



SERVIÇO PÚBLICO FEDERAL - MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA - UFV
CAMPUS FLORESTAL

Trabalho prático 3 - Projeto e análise de algoritmos.

Bruno Vicentini Ribeiro - 5907
Erich Pinheiro Amaral - 5915
Fabrício Henrique Viana Albino - 5925
Pedro Paulo Paz do Nascimento - 5937
Vitor Mathias Andrade - 5901

Florestal - MG
2025

Sumário

Introdução	3
Organização	3
Desenvolvimento	4
Compilação e execução	9
Resultados	10
Conclusão	16
Referências	16

Introdução

O presente trabalho aborda a implementação de um sistema interativo em linguagem C voltado para a simulação de um processo de criptoanálise baseado em substituição alfabética. A proposta integra conteúdos estudados na disciplina de Projeto e Análise de Algoritmos, permitindo aplicar conceitos fundamentais relacionados à análise de frequência, técnicas de casamento exato e aproximado de padrões e manipulação incremental de chaves.

O problema é contextualizado por meio da narrativa das profecias dos “Herdeiros de Chrysos”, na qual o grupo recebe um texto claro que deve ser inicialmente criptografado utilizando uma cifra de deslocamento. A partir do texto cifrado produzido, o sistema desenvolvido tem como objetivo auxiliar o usuário na reconstrução da chave e na recuperação do texto original, oferecendo recursos como visualização do estado atual da decifragem, sugestões baseadas em frequências, buscas por padrões no texto cifrado ou parcialmente decifrado e ajustes manuais da chave.

A implementação proposta permite consolidar, de forma prática, o entendimento dos algoritmos explorados ao longo da disciplina, entre eles, KMP, Shift-And e sua variante aproximada, além de reforçar a importância do uso adequado de estruturas de dados, organização modular e interação com arquivos. Assim, o trabalho se fundamenta na união entre narrativa, técnicas clássicas de criptoanálise e ferramentas de análise de algoritmos, tendo como objetivo final a reconstrução completa da chave de substituição aplicada ao texto.

Organização

Na figura 1, observa-se o panorama geral da organização do projeto, no qual estão elencados os arquivos necessários para o trabalho. Esses arquivos foram separados em pastas para garantir melhor organização e funcionamento do código.

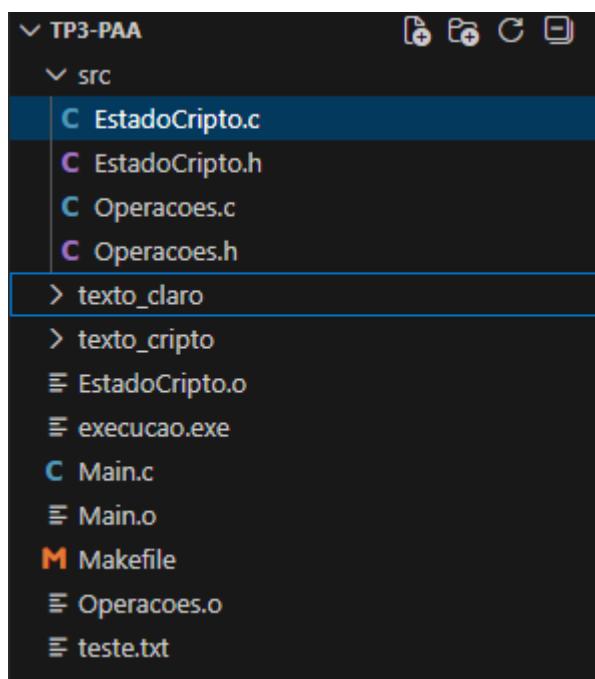


Figura 1: Repositório do projeto.

Desenvolvimento

- Estrutura de dados:

Para que seja possível executar todas as funções previstas, foi necessário criar duas estruturas de dados: “EstadoCripto” (figura 2), e “Frequencia” (figura 3).

```
typedef struct {
    char alfabetoOriginal[TAMANHO_ALFABETO + 1];
    char chaveDecifracao[TAMANHO_ALFABETO + 1];
    char textoCifrado[TAMANHO_MAX_TEXTO];
    char textoParcial[TAMANHO_MAX_TEXTO];
    int shiftCriptografia;
} EstadoCripto;
```

Figura 2: Estrutura “EstadoCripto”.

