Universidade Federal de Ouro Preto - UFOP

Instituto de Ciências Exatas e Biológicas - ICEB

Departamento de Computação - DECOM

Ciência da Computação

Relatório TP1

BCC203 - Estrutura de Dados 2

Mariana Macedo Santos, Gabriel Araújo Saldanha, Gustavo Zacarias, Maria Eduarda Bessa, Gabriel Henrique Rocha, João Victor

Professor: Guilherme Tavares

Ouro Preto
27 de junho de 2023

Sumário

Lista de Figuras

1 Introdução

O trabalho prático consiste em um estudo da complexidade de desempenho de diferentes métodos de pesquisa externa. Os métodos implementados e abordados são: (1) acesso sequencial indexado, (2) árvore binária de pesquisa adequada à memória externa, (3) árvore B e (4) árvore B*. O objetivo é implementar esses métodos em linguagem C, considerando arquivos binários de registros e a disponibilidade de memória interna para armazenar os índices necessários. Serão realizados experimentos para analisar o desempenho desses métodos em termos de transferências de dados entre a memória externa e interna, comparações de chaves de pesquisa e tempo de execução.
(1)Acesso Sequencial Indexado: o procedimento consiste em percorrer as chaves uma a uma até encontrar aquela desejada ou uma que seja maior. A fim de garantir uma eficiência satisfatória, é fundamental que o arquivo esteja previamente ordenado.
(2) Árvore binária: executa a operação de árvore binária utilizando uma abordagem de armazenamento em um arquivo secundário, em vez de utilizar a memória interna. Neste arquivo adicional, serão incluídos dois campos adicionais, juntamente com a chave, que representarão a linha onde está localizado o filho esquerdo ou direito do item correspondente que está sendo analisado.
(3) Árvore B: emprega um método que assegura o equilíbrio constante da árvore, permitindo apenas crescimento ascendente. Com uma ordem definida como m (utilizando o valor 2), a árvore B terá uma página raiz contendo de 1 a 2m (ou seja, 4) itens, enquanto as demais páginas conterão, no mínimo, m (2) itens e m + 1 (ou seja, 3) descendentes, podendo conter no máximo 2m (ou seja, 4) itens e 2m + 1 (ou seja, 5) descendentes.
(4)Arvore B*: uma abordagem alternativa para a implementação da árvore B, que se distingue pelo armazenamento de todos os registros no último nível, conhecido como páginas folhas ou páginas externas. Nos níveis superiores, essa abordagem inclui índices das chaves, organizados como uma árvore B com páginas internas.

1.1 Entrada

A entrada no terminal para executar o programa "pesquisa"segue a seguinte sintaxe:

terminal

```
pesquisa <método><quantidade><situação><chave>[-P]
```

Onde cada valor representa:

- **<método>:** É um número inteiro de 1 a 4 que representa o método de pesquisa externa a ser executado: (1) acesso sequencial indexado, (2) árvore binária, (3) árvore B e (4) árvore B*.
- **<quantidade>:** É a quantidade de registros do arquivo que será considerada nos experimentos. Pode ser um dos valores: 100, 1.000, 10.000, 100.000 ou 1.000.000.
- **<situação>:** Representa a situação de ordem do arquivo. Pode assumir os seguintes valores:
  - 1: Arquivo ordenado ascendentemente pela chave de pesquisa.
  - 2: Arquivo ordenado descendentemente pela chave de pesquisa.
  - 3: Arquivo desordenado aleatoriamente pela chave de pesquisa.
- **<chave>:** É a chave a ser pesquisada no arquivo considerado.
- **[-P] (opcional):** É um argumento que indica se as chaves de pesquisa dos registros do arquivo devem ser apresentadas na tela. Se este argumento estiver presente, as chaves serão exibidas.

Após a execução do comando, o programa retornará se a chave de pesquisa desejada foi encontrada ou não no arquivo.

1.2 Registros

O arquivo contém registros que possui vários campos. Os campos presentes nos registros são os seguintes:

- **Chave:** É um valor inteiro que serve como identificador único para cada registro. A chave é utilizada como referência para a pesquisa nos diferentes métodos de busca externa implementados.
- **Dado1:** É um valor inteiro longo que representa um dado associado ao registro.
- **Dado2:** É uma cadeia de caracteres com comprimento fixo de 1000 caracteres.
- **Dado3:** É uma cadeia de caracteres com comprimento fixo de 5000 caracteres.

2 Análise

2.1 ACESSO SEQUENCIAL INDEXADO

ASI ORDENADO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	84.7	0	6	27.5	50
1000	1045.1	0	10.1	26.1	500
10000	9630.5	0	6.1	293.2	5000
100000	85794.7	0	63.6	28463.2	50000
1000000	794188	0	536.4	292695	500000

Figura 1: ASI ordenado

No acesso sequencial indexado só é possível realizar pesquisas com arquivos ordenados. Apesar de sua fácil implementação, quando comparado aos demais tem um elevado tempo e número de comparações na pesquisa. Vale ressaltar que, mesmo com o arquivo ordenado, o tamanho da chave ainda pode influenciar no desempenho da pesquisa. Quanto maior a chave procurada, mais custosa se torna a busca, uma vez que são necessárias mais operações de comparação para localizar o registro desejado.

