Algoritmos de Casamento de Padrão

Alfredo Gabriel de Sousa Oliveira[1], **Kalil Saldanha Kaliffe**[1], **José Henrique da Silva**[1], **Marcus Maciel Oliveira**[1]

[1]Instituto de Ciências Exatas e Naturais - Faculdade de Computação
Universidade Federal do Pará (UFPA) - Belém, PA - Brasil

{alfredo.oliveira, jose.henrique.silva, kalil.kaliffe,

marcus.oliveira}@icen.ufpa.br

Abstract. *This article describes the pattern matching problem, computational problem which patterns must be found inside words, and describes four main algorithms to solve the problem, presenting their advantages and disadvantages. This work is made for the discipline of Data Structures II of the Bachelor's Degree in Computer Science, taught by teacher Reginaldo Cordeiro dos Santos Filho.*

Resumo. *Este artigo descreve o problema do casamento de cadeias, problema computacional em que padrões devem ser encontrados dentro de palavras, e descreve quatro algoritmos principais que resolvem o problema, apresentando suas vantagens e desvantagens principais. Este trabalho é feito para a disciplina de Projeto de Algoritmos II do curso de bacharelado da Ciência da Computação, ministrado pelo docente Reginaldo Cordeiro dos Santos Filho.*

1. Introdução

O algoritmo de casamento de cadeia de caracteres é uma classe de algoritmos que são usados encontrar subsequências de textos dentro de textos. Existem diversas aplicações para esse algoritmo, desde encontrar instâncias de certas palavras ou frases dentro de textos, até encontrar padrões dentro de sequências genéticas, o que requisita algoritmos eficientes para minimizar o tempo computacional e a quantidade de espaço necessário. Nesse sentido, vários autores da literatura construíram algoritmos ou aprimoraram algoritmos existentes objetivando aumentar a eficiência de algoritmos de casamento de padrões, e este trabalho visa analisar alguns desses algoritmos construídos, como o Força Bruta, variações do algoritmo de Booye-Moore e o algoritmo de Shift-AND exato e sua aproximação.

Nesse sentido, este trabalho será dividido nas seguintes seções: a Seção 2, que descreverá a metodologia de desenvolvimento para o trabalho, padronizando os textos, padrões e algoritmos utilizados para os testes; a Seção 3, que descreverá o problema do casamento de cadeias, e apresentará múltiplos algoritmos que resolvem o problema, discutindo o funcionamento, as vantagens, desvantagens e a complexidade no tempo e espaço assintótica; a Seção 4, que apresenta os resultados dos testes dos algoritmos abordados para os textos de diversos comprimentos e padrões definidos de forma prévia; e a Seção 5, que enfatiza as considerações finais obtidas no desenvolvimento do trabalho.

2. Metodologia de desenvolvimento adotada

O algoritmo de casamento de caracteres é um tipo de algoritmo que, dado uma sequência de texto denominado palavra e uma sequência de texto denominado padrão, determina se o padrão existe dentro do texto, considerando todas suas ocorrências. Contudo, nem todas as aplicações requerem a exatidão do padrão na palavra, o que faz com que esses tipos de algoritmos abranjam algoritmos exatos e aproximados.

Destarte, para o desenvolvimento deste trabalho serão implementados os algoritmos exatos de Força Bruta, Boole-Moyer, Boole-Moyer-Hoorspool, Shift-AND Exato; e os algoritmos aproximados de Shift-AND Aproximado. A linguagem de programação que será utilizada para implementação dos algoritmos será a linguagem C, e os textos que serão utilizados como base para os casos de testes foram o "Lorem Ipsum", com a sequência padrão "ipsum", e o "Lero Lero", com sequência padrão "convincentemente", com tamanhos de textos 500, 1000, 1500, 2000 e 5000, sendo executados 250 vezes cada e extraído a média de tempo de execução, em segundos.

3. Casamento de Cadeia

O problema do casamento de cadeias está repleto de algoritmos da literatura que possuem base fundamentação matemática sólida e que possuem nenhuma fundamentação matemática. Por causa disso, a criação de algoritmos eficientes para lidar com o problema se torna mais complexa, de tal forma que modificar o original é preferido na maioria das vezes. Todavia, os algoritmos abordados neste trabalho possuem uma mesclagem dessas abordagens: possuem tanto algoritmos que utilizam uma nova abordagem de lidar com o problema, que modificam passos do algoritmo original ou que utilizam os dois de forma simultânea. Desse modo, a análise do funcionamento do algoritmo e de sua complexidade é imprescindível.

