UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROJETO E ANÁLISE DE ALGORITMO

CASAMENTO EXATO DE PADRÕES DE CARACTERE

Diogo Augusto Martins Honorato
Rhayan de Sousa Barcelos

São João Del-Rei
2024

Sumário

1. INTRODUÇÃO

Este é um trabalho prático da disciplina de Projeto e Análise de Algoritmos no curso de Ciência da Computação na UFSJ, tendo como docente o professor Leonardo Rocha

1.1. Objetivo

Neste trabalho temos como objetivo assumir o papel de um jovem aprendiz de magia e auxiliar os sábios do reino encantado a encontrar a substring específica dentro de uma série de palavras encantadas.

Portanto, o objetivo é implementar e aplicar dois algoritmos de busca de padrão (casamento de string) que foram ensinados em aula. O sucesso depende da demonstração e compreensão de conceitos teóricos e práticos para solucionar o problema.

1.2. Motivação

Com base na descrição do problema, podemos concluir que se trata de uma questão relacionada a algoritmos de correspondência de padrões. Esses algoritmos possuem grande relevância no campo da computação, sendo amplamente utilizados nas áreas de busca e recuperação de informações. Essa aplicação tem um impacto direto na utilização da internet e na maneira em que nos comunicamos.

2. ABORDAGEM DO PROBLEMA

O primeiro passo é a compreensão do problema, que é realizar a busca de uma substring - um padrão - dentro de outra string (texto).

O segundo passo é a implementação dos algoritmos: no caso, foram escolhidos o “Boyer-Moore-Horspool-Sunday” e o “Shift-and”.

O terceiro passo é a aplicação nos dados fornecidos. Isto é, utilizando os algoritmos escolhidos, processar os N pergaminhos - intervalos de palavras - e verificar se a substring está presente em alguma das palavras dentro dos intervalos especificados. Quando a substring for encontrada, a localização dentro da string deve ser revelada.

O quarto passo é a manipulação de Strings, envolvendo, comparando, manipulando e deslocando conforme ditado pelos algoritmos. A eficiência depende da aplicação nos dados fornecidos.

2.1. Boyer-Moore-Horspool-Sunday

O BMHS começa com uma etapa de pré-processamento. Nesta fase, uma tabela de deslocamento, conhecida como tabela de Sunday, é construída com base na substring (o padrão). A tabela de Sunday difere por considerar o caractere imediatamente subsequente à janela de busca atual no texto. Esse caractere é utilizado para determinar quantas posições o padrão pode ser deslocado quando ocorre uma incompatibilidade durante a comparação.

Para construir essa tabela, o algoritmo percorre o padrão e associa cada caractere à sua última posição de ocorrência, utilizando essa informação para calcular os possíveis deslocamentos. Caso o caractere subsequente à janela de busca não esteja presente no padrão, o deslocamento é ajustado para o comprimento total do padrão mais uma posição. Isso permite que o padrão seja deslocado de maneira mais agressiva, aumentando a eficiência da busca.

Com a tabela de Sunday completa, o próximo passo envolve o alinhamento inicial do padrão com o início da primeira palavra a ser verificada. A busca é conduzida da direita para a esquerda, comparando-se os caracteres do padrão com os da palavra no texto. Esta abordagem permite que, ao encontrar uma

incompatibilidade, o algoritmo possa utilizar as informações da tabela de Sunday para decidir o deslocamento do padrão de maneira a maximizar a eficiência.

Se todos os caracteres do padrão coincidirem com a subsequente seção da palavra no texto, a substring é considerada encontrada e sua posição é registrada. Caso contrário, o algoritmo utiliza a tabela de Sunday para deslocar o padrão e repetir o processo de comparação até que a substring seja encontrada ou o texto seja completamente processado.

Como o problema exige a busca da substring em múltiplas palavras, contidas em diferentes intervalos de pergaminhos, o BMHS deve ser aplicado a cada palavra em cada intervalo separadamente. Para cada palavra, o algoritmo verifica a presença da substring utilizando a metodologia descrita. Quando a substring é encontrada, o algoritmo deve registrar a posição exata dentro da palavra onde a correspondência ocorre.