A estrutura “EstadoCripto” armazena um vetor com o alfabeto original, um vetor com a chave para decifrar, um vetor com o texto decifrado e o texto parcialmente decifrado, além de armazenar qual foi o valor do “shift” de deslocamento.

```
typedef struct {
    char letra;
    int quantidade;
    double frequencia;
} Frequencia;
```

Figura 3: Estrutura “Frequencia”.

A estrutura “Frequencia” armazena a letra que será avaliada a frequência, a quantidade de vezes que ela aparece, além do valor que será exibido em porcentagem.

- Inicialização e normalização do texto:

A primeira etapa do sistema consiste na preparação do ambiente de análise. Ao carregar um arquivo de texto claro, o programa não apenas realiza a leitura, mas aplica um processo de normalização. Caracteres acentuados (como “á”, “õ” e “ç”) são convertidos para seus equivalentes não acentuados e todo o texto é transformado para letras maiúsculas. Isso garante que a criptografia subsequente opere sobre um alfabeto padrão de 26 caracteres, simplificando a análise estatística sem perda de generalidade.

```

int normalizarAcentuacao(FILE* f){
    int c = fgetc(f);
    if (c == EOF) return EOF;

    if (c == 0xC3) {
        int prox = fgetc(f);

        switch (prox) {
        case 0x80: case 0x81: case 0x82: case 0x83: // à, Á, Â, Ã
        case 0xA0: case 0xA1: case 0xA2: case 0xA3: // à, á, â, ã
            return 'A';

        case 0x89: case 0x8A: // É, Ê
        case 0xA9: case 0xAA: // é, ê
            return 'E';

        case 0x8D: // Í
        case 0xAD: // í
            return 'I';

        case 0x93: case 0x94: case 0x95: // ó, ô, Õ
        case 0xB3: case 0xB4: case 0xB5: // ó, ô, õ
            return 'O';

        case 0x9A: // Ú
        case 0xBA: // ú
            return 'U';

        case 0x87: // ç
        case 0xA7: // ç
            return 'C';

        default:
            break;
        }
    }

    if (isalpha(c)) return toupper(c);

    return c;
}

```

Figura 4: Normalização do texto.

- Análise de frequência:

Uma das principais ferramentas implementadas para quebrar a cifra é a análise de frequência. O algoritmo conta a incidência de cada caractere no texto cifrado local e, para aumentar a precisão, realiza também uma contagem global considerando todos os 12 arquivos de profecia disponíveis, assumindo que todos foram cifrados sob a mesma lógica.

Os dados coletados são comparados com a tabela de frequência padrão da língua portuguesa. O sistema então sugere um "chute" inicial de mapeamento, associando as letras mais frequentes do texto cifrado (como “A”, “E” e “O”) às letras mais comuns do idioma, oferecendo ao usuário um ponto de partida estatisticamente embasado para a decifração.

A função que é responsável por fazer a análise da frequência e o “chute” de mapeamento é a “frequenciaCaracter” (figura 5).

```

void frequenciaCaracter() {

    int alfabeto[26] = {0};
    int totalLetras = 0;

    for (int i = 0; estadoAtual.textoCifrado[i] != '\0'; i++) {
        unsigned char c = (unsigned char)estadoAtual.textoCifrado[i];

        if (c >= 'A' && c <= 'Z') {
            int indice = c - 'A';
            alfabeto[indice]++;
            totalLetras++;
        }
    }

    Frequencia listaOcorrencia[26];
    for (int i = 0; i < 26; i++) {
        listaOcorrencia[i].letra = 'A' + i;
        listaOcorrencia[i].quantidade = alfabeto[i];

        if (totalLetras > 0) {
            listaOcorrencia[i].frequencia =
                ((double)listaOcorrencia[i].quantidade / totalLetras) * 100.0;
        } else {
            listaOcorrencia[i].frequencia = 0.0;
        }
    }

    for (int i = 0; i < 25; i++) {
        for (int j = 0; j < 25 - i; j++) {
            if (listaOcorrencia[j].quantidade < listaOcorrencia[j + 1].quantidade) {
                Frequencia temp = listaOcorrencia[j];
                listaOcorrencia[j] = listaOcorrencia[j + 1];
                listaOcorrencia[j + 1] = temp;
            }
        }
    }

    printf("\n==== Frequencia LOCAL do arquivo ====\n");
    printf("Letra | Qtde | Freq(%)\n");
}

