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

ALGORITMOS E ESTRUTURAS DE DADOS I – 2025/1
PROF. FLÁVIO JOSÉ MENDES COELHO

PROJETO PRÁTICO 3 (versão 1)

1 Objetivos

Este **Projeto Prático 3** – **PP3**, tem o objetivo de exercitar e avaliar suas habilidades em codificar os tipos abstratos de dados TABELA HASH, ÁRVORE AVL (e outras auxiliares que venham a ser necessárias tais como LISTAS, FILAS, PILHAS, etc) na linguagem de programação exigida neste enunciado, e solucionar problemas empregando estas estruturas de dados;

2 Descrição do problema

Codifique uma **tabela hash** combinada com **árvores AVL** para resolver colisões em lugar de listas encadeadas, como ilustra a Figura 1. O vetor T armazena as entradas (*slots*/*buckets*) da tabela hash, sendo que cada entrada $T_0, T_1, \ldots, T_{m-1}$ armazena um objeto árvore AVL $A_0, A_1, \ldots, A_{m-1}$, correspondente. Considere $m = 151$.

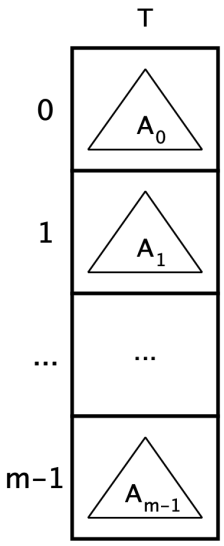

Figura 1: Tabela hash com árvores AVL para resolver colisões.

3 Entradas e saídas do problema

Seu programa processará diversos casos de teste de um juiz online. Cada caso terá entradas e saídas como mostrado a seguir.

Entrada. Cada entrada contém um texto e uma chave de busca localizada na última linha do texto, indicada pela marcação "`###`". Seu programa deve carregar o texto na tabela hash e realizar uma buscar pela chave de busca fornecida. Para isso, seu programa deverá ler cada string do texto da entrada, do início ao fim, sendo cada string uma chave a ser inserida na tabela hash, que não admite repetição de chaves. Exemplo de entrada:

```
"Mr.  Bilbo Baggins of Bag End announced that he would shortly be celebrating
his eleventy-first birthday with a party of special magnificence, there was
much talk and excitement in Hobbiton...
### Baggins"
```

Neste exemplo, cada string (sem pontuações como vírgulas, pontos, etc) do texto que está antes da marcação "`###`" é uma chave a ser inserida na tabela hash. A string `Baggins`, seguinte à "`###`", é a chave de busca.

Saída. Para cada busca processada a partir da **Entrada**, seu programa deverá apresentar como saída o valor da altura da árvore AVL que contenha a chave de busca. Por outro lado, se a chave **não** for encontrada, seu programa deve apresentar como saída a mensagem "Nao encontrado" (sem acentos).

4 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por uma equipe de dois estudantes. Não serão aceitas equipes com número maior de participantes, exceto se permitido pelo professor por alguma excepcionalidade.

2. **Ferramentas e técnicas**. O projeto deve ser codificado em C++20. Será permitido programação procedural, mas todos as estruturas de dados (TADs) devem ser programadas orientadas a objeto, generalizadas (com templates) e encapsuladas.

3. **Padrões de codificação**. Escolha um padrão de nomeação e formatação para todo o seu código. Veja `https://webkit.org/code-style-guidelines/` e `en.wikipedia.org/wiki/Indent_style#K.26R_style`.

4. **Compilador**. Indica-se o uso do compilador *GCC - the GNU Compiler Collection* encontrado no GNU/Linux ou sua variante *Mingw* para para Microsoft Windows (versão c++20). Para MacOS há o CLang ou mesmo o GCC. Você também pode usar um compilador C++20 on-line como o `https://cpp.sh/`, por exemplo.

5. **Submissão**. Todo o projeto deverá ser submetido ao juiz on-line em um único arquivo fonte com extensão `cpp`. A dupla dele eleger qual dos alunos realizará as submissões de

código para o juiz online. No juiz online deve aparecer somente a identificação de login desse mesmo aluno que realizará as submissões.

