IBILCE/UNESP
Dpto. de Ciências de Computação e Estatística

Professor: Dr. Lucas C. Ribas
lucas.ribas@unesp.br
Disciplina: Estrutura de Dados II

Lista de Exercícios - Arquivos, Índices, Hashing e Tries

1 Explique a diferença entre arquivo lógico e arquivo físico.

2 No que consiste a operação de posicionamento (seeking) em um arquivo? Qual a sua utilidade?

3 Explique o que é um cilindro, e a razão para a organização de arquivos em cilindros.

4 Por que os discos são considerados o gargalo de um sistema computacional? Explique como este problema pode ser minimizado.

5 Como são organizados fisicamente os discos? De que forma os discos armazenam os arquivos?

6 O que são "buffers" de E/S (ou I/O)? Quais os passos executados para que ler um byte do disco de forma que ele possa ser utilizado por um programa?

7 Explique o que é um *cluster* e o que é um *extent*.

8 O que é a fragmentação de um arquivo no disco? Quais os tipos de fragmentação do arquivo, porque elas ocorrem e quais seus efeitos?

9 Discuta as vantagens e desvantagens de organizar arquivos em blocos de tamanho definido pelo usuário, ao invés de setores de tamanho fixo.

10 As aplicações usualmente armazenam as informações em arquivos organizando as em campos e registros. Explique as diferentes maneiras pelas quais um campo pode ser armazenado em um arquivo para posterior recuperação.

11 Explique as diferentes estratégias que podem ser utilizadas para separar um registro de outro. Discuta as vantagens e desvantagens de cada uma delas.

12 Explique o que é fragmentação de campos e registros. Quando e por que ela o corre?

13 Se a separação entre registros e campos é feita por delimitadores, quais as restrições para a escolha desses delimitadores. Descreva uma situação que exemplifique sua resposta.

14 Como um registro é identificado para acesso aleatório? Qual operação permite localizar um registro no arquivo em C?

15 Explique como é possível melhorar o desempenho de um acesso sequencial a todo o conteúdo de um arquivo. Tal solução também garante um melhor desempenho de uma sequência arbitrária de acessos aleatórios? Discuta.

16 Considere um arquivo composto de registros de tamanho fixo de 64 bytes com 3 campos de tamanho fixo: RA (10 bytes), nome (40 bytes) e curso (14 bytes). Desconsidere os bytes usados como delimitadores. Qual será o comando para acessar o primeiro byte do campo nome do RRN=3? (considerando que o RRN do primeiro registro é 1).

17 Quais as vantagens e as desvantagens de utilizar arquivos organizados em registros de tamanho fixo?

18 O que é RRN? Como é possível fazer acessos aleatórios em arquivos a registros de tamanho variável?

19 Como se faz a alocação de registros de tamanho variável tendo em vista a existência de espaços vazios, existem 3 abordagens. Indique quais são e explique suas vantagens e desvantagens.

20 É vantajoso manter um arquivo separado para armazenar apenas as chaves e os byte offsets ou RRNs, dos registros no arquivo de dados? Como isto afeta a inserção de um novo registro?

21 Como um registro pode ser eliminado de um arquivo?

22 Explique a diferença entre fragmentação externa e interna?

23 Por que é interessante utilizar cabeçalhos nos arquivos?

24 Considere um arquivo contendo os 3 registros de tamanho fixo apresentados abaixo e mostre como fica o arquivo nas seguintes situações.
a) Quando o 2.º registro é excluído.
b) Quando o arquivo é compactado após a exclusão do 2o. registro.

```
Carla|Guimarães|Rua Riachuelo 123|Jardim América|033|720|.....
Nivaldo|Soares|Rua Moraes Barros 100|Centro|145|162|.............
Djavan Carlos| Moura|Av D. Pedro I 1234|Vila Independência|033|730|
```

25 Considere um arquivo contendo registros de tamanho variável apresentados abaixo e mostre como fica a lista Dispo e o arquivo após as exclusões/inclusões:
a) Excluir o segundo registro
b) Incluir `Gabriel|Goes|Av Dois 123|Vila Zac|033|124|` permitindo a fragmentação interna
c) Incluir `Gabriel|Goes|Av Dois 123|Vila Zac|033|124|` sem fragmentação interna
d) Excluir o primeiro registro

```
57 Carla|Guimarães|Rua Riachuelo 123|Jardim América|033|720|51 Nivaldo|Soares|
Rua Moraes Barros 100|Centro|145|162|65 Djavan Carlos|Moura|Av D.Pedro I
1234|Vila Independência|033|730|50 Aline|Miranda|Rua Treze de Maio 10|Cen-
tro|010|320|
```

26 Em princípio, não é possível fazer busca binária em um arquivo de dados com registros de tamanho variável. Por que a indexação do arquivo torna a busca binária possível?

Com um arquivo com registros de tamanho fixo é possível fazer busca binária. Isto significa que indexação não é necessária para arquivos de registros de tamanho fixo?

27 Qual o propósito em deixar um indicador de desatualizado (flag "out-of-date") no cabeçalho de um índice utilizado num ambiente de multiprogramação.? Dê um exemplo de situação em que manter esse indicador poderia ajudar.

