Introdução

Em ciência da computação, um algoritmo que realiza casamento de cadeias, de forma geral, é um algoritmo projetado para encontrar uma determinada sequência de caracteres (denominada padrão) em um outro conjunto de caracteres (denominado texto). Nesse contexto, encontrar uma sequência de caracteres pode ter dois significados: a sequência encontrada é exatamente igual ou semelhante àquela procurada. Por este motivo, há dois tipos de algoritmos que realizam casamento de cadeias: algoritmos que realizam casamento exato de cadeias e algoritmos que realizam casamento aproximado de cadeias. O primeiro tipo só reconhece um casamento quando a sequência que representa o padrão existe, sem nenhuma diferença, no texto. Já o segundo tipo, além de reconhecer casamentos exatos, também considera como casamentos aquelas sequências do texto que sejam parecidas com o padrão. Nesse contexto, o termo parecida indica o quão diferente uma sequência encontrada no texto pode ser do padrão. Neste trabalho prático, todos os algoritmos analisados realizam casamento exato de cadeias.

Basicamente, algoritmos que realizam casamento de cadeias precisam percorrer o texto em busca do padrão. Quando a sequência de caracteres que representa o padrão é encontrada no texto, o algoritmo deve armazenar (ou exibir) a posição do texto onde o casamento ocorreu e prosseguir com as tentativas de casamento até que o texto termine. Eventuais diferenças entre o padrão e o texto devem ser consideradas, caso o algoritmo seja do tipo que realiza casamento aproximado de cadeias.

Todos os algoritmos implementados nesse trabalho prático são *case sensitive.* Isto é, consideram que letras maiúsculas e minúsculas são diferentes. A interpretação dos resultados obtidos deve levar essa característica em conta.

Este trabalho apresentará uma análise detalhada da implementação de cinco métodos de casamento exato de cadeias, vistos em sala de aula, bem como sua análise de desempenho.

Descrição do Funcionamento do Programa

Para atender as especificações deste trabalho prático, um algoritmo foi desenvolvido com um fluxo de execução simples.

A função principal (main) é a responsável por controlar o fluxo do programa e invocar os métodos de casamento exato de cadeias implementados.

Assim que a execução se inicia, a primeira tarefa do programa é realizar uma verificação dos argumentos informados via linha de comando, verificando se estes foram informados conforme a especificação do trabalho. Caso a validação não seja aprovada, uma mensagem de erro é retornada ao usuário, informando que os argumentos enviados ao programa são inválidos. Caso contrário, os parâmetros necessários à execução do programa são confiáveis e o programa pode prosseguir normalmente. Antes de identificar qual o método de casamento será realizado, e a fim de armazenar as posições do texto onde os casamentos ocorreram, uma lista duplamente encadeada é criada. A opção por uma lista deste tipo pode ser justificada pela redução da complexidade da operação que desaloca a memória alocada para essa estrutura de dados (de O(n^2) para O(n)). Após isso, a função principal se encarrega de realizar a chamada ao método de casamento de cadeias apropriado, com base no argumento que indica aquele que foi escolhido. O funcionamento detalhado de cada desses métodos será explicado a seguir.

Uma vez que o fluxo de execução do programa volte à função principal, ela verificará a necessidade de exibir os resultados solicitados pela especificação deste trabalho prático. Essa exibição, então, estará condicionada à presença do argumento -P durante a chamada do programa.

Antes de terminar, o programa precisa liberar a memória utilizada para a lista de ocorrências e a função *freeList()*, chamada pela função principal, se encarrega de tal tarefa.

Observação: Durante a fase de teste dos algoritmos, foi criada uma função que realiza a impressão dos padrões encontradas. Na versão final do código, essa função ainda existe, mas sua chamada está comentada.

Força Bruta

O método de casamento exato de cadeias conhecido como Força Bruta é, provavelmente, um dos mais intuivos algoritmos que realizam esse tipo de tarefa.

Inicialmente, o algoritmo tenta abrir o arquivo que representa o texto e, caso não consiga, o método termina sem realizar nenhuma outra instrução. Caso contrário, entra em cena a série de comparações efetuadas pela algoritmo. É importante ressaltar que o algoritmos faz uso de índices que o ajudam a controlar o acesso aos caracteres do texto e do padrão. A fim de realizar o casamento de cadeias, o Força Bruta verifica se o caractere do padrão (aquele que estiver na posição indicada pelo índice que controla o acesso ao padrão) casa com o caractere do texto (aquele que estiver na posição indicada pelo índice que controla o acesso ao texto). Caso os caracteres sejam iguais, ambos os índices avancam. Nesse momento, o algoritmo verifica se o padrão possui mais algum caractere a ser comparado com o texto. Em caso positivo, uma nova comparação é feita. Caso contrário, o algoritmo entende que um casamento foi realizado, o índice que controla o aecsso ao padrão é reiniciado, o índice que controla o acesso ao texto é incrementado, em uma unidade, e esse procedimento continua até que o fim do texto seja atingido.

