UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
NÚCLEO DE COMPUTAÇÃO - NUCOMP

ALGORITMOS E ESTRUTURAS DE DADOS II 2024/2
PROF. FLÁVIO JOSÉ MENDES COELHO

PROJETO PRÁTICO 4

1 Objetivos

Este **Projeto Prático 4** – **PP4**, tem o objetivo de exercitar e avaliar suas habilidades em:

- Aplicar os algoritmos e estruturas de dados abordados em AED 2 em um problema contextualizado apresentado neste enunciado;
- Aplicar e codificar em C++20 um TAD Tabela Hash de acordo com as referências [4] e [5]), para o problema proposto nesse enunciado;
- Aplicar e codificar em C++20 o algoritmo de *string matching* (veja os slides das aulas de AED 2 e as referências [4] e [5]) **Boyer-Moore**, para o problema proposto nesse enunciado;
- Solucionar diversos subproblemas relacionados ao problema principal;
- Avaliar seu domínio sobre o código que você próprio desenvolveu neste projeto, a partir dos assuntos abordados na disciplina AED 2.

2 Descrição do problema

O artefato enigmático. Durante uma expedição arqueológica no Ártico, a equipe sueca liderada pelo Dr. Olaf Rodavlasz, descobriu centenas de grandes placas de pedra enterradas sob o gelo, contendo inscrições com símbolos estranhos. Em umas das placas havia o seguinte fragmento de símbolos:

```
:|....:.||.|::.:|:...:|.---.--|.:...:|.|..|.|---|.|..|::..--|.:...:|.---..|--.
--.|..-.-||..||.||.|||:::::|::::::||::.---|..-.-||..||::.---|..:.:.:.:.::|...
|.|---...|.|||..||...|:.::.~~~
```

Como os cientistas perceberam não se tratar de uma língua antiga conhecida, chamaram o professor Kaninchen para examinar o misterioso artefato. Passados alguns dias dedicados à investigação, o professor reuniu a equipe de colegas arqueólogos e linguistas, e apresentou suas descobertas:
— Caros colegas, após examinar o fragmento de símbolos, consegui chegar a alguns resultados tão surpreendentes quanto estranhos.
— Por favor, continue, professor — Retrucou o Dr. Rodavlasz.
— Como vocês mesmos verificaram, a datação do artefato está próxima de 40.000 anos. No entanto, os artefatos estavam envoltos em tecidos típicos do período em que Roma foi governada por Júlio César, de 40 a 60 a.C. Bem... além disso, todos sabemos que Cook ou Pery, não importa quem, botou os pés no Ártico, pela primeira vez, por volta de 1908 ou 1909. Então, a primeira pergunta, ainda sem explicação é: como os artefatos vieram parar aqui? Eu deixo essa pergunta para você investigar, Olaf.
O arqueólogo sueco confirmou com os olhos, e o professor Kaninchen continuou:
– Muito bem. Sabemos que a simbologia encontrada nas inscrições não pertence à língua latina ou a qualquer outra língua antiga conhecida. Então, primeiramente, percebi que as inscrições possuem um número de símbolos múltiplo de três. Todas elas! Esse padrão me levou a supor que os símbolos em uma inscrição, na verdade, deveriam ser lidos em grupos de três símbolos. Vejam:

$$\text{grupo}_1 \rightarrow \texttt{:|.} \qquad \text{grupo}_2 \rightarrow \texttt{...} \quad \text{grupo}_3 \rightarrow \texttt{:.|} \qquad \text{grupo}_4 \rightarrow \texttt{|.|} \text{ etc.}$$

Com isso, observei que vários desses grupos de símbolos de repetiam. Então, imaginei que cada grupo pudesse representar uma letra de um alfabeto estranho. Como os panos que protegem as pedras parecem ter sido confeccionados por algum tecelão romano do tempo de Júlio César, criei um mapeamento entre cada grupo de símbolos e uma letra do latim de 60 a.C., mas complementei com as letras faltantes, J, U e W. Assim, surgiu o seguinte dicionário ao qual estou chamando de Artefato-Latim.

