UNIVERSIDADE FEDERAL DE VIÇOSA - *CAMPUS* FLORESTAL

Fábio Trindade Ramos - mat.3869
Gabriel Vitor da Fonseca Miranda - mat.3857
Victória Caroline Silva Rodrigues - mat.3584

Trabalho Prático 3
Projeto e Análise de Algoritmos

Trabalho prático da disciplina Projeto e Análise de Algoritmos - CCF 330, do curso de Ciência da Computação da Universidade Federal de Viçosa - Campus Florestal.

Professor: Daniel Mendes

Florestal
2022

SUMÁRIO

1. INTRODUÇÃO

Este trabalho tem como objetivo realizar uma criptoanálise clássica de forma interativa na linguagem de programação C para decifrar a profecia que irá restaurar o equilíbrio do reino de Hyrule. Deve ser possível realizar algumas operações do processo de criptoanálise, juntamente com a chave resultante da criptografia e o texto decifrado.

Desse modo, o presente trabalho tem como desafio desenvolver e implementar um algoritmo capaz de decifrar a mensagem secreta no texto utilizando os conceitos vistos em sala de aula sobre criptoanálise.

Outrossim, é importante ressaltar que o texto de frequência de letras da língua portuguesa foi tirada do wikipédia.

2. Desenvolvimento

Para a implementação do trabalho proposto, foi utilizada a linguagem de programação C, juntamente com a ferramenta Git para gerenciar o controle das versões do código. Além disso, com intuito de manter a organização do código e facilitar o processo de análise dos algoritmos, foram criados 2 TADs: a *criptografado,* contendo a chave de criptografia, o texto decifrado e o texto criptografado. O *analiseFrequencia,* que será usada com um vetor de 26 posições onde cada posição representa uma letra do alfabeto, ademais cada posição da estrutura tera uma variável letra, um contador, uma variável para saber a frequência que aquela letra aparece no texto.

3. Implementação

A seguir será apresentado e explicado detalhadamente o funcionamento de cada uma das principais funcionalidades implementadas no trabalho prático. Para melhor visualização, estes foram organizados em subtópicos.

3.1 Estruturas de dados

Para implementar o sistema foi utilizado 2 TADs: *ChaveCriptografada*, e *Frequencia.* A implementação do primeiro TAD pode ser vista na figura 1, e possui os seguintes campos e usos:

- **tam**: número inteiro que armazena o tamanho do texto;
- **chave**: vetor de caracteres que contém o alfabeto em de A-Z, nesta sequência;

- **criptografia**: vetor de caracteres que contém as letras decifradas, sendo que, cada posição corresponde a sua chave no alfabeto, ou seja, se a primeira posição deste vetor possui a letra *S,* então quer dizer que a letra *A* do texto a ser decifrado corresponde a letra *S*.
- **textoDecifrado**: vetor de caracteres que possui o texto completamente ou parcialmente decifrado.
- **textoCriptografado**: vetor de caracteres que armazena o texto criptografado.
- **textoMudanca**: sinaliza que há uma mudança em um carácter do texto a ser decifrado, considerando a sequência A-Z do alfabeto.
- **textoAntigo**: Considerando novamente a sequência A-Z do alfabeto, onde a posição 0 do vetor corresponde a letra A e a posição 25 a letra Z, caso uma letra A seja trocada para a letra B, esse campo armazena na posição 0 a Letra B, já que a posição 0 corresponde a letra A.

Já o TAD *Frequencia* pode ser observado na figura X, e possui as seguintes propriedades:

- **letra:** armazena o carácter que será usado para identificar sua frequência relativa e absoluta.
- **cont:** conta quantas vezes o carácter armazenado no campo letra aparece no texto.
- **freq:** armazena a frequência relativa do carácter no texto.

```
typedef struct{
    int tam;
    char *chave;
    char *criptografia;
    char *textoDecifrado;
    char *textoCriptografado;
    char *textoMundanca;
    char *textoAntigo;

}ChaveCriptografada;
```

Figura 1. TAD *chaveCriptografada*.

```c
typedef struct{
    char letra;
    int cont;
    float freq;
}Frequencia;
```

Figura 2. TAD *frequencia*.

