | 0.6786 0.0032 | 0.0610 0.0031 |
| *3* | 0.7052 0.0027 | 0.9106 0.0026 | 0.0710 0.0031 |
| *4* | 0.7854 0.0029 | 1.0854 0.0025 | 0.0810 0.0031 |
| *5* | 0.8878 0.0027 | 1.2301 0.0023 | 0.1010 0.0031 |
| → **Padrão 16: "Low wages and a .."** | | | |
| *0* | 0.5144 0.0033 | 0.4116 0.0033 | 0.0000 0.0000 |
| *1* | 0.5853 0.0029 | 0.5102 0.0032 | 0.0000 0.0000 |
| *2* | 0.6870 0.0028 | 0.6479 0.0023 | 0.0000 0.0000 |
| *3* | 0.7586 0.0026 | 0.8989 0.0026 | 0.0000 0.0000 |
| *4* | 0.8303 0.0024 | 1.0372 0.0025 | 0.0000 0.0000 |
| *5* | 0.9018 0.0025 | 1.1936 0.0028 | 0.0000 0.0000 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| → **Padrão 17: "Mr. Jordan said .."** | | | |
| *0* | 0.5457 0.0028 | 0.4257 0.0032 | 0.0000 0.0000 |
| *1* | 0.6166 0.0031 | 0.5222 0.0032 | 0.0000 0.0000 |
| *2* | 0.7183 0.0031 | 0.6561 0.0034 | 0.0000 0.0000 |
| *3* | 0.7903 0.0031 | 0.9012 0.0027 | 0.0000 0.0000 |
| *4* | 0.8649 0.0026 | 1.0473 0.0022 | 0.0000 0.0000 |
| *5* | 0.9472 0.0025 | 1.1967 0.0039 | 0.0000 0.0000 |

Tabela 22: Dados de tempo de execução e desvio padrão da execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88_20`.

## Para a Coleção Wsj88_10

| *Aproximação* | **SAA** | **PD** | **AGrep** |
|---|---|---|---|
| → **Padrão 1: "DCC"** | | | |
| *0* | 0.2556 0.1627 | 0.1852 0.0122 | 0.0340 0.0126 |
| *1* | 0.2693 0.0109 | 0.2510 0.0127 | 0.0840 0.0126 |
| *2* | 0.8392 0.0091 | 0.8158 0.0107 | 0.2560 0.0126 |
| → **Padrão 2: "That"** | | | |
| *0* | 0.2080 0.0106 | 0.1889 0.0108 | 0.0400 0.0000 |
| *1* | 0.3309 0.0103 | 0.3226 0.0109 | 0.1740 0.0096 |
| *2* | 1.4442 0.0902 | 1.4138 0.0051 | 0.5010 0.0031 |
| → **Padrão 3: "Stock"** | | | |
| *0* | 0.2069 0.0044 | 0.1872 0.0042 | 0.0320 0.0063 |
| *1* | 0.2628 0.0043 | 0.2612 0.0052 | 0.0810 0.0031 |
| *2* | 0.3646 0.0042 | 0.3886 0.0047 | 0.2140 0.0069 |
| → **Padrão 4: "Dollar"** | | | |
| *0* | 0.2044 0.0046 | 0.1833 0.0043 | 0.0300 0.0000 |
| *1* | 0.2477 0.0040 | 0.2410 0.0039 | 0.0510 0.0031 |
| *2* | 0.3197 0.0042 | 0.3322 0.0037 | 0.1010 0.0031 |
| *3* | 0.5041 0.0042 | 0.5799 0.0033 | 0.3600 0.0047 |
| → **Padrão 5: "Uberaba"** | | | |
| *0* | 0.2044 0.0039 | 0.1811 0.0006 | 0.0300 0.0000 |
| *1* | 0.2379 0.0012 | 0.2230 0.0041 | 0.0400 0.0000 |
| *2* | 0.2917 0.0054 | 0.3001 0.0039 | 0.0810 0.0031 |
| *3* | 0.3419 0.0041 | 0.4159 0.0044 | 0.2710 0.0031 |
| → **Padrão 6: "Exchange"** | | | |
| *0* | 0.2106 0.0039 | 0.1905 0.0038 | 0.0300 0.0000 |
| *1* | 0.2588 0.0043 | 0.2405 0.0042 | 0.0510 0.0031 |
| *2* | 0.3309 0.0041 | 0.3193 0.0044 | 0.1150 0.0070 |
| *3* | 0.4006 0.0047 | 0.4483 0.0040 | 0.1430 0.0067 |
| *4* | 0.5118 0.0044 | 0.5914 0.0041 | 0.3640 0.0069 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| → **Padrão 7: "Manacapuru"** | | | |
| *0* | 0.2073 0.0045 | 0.1854 0.0039 | 0.0200 0.0000 |
| *1* | 0.2429 0.0040 | 0.2410 0.0036 | 0.0300 0.0000 |
| *2* | 0.2936 0.0037 | 0.3134 0.0010 | 0.0430 0.0048 |
| *3* | 0.3295 0.0041 | 0.4111 0.0037 | 0.0810 0.0031 |
| *4* | 0.3657 0.0040 | 0.4770 0.0037 | 0.0910 0.0031 |
| *5* | 0.4100 0.0038 | 0.5367 0.0036 | 0.3420 0.0063 |
| → **Padrão 8: "Price Index"** | | | |
| *0* | 0.2089 0.0039 | 0.1848 0.0039 | 0.0210 0.0031 |
| *1* | 0.2444 0.0039 | 0.2347 0.0036 | 0.0400 0.0000 |
| *2* | 0.2960 0.0045 | 0.3106 0.0034 | 0.0600 0.0000 |
| *3* | 0.3332 0.0041 | 0.4222 0.0039 | 0.1210 0.0031 |
| *4* | 0.3742 0.0047 | 0.5114 0.0043 | 0.1610 0.0031 |
| *5* | 0.4390 0.0044 | 0.6257 0.0043 | 0.3810 0.0031 |
| → **Padrão 9: "Administration"** | | | |
| *0* | 0.2113 0.0043 | 0.1886 0.0043 | 0.0200 0.0000 |
| *1* | 0.2507 0.0042 | 0.2355 0.0041 | 0.0400 0.0000 |
| *2* | 0.3072 0.0040 | 0.3043 0.0043 | 0.0710 0.0031 |
| *3* | 0.3498 0.0042 | 0.4157 0.0040 | 0.0910 0.0031 |
| *4* | 0.3912 0.0006 | 0.4878 0.0035 | 0.2180 0.0063 |
| *5* | 0.4350 0.0041 | 0.5600 0.0045 | 0.2450 0.0070 |
| → **Padrão 10: "DCC UFMG dollar"** | | | |
| *0* | 0.2123 0.0039 | 0.1867 0.0040 | 0.0200 0.0000 |
| *1* | 0.2478 0.0039 | 0.2237 0.0011 | 0.0310 0.0031 |
| *2* | 0.2984 0.0042 | 0.2748 0.0041 | 0.0340 0.0051 |
| *3* | 0.3343 0.0045 | 0.3812 0.0041 | 0.0600 0.0000 |
| *4* | 0.3701 0.0045 | 0.4400 0.0048 | 0.0610 0.0031 |
| *5* | 0.4057 0.0041 | 0.5194 0.0044 | 0.1820 0.0063 |
| → **Padrão 11: "Brazilian coffee"** | | | |
| *0* | 0.2123 0.0047 | 0.1878 0.0047 | 0.0220 0.0063 |
| *1* | 0.2476 0.0045 | 0.2373 0.0041 | 0.0300 0.0000 |
| *2* | 0.2987 0.0039 | 0.3107 0.0034 | 0.0500 0.0000 |
| *3* | 0.3344 0.0038 | 0.4122 0.0039 | 0.0710 0.0031 |
| *4* | 0.3707 0.0036 | 0.4812 0.0036 | 0.0900 0.0000 |
| *5* | 0.4075 0.0040 | 0.5412 0.0040 | 0.2600 0.0047 |
| → **Padrão 12: "New York Stock .."** | | | |
| *0* | 0.2187 0.0042 | 0.1924 0.0006 | 0.0200 0.0000 |
| *1* | 0.2579 0.0044 | 0.2541 0.0037 | 0.0300 0.0000 |
| *2* | 0.3125 0.0041 | 0.3377 0.0041 | 0.0400 0.0000 |
| *3* | 0.3518 0.0042 | 0.4708 0.0042 | 0.0500 0.0000 |
| *4* | 0.3917 0.0045 | 0.5488 0.0042 | 0.0720 0.0063 |
| *5* | 0.4308 0.0045 | 0.6244 0.0038 | 0.0890 0.0031 |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| → **Padrão 13: "Macunaima admin .."** | | | |
| *0* | 0.2167 0.0048 | 0.1899 0.0047 | 0.0200 0.0000 |
| *1* | 0.2523 0.0041 | 0.2409 0.0039 | 0.0300 0.0000 |
| *2* | 0.3032 0.0041 | 0.3097 0.0035 | 0.0410 0.0031 |
| *3* | 0.3390 0.0040 | 0.4168 0.0037 | 0.0600 0.0000 |
| *4* | 0.3748 0.0039 | 0.4773 0.0035 | 0.0810 0.0031 |
| *5* | 0.4104 0.0037 | 0.5382 0.0033 | 0.1460 0.0069 |
| → **Padrão 14: "The agency sees .."** | | | |
| *0* | 0.2207 0.0045 | 0.1933 0.0044 | 0.0200 0.0000 |
| *1* | 0.2564 0.0043 | 0.2488 0.0040 | 0.0300 0.0000 |
| *2* | 0.3068 0.0043 | 0.3341 0.0039 | 0.0330 0.0048 |
| *3* | 0.3428 0.0041 | 0.4611 0.0036 | 0.0510 0.0031 |
| *4* | 0.3788 0.0043 | 0.5442 0.0039 | 0.0620 0.0063 |
| *5* | 0.4146 0.0046 | 0.6122 0.0042 | 0.0900 0.0000 |
| → **Padrão 15: "many investors .."** | | | |
| *0* | 0.2331 0.0048 | 0.2005 0.0043 | 0.0210 0.0031 |
| *1* | 0.2687 0.0044 | 0.2627 0.0040 | 0.0210 0.0031 |
| *2* | 0.3202 0.0037 | 0.3436 0.0038 | 0.0300 0.0000 |
| *3* | 0.3567 0.0045 | 0.4597 0.0037 | 0.0310 0.0031 |
| *4* | 0.3958 0.0039 | 0.5472 0.0041 | 0.0400 0.0000 |
| *5* | 0.4440 0.0038 | 0.6202 0.0047 | 0.0500 0.0000 |
| → **Padrão 16: "Low wages and a .."** | | | |
| *0* | 0.2619 0.0042 | 0.2100 0.0042 | 0.0000 0.0000 |
| *1* | 0.2961 0.0013 | 0.2592 0.0035 | 0.0000 0.0000 |
| *2* | 0.3481 0.0042 | 0.3278 0.0034 | 0.0000 0.0000 |
| *3* | 0.3837 0.0041 | 0.4533 0.0041 | 0.0000 0.0000 |
| *4* | 0.4197 0.0040 | 0.5227 0.0038 | 0.0000 0.0000 |
| *5* | 0.4555 0.0045 | 0.6012 0.0043 | 0.0000 0.0000 |
| → **Padrão 17: "Mr. Jordan said .."** | | | |
| *0* | 0.2777 0.0042 | 0.2171 0.0045 | 0.0000 0.0000 |
| *1* | 0.3130 0.0042 | 0.2652 0.0046 | 0.0000 0.0000 |
| *2* | 0.3639 0.0043 | 0.3317 0.0040 | 0.0000 0.0000 |
| *3* | 0.3999 0.0040 | 0.4542 0.0040 | 0.0000 0.0000 |
| *4* | 0.4374 0.0036 | 0.5276 0.0041 | 0.0000 0.0000 |
| *5* | 0.4787 0.0040 | 0.6025 0.0037 | 0.0000 0.0000 |

