



ISEL
INSTITUTO SUPERIOR DE
ENGENHARIA DE LISBOA

Comunicação Digital

Trabalho Prático - Módulo 1

1º Módulo

Grupo

49470 Ana Carolina Pereira
49465 Carolina Tavares
49988 Danilo Vieira

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2022/2023

16/4/2023

Índice

1. INTRODUÇÃO	2
2. EXERCÍCIO 1 - AULA PRÁTICA	3
2.1. EXERCÍCIO 4	3
2.1.1. Alínea a)	3
2.1.2. Alínea b)	3
2.1.3. Alínea c)	4
2.1.4. Alínea d)	4
2.2. EXERCÍCIO 5	5
2.2.1. Alínea a)	5
2.2.1. Alínea b)	5
2.2.1. Alínea c)	5
2.2.1. Alínea d)	6
3. EXERCÍCIO 2 - MÓDULO 1	7
3.1. Alínea a)	7
3.2. Alínea b)	8
4. EXERCÍCIO 3	9
4.1. Alínea a)	9
4.2. Alínea b)	9
4.3. Alínea c)	9
5. Exercício 4	10
5.1.	10
6. Exercício 5	10
6.1. Alínea a)	10
6.2. Alínea b)	10
6.2.1. sub-Alínea i)	10
7. Conclusão	10
REFERÊNCIAS	10

1. Introdução

Este trabalho consiste na realização de várias funções em linguagem C e Python, de forma a que fossem aplicados conhecimentos teóricos na prática destes exercícios e usa os conhecimentos das cadeiras: PSC, ALGA, PG e PE.

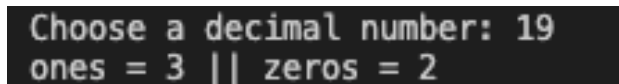
2. Exercício 1

2.1. Exercício 4 - Aula Prática

2.1.1. Alínea a)

A função `count_ones()` recebe um número inteiro `val` como argumento e retorna o número de bits iguais a 1 nesse número. O processo utilizado para contar o número de bits é através do uso de um ciclo `while` e uma operação bit-a-bit com o operador `&` e o valor 1. Dentro do ciclo, enquanto a condição `val != 0` for verdadeira, vai-se verificar se o bit menos significativo do número passado como parâmetro é igual a 1 e incrementa a variável `count`. Posto isto, foi necessário realizar um deslocamento para a direita do valor de `val`. Este processo é repetido até que todos os bits do número sejam testados e o valor de `count` seja retornado.

A função `count_zeros()` que recebe como parâmetro um `int` denominado de '`val`'. Na implementação da função foi criado uma variável de tipo `int`, '`count`', e é criado um `while`, no qual enquanto o valor de '`val`' for diferente de '0', cria-se uma condição `if` dentro da qual se a operação lógica de `val` com o valor '1' for igual a '0', então incrementa-se o valor do `counter` e já fora da mesma condição, realiza-se um `shift right` do '`val`' de 1 bit. O retorno será o valor presente na variável '`count`'.



```
Choose a decimal number: 19
ones = 3 || zeros = 2
```

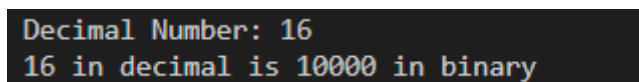
Figura 1: Output da função `count_ones()` e `count_zeros()` para o valor decimal 19.

2.1.2. Alínea b)

A função `print_bits()` tem como objetivo converter um número decimal num número binário. Para realizar esta conversão, é realizado uma divisão sucessiva por 2 e os resultados dos restos de cada divisão são guardados numa variável. A cada iteração do loop `while`, o valor de "`n`" é dividido por 2 e o resultado é guardado novamente. Desta forma o valor binário é construído pela soma sucessiva de potências de base 10 multiplicadas por cada resto obtido.

O valor final da conversão é retornado como um número `long long` sendo, assim, capaz de armazenar números binários com muitos dígitos.