3.2 Análise de frequência no texto criptografado

A estratégia utilizada para essa implementação foi a confecção de uma tabela de frequência obtida a partir da análise do texto que deve ser decifrado. Esta análise consiste em contar a quantidade de caracteres iguais existentes no texto para que se possa obter a porcentagem de cada caracter presente no texto.

Para realizar essa comparação e contagem utilizou-se a tabela ASCII, no qual foram utilizados os números dos caracteres de A até Z, ou seja, o intervalo entre 65 e 90. Dessa forma, cada caractere será transformado em um número inteiro e caso esteja neste intervalo é incrementada a sua respectiva posição da chave, na figura 1 ilustra essa estratégia.

```c
int i = 0;
while(texto[i] != '\0'){
    valor = (int)texto[i];
    if(valor > 64 && valor < 91){
        chave[valor-65]++;
        contletras++;
    }
    i++;
}
```

Figura 3. Contagem dos caracteres presentes no texto.

A partir da contagem dos caracteres é possível realizar o cálculo da porcentagem de cada caracter presente no texto. Como os valores estão ordenados de acordo com o alfabeto do português, foi possível adicionar os resultados em suas perspectivas posições através da estrutura de dados *Frequencia*, como pode-se observar na figura 2. Além disso, a fim de facilitar o manuseio da estrutura optou-se por ordenar de forma decrescente a partir da porcentagem de frequência, por meio da função *Shellsort()*, algoritmo de ordenação.

```c
for(int i = 0; i < 26; i++){
    freq[i].cont = chave[i];
    freq[i].letra = alfabeto[i];
    freq[i].freq = (100) * (freq[i].cont / (double)contletras);
  }
}
Shellsort(freq);
```

Figura 4. Implementação do cálculo de frequência.

Desse modo, obteve-se a tabela de frequência do texto abaixo:

LETRA	FREQUÊNCIA	LETRA	FREQUÊNCIA
I	13.74%	W	2.51%
M	13.31%	Q	1.73%
K	10.98%	C	1.56%
L	10.46%	R	1.30%
U	6.83%	S	0.95%
P	5.96%	G	0.86%
N	5.96%	H	0.78%
X	5.27%	Y	0.52%
B	3.80%	A	0.43%
T	3.63%	O	0.26%
D	3.63%	E	0.17%
J	2.68%	F	0.09%
Z	2.51%	V	0.09%

Tabela 1. Contagem dos caracteres presentes no texto.

A partir desses valores obtidos pode-se ter uma relação entre a tabela de frequência obtida no trabalho e a tabela do idioma português, apresentada abaixo. Dessa forma, pode-se realizar suposições sobre os valores dos caracteres.

LETRA	FREQUÊNCIA	LETRA	FREQUÊNCIA
A	14.63%	**P**	2.52%
E	12.57%	**V**	1.67%
O	10.73%	**G**	1.30%
S	7.81%	**H**	1.28%

R	6.53%	**Q**	1.20%
I	6.18%	**B**	1.04%
N	5.05%	**F**	1.02%
D	4.99%	**Z**	0.47%
M	4.74%	**J**	0.40%
U	4.63%	**X**	0.21%
T	4.34%	**K**	0.02%
C	3.88%	**W**	0.01%
L	2.78%	**Y**	0.01%

Tabela 2. Frequências relativas das letras em português

3.3 Estado atual da criptoanálise

Esta funcionalidade é responsável por mostrar aos poucos a criptoanálise acontecendo a partir das mudanças dos caracteres feitas pelo usuário. Dessa forma, utilizou-se a função *modificarCaracter()* para armazenar os valores alterados e seus valores anteriores da alteração. Além disso, é utilizado um marcador para mostrar quais são as posições que foram alteradas.

```
void modificaCaracter(ChaveCriptografada *chave, char c1, char c2){
    int value = (int)c1;
    chave->criptografia[value - 65] = c2;
    chave->textoMundanca[value - 65] = '*';
    chave->textoAntigo[value - 65] = c1;
}
```

Figura 5. Implementação da função *modificaCaracter()*.

