

Índice

1. Arvore B, sua Implementação.
2. Análise do Algoritmo com o Intuito de Armazenar Dados Estatísticos
3. Gerador de Registros / Fisher-Yates Shuffle
4. Gerador de Gráficos
5. Utilização do Programa Principal
6. Artigo Científico Escolhido

1. Implementação da Árvore B:

Arquivos (b_tree.h e b_tree.c).

Foi utilizado o código disponibilizado pelo professor Guilherme[4], com base neste código foram implementadas de três novas funções, BTREE_SSEARCH, BTREE_HHEIGHT e BTREE_HEIGHT, a primeira para auxiliar na contagem de acessos ao disco ignorando a raiz. As duas posteriores para o cálculo da altura. Foram também modificadas as funções originais ao código para a inserção de flags com o intuito de auxiliar no processo de contagem dos acessos ao disco.

2. Análise do Algoritmo com o Intuito de Armazenar Dados Estatísticos

Arquivos (statistic.h, statistic.c e statistic.txt);

Modelo de Análise:

1.1

Se houver leitura no primeiro acesso, pagina é levada para a memória DISKREAD++
Se houver outra leitura, pagina está na memória NULL

1.2

Se houver escrita após a leitura
-Pagina é atualizada primeiro na memória NULL
-Pagina é atualizada posteriormente no disco DISKWRITE++

2.1

Se houver escrita no primeiro acesso
- Pagina é levada pra memória DISKREAD++
- Pagina é atualizada na memória NULL
- Pagina é atualizada no disco DISKWRITE++

2.2

Se houver leituras após a escrita, pagina está na memória NULL

O cabeçalho apresenta as principais funções para a criação de uma estatística referente à comparações e acessos de leitura e escrita no disco.

```
void STATS_RESET();
void STATS_PRINT();
int STATS_SCAN(char selection);
void STATS_DISKREAD(int* isRoot);
void STATS_DISKWRITE(int* isRoot);
void STATS_COMPARISON();
```

A função STATS_RESET zera os valores do arquivo statistic.txt. Exemplo:

```
////////////////////////Arquivo////////////////////////////////////////
Reads: 0
Writes: 0
Comparisons: 0
////////////////////////Fim do Arquivo////////////////////////////////////////
```

A função STATS_PRINT imprime os dados contidos no arquivo statistic.txt no terminal.

A função STATS_SCAN retorna um valor específico referente à leitura, escrita ou comparações os quais podem ser acessados através dos parâmetros 'r', 'w' e 'c'.

A função STATS_DISKREAD incrementa o valor de leituras de uma unidade, caso fosse aplicado após a função STATS_RESET geraria o resultado:

```
////////////////////////Arquivo////////////////////////////////////////
Reads: 1
Writes: 0
Comparisons: 0
////////////////////////Fim do Arquivo////////////////////////////////////////
```

As funções STATS_DISKWRITE e STATS_COMPARISON executam a mesma função porém incrementando os valores de "Writes" e "Comparisons".

3. Gerador de Registros / Fisher-Yates Shuffle

Arquivos (registry.h, registry.c, registries_generator.h, registries_generator.h, registries_generated.txt, names_database.txt, Fisher_Yates.h, Fisher_Yates.c);

O cabeçalho registry.h define o TAD "Registry" que é utilizado por todo o programa. Ele contém os protótipos de funções para imprimir o valor de um registro no terminal (REG_PRINT), imprimir o valor de um registro em um arquivo (REG_FPRINT) e obter um registro de um arquivo (REG_FSCAN).

```
typedef struct{
    int admission_number;
    char name[NAME_LENGTH];
    char admission_date[ADMISSDATE_LENGTH];
    int age;
    char occupation[OC_LENGTH];
    char civil_status[CIVILSTAT_LENGTH];
}Registry;
```

Os arquivos registries_generator.h e registries_generator.c tem como finalidade criar registros com informações aleatórias utilizando como base uma lista de 10.000 nomes, e gerando como saída 10.000 registros do tipo “Registry”.

Os arquivos Fisher_Yates.h e Fisher_Yates.c tem como finalidade implementar o algoritmo “Fisher-Yates Shuffle”[3][5], que consiste em um método para embaralhar números a um custo de $O(n)$.

Com isto é possível gerar registros com chaves diferentes em toda a execução a um custo baixo na função REGEN_GEN no arquivo registries_generator.c.

4. Gerador de Gráficos

Arquivos (graph_generator.h, graph_generator.c, graph_data.html);

Os arquivos graph_generator.h e graph_generator.c trabalham com três funções:

```
#define GRAPHSTEP 500

void GRAPH_DATA_RESET();
void GRAPH_ADDLINE(int reg_counter);
void GRAPH_GEN();
```