O resultado obtido no output é apresentado na Figura 1.



```
Decimal Number: 16
16 in decimal is 10000 in binary
```

Figura 2: Output da função `print_bits()` para o valor decimal 16.

2.1.3. Alínea c)

A função `most_frequent_symbol` recebe como parâmetro um `char` com o nome do ficheiro e retorna um `int` que representa o número de vezes que o número mais frequente aparece no ficheiro. fizemos a implementação dessa função em `c` e para tal usou-se um `file pointer` que percorre o ficheiro, e coloca num array com tamanho para receber todos os `chars` possíveis, o número de vezes que esse `char` já apareceu, e colocando o `char` como índice do array. atualizando sempre que ele volta a aparecer. e como em simultâneo ficamos sempre a comparar o `char` corrente com o melhor `char` no final é só usar o melhor `char` para buscar no array o número de vezes que ele aparece.

2.1.4. Alínea d)

A função `negative_file` recebe como parâmetros um `'input_file_name'` e `'output_file_name'` ambos dos tipos `char` e retorna `void`. Primeiramente, começa por abrir o `input file` com o nome do valor de `'input_file_name'` e lê o conteúdo do ficheiro e verifica se o ficheiro é possível ser lido, de seguida, abre e cria o `output file`, dando como nome o valor de `'output_file_name'` e verifica se é possível a criação do ficheiro. O passo seguinte será obter o tamanho total do `input file`, e de forma dinâmica alocar um `buffer` com o espaço de memória igual ao tamanho do `input file`. A seguir, o `input file` será colocado no `buffer` através da leitura do mesmo. É então criado um `for` que irá iterar os valores presentes no `buffer`, indo de `index` em `index` do `buffer`, e para cada valor presente no `buffer`, esse mesmo valor é negado. Após a negação do ficheiro, os valores negados são escritos no `output file` e garante que os valores são guardados no ficheiro. Finalmente, a memória anteriormente alocada será libertada e os ficheiros tanto de `input` como `output` serão fechados.

```
int main() {  
    negative_file("a.txt", "out.txt");  
}
```

Figura 4: Demonstração da função `main()` usada para teste

```
ut > ≡ a.txt  
Comunicaçõo Digital - início de ficheiro de teste.  
1234567890  
The quick brown fox jumps over the lazy dog.  
Comunicaçõo Digital - final de ficheiro de teste.
```

Figura 5: Ficheiro usado como `input file`, referenciado na função `main()` com `"a.txt"`

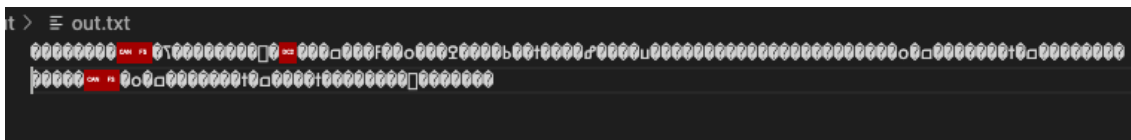


Figura 6: Ficheiro usado como output file, referenciado na função main() com "out.txt"

2.2. Exercício 5 - Aula Prática

2.2.1. Alínea a)

A função `geometric_progression` que recebe os parâmetros denominados de `N`, `u`, `r`, em que '`N`' representa o número do termo da progressão, '`u`' representa o primeiro termo da progressão e '`r`' é a razão da progressão. Na função, é feito um `for` em que enquanto uma variável '`n`' estiver dentro do intervalo entre 1 e `N + 1` a variável '`gp`' será o valor do resultado da operação de '`u`' multiplicado com `r` elevado a '`n - 1`'. No fim da ocorrência do `for`, o valor de '`gp`' é retornado.