O algoritmo à ser utilizado depende, em geral, ao tamanho da palavra, ao tamanho do alfabeto e do quão próximo os padrões estão da palavra, uma vez que afetam consideravelmente a complexidade de tempo dos algoritmos para esse problema. [Skiena 1997]

3.1. Força Bruta

O algoritmo Força Bruta é um algoritmo de comparação de equidade de caracteres, e consiste em uma das abordagens mais simples para resolver o problema do casamento de cadeias: para cada caractere da palavra que tenha potencial de possuir o padrão em questão, comparar com cada caractere do padrão que se deseja encontrar, cessando a busca caso ocorra algum caractere em que os dois não são iguais, e voltando à comparação do caractere da palavra que tenha potencial de possuir o padrão em questão. Desse modo, necessita apenas fazer um laço de repetição para garantir que sejam percorridos o número de caracteres da palavra subtraído do número de caracteres do padrão, e outro laço para percorrer pelo número de caracteres do padrão, quebrando o laço de dentro no momento em que se identifica a não equidade de valores.

Esse algoritmo é um algoritmo consideravelmente ineficiente considerando outros algoritmos da literatura para o problema do casamento de cadeias, mas é demasiado simples de se entender, codificar e não consome espaço adicional extra para processamento. Possui complexidade de tempo médio de $O(nm)$, n sendo o tamanho da palavra e m sendo o tamanho do padrão, tendo em vista que executa dois laços de repetição.

3.2. Boyer-Moore

O algoritmo Boyer-Moore (BM) é um algoritmo de comparação de equidade de caracteres, similar ao força bruta, mas possui estratégias para ignorar caracteres que definitivamente não fazem parte do padrão. Possui fundamentação fundamenta na ideia de que percorrer o padrão da palavra do fim ao início produz informações mais relevantes quanto ao número de caracteres, e consiste em dois passos principais: a etapa de pré-processamento, que constrói uma primeira tabela com indexação referente aos caracteres do alfabeto, e uma segunda tabela referente ao valor do pulo para cada caractere no momento que ocorrer uma incompatibilidade no casamento; e a regra de *shift*, que define os valores do pulo do pré-processamento, e os valores consistem no maior valor entre as heurísticas de sufixos pré-encontrados (Good Suffix Rule), que constrói a tabela baseado nos sufixos que se repetem, e de mal caractere (Bad Character Rule), que constrói a tabela baseado no maior valor que é possível pular.

Esse algoritmo é um algoritmo ótimo para o problema do casamento de cadeias, com complexidade geral de espaço $O(m)$, m sendo o tamanho do padrão que se quer encontrar, apesar de ser um algoritmo consideravelmente complexo de se entender pela forma que se define os valores de pulo. Ademais, possui complexidade de tempo média de $O(\frac{n}{m})$, tendo em vista que as heurísticas permitem pular caracteres que não precisam ser visitados, em geral. O principal problema é o cenário de pior caso, o algoritmo pode possuir complexidade de O(nm), n sendo o tamanho da palavra e m o tamanho do padrão, e para textos assintoticamente grandes, essa complexidade é consideravelmente relevante. Os algoritmos Boyer-Moore-Horspool e Boyer-Moore-Horspool-Sunday visam lidar com uma das principais desvatangens do algoritmo de Boyer-Moore, como o fato de ser complexo, possuir heurísticas mal documentadas ou o pior caso, fazendo modificações que permitam o mesmo.

3.2.1. Boyer-Moore-Horspool

O algoritmo Boyer-Moore-Horspool (BMH) é uma simplificação do algoritmo de Boyer-Moore, em que ao invés de se utilizar duas tabelas para estimar o número de caracteres para serem pulados, se utiliza apenas uma, que é a tabela baseado na primeira ocorrência do caractere no padrão; todavia, foi no desenvolvimento desse algoritmo que houve uma otimização do pulo, em que se salvam as últimas ocorrências do padrão na tabela de mal caractere, e nomeia-se a mesma como tabela de *shift*, e uma melhor escrita quanto à construção de como fazer a tabela, uma vez que os autores visavam comprovar a eficiência do algoritmo de Boyer-Moore na literatura. [Horspool 1980]

Esse algoritmo continua sendo ótimo para o problema do casamento de padrões, com complexidade geral de espaço $O(m)$, m sendo o tamanho do padrão que se quer encontrar, e maior compreensibilidade e facilidade de implementação, pela simplificação que faz do algoritmo que é baseado em e da legibilidade do algoritmo. A complexidade de tempo média do algoritmo é de $O(\frac{n}{m})$ tendo em vista que a heurística que utiliza para criar a tabela de *shift* ainda o permite pular caracteres que não precisam ser visitado, e enquanto ainda possui o cenário de pior caso de complexidade de O(nm), n sendo o tamanho da palavra e m o tamanho do padrão, consegue ser uma abordagem eficiente na maioria dos casos.