A característica marcante desse algoritmo, que é justamente a que o define como força bruta, está relacionada à forma como ele trata situações de colisão. Nesse contexto, uma colisão refere-se ao momento em que um caractere do texto não casa com um caractere do padrão. Quando o algoritmo depara-se com tal situação, o número de caracteres deslocados à partir do ponto onde a colisão ocorreu é sempre 1. Essa característica faz com que a coomplexidade desse algoritmo seja O(n * m), em que n representa o tamanho do texto e m representa o tamanho do padrão, ambos em quantidade de caracteres.

Boyer-Moore (BM)

A fim de conseguir um desempenho melhor que o Força Bruta, a ideia principal desse algoritmo é fazer com que, no momento em que acontece uma colisão, a quantidade de caracteres deslocados seja maior que 1.

O funcionamento do algoritmo Boyer-Moore pode ser entendido como uma pesquisa do padrão em uma janela que desliza pelo texto. Para cada posição desta janela, o algoritmo pesquisa por um sufixo dela que casa com um sufixo do padrão, comparando da direita para esquerda. Essa comparação segue até que todo padrão seja casado com o texto, o que define um casamento realizado com sucesso, ou até que aconteça uma colisão, situação que indica que um caractere do texto não é igual ao padrão.

BM sugere duas abordagens, heurísticas para tratar colisões: ocorrência e casamento. Na heurística ocorrência, que é a heurística utilizada na implementação do algoritmo Boyer-Moore deste trabalho prático, o intuito é alinhar o caractere do texto que causou a colisão com o primeiro caractere (à esquerda do ponto de colisão) do padrão que casa com ele. Caso esse caractere exista no padrão, sua posição é identificada e a janela é deslocada de tal modo a atender à necessidade de alinhamento, o que pode fazer com que o deslocamento seja maior que 1 (isso depende da posição relativa do caractere encontrado no padrão). Caso contrário, a janela é deslocada em “comprimento do padrão” unidades e a comparação continua a partir desse novo ponto. Esse comportamento se repete até que o final do arquivo seja atingido.

Apesar de ser mais elaborado que o Força Bruta, a forma como BM trata colisões, durante o próprio processo de localização do padrão no texto, proporcionou que algumas melhorias fossem feitas a partir dele. Esses novos algoritmos serão apresentados logo abaixo.

Boyer-Moore-Horspool (BMH)

Em 1980, Horspool apresentou uma simplificação do algoritmo BM que o tornou mais eficiente. A essa nova implemtação do BM, damos o nome de Boyer-Moore-Horspool (BMH).

O princípio deste algoritmo é, diferentemente do Força Bruta e do BM, realizar um pré-processamento do padrão, com o objetivo de melhorar o tempo de pesquisa e aumentar o deslocamentos de janela de comparação.

Assim como o BM, que é sua base, o BMH tem como objetivo aumentar o tamanho dos deslocamentos que são realizados no processo de comparação. Acontece que, ao invés de basear na posição relativa do caractere que causou a colisão, que é, na verdade, uma busca por esse caracter, esse algoritmo constrói, antes da fase de comparação, uma tabela de deslocamentos. Para que o entendimento dessa tabela seja pleno, é necessário compreender o conceito de alfabeto. Um alfabeto refere-se ao conjunto de caracteres que um algoritmo de casamento de cadeias entende como válido. Como exemplo, um alfabeto pode ser o conjunto de símbolos que representam letras, números e sinais de pontuação que são usados num idioma qualquer. Na implementação do algoritmo BMH deste trabalho prático, o alfabeto utilizado é o ASC. Ou seja, a implementação do algoritmo BMH deste trabalho prático não aceita caracteres acentuados ou outros caracteres cuja representação decimal inteira esteja fora do intervalo [0; 127]. Usando o conceito de alfabeto, BMH realiza a construção da tabela de deslocamentos.

A criação da tabela de deslocamentos refere-se à fase conhecida como pré-processamento do padrão. Tal tabela é implementada como uma estrutura de dados que armazena o deslocamento que deverá ser utilizado para cada um dos caracteres do alfabeto. Então, quando uma colisão é detectada durante o processo de comparação, ao invés de realizar um processamento para descobrir o quanto a janela será deslocada, o algoritmo faz uma consulta a essa tabela. Neste trabalho prático, a tabela de deslocamentos foi implementada como um vetor. Por este motivo, o processo de identificar o tamanho do deslocamento possui complexidade O(1), pois não há mais necessidade da realização de uma consulta (o acesso é direto), o que é muito bom.

