U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Faculdade de Engenharia da Universidade do Porto

Conceção e Análise de Algoritmos

CAL

2º Ano

2º Semestre

Tema 5: Similaridade de Ficheiros

1 de Junho de 2014

Autores:

Pedro Lemos Faria – ei12097@fe.up.pt

João Correia Ferreira – meinf1208808@fe.up.pt

Índice

Introdução **3**

Conceção e implementação da solução **4**

Análise de Complexidade **5**

Conclusão **6**

Bibliografia **7**

Introdução

Este projeto surge no âmbito da Unidade curricular, Conceção e Análise de Algoritmos. Todo este projeto se baseia na pesquisa por *strings*, um tema que tem sido abordado tanto nas aulas teóricas pelo docente, como desenvolvido nas aulas teórico-práticas.

O intuito deste projeto desenvolvido em C++, e permitir ao utilizador que escolha dois ficheiros de texto e os compare de diferentes formas:

- Fazer uma comparação percentual em que nos é indicada a semelhança entre os dois ficheiros.
- Analisar os ficheiros como se tratasse de um sistema de controlo de versões, no qual é indicado o que foi removido e o que foi adicionado ao ficheiro original, quando comparado com o ficheiro mais recente.

Para a resolução do problema utilizamos o algoritmo proposto no enunciado, longest common subsequence (LCS), assim como o algoritmo Levenshtein distance.

Conceção e implementação da solução

De modo a resolver o problema que nos foi proposto neste projeto, optamos por fazer um simples programa em consola, através do qual o utilizador iria comprar os dois ficheiros que lhe despertavam interesse.

Com o executar do programa, é pedido ao utilizador para introduzir o nome do ficheiro que este pretende utilizar como sendo o ficheiro de referência. Assim que introduzido, é apresentada uma de duas mensagens possíveis: uma mensagem de sucesso, na qual é dito ao utilizador que o ficheiro foi aberto; ou uma outra mensagem que diz ao utilizador que o ficheiro que o mesmo introduziu não existe. Neste último caso, o utilizador deve voltar a introduzir o nome de um ficheiro existente.

Após o ficheiro de referencia ter sido aberto, é pedido ao utilizador o nome de um ficheiro a ser comparado. A introdução deste nome segue as mesmas regras e produz os mesmos efeitos do que o ficheiro anterior.

Assim que ambos, o ficheiro de referencia e o ficheiro de comparação , têm sido corretamente introduzidos , é pedido ao utilizador qual considera ser a taxa percentual de igualdade máxima admissível, (o valor introduzido deve varia entre 1 e 100, sendo pedido ao utilizador para introduzir um novo valor no caso de o valor anterior não estar compreendido nesta gama).

Neste momento da execução do programa, os ficheiros são analisados segundo o algoritmo “longest common subsquence”. Assim que terminada a análise, o resultado é apresentado na consola, juntamente com uma mensagem que indica se o valor introduzido pelo utilizador foi ou não ultrapassado.

