Universidade Federal de Ouro Preto - UFOP

Ana Letícia Chaves Neiva

Trabalho Prático III

Casamento Exato de Cadeias

Ouro Preto
2017

Ana Letícia Chaves Neiva

Trabalho Prático III

Casamento Exato de Cadeias

Trabalho apresentado no curso de Graduação em Ciência da Computação

Prof. Dr. Guilherme Tavares de Assis

Ouro Preto
2017

1 - Introdução

Um texto é uma ocorrência linguística, escrita ou falada de qualquer extensão, dotada de unidade sócio comunicativa, semântica e formal. É uma unidade de linguagem em uso. Textos estão presentes em livros, jornais, revistas, artigos, publicidades, redes sociais, dentre outros; e são o principal meio de comunicação após a linguagem falada. É comum que se queira localizar dentro de um texto, uma determinada palavra ou frase, e essa tarefa pode ser trabalhosa e cansativa para humanos conforme o tamanho deste texto.

Com o desenvolvimento da tecnologia, novas possibilidades realizarem essa tarefa de forma automática, inclusive mais rápida e confiável. A este processo computadorizado de localização de padrões dentro de um texto é que se dá o nome de casamento de cadeias.

Existem atualmente diversos algoritmos de casamento, e a maioria das linguagens de programação modernas já possuem bibliotecas com funções para lidar com cadeias, incluindo uma ou mais funções de pesquisa de um padrão dentro de uma cadeia maior. Entretanto, é interessante entender os princípios destes algoritmos e para quais casos cada um melhor se adequa.

1.1 - Especificação do problema a ser resolvido

Este trabalho tem como objetivo realizar uma análise experimental, afim de comparar alguns dos métodos de casamento exato de cadeia.

1.2 - Descrição dos testes realizados

A seguir são apresentados os métodos considerados e os dados de suas execuções sobre um texto e um padrão de tamanhos variados.

Os textos foram gerados usando-se um gerador de Lipsum, que nada mais é do que

um texto utilizado para preencher o espaço de texto em publicações (jornais, revistas, e websites), com a finalidade de verificar o layout, tipografia e formatação antes de utilizar conteúdo real. Muitas vezes este texto também é utilizado em catálogos de tipografia para demonstrar textos e títulos escritos com as fontes

Os padrões foram escolhidos a partir de cada arquivo, selecionando-se fragmentos do texto com o tamanho adequado e que possuíam pelo menos duas ocorrências, a m de evitar o pior caso dos algoritmos.

2 - Métodos

2.1 - Força Bruta

Tamanho do texto	Tamanho do padrão	Comparações	Tempo (segundos)
pequeno	pequeno	28720	0.000416
pequeno	médio	29421	0.0002
pequeno	grande	29597	0.000233
médio	pequeno	19840	0.000159
médio	médio	20490	0.000299
médio	grande	20625	0.000329
grande	pequeno	5052970	0.027367
grande	médio	5221357	0.024421
grande	grande	5336326	0.026215

Observando a tabela é possível concluir que apesar do tempo de execução ter sido semelhante aos outros métodos, o Casamento por Força Bruta faz um número de comparações desnecessárias extremamente grande, mais que o dobro constatado nas outras tabelas.

Os números de comparações elevados indicam claramente o custo de processamento maior requerido por esse método, o que pode também afetar o tempo. Os testes foram realizados com arquivos pequenos e processadores relativamente poderosos, ao contrário de micro- computadores que podem ser encarregados de realizar essas tarefas.

No caso, o custo das comparações estaria muito mais evidente no tempo gasto. Para arquivos pequenos, mesmo o maior usado para testes, a Força Bruta se mostra viável como um código rápido de ser feito em casos de necessidade instantânea e da disponibilidade de um computador que torne indiferente o custo computacional das comparações. Já para arquivos com muitos dados, por exemplo na casa dos milhões, esse método fará tbm milhões de comparações, sendo assim inviável.

2.2 - Boyer-Moore

Tamanho do texto	Tamanho do padrão	Comparações	Deslocamentos	Tempo (segundos)
pequeno	pequeno	19301	14361	0.00407
pequeno	médio	16849	14379	0.000377
pequeno	grande	15837	14382	0.000381
médio	pequeno	13314	9913	0.000273
médio	médio	11816	9912	0.000329
médio	grande	11498	9912	0.000252
grande	pequeno	3388881	2523006	0.031624
grande	médio	3005130	2523048	0.029414
grande	grande	2906686	2523015	0.024144

Neste trabalho, a implementação do algoritmo utiliza a heurística por ocorrência. Analisando a tabela acima, percebe-se que os números de deslocamentos são bem próximos quando se compara os resultados para cada tamanho de texto, ou seja, o aumento apenas do tamanho do padrão resulta em pouca diferença com relação à quantidade de deslocamentos. Quanto às comparações, temos que quanto menor o tamanho do padrão, mais comparações serão realizadas. Isso deve-se ao fato de que padrões maiores resultarão em deslocamentos maiores após uma colisão, o que garante que mais caracteres serão ignorados e o número de comparações tende a diminuir. Para os tamanhos de textos pequeno e médio, temos um tempo de execução bem parecido, devido ao fato de que o número de comparações e deslocamentos são muito próximos, quando se compara estes números sem considerar o contexto do algoritmo.