Nessa tabela, os valores dos deslocamentos são calculados da seguinte forma:

A variável que faz referência à estrutura de dados que representa a tabela é percorrida e, à cada posição, um valor é atribuído. Porém, o preenchimento da tabela é feito em dois momentos, cada um deles representado por um laço de repetição. No primeiro, todas as posições da tabela são preenchidas com um valor igual ao tamanho do padrão (m) a ser pesquisado. No próximo momento, as posições referentes aos m – 1 primeiros caracteres do padrão têm seu valor de deslocamento alterado. Essa alteração é baseada no cálculo que indica a distância de um determinado caractere do padrão até o último caractere do padrão.

Com a tabela de deslocamentos preenchida, o algoritmo parte para a fase de busca do padrão no texto, que acontece de forma muito parecida àquele vista no algoritmo anterior, BM. A única diferença aqui é que o BMH faz uso da tabela de ocorrências para determinar qual será o tamanho do deslocamento da janela de comparações ao invés de procurar, no padrão, pelo elemento que causou a colisão.

Boyer-Moore-Horspool-Sunday (BMHS)

Em 1990, Sunday propôs uma simplificação importante para o algoritmo BMH. A essa variante, damos o nome de Boyer-Moore-Horspool-Sunday (BMHS).

O funcionamento desse algoritmo é bastante semelhante ao anterior, BMH. A diferença na fase de pré-processamento do padrão. No BMHS, a intenção é fazer com que o valor do deslocamento de cada caracter esteja baseado no caractere do texto correspondente ao próximo caractere após o último caractere do padrão. Devido a essa mudança, o primeiro momento da construção da tabela de deslocamentos, deve iniciar o valor referente ao deslocamento de todos os caracteres com o valor m + 1, em que m representa o tamanho do padrão. Após isso, um novo laço percorre a tabela de deslocamentos, ajustando os valores de todos os caracteres que estão presentes no padrão (no BMH, apenas os m – 1 primeiros caracteres do padrão eram atualizados).

Shift-And Exato

Uma forma mais elegante de abordar o problema de casamento de cadeias consiste na aplicação do conceito paralelismo de bits, que permite que vários valores sejam “empacotados” em uma única palavra e atualizados por uma única operação.

O algoritmo Shift-And, proposto por Baeza-Yates e Gonnet, em 1989, é o algoritmo de casamento de cadeias que implementa essa ideia. Neste trabalho prático, o Shift-And implementado apenas realiza casamentos exatos.

Assim como BMH e BMHS, Shift-And-Exato realiza um pré-processamento do padrão, a fim de agilizar o procedimento de pesquisa. No caso desse algoritmo, pré-processar o padrão significa criar uma tabela formada por máscaras de bits. Uma máscara de bits é uma sequência de bits que representa o comprimento do padrão a ser pesquisado. Cada valor dessa sequência pode ser interpretado de duas formas: o valor 1 indica que o caractere está presente naquela posição, enquanto 0 indica o contrário. Para exemplificar o processo de criação de uma tabela de máscaras de bits, veja o exemplo abaixo.

- Padrão a ser pesquisado no texto: “dados”

Máscara de bits para o padrão “dados”

Caractere	Representação binária	Representação decimal
d	0 0 0 1 0 1 0 0	20
a	0 0 0 0 1 0 0 0	8
o	0 0 0 0 0 0 1 0	2
s	0 0 0 0 0 0 0 1	1

Cada uma das linhas da tabela acima refere-se a um caractere existente no padrão. Nesse exemplo, o tamanho da tabela foi, propositalmente, reduzido, a fim de limitar o espaço ocupado por ela no documento. Uma tabela de máscara de bits, na verdade, contém uma máscara de bits para cada um dos caracteres existentes no alfabeto considerado pelo algoritmo. Na tabela acima, os 5 caracteres que formam o padrão “dados” são representados pelos 5 últimos bits da sequência. Caso o valor seja de um bit da sequência seja igual a 1, o caractere está presente naquela posição do padrão, caso contrário, não está. O caracter “d” aparece duas vezes no padrão (posições 0 e 2), logo sua máscara possui dois bits 1, cada um em sua devida posição. Os demais caracteres do padrão ocorrem apenas 1 vez cada, fazendo com que sua máscara apresente apenas um bit ligado cada.

Para criar uma tabela com máscaras de bits, o algoritmo Shift-And-Exato faz uso de dois laços auxiliares. O primeiro deles é responsável por iniciar a máscara de cada um dos caracteres do alfabeto com o valor padrão (0), enquanto o segundo utiliza um cálculo, baseado na posição daquele caractere no padrão, para determinar a máscara de bits adequada para aquele caractere específico. Note que, assim como no exemplo – caractere “d”, caso haja caracteres iguais no padrão, a máscara de bits deste caractere armazenará um valor que seja suficiente para representar, de forma binária, que ele ocorre mais de uma vez no padrão.

