SCC-203 ALGORITMOS E ESTRUTURAS DE DADOS II

Nome: _____ Nro. USP_____

Prova 2(21/5/2012) – Gabarito Parcial
Obs.: Esta prova vale 11 pontos

1) (1.5)
a) Registros de tamanho fixo com campos de tamanhos variáveis podem causar fragmentação interna. Explique por que.

Porque pode haver sobra de espaço se a soma dos campos não alcançar o tamanho estipulado do registro.

b) A fim de evitar qualquer fragmentação (espaço não ocupado num arquivo), cite uma forma de organizar campos e registros.

Por exemplo, tanto campos como registros seriam de tamanhos variáveis. Para delimitar campos no registro, e registros no arquivo, delimitadores poderiam ser usados (um para cada tipo – campo ou registro).

c) Considerando sua resposta em b), como fazer quando:

i) um novo registro deve ser inserido no arquivo de dados (considere que o arquivo não é ordenado)

Basta inseri-lo no final do arquivo *(entry-sequenced)*

ii) um registro é eliminado do arquivo

Neste caso, pode haver fragmentação. Para que esse espaço possa ser reaproveitado em novas inserções, é necessário um esquema de gerenciamento do espaço disponível.

2) (1.0) Setores são as regiões endereçáveis de um disco. Um setor típico tem 512 bytes. *Clusters* são conjuntos de setores contíguos no disco. Arquivos são gravados em *clusters*.
Comente sobre a questão do tamanho ideal para um *cluster*.
Se for necessário gravar um arquivo maior do que um cluster, o mesmo pode ficar particionado em diferentes (e não contíguos) clusters no disco, necessitando mais de um seek para ler todo o arquivo. No entanto, se o tamanho do arquivo for muito menor do que o do cluster, causará fragmentação externa no disco.

3) (2.0) Manter arquivos dinâmicos grandes, que não cabem na RAM, ordenados é uma tarefa muito cara, pois, além de ordenar pela primeira vez, é necessário garantir que futuras inserções e eliminações mantenham a ordem dos registros, e não acarretem operações muito caras (deslocamentos, fragmentação, etc.).

a) Cite um método de ordenação de grandes arquivos. Explique como ele funciona e fale de sua complexidade.

K-way Mergesort. O arquivo é dividido em k conjuntos de chaves, de modo que cada conjunto caiba na RAM e é ordenado por um bom método de ordenação interna. Esses conjuntos ordenados são chamados corridas ou runs. As corridas são gravadas em disco. Num segundo passo, faz-se um merge de ordem k com essas k corridas, obtendo o arquivo ordenado. Conforme o merging é realizado, blocos de registros são gravados sequencialmente no disco.

A complexidade do K-way Mergesort é dada pelo número de seeks que exige. Esse número é O(K2). Como K é função de N (K = N/x), então pode-se dizer que é O(N2). Isso é muito ruim, mas com algumas melhorias, o mergesort externo ainda é a base dos métodos conhecidos de ordenação externa.

b) Como evitar que futuras inserções/eliminações não provoquem deslocamentos ou fragmentação no arquivo de dados?

Construindo um arquivo índice, que será menor do que o de dados, eventualmente cabendo totalmente em RAM. Enquanto que o arquivo de dados pode ser *entry-sequenced*, o índice é ordenado pelas chaves. Nova inserção no arquivo de dados é feita no final do arquivo, provocando inserção ordenada no índice, que é muito menor. Eliminação no arquivo de dados continua a provocar fragmentação no mesmo. Estratégias para combater incluem: deixar os espaços vagos e reorganizar de tempos em tempos o arquivo; gerenciar os espaços vagos de modo a utiliza-los nas próximas inserções, por meio de uma lista encadeada. Várias estratégias para escolher o próximo espaço a ser usado: first-fit, best-fit; worst-fit.

4) (6.5) Considere o problema de armazenar em um arquivo registros referentes a sites da internet. Os dados a armazenar são:

- o título do site,
- a URL,
- a palavra mais frequente no site que não esteja no título,
- a frequência dessa palavra,
- e a data (formato MMDDAA) do armazenamento do site no arquivo.

Exemplo:
WIKIPEDIA, EN.WIKIPEDIA.ORG, ENCYCLOPEDIA, 9, 05252011.

Considerando que a prioridade é que o arquivo seja armazenado da maneira **mais compacta possível**:

(a) (1.0) Escreva o projeto de uma organização de arquivo para guardar os dados:

i) estrutura dos campos e registros

_____
_____
_____
_____
_____

ii) como serão feitas as operações de inserção e busca de registros.

_____
_____
_____
_____
_____

(b) (0.5) O seu arquivo estará sujeito à fragmentação interna? E à externa? Justifique.

_____
_____
_____
_____
_____

(c) (0.5) Escreva uma representação textual de como estariam armazenados os seguintes registros no arquivo, segundo a organização que você descreveu:

(i) FREE YR, YEP.IT, IT, 12, 122210.
(ii) TINYURL, TINYURL.COM, URL, 9, 072511.

(d) (1.5) Escreva uma representação binária do registro (i) acima, se o arquivo fosse codificado usando o método de Huffman. Considere apenas a frequência dos símbolos que aparecem no próprio registro (i). Desenhe a árvore de Huffman correspondente e dê os códigos de cada símbolo.

OBS: na construção da árvores, o elemento de menor frequência vai sempre à esquerda; associe "0" à esquerda e "1" à direita.

Árvore de Huffman:

Símbolo	Frequência	Código
F		
R		
E		
Y		
P		
I		
T		
1		
2		
0		

Representação codificada do registro (i):

(e) (0.5) Baseado nos registros exemplificados, seria interessante aplicar um método de compressão? Que método você sugere? Justifique.

_____
_____
_____
_____
_____

(f) Considere o uso de índices para o arquivo de sites.

i) (0.5) Qual seria o índice primário? Justifique. Descreva a estrutura desse índice (registro, campos).

_____
_____
_____
_____
_____

ii) (0.5) Descreva em palavras como funcionaria uma operação de busca por um site usando o índice primário.

_____
_____
_____
_____
_____

iii) (0.5) Descreva em palavras o que aconteceria numa operação de inserção de um novo site.

_____
_____
_____
_____
_____

iv) (0.5) Como seria um possível índice secundário baseado no título do site? Descreva a estrutura desse índice

_____
_____
_____
_____
_____

v) (0.5) Como funcionaria uma operação de busca por um site utilizando o índice secundário?

_____
_____
_____
_____
_____