



UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS FLORESTAL

Trabalho Prático III

Projeto e Análise de Algoritmos

Luciano Belo - 3897
Mariana Souza - 3898
Guilherme Correa - 3509

Trabalho Prático apresentado à disciplina de
Projeto e Análise de Algoritmos do curso
de Ciência da Computação da Universidade
Federal de Viçosa.

Florestal
Março de 2022

CCF 330 - Projeto e Análise de Algoritmos

TP 03 - Reino de Hyrule

Luciano Belo - 3897
Mariana Souza - 3898
Guilherme Correa - 3509

22 de Março de 2022

Contents

1	Introdução	2
2	Desenvolvimento	2
2.1	Texto Criptografado	3
2.2	Tipos Abstratos de Dados	3
2.3	Criptoanálise	4
2.4	Boyer Moore	5
2.5	Análise de Frequência	5
2.6	Casamento exato de caracteres	5
2.7	Casamento aproximado de caracteres	6
3	Execução	6
4	Conclusão	8

1 Introdução

O presente Trabalho Prático desenvolvido para a disciplina de Projeto e Análise de Algoritmos tem como objetivo decodificar o texto recebido por Link para assim salvar o Reino de Hyrule que está em perigo, dessa forma decifrando a mensagem com o algoritmo pondendo salvar o reino.

Para um melhor desenvolvimento do grupo foi utilizado o GitHub para realizar o versionamento do código e a colaboração dos integrantes do grupo, a IDE Visual Studio Code para implementação do algoritmo em C e o Notion para uma melhor visualização as funcionalidades a serem implementadas.

2 Desenvolvimento

O algoritmo foi desenvolvido na linguagem C e está dividido em pastas onde a pasta *src* é encontrado os arquivos .c da implementação e a pasta *headers* os arquivos .h, onde o arquivo para apresentar a interface com o usuário estão no "menu.c", contendo a interface para ler o arquivo de entrada desejado. A pasta *data* contém os arquivos de entrada com diferentes textos criptografados sendo que o arquivo **main.txt** é o arquivo enviado pelo monitor da disciplina. Temos a pasta *dist* que contém o executável da aplicação. Também temos a pasta *out* que contém arquivos .txt com os resultados que são exportados na opção 6. Por fim temos a pasta *language* que possui arquivos .txt com as tabelas de frequência da língua portuguesa e inglesa que foram encontrados nos sites [1] e [2] respectivamente. Dessa forma, o trabalho segue a seguinte estrutura de pastas apresentadas abaixo.

```
TP03-PAA
├── data
├── out
├── headers
├── language
├── dist
├── src
│   ├── main.c
│   ├── crypto.c
│   ├── frequency.c
│   ├── key.c
│   ├── menu.c
│   ├── occurrence.c
│   ├── message.c
│   ├── phrases.c
│   └── word.c
```

2.1 Texto Criptografado

Dado o texto criptografado concedido ao grupo, foi realizado sobre esse texto criptografado a implementação das funcionalidades que foram implementadas no algoritmo, estão foi implementada sobre ele a análise de frequência, mostra cada estado da criptoanálise, é possível realizar uma busca no texto criptografado, uma busca no texto parcialmente decifrado, também alterar a chave de criptografia e a última funcionalidade é exportar esse resultado para um arquivo que pode ter seu caminho inserido e armazenar a chave e o texto decifrado.

2.2 Tipos Abstratos de Dados

No TAD "frequency" apresentado no trecho de código abaixo, mostra o uso de duas variáveis usadas para realizar a implementação da frequência de acordo com o texto criptografado, dessa forma foram criados arquivos frequency.h e .c contendo o TAD utilizado para inserir os valores da tabela de frequência, inserir os valores fictícios a tabela da profecia, imprimir as tabelas e ordenar as tabelas usando o algoritmo de ordenação quicksort.

```
typedef struct
{
    char *letter;
    float *value;
} frequency;
```

No trecho de código abaixo é apresentando o Tipo Abstrato de Dado "key" que foi implementado nos arquivos Key.h e .c, contendo a variável letter, usada para preencher e mudar a chave de criptografia, exportar a chave para o arquivo inserido na execução do algoritmo, retornar o total de chaves que não foram preenchidas.

```
typedef struct
{
    char *letter;
} key;
```

