Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

Trabalho Prático Módulo 1

47206 : Tiago Alexandre Figueiredo Pardal (47206@alunos.isel.pt)

48253 : Carlos Guilherme Cordeiro Pereira (A48253@alunos.isel.pt)

Relatório para a Unidade Curricular de Course
da Licenciatura em Engenharia Informática e de Computadores

Professora : Doutor Artur Ferreira

11 – 05 – 2022

Resumo

No âmbito da primeira fase do trabalho prático da cadeira, este relatório tem como propósito a elaboração dos algoritmos pretendidos, e análise dos resultados experimentais.

Através do trabalho realizado pretendemos demonstrar conhecimento sobre a primeira parte da matéria contendo tópicos tais como entropia e codificação e descodificação com técnicas de estatística e de dicionário.

Este relatório parte do pressuposto do acesso por parte do leitor ao código desenvolvido no âmbito do mesmo, não sendo assim necessário enunciá-lo em extensão, bastando apenas mencionar trechos do mesmo.

Índice

Lista de Tabelas

1

String Fountain

1.1 String Fountain

A função geração de fonte com foi implementada com o histograma e a entropia da mesma.

A entropia foi desenvolvida de forma a ser a entropia das strings geradas, e não da fonte.

Não compreendemos precisamente se era esta que deveria ser gerada ou se a da fonte, caso fosse pura e simplesmente a da fonte bastava efetuar o cálculo da entropia para a fmp associada à fonte.

1.2 Geração Passwords

A função de geração das passwords, gera as mesmas a partir da fonte de strings já mencionada.

Na presente implementação só são aceites passwords que contenham símbolos, carateres lower e upper case e dígitos.

Foram gerados 5 ficheiros txt contendo 100 palavras-passe cada um. Estes encontram se no mesmo diretório do código relativo à geração de passwords.

1.3 Avaliação nivel de segurança de palavras passe

Para avaliar o nível das passwords, começa se por verificar se as mesmas cumprem a regra de conterem símbolos, carateres lower e upper case e dígitos, critério este que como já foi mencionada é usado na criação das palavras passe, por isso de certa forma se garantirmos que esta função de avaliação das palavras chave só é o usada para avaliar palavras chaves geradas pelo programa a primeira verificação torna se redundante, no entanto achamos relevante implementa la para verificar palavras passe inseridas ou de qualquer outra origem.

Em seguida o nível das palavras-chave é avaliado consoante a sua entropia:

- Palavra-Passe Fraca - Entropia menor que 3;
- Palavra-Passe Média - 3 >= Entropia > 4;
- Palavra-Passe Forte - Entropia igual e superior a 4;

Foram efetuadas avaliações do nível de segurança das palavras-passe sobre os ficheiros criados, usando a função já mencionada com este efeito.

Fez se uso também de histogramas para avaliar os resultados obtidos.

Os ficheiros como já mencionado contêm 100 palavras-passe cada um. O que os distingue é a dimensão das palavras-passe geradas:

- pass1.txt - Dimensão 4
- pass2.txt - Dimensão 10
- pass3.txt - Dimensão 15 a 16
- pass4.txt - Dimensão 20 a 22
- pass5.txt - Dimensão 30

Os valores mencionados em seguida surgem da média dos valores do histograma de cada ficheiro, em 5 ocorrências diferentes, ou seja para 5 ficheiros diferentes com a mesma dimensão.

No primeiro ficheiro verificou se que todas as palavras-chave são, de acordo com os nossos critérios, fracas, devido à sua entropia ser sempre inferior a 3.

No segundo ficheiro as palavras-passe são todas classificadas como médias.

No terceiro ficheiro cerca de 65% das palavras-chaves são classificadas como médias e as restantes 25% como fortes.

No quarto ficheiro verificámos que quase a totalidade das palavras-passe são classificadas como fortes, existindo apenas 2% que são classificadas como médias com certeza, que devido a aleatoriedade das mesmas estes terem calhado com vários carateres repetidos, e assim terem um valor de entropia menor.

No último ficheiro a totalidade das palavras-passe são classificadas como forte.

Análise nivel de segurança palavras passe

Ficheiros	Dim Password	Password Fracas	Password Médias	Password Fortes
pass1.txt	4	100%	0%	0%
pass2.txt	10	0%	100%	0%
pass3.txt	15 a 0	65%	35%	0%
pass4.txt	20 a 22	0%	2%	98%
pass5.txt	30	0%	0%	100%

Tabela 1.1: Analise Palavras Passe

1.4 Geração automática de sequências alfanuméricas

A função de geração de sequências alfanuméricas, gera as mesmas a partir da fonte de strings já mencionada.

À semelhança das palavaras passe também foram gerados 5 ficheiros contendo cada um 50 sequências alfanuméricas, estes encontram se também no mesmo diretório do código associada a todo este exercício.