$$a_n = a_1 \cdot r^{n-1}$$

Figura 7: Termo Geral de uma Progressão Geométrica

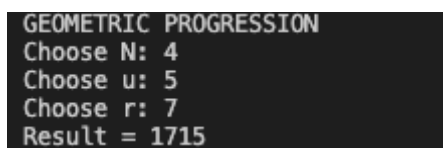


Figura 8: Exemplo do teste dando o valor '`N`' a 4, '`u`' a 5 e '`r`' a 7

2.2.1. Alínea b)

A função que determina o máximo divisor é a `maxDivisor`, que foi escrita em python e recebe dois `int` `a` e `b` passados como parâmetro e é retornado um `int` que representa o máximo divisor entre `a` e `b`. e para implementar esta função usou-se o algoritmo de Euclides que diz que o máximo divisor entre `a` e `b` é igual ao máximo de divisor de `b` e `r` que é o resto da divisão de `a` por `b`, ou seja, fizemos o algoritmo de maneira recursiva e quando o resto dar zero é retornar o '`a`' que começou com `a` mas passou por algumas iterações até chegar ao valor de máximo divisor comum entre `a` e `b`.

2.2.1. Alínea c)

A função `most_least_frequency` recebe o nome de um ficheiro como parâmetro e calcula a frequência de cada caractere, não considerando os espaços em branco.

Para realizar este processo foi necessário importar a biblioteca `collections` de forma a ser possível criar, por exemplo, dicionários. Foi também necessário importar a classe `counter` do módulo `collections` de forma ser possível calcular a frequência dos elementos num dado ficheiro.

Desta forma, com o comando `with open` o ficheiro passado é aberto e é contabilizada a frequência de cada carácter. Por fim, utilizando os métodos `min` e `max`, são devolvidos os caracteres menos frequentes e mais frequentes respetivamente.

2.2.1. Alínea d)

Para esta alínea, o objetivo é a criação de uma função que representa o histograma de um ficheiro, o valor da informação própria de cada símbolo e a entropia do ficheiro “alice29.txt”. Para realizar este objetivo, criaram-se 3 funções, sendo elas `counting_letters`, `plot_bar_from_counter`, `calculate_information_entropy`. A primeira função recebe como parâmetro o filename, sendo este o nome do ficheiro, realizando a abertura e leitura do ficheiro, guardando na variável ‘`letters_count`’ a função `Counter`, que para cada caractere conta as frequências dos mesmos, retirando os espaços e os ‘`enters`’ e retorna a variável ‘`letters_count`’. A segunda função recebe, novamente, o filename como parâmetro e chama a função `counting_letters` dando como parâmetro o filename de `plot_bar_from_counter`, de seguida chama a função `calculate_information_entropy`, que será pontuada mais à frente. Dentro da função `plot_bar_from_counter`, são definidas as barras do histogramas, juntamente com o gráfico da variação e o valor da entropia. A terceira e última função, `calculate_information_entropy`, recebe como parâmetro as frequências dos caracteres, e é responsável pelo cálculo tanto da informação própria como da entropia, retornando ambos os valores.

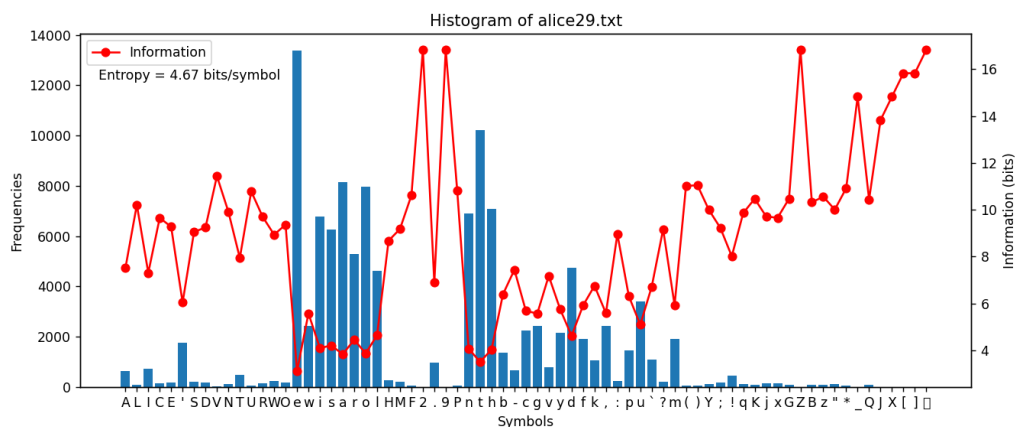


Figura 9: Imagem do histograma do ficheiro “alice29.txt”, com os valores de informação própria e entropia