Uma vez que todas as máscaras de bits tenham sido geradas, o algoritmo entra em sua próxima etapa, que é aquela que realiza o casamento do padrão com o texto.

De acordo com a própria concepção do algoritmo Shift-And-Exato, a tarefa que verifica casamentos entre os caracteres do padrão e do texto não realiza comparações, como acontece nos demais métodos analisados. Ao inveś disso, operações bit a bit (tais como |, &, << e >>) são realizadas com as variáveis que representam tais caracteres e, a partir do resultado dessas operações, o algoritmo determina se um casamento foi ou não realizado com sucesso dentro do texto.

Descrição da Metodologia de Análise

Após a etapa de implementação dos métodos de casamento exato de cadeias descritos acima, testes foram realizados, para que a eficiência da cada um deles pudesse ser constatada. Tais testes seguiram os seguintes critérios:

1. A fim de contemplar todos os testes solicitados na especificação desse trabalho prático, cada método de casamento exato de cadeias (exceto o Shift-And) foi executado 27 vezes. A quantidade de execuções é justificada pela seguinte fórmula: 3 tamanhos de texto diferentes * 3 tamanhos de padrão diferentes * 3 execuções para cada tamanho de padrão.
2. Para cada um dos tamanhos do padrão foram executados 3 testes: o primeiro utilizou um tamanho padrão mais próximo do limite inferior o intervalo, o segundo, um tamanho próximo ao valor médio do intervalo e, por fim, o terceiro, com o tamanho do padrão próximo ao limite superior do intervalo. Sendo assim, o resultado final apresentado é, na verdade, a média aritmética dos resultados obtidos após cada execução para um determinado tipo de padrão (pequeno, médio e grande), que representa um valor mais próximo daquele encontrado numa situação real de uso.
3. Conforme descrito anteriormente, os arquivos de entrada, que representam o texto onde o padrão será procurado, não devem possuir caracteres acentuados. Por esse motivo, textos na língua inglesa foram utilizados para todos os testes.
4. As palavras ou frases que representam o padrão foram mantidas entre um método e outro, facilitando assim a comparação de seus resultados.
5. A leitura do texto ocorre linha a linha, o que reduz a quantidade de acessos a disco e, consequentemente, torna o processo mais rápido. Originalmente, todas as linhas dos textos utilizados nos testes possuiam um comprimento igual ao inferior a 80 caracteres. Então, para que o teste pudesse prosseguir conforme o especificado, a quantidade de caracteres das linhas finais de cada arquivo foi alterada para, no mínimo, 120 caracteres.

6. Como a leitura do texto ocorre linha a linha, todos os algoritmos implementados evitam fazer comparações com linhas do texto que possuam menos caracteres que o padrão a ser pesquisado. Essa característica fez com que o número de comparações entre texto e padrão, quando o padrão utilizado é o de tamanho grande, fosse bem menor que aquele observado para os demais tamanhos de padrão.

7. A unidade de medida utilizada para mensurar o tempo gasto pelos métodos de casamento exato de cadeias foi o microsegundo(ms), com precisão de duas casas decimais.

8. Sobre os critérios analisados:

   - Número de comparações: Exibe o número de comparações entre caracteres do padrão e do texto realizadas pelo método de casamento exato de cadeias.

   - Deslocamentos (horizontal): Refere-se à quantidade de caracteres ignorados, de maneira segura, pelo método em questão. A lógica utilizada na concepção e construção de cada um dos algoritmos analisados é quem dita o comprimento do deslocamento.

   - Média de deslocamentos: Um valor que representa o número médio de caracteres ignorados em um deslocamento e é obtido através do seguinte cálculo: quantidade total de caracteres ignorados pelo processo de busca divida pela quantidade total de deslocamentos. Apesar de não fazer parte da especificação deste trabalho prático, acredito que essa informação ajude a interpretar os resultados obtidos.

   - Deslocamentos (vertical): Refere-se à quantidade de linhas ignoradas, de maneira segura, pelo método em questão. Apesar de não fazer parte da especificação deste trabalho prático, acredito que essa informação ajude a interpretar os resultados obtidos. Como a implementação dos métodos realiza a leitura linha a linha do arquivo, o processo de busca ignora aquelas que possuem um comprimento menor que o do padrão e como a maior parte das linhas dos textos utilizados possuem uma quantidade de caracteres igual ou inferior a 80, essa informação ajuda, especialmente, no entendimento relacionado ao baixo número de comparações realizadas quando o padrão é grande ([100;120]).