Tabela 23: Dados de tempo de execução e desvio padrão da execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj88_10`.

### Para a Coleção Wsj89

| *Aproximação* | **SAA** | **PD** | **AGrep** |
|---|---|---|---|
| → **Padrão 1: "DCC"** | | | |
| *Continua na próxima página ..* | | | |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *0* | 0.0707 0.0228 | 0.0565 0.0004 | 0.0110 0.0031 |
| *1* | 0.0820 0.0007 | 0.0812 0.0174 | 0.0200 0.0000 |
| *2* | 0.2655 0.0050 | 0.2594 0.0138 | 0.0710 0.0031 |
| → **Padrão 2: "That"** | | | |
| *0* | 0.0673 0.0106 | 0.0597 0.0092 | 0.0100 0.0000 |
| *1* | 0.1000 0.0124 | 0.0949 0.0089 | 0.0430 0.0094 |
| *2* | 0.3859 0.0110 | 0.3848 0.0096 | 0.1340 0.0096 |
| → **Padrão 3: "Stock"** | | | |
| *0* | 0.0645 0.0007 | 0.0606 0.0101 | 0.0100 0.0000 |
| *1* | 0.0832 0.0103 | 0.0803 0.0093 | 0.0200 0.0000 |
| *2* | 0.1075 0.0008 | 0.1152 0.0094 | 0.0630 0.0094 |
| → **Padrão 4: "Dollar"** | | | |
| *0* | 0.0670 0.0106 | 0.0560 0.0005 | 0.0100 0.0000 |
| *1* | 0.0788 0.0113 | 0.0722 0.0019 | 0.0100 0.0000 |
| *2* | 0.0974 0.0115 | 0.1004 0.0151 | 0.0300 0.0000 |
| *3* | 0.1444 0.0114 | 0.1643 0.0110 | 0.0900 0.0000 |
| → **Padrão 5: "Uberaba"** | | | |
| *0* | 0.0673 0.0116 | 0.0593 0.0107 | 0.0100 0.0000 |
| *1* | 0.0771 0.0110 | 0.0707 0.0109 | 0.0100 0.0000 |
| *2* | 0.0913 0.0111 | 0.0917 0.0110 | 0.0200 0.0000 |
| *3* | 0.1065 0.0113 | 0.1234 0.0112 | 0.0700 0.0000 |
| → **Padrão 6: "Exchange"** | | | |
| *0* | 0.0688 0.0114 | 0.0580 0.0005 | 0.0100 0.0000 |
| *1* | 0.0817 0.0114 | 0.0715 0.0005 | 0.0100 0.0000 |
| *2* | 0.1005 0.0114 | 0.0956 0.0114 | 0.0330 0.0094 |
| *3* | 0.1187 0.0116 | 0.1300 0.0106 | 0.0400 0.0000 |
| *4* | 0.1481 0.0112 | 0.1671 0.0105 | 0.1030 0.0094 |
| → **Padrão 7: "Manacapuru"** | | | |
| *0* | 0.0648 0.0004 | 0.0608 0.0121 | 0.0010 0.0031 |
| *1* | 0.0788 0.0106 | 0.0760 0.0108 | 0.0100 0.0000 |
| *2* | 0.0934 0.0107 | 0.0970 0.0108 | 0.0100 0.0000 |
| *3* | 0.1040 0.0108 | 0.1242 0.0105 | 0.0200 0.0000 |
| *4* | 0.1145 0.0113 | 0.1434 0.0106 | 0.0200 0.0000 |
| *5* | 0.1270 0.0108 | 0.1596 0.0107 | 0.0930 0.0094 |
| → **Padrão 8: "Price Index"** | | | |
| *0* | 0.0687 0.0106 | 0.0566 0.0005 | 0.0000 0.0000 |
| *1* | 0.0783 0.0107 | 0.0735 0.0105 | 0.0100 0.0000 |
| *2* | 0.0926 0.0108 | 0.0943 0.0106 | 0.0100 0.0000 |
| *3* | 0.1027 0.0113 | 0.1253 0.0109 | 0.0330 0.0094 |
| *4* | 0.1138 0.0111 | 0.1498 0.0109 | 0.0400 0.0000 |
| *5* | 0.1313 0.0114 | 0.1802 0.0112 | 0.1050 0.0126 |
| → **Padrão 9: "Administration"** | | | |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *0* | 0.0693 0.0111 | 0.0576 0.0003 | 0.0060 0.0051 |
| *1* | 0.0809 0.0114 | 0.0711 0.0004 | 0.0100 0.0000 |
| *2* | 0.0970 0.0109 | 0.0942 0.0105 | 0.0200 0.0000 |
| *3* | 0.1101 0.0108 | 0.1258 0.0112 | 0.0200 0.0000 |
| *4* | 0.1223 0.0108 | 0.1457 0.0107 | 0.0600 0.0000 |
| *5* | 0.1349 0.0110 | 0.1662 0.0104 | 0.0720 0.0103 |
| → **Padrão 10: "DCC UFMG dollar"** | | | |
| *0* | 0.0696 0.0105 | 0.0610 0.0106 | 0.0000 0.0000 |
| *1* | 0.0798 0.0107 | 0.0687 0.0007 | 0.0100 0.0000 |
| *2* | 0.0943 0.0105 | 0.0864 0.0106 | 0.0100 0.0000 |
| *3* | 0.1049 0.0109 | 0.1162 0.0106 | 0.0100 0.0000 |
| *4* | 0.1159 0.0110 | 0.1325 0.0109 | 0.0160 0.0051 |
| *5* | 0.1271 0.0113 | 0.1556 0.0110 | 0.0510 0.0031 |
| → **Padrão 11: "Brazilian coffee"** | | | |
| *0* | 0.0692 0.0100 | 0.0609 0.0113 | 0.0000 0.0000 |
| *1* | 0.0794 0.0113 | 0.0744 0.0109 | 0.0100 0.0000 |
| *2* | 0.0936 0.0110 | 0.0947 0.0110 | 0.0100 0.0000 |
| *3* | 0.1037 0.0115 | 0.1227 0.0109 | 0.0200 0.0000 |
| *4* | 0.1170 0.0150 | 0.1414 0.0112 | 0.0240 0.0126 |
| *5* | 0.1236 0.0117 | 0.1585 0.0110 | 0.0700 0.0115 |
| → **Padrão 12: "New York Stock .."** | | | |
| *0* | 0.0679 0.0005 | 0.0590 0.0004 | 0.0000 0.0000 |
| *1* | 0.0787 0.0005 | 0.0791 0.0113 | 0.0130 0.0094 |
| *2* | 0.0975 0.0116 | 0.1023 0.0108 | 0.0100 0.0000 |
| *3* | 0.1084 0.0113 | 0.1395 0.0106 | 0.0100 0.0000 |
| *4* | 0.1201 0.0106 | 0.1603 0.0105 | 0.0200 0.0000 |
| *5* | 0.1302 0.0111 | 0.1819 0.0108 | 0.0200 0.0000 |
| → **Padrão 13: "Macunaima admin .."** | | | |
| *0* | 0.0676 0.0007 | 0.0611 0.0095 | 0.0000 0.0000 |
| *1* | 0.0807 0.0107 | 0.0754 0.0106 | 0.0100 0.0000 |
| *2* | 0.0952 0.0111 | 0.0947 0.0107 | 0.0100 0.0000 |
| *3* | 0.1050 0.0109 | 0.1241 0.0107 | 0.0110 0.0031 |
| *4* | 0.1151 0.0107 | 0.1417 0.0107 | 0.0200 0.0000 |
| *5* | 0.1251 0.0105 | 0.1585 0.0106 | 0.0430 0.0094 |
| → **Padrão 14: "The agency sees .."** | | | |
| *0* | 0.0720 0.0107 | 0.0591 0.0004 | 0.0040 0.0126 |
| *1* | 0.0784 0.0005 | 0.0744 0.0006 | 0.0100 0.0000 |
| *2* | 0.0962 0.0112 | 0.1011 0.0107 | 0.0100 0.0000 |
| *3* | 0.1059 0.0111 | 0.1369 0.0111 | 0.0100 0.0000 |
| *4* | 0.1158 0.0108 | 0.1590 0.0105 | 0.0200 0.0000 |
| *5* | 0.1259 0.0116 | 0.1776 0.0110 | 0.0200 0.0000 |
| → **Padrão 15: "many investors .."** | | | |
| | | | *Continua na próxima página ..* |