```

Figura 5: Função “frequenciaCaracter”.

- Casamento:

Para refinar a chave sugerida pela frequência, foram implementados algoritmos eficientes de busca textual, permitindo ao usuário encontrar palavras ou fragmentos suspeitos no texto cifrado:

- Casamento exato (Shift-And): para buscas precisas, utilizou-se o algoritmo Shift-And. Este método tira proveito das operações paralelismo de bits do processador. O padrão de busca é pré-processado em máscaras de bits, e o estado da busca é mantido em uma variável inteira que é atualizada a cada caractere lido. Isso permite verificar a ocorrência do padrão com extrema rapidez, essencialmente em tempo linear (figura 6).

```

void casamentoExato(char* padrao) {
    //Algoritmo Shift-And
    int m = strlen(padrao);
    int n = strlen(estadoAtual.textoCifrado);

    if (m > 64) {
        printf("Erro: O padrão é muito longo para o algoritmo Shift-And (max 64 caracteres).\n");
        return;
    }

    unsigned long M[256]; //Tabela M

    for (int i = 0; i < 256; i++) {
        M[i] = 0;
    }
    for (int j = 0; j < m; j++) {
        unsigned char c = (unsigned char)padrao[j];
        M[c] |= (1UL << j);
    }

    unsigned long R = 0;
    int ocorrencias = 0;

    for (int i = 0; i < n; i++) {
        unsigned char c = (unsigned char)estadoAtual.textoParcial[i];

        R = ((R << 1) | 1UL) & M[c];

        if (R & (1UL << (m - 1))) ocorrencias++;
    }

    printf("\nOcorrencias: %d\n", ocorrencias);
}

```

Figura 6: Algoritmo de casamento exato (Shift-And).

- Casamento aproximado: reconhecendo que o texto pode estar apenas parcialmente decifrado ou conter erros, implementou-se uma variação do Shift-And capaz de tolerar erros de substituição. O algoritmo mantém múltiplos vetores de estado, onde cada vetor $R[j]$ representa o casamento do padrão com até “j” erros. Dessa forma, é possível localizar palavras que “quase” correspondem ao padrão desejado, facilitando a identificação de palavras onde apenas algumas letras ainda estão cifradas incorretamente (figura 7). Além disso, após a inserção do padrão e da quantidade de erros tolerados, são exibidas as posições de cada ocorrência e a quantidade total de ocorrências.

```

void casamentoAproximado(char* padrao, int k) {
    int m = strlen(padrao); // é o tamanho do padrão
    int n = strlen(estadoAtual.textoParcial); // é o tamanho do texto decifrado

    if (m > 64) {
        printf("Erro: O padrao e muito longo (max 64 caracteres).\n");
        return;
    }
    char* textoDecifradoTemp = (char*) malloc((n + 1) * sizeof(char));
    if (!textoDecifradoTemp) return;

    for (int i = 0; i < n; i++) {
        char c = estadoAtual.textoParcial[i];
        if (isalpha(c)) {
            int indice = c - 'A';
            if (estadoAtual.chaveDecifracao[indice] != '\0') {
                textoDecifradoTemp[i] = estadoAtual.chaveDecifracao[indice];
            } else {
                textoDecifradoTemp[i] = c;
            }
        } else {
            textoDecifradoTemp[i] = c;
        }
    }
    textoDecifradoTemp[n] = '\0';

    unsigned long M[256];
    for (int i = 0; i < 256; i++) M[i] = 0;

    for (int j = 0; j < m; j++) {
        unsigned char c = (unsigned char)padrao[j];
        M[c] |= (1UL << j);
    }
    unsigned long R[k + 1];
    unsigned long R_antigo[k + 1];

    for(int j=0; j <= k; j++) R[j] = 0;

    int ocorrencias = 0;

    printf("\n--- Ocorrencias Aproximadas (Erro max: %d) ---\n", k);
    for (int i = 0; i < n; i++) {
        unsigned char c = (unsigned char)textoDecifradoTemp[i];
        for(int j=0; j <= k; j++) R_antigo[j] = R[j];