O TAD "message" está nos arquivos message.h e .c da implementação do presente trabalho prático, dessa forma foram criados variáveis inteiras para o tamanho e do tipo phase que é importada do outro tipo abstrato de dado, é usada nesse TAD para inserir e imprimir uma frase, além de retornar o total de letras que contém a mensagem.

```
typedef struct
{
    int size;
    phrases *phrases;
} message;
```

O Tipo Abstrato de Dado "phrases" está nos arquivos pharase.h e .c contendo variáveis inteiras para os tamanhos e um tipo word do TAD word, dessa forma esse TAD é utilizado para inserir uma palavra e exibir a frase.

```
typedef struct
{
    int size;
    int start, final;
    word *words;
} phrases;
```

Nos arquivos `word.h` e `.c` são utilizados dois TADs onde o primeiro chamado `typeLetter` com a variável para armazenar a letra e o outro com o nome `word`, contendo os tamanhos e o tipo `typeLetter` como apresentado no trecho de código abaixo retirado do algoritmo implementado para o presente trabalho prático.

```
typedef struct
{
    char letter;
} typeLetter;

typedef struct
{
    int size;
    int start, final;
    typeLetter *letters;
} word;
```

2.3 Criptoanálise

O processo de criptoanálise (funcionalidade 1) é dividida em três partes, primeiramente printamos o texto criptografado, posteriormente printamos a chave atual de criptografia e por fim o texto descriptografado a partir da chave atual.

A descriptografia é feita pela função *decrypt* que recebe o texto criptografado e a chave. Percorremos então toda a estrutura verificando se o caractere é uma letra caso seja e a sua correspondente na chave de descriptografia exista, printamos a letra já trocada e colorida, sendo que cada letra possui uma cor assim como na imagem a seguir:



Figure 1: Cor de cada letra no programa

O processo de troca das letras é feito pela função *positionOfLetterInKey*, que recebe a chave e a letra desejada, caso a letra esteja na chave retornamos sua posição, caso contrário retorna-se `-1`. A ideia desta função é em conjunto com a tabela *ASCII* [3] trocarmos a letra do texto criptografado por sua possível correspondente, isso porque como as chaves correspondem ao alfabeto ao somarmos a posição com o valor **65** (valor da letra 'A') encontramos sua correspondente na tabela *ASCII*. Por exemplo, se nossa chave é **STGB{...}** e quero transformar a letra *S*, a função *positionOfLetterInKey* retorna o valor **0** já que está na primeira posição da chave, sendo assim, ao somarmos 65 com o valor de retorno, temos que a *S* equivale a letra *A*.

A seguir temos uma imagem do texto informado na documentação após a chamada da criptoanálise:

```
=== Texto criptografado ===
R KHXRQ MQZI EXHGQYS BS SOPBS BS EXQZGHYS UHMS ESXS APHTXSX R GRBQLR.

=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ
STGBHVLKQOIMCZREAXYWPDNFJU

=== Texto parcialmente decifrado ===
O HEROI LINK PRECISA DA AJUDA DA PRINCESA ZELDA PARA QUEBRAR O CODIGO.
```

Figure 2: Caso base após criptoanálise

2.4 Boyer Moore

O algoritmo boyer moore [4] faz um pré-processamento sobre o padrão para que o padrão possa ser deslocado por mais de um. Ele processa o padrão e cria diferentes arrays para cada uma das duas Heurísticas (Heurística de Caracter ruim/ Heurística de Sufixo boa).

A cada passo ele desliza o padrão pelo máximo de slides sugeridos por cada uma das Heurísticas. Portanto ele usa o maior deslocamento sugerido e além disso o algoritmo de Boyer Moore inicia a correspondência a partir do último caractere do padrão. A ideia da heurística de caracter ruim é simples, o caractere do texto que não corresponde ao caractere atual do padrão é chamado Bad Character. Em caso de incompatibilidade mudamos o padrão até:

- 1) A incompatibilidade se torna uma correspondência
- 2) O padrão P passa pelo caractere incompatível

2.5 Análise de Frequência

A tabela de frequência foi contado quantas vezes cada letra irá aparecer no texto, apresentando a contagem exibida em ordem decrescente, apresentando a tabela de frequência da profecia e a tabela da língua portuguesa, onde no resultado apresentado abaixo na Figura 3 a letra C foi a mais frequente na tabela da profecia, já na tabela da língua portuguesa a mais frequente foi a letra A. Então para a implementação foram usadas as funções *insertFrequencyValues* para a inserir os valores na tabela na língua deseja e a função *insertDummyValuesFrequencyProphecy* para assim inserir os valores fictícios para a tabela de frequência da profecia.

