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

ALGORITMOS E ESTRUTURAS DE DADOS I – 2017/1
PROF. FLÁVIO JOSÉ MENDES COELHO

PROJETO PRÁTICO 2

1 Objetivos

Este **Projeto Prático 2 – PP2**, 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 no, tais como LISTAS, FILAS, PILHAS, etc) na linguagem de programação exigida neste enunciado, e solucionar problemas empregando estas estruturas de dados;
- Aprender a lidar com desenvolvimento de software em equipe;

2 Descrição do problema

Tabelas hash consomem tempo $O(1)$ no caso médio, isto é, em cenários típicos de busca, quando a chave buscada nem está em uma posição que facilite a busca, nem em uma posição muito onerosa para ser alcançada. Desta forma, na prática, tabelas hash são estruturas de dados extremente eficientes. Entretanto, em seu pior caso, quando a chave buscada está localizada na n-ésima posição da lista de colisões, podem consumir um tempo equivalente a $O(n)$. Uma ideia para reduzir este tempo, seria trocar a lista encadeada por uma árvore balanceada de busca que consome tempo $O(\log n)$, no pior caso de busca na árvore.

Suponha que você precisa codificar um módulo de um sistema crítico, sistema no qual seja proibitivo um tempo de busca ultrapassar $O(\log n)$, sob o risco de ameaçar a vida de um ou mais pessoas. Para isto, você utilizará uma tabela hash combinada com uma árvore AVL para resolver colisões, como ilustra a Figura 1. O vetor $T = [T_0, T_1, \ldots, T_{m-1}]$ armazena as entradas da tabela hash, sendo cada entrada $T_0, T_1, \ldots, T_{m-1}$ um ponteiro para uma árvore AVL $A_0, A_1, \ldots, A_{m-1}$, correspondente.

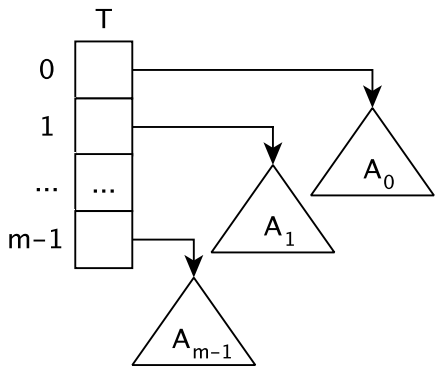

Figura 1: Tabela hash com árvores AVL para resolver colisões.

3 Entradas e saídas do problema

Seu programa deverá ler um arquivo denominado `chaves.txt` contendo 20.000 chaves na forma de inteiros positivos, como mostrado abaixo:

Arquivo `chaves.txt`:

```
2
1525
3082
56
701
11
90830
115
1016
722
4001
...
17
```

Seu programa deve ler cada uma das chaves do arquivo, e inseri-la na tabela hash, sendo que não se admite repetição de chaves na tabela.

Entrada. Seu programa processará diversos casos de teste de um juiz online. Cada caso terá como entrada uma chave inteira não negativa a ser buscada na tabela hash.

Saída. Para cada busca processada a partir da **Entrada**, seu programa deverá apresentar como saída a sequência de chaves colididas com a chave da entrada, apresentadas em ordem crescente, separadas por um espaço em branco. Por exemplo, suponha que a chave `3082` possua as chaves `230`, `14`, `4467`, `3082` e `670` como chaves colididas. Então, se `3082` fosse dada como entrada, o resultado da busca deveria ser assim:

```
14 230 670 3082 4467
```

Por outro lado, se a chave **não** for encontrada, seu programa deve apresentar como saída a mensagem "Chave não encontrada.".

4 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por uma equipe de dois ou três estudantes. Não serão aceitas equipes com número menor ou maior de participantes.

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

3. **Padrões de codificação**. Siga o *CamelCase* do Java como padrão de nomeação de identificadores. A indentação e posicionamento de chaves deve seguir o padrão K&R. variante de Stroustrup (`en.wikipedia.org/wiki/Indent_style#K.26R_style`).

4. **Compilador**. Indica-se o uso do compilador *GCC - the GNU Compiler Collection* (http://gcc.gnu.org), ou sua variante *Mingw* para para Microsoft Windows.

5. **Submissão**. Todo o projeto deverá ser submetido ao juiz online (a ser informado pelo professor) em um único arquivo fonte com extensão `cpp`.

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. É suficiente o uso de `iostream` e `cstdlib` (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 **PP2** vale no mínimo 0.0 e no máximo 10.0. As notas são distribuídas em duas partes:

(1) A **avaliação funcional** avalia o quanto foi implementado do que foi pedido, e se as saídas corretas são obtidas. Esta parte vale de 0.0 à 7.0 pontos.

O código do projeto será submetido a um juiz online com 30 casos de teste. A pontuação será atribuída da seguinte forma:

(a) Se o projeto utiliza apenas uma tabela hash com listas encadeadas: cada caso de teste correto valerá $5.0/30 = 0.1666$, ou 0.0, se incorreto.

(b) Se o projeto utiliza uma tabela hash integrada a uma árvore AVL: cada caso de teste correto valerá $(5.0 + 2.0)/30 = 0.2333$, ou 0.0, se incorreto.

(2) A **inspeção de código** avalia critérios de qualidade do código especificados à seguir. Esta parte vale de 0.0 à 3.0 pontos. Os critérios são:

- Indentação correta, padrão de nomeação e comentários: 0.0 à 0.5.
- Nomes adequados para identificadores: 0.0 à 0.5.
- Implementação correta/coerente dos tipos abstratos de dados (seguindo o que foi apresentado nas aulas): 0.0 à 0.5.
- Implementação orientada a objetos (com encapsulamento) das estruturas de dados: 0.0 à 0.5.
- Código genérico (templates): 0.0 à 0.5.
- Codificação racional e eficiente do código: 0.0 à 0.5.

Importante: se o projeto não acertar nenhum caso de teste, ou se for implementado em outra linguagem de programação, ganhará integralmente a nota mínima.

6 Datas

- Emissão deste enunciado: 12/06/2017.
- Abertura do juiz online: 16/06/2017.
- Fechamento do juiz online: 26/06/2017 às 08h00min (hora local).

O projeto não terá adiamentos.

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 I*. Disponível em https://est.uea.edu.br/fcoelho. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP. Semestre letivo 2016/1.

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

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

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

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

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

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

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

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

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