| *Continuação da página anterior ..* | | | |
|---|---|---|---|
| *Aproximação* | **SAA** | **PD** | **AGrep** |
| *0* | 0.0754 0.0097 | 0.0645 0.0109 | 0.0000 0.0000 |
| *1* | 0.0858 0.0110 | 0.0817 0.0104 | 0.0030 0.0048 |
| *2* | 0.1010 0.0108 | 0.1038 0.0109 | 0.0090 0.0031 |
| *3* | 0.1106 0.0111 | 0.1365 0.0111 | 0.0100 0.0000 |
| *4* | 0.1215 0.0108 | 0.1607 0.0109 | 0.0100 0.0000 |
| *5* | 0.1361 0.0108 | 0.1801 0.0105 | 0.0100 0.0000 |
| → **Padrão 16: "Low wages and a .."** | | | |
| *0* | 0.0845 0.0095 | 0.0676 0.0104 | 0.0000 0.0000 |
| *1* | 0.0946 0.0107 | 0.0813 0.0105 | 0.0000 0.0000 |
| *2* | 0.1089 0.0104 | 0.1013 0.0106 | 0.0000 0.0000 |
| *3* | 0.1187 0.0107 | 0.1315 0.0005 | 0.0000 0.0000 |
| *4* | 0.1285 0.0107 | 0.1550 0.0110 | 0.0000 0.0000 |
| *5* | 0.1386 0.0112 | 0.1759 0.0109 | 0.0000 0.0000 |
| → **Padrão 17: "Mr. Jordan said .."** | | | |
| *0* | 0.0898 0.0113 | 0.0663 0.0005 | 0.0000 0.0000 |
| *1* | 0.1004 0.0136 | 0.0832 0.0109 | 0.0000 0.0000 |
| *2* | 0.1138 0.0112 | 0.1016 0.0110 | 0.0000 0.0000 |
| *3* | 0.1238 0.0118 | 0.1355 0.0112 | 0.0000 0.0000 |
| *4* | 0.1337 0.0116 | 0.1556 0.0114 | 0.0000 0.0000 |
| *5* | 0.1455 0.0115 | 0.1769 0.0111 | 0.0000 0.0000 |

Tabela 24: Dados de tempo de execução e desvio padrão da execução dos algoritmos para cada um os padrões variando-se a aproximação e usando a coleção `Wsj89`.

## A.3 Comparações no BMH e BMHS

### Numero de Comparacoes por Padrão

| *Padrao* | Wsj89 | | Wsj88_10 | | Wsj88_20 | | Wsj88 | |
|---|---|---|---|---|---|---|---|---|
| | **BMHS** | **BMH** | **BMHS** | **BMH** | **BMHS** | **BMH** | **BMHS** | **BMH** |
| *1.DCC* | 702179 | 924994 | 2521191 | 3324321 | 5044578 | 6652771 | 27222030 | 35893551 |
| *2.That* | 695371 | 778383 | 2522768 | 2825337 | 5046149 | 5653712 | 27217402 | 30492533 |
| *3.Stock* | 496691 | 594600 | 1794942 | 2156705 | 3593515 | 4317829 | 19391841 | 23298433 |
| *4.Dollar* | 488271 | 526370 | 1772677 | 1906930 | 3548096 | 3817368 | 19163950 | 20610943 |
| *5.Uberaba* | 422764 | 459240 | 1535332 | 1668202 | 3070499 | 3335084 | 16582204 | 18001661 |
| *6.Exchange* | 421669 | 410066 | 1541556 | 1492614 | 3079497 | 2983855 | 16641231 | 16112353 |
| *7.Manacapuru* | 284185 | 309156 | 1034161 | 1123498 | 2068549 | 2251295 | 11160998 | 12135457 |
| *8.Price Index* | 286181 | 321363 | 1046856 | 1176287 | 2094251 | 2349838 | 11293725 | 12682289 |
| *9.Administration* | 284523 | 285453 | 1041257 | 1042324 | 2081609 | 2086456 | 11238370 | 11256751 |
| *10.DCC UFMG dollar* | 244614 | 236692 | 895481 | 868063 | 1793720 | 1737392 | 9686361 | 9385874 |
| *11.Brazilian coffee* | 261477 | 263406 | 965136 | 968735 | 1927209 | 1936213 | 10401293 | 10447862 |
| *12.New York Stock ..* | 207020 | 182135 | 765560 | 679343 | 1532605 | 1356482 | 8285835 | 7338881 |
| *Continua na próxima página ..* | | | | | | | | |

| *Continuação da página anterior ..* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Padrao* | **Wsj89** | | **Wsj88_10** | | **Wsj88_20** | | **Wsj88** | |
| | **BMHS** | **BMH** | **BMHS** | **BMH** | **BMHS** | **BMH** | **BMHS** | **BMH** |
| *13.Macunaima ..* | 196461 | 192050 | 733188 | 717990 | 1463515 | 1433901 | 7906066 | 7749395 |
| *14.The agency ..* | 170367 | 164529 | 645103 | 623834 | 1285118 | 1246068 | 6945024 | 6729275 |
| *15.many investors ..* | 114503 | 114124 | 441777 | 437989 | 881063 | 879129 | 4751014 | 4736606 |
| *16.Low wages and ..* | 54058 | 54281 | 206051 | 207732 | 412439 | 416735 | 2217533 | 2239220 |
| *17.Mr. Jordan ..* | 42969 | 38600 | 163441 | 148425 | 327986 | 299207 | 1765192 | 1602195 |

Tabela 25: Dados de comparações realizadas pelo BMHS e pelo BMH para cada padrão padrão utilizando todas as coleções.