A partir da execução da função *estadoAtualCriptografia(),* que tem como objetivo mostrar o texto criptografado, a chave e o texto parcialmente decifrado Caso não ocorra nenhuma mudança na chave do texto essa deve-se apresentar sem nenhum valor, visto que nenhuma letra foi mapeada no momento. Caso exista alguma mudança no texto, sinalizado

pelo caractere “*”, será buscado no texto o antigo caractere e logo em seguida será inserido o novo caractere. E além disso, será utilizado um vetor de auxiliar para indicar as mudanças realizadas no texto parcialmente decifrado.

```
for(i = 0; i< 26; i++){
    for( j=0;j < chave->tam;j++){
        if(chave->textoMundanca[i] == '*'){
            char t1 = chave->textoAntigo[i];
            char t2 = chave->textoCriptografado[j];
            if(t1==t2){
                chave->textoDecifrado[j] = chave->criptografia[i];
                aux[j] = '^';
            }else{
                if(aux[j]!='^')
                    aux[j] = '-';
            }
        }

    }
}
```

Figura 6. Implementação da mudança do texto decifrado e marcação das suas mudanças.

A estratégia utilizada para mostrar o texto parcialmente decifrado foi mostrar 90 caracteres por linhas, facilitando a visualização das marcações. Logo em seguida, é mostrada a marcação por meio de um vetor auxiliar que foi anteriormente preenchido durante o processo de decifração do texto.

```
for(j= 0; j<chave->tam;j++){
    printf("%c", chave->textoDecifrado[j]);
    if(contador%90==0){
        printf("\n");
        for( k = inicio; k<j+1;k++){
            if(aux[k]=='-')
                printf(" ");
            else
                printf("%c", aux[k]);
        }
        inicio = j+1;
    }contador++;
    if(j==chave->tam-1){
        printf("\n");
        for( k = inicio; k<j;k++){
            if(aux[k]=='-')
                printf(" ");
            else
                printf("%c", aux[k]);
        }
    }
}
```

Figura 7. Implementação do método para mostrar as marcações realizadas.

3.4 Realizar casamento exato de caracteres no texto criptografado

Para se fazer o casamento exato de caracteres no texto criptografado foi necessário criar duas funções a *casamentoPadrao()* e a *casaPalavra()* que se encontram no arquivo *criptografado.c*. A primeira função recebe o texto criptografado e um padrão de palavra que será consultada no texto criptografado, a partir disso é verificando dentro de um while se exite algum caractere do texto criptografado que casa com o primeiro caractere do padrão passado. Logo, se o primeiro caractere do padrão casar com algum caractere do texto criptografado na i-ésima posição é chamada a função *casaPalavra()* que irá verificar se existe um casamento de padrão entre o texto criptografado e a palavra passada, então passada a i-ésima posição, o padrão e o texto criptografado na função *casaPalavra()* ela irá percorrer um for até o tamanho do padrão passado e percorrer a criptografia a partir da i-ésima posição e verificar se existe um casamento de padrão, caso exista a função retorna para a *casamentoPadrao()* com a i-ésima posição atualizada, e se não existe retorna com a i-ésima posição não atualizada. É válido ressaltar que quando é encontrado um casamento de padrão sempre é incrementado +1 em um contador para saber quantas vezes aquele padrão aparece no texto criptografado.Veja abaixo as funcionalidades para se fazer o casamento de padrão.

```
int casamentoPadrao(ChaveCriptografada *chave, char *padrao){
    int i = 0;
    int cont = 0;
    while (i < chave->tam){
        if(chave->textoCriptografado[i] == padrao[0]){
            casaPalavra(chave,  padrao, i, &cont);
        }
        i++;
    }
    return cont;
}
```

