Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

1º Módulo

Comunicações Digitais

Estudo e aplicação de conceitos fundamentais sobre SCD, teoria de informação e codificação de fonte.

Autores:

48285 Aureliu Iurcu

48349 José Silva

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2021/2022

15/05/2022

Introdução

O objetivo foi o desenvolvimento de programas e aplicações nas linguagens *Python* e *C* a fim de aprofundarmos os estudos e aplicar os conceitos fundamentais sobre os *Sistemas de Comunicação Digital*, teoria de informação e codificação fonte através dos programas e aplicações acima referidos.

Desenvolvimento

- Exercícios 4 e 5 apresentados no guia da primeira aula prática.

Estes exercícios foram realizados em sua maioria nas primeiras duas aulas práticas que tivemos, sendo terminados remotamente fora das aulas.

Houve uma maior facilidade na implementação dos programas em *Python* do que na linguagem *C*, sendo assim, houve mais demora na resolução de tais exercícios.

Para a obtenção da palavra mais frequente contida num ficheiro de teste implementamos a função *most_occurent_word* que retorna um objeto contendo todas as palavras contidas no ficheiro e o seu número de ocorrência, na sua implementação foi usado apenas um ciclo para garantir uma complexidade temporal de *O(n)*. Após os testes realizados, foi concluído que o tempo de execução da função quase que não é influenciado pelo tamanho do ficheiro fonte, pois o tempo de processamento é quase idêntico para os dois tipos de ficheiro. Testamos com os ficheiros 'alice29.txt' e 'a.txt', e não houve uma grande diferença temporal em relação a execução. O que vai de acordo com o pretendido.

As funções para obtenção da sequencia de *fibonacci* e progressão aritmética foram simples de ser implementadas, para testar não usamos nenhum ficheiro, apenas valores aleatórios.

- Fonte de *strings* (*Python*)

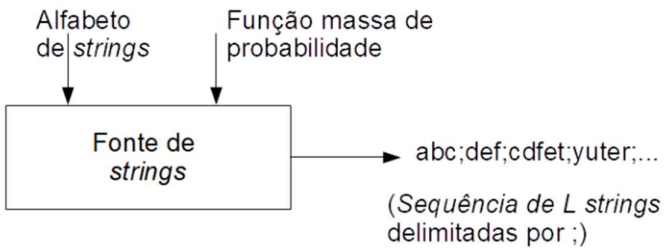

Figura 1 - Fonte de string

a) Implementação da fonte de *string* e apresentação do valor da entropia e o histograma do ficheiro gerado.

Para a implementação da fonte de *string*, implementámos a função *source()* que recebe dois ficheiros (um para o alfabeto de *strings* e outro para a função massa probabilidade) e o tamanho para sequencia gerada, com base na Figura 1.

A tempo de execução da função também é mínimo, apesar de ser *O(n* x *n)*. Igualmente não houve muita diferença entre os dois ficheiros usados como testes, a saber o 'alice29.txt' e 'a.txt'.

Essa quase igualdade é logo colmatada nos valores da entropia, com os mesmos ficheiros de texto, foi concluído que quanto maior e mais palavras tiver um ficheiro, maior será o valor da sua entropia e o tempo do cálculo da entropia, consideravelmente. No primeiro caso, com o de maior tamanho, tivemos uma entropia de 6.535913 bit/símbolo e já no segundo temos 1.098612 bit/símbolo.

Quanto aos histogramas, não há muito que comentar, pois o seu resultado depende da frequência das palavras ou caracteres de um ficheiro de texto.

b) Geração de sequência de símbolos´

Nesta alínea, foram realizadas funções que gerassem *i*) palavras-passe e *ii*) sequência de caracteres alfanuméricos a partir de ficheiros fonte.

Para tal, implementou-se as funções *create_pass_file()* e *create_alphanumeric_sequence()*, para geração de palavra-passe e sequência alfanumérica, respetivamente. Relativamente a sua execução, com sucesso obtemos os resultados pretendidos. As sequências de símbolos são realizadas com os caracteres extraídos dos ficheiros de texto, e não houve muita diferença mesmo quando o tamanho da sequencia fosse maior ou menor.

No processo de testagem da força da palavra-passe, foi utilizado um algoritmo que extraímos de sites na web. Este teste consiste na análise dos caracteres envolvidos e do tamanho da palavra-passe.

c) Geração automática de conteúdos de tabelas a utilizar num sistema de informação

Nesta fase, foram utilizadas muitas das funções implementadas anteriormente, bem como bibliotecas próprias da linguagem em questão.

Conseguiu-se implementar funções que, a partir de ficheiros fontes, gerassem conteúdos de tabelas, *create_citizen_numbers()* para geração automática de números de cidadão aleatórios com tamanho *num_len*; *create_names()*, que gera nomes aleatórios, incluindo mais de um nome próprio e mais de um sobrenome; *create_locations()* que gera *n* localizações; *create_professions()* responsável por gerar profissões aleatórias; e *create_people_out_file()* que cria um ficheiro de texto, sendo que cada linha tem o formato da Figura 2.

```
Número de Cidadão | Nome(s) Próprio(s) e Apelido(s) | Concelho de Residência | Profissão
```

Figura 2 - Forma dos registos das pessoas

Conseguiu-se também a geração das apostas, com a exceção do facto de não termos conseguido terminar o processo de geração aleatória das datas das apostas baseados num ficheiro relativo às datas.

Foi possível constatar, que para cada teste realizado, os tempos de execução foram praticamente os mesmos, não tendo o tamanho do ficheiro influenciado de forma significativa o resultado.

- *Tokens* LZ77 (*Python*/*C*)

Para a geração dos *tokens* foi implementada a função *lz77_tokenizer*, que infelizmente está inconclusa pois não cumpre com todas as regras da geração de *tokens* baseado no teorema LZ77.

Mesmo assim, prosseguimos para a realização dos testes, onde verificamos que o tempo de execução da função supracitada é proporcional ao tamanho dos ficheiros de entrada. Quanto aos valores das entropias dos campos *position* e *length* de um *token*, o seu valor também é proporcional ao tamanho dos ficheiros fontes.

Conclusão

Durante a realização deste trabalho houve muitas dúvidas devido a má extração das informações do enunciado do módulo, bem como a compreensão do mesmo. Após vários esclarecimentos de dúvidas, podemos, com certeza, realizar a maior parte do trabalho em causa. Não conseguimos de todo implementar com sucesso todas as metas, porém garantimos o bom funcionamento das que apresentamos. Foram testadas todas as funções e aplicações realizadas com valores aleatórios, dados os resultados satisfatórios, conseguimos garantir a correta funcionalidade da maior parte trabalho que entregamos, apesar de alguns falharem em casos muitos específicos.