

1. Objetivo

Desenvolver um programa em linguagem C, compilável pelo compilador gcc (<https://gcc.gnu.org/>) para realizar operações em estrutura de dados de **árvores AVL**.

Os **elementos** da árvore são números inteiros positivos. Não existe repetição de elementos na árvore.

O programa desenvolvido deve conter funções para:

- Criar uma árvore AVL a partir de uma lista de elementos;
- Incluir um elemento em uma árvore AVL;
- Excluir um elemento de uma árvore AVL;
- Calcular o fator de balanceamento de um nó;
- Verificar se uma árvore binária **de busca** é uma árvore AVL.

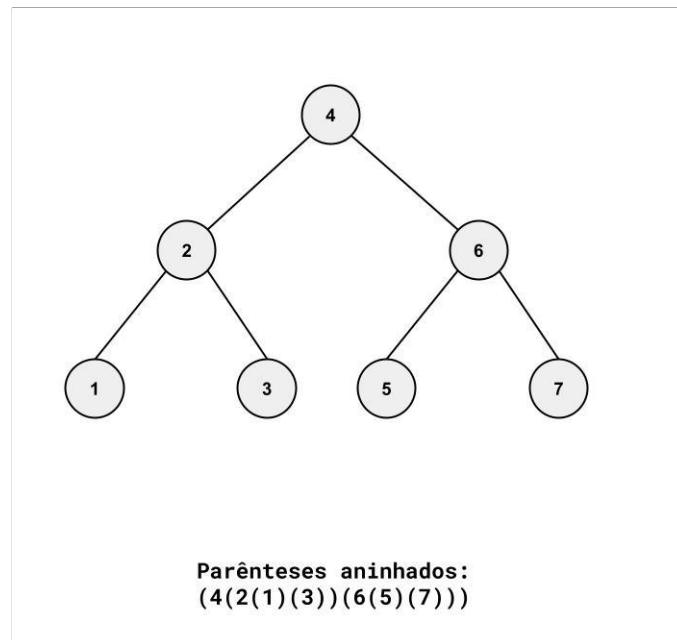


Fig 1 - Árvore AVL e sua representação por parênteses aninhados

2. Forma de entrega dos trabalhos e requisitos

- O programa deve ser compilável em gcc padrão. Não devem ser usadas nenhuma opção de compilação e “*linkedição*” específicas de qualquer sistema operacional. Não deve ser dependente de um Makefile.
- O programa **deve** consistir de um único arquivo chamado `arvore.c` (opcionalmente um `arvore.h` também) que **deve** poder ser compilado apenas com o comando abaixo:

```
$ gcc arvore.c -o arvore.exe
```

- O programa executável **deve obrigatoriamente** aceitar dois parâmetros, arquivo de entrada com o nome `entrada.txt` e arquivo de saída com o

nome `saida.txt`. O arquivo de entrada (`entrada.txt`) **deve** ser lido no mesmo diretório que esteja o executável, ou seja, no diretório de trabalho. O arquivo de saída (`saida.txt`) **deve** ser gerado nesse mesmo diretório. Não crie subdiretórios extras. Todos os arquivos `.c` `.h` e `.txt` devem estar no diretório de trabalho do executável, pois serão executados como abaixo:

```
$ arvore.exe entrada.txt saida.txt
```

3. Definição dos arquivos de entrada e saída

a. Definição do arquivo de entrada

As linhas do arquivo de entrada contêm instruções para a execução do programa e das funções. A execução de cada função é determinada por um código numérico definido na **Tabela 1**. As linhas contêm as seguintes informações:

- i. A primeira linha do arquivo de entrada contém a quantidade de operações contidas no arquivo. **Não se esqueça dessa linha no arquivo de saída!**
- ii. Em seguida uma linha com um código numérico da operação (**números de 1 a 5**) e, conforme a operação, 1 ou 2 linhas seguintes contêm os argumentos (**Tabela 1**);
- iii. Um espaço será utilizado como separador de campos dentro da linha, exceto na representação por parênteses aninhados que não tem espaços separadores;
- iv. **Na representação de parênteses aninhados, um nó vazio é representado por abre e fecha parênteses “()” quando o nó pai possuir pelo menos um filho.**
- v. As operações não precisam estar em ordem de código de operação no arquivo de entrada (ou seja, podem aparecer em qualquer ordem);

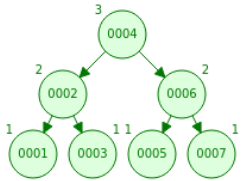
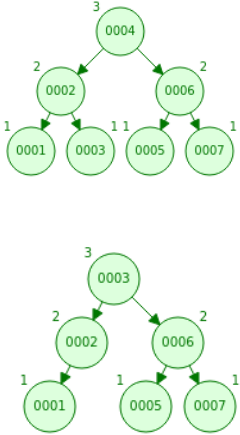
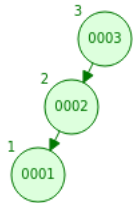
b. Definição do arquivo de saída

