Relatório do Trabalho Prático I

Alunos:
Gabriel Scotá Arruda - 19.1.4054
João Paulo Prata Costa - 19.1.4020
Murilo de Paula Vieira Neto - 19.1.4011

Método 1 - Acesso Sequencial Indexado

O acesso sequencial indexado como o próprio nome já diz é um método de pesquisa onde cada item é lido sequencialmente até encontrar uma chave maior ou igual a chave pesquisada. Dentro deste método alguns requisitos são necessários para aumentar a eficiência, como o arquivo já estar ordenado pelo campo da chave do item e a criação de um índice de páginas, contendo o valor da chave de um item e o endereço da página na qual o primeiro item contém este valor. Esse índice de páginas faz com que a leitura na memória secundária seja feita uma primeira vez para carregar todos os itens e as próximas vezes indo mais diretamente ao endereço da chave, economizando leituras.

A seguir são apresentados alguns testes realizados para mostrar a complexidade do algoritmo em arquivos onde os registros estavam organizados de forma ascendente, decrescente e randômica.

Registros de forma ascendente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	57	0.0010 s	101	1	0.0090 s
1000	678	0.0040 s	1001	2	0.0260 s
10000	7824	0.0220 s	10001	4	0.0180 s
100000	37824	0.1920 s	100001	4	0.0240 s
1000000	924824	1.9950 s	1000001	4	0.0140 s

Registros de forma decrescente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	57	0.0020 s	101	4	0.0010 s
1000	678	0.0040 s	1001	3	0.0110 s
10000	7824	0.0240 s	10001	1	0.0030 s
100000	37824	0.1970 s	100001	1	0.0200 s
1000000	924824	2.0680 s	1000001	1	0.0170 s

Registros de forma randômica					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	57	0.0020 s	120	20	0.0160 s
1000	678	0.0030 s	1532	532	0.0190 s
10000	7824	0.0240 s	10368	368	0.0240 s
100000	37824	0.2630 s	120093	20093	0.0260 s
1000000	924824	3.7050 s	1860144	860144	1.3900 s

Através destes testes foi possível perceber que mesmo os registros estando organizados de forma ascendente ou decrescente os tempos de pesquisas são bem parecidos devido a tabela de índices que pode ser construída de forma correta, porém, quando se trata de registros com chaves randômicas, o processo de busca se torna mais custoso por ter que verificar chave uma por uma mesmo após ter criada uma tabela de índices onde as chaves não ajudam no processo de busca.

Uma das principais dificuldades de implementação desse método foi fazer a pesquisa com as possíveis situações, onde o arquivo de registros estava organizado de maneira decrescente ou randômica, fazendo com que tenha que adicionar algumas estruturas adicionais para verificar essas duas situações.

Método 2 - Árvore Binária

A árvore binária em arquivo, é uma simulação de árvore binária feita em arquivo binário. Cada nodo gravado no arquivo, possui dois apontadores, para seu filho à direita e à esquerda.
Assim como a árvore binária em memória principal, a árvore em arquivo possui uma ordem de complexidade igual a log(n);

A seguir são apresentados alguns testes realizados para mostrar a complexidade do algoritmo em arquivos onde os registros estavam organizados de forma ascendente, decrescente e randômica.

Registros de forma ascendente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	41	0.0010 s	41	41	0.0010 s
1000	883	0.0010 s	883	883	0.0010 s
10000	1418	0.0080 s	1418	1418	0.0110 s

Registros de forma decrescente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	41	0.0010 s	60	60	0.0020 s
1000	883	0.0010 s	168	168	0.0010 s
10000	1418	0.0080 s	8583	8583	0.0730 s

Registros de forma randômica					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	41	0.0010 s	6	6	0.0020 s
1000	883	0.0030 s	20	20	0.0010 s
10000	1418	0.0080 s	13	13	0.0010 s

Através destes testes foi possível perceber que quando os registros estão ordenados de forma crescente ou decrescente, a busca ocorre em tempo linear, pois todos os nodos estão colocados em um mesmo lado da árvore. Já quando os arquivos são implementados de forma aleatória, o tempo de busca se altera, pois os registros se distribuem entre os dois lados da àrvore.

Uma das principais dificuldades de implementação desse método foi fazer a alteração dos apontadores para a posição dos filhos. Identificar a posição do pai de um nodo que seria inserido também foi um desafio.

Método 3 - Árvore B

A Árvore B é um algoritmo bem viável quando se possui uma grande quantidade de dados e as chaves não conseguem ser armazenadas somente em memória principal, pois com seu método de pesquisa e um sistema de paginação, ele consegue otimizar bastante o tempo de busca em memória secundária (Otimizada para acesso a grandes volumes de dados em disco e consegue armazenar índice e dados na mesma estrutura ou seja no mesmo arquivo físico).

Para poder ter uma noção da complexidade dos algoritmos realizamos testes , calculando o número de comparações e o tempo necessário para executar as funções:

No caso da Árvore B calculamos o tempo para montar a Árvore e para realizar uma pesquisa na mesma, os dados foram:

Registros de forma ascendente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	98	0.000 s	100	17	0.000 s
1000	990	0.0030 s	1000	17	0.000 s
10000	990	0.0130 s	10000	20	0.000 s
100000	790	0.2160 s	100000	32	0.000 s
1000000	7790	6.0210 s	1000000	41	0.000 s

Registros de forma decrescente					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	77	0.0030 s	100	9	0.000 s
1000	678	0.0010 s	1000	20	0.0000 s
10000	7824	0.0120 s	10000	26	0.0000 s
100000	37824	0.1350 s	100000	36	0.0000 s
1000000	924824	5.2240 s	1000000	40	0.0000 s

Registros de forma randômica					
Qtd. Registros	Chave pesquisada	Tempo para criar o arquivo	Número de leituras na memória externa	Total de comparações entre as chaves	Tempo gasto na pesquisa
100	57	0.0110 s	100	13	0.000 s
1000	678	0.0020 s	1000	18	0.000 s
10000	7824	0.0140 s	10000	21	0.000 s
100000	37824	0.1340 s	100000	33	0.000 s
1000000	924824	4.8620 s	1000000	44	0.000 s

As principais dificuldades foram encontradas na leitura do arquivo, onde usei o mesmo para inserir minhas chaves dentro da árvore, montando assim um sistema de mapeamento, outra dificuldade foi desbravar os erros de segmentação e passagens por referência de maneira incorreta.

Método 4 - Árvore B*

A árvore B* (B estrela) é outra alternativa para implementar a árvore B, porém, nessa árvore todos os registros estão armazenados nas páginas folhas (último nível da árvore), e os níveis acima do último são construídos contendo um índice responsável pela organização de uma árvore B.

Uma das principais dificuldades de implementação desse método foi realizar a leitura da árvore e a diferenciação de página interna ou externa na hora de construir a árvore nos métodos de inserção.