SCC0503 — Algoritmos e Estruturas de Dados II (ICMC/USP)

Lista de Exercícios 6: Hashing Externo

Professor: Moacir Pereira Ponti Jr. *PAE: Paulo Henrique Ribeiro Gabriel*

1. (*POSCOM 2009*) Um arquivo é organizado logicamente como uma sequência de registros. Esses registros são mapeados em blocos de discos. Com base no conhecimento sobre organização de arquivos, considere as afirmativas a seguir.

   I. As organizações de arquivos sequenciais exigem uma estrutura de índice para localizar os dados. De outra forma, organizações de arquivos baseadas em *hashing* permitem-nos encontrar o endereço de um item de dado diretamente por meio do cálculo de uma função sobre o valor da chave de procura do registro desejado.

   II. As operações em arquivos são geralmente divididas em operações de recuperação e operações de atualização: as primeiras não alteram nenhum valor no arquivo, apenas localizam certos registros, de forma que seus valores de campo possam ser examinados e processados; as últimas mudam o arquivo por meio da inclusão ou da exclusão de registros ou pela modificação de valores dos campos.

   III. Registros de tamanho fixo permitem campos repetidos, tamanhos variáveis para um ou mais campos e ainda o armazenamento de múltiplos tipos de registro.

   IV. Nos arquivos desordenados (também conhecidos como arquivos pilha), os registros são posicionados no arquivo segundo a ordem pela qual foram incluídos, ou seja, novos registros são acrescentados no final do arquivo. Incluir um novo registro é muito eficiente, entretanto a pesquisa por um registro, usando qualquer condição, envolve uma pesquisa sequencial bloco a bloco do arquivo, procedimento dispendioso.

   Assinale a alternativa correta.

   (a) Somente as afirmativas I e II são corretas.

   (b) Somente as afirmativas I e III são corretas.

   (c) Somente as afirmativas III e IV são corretas.

   (d) Somente as afirmativas I, II e IV são corretas.

   (e) Somente as afirmativas II, III e IV são corretas.

2. Se usássemos *hash* em um arquivo em que algumas vezes o item não fosse encontrado, seria interessante manter estatísticas sobre o comprimento médio das buscas mal sucedidas. Se uma alta porcentagem das buscas for mal sucedida, como você imagina que isto afetará o desempenho geral se o *overflow* for tratado:

   (a) Por overflow progressivo.

   (b) Por overflow progressivo encadeado.

   (c) Usando uma área de overflow em separado.

3. Explique como funciona o hashing extensível.

4. Qual a diferença entre o espalhamento extensível e o espalhamento convencional? Porque o segundo não é adequado para representar índices armazenados em disco?

5. Qual a vantagem de aplicar uma função de espalhamento sobre a chave para definir o seu endereço (cesto), ao invés de amostrar diretamente o valor da chave, como feito nas *tries*?

6. Qual a vantagem de usar a representação em diretório no hashing extensível, ao invés de usar a representação por árvore da *trie*?

7. Considere a seguinte *trie* de ordem (raio) 2, com ponteiros para buckets com capacidade para abrigar 100 chaves (ou registros):

   (a) Desenhe a trie estendida e o diretório de endereços hash correspondente.

   (b) Considerando que os buckets A, B e C contém, respectivamente, 100, 50 e 03 registros, dê a configuração do diretório, e a condição de cada bucket após a inserção de uma nova chave cujo valor da função hash é 00.

   (c) Ainda na configuração inicial, considere agora que todas as chaves de B são eliminadas. O que acontece com o diretório?

8. Escreva uma função em C chamada `search(Tabela,Chave)` que busque uma chave em uma tabela hash. A função aceita uma chave inteira e uma tabela declarada por

```
struct record {
    KeyType k;
    RecType r;
    int flag;
} array[TableSize];
```

   sendo que `tabela[i].k` e `tabela[i].r` são a i-ésima chave e o i-ésimo registro, respectivamente. `tabela[i].flag` é igual a FALSE se a i-ésima posição da tabela estiver vazia e TRUE se estiver ocupada. A rotina retorna um inteiro entre 0 e `TableSize-1` se um registro com a chave `Chave` estiver presente na tabela. Se este registro não existir a função retorna -1. Assuma a existência de uma rotina de hash, `h(Chave)`, e uma rotina de rehash `rh(Índice)` que também retorna valores entre 0 e `TableSize-1`.

9. Escreva uma função em C `insert(table,key,rec)` para buscar e inserir chaves numa tabela hash como a do exercício anterior.

10. Desenvolva um mecanismo para detectar quando todas as posições possíveis para reespalhamento foram acessadas. Incorpore este método nas rotinas `search` e `insert` dos exercícios anteriores.