Trabalho 1 - Algoritmo e Estruturas de Dados II

Profa. Elaine Parros

Estagiario PAE: Evandro Ortigossa

Aluno: Sen Chai - NUSP 10727830

Utilização

População automática da arvore

- A opção 3 da interface permite por padrão inserção de aproximadamente 150~200 registros com valores gerados por um script em python(`populate.py`)
- Esse script gera o um C file header chamado `populate.h` que serve como array pre definida da estrutura dos registros, contudo é fácil gerar um novo rodando o mesmo script. Para aumentar o tamanho de elementos, basta fornecer uma lista de nomes pelo arquivo `names.input` .
    - Podem ser encontrados na pasta `testing` .

Utilização do código

- Utiliza compilador gcc
- `make clear` limpa todos arquivos de dados `.dat` e `.o` existentes
- `make go` compila e executa o programa principal

OBS

- Não inserir valores inapropriados para os campos, tal como: valores numéricos em campos de texto ou texto em campos numéricos;

Implementação e Discussão

Arquivos de dados

- `index_header.dat` mantem cabeçalho da arvore
- `index.dat` guarda páginas da btree
- `data.dat` mantem registros dos estudantes
- Formato utilizado é o de texto, pois assim é mais fácil de enxergar o estado atual das nossas estruturas de dados, inclusive foi empregado separadores entre as linhas para facilitar e transparecer o estado atual da arvore.
    - Espaços em brancos são inseridos em locais vazios

Estrutura de dados Utilizada

Nos/Páginas da arvore:

```
struct PAGE{
    int isparent;
    int pagerrn;
    int nkeys;
    int *keys;
    int *rrns;
    int *children;
};
```

A pagina contem respectivamente se esta é pai, seu rrn dentre as páginas de arquivo de índice, quantas chaves está comportando, e as respectivas chaves, referências ao rrn do arquivo de dados e seus filhos.

Estrutura dos cabeçalhos da arvore:

```
struct BTREE {
    int last_page_rrn;
    int last_data_rrn;
    int height;
    int root;
};
```

Esta estrutura está sempre carregada em memória durante execução,

Estrutura do Registro de aluno:

```
struct STUDENT{
    int numUSP;
    char name[STRFIELD];
    char surname[STRFIELD];
    char course[STRFIELD];
```

```
    float grade;
} ;
```

Tamanhos utilizados para INT e STRFIELD são 10 e 16 respectivamente, então nenhum valor numérico deve ter mais de 10 dígitos, e nenhuma string deve ter mais de 16 caracteres

Inserção de chaves já existentes:

- Não fora implementada a possibilidade de atualização, uma vez que sua presença não era necessária no escopo do projeto

Busca e Inserção:

Tanto a busca, quanto a inserção são iterativas, e em contraste com as abordagens recursivas, as perdas ou ganhos em desempenho não devem ser significativas, uma vez que o número de consulta a disco para ambas será igual.