### Numero de Comparacoes por Tamanho do Texto

| *Aproximação* | **BMHS** | **BMH** |
|---|---|---|
| → **Padrão 1: "DCC"** | | |
| *2.8* | 702179 | 924994 |
| *10.0* | 2521191 | 3324321 |
| *20.0* | 5044578 | 6652771 |
| *100.0* | 27222030 | 35893551 |
| → **Padrão 2: "That"** | | |
| *2.8* | 695371 | 778383 |
| *10.0* | 2522768 | 2825337 |
| *20.0* | 5046149 | 5653712 |
| *100.0* | 27217402 | 30492533 |
| → **Padrão 3: "Stock"** | | |
| *2.8* | 496691 | 594600 |
| *10.0* | 1794942 | 2156705 |
| *20.0* | 3593515 | 4317829 |
| *100.0* | 19391841 | 23298433 |
| → **Padrão 4: "Dollar"** | | |
| *2.8* | 488271 | 526370 |
| *10.0* | 1772677 | 1906930 |
| *20.0* | 3548096 | 3817368 |
| *100.0* | 19163950 | 20610943 |
| → **Padrão 5: "Uberaba"** | | |
| *2.8* | 422764 | 459240 |
| *10.0* | 1535332 | 1668202 |
| *20.0* | 3070499 | 3335084 |
| *100.0* | 16582204 | 18001661 |
| → **Padrão 6: "Exchange"** | | |
| *2.8* | 421669 | 410066 |
| *10.0* | 1541556 | 1492614 |
| *Continua na próxima página ..* | | |

| *Continuação da página anterior ..* | | |
|---|---|---|
| *Aproximação* | **BMHS** | **BMH** |
| *20.0* | 3079497 | 2983855 |
| *100.0* | 16641231 | 16112353 |
| → **Padrão 7: "Manacapuru"** | | |
| *2.8* | 284185 | 309156 |
| *10.0* | 1034161 | 1123498 |
| *20.0* | 2068549 | 2251295 |
| *100.0* | 11160998 | 12135457 |
| → **Padrão 8: "Price Index"** | | |
| *2.8* | 286181 | 321363 |
| *10.0* | 1046856 | 1176287 |
| *20.0* | 2094251 | 2349838 |
| *100.0* | 11293725 | 12682289 |
| → **Padrão 9: "Administration"** | | |
| *2.8* | 284523 | 285453 |
| *10.0* | 1041257 | 1042324 |
| *20.0* | 2081609 | 2086456 |
| *100.0* | 11238370 | 11256751 |
| → **Padrão 10: "DCC UFMG dollar"** | | |
| *2.8* | 244614 | 236692 |
| *10.0* | 895481 | 868063 |
| *20.0* | 1793720 | 1737392 |
| *100.0* | 9686361 | 9385874 |
| → **Padrão 11: "Brazilian coffee"** | | |
| *2.8* | 261477 | 263406 |
| *10.0* | 965136 | 968735 |
| *20.0* | 1927209 | 1936213 |
| *100.0* | 10401293 | 10447862 |
| → **Padrão 12: "New York Stock .."** | | |
| *2.8* | 207023 | 182135 |
| *10.0* | 765560 | 679343 |
| *20.0* | 1532605 | 1356482 |
| *100.0* | 8285835 | 7338881 |
| → **Padrão 13: "Macunaima admin .."** | | |
| *2.8* | 196461 | 192050 |
| *10.0* | 733188 | 717990 |
| *20.0* | 1463515 | 1433901 |
| *100.0* | 7906066 | 7749395 |
| → **Padrão 14: "The agency sees .."** | | |
| *2.8* | 170367 | 164529 |
| *10.0* | 645103 | 623834 |
| *20.0* | 1285118 | 1246068 |
| *100.0* | 6945024 | 6729275 |
| *Continua na próxima página ..* | | |

| *Continuação da página anterior ..* | | |
|---|---|---|
| *Aproximação* | **BMHS** | **BMH** |
| → **Padrão 15: "many investors .."** | | |
| *2.8* | 114503 | 114124 |
| *10.0* | 441777 | 437989 |
| *20.0* | 881063 | 879129 |
| *100.0* | 4751014 | 4736606 |
| → **Padrão 16: "Low wages and a .."** | | |
| *2.8* | 54058 | 54281 |
| *10.0* | 206051 | 207732 |
| *20.0* | 412439 | 416735 |
| *100.0* | 2217533 | 2239220 |
| → **Padrão 17: "Mr. Jordan said .."** | | |
| *2.8* | 42969 | 38600 |
| *10.0* | 163441 | 148425 |
| *20.0* | 327986 | 299207 |
| *100.0* | 1765192 | 1602195 |

Tabela 26: Dados de número de comparações da execução dos algoritmos BMHS e BMH por tamanho de texto.

# B Códigos Fonte

## B.1 Implementação

### Arquivo makefile.h.

```
# Sistema Unix, compilador gnu
CC = /usr/bin/gcc

LIBS = -lm

# Diretorio raiz
T = .

#Diretorio dos arquivos .h
INCLUDE = $(T)/

# Opcoes de compilacao
OPTIONST = -c -g -O3 -Wall -I $(INCLUDE)

# Insira aqui o diretorio dos arquivos objetos sem a barra no final
obj_T = $(T)

# Nome do arquivo executavel
EXE=$(T)/PaaTP3.exe

# OBJs
OBJS=Arquivos.o BoyerMooreHorspool.o BoyerMooreHorspoolSunday.o Geral.o Main.o ProgDin.o ShiftAndAprox.o ShiftAnd.o

.c.o:
	$(CC) $(OPTIONST) $<

all:	$(OBJS)
	$(CC)  -o $(EXE)     $(obj_T)/*.o $(LIBS)
	rm -rf $(obj_T)/*.o *~

run:
	$(EXE)

clean:
	rm -rf $(obj_T)/*.o
	rm -rf $(EXE)
	rm -rf *~
```

### Arquivo Module.h.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/time.h>
```

```
#include <time.h>

#define _MODULE_H
#define _TIMEVAL

#define MAX_PARAM  127   // Tamanho Máximo de um Parametro da Linha de Comando
#define MAX_LINHA  1000  // Tamanho Maximo de uma Linha no Arquivo de Entrada
#define MAX_CHAR   256   // Inteiro Maximo que representa um Caractere
#define MAX_ERRORS 10    // Maximo de Erros na Aproximacao
#define DEBUG      0    // Nivel de Debug ( 0=Minimo )

// Definicao das Funcoes Principais
//   --> BoyerMooreHorspool.c
int BoyerMooreHorspool(char *linha, char *padrao, int numLinha, int compat);
//   --> BoyerMooreHorspoolSunday.c
int BoyerMooreHorspoolSunday(char *linha, char *padrao, int numLinha, int compat);
//   --> ShiftAnd.c
int ShiftAnd(char *linha, char *padrao, int numLinha, int compat);
//   --> ShiftAndAprox.c
int ShiftAndAprox(char *linha, char *padrao, int numLinha, int aprox);
//   --> ProgDin.c
int ProgDin(char *linha, char *padrao, int numLinha, int aprox);

// Definicao das Funcoes Auxiliares --> Geral.c
void printCasamento(char *linha, char *padrao, int numLinha, int posicao);
void printHelp();
void chomp(char *string);
int timevalSubtract(struct timeval *result, struct timeval *x,struct timeval *y);

// Definicao das Funcoes de Arquivo --> Arquivos.c
FILE *abreArqEntrada(char *nomeArqEntrada);
```

## Arquivo Main.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

int main(int argc, char *argv[]) {

  char algoritmo[MAX_PARAM];                // Parametro Algoritmo
  char nomeArqEntrada[MAX_PARAM];           // Parametro Arquivo de Entrada
  char padrao[MAX_PARAM];                   // Parametro Padrao
  int  aprox = 0;                           // Parametro Aproximacao
  FILE *arqEntrada;                         // Ponteiro para o Arquivo de Entrada
  char linha[MAX_LINHA];                    // Uma linha do Arquivo de Entrada
  int (*funcAlgoritmo)();                   // Ponteiro para a Funcao que sera executada para a Busca
  int  numLinha = 1;                        // Numero da Linha Atual
  int comp=0;

  #ifdef _TIMEVAL
  struct timeval inicioTotal, fim, tempoDec; // Tempos de Execucao
  #endif
```

```
  // Calculo do tempo de Execucao
  #ifdef _TIMEVAL
  gettimeofday(&inicioTotal, NULL);
  #endif

  if (DEBUG>0) printf("\n-= Casamento de Padrao com Busca Aproximada - PAA =-\n\n");

  // Verifica os Parametros de Entrada
  if ((argc!=7) && (argc!=9)) { printHelp(); return 0; }    // Faltam Parametros na Linha de Comando
  // Salva os parametros de entrada
  strcpy(algoritmo, argv[2]);                // Copia a entrada em argv[2] para string e o coloca em algoritmo
  strcpy(nomeArqEntrada, argv[4]);           // Copia a entrada em argv[4] para string e o coloca em nomeArqEntrada
  strcpy(padrao, argv[6]);                   // Copia a entrada em argv[6] para string e o coloca em padrao
  if (!strcmp(algoritmo, "SAA") || !strcmp(algoritmo, "PD"))
    sscanf(argv[8],"%d", &aprox);            // Converte a entrada em argv[8] de string para string e o coloca em aprox

  // Abre o Arquivo de Entrada
  arqEntrada = abreArqEntrada(nomeArqEntrada);
  if (arqEntrada == NULL) {
    printf("[Main] Erro no abreArqEntrada. Saindo.\n");
    return 0;
  }

  // Verifica qual funcao sera executada dependendo do Algoritmo escolhido
  if      (!strcmp(algoritmo, "BMH"))  funcAlgoritmo = BoyerMooreHorspool;
  else if (!strcmp(algoritmo, "BMHS")) funcAlgoritmo = BoyerMooreHorspoolSunday;
  else if (!strcmp(algoritmo, "SA"))   funcAlgoritmo = ShiftAnd;
  else if (!strcmp(algoritmo, "SAA"))  funcAlgoritmo = ShiftAndAprox;
  else if (!strcmp(algoritmo, "PD"))  funcAlgoritmo = ProgDin;
  else { printHelp(); return 0; }            // Algoritmo Invalido

  while (fgets(linha, sizeof(linha), arqEntrada)) {
    chomp(linha);
    comp += funcAlgoritmo(linha, padrao, numLinha, aprox);
    numLinha++;
  }

  // Imprime o tempo de Processamento do Programa se for no Unix
  #ifdef _TIMEVAL
  gettimeofday(&fim, NULL);
  timevalSubtract(&tempoDec, &fim, &inicioTotal);
  if (DEBUG>=0) printf("\n\n[Main] Tempo Final: %d.%06d\n", (int) tempoDec.tv_sec, (int) tempoDec.tv_usec);
  #endif

  if (comp!=0) printf("[Main] Comparacoes: %d\n", comp);

  // getchar();
  return 0;
}
```

