Relatório — Projeto ABB_SBD

Nome: Bruno Videira Pinho
DRE: 119.161.539

Objetivo

Este projeto implementa uma **Árvore Binária de Busca (ABB)** para indexação e manipulação eficiente de registros de pessoas em um sistema de arquivos simulado em memória. O objetivo é demonstrar operações clássicas de ABB (inserção, busca, remoção, travessias) integradas a um sistema de armazenamento de registros, simulando um pequeno SGBD (Sistema de Gerenciamento de Banco de Dados).

Estrutura de Diretórios

```
ABB_SBD/
├── ABB.py          # Arvore binária
├── edl.py          # Funções principais
├── file_sys.py     # Lógica do sistema de arquivos
├── node.py         # Nó da arvore binária
├── person.py       # Classe que armazena informação das pessoas (cpf,
nome, ...)
└── record.py       # Classe que interliga a arvore binária com o sistema
de arquivos
```

Principais Rotinas e Métodos

ABB (Árvore Binária de Busca)

- `insert(record: Record)`: Insere um novo registro na árvore.
- `find(cpf: int) -> Optional[Record]`: Busca um registro pela chave (CPF).
- `remove(cpf: int)`: Remove um registro da árvore pela chave (CPF).
- `orded_record_list() -> list[Record]`: Retorna uma lista ordenada dos registros.
- `breadth_traversal() -> list[Record]`: Retorna os registros em 'percurso em largura'.
- `clear()`: Limpa toda a árvore.

FilesSys

- `insert(person: Person) -> int`: Insere uma pessoa e retorna o índice.
- `find_person(idx: int) -> Optional[Person]`: Busca uma pessoa pelo índice.
- `delete(idx: int)`: Marca um registro como deletado (deleção lógica).
- `list() -> list[Person]`: Retorna todos os registros.

Integração (edl.py)

- `inserir_person(tree, file, person)`: Insere uma pessoa no sistema (arquivo + ABB).
- `remove_person(tree, file, cpf)`: Remove uma pessoa do sistema (deleção lógica + ABB).

- `find_cpf(tree, file, cpf)`: Busca uma pessoa pelo CPF.
- `export_orded_data(tree, file) -> list[Person]`: Exporta os dados ordenados (ignorando deletados).

Fluxo de Operações

1. **Inserção**:
    - Um novo objeto `Person` é criado e inserido no `FilesSys`, que retorna o índice.
    - Um objeto `Record` (CPF, índice) é criado e inserido na ABB.
2. **Busca**:
    - A ABB é consultada pelo CPF. Se encontrado, o índice é usado para buscar o registro no arquivo.
3. **Remoção**:
    - O registro é marcado como deletado no arquivo e removido da ABB.
4. **Exportação Ordenada**:
    - A ABB é percorrida em ordem, e os registros válidos (não deletados) são exportados.

Complexidade de Tempo e Espaço

- **ABB**:
    - Inserção, busca e remoção: O(h), onde h é a altura da árvore.
- **FilesSys**:
    - Inserção e acesso por índice: O(1).
    - Deleção lógica: O(1).

Observações e Limitações

- **Deleção lógica**: Os registros não são removidos fisicamente do arquivo, apenas marcados como deletados.
- **Sem balanceamento**: A ABB não é balanceada, podendo degradar para O(n) em casos degenerados.
    - Dependendo da ordem de inserção dos elementos (`Record`) na arvore.
    - No exemplo que utilizei (pelos valores de cpf estarem ordenados de forma crescente) todos os nós foram inseridos no 'galho' mais a direita.
- **Integração**: O sistema simula um SGBD básico, mas todo armazenamento é feito em memória.
- **Travessia ordenada**: A função `orded_record_list` faz travessia in-order, garantindo ordenação por CPF.

Conclusão

O projeto ABB_SBD demonstra a integração entre uma árvore binária de busca e um sistema de arquivos simplificado, permitindo operações eficientes de inserção, busca, remoção e exportação ordenada de registros.