Se a substring não for encontrada em uma palavra ou em qualquer palavra de um intervalo, essa informação também deve ser armazenada, indicando que o padrão não está presente naquele segmento do texto.

Após a aplicação do BMHS em todos os intervalos de palavras fornecidos, o algoritmo deve retornar um conjunto de resultados que indicam as posições exatas da substring dentro das palavras onde foi encontrada.

2.2. Shift-and

A primeira etapa do Shift-And envolve o pré-processamento, em que uma tabela de bits é construída para representar o padrão (a substring). Nessa tabela, cada caractere do alfabeto é mapeado para uma máscara binária, onde cada bit indica a presença ou ausência do caractere correspondente em uma posição específica do padrão.

Por exemplo, se o padrão for "ABAB", a máscara para o caractere 'A' seria "0101", indicando que 'A' aparece nas posições 1 e 3 do padrão, enquanto a máscara para 'B' seria "1010", indicando que 'B' aparece nas posições 2 e 4. Essas

máscaras permitem que o algoritmo faça comparações rápidas e paralelas durante a busca.

Com a tabela de máscaras de bits preparada, o próximo passo é realizar a busca pelo padrão no texto. O Shift-And utiliza um registro de estado, representado por um valor binário, que é atualizado a cada iteração sobre os caracteres do texto.

O algoritmo começa alinhando o padrão com o início da primeira palavra a ser verificada. Para cada caractere do texto, o registro de estado é deslocado um bit à esquerda (daí o nome "Shift") e, em seguida, é realizada uma operação AND entre o registro de estado e a máscara binária correspondente ao caractere atual do texto.

Se, após essa operação, o bit mais significativo do registro de estado for 1, isso indica que o padrão foi encontrado começando naquela posição específica do texto. Nesse caso, o algoritmo registra a posição da correspondência e continua a busca no restante da palavra.

O problema envolve a busca pela substring em múltiplas palavras contidas em diferentes intervalos de pergaminhos. O Shift-And deve ser aplicado a cada palavra separadamente. Para cada palavra no intervalo, o algoritmo realiza a busca conforme descrito, utilizando o registro de estado para determinar se e onde a substring está presente.

Se o padrão for encontrado em uma palavra, a posição exata dentro da palavra é registrada. Se o padrão não for encontrado em uma palavra ou em qualquer palavra de um intervalo, essa informação também deve ser registrada.

Após aplicar o Shift-And a todas as palavras em todos os pergaminhos fornecidos, o algoritmo deve retornar os resultados que indicam as posições exatas da substring dentro das palavras onde foi encontrada. A precisão e a eficiência dessa abordagem dependem da correta implementação do Shift-And e da preparação adequada das máscaras de bits.

3. Execução do Programa

./tp3 <algoritmo> entrada.txt

Onde: <algoritmo> é B para o algoritmo BMHS e S para o algoritmo Shift-And.

O programa deve criar um arquivo saida.txt com o resultado.

Exemplo:

“./tp3 B entrada.txt”

4. ANÁLISE DE COMPLEXIDADE

4.1. Algoritmo Shift-And

O algoritmo Shift-And é baseado em operações bit a bit e é conhecido pela sua eficiência em padrões curtos. E funciona da seguinte forma: pré-processamento, cada caractere do padrão é mapeado para uma máscara de bits, onde cada posição do bit representa uma posição no padrão, este passo tem complexidade O(m), onde “m” é o tamanho do padrão.

Na pesquisa o texto é percorrido e a máscara de bits correspondente ao caractere atual do texto é combinada com o estado anterior usando operações de deslocamento e conjunção. A complexidade deste passo é O(n), onde “n” é o tamanho do texto, complexidade total, O(m+n).