#	Símbolo	Latim	#	Símbolo	Latim
1.	`:::`	A	15.	`|:.`	O
2.	`.::`	B	16.	`:|.`	P
3.	`:.:`	C	17.	`:.|`	Q
4.	`::.`	D	18.	`|..`	R
5.	`:..`	E	19.	`.|.`	S
6.	`.:.`	F	20.	`..|`	T
7.	`..:`	G	21.	`.||`	U
8.	`...`	H	22.	`|.|`	V
9.	`|::`	I	23.	`||.`	W
10.	`:|:`	J	24.	`-.-`	X
11.	`::|`	K	25.	`.--`	Y
12.	`|.:`	L	26.	`--.`	Z
13.	`.|:`	M	27.	`---`	branco
14.	`.:|`	N	28.	`~~~`	. (ponto)

Ora, ao traduzir as inscrições utilizando o dicionário, obtive como resultado uma cadeia com caracteres misturados, aparentemente, sem sentido. Vejam:

`PHQVDJHP YLHPRV VTDYLHP TZZRXWUUUIDIDUD RXWUD REEEKHV HVWUHOD.`

Ao ver essa mensagem ininteligível para nós, imaginei ter fracassado com o dicionário. No entanto, tive um lampejo... uma intuição. Conta-se que, o imperador Júlio César, trocava mensagens secretas com seus generais em campo de batalha, utilizando um método muito simples para cifrar as mensagens e mantê-las ininteligíveis para os inimigos. O método consistia em substituir cada letra da mensagem original por uma outra do mesmo alfabeto, obtida saltando-se um certo número k fixo de letras após cada letra original. Portanto, a partir de uma mensagem m original e um inteiro não negativo $k \leq 26$, o método gerava a mensagem cifrada m'. E, para decifrar a mensagem cifrada m', ou seja, converte-la de volta à mensagem original m, bastava aplicar o mesmo método saltando k letras para trás. Por exemplo, se o imperador considerasse $k = 3$ e a mensagem fosse...

LEGIAO LOBO AVANCE PARA SUL ENCONTRAR LEGIAO NEVE,

ao aplicar o método, considerando o alfabeto "ABCDEFGHIJKLMNOPQRSTUVWXYZ", ele obteria a mensagem cifrada:

OHJLDR ORER DYDQFH SDUD VXO HQFRQWUDU OHJLDR QHYH.

Então, a partir dessa intuição, conjecturei que a mensagem do artefato pudesse ter sido cifrada com esse método simples. Porém, para decifrar a mensagem PHQVDJHP YLHPRV VTDYLHP TZZRXWUUUIDIDUD RXWUD REEEKHV HVWUHOD, eu precisaria conhecer o valor do número k. Infelizmente, não temos nenhuma pista de quanto vale o número k nos panos romanos. Então, tive que empregar uma estratégia de força bruta para descobrir esse número. Com essa estratégia, tentei decifrar a mensagem utilizando todos os valores de k entre 0 e 26. E, caros colegas, após algum tempo de tentativas, obtive a surpreendente mensagem:

MENSAGEM VIEMOS SQAVIEM QWWOUTRRRFAFARA OUTRA OBBBHES ESTRELA.

— O grifo em vermelho é meu, senhores — disse o prof. Kaninchen.
Os membros da expedição ficaram perplexos! Olaf Rodavlasz levantou-se devagar e, dirigindo-se ao professor, perguntou pausadamente:
— Você não quer que acreditemos... "Viemos de outra estrela", Kaninchen?
— Também, me parece inacreditável, Olaf. — respondeu Kaninchen, que continuou:
— A única forma de entendermos o sentido dessa mensagem, é decifrando todas as outras. E todos aqui, sabemos que este é o primeiro passo de uma longa jornada de pesquisas.
E, o professor continuou:
— A tarefa de tradução das inscrições deve resultar em um texto de proporções colossais, cujas palavras estão espalhadas entre cadeias de letras sem nenhum significado aparente. Ainda que empreguemos todo poder computacional de que dispomos, essa tarefa consumiria uma longa década! Assim, proponho dividirmos as inscrições em fragmentos complementares de forma que o trabalho possa ser processado por vários grupos, de forma paralela.

A assembleia acatou a ideia do professor e iniciou uma série de discussões sobre como essa operação seria distribuída e executada por vários grupos e centros de computação ao redor do planeta.

* * *

Passados alguns dias, de volta ao seu laboratório, o prof. Kaninchen reuniu seus alunos para contar as descobertas, e convidou você e mais um colega ou uma colega para codificarem todos as os algoritmos e estruturas de dados necessárias para decifrar os códigos dos artefatos. A partir desse momento, para alcançar esse objetivo, sua dupla precisa:

(1) Implementar o dicionário Artefato-Latim utilizando uma tabela *hash*. Seja s uma *string* que representa um símbolo do alfabeto do artefato. Calcule a chave $k = s[0] \times K^{n-1} + s[1] \times K^{n-2} + ... + s[n-1] \times K^0$, onde $K = 5$ (número de símbolos do alfabeto do artefato). Para obter o valor *hash* escreva uma função que faça $k\%M$, onde $M = 11$;

(2) Utilizar o dicionário para traduzir o texto de um fragmento do artefato para o latim ajustado e obter o texto traduzido (mas, ainda cifrado);

(3) Aplicar o método de criptografia inverso ao explicado pelo prof. Kaninchen para decifrar o texto traduzido e obter o texto decifrado;

(4) Utilizar o algoritmo de Boyer-Moore de *string matching* para buscar uma lista de padrões (palavras) no texto decifrado.