3. Exercício 2 - Módulo 1

3.1. Alínea a)

Para realizar esta função foi necessário importar alguns módulos como os, math - para realizar vários cálculos, counter do módulo collections - para contar a frequência dos caracteres num dado ficheiro e matplotlib.pyplot de forma a ser possível gerar o histograma.

Numa primeira fase foi criada a função histogram que recebe o nome de um ficheiro como parâmetro. O ficheiro é aberto e a frequência de cada caracter é contabilizada. Posto isto, é calculada a:

- A probabilidade de cada símbolo: dividindo a contagem do símbolo pelo número total de símbolos presentes no ficheiro.
- A informação própria: recorrendo ao uso da fórmula lecionada em aula $-\log_2(\text{probabilidade})$.
- A entropia do ficheiro: somando a multiplicação da probabilidade de cada símbolo pela sua informação própria.

Por fim, cria-se o histograma como se pode ver na Figura 2.

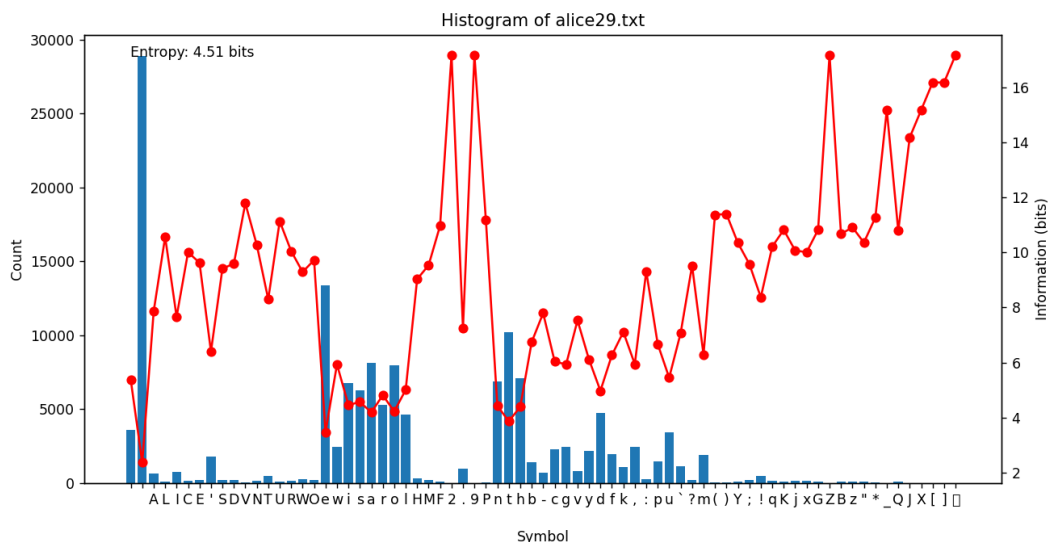


Figura 10: Histograma obtido para o ficheiro "alice.txt"

Na Figura 1, é importante ainda referir que a azul está representada a contagem de cada caracter e a vermelho a informação própria dos mesmos.

3.2. Alínea b)

O objetivo deste exercício é calcular a frequência de símbolos e entropia de dois ficheiros de texto - ListaPalavrasEN.txt e ListaPalavrasPT.txt.

A função `symbol_frequency` recebe um ficheiro como parâmetro e lê esse mesmo ficheiro linha a linha, contando a frequência de cada caracter no ficheiro.

