**Comparação de performance de algoritmos de pesquisa em Strings[[1]](#footnote-0)\***

**Leonardo Valério Anastácio¹, Lucas Litter Mentz²**

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

{leonardo.anastacio, lucas.mentz}@edu.udesc.br

***Resumo.*** *Na literatura há diversos algoritmos com o objetivo de encontrar padrões em strings. O presente trabalho retrata, por meio de tabelas, as características de cada algoritmo, evidenciando os momentos em que as operações se deterioram/destacam.*

1. **Introdução**

Neste trabalho será investigado a performance dos algoritmos de busca de Substrings: *Naive*, *Rabin-Karp, Knuth-Morris-Pratt, Boyer-Moore, Aho-Corasick*. O objetivo é estimar os momentos em que ocorre a degradação e destaque destes, para assim estabelecer padrões e definir a escolha coerente do algoritmo para determinada ocasião. Os testes foram realizados em um computador equipado com processador Intel Core i5-2500k rodando à 4.6GHz com 16GB de RAM.

1. **Desenvolvimento**

A tabela 1 apresenta os casos de teste executados sobre o texto “Lorem Ipsum” completo. A tabela 2 relaciona os testes sobre um texto composto por dez milhões de letras ‘a’ seguidos de 10 milhões de letras ‘b’ e 10 milhões de ‘a’s e ‘b’s aleatórios. Na tabela 3 temos os resultados dos testes sobre uma cópia em texto simples do livro “War and Peace”, de Leo Tolstoy. O algoritmo *Aho-Corasick* não teve sucesso em nenhum caso devido a erros na implementação. Todos os tempos das tabelas estão em milissegundos.

Tabela 1 – Testes com Lorem Ipsum

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritmo | Teste 1 - 23,9kB - Lorem Ipsum (tempo em milissegundos) | | | | | | |
| 5B, OK | 5B, OK | 5B, OK | 40B, NOK | 7B, NOK | 1B, OK | 875B, OK |
| Naive | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BoyerMoore | 3 | 1 | 1 | 1 | 1 | 4 | 1 |
| RabinKarp | 2 | 1 | 1 | 3 | 0 | 1 | 11 |
| KMP | 3 | 3 | 2 | 1 | 1 | 0 | 1 |
| AhoCorasick | - | - | - | - | - | - | - |

Tabela 2 – Testes com 30 milhões de caracteres ‘a’ e ‘b’.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritmo | Teste 2 - 28,6MB - 10M×(A + B + AB) (tempo em milissegundos) | | | | | | |
| 1B, OK | 34B, OK | 35B, OK | 2B, OK | 1B, OK | 711B, OK | 28B, OK |
| Naive | 159 | 406 | 417 | 163 | 159 | 3416 | 166 |
| BoyerMoore | 138 | 756 | 47 | 135 | 145 | 42 | 49 |
| RabinKarp | 721 | 971 | 718 | 526 | 624 | 7595 | 701 |
| KMP | 138 | 132 | 152 | 138 | 116 | 152 | 169 |
| AhoCorasick | - | - | - | - | - | - | - |

Tabela 3 – Testes com 30 milhões de caracteres ‘a’ e ‘b’.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algoritmo | Teste 3 - 3,2MB - War and Peace (tempo em milissegundos) | | | | | | |
| 5B, OK | 1B, OK | 3B, OK | 24B, OK | 8B, OK | 1B, OK | 10kB, OK |
| Naive | 13 | 14 | 12 | 13 | 12 | 12 | 12 |
| BoyerMoore | 4 | 13 | 5 | 2 | 2 | 12 | 2 |
| RabinKarp | 47 | 47 | 47 | 58 | 52 | 42 | 14181 |
| KMP | 16 | 16 | 15 | 15 | 15 | 15 | 13 |
| AhoCorasick | - | - | - | - | - | - | - |

1. **Resultados**

O algoritmo *Naive* é eficiente em padrões em que o texto e a *string* de busca são curtos, e até compete quando os textos são grandes, isso é devido ao fato de não haver pré-processamento. Em contrapartida, o algoritmo de *Rabin-Karp* apresentou resultados bem piores em relação ao *Naive*, apesar de ambos usam a mesma forma de deslocamento, visto que que este possui uma eficiência reduzida devido ao uso de *hashing*. Já o algoritmo de *Knuth-Morris-Pratt* se mostrou bem eficiente e consistente ao analisar textos extremamente grandes com padrões de busca tanto curtos como longos. Apesar disso, em alguns casos foi superado pelos resultados obtidos por *Boyer-Moore* que mostram maior eficiência com testes maiores. Por último, exceto em situações em padrões de busca muito presentes no texto, *Boyer-Moore* alcançou melhores resultados comparados a todos os outros algoritmos. Por motivos ainda não reconhecidos a implementação de *Aho-Corasick* teve mal funcionamento e não pôde ter seus dados analisados.

1. **Conclusão**

A partir dos dados obtidos, conclui-se que na maioria das situações o algoritmo de *Boyer-Moore* se identifica como a melhor escolha, em contrapartida, o restante dos algoritmos necessitam de cenários específicos para se sobressaírem. A operação de força bruta do algoritmo *naive* é uma boa escolha conseguindo manter um desempenho respeitável até com os casos de teste maiores, além de economizar tempo na implementação e ter o desempenho parecido com algoritmos de busca refinados.

1. **Referências**

G. A. Moreira. Algoritmos para busca de padrões: **Uma Análise Comparativa Empírica**. Trabalho de Conclusão de Curso, Faculdade de Tecnologia de São Paulo. Junho, 2012.

**Busca de palavras em um texto**. Disponível em: <https://www.ime.usp.br/~pf/mac0122-2002/aulas/str-matching.html>. Acesso em: 08/10/2017.

**Algoritmo KMP para busca em substring.** Disponível em: <https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/kmp.html>. Acesso em: 08/10/2017.

1. \*Este trabalho foi desenvolvido para a matéria de PRA0001, do Bacharelado em Ciência da

   Computação, UDESC-CCT. [↑](#footnote-ref-0)