=== TABELA DE FREQUÊNCIA DA PROFECIA ===			=== TABELA DE FREQUÊNCIA DA LINGUA PORTUGUESA ===	
Letra,	Cont.,	Freq.	Letra,	Freq.
C	159	13.66%	A	14.63%
M	154	13.23%	E	12.57%
P	127	10.91%	O	10.73%
D	121	10.40%	S	7.81%
S	79	6.79%	R	6.53%
H	69	5.93%	I	6.18%
O	69	5.93%	N	5.05%
K	61	5.24%	D	4.99%
R	44	3.78%	M	4.74%
U	42	3.61%	U	4.63%
I	42	3.61%	T	4.34%
X	31	2.66%	C	3.88%
J	28	2.49%	L	2.78%
T	28	2.49%	P	2.52%
B	20	1.72%	V	1.67%
N	18	1.55%	G	1.30%
Z	15	1.29%	H	1.28%
Q	11	0.95%	Q	1.20%
F	10	0.86%	B	1.04%
A	9	0.77%	F	1.02%
G	5	0.52%	Z	0.47%
W	5	0.43%	J	0.40%
Y	2	0.26%	X	0.21%
L	2	0.17%	K	0.02%
V	1	0.09%	Y	0.01%
E	1	0.09%	W	0.01%

Figure 3: Tabela de frequência.

2.6 Casamento exato de caracteres

O casamento exato de caracteres do texto criptografado utiliza do algoritmo boyer moore para fazer a busca do padrão no texto e retorna a quantidade de ocorrências do mesmo no texto.

2.7 Casamento aproximado de caracteres

Para o casamento aproximado de caracteres utilizamos o algoritmo shift and aproximado [5], onde esse faz a busca pelo padrão no texto utilizando da operação de substituição com uma tolerância "k" fornecida pelo usuário e dessa forma retorna quantas ocorrências existem do padrão e também a posição da palavra no texto, além disso apresenta como a palavra se apresenta no texto parcialmente decifrado.

```
Opção: 4
Qual o padrão e a tolerância utilizados?
> QUE 1
Ocorrencias: 14
@[1303 1306): ZUE
```

Figure 4: Casamento aproximado de caracteres.

3 Execução

O trabalho possui um arquivo makefile contendo um conjunto de diretivas usadas para automação de compilação, execução e remoção de arquivos binários e serão apresentados em seguida.

```
all:
    gcc src/main.c src/menu.c src/message.c src/phrases.c src/word.c
    src/frequency.c src/crypto.c src/key.c src/ocurrence.c -o dist/main

run:
    dist/main

clean:
    rm dist/main

clean_exec:
    rm dist/main.exe
```

Foi implementado um menu com as funcionalidades que foram implementadas no trabalho como está apresentado na Figura 5 abaixo com seis opções podendo apresentar o estado da criptografia, fazer a análise de frequência, casamento de caracteres, alterar a chave e exportar o resultado no caminho desejado.

```
printf("\n");
printf("----- Trabalho Pratico 03 -----|\n");
printf("|\n");
printf("|      (1) Apresentar o estado atual da criptoanálise      |\n");
printf("|\n");
printf("|      (2) Fazer análise de frequência no texto          |\n");
printf("|      criptografado                                       |\n");
printf("|\n");
printf("|      (3) Realizar casamento exato de caracteres no     |\n");
printf("|      texto criptografado                                 |\n");
printf("|\n");
printf("|      (4) Realizar casamento aproximado de caracteres   |\n");
printf("|      no texto parcialmente decifrado                   |\n");
printf("|\n");
printf("|      (5) Alterar chave de criptografia;               |\n");
printf("|\n");
printf("|      (6) Exportar resultado e encerrar o programa.     |\n");
printf("|\n");
printf("-----|\n");
```

Figure 5: Menu.

