Universidade Federal da Bahia
Graduação em Ciência da Computação
MATA54 - Estruturas de Dados e Algoritmos II
Primeiro Trabalho Prático
Prof. Flávio Assis
Semestre 2017.2 - 28 de outubro de 2017

Resolução de *Hashing* com Encadeamento e *Hashing* Duplo

1 Descrição Geral do Trabalho

Neste trabalho o aluno implementará dois métodos de resolução de colisões em arquivos estruturados como uma tabela *hash*: resolução de colisão através de encadeamento e *hashing* duplo.

Cada registro a ser armazenado no arquivo conterá, além dos dados de controle necessários para gerenciar o método de resolução de colisão: uma *chave*, de valor inteiro não negativo; uma cadeia de, no máximo, 20 caracteres, que irá armazenar um *nome*; e um outro valor inteiro não negativo, que irá armazenar uma *idade*. O programa deverá conter uma constante definida com o seguinte identificador:

TAMANHO_ARQUIVO: indica o número máximo de registros do arquivo.

O valor inicial da constante TAMANHO_ARQUIVO deve ser 11. O programa deve ser feito de forma que o valor desta constante possa ser modificado.

2 Sobre o Método de Resolução de Colisão com Encadeamento

A implementação deste método deve manter um apontador em cada registro para encadear os elementos. Não deve haver junção de cadeias, ou seja, uma cadeia que começa em uma posição i do arquivo deve somente conter elementos cujo endereço original é i. Caso, ao inserir um elemento em uma posição i do arquivo, já haja um registro nesta posição com endereço original é $j \neq i$, este elemento deve ser armazenado em outra posição, para permitir o armazenamento do novo registro na posição correspondente a seu endereço original.

Sempre que uma posição vazia for necessária (por exemplo, para inserir um novo elemento em uma cadeia não vazia ou realocar um registro), o programa deve usar a última posição vazia do arquivo.

Ao inserir um registro em uma cadeia não vazia, este elemento deve ser inserido como *último* elemento da lista.

Remoção de registro deve ser feita de forma análoga a remoção em lista encadeada em memória principal. O elemento anterior (se houver), deve apontar para o próximo elemento da lista (se houver), sem realocar registros de posições do arquivo. Caso, no entanto, o registro a ser removido seja o primeiro de uma cadeia que contenha mais elementos, o segundo elemento deve passar a ocupar a posição do primeiro registro (e a posição originalmente ocupada por esse registro deve passar a ficar vazia).

A função de *hashing* a ser utilizada, denominada h, é:

$$h(chave) = chave \ mod \ \text{TAMANHO\_ARQUIVO}$$

3 Sobre *Hashing* Duplo

A implementação deste método deve seguir a descrição em Cormen et al. As funções de *hashing* a serem utilizadas são:

$$h_1(chave) = chave \ mod \text{ TAMANHO\_ARQUIVO}$$
$$h_2(chave) = \lfloor chave/m \rfloor \ mod \text{ TAMANHO\_ARQUIVO}$$

Se $h_2(chave) = 0$, deve-se considerar $h_2(chave) = 1$.

4 Observações importantes

O programa deve manter as atualizações em arquivo. A correção levará em consideração que o estado dos dados é persistente. Com isto, um teste pode ser feito, por exemplo, inserindo-se um registro, terminando a execução do programa e fazendo uma consulta ao registro em nova invocação do programa. Neste caso o registro deve ainda estar no arquivo.

Adicionalmente, lembre-se de que é assumido que a memória principal é insuficiente para armazenar todos os dados. Portanto, por exemplo, uma implementação que mantenha a estrutura do arquivo em memória (em um vetor, por exemplo) e o salva por completo no arquivo será considerada inaceitável.

O arquivo deve ser armazenado em formato binário.

Este trabalho envolve a implementação de dois métodos de resolução de colisão. Obviamente em um dado arquivo será utilizado apenas um dos métodos.

5 Formato de Entrada e Saída

A entrada conterá inicialmente uma linha contendo ou a letra 'l' ou a letra 'd'. Caso essa linha contenha a letra 'l', o método de *hashing* com encadeamento deve ser usado. Caso contenha a letra 'd', o método de *hashing* duplo deve ser usado.