## Arquivo BoyerMooreHorspool.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
```

```
#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao ShiftAnd.                                      *
 *    Faz o casamento de padrao utilizando o algoritmo *
 *    Boyer-Moore-Horspool.                             *
 *                                                       *
 *    @param linha String com a linha atual do arquivo.*
 *    @param padrao String com o padrao a ser buscado. *
 *    @param numLinha Numero da linha atual do arquivo.*
 *    @param compat Nao utilizado. Compatibilidade.    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
int BoyerMooreHorspool(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long i, j, k;
  long d[MAX_CHAR+1];
  int comp=0;

  for (j=0; j<=MAX_CHAR; j++) d[j] = m;
  for (j=1; j<m; j++) d[(int)padrao[j-1]] = m-j;
  i = m;
  while (i<=n) {
    k=i;
    j=m;
    while (linha[k-1] == padrao[j-1] && j>0) { k--; j--; }
    if (j==0) printCasamento(linha, padrao, numLinha, k+1);
    i+=d[(int)linha[i-1]];
  }

  return comp;
}
```

## Arquivo BoyerMooreHorspoolSunday.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada             *
 * --------- -- ------ --- ----- ----------             *
 *    Recuperacao de padroes utilizando:                *
 *       - Boyer-Moore-Horspool Exato                   *
 *       - Boyer-Moore-Horspool-Sunday Exato            *
 *       - Shift-And Exato                              *
 *       - Shift-And Aproximado                         *
 *       - Programacao Dinamica                         *
 *                                                       *
 *    Baseado na Referencia: Nivio Ziviane.             *
 *                                                       *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>           *
 *            http://www.dcc.ufmg.br/~horacio           *
 *                                                       *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao ShiftAnd.                                      *
 *    Faz o casamento de padrao utilizando o algoritmo *
 *    Boyer-Moore-Horspool-Sunday.                      *
 *                                                       *
 *    @param linha String com a linha atual do arquivo.*
 *    @param padrao String com o padrao a ser buscado. *
 *    @param numLinha Numero da linha atual do arquivo.*
 *    @param compat Nao utilizado. Compatibilidade.    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
int BoyerMooreHorspoolSunday(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long i, j, k;
  long d[MAX_CHAR+1];
```

```
  int comp=0;

  for (j=0; j<=MAX_CHAR; j++) d[j] = m + 1;
  for (j=1; j<=m; j++) d[(int)padrao[j-1]] = m - j + 1;
  i = m;
  while (i<=n) {
    k=i;
    j=m;
    while (linha[k-1] == padrao[j-1] && j>0) { k--; j--; }
    if (j==0) printCasamento(linha, padrao, numLinha, k+1);
    i+=d[(int)linha[i]];
  }

  return comp;
}
```

## Arquivo ShiftAnd.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao ShiftAnd.                                   *
 *    Faz o casamento de padrao utilizando o algoritmo *
 *    Shift-And.                                      *
 *                                                    *
 *    @param linha String com a linha atual do arquivo.*
 *    @param padrao String com o padrao a ser buscado. *
 *    @param numLinha Numero da linha atual do arquivo.*
 *    @param compat Nao utilizado. Compatibilidade.    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
int ShiftAnd(char *linha, char *padrao, int numLinha, int compat) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long Masc[MAX_CHAR];
  long i;
  long R = 0;

  if (DEBUG>8) printf("[ShiftAnd] Procurando por %s em %s\n", padrao, linha);

  for (i=0; i<MAX_CHAR; i++) Masc[i] = 0;
  for (i=1; i<=m; i++) Masc[padrao[i-1] + 127] |= 1 << (m-i);
  for (i=0; i<n; i++) {
    R = ((((unsigned long)R) >> 1) | (1 << (m-1))) & Masc[linha[i]+127];
    if ((R&1) != 0)
    printCasamento(linha, padrao, numLinha, i - m + 2);
  }

  return 0;
}
```

## Arquivo ShiftAndAprox.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao ShiftAnd.                                   *
 *    Faz o casamento de padrao utilizando o algoritmo *
 *    Shift-And Aproximado.                           *
 *                                                    *
 *    @param linha String com a linha atual do arquivo.*
 *    @param padrao String com o padrao a ser buscado. *
 *    @param numLinha Numero da linha atual do arquivo.*
 *    @param aprox Numero de erros permitidos.        *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
int ShiftAndAprox(char *linha, char *padrao, int numLinha, int aprox) {
  int n = strlen(linha);
  int m = strlen(padrao);
  long Masc[MAX_CHAR];
  long i, j, Ri, Rant, Rnovo;
  long R[MAX_ERRORS + 1];

  if (DEBUG>8) printf("[ShiftAndAprox] Procurando por %s em %s\n", padrao, linha);

  for (i=0; i<MAX_CHAR; i++) Masc[i] = 0;
  for (i=1; i<=m; i++) Masc[padrao[i-1] + 127] |= 1 << (m - i);
  R[0] = 0;
  Ri = 1 << (m - 1);
  for (j = 1; j<=aprox; j++) R[j] = (1 << (m-j)) | R[j-1];
  for (i=0; i<n; i++) {
    Rant = R[0];
    Rnovo = ((((unsigned long)Rant) >> 1) | Ri) & Masc[linha[i] + 127];
    R[0] = Rnovo;
    for (j = 1; j<=aprox; j++) {
      Rnovo = ((((unsigned long)R[j]) >> 1) & Masc[linha[i] + 127]) | Rant | (((unsigned long)(Rant | Rnovo)) >> 1);
      Rant = R[j];
      R[j] = Rnovo | Ri;
    }
    if ((Rnovo & 1) != 0) printCasamento(linha, padrao, numLinha, i-(strlen(padrao))+2);
  }

  return 0;
}
```

## Arquivo ProgDin.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ----------           *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
```

```
 *       - Shift-And Aproximado                   *
 *       - Programacao Dinamica                   *
 *                                                *
 *    Baseado na Referencia: Nivio Ziviane.       *
 *                                                *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>     *
 *            http://www.dcc.ufmg.br/~horacio     *
 *                                                *
 ** * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao ShiftAnd.                               *
 *    Faz o casamento de padrao utilizando o algoritmo *
 *    Shift-And.                                  *
 *                                                *
 *    @param linha String com a linha atual do arquivo.*
 *    @param padrao String com o padrao a ser buscado. *
 *    @param numLinha Numero da linha atual do arquivo.*
 *    @param aprox Numero de erros permitidos.    *
 ** * * * * * * * * * * * * * * * * * * * * * * * */
int ProgDin(char *linha, char *padrao, int numLinha, int aprox) {
  int n = strlen(linha);
  int m = strlen(padrao);

  int C[MAX_LINHA];
  int i, lact, pos, pC, nC;

  if (DEBUG>8) printf("[ProgDin] Procurando por %s em %s\n", padrao, linha);

  for (i=0; i<=m; i++) C[i]=i;
  lact = aprox + 1;

  for (pos=1; pos<=n; pos++) {
    pC = 0;
    nC = 0;
    for (i=1; i<=lact; i++) {
      if (padrao[i-1] == linha[pos-1]) nC = pC;
      else {
        if (pC < nC) nC = pC;
        if (C[i] < nC) nC = C[i];
        nC = nC + 1;
      }
      pC = C[i];
      C[i] = nC;
    }
    while (C[lact] > aprox) lact = lact-1;
    if (lact == m) printCasamento(linha, padrao, numLinha, pos-m+1);
    else lact = lact + 1;
  }