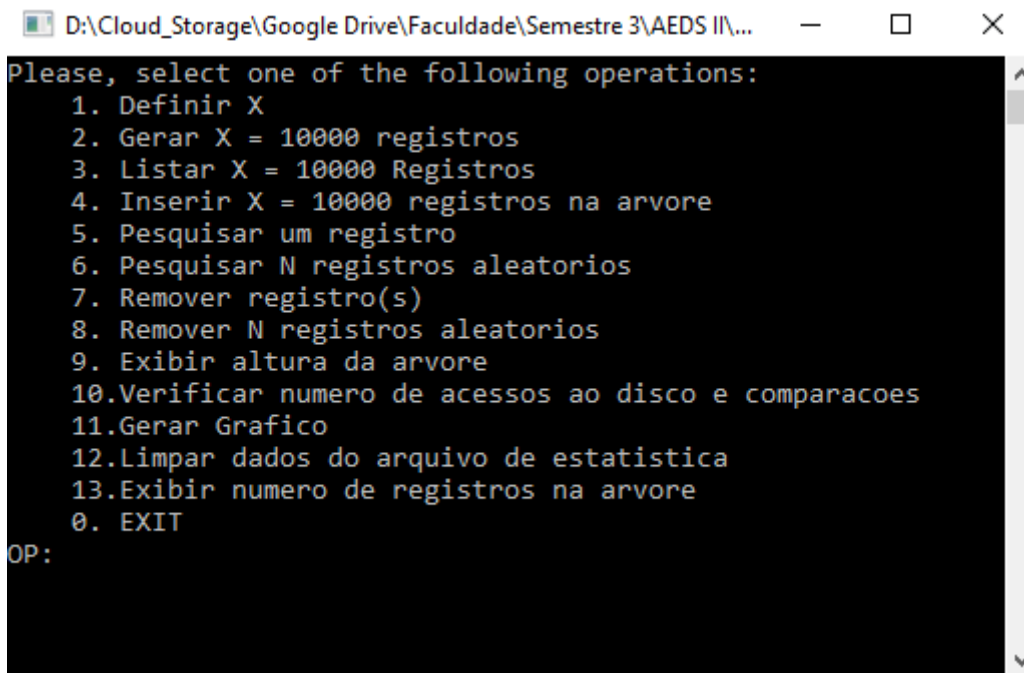
A função GRAPH_DATA_RESET limpa todos os dados coletados na realização de uma operação da Árvore B contidos no arquivo graph_data.html

A função GRAPH_ADDLINE adiciona uma nova linha de dados a ser usada para construir o gráfico, com o auxílio do valor definido GRAPHSTEP o qual define que a cada 500 registros será criada uma linha de dados, no caso de 10.000 registros, 20 linhas de dados seriam usadas para a construção do gráfico.

A função GRAPH_GEN gera o gráfico com os dados obtidos no arquivo graph_data.html, imprimindo um código html com parâmetros moldados a partir deles.

5. Utilização do Programa Principal

Arquivos (main.c);



```
D:\Cloud_Storage\Google Drive\Faculdade\Semestre 3\AEDS II\...
Please, select one of the following operations:
 1. Definir X
 2. Gerar X = 10000 registros
 3. Listar X = 10000 Registros
 4. Inserir X = 10000 registros na arvore
 5. Pesquisar um registro
 6. Pesquisar N registros aleatorios
 7. Remover registro(s)
 8. Remover N registros aleatorios
 9. Exibir altura da arvore
10. Verificar numero de acessos ao disco e comparacoes
11. Gerar Grafico
12. Limpar dados do arquivo de estatistica
13. Exibir numero de registros na arvore
 0. EXIT
OP:
```

Antes de começar é possível alterar número de registros a serem gerados/inseridos. Para fazer isto selecione 1 para redefinir “X”.

Após ter definido o número de registros, selecione 2 para gerá-los.

PS: É necessário gerar os registros antes de inseri-los na árvore.

É possível visualizar os registros gerados selecionando 3.

Selecione 4 para inseri-los na árvore.

Agora é possível:

- Pesquisar por um registros específico ou N registros.

- Remover registros específicos ou N registros.

- Verificar a altura da árvore.

- Verificar o número total de acessos ao e comparações até o momento.

- Gerar o gráfico com os dados calculados até o momento.

- Limpar os arquivos graph_data.html e statistic.txt para obter o número de acessos ao disco e comparações de uma operação específica.

- Exibit a quantidade de registros ainda inseridos na árvore.

PS: A remoção de N registros foi feita para ser executada somente uma única vez por execução, de preferência sendo a última operação a ser executada.

Depois selecione uma das operações

Para inserir 10.000 registros na árvore, selecione 4.

6. Artigo Científico Escolhido:

Foi escolhido o artigo Comer, “The Ubiquitous B-Tree”, 1971 [1], por abranger a maior parte do conteúdo visto neste semestre, como a Árvore B e suas variações, compressão e o custo de operações o qual ajudou a definir uma melhor perspectiva na forma de como organizar as funções para cálculo dos acessos ao disco.

Referências

- [1] Comer, "The Ubiquitous B-Tree", 1971
- [2] Bayer & McCreight, "Organization and Maintenance of Large Ordered Indexes", 1972
- [3] Ronald Fisher and Frank Yates, "*Statistical tables for biological, agricultural and medical research.*", 1978
- [4] G. T. de Assis. Pesquisa Externa. (Notas de Aula) UFOP. 2013. Disponível em< <http://www.decom.ufop.br/guilherme>>.
- [5] *Durstenfeld, R. (July 1964). "Algorithm 235: Random permutation".*