Algoritmos de busca de string em arquivo

Filipe Ramos, Nilton Jose Mocelin Junior, Paula Campigotto

Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (Udesc)
Joinville – SC – Brasil

ik.ben.filipe@gmail.com, nilton.junior@edu.udesc.br,
paula_campigotto@hotmail.com

Resumo. *Este artigo descreve seis algoritmos de busca de string em arquivo para a disciplina Projeto de Arquivos. Tais algoritmos foram implementados em Java, valendo-se dos artifícios da programação orientada a objeto a fim de facilitar a codificação e interpretação. Por fim, as implementações foram comparadas no que tange ao desempenho das aplicações, resultando em uma análise.*

1. Introdução

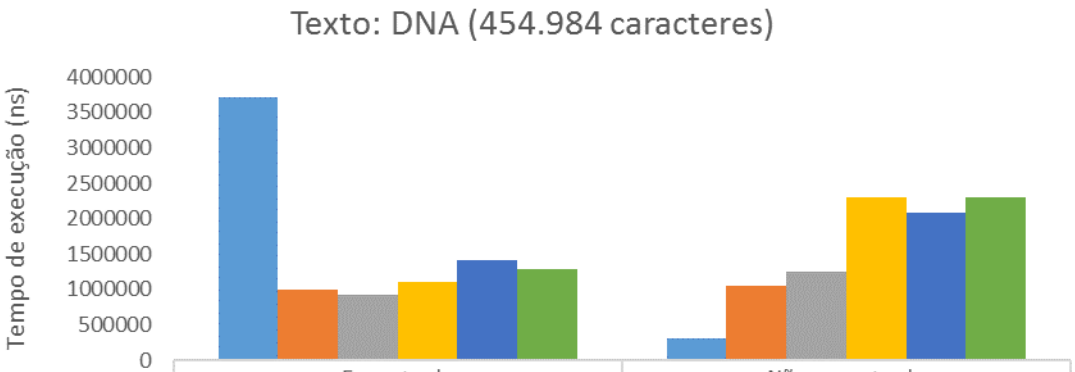
Entre os algoritmos de busca de substring, pode-se citar o Naive, Boyer-Moore, Knuth-Morris-Prat (KMP), Rabin Karp, Aho-Corasick e Radix Tree. O algoritmo Naive é um programa iterativo simples, no qual se percorre o texto caractere a caractere até se encontrar uma dada substring. O tempo de execução dessa estratégia cresce linearmente com a posição da substring no texto [Hume 1991]. Já no algoritmo Boyer-Moore, a substring é pré-processada de modo que seja possível avançar partes do texto. Essa aplicação funciona melhor quando os padrões de busca são maiores [Boyer and Moore 1977].

A estratégia de KMP também pré-processa o padrão a ser buscado, mas emprega uma tabela – vetor – para comparar os sufixos e prefixos da substring, permitindo identificar em que parte do texto fazer a próxima comparação [GeeksforGeeks 2017]. O algoritmo de Rabin Karp é uma estratégia iterativa auxiliada por uma função hash [Wikipedia 2018]. As aplicações Aho-Corasick e Radix Tree são estruturas de dados, árvores, geradas com base no padrão a ser pesquisado [Aho and Corasick 1975]. Ambas as estratégias são aplicações otimizadas de procura de múltiplos padrões, e, diferentemente do Aho-Corasick, o Radix Tree não permite a busca de um único padrão ou de padrões que iniciem com o mesmo caractere [Morin 2012].

2. Diferenças de desempenho

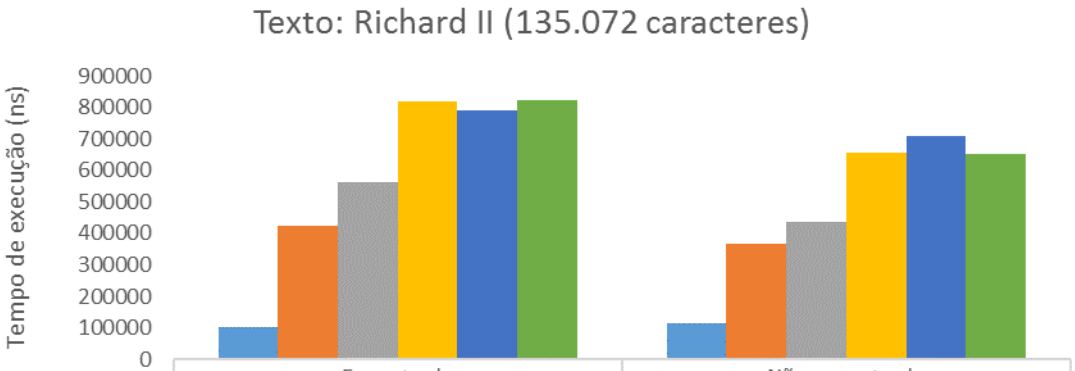
Esta seção apresenta análises de desempenho para os algoritmos descritos anteriormente. Os resultados das análises seguem os Gráficos 1, 2 e 3.

