CCF330 - Projeto e Análise de Algoritmos
Casamento de Padrões

Germano Barcelos dos Santos (3873)[1], **Otávio Santos Gomes (3890)**[1]

[1]Instituto de Ciências Exatas e Tecnológicas, Campus UFV-Florestal
Universidade Federal de Viçosa

1. Introdução

Neste trabalho, foram implementados três algoritmos para o casamento de padrões. Os algoritmos Shift-And e o algoritmo Boyer-Moore foram utilizados para o casamento exato de padrões. Para casamento aproximado foi desenvolvido o algoritmo Shift-And Aproximado, capaz de lidar com operações de inserção, remoção e substituição.

O restante do trabalho está organizado da seguinte forma. A seção 2 apresenta as características da implementação dos algoritmos; a seção 3 apresenta a forma de utilizar a aplicação; a seção 4 apresenta os resultados obtidos com testes desenvolvidos; e a seção 5 apresenta as considerações finais.

2. Implementação

O programa desenvolvido é capaz de executar os algoritmos Shift-And e Boyer-Moore Exato, além do algoritmo Shift-And Aproximado. Para todos os algoritmos, é possível utilizar o modo DEBUG com o objetivo de visualizar o tempo de execução de cada um. [1] foi utilizado como fonte de consulta e entendimento maior acerca dos algoritmos.

Para a implementação do algoritmo Shift-And Exato foi utilizado como referência [3]. O Shift-And Exato usa um dicionário de letras, no caso do trabalho foi utilizado um vocabulário de 256 caracteres. Após a inicialização do dicionário, o padrão é procurado no texto a partir de operações bit a bit, por causa dessas operações a eficiência do algoritmo é grande. Para achar um padrão em um texto, basta uma iteração pelo texto e assim a complexidade assintótica é $O(n)$.

Para a implementação do algoritmo Boyer-Moore foi utilizado como referência [2]. Nesse algoritmo é preciso verificar se cada letra do padrão corresponde a letra atual do texto, caso esse teste não for verdadeiro para todo o padrão, o algoritmo desloca o padrão até que a letra que não correspondeu ao texto, corresponda. A partir desse deslocamento, o processo do teste recomeça no final do padrão e verifica novamente se cada letra é igual ao texto, nas respectivas posições. Pelo fato de ter que percorrer o texto e o padrão, o pior caso é $O(n + rm)$, sendo n o tamanho do texto, m o tamanho do padrão e r a quantidade de casamentos. Logo, se houver muitos casamentos o algoritmo torna-se ineficiente.

Para a implementação do algoritmo Shift-And Aproximado, foi utilizada a implementação proposta em aula com adaptação para que fosse possível escolher quais erros seriam permitidos: inserção, remoção e/ou substituição, além da distância de edição k. Esse algoritmo retorna como valores as posições finais em que o padrão se encontra no texto, funcionando de forma distinta dos algoritmos exatos, que retornam as posições iniciais do padrão no texto.

Dessa forma, dado o exemplo de entrada “Textos Testam”e o padrão “Texto”, os algoritmos exatos retornarão o valor 1, ou seja, a palavra “Texto”é encontrada a partir do primeiro caractere. Já o algoritmo aproximado, considerando k=1 e todas os erros permitidos, retornaria os valores 4, 5 e 6, que indicam, respectivamente, um match aproximado para a posição 4, ou seja, remoção da letra ‘o’ do padrão “Texto”, match exato para a posição 5 e match aproximado para a posição 6, ou seja, inserção da letra ‘s’ ao final do padrão “Texto”.

3. Utilização

Para utilizar o programa no modo normal, é preciso compilar os arquivos utilizando o comando $make$. Para compilar em modo DEBUG, é preciso utilizar o comando $make\ DEBUG = 1$. Em ambos os casos, será gerado um arquivo de saída chamado $main$. Para executar, basta utilizar $./main$.