2.2 ÁRVORE BINÁRIA

ORDENAÇÃO ASCENDENTE

ÁRVORE BINÁRIA ORDENADO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	4124	14851	30	31	100
1000	344388	1498501	313	310	1000
10000	53777720	149985001	5902	3104	10000
100000	839811785501	14998505001	64350	31041	100000
1000000	1,31E+20	1499985001010	5309827	313000	1000000

Figura 2: Binária ordenado ascendente

ORDENAÇÃO DESCENDENTE

ÁRVORE BINÁRIA DESCENDENTE					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	4124	14851	30	50	100
1000	344388	1498501	313	500	1000
10000	53777720	149985001	5902	5000	10000
100000	839811785501	14998505001	64350	50000	100000
1000000	1,3115E+20	1499985001010	5309827	500000	1000000

Figura 3: Binária ordenado decrescendente

ORDENAÇÃO ALEATÓRIO

ÁRVORE BINÁRIA ALEATÓRIO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	2281	2002	14	11	100
1000	21384	30664	12	14	1000
10000	283110	493069	24	20	10000
100000	4009059	5915284	54	28	100000
1000000	48915750	7830793	34	22	1000000

Figura 4: Binária ordenado aleatório

Nas Árvores Binárias ordenadas crescente e decrescente, ambas possuem números de comparações de inserção e de pesquisa bem próximos. Pois ambas as estruturas possuem formato de galho, sendo assim, as duas representam o pior caso, por isso os altos valores encontrados. Quando temos uma Árvore Binária ordenada aleatoriamente, vemos que tivemos ganhos significativos, pois como os valores são melhor distribuídos na estrutura da árvore, a altura da árvore acaba sendo menor do que nos casos anteriores, facilitando a inserção e a pesquisa e diminuindo o número de comparações necessitarias para realizar estas operações.

2.3 ÁRVORE B

ORDENAÇÃO ASCENDENTE

ÁRVORE B ORDENADO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	453	1328	1	11	100
1000	5712	21568	2	15	1000
10000	55795	299016	2	19	10000
100000	565524	3832661	3	24	100000
1000000	5731731	46710223	4	31	1000000

Figura 5: B ordenado ascendente

ORDENAÇÃO DESCENDENTE

ÁRVORE B DESCENDENTE					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	471	1060	1	10	100
1000	4991	150001	2	15	1000
10000	51622	191816	3	20	10000
100000	524156	2331825	3	26	100000
1000000	5109401	27437021	5	31	1000000

Figura 6: B ordenado decrescendente

ORDENAÇÃO ALEATÓRIO

ÁRVORE B ALEATÓRIO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	413	1150	1	9	100
1000	4788	16948	1	14	1000
10000	53307	227020	1	8	10000
100000	608507	2837299	2	28	100000
1000000	7046044	33887435	2	34	1000000

Figura 7: B ordenado aleatório

Na Árvore B para realizar a pesquisa em ambos os casos, no aleatório e nos ordenados crescente e decrescente, tivemos valores bem próximos, pois a Árvore B sempre está balanceada, fazendo com que os números de comparações sejam baixos e próximos, independente dos valores de entrada. Pelo mesmo motivo, para a inserção teve valores próximos. Percebemos uma melhora na inserção quando comparemos com a Árvore Binária, pois o balanceamento feito pela estrutura da Árvore B diminui significativamente os piores casos e mantêm próximo quando tratamos de valores aleatórios.

2.4 ARVORE B*

ORDENAÇÃO ASCENDENTE

ÁRVORE B* ORDENADO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	357	965	1	8	100
1000	5000	17905	2	9	1000
10000	43811	262353	2	14	10000
100000	456009	3465998	2	16	100000
1000000	4516218	4304560	3	20	1000000

Figura 8: B* ordenado ascendente

ORDENAÇÃO DESCENDENTE

ÁRVORE B* DESCENDENTE					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	428	1015	1	6	100
1000	5241	14757	2	8	1000
10000	52778	191151	3	12	10000
100000	537848	2338670	4	14	100000
1000000	5359993	27624077	4	20	1000000

Figura 9: B* ordenado decrescendente

ORDENAÇÃO ALEATÓRIO

ÁRVORE B* ALEATÓRIO					
N. Registros	Tempo Ins	N. Comparações Inserção	Tempo Pesquisa	N. Comparações Pesquisa	N. Tranferencias Inserção
100	376	967	1	6	100
1000	4796	15253	1	8	1000
10000	50795	208965	1	13	10000
100000	618229	2649755	1	15	100000
1000000	6076000	32120976	2	18	1000000

Figura 10: B* ordenado aleatório

Na Árvore B*, tanto a inserção e a pesquisa apresentarem valores próximos. A B* herda as características da Árvore B, contudo, quando trassamos os comparativos entre elas, vemos que a B* é mais otimizada, tendo números de comparações um pouco menor.

3 Conclusão/Considerações Finais

Este trabalho prático proporcionou uma análise aprofundada da complexidade de desempenho de quatro métodos de pesquisa externa: acesso sequencial indexado, árvore binária de pesquisa adequada à memória externa, árvore B e árvore B*. Ao longo do estudo, pudemos observar as vantagens e desvantagens de cada abordagem, levando em consideração fatores como o tempo de acesso, a utilização de memória e a capacidade de gerenciar grandes volumes de dados. Em resumo, ao finalizar este trabalho, chegamos à conclusão de que a Árvore B* se destaca como a melhor opção entre os métodos estudados, apesar de possuir um código mais complexo. Essa abordagem apresenta vantagens significativas, como uma menor ocupação de memória e bons resultados nas métricas avaliadas.

Referências