- Tempo de execução: Tempo total gasto pelo método analisado, excluindo-se apenas o tempo necessário para realizar a abertura/fechamento dos arquivo que representa o texto.

Resultados obtidos:

Força Bruta (Texto pequeno)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Pequeno	Pequeno
Número de comparações		23003
Deslocamentos (horizontal)		22947
Deslocamento médio		1
Deslocamentos (vertical)		46
Tempo de execução		0.14ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Pequeno	Médio
Número de comparações		22088
Deslocamentos (horizontal)		22061
Deslocamento médio		1
Deslocamentos (vertical)		75
Tempo de execução		0.16ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Pequeno	Grande
Número de comparações		299
Deslocamentos (horizontal)		299
Deslocamento médio		1
Deslocamentos (vertical)		357
Tempo de execução		0.04ms

Força Bruta (Texto médio)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Médio	Pequeno
Número de comparações		226309
Deslocamentos (horizontal)		225242
Deslocamento médio		1
Deslocamentos (vertical)		1093
Tempo de execução		1.54ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Médio	Médio
Número de comparações		209446
Deslocamentos (horizontal)		208129
Deslocamento médio		1
Deslocamentos (vertical)		1791
Tempo de execução		1.47ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Médio	Grande
Número de comparações		725
Deslocamentos (horizontal)		724
Deslocamento médio		1
Deslocamentos (vertical)		5108
Tempo de execução		0.32ms

Força Bruta (Texto grande)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Grande	Pequeno
Número de comparações		2344546
Deslocamentos (horizontal)		2305469
Deslocamento médio		1
Deslocamentos (vertical)		5237
Tempo de execução		13.90ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Grande	Médio
Número de comparações		2222052
Deslocamentos (horizontal)		2217106
Deslocamento médio		1
Deslocamentos (vertical)		8526
Tempo de execução		11.70ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Força Bruta	Grande	Grande
Número de comparações		129
Deslocamentos (horizontal)		129
Deslocamento médio		1
Deslocamentos (vertical)		40006
Tempo de execução		3.51ms

Análise dos resultados – Força Bruta

Número de comparações: Como o texto não possui um grande número de palavras semelhantes ao padrão pesquisado, a quantidade de comparações realizadas por esse método está bem próxima da quantidade de caracteres existentes no texto, exceto quando o padrão pesquisado no texto possui tamanho grande. Neste caso, o baixo número de comparações está relacionado à forma como o algoritmo foi implementado, descrita anteriormente.

Deslocamentos (horizontal): Neste método, a quantidade de deslocamentos na horizontal (em uma mesma linha) será igual ao número de caracteres existentes no texto subtraído da quantidade de caracteres existentes nas linhas ignoradas (deslocamento vertical).

Deslocamento médio: No caso do Força Bruta, a média de deslocamento sempre será igual a 1, pois é calculada com base na quantidade total de deslocamentos (critério acima) e no número total de caracteres ignorados (deslocamento vertical). Logo, temos uma divisão de termos de igual valor.

Deslocamentos (vertical): À medida que o tamanho do padrão a ser pesquisado aumenta, a tendência é que esse valor também cresça, que foi o comportamento observado.

Tempo de execução: O tempo de execução é dependente do tamanho do texto, do padrão e da quantidade de linhas onde o padrão foi, realmente, pesquisado (número de comparações). Devido à forma como o algoritmo foi implementado, o tempo cresce do pequeno para o médio, mas decresce entre o médio o grande. Além dos fatores citados acima, a medida do tempo de execução pode ser afetada pelo fato de que o computador precisa realizar, “simultaneamente”, vários programas e pelo sistema de memórias cache. Portanto, é difícil obter o tempo exato.

Boyer-Moore (Texto pequeno)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Pequeno	Pequeno
Número de comparações		22699
Deslocamentos (horizontal)		3577
Deslocamento médio		6.3
Deslocamentos (vertical)		2.7
Tempo de execução		0.22ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Pequeno	Médio
Número de comparações		15160
Deslocamentos (horizontal)		823
Deslocamento médio		17.5
Deslocamentos (vertical)		32
Tempo de execução		0.13ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Pequeno	Grande
Número de comparações		209
Deslocamentos (horizontal)		5
Deslocamento médio		45
Deslocamentos (vertical)		314
Tempo de execução		0.05ms

Boyer-Moore (Texto médio)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Médio	Pequeno
Número de comparações		218702
Deslocamentos (horizontal)		31910
Deslocamento médio		6.9
Deslocamentos (vertical)		45
Tempo de execução		1.84ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Médio	Médio
Número de comparações		131561
Deslocamentos (horizontal)		7408
Deslocamento médio		17
Deslocamentos (vertical)		743
Tempo de execução		0.87ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Médio	Grande
Número de comparações		440
Deslocamentos (horizontal)		10.7
Deslocamento médio		29.54
Deslocamentos (vertical)		4060
Tempo de execução		0.36ms

