Universidade Estadual de Maringá – Departamento de Informática
6884/3 – Estruturas de dados – Ciência da Computação
Prof. Marco Aurélio

Trabalho 02 - Conjuntos

Objetivos

O objetivo do trabalho é avaliar a capacidade do aluno de:

- Implementar e utilizar árvores binárias de busca
- Implementar e utilizar tabelas de dispersão
- Implementar e utilizar um algoritmo de ordenação eficiente

Instruções

O trabalho é em equipe de até duas pessoas e deve ser entregue no classroom até às 23 horas e 59 minutos do dia anterior ao dia da segunda prova.

Cada aluno deve agendar a data da entrevista em uma planilha que será disponibilizada posteriormente.

Trabalhos que não tenham sido feitos pela equipe em sua totalidade serão zerados. O chatgpt e ferramentas a fins não podem fazer parte de nenhum equipe!

Descrição

O seguinte TAD descreve um conjunto de números inteiros e algumas de suas operações

```
class Conjunto:
    '''
    Uma coleção de números inteiros distintos.

    Exemplos

    >>> c1 = Conjunto()
    >>> c1.insere(10)
    >>> c1.insere(3)
    >>> c1.insere(10)
    >>> c1.insere(12)
    >>> c1.insere(7)
    >>> c1.insere(3)
    >>> c1.em_ordem()
    [3, 7, 10, 12]
    >>> c2 = Conjunto()
    >>> c2.insere(20)
    >>> c2.insere(3)
    >>> c2.insere(1)
    >>> c2.insere(10)
    >>> c2.em_ordem
    [1, 3, 10, 20]
    >>> c1.intersecao(c2).em_ordem()
    [3, 10]
    >>> c1.remove(12)
    >>> c2.remove(12)
    >>> c1.uniao(c2).em_ordem()
    [1, 3, 7, 10, 20]
    '''

    def __init__(self) -> None
        '''Cria um novo conjunto vazio'''

    def insere(self, valor: int) -> None:
        '''Insere *valor* no conjunto'''

    def remove(self, valor: int) -> None:
```

```
        '''Remove o *valor* do conjunto se ele estiver presente.'''

    def intersecao(self, outro: Conjunto) -> Conjunto:
        '''Cria um novo conjunto com os elementos que *self* e *outro* têm em comum.'''

    def uniao(self, outro: Conjunto) -> Conjunto:
        '''Cria um novo conjunto com os elementos de *self* e *outro*.'''

    def em_ordem(self) -> list[int]:
        '''Devolve uma lista com os elemento do conjunto em ordem.'''
```

Atividades

O trabalho consiste em:

a) Implementar o TAD Conjunto usando ABB (crie uma cópia do arquivo `conjunto_tad.py` com o nome `conjunto_abb.py`).
   - A implementação das operações deve tirar proveito da propriedade de busca.
   - Se você utilizar árvore AVL, pode ganhar 2 pontos extra na nota do trabalho!

b) Implementar o TAD Conjunto usando tabelas de dispersão (crie uma cópia do arquivo `conjunto_tad.py` com o nome `conjunto_dispersao.py`) .
   - A implementação dos métodos `intersecao` e `uniao` devem ter tempo de execução de $O(n)$, onde n é a soma das quantidades de elementos dos conjuntos
   - A implementação do método `em_ordem` deve usar ou o quicksort com o esquema de particionamento de Tony Hoare, ou a ordenação por heap sem recursividade.