```

Figura 7: Algoritmo de casamento aproximado (Shift-And aproximado).

- Gerenciamento dinâmico de chave:

A interação com o usuário é centralizada na manipulação da chave de criptografia. A função de alteração de chave foi desenvolvida para garantir a integridade do mapeamento: ela assegura que cada letra original seja mapeada por apenas uma letra cifrada (relação 1:1). Caso o usuário insira um mapeamento conflitante, o sistema remove automaticamente a associação anterior, prevenindo inconsistências na decifração final (figura 8).

```

void alterarChave(char charCifrado, char charOriginal) {
    charCifrado = toupper(charCifrado);
    charOriginal = toupper(charOriginal);

    if (charCifrado >= 'A' && charCifrado <= 'Z' &&
        charOriginal >= 'A' && charOriginal <= 'Z')
    {
        // Remove mapeamento antigo do caractere ORIGINAL, se existir
        for (int i = 0; i < TAMANHO_ALFABETO; i++) {
            if (estadoAtual.chaveDecifracao[i] == charOriginal) {
                estadoAtual.chaveDecifracao[i] = '\0';
                printf("Aviso: Mapeamento antigo %c -> %c removido.\n", 'A' + i, charOriginal);
            }
        }

        // Verifica se a letra cifrada já estava mapeada (sobrescreve)
        if (estadoAtual.chaveDecifracao[charCifrado - 'A'] != '\0') {
            printf("Aviso: O caractere cifrado '%c' já estava mapeado. Mapeamento sobreescrito.\n", charCifrado);
        }

        // Aplica o novo mapeamento (Cifrado -> Original)
        estadoAtual.chaveDecifracao[charCifrado - 'A'] = charOriginal;
        printf("Registrado: %c (cifrado) -> %c (original)\n", charCifrado, charOriginal);
    } else {
        printf("Erro: Caracteres inválidos. Use apenas letras de A-Z.\n");
    }
}

```

Figura 8: Alteração de mapeamento de chave.

O resultado desse processo interativo é visualizado em tempo real, com o texto parcialmente decifrado sendo exibido com destaque visual para facilitar a distinção entre caracteres cifrados e decifrados (figura 9).

```

== Chave ==
ABCDEFGHIJKLMNOPQRSTUVWXYZ
F

== Texto Parcialmente Decifrado ==
K XHOMQF ZW FLEBNEM, K0ZF K YC0DVEIJPK BHKFOYE F KO BEHKOBKO JWDYFI. JK FJRMOKP, NMQE FOFM K0JNTWCKMO, K XHNOBFIK, K DFNZENK ZF YDNUOKO MQF ZFOMBW W YOWW ZW NMWIK: RKYF NFHHTDPF XWNYWHEW W EJBWIEW F ZFOMBWNEW
W LNKBYEW, YMWDZJK KO FOLEDJKO ZW QREZW JM WRKRF OMWNZW ZW QWKFZK0NMZ

"PO PMDOYF JZFMW KO LEYKO ZW LQNW F NFPIKOMMNO W YKNNQLYWK F W MZRNOEZMF."

```

Figura 9: Distinção entre caracteres decifrados.