Figura 8. Função *casamentoPadrão()*

```
int casaPalavra(ChaveCriptografada *chave, char *padrao, int i, int *cont){
    int antI = i;
    for(int j = 0; j < strlen(padrao); j++){
        if(chave->textoCriptografado[i] == padrao[j]){
            i++;
        }else{
            return antI;
            break;}
    }
    *cont = *cont + 1;
    return i;
}
```

Figura 9. Função *CasaPalavra()*

3.5 Realizar casamento aproximado de caracteres no texto parcialmente decifrado

Para implementar o algoritmo shift-and aproximado, que realiza o casamento aproximado de caracteres com erro de substituição, foram utilizadas outras 3 funções auxiliares: *calcularR0Linha, calcularRjLinha* e *shift.* A função *shift* é responsável por deslocar os bits de um vetor para a direita, atribuindo o bit 0 a sua primeira posição. As outras duas funções são responsáveis por calcular os novos valores de bits de cada vetor Rj a medida que as letras são lidas. Vale lembrar que cada vetor Rj é utilizado para verificar se houve casamento de bits de acordo com o seu respectivo erro. Os cálculos dos novos valores são feitos através da operação de shift-and em um respectivo vetor de bits Rj, sendo j equivalente a quantidade de erros, e $0 < j \leq k$, sendo k a quantidade de erros. As operações shift-and utilizam das seguintes fórmulas:

- R'0 = ((R0 >> 1) | 10^{m-1}) & M[T[i]]
- R'j = ((Rj >> 1) & $M[T[i]]$) | Rj-1 >> 1 | 10^{m-1}

Pelas fórmulas acima temos que m é o tamanho do padrão a ser analisado e M[T[i]] é a máscara de bit de um caractere do padrão a ser analisado. A máscara de uma letra possui bit um onde está sua localização na palavra e 0 nos demais. Como exemplo, ao procurar por um padrão *TESTE* em um texto, seria construído a seguinte máscara de bit:

- T: 10010
- E: 01001
- S: 00100

Vale dizer que os vetores R0 e Rj são inicializados através da fórmula:

- R0 = 0^m
- Rj= $1^j 0^{m-j}$

Na figura 10 podemos observar a implementação deste algoritmo.

```
void shiftAndAproximado(ChaveCriptografada* chaveCriptografada,char* padrao, int tolerancia){
    printf("%s",chaveCriptografada->textoDecifrado);
    int tamPadrao=strlen(padrao);
    char* letrasMascaras;
    int casou=0;
    int i=0, j=0, k =0;
    int mascara0[tamPadrao];
    int ocorrencias=0;

    for(i=0;i<tamPadrao;i++){
        mascara0[i]=0;
    }
    letrasMascaras=calcMascaras(padrao);

    int qtdMascaras=strlen(letrasMascaras);
    int** binMascaras=  (int**)malloc(qtdMascaras*sizeof(int*));
    for(i=0;i<qtdMascaras;i++){
        binMascaras[i] = (int*)malloc(tamPadrao*sizeof(int));
    }
    inicializarBinMascaras(binMascaras,letrasMascaras,padrao);

    int** rs = (int**)malloc((tolerancia+1)*sizeof(int*));
    int** rsLinha = (int**)malloc((tolerancia+1)*sizeof(int*));

    for(i=0;i<tolerancia+1;i++){
        rs[i] = (int*) malloc(tamPadrao*sizeof(int));
        rsLinha[i] = (int*) malloc(tamPadrao*sizeof(int));
    }
    inicializarRs(rs,tolerancia+1,tamPadrao);
    i=0;
```

