String search

Guilherme M. Utiama[1], Lucas Camilo[1], Peter L. Brendel[1]

[1]Centro de Ciências Tecnológicas - Universidade do Estado de Santa Catarina (UDESC)
CEP 89.219-710 – Joinville – SC – Brazil

{guilherme.utiama, peter.brendel, lucas.camilo}@edu.udesc.br

Resumo. *Este trabalho consiste na busca de strings repetidas em um contexto, para se fazer a busca, foi implementado algoritmos genericos que buscam strings iguais. Com isso, o objetivo deste é analisar uma estrutura de texto verificando se alguma palavra foi detectada no texto, caso isso ocorra, mostrar na tela seu índice(linha e coluna onde se encontra a palavra).*

Abstract. *This work review in strings relapses in context, could be found for a search, was implemented generic algorithms that look for equal strings. With this, the goal is a way to get detailed information about the existence of some type of text, if this occurs, to show its content.*

1. Introdução

Este trabalho consiste na comparação de performance dos algoritmos de busca de substrings, sendo eles: Naive, Rabin Karp, Knuth-Morris-Prat (KMP), Boyer-Moore, Aho-Corasick, Radix tree. O objetivo é mostrar o melhor auge de cada algoritmo, mostrando suas decadências umas com as outras. Com isso, analiticamente mostrando num ponto geral os melhores e piores momentos de cada algoritmo. Os teste foram realizados nos computadores da UDESC.

2. Analise e desenvolvimento

A partir da construção dos algoritmos genéricos, obteve-se os seguintes resultados com consultas de palavras contidas nos textos:

Algoritmo	Teste 1 - Palavras soltas (segundos)			
	a	John	is	just
Naive	0.33	0.128	0.127	0.119
BoyerMoore	0.129	0.163	0.134	0.16
RabinKarp	0.29	0.142	0.126	0.115
KMP	13	19	12	10
AhoCorasick	-	-	-	-
Radix Tree	-	-	-	-

Figura 1. Tabela palavras/tempo

Resultados obtidos na pesquisa com base de dois arquivos.

3. Resultados obtidos

O algoritmo Naive tem uma performance eficiente para textos e strings curtas, isso ocorre devido a ele não precisar ter um processamento de calculo antes da busca. Porém, o algoritmo de Rabin-Karp apresenta resultados ”opostos”ao algoritmo Naive, mesmo que usem o mesmo estilo de busca, a eficiência dele é reduzida por usar hashing. O algoritmo KMP se mostrou mais consistente ao analisar textos maiores, ele teve um mau desempenho. O algoritmo Boyer-Moore que teve um desempenho parecido com o Rabin-Karp. e por fim, não pudemos concluir os resultados dos algoritmos de Aho-Corasick, Radix Tree, por falta da conclusão dos algoritmos.

4. Conclusão

Com os dados obtidos em experiências, foi concluido que o algoritmo de Boyer-Moore se destaca nos casos de analise de textos gerais. O restante dos algoritmos mostram ter um resultado eficiente em casos específicos, como por exemplo, o algoritmo naive que teve um desempenho aceitável, por ser um algoritmo de força bruta e de fácil implementação.

Referências

Dhruv Gairola. The MVC pattern and SWING. Disponível em: https://stackoverflow.com/questions/5217611/the-mvc-pattern-and-swing/ Acesso em: 14 de novembro de 2018

S. Saurel. Learn to make a MVC application with Swing and Java 8s. Disponível em: https://www.ssaurel.com/blog/learn-to-make-a-mvc-application-with-swing-and-java-8/ Acesso em: 14 de novembro de 2018