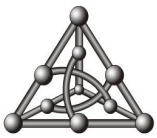


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

# Estruturas de Dados

## Trabalho 2 — Operações avançadas com árvores AVL

## 1 Descrição

Imagine que você está trabalhando em uma aplicação de banco de dados para um sistema de gestão de inventários. Cada unidade de produto possui um código numérico único e o sistema precisa realizar operações rápidas de consulta, inserção e remoção de dados. O sistema também deve permitir a união de dois bancos de dados de produtos e a busca de produtos que estejam dentro de um determinado intervalo de códigos. Neste cenário, o uso de árvores AVL é essencial para manter as operações eficientes, mesmo com grandes volumes de dados, garantindo que o tempo de execução seja logarítmico na quantidade de itens armazenados, para as operações de inserção, remoção e busca. Além disso, operações de união e interseção permitem a consolidação de dados de diferentes fontes de maneira eficiente. As operações de união, interseção e busca em intervalos são essenciais em aplicações de sistemas de banco de dados, onde o balanceamento dinâmico pode impactar diretamente a eficiência do sistema. Ao implementar essas operações, você estará lidando com problemas reais enfrentados por engenheiros de software ao gerenciar grandes volumes de dados. Além das operações básicas de busca, inserção e remoção, você deve implementar as seguintes operações avançadas em árvores AVL:

1. **União de Árvores AVL**: desenvolva um algoritmo que, dadas duas árvores AVL $T_1$ e $T_2$, realize a união de ambas, devolvendo uma nova árvore AVL $T_3$ que contenha todos os elementos de $T_1$ e $T_2$, sem repetições.
2. **Interseção de Árvores AVL**: implemente um algoritmo que, dadas duas árvores AVL $T_1$ e $T_2$, devolva uma nova árvore AVL $T_3$ que contenha apenas os elementos presentes em ambas as árvores $T_1$ e $T_2$.
3. **Busca em Intervalo**: Implemente um algoritmo que, dada uma árvore AVL $T$, devolva todos os elementos cujos valores estejam dentro de um intervalo definido por dois valores $A$ e $B$.

## 2 Requisitos

- Todas as operações devem ser implementadas em C ou C++.
- O código deve ser modularizado, permitindo fácil leitura e entendimento das funções de inserção, remoção, busca, união e interseção.
- Inclua um programa principal que permita a inserção de dados manualmente ou por meio de um arquivo de entrada, permitindo realizar todas as operações.
- A árvore AVL deve ser exibida após cada operação, de modo que seja possível verificar visualmente o balanceamento.

## 3 Testes

Você deve submeter um conjunto de arquivos de entrada que contenham operações de inserção e remoção, além de testes específicos para verificar o funcionamento correto da união, interseção e busca em intervalos. Teste seu programa com diferentes tamanhos de árvores (pequenas, médias e grandes) para garantir a robustez e eficiência.

## 4 Avaliação

- **Implementação (50%)**: correção e eficiência das operações de união, interseção e busca em intervalos.
- **Testes (20%)**: qualidade e abrangência dos casos de teste.
- **Relatório (10%)**: clareza e detalhamento das decisões de implementação e análise do consumo de tempo.
- **Eficiência (20%)**: desempenho do programa em árvores de diferentes tamanhos e desempenho das operações.

## 5 Orientações gerais

O programa pode ser desenvolvido usando programação estruturada OU pode ser desenvolvido usando orientação a objetos. Caso opte por programação estruturada, use a linguagem C padrão. Caso escolha programação orientada a objetos, use a linguagem C++. Caso escolha a linguagem C++ você tem um bônus automático de `+0,5` ponto na sua Prova 2.

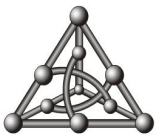
Não use códigos/funções/classes já existentes, especialmente caso escolha uma linguagem orientada a objetos. NÃO É PERMITIDO usar qualquer estrutura de dados já implementada, como por exemplo em bibliotecas do C++ ou de terceiros (**`vector`**, **`string`**, etc). Portanto, você deve implementar todas as estruturas de dados necessárias para seu programa.

Você DEVE, ao final do seu programa, liberar toda memória alocada dinamicamente tais como vetores, listas, e outras, e também garantir que seu programa não realiza acessos inválidos à memória. Para verificar essa questão, será utilizado o utilitário *Valgrind*, com o comando

```
valgrind --leak-check=full --show-reachable=yes --track-fds=yes ./meu_programa meu_codigo.cpp
```

(ou **`meu_codigo.c`**, se for o caso). Embora execute mais lentamente, você pode **compilar** seu programa com a opção de depuração **`-g`**, permitindo que o **`valgrind`** detalhe mais a saída, incluindo os números das linhas de seu programa com eventuais problemas.

O uso das estruturas de dados conforme as especificações propostas é OBRIGATÓRIO.





## 6 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

   Seu trabalho deve ter um cabeçalho com o seguinte formato:

```
/*********************************************
*
* Nome do(a)(s) estudante(s)
* Trabalho 2
* Professor(a): Nome do(a) professor(a)
*
*/
```

   ATENÇÃO! Se você fizer o trabalho 1 em dupla, o trabalho 2 deve ser feito com a mesma dupla, a não ser que um dos membros desista do curso. Neste último caso, o membro restante deve entregar o trabalho sozinho. NÃO troque de dupla durante o curso!

2. **Compilador**

   Para a correção do trabalho, será utilizado o compilador da linguagem C++ da coleção de compiladores GNU `g++`, com as opções de compilação `-Wall -pedantic -std=c++11` ao corrigir os programas. Ou o compilador da linguagem C da coleção de compiladores GNU `gcc`, com as opções de compilação `-Wall -pedantic -ansi`. Opcionalmente, você pode utilizar a flag `-g` para compilar seu programa e testá-lo pelo `valgrind`. Antes de entregar seu programa, verifique se ele tem extensão `.cpp` ou `.c`, compila sem mensagens de alerta e executa corretamente e não possui problemas de memória acusados pelo `valgrind`.

3. **Forma de entrega**

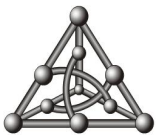
   A entrega será realizada diretamente na página da disciplina no AVA/UFMS. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico "Trabalhos", e escolha "Trabalho 2" e "Entrega". Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 59 minutos** do dia **28 de novembro de 2024**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.





6. **O que entregar?**

   Você, ou sua dupla, deve entregar o seguinte:

   - Código-fonte completo em C ou C++.
   - Arquivos de entrada de teste.
   - Relatório simples explicando as decisões de implementação e análise do consumo de tempo das operações.

   Você pode entregar todos os arquivos acima em um único arquivo compactado ou entregá-los separadamente, sem compactação, no mesmo local de entrega no AVA/UFMS.

7. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa-fonte**

   Seu arquivo contendo o programa-fonte deve estar bem organizado. Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação[1] de seu programa e de suas funções/métodos.

   Dê um nome significativo para seu programa e adicione a extensão adequada (`.cpp` ou `.c`) a este arquivo. Por exemplo, `operacoes_AVL.cpp` é um nome válido.

9. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE ou EM DUPLA. Cada estudante ou dupla tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir ideias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.

[1] Você sabe o que é documentação de um programa?