6. **Bibliotecas e funções**. Sua equipe não deve utilizar nenhuma estrutura de dados pronta (listas, pilhas, filas, vetores, grafos, etc) da *Standard Template Library* - STL, ou de qualquer outra biblioteca C++. É suficiente o uso de `iostream`, `cmath`, `cctype`, `cstdlib` e `string` e `cstring`. Para uso de qualquer outro arquivo de cabeçalho, fale com o professor. O uso de estruturas de dados prontas ou funções de bibliotecas conforme explicado acima, implicará na atribuição da nota mínima ao projeto.

5 Pontuação

O **PP3** vale no mínimo 0,0 e no máximo 10,0, de acordo com os seguintes critérios:

(1) **Correção funcional - CF**. Essa correção é realizada de forma automática pelo juiz on-line ao qual a equipe submeterá o solução/código do PP2. Essa correção tem pontuação máxima igual a $10,0$ pts. Somente um dos membros da equipe deve 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 verificar a correção da solução submetida. Portanto, a pontuação da solução submetida corresponde ao número de casos de teste corretos, ou seja, $10/N * C$, 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 Google Classroom.

(2) **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 o código atende os requisitos deste enunciado. Em caso positivo, o professor confirma os pontos obtidos na CF. Em caso contrário, se um ou mais requisitos não forem atendidos, a pontuação obtida na CF sofrerá penalizações acumulativas 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, codifique seu código atendendo a todos os requisitos da tabela abaixo. Portanto, a pontuação final fica assim: **Pontuação_Final** $= CF - (\sum_{i=1}^{7} P_i)$ (com a restrição **Pontuação_Final** ≥ 0).

6 Datas

- Emissão deste enunciado: 27/05/2025.
- Abertura do juiz online: 27/05/2025.
- Fechamento do juiz online: 09/06/2025 às 22h (hora local).

Tabela 1: Requisitos não atendido e penalidades

Tipo	**Requisito**	**Penalidade por não atender o requisito**
AED	Implementação da árvore AVL conforme as referências [1, 9, 10] (incluindo os slides de AED 1 – prof. Flávio)	P_1 = -40% do CF
AED	Implementação da tabela hash com resolução de colisões por encadeamento (com árvores AVL em lugar de listas) conforme descrito neste enunciado e nas referências [3, 11] (incluindo os slides de AED 1 – prof. Flávio)	P_2 = -40% do CF
Linguagem	Uso da linguagem C++ (C++20)	P_3 = -50% do CF
Linguagem	Uso das bibliotecas `iostream`, `cmath` (uso optativo de `cstdlib`, `cctype` e `cstring`)	P_4 = -50% do CF
Linguagem	Uso correto de funções e parâmetros por valor e/ou referência; uso de ponteiros, constantes, referências, templates, classes C++; não uso de variáveis globais	P_5 = -10% do CF
Técnicas	Uso de POO (classes e templates) para os TADs	P_6 = -10% do CF
	Estilo de código e endentação como descrito acima	P_7 = -10% do CF

7 Contribuições

Agradeço ao Francisco Elio Parente Arcos Filho, aluno do curso de Engenharia de Computação e participante do GAPA, que ajudou na geração dos casos de teste do Juiz Online Run.codes para AED1/2017. Agradeço também aos meus monitores de AED1/2025 Oziel Bezerra de Lima do curso de Bac. em Sistemas de Informação e Guilherme Gonçalves Moraes do curso de Engenharia de Computação que trabalharam na geração dos casos de teste do Juiz Online Hackerank.

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 Estruturas de Dados I*. Disponível no Classroom. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP.

[2] C++. In: *cppreference.com*, 2025. Disponível em <https://cppreference.com/>.

[3] CORMEN, T. H., Leiserson, C. E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 4th edition, MIT Press, 2021.

[4] KNUTH, Donal E. *Fundamental Algorithms*, 3rd.ed., (vol. 1 de The Art of Computer Programming), Addison-Wesley, 1997.

[5] KNUTH, Donal E. *Seminumerical Algorithms*, 3rd.ed., (vol. 2 de The Art of Computer Programming), Addison-Wesley, 1997.

[6] KNUTH, Donal E. *Sorting and Searching*, 2nd.ed., (vol. 3 de The Art of Computer Programming), Addison-Wesley, 1998.

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

[8] STROUSTRUP, Bjarne. *A Tour of C++*. 3rd Edition. Addison-Wesley, 2022.

[9] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. 3a. Ed. Grupo Gen LTC: Rio de Janeiro, 2010.

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

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

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