Figura 10. Função *shiftAndAproximado()*

```
while (chaveCriptografada->textoDecifrado[i]!='\0')
{

    int* binCaracter=(int*)malloc(tamPadrao*sizeof(int));
    char c= chaveCriptografada->textoDecifrado[i];
    for(j=0;j<qtdMascaras;j++){
        if(c==letrasMascaras[j]){
                binCaracter=binMascaras[j];
            break;
        }else{
                binCaracter=mascara0;
        }
    }


    for(j=0;j<tolerancia+1;j++){
        if(j==0){
            rsLinha[j]=calcularR0Linha(rs[0],binCaracter,tamPadrao);
        }else{
            rsLinha[j]=calcularRjLinha(rs[j],rs[j-1],binCaracter,tamPadrao);
        }
        }


        for(j=0;j<tolerancia+1;j++){
         for(k=0;k<tamPadrao;k++){
                rs[j][k]=rsLinha[j][k];
        }
        }
```

Figura 11.

```
        for(j=0;j<tolerancia+1;j++){
            if(rsLinha[j][tamPadrao-1]==1){
                casou=1;
                break;
            }
        }

        if(casou){
            ocorrencias++;
              printf("@[%d,%d): ",i-tamPadrao+1,i+1);
            for(j=i-tamPadrao+1;j<=i;j++){
            printf("%c",chaveCriptografada->textoDecifrado[j]);
        }
            printf("\n");
            casou=0;
        }
        i++;
    }
     printf("Quantidade de ocorrencias: %d",ocorrencias);

}
```

Figura 12. Algoritmo que realiza o casamento aproximado()

3.6 Alterar chave de criptografia

Para se alterar a chave de criptografia foi algo bem simples, bastou saber qual o caracter que seria alterado, então foi verificado qual seria seu valor inteiro na tabela ASC, e colocado na i-ésima posição do vetor chave→criptografia[], e então naquela posição - 65, foi colocado o valor que representa um determinado valor do vetor chave→chave[]. Veja abaixo a alterar caracteres:

```
void modificaCaracter(ChaveCriptografada *chave, char c1, char c2){
    int value = (int)c1;
    chave->criptografia[value - 65] = c2;
    chave->textoMundanca[value - 65] = '*';
    chave->textoAntigo[value - 65] = c1;
}
```

Figura 13. Função modificaCaractere()

Vale ressaltar que ao alterar uma chave, a alteração da letra corresponde a troca da letra no texto parcialmente decifrado, ou seja, no texto “PEROI” se formos supor que a letra seja igual a letra H então teríamos que inserir nesta funcionalidade primeiro a letra P e depois a letra H.

3.7 Exportar resultado e encerrar o programa

Para exportar o resultado de um programa é necessário que o usuário insira dois caminhos para arquivos com extensão *txt*. Cada caminho deve ser inserido de acordo com o seguinte padrão: *nomeArquivo.txt.* Será exportado para o primeiro arquivo as chaves mapeadas com seus valores, e para o segundo arquivo será exportado o texto decifrado. Vale lembrar que ao utilizar esta opção, será enviado as chaves que foram encontradas nas suposições feitas e o texto decifrado a partir delas. Os arquivos são exportados para a pasta raíz do projeto.

3.8 Execução

Para fazer a execução do programa no Linux, basta abrir o terminal (ctrl+c) e digitar o comando: **make**. E para executar o programa no Windows basta digitar o seguinte comando:**gcc main.c -o run sources/analiseFrequencia.c sources/criptografado.c**, e logo em seguida digitar: **run.exe**.

4. Conclusão

A seguir serão apresentados, respectivamente, os resultados encontrados a partir da execução das funções descritas na aba Implementação e as considerações finais a respeito deste trabalho prático.