Boyer-Moore (Texto grande)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Grande	Pequeno
Número de comparações		2269929
Deslocamentos (horizontal)		337081
Deslocamento médio		6.6
Deslocamentos (vertical)		173.7
Tempo de execução		16.84ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Grande	Médio
Número de comparações		1468080
Deslocamentos (horizontal)		53616
Deslocamento médio		30.3
Deslocamentos (vertical)		3463.3
Tempo de execução		10.47ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BM	Grande	Grande
Número de comparações		110
Deslocamentos (horizontal)		0
Deslocamento médio		0
Deslocamentos (vertical)		34943
Tempo de execução		3.42ms

Análise dos resultados – Boyer-Moore

Número de comparações: Foi constatado que, a medida que o tamanho do texto e do padrão diminuem, o número de comparações também diminui. Isso pode ser explicado pelo aumento do deslocamento médio, cujo objetivo foi descrito anteriormente e a análise foi realizada logo abaixo. Como este método precisa verificar se o caractere causador da colisão faz parte do padrão, as comparações realizadas durante essa busca também foram consideradas na contagem e isso fez com que o número de comparações deste método fosse semelhante ao Força Bruta.

Deslocamentos (horizontal): Neste método, a quantidade de deslocamentos na horizontal (em uma mesma linha) diminui com o crescimento do comprimento do padrão, que é o comportamento esperado se considerarmos que um padrão de tamanho pequeno pode ser deslocado várias vezes dentro de uma mesma linha, o que não ocorre com padrões grandes.

Deslocamento médio: No caso do BM, deslocamentos são calculados com base no caractere que causou a colisão. Foi observado que, à medida que o padrão cresce, o tamanho médio do deslocamento também aumenta. Isso acontece porque é mais provável que colisões ocorram quando o padrão possui mais caracteres a serem comparados. Como cada caractere causador de uma colisão gera um deslocamento de tamanho diferente, as características do texto fizeram com que esse deslocamento médio fosse cada vez maior.

Deslocamentos (vertical): À medida que o tamanho do padrão a ser pesquisado aumenta, a tendência é que esse valor também cresça, que foi o comportamento observado.

Tempo de execução: O tempo de execução é dependente do tamanho do texto, do padrão e da quantidade de linhas onde o padrão foi, realmente, pesquisado (número de comparações). Devido à forma como o algoritmo foi implementado, o tempo cresce do pequeno para o médio, mas decresce entre o médio o grande. Além dos fatores citados acima, a medida do

tempo de execução pode ser afetada pelo fato de que o computador precisa realizar, “simultaneamente”, vários programas e pelo sistema de memórias cache. Portanto, é difícil obter o tempo exato.

Boyer-Moore-Horspool (Texto pequeno)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Pequeno	Pequeno
Número de comparações		4426
Deslocamentos (horizontal)		3569.7
Deslocamento médio		6.3
Deslocamentos (vertical)		2.7
Tempo de execução		0.14ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Pequeno	Médio
Número de comparações		1152
Deslocamentos (horizontal)		818.3
Deslocamento médio		17.7
Deslocamentos (vertical)		32
Tempo de execução		0.07ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Pequeno	Grande
Número de comparações		355
Deslocamentos (horizontal)		3
Deslocamento médio		59.6
Deslocamentos (vertical)		314
Tempo de execução		0.07ms

Boyer-Moore-Horspool (Texto médio)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Médio	Pequeno
Número de comparações		34617
Deslocamentos (horizontal)		31868.7
Deslocamento médio		6.9
Deslocamentos (vertical)		45
Tempo de execução		0.88ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Médio	Médio
Número de comparações		7879
Deslocamentos (horizontal)		7376.3
Deslocamento médio		17
Deslocamentos (vertical)		743.3
Tempo de execução		0.49ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Médio	Grande
Número de comparações		361
Deslocamentos (horizontal)		10.3
Deslocamento médio		27.9
Deslocamentos (vertical)		4060
Tempo de execução		0.36ms

Boyer-Moore-Horspool (Texto grande)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Grande	Pequeno
Número de comparações		365659
Deslocamentos (horizontal)		336579
Deslocamento médio		6.6
Deslocamentos (vertical)		173.6
Tempo de execução		17.61ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Grande	Médio
Número de comparações		56048
Deslocamentos (horizontal)		53510.7
Deslocamento médio		30.2
Deslocamentos (vertical)		3463.3
Tempo de execução		4.20ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMH	Grande	Grande
Número de comparações		347
Deslocamentos (horizontal)		0
Deslocamento médio		0
Deslocamentos (vertical)		34943
Tempo de execução		3.24ms