Após a execução do trabalho prático com o texto criptografado dado, podemos encontrar a chave de criptografia apresentada na Figura 7 e o texto decifrado apresentado na Figura 6, onde é possível observar o resultado da execução contendo inicialmente o texto criptografado, posteriormente a chave de criptografia que foi utilizada e as etapas para descriptografar o texto até encontrar a mensagem com o texto decifrado por completo.

```

MKD ZRM C XCJCIKOCOM DM CGCHM ICKD RIC BMF DPGSM QLSRJM. HCUHCD BMFMD C BKOC UMDHC HMSSC APK CIMXCOC, HCUHCD
BMFMD P SMK ICJMAKXP DM PTPD C NSCX C OGD HSMD OMRDCD. TPS BMFMD P TPBP BKBMR CXKIC OPD XMRD, ARNKUOP OGD DPI
GSCD OC DRTMSAKXKM, TPS BMFMD C HMSSC DM CAPNPR MI OKJRBKP, M TPS PRHSCD HCUHCD BMFMD C SMCJKOCOM DM OKDHPSXM
R MUHSM TJCUOP. M CNPSC MDHC HCJBMF DMWC C GCHCJQC AKUCJ. HPOCD CD JKUQCD OP HMITP DM XPJKOKSCP, M HPOPD PD C
SZRKUKIKMPD, RI OKC OMSSPHCOPD, SMHPSUCSCP.

TPSMI, ZRCUOP APK ZRM P SMKUP MDHMBM OMDCITCSCOP? CKUOC ZRM HCSOKP, P QMSPK OP HMITP DMITSM DRSNM ZRCUOP QLSR
JM MDHC MI TMSKNP. MI UPIM OM XCOC CSBPSM M XCOC QCGKHCUM OC AJPSMDHC, MI UPIM OM XCOC SKP, ICS M JCNP TPBPC
OP TMJPD FPSCD, MI UPIM OM XCOC IPUHCUQC NRCSOCOC TMJPD NPSPUD, M TPS HPOCD CD PRHSCD XSKCHRS CD ZRM XPMYKDHMI
MI QCSIPUKC, OMDOM PD HVKJK CHM CD ACOC. TPS HPOPD MDDMD, P QMSPK OP HMITP DMITSM JRHPR M DCKR BKHPSKPPD.

TCSC XPITJMHCS DRC TSPYKIC IKDDCP, JKUE, MDHMMC CHMUHP. XPUHSC HPOCD CIMXC CD DCP MYKNKOPD HPOPD PD SMXRSDPD.
ZRCUOP PD CDHSPD DM CJKUQCSMI, RIC XPUBMSNMUXKC MZRKBCEMUHM OMBMSC DM MSNRMS UC HMSSC: ZRM PD DMHM DCGKPD MDH
MWC MI DMRD TPDHPD XKSXRUDXSKHPD; ZRM PD ZRCHSP NKNCUHM DMWC CXPSOCOPD M XPUBXCOPD; ZRM PD ZRCHSP MDTKSH
PD OC JRF DM CJKUQMI CPD NKNCUHM MI XCOC TPUHP XCSOMCJ; M ZRM CD TMOSCD OGD HSMD OMRDCD MDHMC MI JKNCOC UP X
MUHSP OM HROP. DP CDDKI HPOP P TPHMUXKCJ OM FVJOC DMSC OMDTMSHOP M P OMDHKUP MDHCSC DMJCOP.

=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ
CGXOMANQWJEIUP TZSDH RBVYLF

=== Texto parcialmente decifrado ===
EIS QUE A CALAMIDADE SE ABATE MAIS UMA VEZ SOBRE HYULE. TANTAS VEZES A VIDA NESTA TERÇA FOI AMEACADA, TANTAS
VEZES O REI MALEFICO SE OPÕS A GACIA DAS TRÊS DEUSAS. POR VEZES O POVO VIVEU ACIMA DOS CEUS, FUGINDO DAS SOM
BRAS DA SUPERFÍCIE, POR VEZES A TERÇA SE AFOGOU EM DILUVIO, E POR OUTRAS TANTAS VEZES A REALIDADE SE DISTORCE
U ENTRE PLANOS. E AGORA ESTA TALVEZ SEJA A BATALHA FINAL. TODAS AS LINHAS DO TEMPO SE COLIDIAM, E TODOS OS A
QUINIMIGOS, UM DIA DESTRUTOS, FICAM NA AO.

PORÉM, QUANDO FOI QUE O REINO ESTEVE DESAMPARADO? AINDA QUE TÁ DIO, O HEIO DO TEMPO SEMPRE SURTE QUANDO HYUL
LE ESTA EM PERIGO. EM NOME DE CADA ARVO E E CADA HABITANTE DA FLORESTA, EM NOME DE CADA RIO, MAR E LAGO POVOA
DO PELOS ZONAS, EM NOME DE CADA MONTANHA GUARDADA PELOS GOONS, E POR TODAS AS OUTRAS CRIATURAS QUE COEXISTEM
EM HARMONIA, DESDE OS TWILI ATÉ AS FADAS. POR TODOS ESSES, O HEIO DO TEMPO SEMPRE LUTOU E SAIU VITÓRIOSAMENTE.

PARA COMPLETAR SUA PRÓXIMA MISSÃO, LINK, ESTEJA ATENTO. CONTRA TODAS AMEACAS SÃO EXIGIDOS TODOS OS RECURSOS.
QUANDO OS ASTROS SE ALINHAM, UMA CONVENIÊNCIA EQUIVALENTE DEVERÁ SE EGUIVALENTE NA TERÇA: QUE OS SETE SÁBIOS ESTE
JAM EM SEUS POSTOS CIRCUNSCRITOS; QUE OS QUATRO GIGANTES SEJAM ACONTOADOS E CONVOCADOS; QUE OS QUATRO ESPÍRIT
OS DA LUZ SE ALINHEM AOS GIGANTES EM CADA PONTO CADEAL; E QUE AS PEDRAS DAS TRÊS DEUSAS ESTEJAM LIGADAS NO C
ENTRO DE TUDO. SO ASSIM TODO O POTENCIAL DE XELDA SERÁ DESPERTADO E O DESTINO ESTARÁ SELADO.