A função `entropy` recebe também um ficheiro como parâmetro e usa a função `symbol_frequency` para calcular a frequência de cada símbolo no ficheiro. Desta forma, é utilizada a fórmula da entropia para calcular a entropia do ficheiro.

<pre>English symbol frequencies: : 9.56% : 0.00% ": 0.00% &: 0.00% ': 0.10% /: 0.00% 0: 0.00% 1: 0.00% 2: 0.00% 3: 0.00% 4: 0.00% 5: 0.00% 6: 0.00% 7: 0.00% 8: 0.00% 9: 0.00% a: 7.36% b: 1.67% c: 3.93% d: 2.91% e: 9.79% f: 1.04% g: 2.16% h: 2.33% i: 8.02% j: 0.14% k: 0.69% l: 5.05% m: 2.69% n: 6.52% o: 6.51% p: 2.94% q: 0.15% r: 6.41% s: 6.61% t: 6.03% u: 3.41% v: 0.87% w: 0.59% x: 0.27% y: 1.83% z: 0.38% English entropy: 4.26</pre>	<pre>Portuguese symbol frequencies: : 7.16% : 0.00% -: 7.38% A: 0.00% B: 0.00% C: 0.00% D: 0.00% E: 0.00% F: 0.00% G: 0.00% H: 0.00% I: 0.00% J: 0.00% K: 0.00% L: 0.00% M: 0.00% N: 0.00% O: 0.00% P: 0.00% Q: 0.00% R: 0.00% S: 0.00% T: 0.00% U: 0.00% V: 0.00% W: 0.00% X: 0.00% á: 1.55% â: 0.00% ã: 0.42% ç: 0.20% è: 0.00% é: 0.04% ê: 0.06% í: 0.57% î: 0.00% ó: 0.03% ô: 0.00% õ: 0.02% ú: 0.01% ü: 0.00% Portuguese entropy: 4.24</pre>
--	---

Figura 11: Output da função `symbol_frequency()`.

4. Exercício 3

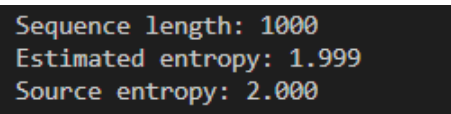
4.1. Alínea a)

A função `gerar_símbolos()` gera uma sequência de símbolos com base numa determinada função massa de probabilidade. Esta função recebe 3 parâmetros, um ficheiro, uma função massa de probabilidade e um `n` que representa o número de símbolos a serem gerados.

Inicialmente a função cria uma lista vazia para armazenar os símbolos gerados. De seguida é inicializado um loop que, usando a função `random.choices`, vai selecionar aleatoriamente um símbolo com a distribuição de probabilidade definida pela função massa de probabilidade.

4.2. Alínea b)

Este exercício teve como objetivo calcular a entropia estimada e comparar como o resultado da entropia da fonte. Desta forma é definida a função `estimate_entropy` em que primeiro é criado um dicionário vazio para armazenar a frequência de cada símbolo na sequência. Desta forma, a função vai iterar sobre cada símbolo e atualiza o valor da contagem de frequência no dicionário. A seguir é calculada a probabilidade de cada símbolo dividindo a contagem de cada símbolo pelo comprimento total da sequência.

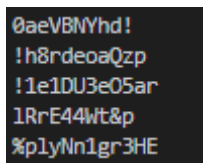


```
Sequence length: 1000  
Estimated entropy: 1.999  
Source entropy: 2.000
```