  return 0;
}
```

## Arquivo Geral.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada       *
 * --------- -- ------ --- ----- ----------       *
 *    Recuperacao de padroes utilizando:          *
 *       - Boyer-Moore-Horspool Exato             *
 *       - Boyer-Moore-Horspool-Sunday Exato      *
 *       - Shift-And Exato                        *
 *       - Shift-And Aproximado                   *
 *       - Programacao Dinamica                   *
 *                                                *
 *    Baseado na Referencia: Nivio Ziviane.       *
 *                                                *
```

```
 *    Horácio Fernandes <horacio@dcc.ufmg.br>          *
 *            http://www.dcc.ufmg.br/~horacio          *
 *                                                     *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"


/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao printCasamento.                              *
 *    Imprime a realizacao de m casamento.             *
 *                                                     *
 *    @param linha Linha onde teve o casamento.        *
 *    @param Padrao que fez o casamento.               *
 *    @param numLinha Numero da linha atual.           *
 *    @param posicao Numero da coluna onde casou.      *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
void printCasamento(char *linha, char *padrao, int numLinha, int posicao) {
  int i;
  printf("Linha %5d: %s\n", numLinha, linha);
  for (i=0; i<posicao+12; i++) printf(" ");
  for (i=0; i<strlen(padrao); i++) printf("^");
  printf("\n");
}

/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao printHelp.                                   *
 *    Imprime a Ajuda do Sistema.                      *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
void printHelp() {
    printf("      Utilizacao: ./HGrep <-a Algoritmo> <-f ArquivoEntrada> <-p Padrao> [-k Aproximacao] \n");
    printf("                                                             \n");
    printf("           Algoritmos: [BMH]  Boyer-Moore-Horspool           \n");
    printf("                       [BMHS] Boyer-Moore-Horspool-Sunday    \n");
    printf("                       [SA]   Shift-And                      \n");
    printf("                       [SAA]  Shift-And-Aproximado           \n");
    printf("                                                             \n");
    printf("           A ordem dos parametros deve ser considerada.      \n");
    printf("                                                             \n");
}

/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao chmop.                                       *
 *    Retira o retorno de linha do final de uma string *
 *    caso esse caractere exista. Diminui o  tamamanho *
 *    da string em 1.                                  *
 *                                                     *
 *    @param string Ponteiro para a string modificada. *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
void chomp(char *string) {
  if (string[strlen(string)-1]=='\n') string[strlen(string)-1] = string[strlen(string)];
}


/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao timevalSubtract                              *
 *    Subtrai  os  valores  struct  timeval  X   e   Y *
 *    armazenando o resultado em  RESULT. Retorna 1 se *
 *    a diferenca e negativa, 0 caso contrario.        *
 *                                                     *
 *    @param result Ponteiro onde ficara o resultado.  *
 *    @param x Tempo 1.                                *
 *    @param y Tempo 2.                                *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
int timevalSubtract (result, x, y)
     struct timeval *result, *x, *y;
{
  /* Perform the carry for the later subtraction by updating y. */
  if (x->tv_usec < y->tv_usec) {
    int nsec = (y->tv_usec - x->tv_usec) / 1000000 + 1;
    y->tv_usec -= 1000000 * nsec;
    y->tv_sec += nsec;
```

```
  }
  if (x->tv_usec - y->tv_usec > 1000000) {
    int nsec = (y->tv_usec - x->tv_usec) / 1000000;
    y->tv_usec += 1000000 * nsec;
    y->tv_sec -= nsec;
  }

  /* Compute the time remaining to wait.
     tv_usec is certainly positive. */
  result->tv_sec = x->tv_sec - y->tv_sec;
  result->tv_usec = x->tv_usec - y->tv_usec;

  /* Return 1 if result is negative. */
  return x->tv_sec < y->tv_sec;
}
```

### Arquivo Arquivos.c.

```
/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Casamento de Padrao com Busca Aproximada           *
 * --------- -- ------ --- ----- ---------            *
 *    Recuperacao de padroes utilizando:              *
 *       - Boyer-Moore-Horspool Exato                 *
 *       - Boyer-Moore-Horspool-Sunday Exato          *
 *       - Shift-And Exato                            *
 *       - Shift-And Aproximado                       *
 *       - Programacao Dinamica                       *
 *                                                    *
 *    Baseado na Referencia: Nivio Ziviane.           *
 *                                                    *
 *    Horácio Fernandes <horacio@dcc.ufmg.br>         *
 *            http://www.dcc.ufmg.br/~horacio         *
 *                                                    *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */

#include "Module.h"

/** * * * * * * * * * * * * * * * * * * * * * * * * * *
 * Funcao abreArqEntrada.                             *
 *    Abre o arquivo de entrada com os dados.         *
 *                                                    *
 *    @param nomeArqEntrada Nome do arquivo.          *
 *    @param arqEntrada  Ponteiro para o arquivo.     *
 *    @result Sucesso ou Erro.                        *
 ** * * * * * * * * * * * * * * * * * * * * * * * * * */
FILE *abreArqEntrada(char *nomeArqEntrada) {
  FILE *arqEntrada;

  if (DEBUG>4) printf("[abreArqEntrada] Abrindo Arquivo de Entrada %s .. ", nomeArqEntrada);

  if ((arqEntrada = fopen(nomeArqEntrada, "r")) == NULL) {
    if (DEBUG>4) printf("ERRO\n");
    return NULL;
  }

  if (DEBUG>4) printf("OK\n");

  return arqEntrada;
}
```

## B.2 Scripts de Análise

### Arquivo ExecutaGraf1.sh.

```
#!/bin/bash
```

```
DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"

HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="BMHS BMH SA SAA PD AGrep"

NumTestes="10"
Aprox="0"

echo "Gerando Arquivos de Dados para o Grafico 1 .."

# Para cada Arquivo de Colecoes ..
for Colecao in $Colecoes ; do

    echo "Gerando para a colecao $Colecao .."

    ArqSaidaGra="${DirSaida}DadosGraf1-$Colecao.dat"
    ArqSaidaDes="${DirSaida}DadosGraf1-$Colecao-Desvio.dat"
    ArqSaidaTex="${DirSaida}DadosGraf1-$Colecao-Tex.dat"

    echo "" > $ArqSaidaGra
    echo "" > $ArqSaidaDes
    echo "" > $ArqSaidaTex

    # Para cada Padrao
    cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

        Padrao=`echo "$PadraoLinha" | cut -f1`
        PadraoRes=`echo "$PadraoLinha" | cut -f2`

        echo "    Usando o padrao $PadraoRes"

        echo -n "$PadraoRes"             >> $ArqSaidaGra
        echo -n "$PadraoRes"             >> $ArqSaidaDes
        echo -n "            $PadraoRes" >> $ArqSaidaTex

        # Para cada Algoritmo
        for Algoritmo in $Algoritmos ; do

            echo -n "        --> $Algoritmo:"

            SomaTempos="0"
            Tempos=""
            # Executa n vezes
            for (( i=1 ; i<=$NumTestes ; i=$[ $i + 1 ] )) ; do
                if [ "$Algoritmo" != "AGrep" ] ; then
                    Tempo=`$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox |
                    --> grep "Tempo Final" | cut -d" " -f 4`
                else
                    time -o /tmp/temptime $AGrep -$Aprox "$Padrao" $DirEntrada$Colecao 2> /dev/null | tail -0
                    Tempo=`cat /tmp/temptime | grep elapsed | cut -d" " -f 3 | cut -c 3-7`
                fi
                Tempos="$Tempos $Tempo"
                SomaTempos=$(echo "$SomaTempos + $Tempo" | bc -l)
                echo -n " $Tempo"
            done

            # Calcula a Media
            TempoMedia=$(echo "scale=4; $SomaTempos / $NumTestes" | bc -l)
            if [ $(echo "$TempoMedia < 1" | bc -l) -eq 1 ] ; then TempoMedia="0$TempoMedia" ; fi
            if [ "$TempoMedia" == "0" ] ; then TempoMedia="0.0000" ; fi

            # Calcula o Desvio Padrao
            DesvSomatoria=0;
            for TempoTemp in $Tempos ; do
                DesvSomatoria=$(echo "$DesvSomatoria + ($TempoTemp - $TempoMedia)^2" | bc -l)
            done
```

```
            DesvTemp=$(echo "$DesvSomatoria / ($NumTestes - 1)" | bc -l)
            Desvio="0"$(echo "sqrt($DesvTemp)" | bc -l)
            if [ "$Desvio" == "00" ] ; then Desvio="0.0000" ; fi
            Desvio4=`echo $Desvio | cut -c1-6`

            # Salva os Resultados
            echo -n -e "\t$TempoMedia"           >> $ArqSaidaGra
            echo -n -e "\t$Desvio4"              >> $ArqSaidaDes
            echo -n -e " & $TempoMedia $Desvio4" >> $ArqSaidaTex

            echo " $TempoMedia $Desvio4"

        done

        echo ""            >> $ArqSaidaGra
        echo ""            >> $ArqSaidaDes
        echo " \\\\ \\hline" >> $ArqSaidaTex
    done
done
```

