Implementação de Heap em C

Aluno: Gabriel Baeta Pereira

1. Visão Geral do Projeto

Este projeto consiste em uma implementação completa de uma estrutura de dados de Heap (Max-Heap) em C, projetada para gerenciar de forma eficiente um grande volume de registros em memória secundária. A estrutura é focada nas operações fundamentais de uma Fila de Prioridades: inserção de novos elementos, remoção do elemento de maior prioridade (neste caso, a maior nota) e busca.

A implementação foi modularizada em três arquivos (HEAP.h, HEAP.c, main.c).

2. Decisões de Design e Arquitetura

2.1. Estruturas de Dados

A fundação do projeto reside na struct Dados e na representação implícita do heap:

- **Dados**: Dados: Modela o registro apresentado. A escolha por arrays de char de tamanho fixo (cpf[13], nome[50]) é uma abordagem direta para o escopo do trabalho, evitando a complexidade do gerenciamento dinâmico de memória para cada registro.
- **Estrutura Implícita de Árvore**: Diferente de estruturas baseadas em nós com ponteiros, o Heap é implementado como uma árvore binária quase completa armazenada sequencialmente em um único arquivo. As relações entre pais e filhos não são mantidas por ponteiros, mas calculadas por meio de fórmulas matemáticas.

2.2. Gerenciamento de Arquivos

Foi adotada uma arquitetura de múltiplos arquivos para separar as responsabilidades:

- **Arquitetura de Três Arquivos**: Foi adotada uma separação entre os dados brutos, o índice (heap) e o controle de tamanho:
    - dados.bin: Contém os registros de Dados completos e não ordenados, servindo como fonte para a construção inicial.
    - heap.bin: Armazena os nós da estrutura de Heap, organizados para manter a propriedade de Max-Heap.
    - reg_heap.bin: Um arquivo auxiliar dedicado a armazenar um único valor: o tamanho atual do heap.

3. Análise dos Algoritmos Implementados

3.1. Inserção (escrever e subir)

A inserção adiciona um novo elemento ao heap mantendo a propriedade de Max-Heap. O processo segue dois passos:

1. **Adição na Base**: O novo elemento é sempre inserido no final do arquivo, o que corresponde a adicioná-lo na primeira posição livre da última camada da árvore.
2. **Heapify-Up (subir)**: Após a inserção, a função subir é chamada. Ela compara o novo nó com seu pai e, se o filho for maior, realiza uma troca. O processo se repete, "subindo" o elemento na árvore até que ele encontre sua posição correta ou chegue à raiz.

3.2. Busca (busca)

A busca por um CPF específico foi implementada, mas é importante notar que o Heap não é uma estrutura otimizada para buscas de chaves arbitrárias. O algoritmo implementado realiza uma **busca linear**, percorrendo o arquivo heap.bin desde o início até encontrar o CPF desejado ou atingir o final do arquivo. Seu desempenho é O(n), contrastando com a busca logarítmica de árvores balanceadas.

3.3. Remoção (remover e descer)

A remoção é a operação mais complexa e, em um Max-Heap, corresponde exclusivamente a remover o elemento de maior valor (a raiz). Os passos são:

1. **Localização e Troca**: O elemento na raiz (posição 0) é substituído pelo último elemento do heap.
2. **Redução de Tamanho**: O tamanho do heap é decrementado, efetivamente removendo o último elemento (que agora está na raiz).
3. **Heapify-Down (descer)**: A função descer é chamada na raiz. Ela compara o novo elemento da raiz com seus filhos e o troca com o maior deles. Esse processo de "descer" continua recursivamente até que o nó seja maior que seus filhos ou se torne uma folha, restaurando a propriedade de Max-Heap.

3.4. Testes (main.c)

A função main foi projetada não apenas como um ponto de entrada, mas como um **ambiente de testes robusto**. A metodologia consiste em primeiro gerar um arquivo com 10.000 registros e, em seguida, usar a função constroi para organizar esses dados em um Max-Heap válido. Após a construção, o sistema oferece um menu interativo que permite ao usuário testar as funcionalidades de inserir novos alunos, remover o aluno de maior nota e buscar por um CPF em tempo real.

https://github.com/GabrielBaetaP/Trab_EDA/tree/main/TRAB_EDA_HEAP