Figura 12: Resultado obtido após execução da função da alínea b) do exc 3

4.3. Alínea c)

Esta função consistiu em gerar uma palavra passe forte e aleatória. Para isto, foi criada uma string chamada `alfabeto` que contém todos os caracteres que podem ser usados para constituir a password. Tendo nos sido pedido que a palavra passe tivesse entre 8 e 12 caracteres então foi guardado na variável `'tamanho'` um valor random entre esses mesmos dois valores. Desta forma, foi utilizado o método `choices` da biblioteca `random` para escolher caracteres aleatórios da string `alfabeto` criada anteriormente. Por fim, a função é chamada a imprimir 5x - número de passwords pedidas. Num primeiro momento de execução, estas foram as palavras passes geradas:



```
0aeVBNYhd!  
!h8rdeoaQzp  
!1e1DU3e05ar  
lRrE44%t&p  
%plyNn1gr3HE
```

Figura 13: Resultado obtido após execução da função da alínea c) do exc 3

5. Exercício 4

5.1. Alínea a)

Nesta alínea, implementou-se a função `makeVernamCypher`, que recebe de parâmetro `'plainText'` que representa o texto a ser cifrado, e `'theKey'` que representa a chave que será usada para codificar o texto que se pretende codificar. Na implementação da função, criou-se a

variável 'cypherText' que começa por ser uma string vazia, à qual serão adicionados caracteres que resultam da cifra do plainText através de um for com uma variável i que itera os índices dos caracteres do plainText e realiza um xor entre o caractere de índice i da chave com o plainText, transformados para o código ASCII, depois da operação lógica o código ASCII torna-se um char que é adicionado à string 'cypherText'. Quando a iteração do for terminar, 'cypherText' é retornado.

5.2. Alínea b)

A alínea pretendia a cifra do ficheiro de texto "alice29.txt" com uma chave constante e uma chave aleatória de caracteres. Para a implementação utiliza-se as funções presentes nas alíneas 5.1 a) e 2.2.1 Alínea d). Criou-se uma função nova chamada de getAllWords que recebe um filename, correspondente ao nome do ficheiro de input, realiza-se a abertura do ficheiro de entrada e cria dois ficheiros para escrita da codificação com chave constante e chave aleatória. Para cada ficheiro, respetivamente, utiliza-se makeVernamCypher de forma a codificar com as diferentes chaves e escrevê-las nos ficheiros.

Nesta implementação ocorreram erros com a utilização das funções de 2.2.1 Alínea d), por isso não foi possível obter os histogramas, valores de informação própria nem entropias.

6. Exercício 5

6.1. Alínea a)

A função que implementa o modelo Binary Symmetric Channel (BSC) chama-se simulatedBSC, que recebe uma sequência de bits(tipo int) e uma probabilidade de erro(ber)(tipo double)que deve estar entre 0 e 1. a função foi implementada em python, que cria um while que itera pela sequência de bits, dividindo a sequência por dez e recebendo o resto da divisão para como digit que pode tornar-se no errado se o resultado aleatório entre 0 e 1 for menor ou igual ao BER. e a metade se passa por cada bit usa-se a função auxiliar numConcat para concatenar os valores de volta num int que pode ou não conter valores modificados e que esse valor depois de terminar de concatenar é retornado.

6.2 Alínea b)

6.2. 1. Sub-Alínea i)

a função interleaving por escrita em python e recebe uma sequência de caracteres(tipo String) e uma matrizSize que cria uma matriz de tamanho matrizSize x matrizSize, essa matriz é usada para fazer o interleaving que é um método para corrigir/detetar rajadas de erros. para fazer isso colocasse a sequência na matriz criada e envia os valores pela sequência de caracteres que ficam

tirando as pelas colunas, ou seja, enviar o resultado da concentração das colunas começando pela primeira e indo até a última. e é essa concatenação que é o retorno da nossa função. notasse que para fazer o de-interleaving basta colocar o a sequência recebida da função interleaving e passar como parâmetro de novo na função interleaving e colocar mesmo matrizSize.

7. Conclusão

Com este trabalho foi possível aplicar conhecimentos teóricos lecionados nas aulas de forma a perceber o seu contexto na resolução de problemas. Com isto, foi também necessário aplicar conhecimentos de linguagem C e Python uma vez que ainda não tinham sido trabalhados ao longo do curso.