## Arquivo ExecutaGraf2.sh.

```
#!/bin/bash

DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"

HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="BMHS BMH SA SAA PD AGrep"

NumTestes="10"
Aprox="0"

echo "Gerando Arquivos de Dados para o Grafico 2 .."

# Para cada Padrao ..
PadraoNum="0"
cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

    Padrao=`echo "$PadraoLinha" | cut -f1`
    PadraoRes=`echo "$PadraoLinha" | cut -f2`
    PadraoNum=$[ $PadraoNum + 1 ]

    ArqSaidaGra="${DirSaida}DadosGraf2-$PadraoNum.dat"
    ArqSaidaDes="${DirSaida}DadosGraf2-$PadraoNum-Desvio.dat"
    ArqSaidaTex="${DirSaida}DadosGraf2-$PadraoNum-Tex.dat"

    echo "" > $ArqSaidaGra
    echo "" > $ArqSaidaDes
    echo "" > $ArqSaidaTex

    echo "Gerando para o padrao $PadraoNum - $PadraoRes .."

    # Para cada Colecao ..
    for Colecao in $Colecoes ; do

        if [ "$Colecao" == "wsj89" ]    ; then ColecaoTam="2.8"   ; ColecaoTam2="2.8  " ; fi
        if [ "$Colecao" == "wsj88_10" ] ; then ColecaoTam="10.0"  ; ColecaoTam2="10.0 " ; fi
        if [ "$Colecao" == "wsj88_20" ] ; then ColecaoTam="20.0"  ; ColecaoTam2="20.0 " ; fi
        if [ "$Colecao" == "wsj88" ]    ; then ColecaoTam="100.0" ; ColecaoTam2="100.0" ; fi

        echo "    Usando a colecao $Colecao ($ColecaoTam) .."

        echo -n "$ColecaoTam"              >> $ArqSaidaGra
        echo -n "$ColecaoTam"              >> $ArqSaidaDes
        echo -n "            $ColecaoTam2" >> $ArqSaidaTex
```

```
        # Para cada Algoritmo ..
        for Algoritmo in $Algoritmos ; do

            echo -n "        --> $Algoritmo: "

            # Executa 10 vezes.
            SomaTempos="0"
            Tempos=""
            for (( i=1 ; i<=$NumTestes ; i=$[ $i + 1 ] )) ; do

                if [ "$Algoritmo" != "AGrep" ] ; then
                    Tempo=`$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox
                    --> | grep "Tempo Final" | cut -d" " -f 4`
                else
                    time -o /tmp/temptime $AGrep -$Aprox "$Padrao" $DirEntrada$Colecao 2> /dev/null | tail -0
                    Tempo=`cat /tmp/temptime | grep elapsed | cut -d" " -f 3 | cut -c 3-7`
                fi
                Tempos="$Tempos $Tempo"
                SomaTempos=$(echo "$SomaTempos + $Tempo" | bc -l)
                echo -n " $Tempo"
            done

            # Calcula a Media
            TempoMedia=$(echo "scale=4; $SomaTempos / $NumTestes" | bc -l)
            if [ $(echo "$TempoMedia < 1" | bc -l) -eq 1 ] ; then TempoMedia="0$TempoMedia" ; fi
            if [ "$TempoMedia" == "0" ] ; then TempoMedia="0.0000" ; fi

            # Calcula o Desvio Padrao
            DesvSomatoria=0;
            for TempoTemp in $Tempos ; do
                DesvSomatoria=$(echo "$DesvSomatoria + ($TempoTemp - $TempoMedia)^2" | bc -l)
            done
            DesvTemp=$(echo "$DesvSomatoria / ($NumTestes - 1)" | bc -l)
            Desvio="0"$(echo "sqrt($DesvTemp)" | bc -l)
            if [ "$Desvio" == "00" ] ; then Desvio="0.0000" ; fi
            Desvio4=`echo $Desvio | cut -c1-6`

            # Salva os Resultados

            echo -n -e "\t$TempoMedia"           >> $ArqSaidaGra
            echo -n -e "\t$Desvio4"              >> $ArqSaidaDes
            echo -n -e " & $TempoMedia $Desvio4" >> $ArqSaidaTex

            echo " $TempoMedia $Desvio4"

        done

        echo ""           >> $ArqSaidaGra
        echo ""           >> $ArqSaidaDes
        echo " \\\\ \\hline" >> $ArqSaidaTex

    done
done
```

## Arquivo ExecutaGraf3.sh.

```
#!/bin/bash

DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"

HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="SAA PD AGrep"

NumTestes="10"
```

```
Aprox="0"

echo "Gerando Arquivos de Dados para o Grafico 4 .."

# Para cada Arquivo de Colecoes ..
for Colecao in $Colecoes ; do

    echo "Gerando para a colecao $Colecao .."

    # Para cada Algoritmo ..
    for Algoritmo in $Algoritmos ; do

        echo "    Usando o Algoritmo $Algoritmo .."

        ArqSaidaGra="${DirSaida}DadosGraf4-$Colecao-$Algoritmo.dat"
        ArqSaidaDes="${DirSaida}DadosGraf4-$Colecao-$Algoritmo-Desvio.dat"
        ArqSaidaTex="${DirSaida}DadosGraf4-$Colecao-$Algoritmo-Tex.dat"

        echo "" > $ArqSaidaGra
        echo "" > $ArqSaidaDes
        echo "" > $ArqSaidaTex

        # Para cada Padrao ..
        PadraoNum="0"
        cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

            Padrao=`echo "$PadraoLinha" | cut -f1`
            PadraoRes=`echo "$PadraoLinha" | cut -f2`
            PadraoNum=$[ $PadraoNum + 1 ]

            echo "        Usando o padrao $PadraoRes"

            echo -n "$PadraoRes"             >> $ArqSaidaGra
            echo -n "$PadraoRes"             >> $ArqSaidaDes
            echo -n "            $PadraoRes" >> $ArqSaidaTex

            # Para cada Aproximacao ..
            if [ $PadraoNum -eq 1 ] ; then AproxMax=2
            elif [ $PadraoNum -eq 2 ] ; then AproxMax=2
            elif [ $PadraoNum -eq 3 ] ; then AproxMax=2
            elif [ $PadraoNum -eq 4 ] ; then AproxMax=3
            elif [ $PadraoNum -eq 5 ] ; then AproxMax=3
            elif [ $PadraoNum -eq 6 ] ; then AproxMax=4
            else AproxMax=5 ; fi
            for (( Aprox=0 ; Aprox<=$AproxMax ; Aprox=$[ $Aprox + 1 ] )) ; do

                echo -n "            Aproximacao $Aprox/$AproxMax:"

                SomaTempos="0"
                Tempos=""
                # Executa n vezes
                for (( i=1 ; i<=$NumTestes ; i=$[ $i + 1 ] )) ; do
                    if [ "$Algoritmo" != "AGrep" ] ; then
                        Tempo=`$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox |
                        --> grep "Tempo Final" | cut -d" " -f 4`
                    else
                        time -o /tmp/temptime $AGrep -$Aprox "$Padrao" $DirEntrada$Colecao 2> /dev/null | tail -0
                        Tempo=`cat /tmp/temptime | grep elapsed | cut -d" " -f 3 | cut -c 3-7`
                    fi
                    Tempos="$Tempos $Tempo"
                    SomaTempos=$(echo "$SomaTempos + $Tempo" | bc -l)
                    echo -n " $Tempo"
                done

                # Calcula a Media
                TempoMedia=$(echo "scale=4; $SomaTempos / $NumTestes" | bc -l)
                if [ $(echo "$TempoMedia < 1" | bc -l) -eq 1 ] ; then TempoMedia="0$TempoMedia" ; fi
                if [ "$TempoMedia" == "0" ] ; then TempoMedia="0.0000" ; fi

                # Calcula o Desvio Padrao
                DesvSomatoria=0;
                for TempoTemp in $Tempos ; do
```

```
                DesvSomatoria=$(echo "$DesvSomatoria + ($TempoTemp - $TempoMedia)^2" | bc -l)
            done
            DesvTemp=$(echo "$DesvSomatoria / ($NumTestes - 1)" | bc -l)
            Desvio="0"$(echo "sqrt($DesvTemp)" | bc -l)
            if [ "$Desvio" == "00" ] ; then Desvio="0.0000" ; fi
            Desvio4=`echo $Desvio | cut -c1-6`

            # Salva os Resultados
            echo -n -e "\t$TempoMedia"           >> $ArqSaidaGra
            echo -n -e "\t$Desvio4"              >> $ArqSaidaDes
            echo -n -e " & $TempoMedia $Desvio4" >> $ArqSaidaTex

            echo " $TempoMedia $Desvio4"

        done

        echo ""           >> $ArqSaidaGra
        echo ""           >> $ArqSaidaDes
        echo  " \\\\ \\hline" >> $ArqSaidaTex

    done
  done
done
```