Gráfico 1. Comparação entre os algoritmos de busca quando uma palavra é encontrada após percorrer 40% do texto e quando não é encontrada



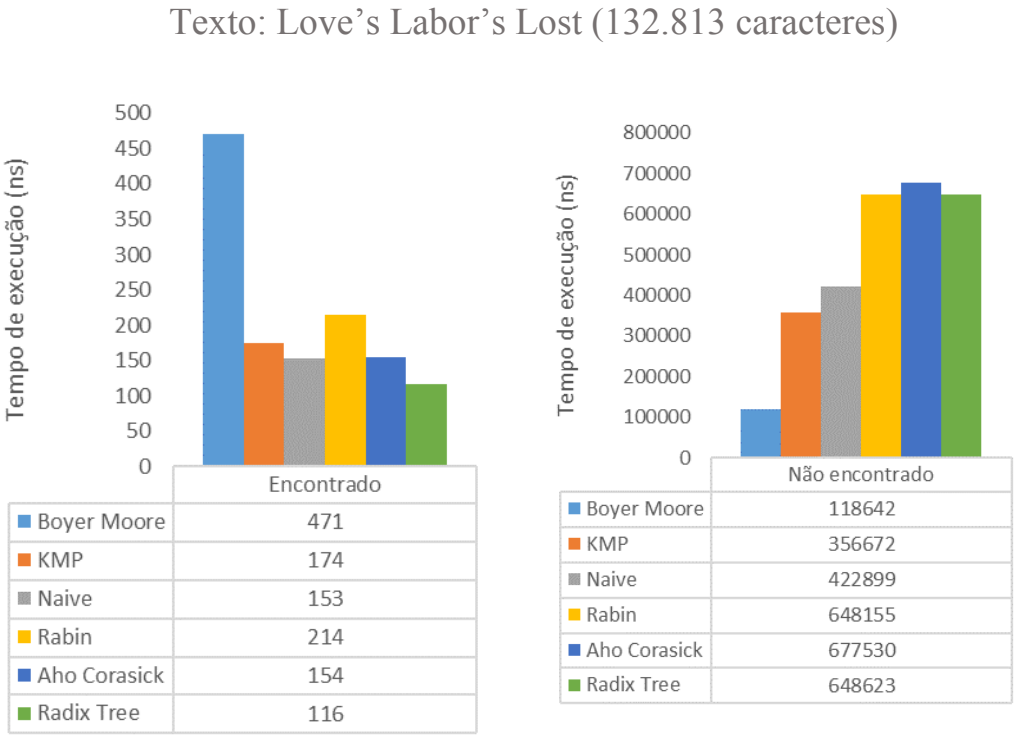
	Encontrado	Não encontrado
Boyer Moore	3719104	309764
KMP	1001120	1043116
Naive	921645	1248139
Rabin	1105006	2305912
Aho Corasick	1421233	2088820
Radix Tree	1289749	2298582

Gráfico 2. Comparação entre os algoritmos de busca quando uma palavra é encontrada após percorrer 99% do texto e quando não é encontrada



	Encontrado	Não encontrado
Boyer Moore	100921	116262
KMP	422919	367725
Naive	563113	436374
Rabin	821287	655059
Aho Corasick	790389	710018
Radix Tree	822824	651428

Gráfico 3. Comparação entre os algoritmos de busca quando uma palavra é encontrada no primeiro caractere (índice 0) e quando não é encontrada



A partir da análise dos gráficos é possível notar que o algoritmo Boyer Moore obtém os melhores resultados quando o texto possui grande número de caracteres e a busca percorre a maior parte deste texto, pois quando o texto é pequeno ou a palavra é encontrada percorrendo-se menos da metade do texto, o algoritmo passa a ser o pior de todos.

O algoritmo Rabin Karp possui um comportamento atípico apenas quando uma palavra é encontrada aproximadamente na metade do texto, conforme o Gráfico 1, nessa situação ele obtém a terceira melhor performance, enquanto que após percorrer o texto todo, a sua posição é a última.

Os demais algoritmos não possuem grandes variações com os diferentes casos testados, obtendo geralmente a seguinte ordem de desempenhos: KMP, Naive, Aho corasick e Radix Tree.

Referências

Aho, A. V. and Corasick, M. J. (1975) "Efficient string matching: An aid to bibliographic search". Disponível em: <https://dl.acm.org/citation.cfm?doid=360825.360855>. Acesso em: 20 nov. 2018.

Boyer, R. S. and Moore, J. S. (1977) "A fast string searching algorithm". Disponível em: <https://dl.acm.org/citation.cfm?doid=359842.359859>. Acesso em: 12 nov. 2018.

GeeksforGeeks (2017) “KMP Algorithm for Pattern Searching”. Disponível em: <https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>. Acesso em: 12 nov. 2018.

Hume, S. (1991) “Fast String Searching”. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211105>. Acesso em: 12 nov. 2018.

Morin, P. (2012) “Data Structures for Strings”. Disponível em: <http://cglab.ca/~morin/teaching/5408/notes/strings.pdf>. Acesso em: 20 nov. 2018.

Wikipedia (2018) “Rabin–Karp algorithm”. Disponível em: <https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm>. Acesso em: 13 nov. 2018.