2.3 - Boyer-Moore-Horspool

Tamanho do texto	Tamanho do padrão	Comparações	Deslocamentos	Tempo (segundos)
pequeno	pequeno	3535	3307	0.000266
pequeno	médio	3475	3281	0.000066
pequeno	grande	3475	3281	0.000083
médio	pequeno	2466	2304	0.000219
médio	médio	2434	2304	0.000091
médio	grande	2434	2304	0.000085
grande	pequeno	628206	585377	0.020156
grande	médio	618314	585339	0.009254
grande	grande	618314	585339	0.011693

Analisando e comparando a tabela acima com a tabela do algoritmo Boyer-Moore, podemos perceber que os números de deslocamento continuam próximos quando se compara os resultados para cada tamanho de texto, mas isso também ocorre com os números de comparações. Podemos notar também que ambos os números são muito menores no Boyer-Moore-Horspool, já que se não for

encontrado um casamento no laço while interno, o algoritmo verifica qual é, naquele momento, o i-ésimo caractere no texto, correspondente ao último caractere do padrão. Dado esse caractere, verifica-se qual será seu valor de deslocamento. Essa informação diz quanto o padrão deveria andar para casar esse i-ésimo caractere no texto com um possível caractere igual a este no padrão.

2.4 - Boyer-Moore-Horspool-Sunday

Tamanho do Texto	Tamanho do Padrão	Comparações	Deslocamentos	Tempo(s)
Pequeno	Pequeno	2769	2610	0.015s
Pequeno	Médio	2016	1889	0s
Pequeno	Grande	1787	1611	0s
Médio	Pequeno	50688	47051	0s
Médio	Médio	36401	34672	0s
Médio	Grande	30968	27687	0s
Grande	Pequeno	1919885411	1629498477	0s
Grande	Médio	1618895423	1429468434	0s
Grande	Grande	1485466231	1263458920	0s

Quanto maior o tamanho do padrão , menor é o número de comparações, mesmo para tamanhos de textos diferentes, podemos dizer então que esse método é uma boa opção quando se deseja pesquisar por padrões muito grandes.

2.5 - Shift-And Exato

Tamanho do texto	Tamanho do padrão	Comparações	Tempo (segundos)
pequeno	pequeno	14355	0.001411
pequeno	$20 \leq m \leq 30$	14355	0.001264
médio	pequeno	9911	0.000896
médio	$20 \leq m \leq 30$	9911	0.000446
grande	pequeno	2523004	0.065567
grande	$20 \leq m \leq 30$	2523004	0.057232

Ao observar a tabela percebe-se que o número de comparações dentro de um texto é constante, independente do tamanho do padrão. Entretanto, a utilização de máscara de bits limita o tamanho máximo do padrão. Outra observação interessante, é de que, num mesmo arquivo de texto, ao aumentar o tamanho do padrão, o tempo de busca cai, se tornando maior a diferença quanto maior for o arquivo. Isso acontece porque, com o uso do paralelismo de bit, é possível computar novas máscaras com o custo O(1).

3 - Conclusão

3.1 - Análise comparativa de desempenho entre os métodos

Inicialmente, podemos dizer que o algoritmo mais engenhoso e que, de certa forma, merece mais crédito pela criatividade de seus autores é o Shift-And, mas não deve-se descartar a engenhosidade dos algoritmos de Boyer-Moore , Boyer-Moore-Horspool e Boyer-Moore-Horspool-Sunday . Como é possível analisar na tabela abaixo, ao pegar-se um texto médio no qual se deva encontrar um padrão pequeno (que é uma situação mais comum se tratando do que editores de texto frequentemente são colocados pra localizar), perceberemos que:

1. O algoritmo de força bruta é inviável, não apenas neste caso como em todos os outros.

2. O algoritmo de Boyer-Moore-Horspool possui o melhor desempenho e seria, portanto, a melhor escolha caso o grupo criasse um editor de texto com a função localizar.

3. O algoritmo de Boyer-Moore possui um bom desempenho e poderia também ser escolhido em um projeto de implementação de um editor de texto com a função localizar.

4. O algoritmo Shift-And seria uma boa escolha caso fosse de interesse do projeto fazer com que o número de comparações variasse o mínimo possível para diferentes tamanhos de padrão. Caso contrário, os algoritmos de Boyer-Moore e Boyer-Moore-Horspool viriam a ser melhores escolhas.

5 . O algoritmo de Boyer-Moore-Horspool-Sunday também possui um bom desempenho e seria, portanto, uma boa escolha caso o grupo criasse um editor de texto com a função localizar.

Não houve muitas dificuldades com relação à implementação dos métodos e a realização dos testes. Talvez a maior dificuldade encontrada tenha sido os erros corriqueiros de programação rapidamente corrigidos e que são comuns a todo processo de implementação de programas.