3 Entradas e saídas do problema

Entrada. O texto na linguagem de símbolos do artefato seguido da lista de padrões que serão buscados no texto decifrado T, exceto a palavra "fim" que serve para indicar o fim da entrada de dados.

```
:|....:.||.|::.:|:...:|.-----.::.:..---...|.|||..||...|:.::.---..|...|.||.|
|.|---||.|..-.-.|||..---|:.:.:::.-.-|.||::----.-|:.::.|.|.--..|..|.|::::---
|.:|.:---:::.:|....::..:.::.---|.:::...:|::.:.|..---|.:::.:.|||.....:.|..---
.--::.::.---...|.|||..||...|:.::.~~~
touro
estrela
ataque
ameaca
fim
```

O texto traduzido (e cifrado) obtido a partir do texto da entrada é o seguinte:

```
PHQVDJHP ZDE HVWUHOD THVVV WRXUR OCDXVI XODVYTTMA LL DPHDFD LDGIFR LDQWHFR YDD HVWUHOD.
```

O texto decifrado obtido a partir do texto traduzido acima é:

```
MENSAGEM WAB ESTRELA QESSS TOURO LZAUSF ULASVQQJX II AMEACA IADFCO IANTECO VAA ESTRELA.
```

Saída. Para cada padrão P da lista de padrões a ser buscado pelo algoritmo Boyer-Moore no texto decifrado T: (1) imprima o padrão buscado, em maiúsculo, seguido de espaço em branco; (2) seguido dos valores dos saltos que o algoritmo Boyer-Moore executa até achar uma ocorrência de P (incluindo o salto 0), cada valor seguido de espaço em branco; (3) seguidos do deslocamento dessa ocorrência de P, entre parênteses, (**se essa ocorrência de P existir**), seguida de espaço em branco; (4) repetir (2)-(3) para as próximas possíveis ocorrências do padrão P em T.

```
TOURO: 5 5 5 5 5 2 0 (27) 5 5 5 5 5 5 5 5 5 1 5
ESTRELA: 2 7 4 0 (13) 7 4 7 7 7 7 7 7 7 5 0 (79) 7
ATAQUE: 6 6 5 1 6 1 6 1 6 6 6 5 6 6 6 6 6
AMEACA: 6 6 3 6 6 6 6 6 6 2 0 (53) 6 6 3 6 6 6
```

Nesse exemplo de saída, `TOURO` possui deslocamento 18 (isto é, foi encontrado na posição 18 do texto decifrado T), `ESTRELA` possui os deslocamentos 4 e 70, `ATAQUE` não foi encontrado no texto T e, portanto, possui apenas saltos não nulos, e `AMEACA` possui deslocamento 44.

4 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por duplas ou individualmente. **Não serão aceitas equipes com mais de dois participantes**.

2. **Ferramentas e técnicas**. O projeto deve ser codificado em C++20. Os contêineres da STL permitidos são `std::vector`, `std::list` e `std::pair`. O uso de qualquer outra biblioteca não é permitido, com exceção de `iostream`, `list`, `vector`, `cstdlib`, `limits` e `iomanip`. Pelo menos os TADs das estruturas de dados devem ser codificadas com programação orientada a objetos. Como compilador local, sugerimos o GCC (Linux e Windows) ou o Clang (Mac OS). Para codificação local, sugerimos o editor/IDE `https://www.sublimetext.com/` ou `https://code.visualstudio.com/` combinado com o compilador local C++20 sugerido acima. Para codificação online, sugerimos o IDE `https://cpp.sh/` com suporte ao C++20.

3. **Correção e pontuação**. A correção e pontuação do PP4 é dividida em duas etapas:

   Correção funcional - CF. Essa correção é realizada de forma automática pelo juiz on-line ao qual a equipe submeterá o solução/código do PP4. Essa correção tem pontuação máxima igual a $10,0$ pts. A equipe ou aluno pode submeter o código ao juiz on-line quantas vezes quiser, até a data e hora do prazo final. Ao submeter uma solução, o juiz on-line executará N casos de teste para testar a corretude da solução submetida. Portanto, pontuação da solução submetida corresponde ao número de casos de teste corretos, ou seja, $10,0/N * C$, onde N é o número de casos de teste do juiz on-line e C é o número de casos de teste corretos. Os detalhes sobre a submissão serão repassados pelo professor, por e-mail ou pelo Classroom.