4. Resultados

Os testes para os algoritmos Boyer-Moore e Shift-And Exato foram feitos utilizando 3 arquivos com tamanhos diferentes, um com 4 milhões de palavras, outro com 395 mil palavras e o último 7 palavras.

As tabelas abaixo mostram o tempo necessário para o casamento dos padrões utilizando cada um dos algoritmos. Os padrões foram escolhidos aleatoriamente com uma simples busca em cada arquivo.

Padrão	Tempo Boyer-Moore	Tempo Shift-And Exato
scandinavian	0.035391	0.097361
moment lovelace writes	0.036620	0.097413
glorious age of irish history	0.027067	0.117789
prova	0.079457	0.137178
p	2.164941	1.872866
V	0.368231	0.183745
i	7.291479	6.753407

Tabela 1. Boyer-Moore e Shift-And Exato para o arquivo 4M palavras

Padrão	Tempo Boyer-Moore	Tempo Shift-And Exato
scandinavian	0.019483	0.057105
moment lovelace writes	0.019823	0.052847
glorious age of irish history	0.014169	0.052584
prova	0.037444	0.064851
p	0.655616	0.539392
V	0.170641	0.096043
i	2.196384	1.977714

Tabela 2. Boyer-Moore e Shift-And Exato para o arquivo 395k palavras

É possível perceber que o algoritmo Boyer-Moore possui um desempenho menor em relação ao Shift-And Exato quando há muitas ocorrências de padrão pois se assemelha

Padrão	Tempo Boyer-Moore	Tempo Shift-And Exato
scandinavian	0.000020	0.000015
moment lovelace writes	0.000026	0.000018
glorious age of irish history	0.000029	0.000020
prova	0.000039	0.000035
p	0.000063	0.000069
V	0.000015	0.000028
i	0.000018	0.000015

Tabela 3. Boyer-Moore e Shift-And Exato para o arquivo 7 palavras

ao algoritmo de força bruta e não faz a operação deslocamento com frequência. Esse comportamento pode ser visto mais claramente no gráfico 1

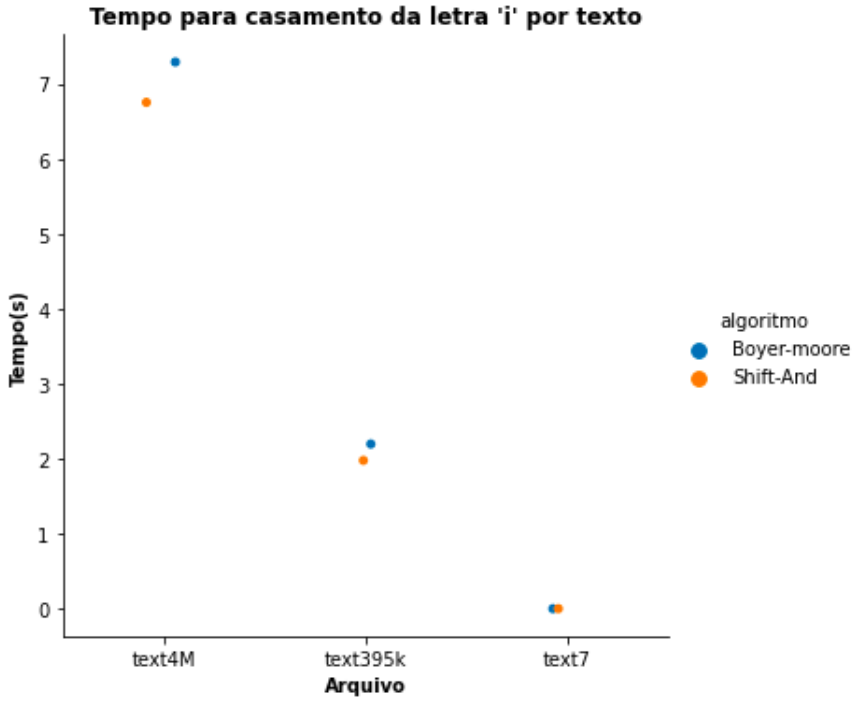
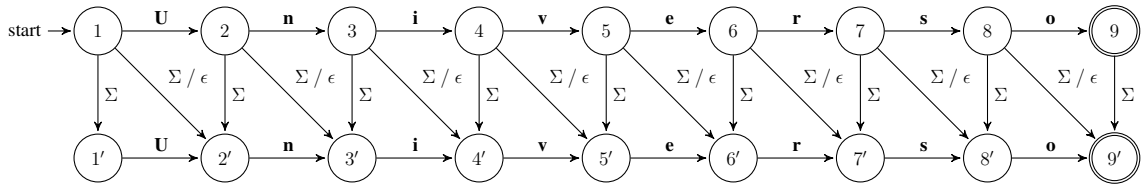