Compilação e execução

Para facilitar a compilação e execução do programa, foi utilizado um arquivo Makefile. Para compilar, utiliza-se o seguinte comando: make

Após isso, o programa é executado seguindo o seguinte formato: .\execucao.exe

Em seguida, é necessário informar ao programa o nome do arquivo com o texto claro e o nome do arquivo que será gerado com a criptografia. É válido destacar que o arquivo do texto claro está dentro da pasta “texto_claro” e o texto criptografado está dentro da pasta “texto_cripto”. Logo, ao informar o nome do arquivo, deve-se escrever na forma: “texto_claro/Anaxa.txt”.

```

--- Trabalho Prático 3 - Criptoanalise ---
Informe o nome do arquivo de texto claro a ser lido (ex: texto_claro/Anaxa.txt): texto_claro/Anaxa.txt
Informe o nome do arquivo para escrever o texto criptografado (ex: texto_cripto/Anaxa.txt): texto_cripto/Anaxa.txt

```

Figura 10: Inserção do nome dos arquivos.

Resultados

Os testes realizados demonstraram a eficácia das ferramentas de criptoanálise implementadas. A seguir, apresentamos os resultados obtidos em cada etapa crítica do processo de decifração.

- Cifragem e estado inicial:

Logo após a leitura do arquivo, o sistema aplicou corretamente a Cifra de Deslocamento com um shift aleatório, gerando um texto criptografado. A interface inicial (opção 1) apresentou o texto cifrado e a chave vazia, confirmando que o sistema estava pronto para a análise sem revelar a resposta prematuramente (figura 11).

```
Escolha uma opção: 1
*** Estado Atual da Criptoanalise ***
*** Texto Criptografado ***
S FSWIYI HE ITMOERME, SRHI S GSRLIQMIRKS JPSVIMGI I SW JPMPSNISW REMGIO, RS IRXERKS, EUYM IXKE EREBEKSVEM, S FPENDIQS, S LIVHIMVS HI GLVOWSW UVI HIMEJME E GLEQE HE VEDES: ZSGI VIEPQIRXI EFVEGEVME E MRJEQME I HIMEJMEVME
E TVSJIGHE, GVEZERHS SW INTMRLSW HE HYZHME RE EVZSVI WEKVEHE HE WEFHSVME?
"XY XVERMGIRHIVEM SW TMGSW HE TYVIDE I VIXSVREVEN E GSAYTGES I E EHIZVMMHEH.".

*** Chave ***
ABCDEFGHIJKLMNPQRSTUVWXYZ

*** Texto Parcialmente Decifrado ***
S FSWIYI HE ITMOERME, SRHI S GSRLIQMIRKS JPSVIMGT I SW JPMPSNISW REMGIO, RS IRXERKS, EUYM IXKE EREBEKSVEM, S FPENDIQS, S LIVHIMVS HI GLVOWSW UVI HIMEJME E GLEQE HE VEDES: ZSGI VIEPQIRXI EFVEGEVME E MRJEQME I HIMEJMEVME
E TVSJIGHE, GVEZERHS SW INTMRLSW HE HYZHME RE EVZSVI WEKVEHE HE WEFHSVME?
"XY XVERMGIRHIVEM SW TMGSW HE TYVIDE I VIXSVREVEN E GSAYTGES I E EHIZVMMHEH.".
```

Figura 11: Opção 1 em estado inicial.

- Análise de frequência:

Ao selecionar a opção 2, o algoritmo calcula a frequência dos caracteres no texto cifrado e compara com a tabela global (obtida dos 12 arquivos de profecia) (figura 12).

Letra	Qtde	Freq(%)
E	56	18.48%
I	35	11.55%
S	32	10.56%
V	25	8.25%
W	24	7.92%
H	19	6.27%
M	18	5.94%
R	17	5.61%
G	14	4.62%
J	9	2.97%
X	8	2.64%
Y	7	2.31%
Q	6	1.98%
T	6	1.98%
L	5	1.65%
Z	5	1.65%
F	4	1.32%
P	4	1.32%
U	3	0.99%
D	2	0.66%
K	2	0.66%
B	1	0.33%
C	1	0.33%

== Frequencia GLOBAL (12 arquivos juntos) ==		
Letra	Qtde	Freq(%)
B	3	0.08%
C	17	0.43%
D	13	0.33%
E	599	15.28%
F	27	0.69%
G	184	4.69%
H	256	6.53%
I	494	12.60%
J	38	0.97%
K	42	1.07%
L	80	2.04%
M	212	5.41%
N	11	0.28%
O	2	0.05%
P	93	2.37%
Q	174	4.44%
R	195	4.97%
S	397	10.13%
T	77	1.96%
U	26	0.66%
V	326	8.32%
W	306	7.81%
X	152	3.88%
Y	140	3.57%
Z	56	1.43%

Figura 12: Tabelas de frequênciа.