## Arquivo ExecutaGraf4.sh.

```
#!/bin/bash

DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"

HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="SAA PD AGrep"

NumTestes="10"
AproxMax="5"

echo "Gerando Arquivos de Dados para o Grafico 4 .."

# Para cada Colecao ..

for Colecao in $Colecoes ; do

    echo "Usando colecao $Colecao .."

    # Para cada Padrao ..
    PadraoNum="0"
    cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

        Padrao=`echo "$PadraoLinha" | cut -f1`
        PadraoRes=`echo "$PadraoLinha" | cut -f2`
        PadraoNum=$[ $PadraoNum + 1 ]

        ArqSaidaGra="${DirSaida}DadosGraf4-$Colecao-$PadraoNum.dat"
        ArqSaidaDes="${DirSaida}DadosGraf4-$Colecao-$PadraoNum-Desvio.dat"
        ArqSaidaTex="${DirSaida}DadosGraf4-$Colecao-$PadraoNum-Tex.dat"

        echo "" > $ArqSaidaGra
        echo "" > $ArqSaidaDes
        echo "" > $ArqSaidaTex

        echo "    Gerando para o padrao $PadraoNum - $PadraoRes .."

        # Para cada Aproximacao
        if [ $PadraoNum -eq 1 ] ; then AproxMax=2
        elif [ $PadraoNum -eq 2 ] ; then AproxMax=2
```

```
        elif [ $PadraoNum -eq 3 ] ; then AproxMax=2
        elif [ $PadraoNum -eq 4 ] ; then AproxMax=3
        elif [ $PadraoNum -eq 5 ] ; then AproxMax=3
        elif [ $PadraoNum -eq 6 ] ; then AproxMax=4
        else AproxMax=5 ; fi
        for (( Aprox=0 ; Aprox<=$AproxMax ; Aprox=$[ $Aprox + 1 ] )) ; do

            echo "        Usando a Aproximacao $Aprox / $AproxMax.."

            echo -n "$Aprox"              >> $ArqSaidaGra
            echo -n "$Aprox"              >> $ArqSaidaDes
            echo -n "            $Aprox" >> $ArqSaidaTex

            # Para cada Algoritmo ..
            for Algoritmo in $Algoritmos ; do

                echo -n "           --> $Algoritmo: "

                # Executa 10 vezes.
                SomaTempos="0"
                Tempos=""
                for (( i=1 ; i<=$NumTestes ; i=$[ $i + 1 ] )) ; do

                    if [ "$Algoritmo" != "AGrep" ] ; then
                        Tempo=‘$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox |
                        --> grep "Tempo Final" | cut -d" " -f 4‘
                    else
                        time -o /tmp/temptime $AGrep -$Aprox "$Padrao" $DirEntrada$Colecao 2> /dev/null | tail -0
                        Tempo=‘cat /tmp/temptime | grep elapsed | cut -d" " -f 3 | cut -c 3-7‘
                    fi
                    Tempos="$Tempos $Tempo"
                    SomaTempos=$(echo "$SomaTempos + $Tempo" | bc -l)
                    echo -n " $Tempo"
                done

                # Calcula a Media
                TempoMedia=$(echo "scale=4; $SomaTempos / $NumTestes" | bc -l)
                if [ $(echo "$TempoMedia < 1" | bc -l) -eq 1 ] ; then TempoMedia="0$TempoMedia" ; fi
                if [ "$TempoMedia" == "0" ] ; then TempoMedia="0.0000" ; fi

                # Calcula o Desvio Padrao
                DesvSomatoria=0;
                for TempoTemp in $Tempos ; do
                    DesvSomatoria=$(echo "$DesvSomatoria + ($TempoTemp - $TempoMedia)^2" | bc -l)
                done
                DesvTemp=$(echo "$DesvSomatoria / ($NumTestes - 1)" | bc -l)
                Desvio="0"$(echo "sqrt($DesvTemp)" | bc -l)
                if [ "$Desvio" == "00" ] ; then Desvio="0.0000" ; fi
                Desvio4=‘echo $Desvio | cut -c1-6‘

                # Salva os Resultados

                echo -n -e "\t$TempoMedia"           >> $ArqSaidaGra
                echo -n -e "\t$Desvio4"              >> $ArqSaidaDes
                echo -n -e " & $TempoMedia $Desvio4" >> $ArqSaidaTex

                echo " $TempoMedia $Desvio4"

            done

            echo ""           >> $ArqSaidaGra
            echo ""           >> $ArqSaidaDes
            echo " \\\\ \\hline" >> $ArqSaidaTex

        done
    done
done
```

## Arquivo ExecutaGraf5.sh.

```
#!/bin/bash

DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"

HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="BMHS BMH"

NumTestes="10"
Aprox="0"

echo "Gerando Arquivos de Dados para o Grafico 5 .."

# Para cada Arquivo de Colecoes ..
for Colecao in $Colecoes ; do

    echo "Gerando para a colecao $Colecao .."

    ArqSaidaGra="${DirSaida}DadosGraf5-$Colecao.dat"
    ArqSaidaTex="${DirSaida}DadosGraf5-$Colecao-Tex.dat"

    echo "" > $ArqSaidaGra
    echo "" > $ArqSaidaTex

    # Para cada Padrao
    cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

        Padrao=`echo "$PadraoLinha" | cut -f1`
        PadraoRes=`echo "$PadraoLinha" | cut -f2`

        echo "    Usando o padrao $PadraoRes"

        echo -n "$PadraoRes"             >> $ArqSaidaGra
        echo -n "            $PadraoRes" >> $ArqSaidaTex

        # Para cada Algoritmo
        for Algoritmo in $Algoritmos ; do

            echo -n "        --> $Algoritmo:"

            NumComp=`$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox |
            --> grep "Comparacoes" | cut -d" " -f 3`

            # Salva os Resultados
            echo -n -e "\t$NumComp"  >> $ArqSaidaGra
            echo -n -e " & $NumComp" >> $ArqSaidaTex

            echo " $NumComp"

        done

        echo ""           >> $ArqSaidaGra
        echo " \\\\ \\hline" >> $ArqSaidaTex

    done
done
```

## Arquivo ExecutaGraf6.sh.

```
#!/bin/bash

DirEntrada="../Entradas/"
DirSaida="Testes/Saidas/"
```

```
HGrep="./PaaTP3.exe"
AGrep="/usr/local/bin/agrep"

Colecoes="wsj89 wsj88_10 wsj88_20 wsj88"
ArqPadroes="wsjconsultas2.txt"
Algoritmos="BMHS BMH"

NumTestes="10"
Aprox="0"

echo "Gerando Arquivos de Dados para o Grafico 6 .."

# Para cada Padrao ..
PadraoNum="0"
cat $DirEntrada$ArqPadroes | while read PadraoLinha ; do

    Padrao=`echo "$PadraoLinha" | cut -f1`
    PadraoRes=`echo "$PadraoLinha" | cut -f2`
    PadraoNum=$[ $PadraoNum + 1 ]

    echo "Usando o padrao $PadraoRes"

    ArqSaidaGra="${DirSaida}DadosGraf6-$PadraoNum.dat"
    ArqSaidaTex="${DirSaida}DadosGraf6-$PadraoNum-Tex.dat"

    echo "" > $ArqSaidaGra
    echo "" > $ArqSaidaTex

    # Para cada Colecao ..
    for Colecao in $Colecoes ; do

        if [ "$Colecao" == "wsj89" ]    ; then ColecaoTam="2.8"   ; ColecaoTam2="2.8  " ; fi
        if [ "$Colecao" == "wsj88_10" ] ; then ColecaoTam="10.0"  ; ColecaoTam2="10.0 " ; fi
        if [ "$Colecao" == "wsj88_20" ] ; then ColecaoTam="20.0"  ; ColecaoTam2="20.0 " ; fi
        if [ "$Colecao" == "wsj88" ]    ; then ColecaoTam="100.0" ; ColecaoTam2="100.0" ; fi

        echo "    Usando a colecao $Colecao ($ColecaoTam) .."

        echo -n "$ColecaoTam"              >> $ArqSaidaGra
        echo -n "            $ColecaoTam2" >> $ArqSaidaTex

        # Para cada Algoritmo ..
        for Algoritmo in $Algoritmos ; do

            echo -n "        --> $Algoritmo: "

            NumComp=`$HGrep -a $Algoritmo -f $DirEntrada$Colecao -p "$Padrao" -k $Aprox |
            --> grep "Comparacoes" | cut -d" " -f 3`

            # Salva os Resultados
            echo -n -e "\t$NumComp"  >> $ArqSaidaGra
            echo -n -e " & $NumComp" >> $ArqSaidaTex

            echo " $NumComp"

        done

        echo ""           >> $ArqSaidaGra
        echo " \\\\ \\hline" >> $ArqSaidaTex

    done
done
```