Análise dos resultados – Boyer-Moore-Horspool

Número de comparações: Foi constatado que, a medida que o tamanho do texto e do padrão diminuem, o número de comparações também diminui. Isso pode ser explicado pelo aumento do deslocamento médio, cujo objetivo foi descrito anteriormente e a análise foi realizada logo abaixo. Como este método realiza um pré-processamento do padrão, a fim de montar a tabela de deslocamentos, não é necessário pesquisar, no padrão, pelo caractere causador da colisão. Portanto, quando comparado aos 2 métodos anteriores, é possível perceber um ganho considerável na quantidade de comparações realizadas por ele.

Deslocamentos (horizontal): Neste método, a quantidade de deslocamentos na horizontal (em uma mesma linha) diminui com o crescimento do comprimento do padrão, que é o comportamento esperado se considerarmos que um padrão de tamanho pequeno pode ser deslocado várias vezes dentro de uma mesma linha, o que não ocorre com padrões grandes.

Deslocamento médio: No caso do BMH, deslocamentos são calculados com base no caractere que causou a colisão. Foi observado que, à medida que o padrão cresce, o tamanho médio do deslocamento também aumenta. Isso acontece porque é mais provável que colisões ocorram quando o padrão possui mais caracteres a serem comparados. Como cada caractere causador de uma colisão gera um deslocamento de tamanho diferente, as características do texto fizeram com que esse deslocamento médio fosse cada vez maior.

Deslocamentos (vertical): À medida que o tamanho do padrão a ser pesquisado aumenta, a tendência é que esse valor também cresça, que foi o comportamento observado.

Tempo de execução: O tempo de execução é dependente do tamanho do texto, do padrão e da quantidade de linhas onde o padrão foi, realmente, pesquisado (número de comparações). Devido à forma como o algoritmo foi implementado, o tempo cresce do pequeno para o médio, mas decresce entre o médio o grande. Além dos fatores citados acima, a medida do

tempo de execução pode ser afetada pelo fato de que o computador precisa realizar, “simultaneamente”, vários programas e pelo sistema de memórias cache. Portanto, é difícil obter o tempo exato.

Boyer-Moore-Horspool-Sunday (Texto pequeno)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Pequeno	Pequeno
Número de comparações		5002
Deslocamentos (horizontal)		3877
Deslocamento médio		6.3
Deslocamentos (vertical)		2.7
Tempo de execução		0.15ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Pequeno	Médio
Número de comparações		1358
Deslocamentos (horizontal)		946.7
Deslocamento médio		15.8
Deslocamentos (vertical)		32
Tempo de execução		0.07ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Pequeno	Grande
Número de comparações		357
Deslocamentos (horizontal)		4.7
Deslocamento médio		28.8
Deslocamentos (vertical)		314
Tempo de execução		0.06ms

Boyer-Moore-Horspool-Sunday (Texto médio)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Médio	Pequeno
Número de comparações		36015
Deslocamentos (horizontal)		31849
Deslocamento médio		7
Deslocamentos (vertical)		45
Tempo de execução		0.92ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Médio	Médio
Número de comparações		8155
Deslocamentos (horizontal)		7423.3
Deslocamento médio		17.9
Deslocamentos (vertical)		743.3
Tempo de execução		0.57ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Médio	Grande
Número de comparações		359
Deslocamentos (horizontal)		9
Deslocamento médio		31.8
Deslocamentos (vertical)		4060
Tempo de execução		0.34ms

Boyer-Moore-Horspool-Sunday (Texto grande)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Grande	Pequeno
Número de comparações		381249
Deslocamentos (horizontal)		336989
Deslocamento médio		6.7
Deslocamentos (vertical)		173.7
Tempo de execução		8.72ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Grande	Médio
Número de comparações		61490
Deslocamentos (horizontal)		57027.3
Deslocamento médio		30.2
Deslocamentos (vertical)		3463.3
Tempo de execução		3.44ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
BMHS	Grande	Grande
Número de comparações		347
Deslocamentos (horizontal)		0
Deslocamento médio		0
Deslocamentos (vertical)		34943
Tempo de execução		4.11ms

Análise dos resultados – Boyer-Moore-Horspool-Sunday

Número de comparações: Foi constatado que, a medida que o tamanho do texto e do padrão diminuem, o número de comparações também diminui. Isso pode ser explicado pelo aumento do deslocamento médio, cujo objetivo foi descrito anteriormente e a análise foi realizada logo abaixo. Como este método realiza um pré-processamento do padrão, a fim de montar a tabela de deslocamentos, não é necessário pesquisar, no padrão, pelo caractere causador da colisão. Portanto, quando comparado aos 2 primeiros métodos, é possível perceber um ganho considerável na quantidade de comparações realizadas por ele.