A tabela comparativa exibida no terminal facilitou a visualização das discrepâncias estatísticas (figura 13).

== Comparacao LOCAL x GLOBAL ==			
Letra	Local(%)	Global(%)	Diferenca
A	0.00	0.00	0.00
B	0.33	0.08	0.25
C	0.33	0.43	-0.10
D	0.66	0.33	0.33
E	18.48	15.28	3.20
F	1.32	0.69	0.63
G	4.62	4.69	-0.07
H	6.27	6.53	-0.26
I	11.55	12.60	-1.05
J	2.97	0.97	2.00
K	0.66	1.07	-0.41
L	1.65	2.04	-0.39
M	5.94	5.41	0.53
N	0.00	0.28	-0.28
O	0.00	0.05	-0.05
P	1.32	2.37	-1.05
Q	1.98	4.44	-2.46
R	5.61	4.97	0.64
S	10.56	10.13	0.43
T	1.98	1.96	0.02
U	0.99	0.66	0.33
V	8.25	8.32	-0.07
W	7.92	7.81	0.11
X	2.64	3.88	-1.24
Y	2.31	3.57	-1.26
Z	1.65	1.43	0.22

Figura 13: Tabela comparativa.

A sugestão automática de mapeamento exibida após a tabela comparativa, mostrou-se precisa para as vogais mais comuns (como “A” e “E”), oferecendo um ponto de partida sólido para a quebra da cifra (figura 14).

```
== CHUTE DE MAPEAMENTO SUGERIDO ==
(com base em frequencia local x lingua portuguesa)
```

```
E (cifrado) -> A (provavel original)
I (cifrado) -> E (provavel original)
S (cifrado) -> O (provavel original)
V (cifrado) -> S (provavel original)
W (cifrado) -> R (provavel original)
H (cifrado) -> I (provavel original)
M (cifrado) -> N (provavel original)
R (cifrado) -> D (provavel original)
G (cifrado) -> M (provavel original)
J (cifrado) -> U (provavel original)
X (cifrado) -> T (provavel original)
Y (cifrado) -> C (provavel original)
Q (cifrado) -> L (provavel original)
T (cifrado) -> P (provavel original)
L (cifrado) -> V (provavel original)
Z (cifrado) -> G (provavel original)
F (cifrado) -> H (provavel original)
P (cifrado) -> Q (provavel original)
U (cifrado) -> B (provavel original)
D (cifrado) -> F (provavel original)
K (cifrado) -> Z (provavel original)
B (cifrado) -> J (provavel original)
C (cifrado) -> X (provavel original)
A (cifrado) -> K (provavel original)
N (cifrado) -> W (provavel original)
O (cifrado) -> Y (provavel original)
```

Figura 14: Mapeamento sugerido.