```

Figure 6: Texto decifrado.

A	B	C	D	E	F	G	H	I	J	K	L	M
C	G	X	O	M	A	N	Q	K	W	E	J	I

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
U	P	T	Z	S	D	H	R	B	V	Y	L	F

Figure 7: Chave de Criptografia Encontrada.

4 Conclusão

Desta forma, podemos concluir que o trabalho em questão foi desenvolvido conforme o esperado, atingindo todas as especificações requeridas na descrição do mesmo, já que o intuito principal do trabalho foi atingido, a criação de um programa interativo na linguagem C que seja capaz de realizar algumas operações que fazem parte do processo de criptoanálise, e fornecer como saída final a chave de criptografia e o texto decifrado.

Vale ressaltar que, percebemos durante nossas análises como todas as funcionalidades foram necessárias para o processo de criptoanálise, desde as tabelas de frequência em que no caso do grupo as cinco letras de maior frequência do texto criptografado realmente foram as correspondente finais às de maior frequência na língua portuguesa, até os casamentos de cadeias seja exato ou aproximado.

Tivemos algumas dificuldades relacionadas a implementação do algoritmo shift-and e também da interpretação da criptoanálise porém posteriormente conseguimos compreender melhor a lógica e funcionamento do algoritmo e da teoria de criptografia principalmente que todo processo deve ser feito de forma interativa e iterativa.

Posto isso, é válido dizer que apesar das dificuldades na implementação do código, o grupo foi capaz de superar e corrigir quaisquer erros no desenvolvimento dos algoritmos além de implementar tarefas extras como a coloração das letras. Por fim, verificou-se a assertiva para o objetivo do projeto em implementar um programa interativo na linguagem C que seja capaz de realizar algumas operações que fazem parte do processo de criptoanálise, e fornecer como saída final a chave de criptografia e o texto decifrado.

References

- [1] Decifrando textos em português. https://www.gta.ufrj.br/grad/06_2/alexandre/criptoanalise.html. (Accessed on 22/03/2022).
- [2] Decifrando textos em português. https://stringfixer.com/pt/Letter_frequencies. (Accessed on 22/03/2022).
- [3] Tabela ascii. <https://web.fe.up.pt/~ee96100/projecto/Tabela%20ascii.htm>. (Accessed on 22/03/2022).
- [4] boyer-moore. <https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/?ref=gcse>. (Accessed on 18/03/2022).
- [5] Shift and. <https://www2.dcc.ufmg.br/livros/algoritmos/cap8/codigo/c/8.1a8.6e8.8-pesquisacadeia.c>. (Accessed on 20/03/2022).