Disciplina: **Estrutura de dados 2**
Professor: **Jeferson José Baqueta**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Atividade prática 2

Especificações:

- Esse trabalho pode ser realizado em **duplas**;
- O valor total do trabalho representará 10% da nota do bimestre;
- O trabalho deve ser entregue em formato digital no Moodle (data da entrega: 29/11):
  - Compacte todos os arquivos para a entrega.
- Não serão aceitos, em hipótese alguma, trabalhos enviados via e-mail.
- Cópias de trabalhos serão penalizadas com a perda total do valor do trabalho.
- O trabalho enviado deve estar claramente identificado, com nome do aluno.

Objetivo:

Comparar o desempenho de estruturas de dados. No caso, uma árvore AVL e uma tabela hash.

AVL

A árvore AVL será composta por nodos. Cada nodo deve obrigatoriamente ser definido com base na seguinte estrutura:

```
typedef struct No{
    int chave;
    int altDireita;
    int altEsquerda;
    struct No* direita;
    struct No* esquerda;
} Nodo;
```

Tabela hash

A tabela hash deve ser implementada com base em um vetor dinâmico. Para simplificar sua implementação, considere que cada posição da tabela irá armazenar apenas o campo chave (desconsidere o campo valor). Como função hash utilize o método da divisão. Sua implementação deve ser capaz de lidar com colisões. Para isso, utilize a abordagem de endereçamento aberto, mais especificamente o método de sondagem linear visto em aula. Para evitar o esgotamento das posições livres, sempre que o fator de carga da tabela (α) chegar a 0.7, sua tabela deve dobrar de tamanho (implementação do redimensionamento para cima). Não esqueça que o redimensionamento altera o tamanho da tabela, e por isso as chaves precisam ser redistribuídas considerando o novo tamanho da tabela. Sua tabela deve ser iniciada com capacidade de 100 chaves (m).

Inserção

Implementar uma função **inserir**, a qual recebe um valor inteiro como argumento (chave) e insere tal chave na estrutura de dados. No caso da árvore AVL, certifique-se de manter as propriedades da árvore

após cada inserção. Portanto, sua implementação deve garantir que a árvore continue balanceada após cada inserção por meio das rotinas de rotação.

Busca

Implementar uma função **buscar**, a qual recebe um valor inteiro como argumento (chave de busca) e retorna *verdadeiro* se a chave de busca estiver presente na estrutura de dados. Se a chave não for encontrada, sua função deve retornar *falso*. Implemente também um mecanismo para contabilizar o número de comparações realizadas durante a operação de busca para ambas as estruturas de dados.

Remoção

Não é necessário.

Entrada de dados:

Seu programa deve ler as operações que serão realizadas em ambas as estruturas de dados a partir de arquivos de texto. Cada linha de um arquivo especifica qual operação será realizada, inserção de uma chave (i) ou busca por uma chave (b). Por exemplo, a linha “i 25\n” indica que a chave 25 deve ser inserida na estrutura de dados.

Três diferentes arquivos de entrada estão disponíveis no Moodle (in1.txt, in2.txt e in3.txt). As operações de inserção e busca são específicas para cada arquivo de entrada.

Saída de dados:

Além do código, um relatório de no máximo duas páginas deve ser produzido. Neste relatório você deverá responder às seguintes perguntas, considerando o número de comparações realizadas durante as operações de buscas:

1. Qual arquivo representa o pior e o melhor caso para a tabela hash? Justifique sua resposta.
2. Qual arquivo representa o pior e o melhor caso para a árvore AVL? Justifique sua resposta.
3. Se houver diferença de desempenho entre as estruturas de dados, explique o motivo dessa diferença?

Atividade extra (este trabalho passa a valer 20% ponto na média final)

Implemente o encadeamento separado para uma tabela hash de tamanho de fixo (100 posições). Utilize uma lista simplesmente encadeada para gerenciar as colisões entre chaves. Como função hash, utilize o método da divisão. Valide sua implementação com base nos arquivos de entrada disponíveis no Moodle e responda as perguntas definidas no item anterior considerando tal implementação.

Além do número de comparações realizadas, contabilize também o tempo de execução de todas as estruturas de dados implementadas para cada arquivo de entrada (árvore AVL, tabela hash redimensionável com endereçamento aberto e tabela hash de tamanho fixo com encadeamento). Em seguida, indique qual estrutura apresentou o melhor desempenho para cada arquivo de entrada. Explique o motivo da diferença de desempenho, se houver.