3.2.2. Boyer-Moore-Horspool-Sunday

O algoritmo Boyer-Moore-Horspool-Sunday é uma simplificação do algoritmo Boyer-Moore-Horspool. Seu aperfeiçoamento consiste em alterar o modo de como é construído a tabela de *shift*; ao invés de a tabela registrar a última ocorrência do caractere no padrão, registra a primeira ocorrência do caractere no texto o qual corresponde, e quando ocorre um conflito de caractere entre a palavra e o padrão, invés de o valor ser o imediato correspondente na tabela de *shift*, é o imediato valor somado de um. O restante do funcionamento é o mesmo que o do algoritmo BMH.

Esse algoritmo é uma abordagem decente para o problema de casamento de cadeias de caracteres, com complexidade geral de espaço $O(m)$, m sendo o tamanho do padrão que se quer encontrar, e possui boa compreensibilidade e facilidade de implementação. A complexidade de tempo média do algoritmo é de $O(\frac{n}{m})$ tendo em vista que a heurística que utiliza para criar a tabela de *shift* ainda o permite pular caracteres que não precisam ser visitado, similar à abordagem do BH e do BMH, e possui pior caso de complexidade de O(nm), n sendo o tamanho da palavra e m o tamanho do padrão, mas por fazer saltos mais longos devido à abordagem que utiliza, tende a ser um pouco mais rápido que as abordagens anteriores.

3.3. Shift-AND Exato

O algoritmo Shift-And é um algoritmo de busca de padrão em um texto. Ele é chamado assim porque ele usa operações de deslocamento e conjunção lógica (AND) para comparar o padrão com o texto, diferente da abordagem convencional de comparar equidade de caracteres. Este algoritmo de busca de padrão é eficiente, pois ele é capaz de realizar a busca de um padrão em um texto em tempo linear, ou seja, em tempo proporcional ao tamanho do texto. Isso é possível porque o algoritmo Shift-And usa uma técnica de pré-processamento para criar uma tabela de pesquisa, que é usada para comparar o padrão com o texto de forma rápida e eficiente.

Para usar o algoritmo Shift-And, o primeiro passo é criar a tabela de pesquisa, que é uma matriz de bits. Cada coluna da tabela corresponde a um caractere do padrão e cada linha corresponde a um caractere do alfabeto. A tabela é preenchida de forma que, se o padrão contiver o caractere correspondente à linha, então o bit na coluna correspondente é definido como 1, caso contrário é definido como 0. Depois de criar a tabela de pesquisa, o algoritmo Shift-And pode ser usado para procurar o padrão no texto. Para isso, o algoritmo inicia com um conjunto de bits chamado ”filtro”, que é inicializado com todos os bits definidos como 1. Em seguida, o algoritmo lê cada caractere do texto, um de cada vez, e atualiza o filtro usando a tabela de pesquisa. Quando o filtro é atualizado, ele é deslocado para a direita e é conjugado com o valor da tabela de pesquisa correspondente ao caractere lido. Se o filtro for igual a 0 após a atualização, isso significa que o padrão não foi encontrado no texto. Se o filtro for igual a 1 após a atualização, isso significa que o padrão foi encontrado no texto e a posição do padrão no texto é armazenada.

Esse algoritmo é uma abordagem diferente para o problema de casamento de cadeias de caracteres, utilizando da operação binário do *shift* de bits, interagindo melhor com diversos tipos de hardware, manténdo a temporalidade do algoritmo, e é consideravelmente simples de compreensão. Possui complexidade geral de espaço $O(m)$ devido

ao pré-processamento, m sendo o tamanho do padrão que se quer encontrar, e a complexidade de tempo média do algoritmo é de $O(n)$, e possui pior caso de complexidade de $O(nm)$, n sendo o tamanho da palavra e m o tamanho do padrão.

3.4. Shift-AND Aproximado

O algoritmo Shift-And aproximado é uma variante do algoritmo Shift-And que é usada para realizar buscas aproximadas de um padrão em um texto. Na busca aproximada, o padrão é procurado no texto com tolerância a erros ou alterações, ou seja, o padrão pode ter algumas diferenças em relação ao que está sendo procurado no texto.