4.2. Algoritmo BMHS (Boyer-Moore-Horspool-Sunday)

O BMHS é uma variação do algoritmo Boyer-Moore, que é mais eficiente na prática do que o algoritmo original para a maioria dos casos. Funciona da seguinte maneira, no pré-processamento é criada uma tabela de deslocamento para cada caractere do alfabeto, baseada na última aparição do caractere no padrão. A complexidade deste passo é O(m + T), onde “m” é o tamanho do padrão e T é o tamanho do alfabeto (geralmente constante, por exemplo, 256).

O algoritmo começa a comparação do padrão a partir do final em direção ao início e usa a tabela de deslocamento para pular comparações.

Em média, o algoritmo pula várias comparações, resultando em uma complexidade média de O(n / m). No pior caso, a complexidade pode chegar a O(m * n), embora isso seja raro.

O BMHS é geralmente mais eficiente para padrões maiores ou em textos muito longos, devido ao seu uso da tabela de deslocamento, que pode reduzir significativamente o número de comparações necessárias.

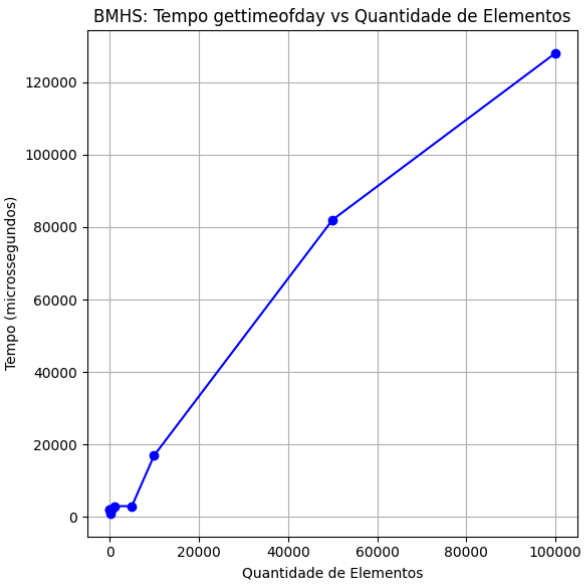
Shift-And sendo O(m + n), onde m é o tamanho do padrão e n é o tamanho do texto e o BMHS: O(m + T) para o pré processamento, com uma complexidade média de busca de O(n / m), mas no pior caso pode chegar a O(m * n).

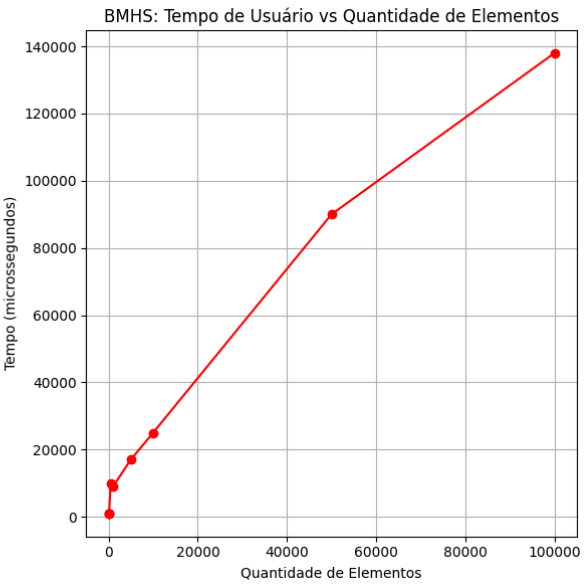
Portanto, o BMHS, em cenários práticos, geralmente será mais rápido devido à sua habilidade de pular várias comparações, enquanto o Shift-And mantém uma complexidade previsível e eficiente em cenários específicos, como padrões curtos.