Após a exibição dos mapeamentos sugeridos, é perguntado ao usuário se deseja aplicar algum mapeamento. Caso queira, é necessário informar a letra cifrada que deseja decifrar, e em seguida a letra que será mapeada no lugar (figura 15).

```

E (cifrado) -> A (provavel original)
I (cifrado) -> E (provavel original)
S (cifrado) -> O (provavel original)
V (cifrado) -> S (provavel original)
W (cifrado) -> R (provavel original)
H (cifrado) -> I (provavel original)
M (cifrado) -> N (provavel original)
R (cifrado) -> D (provavel original)
G (cifrado) -> M (provavel original)
J (cifrado) -> U (provavel original)
X (cifrado) -> T (provavel original)
Y (cifrado) -> C (provavel original)
Q (cifrado) -> L (provavel original)
T (cifrado) -> P (provavel original)
L (cifrado) -> V (provavel original)
Z (cifrado) -> G (provavel original)
F (cifrado) -> H (provavel original)
P (cifrado) -> Q (provavel original)
U (cifrado) -> B (provavel original)
D (cifrado) -> F (provavel original)
K (cifrado) -> Z (provavel original)
B (cifrado) -> J (provavel original)
C (cifrado) -> X (provavel original)
A (cifrado) -> K (provavel original)
N (cifrado) -> W (provavel original)
O (cifrado) -> Y (provavel original)

Deseja aplicar algum mapeamento sugerido? (s/n): s
Letra cifrada: E
Mapear para: A
Registrado: E (cifrado) -> A (original)

```

Figura 15: Aplicação de mapeamento sugerido.

- Busca e refinamento:

A utilização do algoritmo Shift-And para realizar o casamento exato de caracteres (opção 3) permite localizar padrões exatos instantaneamente, validando suspeitas sobre palavras comuns. Após a seleção da opção 3, é necessário escrever o padrão a ser procurado. Caso seja encontrado, o número de ocorrências será atualizado e exibido no terminal (figura 16).

```

Escolha uma opção: 3
Qual o padrao utilizado: SRHI
Ocorrencias: 1

```

Figura 16: Casamento exato.

Ao selecionar a opção 4 para realizar o casamento aproximado de caracteres, é feita uma busca por palavras parcialmente decifradas com uma margem de erro definida pelo

usuário. Além disso, é necessário informar o padrão a ser adotado (figura 17). Nessa opção são informadas as posições das ocorrências, além do número total de ocorrências encontradas.

```
Escolha uma opção: 4
Digite um padrao: SRHI
Digite a tolerancia de erros: 1

--- Ocorrencias Aproximadas (Erro max: 1) ---
Posicao 22: SRHI
Posicao 30: SRLI
Posicao 129: S HI
Posicao 306: IRHI
Total de ocorrencias encontradas: 4
```

Figura 17: Casamento aproximado.

- Decifração final:

Através do processo iterativo de alterar a chave (opção 5) e verificar o texto parcialmente decifrado (figura 18), que é exibido com as letras descobertas destacadas em verde, é possível reconstruir a mensagem original completa.

```
Escolha uma opção: 5
digite a letra que substituir o
digite a string decifrada: E
Registrado: O (cifrado) -> E (original)

--- Opções ---
1. Apresentar o estado atual da criptografia
2. Criar um chave baseado em lista de frequência
3. Realizar casamento exato de caracteres
4. Realizar casamento aproximado de caracteres
5. Alterar chave de criptografia (Ex: A->x)
6. Exportar resultado e encerrar o programa
Escolha uma opção: 1

--- Estado Atual da Criptografia ---
Y LYCAE NK OCSPSK, YXNE Y MYKROMSKDY PVBODNO YC PSVVCYPC XHOM, XY OXHDY, KAES OODK IXKHQZBKC, Y LVKPMY, Y ROBNSBY NO MRBICYC AEO NOCKSK K MRKMK NK BIORY: FVMO BOKWODDO KLBOMKSK K SXPKSK O NOCKPSKSK
K ZBYPDK, MRKFIOMY YC OCZSKYC NK NEFSNK XK KBFYBQ CQQBONK NK CKLONVSKZ

"DE DBOOKHOBKSC YC ZSMYC NK ZEBOK O BODYBXBKC K MYBBEZHKY O K KNFOBSKNO." 

--- Chave ---
ABCDEFGHIJKLMNOPQRSTUVWXYZ

--- Texto Parcialmente Decifrado ---
Y LYCAE NA EZPNSKA, YXNE Y MYKROMSKDY PVBODNO E YC PSVVCYPC XHOM, XY OXHDY, KAES OODK IXKHQZBKC, Y LVKPMY, Y REBNESBY NE MRBICYC AEE NECPISA A PRIMA NA BAJAY: FVMO BEWMEODE ALBMNSBA A SXPMISA E NECPISA
A ZBYPDK, MBFAFWXY YC ECZSKYC NA NEFSNA XA ABFYE CQQBNA NA CKLONVSKZ
"DE DBOOKHOBKSC YC ZSMYC NA ZEBEJA E BODYBXBKC K MYBBEZHKY E A ANFEBCSNNE."
```

Figura 18: Opção 5 e exibição da mensagem.