Desse modo, o funcionamento inicial do algoritmo Shift-AND aproximado é o mesmo do Shift-AND exato. Para isso, o algoritmo inicia com um conjunto de bits chamado "filtro", que é inicializado com todos os bits definidos como 1. Em seguida, o algoritmo lê cada caractere do texto, um de cada vez, e atualiza o filtro usando a tabela de pesquisa. Quando o filtro é atualizado, ele é deslocado para a direita e é conjugado com o valor da tabela de pesquisa correspondente ao caractere lido. Se o filtro for igual a 0 após a atualização, isso significa que o padrão não foi encontrado no texto. Se o filtro for igual a 1 após a atualização, isso significa que o padrão foi encontrado no texto e a posição do padrão no texto é armazenada.

O algoritmo Shift-And aproximado é eficiente para realizar buscas aproximadas de um padrão em um texto, pois ele é capaz de realizar a busca em tempo linear, ou seja, em tempo proporcional ao tamanho do texto. Isso é possível porque o algoritmo usa uma técnica de pré-processamento para criar a tabela.

Esse algoritmo é outra abordagem diferente para o problema de casamento de cadeias de caracteres, uma vez que não é um algoritmo de comparação exata de padrões. Todavia, por ser apenas uma modificação do Shift-AND original, possui complexidade geral de espaço $O(m)$, complexidade de tempo médio do algoritmo é de $O(n)$, e complexidade de tempo no pior caso $O(nm)$, uma vez que continua sendo em essência o mesmo algoritmo, apesar de ser menos estrito quanto aos tipos de cadeia de caracteres que são validados.

4. Cenário de Teste

Na Figura 1, há o gráfico que representa o comportamento assintótico dos algoritmos, utilizando como entrada textos gerados pelo Ipsun (A) e Lero Lero (B), como é possível observar, há pouca variação no comportamento dos algoritmos e os mesmos apresentam um comportamento esperado de acordo com a literatura.

5. Conclusão

Portanto, tendo em vista os fatos objetivados, é possível perceber que o problema de casamento de cadeias possuem diversos algoritmos que conseguem resolver o problema em até tempo polinomial, apesar de possuir trade-offs que podem ser fundamentais dependendo da aplicação em questão; como o tamanho do texto, o tamanho do alfabeto do padrão, a quantidade de acertos no padrão relativo à palavra, e a quantidade de espaço disponível no sistema em questão. Ademais, nem sempre a precisão é algo fundamental nesses tipos de problemas, aceitando uma margem de erro com algoritmos aproximados. A codificação dos algoritmos, bem como os cenários de teste estão disponíveis no repositório @hawkilol/pattern-matching-algos .

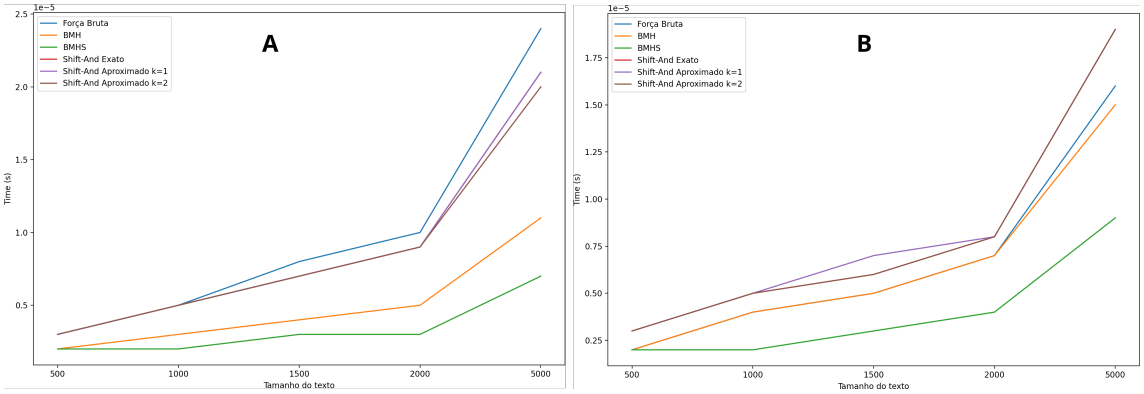

Figura 1. Média de tempo, em segundos, de 250 execuções de todos os algoritmos. Fonte: acervo próprio.

Referências

Horspool, N. (1980). Practical fast searching in strings. *Software: Practice and Experience*, 10(6):501–506.

Skiena, S. S. (1997). *The algorithm design manual*. Springer.