   Correção de Inspeção de código - CIC. Essa correção é realizada manualmente pelo professor que examinará a última versão do código submetido ao juiz on-line para verificar se esse código atende aos requisitos desse enunciado. Em caso positivo, o professor confirma os pontos obtidos na CF. Em caso contrário, se um ou mais requisitos não forem seguidos, a pontuação obtida na CF sofrerá uma série de penalizações reduzindo o valor obtido na CF. Isso é necessário porque uma solução pode obter a nota máxima nas submissões ao juiz on-line, mas não atender a um ou mais requisitos solicitados em um enunciado. Por exemplo, um requisito pode pedir a implementação de uma algoritmo ou estrutura de dados X e, como solução, a equipe implementa um algoritmo ou estrutura de dados Y, diferente do que foi solicitado. Dessa forma, apresentar quaisquer outras soluções (algoritmos, estruturas de dados, linguagem de programação, etc) que não sejam as especificadas

na tabela abaixo, resulta nas penalidades (acumulativas) sobre a CF para cada requisito não atendido:

Tabela 1: Requisitos do PP4 que devem ser atendidos

Tipo	**Requisito**	**Penalidade**
AED	Tabela *hash*, conforme referência	-50% do CF
AED	Algoritmo de Boyer-Moore de *string matching*, conforme referência e utilizando somente std::string e strings padrão ANSI C	-50% do CF
Linguagem	C++20	-50% do CF
Linguagem	Uso incorreto de funções e parâmetros (ponteiros, constantes, referências, etc), templates e classes C++; uso de variáveis globais	-30% do CF
Técnicas	POO (classes encapsuladas) e programação estruturada	-30% do CF
Estilo de código	Indentação K&R ou Allman correta (veja https://en.wikipedia.org/wiki/Indentation_style) e nomeação de variáveis com Camel case ou Snake case ou Pascal case (veja https://en.wikipedia.org/wiki/Naming_convention_(programming))	-10% do CF

Datas

- Emissão deste enunciado: 12/11/2024 às 03:30 (hora local).
- Abertura do juiz online: 15/11/2024 às 15:00 (hora local).
- Fechamento do PP4 no juiz on-line: 29/11/2024 às 22:00 (hora do juiz on-line).

CÓDIGO DE ÉTICA

Este projeto é uma avaliação acadêmica e deve ser concebido, projetado, codificado e testado pela equipe, com base nas referências fornecidas neste enunciado ou nas aulas de Algoritmos e Estruturas de Dados, ou por outras referências indicadas pelo professor, ou com base em orientações do professor para com a equipe, por solicitação desta. Portanto, não copie código pronto da Internet para aplicá-lo diretamente a este projeto, não copie código de outras equipes, não forneça seu código para outras equipes, nem permita que terceiros produzam este projeto em seu lugar. Isto fere o código de ética desta disciplina e implica na atribuição da nota mínima ao trabalho.

Referências

[1] COELHO, Flávio. Slides das aulas de *Algoritmos e Esruturas de Dados II*. Disponível em https://est.uea.edu.br/fcoelho. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP. Semestre letivo 2016/2.

[2] C++. In: *WIKIPÉDIA, a enciclopédia livre*. Flórida: Wikimedia Foundation, 2016. Disponível em: <https://pt.wikipedia.org/w/index.php?title=C%2B%2B&oldid=45048480>. Acesso em: 17 abr. 2016.

[3] C++. In: *cppreference.com*, 2016. Disponível em <http://en.cppreference.com/w/>. Acesso em: 17 abr. 2016.

[4] CORMEN, T. H., Leiserson, C. E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 3rd edition, MIT Press, 2010

[5] R. Sedgewick, K. Wayne. Algorithms. 4th edition, Addison-Wesley Professional, 2011

[6] STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Edition, Addison-Wesley, 2013.

[7] STROUSTRUP, Bjarne. *A Tour of C++*. Addison-Wesley, 2014.

[8] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro. 2a. Ed. LTC, 1994.

[9] WIRTH, Niklaus. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. 1a. Ed. Prentice - Hall do Brasil Ltda., 1989.

[10] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Java e C++*. 2a. Edição. Cengage Learning, 2010.

[11] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Pascal e C*. 3a. Ed. São Paulo: Cengage Learning, 2012.