28 Quando um registro é atualizado num arquivo, as chaves primárias e secundárias do índice podem ser alteradas ou não, dependendo do arquivo ter registros de tamanho fixo ou variável e dependendo do tipo de alteração que foi feita no registro de dados. Faça uma lista das situações diferentes que podem ocorrer devido a atualizações e explique como cada uma pode afetar os índices.

29 O que é uma lista invertida? Quando ela é ú til? Esquematize o conte ú do de um índice secundário organizado como lista invertida para um arquivo de dados hipotético.

30 Suponha que você tenha um arquivo de dados de CD s, como descrito no exemplo em aula, muito grande, com um índice pela chave primária e índices pelas chaves secundárias organizados por compositor e gravadora. Suponha que uma lista invertida é usada para as chaves secundárias. Explique, passo a passo, como um programa poderia responder às seguintes solicitações:
(a) Liste todas as gravações de Chico Buarque;
(b) Liste todas as gravações de Milton Nascimento e gravadora Emi-Odeon.

31 Por que, para a eliminação de um registro do arquivo de dados, é possível eliminar o registro apenas do índice primário, e não do secundário?

32 Explique o conceito de late binding e early binding. Quando é aconselhável o *early binding* e por que?

33 Implemente uma tabela hash externa para armazenar registros de alunos. Cada registro deve conter o número de matrícula (inteiro) e o nome do aluno (string). Utilize uma função de hash que calcula o índice da tabela como o número de matrícula modulado pelo tamanho da tabela (`matrícula % tamanho_da_tabela`). Se ocorrer uma colisão (ou seja, dois alunos com o mesmo índice), armazene os registros em uma lista encadeada (ou um arquivo separado).

Implemente as seguintes operações:

1.) Inserção: Insira um novo registro de aluno na tabela hash.

2.) Busca: Dado o número de matrícula, retorne o nome do aluno correspondente.

3.) Remoção: Remova um aluno da tabela hash dado o número de matrícula.

34 Considere um sistema de armazenamento que usa hashing externo para organizar os registros de uma base de dados de produtos em um arquivo. Cada registro de produto contém um código de produto (inteiro) e uma descrição (string). O arquivo é dividido em blocos de 512 bytes, e cada bloco pode armazenar até 4 registros.

1.) Função de Hash: Defina uma função de hash que distribua uniformemente os registros nos blocos usando o código do produto como chave.

2.) Operações: Implemente as seguintes operações:

(a) **Inserir Produto**: Insira um novo produto no arquivo. Se o bloco calculado estiver cheio, mova o registro para um bloco de overflow.

(b) **Buscar Produto**: Dado o código do produto, retorne a descrição correspondente, considerando que ele pode estar em um bloco de overflow.

(c) **Remover Produto**: Remova um produto do arquivo, atualizando os blocos de overflow se necessário.

35 Dada uma árvore Trie inicialmente vazia, implemente uma função para inserir as seguintes palavras: `casa`, `carro`, `carne`, `caso`, `cartão`. Desenhe a estrutura final da Trie após todas as inserções.

36 Implemente uma função que, dada uma árvore Trie, verifique se uma determinada palavra está presente na Trie. Teste sua função com as palavras `carro`, `carta`, `caso`, `castelo` utilizando a Trie do exercício anterior. Indique se as palavras foram encontradas ou não.

37 Implemente uma função para remover uma palavra de uma Trie. Utilize a Trie resultante do primeiro exercício e remova a palavra `carro`. Desenhe a estrutura da Trie após a remoção e explique como a árvore foi alterada.

38 Considere uma árvore Trie para armazenar sequências de bits (0 e 1). Implemente uma função para inserir as seguintes sequências binárias na Trie: `101`, `1001`, `1010`, `110`, `111`, `11100`. Desenhe a estrutura final da Trie após cada inserção, destacando como os nós são criados para representar cada bit das sequências.

39 Dada uma Trie construída para armazenar sequências de bits (0 e 1), implemente uma função que verifique se uma sequência binária é prefixo de alguma sequência presente na Trie. Utilize a Trie resultante do exercício anterior e verifique se as sequências `10`, `11`, `100`, `1011` são prefixos de alguma sequência armazenada na Trie. Indique o resultado para cada sequência.

40 Transforme a árvore trie do exercício anterior em um diretório.

41 Represente a estrutura de hash extensível para as seguintes chaves: 2; 6; 1; 5; 3; 7; 9; 4 e 10 considerando que a função hash retorna como índice do diretório os dois bits menos significativos do número em ordem inversa Ex. h(2) -> 0010 => 01

42 Considere um máximo de 3 elementos por bucket, e que a função hash gera 4 bits para uma chave. Simule a inserção de chaves que geram os seguintes endereços: 0000, 1000, 1001, 1010, 1100, 0001, 0100, 1111, 1011

43 Considere um máximo de 2 elementos por bucket, e que a função hash pega o 3 bits menos significante do resultado de k mod 8 (profundidade máxima igual a 3). Exemplo: k=3 -> 3 mod 8 = 3 -> 011. Simule a inserção de chaves que geram os seguintes endereços: 1,2,3,4,5,6,7,8 e 9.