4.1 Resultados

Estado atual da criptoanálise e Mudança de caracteres:

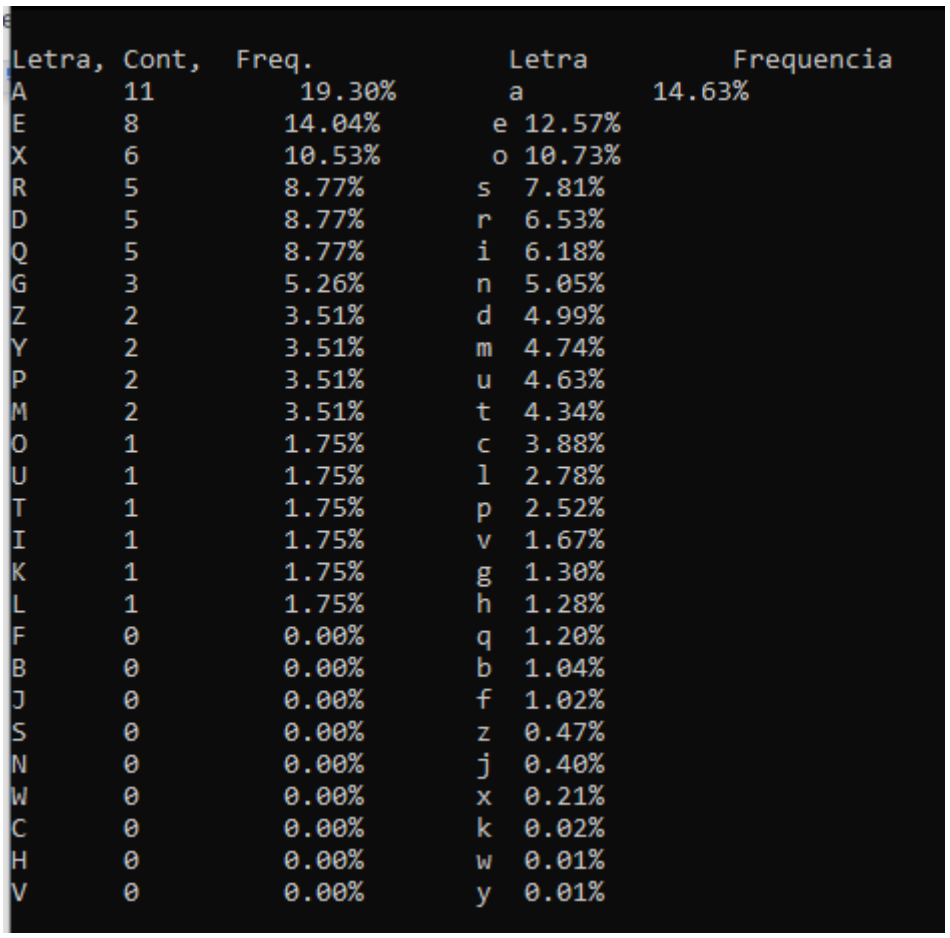
Estado atual da criptoanálise com setas apontando os caracteres que acabaram de ser modificados na opção 5.

```
=== Texto criptografado ===
R KEXRQ MQZI EXEGQYA DA AOPDA DA EXQZGEYA UEMDA EAXA APETXAX R GRDQLR.


=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ
S   H  E          A

==== Texto parcialmento decifrado ===
R KHXRQ MQZI HXHGQYS DS SOPDS DS HXQZGHYS UHMDS HSXS SPHTXSX R GRDQLR.

   ^         ^ ^   ^  ^ ^   ^  ^ ^    ^ ^  ^  ^ ^^ ^ ^ ^  ^
```

Figura 14. Modificações do texto parcialmente decifrado.

Tabela de Frequência:

Esta tabela de frequência apresenta do seu lado esquerdo a frequência de cada letra no arquivo de entrada, e ao lado direito apresenta a frequência de letras na língua portuguesa.



Letra,	Cont,	Freq.	Letra	Frequencia
A	11	19.30%	a	14.63%
E	8	14.04%	e	12.57%
X	6	10.53%	o	10.73%
R	5	8.77%	s	7.81%
D	5	8.77%	r	6.53%
Q	5	8.77%	i	6.18%
G	3	5.26%	n	5.05%
Z	2	3.51%	d	4.99%
Y	2	3.51%	m	4.74%
P	2	3.51%	u	4.63%
M	2	3.51%	t	4.34%
O	1	1.75%	c	3.88%
U	1	1.75%	l	2.78%
T	1	1.75%	p	2.52%
I	1	1.75%	v	1.67%
K	1	1.75%	g	1.30%
L	1	1.75%	h	1.28%
F	0	0.00%	q	1.20%
B	0	0.00%	b	1.04%
J	0	0.00%	f	1.02%
S	0	0.00%	z	0.47%
N	0	0.00%	j	0.40%
W	0	0.00%	x	0.21%
C	0	0.00%	k	0.02%
H	0	0.00%	w	0.01%
V	0	0.00%	y	0.01%

Figura 15. Tabela de frequência obtidas pelo grupo e do idioma português.