A exportação final (opção 6) gera um arquivo contendo tanto a chave descoberta quanto o texto plenamente legível, concluindo com sucesso a missão proposta (figuras 19 e 20).

```
Escolha uma opção: 6
Informe o nome do arquivo para salvar a chave (ex: chave_final.txt): chave_final.txt
Arquivo completo exportado com sucesso para 'chave_final.txt'.
Encerrando o programa...
```

Figura 19: Mensagem da opção 6.

```

===== CHAVE DECRYPTOGRAFIA =====
Cifrado -> Original
-----
A -> (desconhecida)
B -> (desconhecida)
C -> A
D -> (desconhecida)
E -> (desconhecida)
F -> (desconhecida)
G -> E
H -> (desconhecida)
I -> (desconhecida)
J -> (desconhecida)
K -> (desconhecida)
L -> (desconhecida)
M -> (desconhecida)
N -> (desconhecida)
O -> (desconhecida)
P -> (desconhecida)
Q -> O
R -> (desconhecida)
S -> (desconhecida)
T -> S
U ->
V -> (desconhecida)
W -> (desconhecida)
X -> (desconhecida)
Y -> (desconhecida)
Z -> (desconhecida)

===== TEXTO DECIFRADO =====
O DORSME FA ERKHAKPA, OPFE O EOPZEKEOEPVO HNOSERE E OR HKNOROHOR PAREO, PO EPVAVPO, ASIK ERVA APAZAIOSAR, O DNARHEOO, O JESFEKSO FE EJSAROR SWE FERAHKA A EJAOA FA SABAO: XOEES
"VM VSAPREEPESAR OR KKEOR FA RWSEBA E SEVOSPASAR A EOSSWREADO A AFXESRKFAFE."

```

Figura 20: Arquivo com a chave e o texto.

Conclusão

O desenvolvimento do sistema proposto permitiu aplicar, de maneira integrada, diferentes técnicas de análise de algoritmos e métodos clássicos de criptoanálise. As funcionalidades implementadas demonstraram como operações relativamente simples, como análise de frequência, casamento exato e aproximado de padrões e ajustes progressivos da chave, podem ser combinadas para auxiliar na decifração de cifras de substituição.

Durante o processo, observou-se que o uso de algoritmos eficientes, especialmente nas etapas de busca aproximada com a variante do Shift-And, contribuiu para reduzir o custo computacional e tornar o sistema responsivo mesmo em textos maiores. A necessidade de manter um estado consistente da chave e do texto parcialmente decifrado também reforçou a importância de uma arquitetura modular e bem estruturada.

Embora o programa ainda esteja em fase de finalização, os resultados obtidos até o momento indicam um avanço significativo na reconstrução da chave e do texto claro original.

O trabalho cumpriu seu papel ao aprofundar a compreensão dos algoritmos estudados em aula, bem como ao demonstrar, de forma prática, como conceitos teóricos podem ser aplicados no contexto de criptoanálise. Conclui-se que, com as etapas restantes concluídas, o sistema atenderá integralmente às especificações propostas e servirá como ferramenta funcional e didática para a decifração do material fornecido.

Referências

1. GONZAGA, Julia. Estruturas de dados: Casamento de cadeias. Medium, 2024. Disponível em: <https://juliagonnzag.medium.com/estruturas-de-dados-casamento-de-cadeias-69ff6d50433a>. Acesso em: 27 nov. 2025.

2. KHAN ACADEMY. Shift cipher (Cifra de deslocamento). Disponível em: <https://pt.khanacademy.org/computing/computer-science/cryptography/ciphers/a/shift-cipher>. Acesso em: 27 nov. 2025.