1.5 Geração automática de conteúdos de tabelas a utilizar num sistema de informação

Esta função gera dois ficheiros distintos: um com a informação das pessoas e outro com as apostas respetivas.

Servindo como correspondência destas duas o cc da pessoa.

Assim fazemos uso de uma lista global para armazenar todas os cc gerados para um ficheiro, para em primeiro lugar garantirmos que o cc não se repete e em segundo lugar atribuirmos os mesmos cc no ficheiro das apostas.

A função Geração automática de conteúdos de tabelas faz uso dos ficheiros passados e da função String Fountain para gerar o nome composto por 1 ou 2 primeiros nomes e 1 ou 2 apelidos, um concelho e uma profissão.

O cc, as chaves da aposta e o calendário são geradas com o random(), não existindo necessidade de usar a fonte se strings.

2

Código Unário

2.1 Introdução

O objetivo deste exercício era implementar um par codificador/descodificador em modo semi-adaptativo para comprimir dados na linguagem C, ou seja, o ficheiro produzido pelo codificador seria tendencialmente menor que o texto em claro.

Tal com o foi observado devido às características do tipo de codificação nem sempre é possível obter um ficheiro comprimido.

Algumas das ideias presentes neste algoritmo partem do trabalho conjunto e consequente troca de ideias com outros grupos, devido a partilharmos problemas semelhantes de implementação.

2.2 Implementação

De modo a obter a maior taxa de compressão possível e como o exercício pedia foi utilizado o código unário (*comma code*). Foram necessárias 2 passagens no ficheiro original para ser possível de codificar o mesmo:

- A primeira passagem tem como objetivo popular um dicionário com todos os caracteres deste ficheiro bem como as suas ocorrências. Como foi utilizado o código ASCII e como tal apenas e possível de representar 256 caracteres, excluindo assim por exemplo os caracteres latinos.

- Na segunda passagem esta sim já procedeu a codificação do ficheiro original.

No final foi criado um ficheiro com a informação do dicionário bem como o código gerado pela codificação do ficheiro original.

Para descodificar foi apenas necessária uma passagem no ficheiro codificado, para que o codificador saiba que terminou o modelo e começou o ficheiro e último caracter e enviado repetido.

2.3 Compresao de dados

Com a utilização do código unário se o dicionário for considerável, ou seja, se existem diversos caracteres diferentes a sua *performace* é escassa, a quantidade de símbolos e proporcional a ineficácia do algoritmo.

Em particular, se tivermos um ficheiro com todos os caracteres representáveis em código ASCII teríamos um ficheiro 2 vezes maior ao ficheiro original.

Com esta implementação temos um ficheiro de tamanho proporcional ao valor de entropia do ficheiro original.

3

Tokens LZ77

3.1 Introdução

Infelizmente devido à elevada carga de trabalho da semana em que se enquadrava a entrega deste trabalho, com outros 4 trabalhos a terminarem todos num intervalo de 4 dias, a frequência desta cadeira e a falta de aulas práticas devido aos feriados.

Não nos foi possível concluir com o nível de sucesso que pretendíamos este exercício.

Não satisfeitos com isto pretendemos se possível mais tarde voltar ao mesmo e melhora lo de forma que alcance um nível que consideramos satisfatório, o que não se sucede com o presente estado do mesmo.

Além disso algumas das ideias presentes neste algoritmo partem do trabalho conjunto e consequente troca de ideias com outros grupos.

3.2 gerador de tokens

O LZ77 é um codificador baseado em dicionário.

Que analisa o ficheiro com recurso a um dicionário onde vai acrescentando a informação lida, e ao qual faz referência aquando de encontrar algo que já se encontra no seu dicionário.

Os tokens do LZ77 têm o seguinte formato:

(p, l, i)

Em que p - corresponde a posição Em que l - corresponde a dimensão Em que i - corresponde a símbolo inovador

No caso da não existência do símbolo na atual base de dados o token gerado é o seguinte:

$(0, 0, i)$

Em que i é o símbolo a ser inserido no dicionário.

Na presente implementação do gerador de tokens percorremos o LAB da esquerda para a direita e guardamos o com maior dimensão e mais à esquerda possível caso se encontre 2 com a mesma dimensão.

Na presente implementação encontra se implementado o buffer variável, mas não dimensão da janela variável, sendo sempre usado o valor total da dimensão do ficheiro.

Ficou em falta também a realização dos histogramas.

Para a sequência de carateres seguinte obtemos os tokens apresentados em seguida com uma dimensão de buffer de 5:

ABCABCCBACBA

$(0, 0, A)$ lab = ABCABC dicionário vazio

$(0, 0, B)$ lab = BCABCC dicionário = A

$(0, 0, C)$ lab = CABCCB dicionário = AB

$(3, 3, C)$ lab = ABCCBA dicionário = ABC

$(3, 1, A)$ lab = BACBA dicionário = ABCABCC

$(3, 2, A)$ lab = CBA dicionário = ABCABCCBA