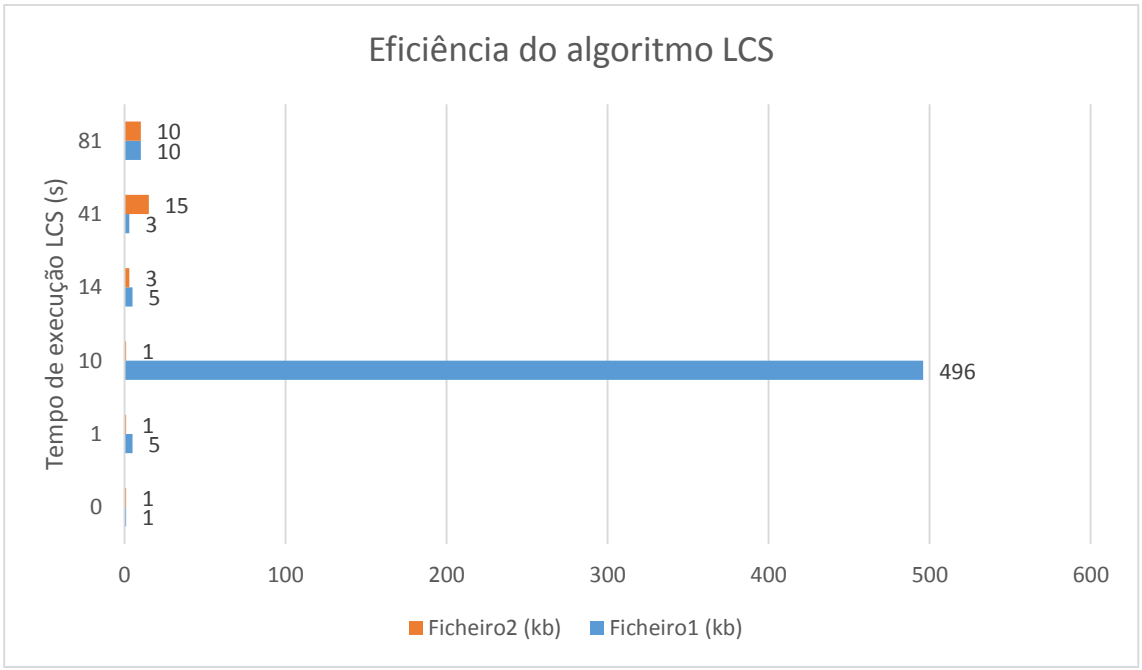
De seguida, os ficheiros são analisados de modo a avaliar e identificar as diferenças entre eles. Sendo o resultado desta avaliação guardado num ficheiro de texto “Output.txt”.

Por fim, o programa pergunta ao utilizador se quer fazer as mesmas verificações para outros ficheiros ou se pretende fechar o programa.

Análise de Complexidade

O algoritmo “longest common subsequence”, apresenta uma elevada complexidade temporal $O\left(2^{n_1}\sum_{i>1} n_i\right)$. , em que n corresponde ao numero de caracteres da sequência.

Nos casos em que são usadas duas sequencias, como por exemplo, na comparação de dois ficheiros distintos feitos, que é feita no programa desenvolvido, a complexidade temporal aproxima-se de O(n*m), em que n é o numero de caracteres de uma sequências, e m o numero de caracteres da segunda sequência.



Como podemos verificar através do gráfico, o tempo de execução vai aumentando de acordo com o tamanho dos ficheiros. Contudo, e ao contrário do que se estaria à espera, um aumento, ainda que bastante elevado, de um dos ficheiros irá provocar um aumento no tempo de execução inferior do que um aumento simultâneo em ambos os ficheiros, ainda que esse aumento seja bastante reduzido.

Isto permite-nos também concluir que a análise de ficheiros através deste algoritmo é algo demorado, mesmo quando se estão a tratar ficheiros com dimensões tão reduzidas.

Conclusão

De modo a concluir, será importante referir que todas as funcionalidades pedidas estão corretamente implementadas à exceção da parte referente às *Track Changes*, que apesar de implementada, se encontra com alguns erros, os quais ainda não conseguimos resolver. À exceção disso, o restante projeto está a funcionar corretamente, apesar das limitações que foram bem visível nesta parte final do relatório, relativas ao algoritmo utilizado.

Em conclusão, os objetivos delineados no inicio do trabalho foram, segundo o nosso ponto de vista, cumpridos. Como ponto a melhorar percebemos que não devemos deixar atrasar tanta quantidade de trabalho para as ultimas semanas de entrega do mesmo, não pelo período de tempo dado mas pela acumulação de trabalhos de outras unidades curriculares que se sobrepuseram.

Bibliografia

- Wikipedia, 1 June 2014, http://en.wikipedia.org/wiki/Diff
- Wikipedia, 1 June 2014, http://en.wikipedia.org/wiki/Levenshtein_distance
- Juan Soulie, “C++ Language Tutorial”, cplusplus.com, http://www.cplusplus.com/doc/tutorial