Universidade Federal do Maranhão
Departamento de Informática
Disciplina: ***Estrutura de Dados II 2024.2***
Prof.: ***João Dallyson Sousa de Almeida***

Atividade Prática 2 (Individual)
20% da segunda avaliação

1. DESCRIÇÃO

O problema da interseção de conjuntos consiste em encontrar elementos que pertençam simultaneamente a dois conjuntos A e B. Há diversas formas de resolvê-lo, sendo a mais simples provavelmente aquela em que, para cada um dos elementos de um dos conjuntos, percorre-se possivelmente todo o outro conjunto à medida que seus elementos vão sendo comparados. Seu programa deve permitir a entrada de caracteres/string ou números.

Você deverá IMPLEMENTAR um programa que resolver o problema de interseção de conjuntos utilizando lista encadeada, árvore binária, árvore rubro-negra e hashing.

Seu programa deverá carregar as chaves que serão lidas via arquivo. O arquivo que tiver menor quantidade de chaves será sempre armazenado em uma lista encadeada (Conjunto A). Já o arquivo com maior quantidade de chaves (Conjunto B) deverá ser carregado em:

1. Lista encadeada (OBS: neste item você pode usar as classes nativas LinkedList ou ArrayList)
2. Árvore AVL (TreeAVL)
3. Árvore Rubro-Negra (TreeRB)
4. Hashing

Implemente as seguintes operações:

1. Buscar os elementos de A que estão em B
2. Inserir em B, os elementos de A que não estão em B
3. Remover os elementos de A que estão em B

Discuta no relatório os resultados a complexidade e o custo em tempo de execução para realizar as operações utilizando cada uma das 4 estruturas de dados utilizadas.

Sobre a implementação:

1. Você pode usar as classes LinkedList ou ArrayList, apenas para a lista encadeada.
2. Você deve criar a interface abaixo que será implementadas pelas Classes TreeAVL e TreeRB

```java
public interface BalancedTree <T extends Comparable<T>> {

    void insert(T value);

    boolean remove(T value);

    boolean find(T value);

    int getHeight();

    void printInOrder();


}
```

3. O nó da Árvore AVL deve ter a seguinte estrutura:

```java
public class NodeAVL <AnyType> {

     AnyType element;       // Dados do no
     NoAVL<AnyType> left;   // Filho a esquerda
     NoAVL<AnyType> right;    // Filho a direita
    int   height;              //Altura

    NodeAVL( AnyType e )
    {
        this (e, null, null );
    }

    NodeAVL(AnyType e, NodeAVL left, NodeAVL  right)
    {
        element  = e;
        left     = left;
        right    = right;
        height   = 0;
    }

}
```

4. O nó da Árvore Rubro-Negra deverá ter a seguinte estrutura:

```java
public class NodeRB <AnyType> {

    AnyType element;
    NodeRB<AnyType> parent, left, right;
    Color color;
    int N;              // conta subarvores

    enum Color {
        RED, BLACK
    }

    NodeRB( AnyType e ){
        this (e, null, null, null, false);
    }

    NodeRB(AnyType e, NodeRB l, NodeRB  r, NodeRB  p, boolean c){
        element  = e;
        left     = l;
        right    = r;
        parent    = p;
        color = Color.RED;
       // N;              // subtree count
    }
}
```

5. Você deverá usar a implementação HashTentativaLinear como base, disponível em: https://drive.google.com/file/d/1HD6IlwR6QcTw1W9ybyV984yRc3uoLAe9/view?usp=sharing Observe que a entrada de dados poderá ser de caracteres/string ou números. Você deverá escolher o método de tratamento de colisões e a função de hashing que será utilizada.

2. Descrição sobre os arquivos de dados

Você deve utilizar arquivos de tamanho pequeno, médio e grande.

3. Análise dos resultados

A análise deve ser feita sobre o número de comparações, atribuições e tempo de execução dos algoritmos. Procure organizar os dados coletados em tabelas/gráficos a partir dos dados.

4. Entrega

- Código fonte do programa em Java (bem identado e comentado).

- Relatório dos resultados do trabalho
- Upload no SIGAA.

O Relatório deve apresentar:

1. Análise de complexidade: apresentar o estudo de complexidade das principais funções implementadas usando a notação O.
2. Testes: apresentação dos testes realizados.
3. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação