

Conceção e análise de algoritmos

***Plagiarism Checker***



Fábio Filipe Jesus Silva - ei11107

Fernando Manuel Rocha Magalhães - ei07046

José Pedro Lobo Marinho Trocado Moreira - ei12002

Porto, 28 de maio de 2013

**Resumo**

Este trabalho contextualiza-se num sistema de deteção de plágio de ficheiros de texto.

Tal deteção é feita através do conceito de **subsequência comum mais comprida**, com recurso a programação dinâmica.

Este sistema compara um ficheiro de texto com uma base de dados (neste caso um diretório com vários ficheiros de texto) e calcula o grau de similaridade entre ambos.

Sendo assim, os dados de entrada são, portanto, os caminhos para a base de dados e para o ficheiro de texto. Tais caminhos podem ser inseridos das seguintes maneiras:

- Manualmente (pedido pelo programa em tempo de execução)

- “Arrastando” a pasta e o ficheiro para o executável (*Windows*)

- Como parâmetros da linha de comandos (*Windows* e *GNU*/*Linux*)

Como se dá a entender, o programa é passível de ser compilado tanto em *Windows* como em *GNU*/*Linux*.

Embora cada uma destas plataformas use funções distintas ao nível dos diretórios, o código-fonte está devidamente preparado para compilação em ambas.

Em termos de resultado final, espera-se que o programa devolva, para cada um dos ficheiros na base de dados:

- *Strings* similares às do ficheiro novo (isto é, com um número maior ou igual de carateres definido)

- Grau de similaridade percentual em relação ao ficheiro novo

**Índice**

Principais algoritmos / Análise de complexidade…………………………………………………………….4

Principais dificuldades e esforço individual...…………………………………………………………………6

Bibliografia………………………………………………………………………………………………………..7

Conclusão………………………………………………………………………………………………………,,,8

**Principais algoritmos**

O “método” usado para este trabalho foi o **LCS** (*Longest Common Sequence*), com recurso a programação dinâmica.

Em termos de complexidade teórica, carateriza-se por ser linear no comprimento das *strings* a comparar (**O** (n x m)), sendo *n* e *m* os comprimentos das *strings*.

Na prática, este comportamento foi verificado, como mostra a tabela abaixo.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| m | n | t1 | t2 | t3 | t médio | α |
| 3305 | 726 | 0.033215 | 0.0332181 | 0.0350442 | 0.033825767 | 0.000047 |
| 3305 | 2135 | 0.0983369 | 0.0990908 | 0.0994 | 0.189894256 | 0.000046 |
| 3305 | 3978 | 0.184175 | 0.184447 | 0.183184 | 0.183935333 | 0.000046 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| m | n | t1 | t2 | t3 | t médio | α |
| 5833 | 7835 | 0.446651 | 0.427481 | 0.426716 | 0.433616 | 0.000074 |
| 2652 | 7835 | 0.193787 | 0.193083 | 0.19376 | 0.193543333 | 0.000073 |
| 518 | 7835 | 0.0383321 | 0.0388155 | 0.0383933 | 0.0385133 | 0.000074 |

Nas tabelas acima, as variáveis *t1*, *t2* e *t3* representam três ensaios feitos ao tempo de execução em seis ensaios diferentes, onde se variou ora *m*, ora *n*.

**α** representa a proporcionalidade entre o comprimento da *string* e o tempo levado a executar o algoritmo.

Através destes gráficos gerados com os valores das tabelas, verifica-se que há pois uma dependência linear entre os tamanhos das *strings* e os tempos de execução, que é demonstrada também pelos valores semelhantes de **α.**

**Principais dificuldades e esforço individual**

Em geral, este segundo projeto foi mais acessível que o primeiro.

No entanto, o mais difícil foi sem dúvida compatibilizar o código tanto para *Windows* como para *GNU*/*Linux*. Isto deve-se ao fato de as funções relativas a diretórios variarem em ambas as plataformas.

A solução que implementámos foi a compilação condicional, com o auxílio de diretivas de pré-processamento.

Quanto ao trabalho desempenhado por cada membro do grupo, esse foi equitativo.

**Bibliografia**

*http://en.wikipedia.org/wiki/Diff* - Página *wiki* sobre o comando ***diff*** (*GNU*/*Linux*), que se assemelha ao nosso projeto

*http://en.wikipedia.org/wiki/Longest\_common\_subsequence\_problem* - Página *wiki* sobre o método LCS (*Longest Common Sequence*), no qual se baseia o comando acima e consequentemente também o nosso projeto.

http://en.wikipedia.org/wiki/C\_preprocessor - Página relacionada com o pré-processador em C/C++, que nos ajudou a criar código multiplataforma.

**Conclusão**

Em suma, podemos afirmar que este sistema de deteção de plágio comporta-se de maneira bastante satisfatória, pois apresenta taxas de similaridade muito reais.

Apesar das dificuldades supra citadas, foi um trabalho que nos agradou realizar, pois permitiu-nos apreender conceitos relacionados com algoritmos direcionados a *strings*, que no fundo era o real objetivo de todo o projeto.