Casamento exato:

Para o arquivo de teste que se encontra na pasta entradas “entradas/teste.txt”, é verificado se existe um casamento exato de um padrão passado pelo usuário no arquivo de texto de entradas. Veja o exemplo abaixo:

```
=== Texto criptografado ===
DADOS PAA PAA PAA MANO SAPIENS PAA

=== Chave ===
ABCDEFGHIJKLMNOPQRSTUVWXYZ


==== Texto parcialmento decifrado ===
DADOS PAA PAA PAA MANO SAPIENS PAA


==================================================
=======================MENU=======================
==================================================
[1] - Apresentar o estado atual da criptoanalise
[2] - Fazer analise de frequencia no texto criptografado
[3] - Realizar casamento exato de caracteres no texto criptografado
[4] - Realizar casamento aproximado de caracteres no texto parcialmente dec
[5] - Alterar chave de criptografia
[6] - Exportar resultado e encerrar o programa
[0] - Sair
Opcao: 3

Digite o padrao:PAA
Qual o padrao utilizado?
> PAA
Ocorrencias:4
```

Figura 16. Resultado obtido do casamento exato de padrão.

Realizar casamento aproximado de caracteres

O arquivo no que se encontra no caminho “entradas/teste.txt” foi utilizado como arquivo de teste para esta funcionalidade. Ao pesquisar pela palavra SAPOESS com uma tolerância igual a 3, podemos verificar diretamente no arquivo que é de se esperar que a palavra SAPIENS seja encontrada. A imagem X apresenta o resultado obtido pelo sistema após este teste. Vale destacar que nenhuma suposição foi feita para realizar este teste.

```
==================================================
=======================MENU=======================
==================================================
[1] - Apresentar o estado atual da criptoanalise
[2] - Fazer analise de frequencia no texto criptografado
[3] - Realizar casamento exato de caracteres no texto criptografado
[4] - Realizar casamento aproximado de caracteres no texto parcialmente decifrado
[5] - Alterar chave de criptografia
[6] - Exportar resultado e encerrar o programa
[0] - Sair
Opcao: 4

Digite o padrao e tolerancia utilizados?
SAPOESS 3
@[24,31): SAPIENS
Quantidade de ocorrencias: 1
```

Resultado da descriptografia do arquivo fornecido pelo monitor

Local do arquivo: “entradas/texto-criptografado-7.txt”