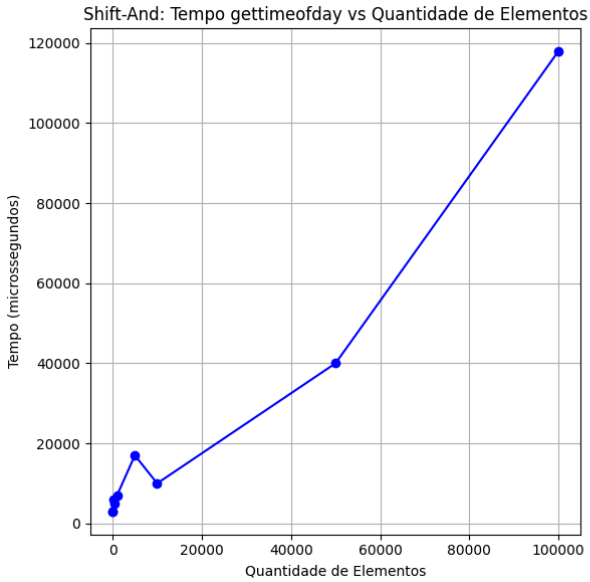
4.3. Análise da performance das soluções

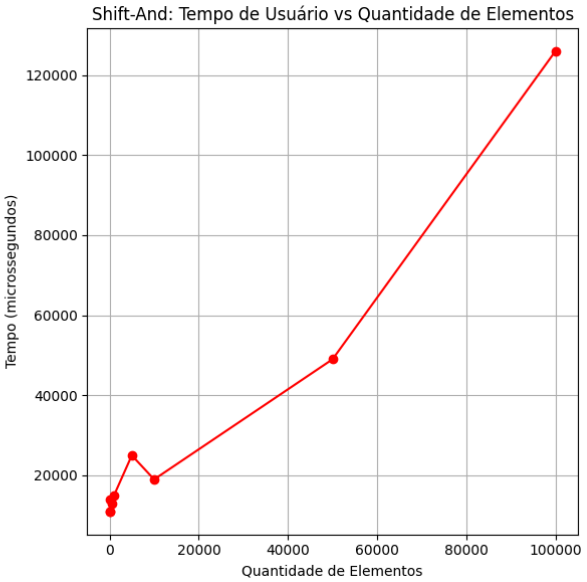
Diversos testes foram realizados com o objetivo de reunir resultados para comparação. Sendo obtidos através das funções “gettimeofday” e “getrusage”.

Os testes foram realizados com entradas com os seguintes valores: 10, 50, 100, 500, 1000, 5000, 10000, 50000 e 100000 elementos, substrings com 4 elementos e 10 queries. Os gráficos a seguir demonstram como cada solução se comportou:









Observando os resultados dos testes é possível concluir que em termos gerais, o BMHS tende a ser mais rápido na prática, especialmente com padrões longos e textos grandes, o Shift-And, embora como observado nos testes foi mais eficiente em textos curtos ou em padrões muito pequenos.

5. CONCLUSÃO

Primeiramente, é fundamental destacar a importância de uma análise cuidadosa das especificações iniciais do trabalho. Considerando as condições de

tamanho das entradas, foi possível desenvolver soluções que resolvem o problema proposto de maneira eficiente.

No contexto deste trabalho específico, observamos que o algoritmo BMHS apresentou o melhor desempenho ao ser avaliado com o uso das funções 'gettimeofday' e 'getrusage', sempre com uma margem pequena em relação ao Shift-and. Por esse motivo, consideramos tanto o BMHS quanto o Shift-and como as soluções recomendáveis para resolver o problema.

A escolha entre essas duas soluções é uma questão complexa. Enquanto o Shift-And é notoriamente estável e apresenta um pior caso melhor, o BMH tende a ter um caso esperado com menor tempo de execução. Portanto, ambas as soluções possuem pontos fortes e fracos e a escolha dependerá das prioridades e necessidades do usuário ao resolver o problema.

Concluímos então, enfatizando a importância de compreender o comportamento assintótico de um algoritmo e de definir uma solução que seja a melhor no geral, ou seja, rápida e adequada às exigências iniciais.

6. REFERÊNCIAS

[CORMEN et al., 2012] CORMEN, T. H., Leiserson, C., Rivest, R., and Stein, C (2012). Algoritmos: teoria e prática. LTC.