Deslocamentos (horizontal): Neste método, a quantidade de deslocamentos na horizontal (em uma mesma linha) diminui com o crescimento do comprimento do padrão, que é o comportamento esperado se considerarmos que um padrão de tamanho pequeno pode ser deslocado várias vezes dentro de uma mesma linha, o que não ocorre com padrões grandes.

Deslocamento médio: No caso do BMHS, deslocamentos são calculados com base no caractere que causou a colisão. Foi observado que, à medida que o padrão cresce, o tamanho médio do deslocamento também aumenta. Isso acontece porque é mais provável que colisões ocorram quando o padrão possui mais caracteres a serem comparados. Como cada caractere causador de uma colisão gera um deslocamento de tamanho diferente, as características do texto fizeram com que esse deslocamento médio fosse cada vez maior.

Deslocamentos (vertical): À medida que o tamanho do padrão a ser pesquisado aumenta, a tendência é que esse valor também cresça, que foi o comportamento observado.

Tempo de execução: O tempo de execução é dependente do tamanho do texto, do padrão e da quantidade de linhas onde o padrão foi, realmente, pesquisado (número de comparações). Devido à forma como o algoritmo foi implementado, o tempo cresce do pequeno para o médio, mas decresce entre o médio o grande. Além dos fatores citados acima, a medida do

tempo de execução pode ser afetada pelo fato de que o computador precisa realizar, “simultaneamente”, vários programas e pelo sistema de memórias cache. Portanto, é difícil obter o tempo exato.

Shift-And Exato (Texto pequeno)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Pequeno	Pequeno
Número de comparações		22970
Deslocamentos (horizontal)		22834.7
Deslocamento médio		1
Deslocamentos (vertical)		45.7
Tempo de execução		0.20ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Pequeno	Médio
Número de comparações		22225
Deslocamentos (horizontal)		22053
Deslocamento médio		1
Deslocamentos (vertical)		75
Tempo de execução		0.23ms

Shift-And Exato (Texto médio)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Médio	Pequeno
Número de comparações		225133
Deslocamentos (horizontal)		224997
Deslocamento médio		1
Deslocamentos (vertical)		1093
Tempo de execução		1.96ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Médio	Médio
Número de comparações		208002
Deslocamentos (horizontal)		207830.3
Deslocamento médio		1
Deslocamentos (vertical)		1791.3
Tempo de execução		1.58ms

Shift-And Exato (Texto grande)

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Grande	Pequeno
Número de comparações		2301669
Deslocamentos (horizontal)		2301532.7
Deslocamento médio		1
Deslocamentos (vertical)		5236.7
Tempo de execução		19ms

Resultados		
Método	Tamanho do texto	Tamanho do padrão
Shift-And Exato	Grande	Médio
Número de comparações		2215139
Deslocamentos (horizontal)		2214967
Deslocamento médio		1
Deslocamentos (vertical)		8526.3
Tempo de execução		13.40ms

Análise dos resultados – Shift-And Exato

Número de comparações: Esse método de casamento exato de cadeias percorre todo o texto e, cada novo caractere lido, verifica se houve um casamento, até que a cadeia inteira seja encontrada. Devido a essa característica, o número de comparações realizadas por ele será igual ao número de caracteres existentes no texto subtraído da quantidade de caracteres existentes nas linhas ignoradas (deslocamento vertical).

Deslocamentos (horizontal): Neste método, a quantidade de deslocamentos na horizontal (em uma mesma linha) será igual ao número de caracteres existentes no texto subtraído da quantidade de caracteres existentes nas linhas ignoradas (deslocamento vertical).

Deslocamento médio: No caso do Shift-And Exato, a média de deslocamento sempre será igual a 1, pois é calculada com base na quantidade total de deslocamentos (critério acima) e no número total de caracteres ignorados (deslocamento horizontal). Logo, temos uma divisão de termos de igual valor.

Deslocamentos (vertical): À medida que o tamanho do padrão a ser pesquisado aumenta, a tendência é que esse valor também cresça, que foi o comportamento observado.

Tempo de execução: O tempo de execução é dependente do tamanho do texto, do padrão e da quantidade de linhas onde o padrão foi, realmente, pesquisado (número de comparações). Devido à forma como o algoritmo foi implementado, o tempo cresce do pequeno para o médio, mas decresce entre o médio o grande. Além dos fatores citados acima, a medida do tempo de execução pode ser afetada pelo fato de que o computador precisa realizar, “simultaneamente”, vários programas e pelo sistema de memórias cache. Portanto, é difícil obter o tempo exato.