A seguir, a entrada conterá uma sequência de operações sobre o arquivo. As operações e seus formatos estão descritos abaixo:

1. **insere registro**: esta operação conterá quatro linhas. A primeira linha conterá a letra 'i'. A segunda conterá um valor de chave. A terceira conterá uma sequência de até 20 caracteres, que corresponderá ao campo *nome*. A quarta linha conterá um valor de idade. A sequência de caracteres da terceira linha conterá qualquer sequência de letras (minúsculas, sem acento, nem cedilha) e espaços, sendo que o primeiro e último caracteres não serão espaço.

   Esta operação verifica se já há registro no arquivo com o valor de chave indicado. Se sim, esta operação gera na saída, em uma mesma linha, a sequência de caracteres *'chave ja existente:'*, seguida de um espaço, seguido do valor da chave. Se a chave não existir, a operação insere o registro no arquivo, sem gerar saída.

2. **consulta registro**: esta operação conterá duas linhas. A primeira linha conterá a letra 'c'. A segunda conterá um valor de chave.

   Se houver registro no arquivo com o valor de chave indicado, esta operação gera na saída a sequência de caracteres *'chave:'*, seguida de um espaço, seguido do valor da chave. Em seguida, na próxima linha escreve o valor do nome associado ao registro, e, na linha seguinte, o valor da idade associada ao registro. Se não houver registro no arquivo com o valor de chave indicado, esta operação gera na saída a sequência de caracteres *'chave nao encontrada:'*, seguida de um espaço, seguido do valor da chave.

3. **remove registro**: esta operação conterá duas linhas. A primeira linha conterá a letra 'r'. A segunda conterá um valor de chave.

   Se houver registro no arquivo com o valor de chave indicado, esta operação causará a remoção do registro e não gerará saída. Se não houver registro no arquivo com o valor de chave indicado, esta operação gera na saída a sequência de caracteres *'chave nao encontrada:'*, seguida de um espaço, seguido do valor da chave.

4. **imprime arquivo**: esta operação conterá apenas uma linha, contendo a letra 'p'. Esta operação imprimirá o arquivo. Os registros serão apresentados, um em cada linha, em ordem, do registro de índice 0 até o registro de índice $TAMANHO\_ARQUIVO - 1$. Cada linha terá um formato de acordo com o método de resolução de colisão utilizado (indicado pelo conteúdo da primeira linha da entrada), como indicado a seguir.

   Se o método de resolução for encadeamento, cada linha conterá: o índice do registro, seguido de dois pontos (':'), seguido de um espaço. Se a posição do arquivo estiver vazia, a sequência de caracteres *'vazio'* deverá ser apresentada. Se a posição estiver ocupada, deve ser apresentada a chave do registro, seguida de um espaço, seguida da sequência de caracteres (nome), seguida de um espaço, seguido da idade, seguida de um espaço, seguido do valor do campo apontador. Se o campo apontador tiver valor nulo, deve ser impressa a sequência de caracteres *'nulo'*.

   Se o método de resolução for *hashing duplo*, cada linha conterá: o índice do registro, seguido de dois pontos (':'), seguido de um espaço. Se a posição do arquivo estiver vazia, a sequência de caracteres *'vazio'* deverá ser apresentada. Se a posição estiver ocupada, deve ser apresentada a chave do registro, seguida de um espaço, seguida da sequência de caracteres (nome), seguida de um espaço, seguido da idade.

5. **média de acessos a registros do arquivo**: esta operação conterá apenas uma linha, contendo a letra 'm'. Esta operação apresenta, em uma linha, apenas o valor da média do número de acessos a registros do arquivo, considerando-se uma consulta a cada um dos registros armazenados no arquivo. Esta média deve ser apresentada sempre como um valor real, com uma única casa decimal.

6. **término da sequência de comandos:** a sequência de comandos será terminada por uma linha com a letra 'e'.

Importante: o programa não deve gerar nenhum caractere a mais na saída, além dos indicados acima. Em particular, o programa não deve conter menus.

Não deve haver espaço entre linhas na saída. A saída deve apresentar os caracteres em letras minúsculas.

6 Observações

Trabalho individual.

Data de entrega: 29/11/2017

Linguagens de programação permitidas: C, C++, Java ou Python.

Observação Importante: Para as linguagens C, C++ e Java, somente trabalhos feitos utilizando-se os seguintes compiladores serão aceitos:

- C: gcc ou djgpp
- C++: g++ ou djgpp
- Java: compilador java recente, disponibilizado pela Oracle.

Não serão compilados trabalhos em outros compiladores! Erros ocasionados por uso de diferentes compiladores serão considerados erros do trabalho!

O aluno deverá armazenar submeter seu trabalho através do *moodle*.