Texto descriptografado:

```
EIS QUE A CALAMIDADE SE ABATE MAIS UMA VEZ SOBRE HYRULE. TANTAS VEZES A VIDA NESTA TERRA FOI AMEACADA,
TANTAS VEZES O REI MALEFICO SE OPOS A GRACA DAS TRES DEUSAS. POR VEZES O POVO VIVEU ACIMA DOS CEUS,
FUGINDO DAS SOMBRAS DA SUPERFICIE, POR VEZES A TERRA SE AFOGOU EM DILUVIO, E POR OUTRAS TANTAS VEZES A
REALIDADE SE DISTORCEU ENTRE PLANOS. E AGORA ESTA TALVEZ SEJA A BATALHA FINAL. TODAS AS LINHAS DO TEMPO SE COLIDIRAO,
E TODOS OS ARQUINIMIGOS, UM DIA DERROTADOS, RETORNARAO. POREM, QUANDO FOI QUE O REINO ESTEVE DESAMPARADO? AINDA QUE
TARDIO, O HEROI DO TEMPO SEMPRE SURGE QUANDO HYRULE ESTA EM PERIGO. EM NOME DE CADA ARVORE E CADA HABITANTE DA FLORESTA,
EM NOME DE CADA RIO, MAR E LAGO POVOADO PELOS ZORAS, EM NOME DE CADA MONTANHA GUARDADA PELOS GORONS,
E POR TODAS AS OUTRAS CRIATURAS QUE COEXISTEM EM HARMONIA, DESDE OS TWILI ATE AS FADAS. POR TODOS ESSES,
O HEROI DO TEMPO SEMPRE LUTOU E SAIU VITORIOSO. PARA COMPLETAR SUA PROXIMA MISSAO, LINK, ESTEJA ATENTO.
CONTRA TODAS AMEACAS SAO EXIGIDOS TODOS OS RECURSOS. QUANDO OS ASTROS SE ALINHAREM,
UMA CONVERGENCIA EQUIVALENTE DEVERA SE ERGUER NA TERRA: QUE OS SETE SABIOS ESTEJAM EM SEUS POSTOS CIRCUNSCRITOS;
QUE OS QUATRO GIGANTES SEJAM ACORDADOS E CONVOCADOS; QUE OS QUATRO ESPIRITOS DA LUZ SE ALINHEM AOS GIGANTES EM
CADA PONTO CARDEAL; E QUE AS PEDRAS DAS TRES DEUSAS ESTEJAM LIGADAS NO CENTRO DE TUDO. SO ASSIM TODO O POTENCIAL DE
ZELDA SERA DESPERTADO E O DESTINO ESTARA SELADO.
```

Figura 17. Resultado obtido da descriptografia realizada.

Chave:

```
ABCDEFGHIJKLMNOPQRSTUVWYZ
JUGMYKZFACOSEDXTVQHNRWLIBP
```

Figura 18. Chaves obtidas da criptoanálise.

4.2 Considerações finais

Neste trabalho prático foi explorado um tema de grande importância para os estudos, principalmente na disciplina de Projeto e Análise de Algoritmos que foi o estudo sobre criptografia e criptoanálise. A criptografia e a criptoanálise são as duas faces da criptologia, que é o estudo dos “códigos secretos” ou “cifras”. A criptografia moderna usa técnicas matemáticas cada vez mais sofisticadas e desempenha um papel crucial em muitas atividades do nosso cotidiano, ao proteger dados confidenciais, assegurando a identificação de interlocutores e a integridade de dados.

Logo, uma cifra consiste num procedimento que transforma um texto num outro, o criptograma, que se pretende ilegível para quem não possua um pedaço de informação (mantido secreto) a que se chama a chave. Essa transformação pode, por exemplo, substituir cada letra da mensagem original por outra letra, eventualmente de um outro alfabeto. Estas cifras de substituição são formadas por funções que fazem corresponder a cada caractere de um alfabeto um (outro) determinado caracter do mesmo (ou de outro) conjunto de símbolos

Outrossim, o objetivo principal deste trabalho foi criar estruturas de dados para se fazer uma análise de um texto criptografado e possivelmente achar a sua chave de criptografia, isto mostrando uma tabela de frequências da língua portuguesa. Ou seja, se existe uma letra n que aparece com mais frequência em um texto criptografado esta letra seria considerada a letra A na linha portuguesa, já que esta tem a maior frequência em textos.

Portanto, esta abordagem mais prática da implementação de um programa de criptoanálise, melhorou ainda mais os conhecimentos adquiridos em sala de aula, pois fez com que conceitos dos algoritmos que foram apresentados em sala fossem utilizados para a implementação dessa estrutura.

5. Referências

[1] Machiavelo, Reis. Criptografia e Criptanálise. Casa das Ciências. Disponível em: Link. Acesso em: 22 de Março de 2022.

[2] Frequência de letras. Wikipedia, 2020. Disponível em: Link. Acesso em: 17 de Março de 2022.