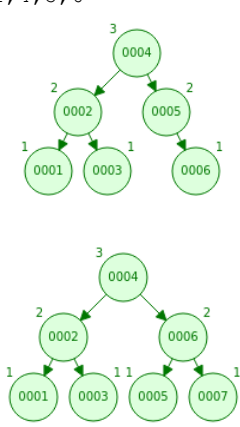
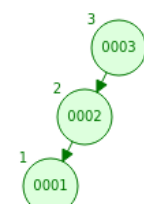
As linhas do arquivo de saída contêm as respostas (ou retornos) das das funções contidas no arquivo de entrada. Os retornos aparecem na mesma ordem de execução de cada função do arquivo de entrada. As linhas contêm as seguintes informações:

- i. A primeira linha do arquivo de saída contém a quantidade de retornos das operações contidas no arquivo de entrada seguindo a mesma ordem. **Não se esqueça dessa linha no arquivo de saída!**
- ii. Em seguida uma linha com um código numérico da operação retornada (**números de 1 a 5**) (**Tabelas 1**), conforme ordem do arquivo de entrada;
- iii. Um espaço será utilizado como separador de campos dentro da linha, exceto na representação por parênteses aninhados que não tem espaços separadores;
- iv. **Na representação de parênteses aninhados, um nó vazio é representado por abre e fecha parênteses “()” quando o nó pai possuir pelo menos um filho.**

- v. O arquivo de saída conterá as operações requisitadas no arquivo de entrada, seguindo a mesma ordem;

Tabela 1 - Definição das operações e exemplos nos arquivos de entrada e saída.

Função	Cód. Operação	Nr. de arg. Entrada	Codificação no arquivo de entrada do exemplo descrito na coluna "Função"	Codificação no arquivo de saída do exemplo descrito na coluna "Função"
<p>Criar uma árvore AVL a partir de uma lista de elementos (o primeiro elemento está à esquerda e o último à direita) Ex.: 3, 2, 1, 4, 5, 6 e 7</p> 	1	1	<p>1 3 2 1 4 5 6 7</p>	<p>1 (4 (2 (1) (3)) (6 (5) (7)))</p>
<p>Excluir um elemento de uma árvore AVL Ex.: Excluir 4 da árvore 3, 2, 1, 4, 5, 6, 7</p> 	3	2	<p>3 4 (4 (2 (1) (3)) (6 (5) (7)))</p>	<p>3 (3 (2 (1) ()) (6 (5) (7)))</p>
<p>Verificar se uma árvore binária de busca é uma árvore AVL. caractere T maiúsculo se True e caractere F maiúsculo se False Ex.:</p> 	5	1	<p>5 (3 (2 (1) ()) ())</p>	<p>5 F</p>
<p>Incluir um elemento em uma árvore AVL Ex.: Incluir 7 na árvore</p>	2	2	<p>2 7 (4 (2 (1) (3)) (5 () (6)))</p>	<p>2 (4 (2 (1) (3)) (6 (5) (7)))</p>

<p>3, 2, 1, 4, 5, 6</p> 				
<p>Calcular o fator de balanceamento de um nó de uma árvore. Ex.:</p> 	4	2	<p>4 3 (3 (2 (1) ()) ())</p>	<p>4 -2</p>

4. Orientações

- O trabalho é INDIVIDUAL, com o nome do aluno e número USP devidamente identificado na primeira linha do código-fonte. Ex:

```

/*****
/* Aluno: João da Silva Sauro
/* Número USP: 09999999
/* Disciplina/Ano/EP: ACH2023/2021/EP2
*****/

```

- A entrega deverá ser realizada em atividade específica para o EP2 do ambiente e-disciplinas, até **09/01/2022 (limite: 08:00h de 10/01/2022)**.
- O projeto deverá ser entregue em arquivo compactado (formato ZIP), nomeado da seguinte forma:
 - d_09999999_NomeCompletoDoAluno (prefixo “d” + “_” underline + número USP com oito dígitos preenchido com zeros à esquerda se necessário + “_” underline + **Nome completo da aluna(o) sem abreviações ou acentuação**)
 - O arquivo ZIP deve conter apenas o programa-fonte em C de nome **arvore.c**, não enviar o executável **arvore.exe** ou exemplos dos arquivos de entrada e saída.
 - Se necessário, um arquivo **readme.txt** contendo a linha de comando para execução do programa **arvore.exe** e a linha de comando de compilação do programa **arvore.c** com a explicação de uma razão muito forte para o uso de alguma opção extra utilizada para compilação. Se tal arquivo não for entregue assume-se que a compilação e execução segue a especificação deste EP.
- A documentação interna do programa é um item de avaliação;

- Utilize nome esclarecedores para suas variáveis;
- Trabalhos com evidências de plágio serão desconsiderados e os autores receberão nota **ZERO**.

5. Referências

- [1] SZWARCFITER, J. & MARKEZON, L. Estruturas de Dados e seus Algoritmos. LTC Editora, 3a. Ed., 2010.
- [2] GALLES, David. Visualizing Algorithms. AVLTree, <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>, acessado em 29/11/2021.
- [3] WANGENHEIM, Aldo von. Simulação de árvores AVL, <https://www.inf.ufsc.br/~aldo.vw/estruturas/simulador/AVL.html>, acessado em 29/11/2021.