Figura 1. Casamento do padrão i

É notável, portanto, que quando o texto possui um tamanho considerável e o padrão ocorre com frequência no texto faz-se necessário uma análise de qual algoritmo utilizar. Quando em ambientes menores, com textos pequenos, independe qual algoritmo utilizar visto que pela complexidade assintótica os dois algoritmos são parecidos.

Por sua vez, para observar o funcionamento do algoritmo Shift-And Exato, foi utilizado o texto “A Universidade atrapalha meu desempenho na Universidade, porque é tanta coisa da Universidade para fazer que eu não tenho tempo para fazer as coisas da Universidade”, sem aspas. E o padrão buscado foi “Universo”, sem aspas.

O autômato referente ao padrão é representado abaixo:



Quando esse teste é executado para a distância de edição k=1 e é permitida a inserção a remoção e a substituição, é obtido o resultado mostrado na Tabela 4.

Saída
Shift-And Aprox: Match no indice: 9
Shift-And Aprox: Match no indice: 10
Shift-And Aprox: Match no indice: 50
Shift-And Aprox: Match no indice: 51
Shift-And Aprox: Match no indice: 89
Shift-And Aprox: Match no indice: 90
Shift-And Aprox: Match no indice: 160
Shift-And Aprox: Match no indice: 161
Tempo gasto: 0.004000

Tabela 4. Shift-And Aproximado

Cada match apresentado na Tabela 4 está relacionado ao encontro aproximado do padrão no texto nos respectivos índices. Nos índices 9, 50, 89 e 160 é encontrado um padrão aproximado em que se considera a remoção de um caractere, sendo um match na posição 9' do autômato representado. Já nos índices 10, 51, 90 e 161 ocorre um match aproximado em que se considera a substituição do caractere 'o' pelo caractere 'i' que é encontrado no texto nessas posições, ocorrendo um match também na posição 9' do autômato.

5. Considerações Finais

Finalmente, tendo os algoritmos programados e testados, podemos perceber a alta eficiência dos três algoritmos, assim como sua simplicidade de programação. Além disso, é possível observar que no caso do casamento exato o algoritmo Boyer-Moore é mais eficiente em situações na qual o padrão não é encontrada repetidas vezes, pois isso impede que o algoritmo prossiga mais rapidamente pelo texto. Dessa forma, o algoritmo Shift-And Exato é mais eficiente nos casos em que o padrão se repete muitas vezes no texto. Por fim, é interessante observar que o algoritmo Shift-And Aproximado tem um padrão de funcionamento diferente dos algoritmos exatos, uma vez que retorna a posição final em que ocorre o match ao invés da posição inicial. De toda forma, o algoritmo também apresenta um desempenho muito bom no processo de busca.

Referências

[1] Guilherme Tavares de Assis. Casamento de cadeias. http://www.decom.ufop.br/guilherme/BCC203/geral/ed2_casamento-cadeias.pdf. (Acessado em 10 de Maio de 2021).

[2] Geeks for Geeks. Boyer moore algorithm for pattern searching. https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/. (Acessado em 12 de Maio de 2021).

[3] Nivio Ziviani. *Projeto de Algoritmos com Implementações em Java e C++*. 2007.