Universidade Federal da Bahia
MATA54 - Estruturas de Dados e Algoritmos II
Segundo Trabalho Prático
Prof. Flávio Assis
Semestre 2022.1 - 13 de abril de 2022

Hashing Perfeito

1 Descrição Geral do Trabalho

Neste trabalho deverá ser implementada uma estrutura de armazenamento de registros baseada em *hashing perfeito*, como descrito no livro de Cormen et al., *Introduction to Algorithms*.

Os campos de cada registro a ser armazenado no arquivo são: uma *chave*, de valor inteiro não negativo; uma cadeia de caracteres, representando um *nome*; e um outro valor inteiro não negativo, representando uma *idade*.

O programa irá receber uma sequência de registros e criará uma estrutura em arquivos para acesso a esses dados. O valor primo a ser usado para se criar o conjunto de funções de *hashing* universal deve ser o menor primo maior do que todas as chaves dos registros. Não haverá chave maior do que 100.

2 Formato de Entrada e Saída

A entrada constará de uma sequência de operações. As operações e seus formatos estão descritos abaixo:

1. **insere registro**: esta operação conterá inicialmente a letra 'i' em uma linha e um número inteiro positivo na segunda linha. Esse valor inteiro indica o número de registros a serem usados para a criação do esquema de *hashing*. Em seguida, haverá os dados dos registros.

   Os dados de cada registro ocuparão três linhas. A primeira linha conterá um valor de chave. A segunda conterá uma sequência de até 20 caracteres, que corresponderá ao campo *nome*. A terceira conterá um valor de idade. A sequência de caracteres da segunda linha conterá qualquer sequência de letras (minúsculas, sem acento, nem cedilha) e espaços, sendo que o primeiro e último caracteres não serão espaço.

   Como se trata de um esquema de *hashing* perfeito, essa operação de inserção somente ocorrerá uma vez na entrada e será sempre a primeira operação. Por se tratar de chaves primárias, não haverá chaves repetidas. As chaves poderão ocorrer em qualquer ordem. No entanto, a primeira chave do primeiro registro será a de maior valor entre todas as chaves.

   Essa operação causará a geração da estrutura de *hashing*. Ao finalizar a operação, o programa deverá gerar na saída a sequência de caracteres *'estrutura de hashing perfeito criada'*.

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 na estrutura de *hashing* com o valor de chave indicado, esta operação gera na saída a sequência de caracteres *'chave:'*, seguida de um espaço, seguido no 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 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. **imprime primeiro nível da estrutura de *hashing***: esta operação conterá apenas uma linha, contendo a letra 'p'.

   O formato da saída desta operação será da seguinte forma. Inicialmente será gerada a seguinte sequência: 'hashing perfeito: primeiro nivel'. Na próxima linha, deverá ser gerada a sequência 'tamanho da tabela:', seguida de espaço, seguida do tamanho (número de posições) da estrutura do primeiro nível. Na próxima linha, deverá ser gerada a sequência 'parametro a:', seguida de espaço, seguido do valor do parâmetro *'a'* da função genérica utilizada para se criar o conjunto de funções de *hashing* universal (de acordo com o livro de Cormen et al.), utilizado para a tabela de primeiro nível. Em seguida, na próxima linha, deverá ser gerada a sequência 'parametro b:', seguida de espaço, seguido do valor do parâmetro *'b'* da função genérica utilizada para se criar o conjunto de funções de *hashing* universal (de acordo com o livro de Cormen et al.), utilizado para a tabela de primeiro nível. Na próxima linha, deve aparecer a sequência 'numero primo:', seguida de um espaço, seguido do número primo utilizado.

   A partir da próxima linha será apresentada a tabela em si do primeiro nível. Somente devem aparecer os índices da tabela em que tiver havido chave associada a eles pela função de *hashing* do primeiro nível. Devem ser apresentadas as entradas em ordem crescente de índice. Para cada índice, deve-se gerar em uma única linha: o número do índice, seguido de dois pontos (':'), seguido de um espaço, seguido da sequência de chaves associadas àquele índice, separadas por um espaço.

4. **imprime segundo nível da estrutura de *hashing***: esta operação conterá duas linhas. A primeira linha conterá a letra 's'. A segunda conterá um número natural, indicando um índice (válido) da tabela de primeiro nível da estrutura de *hashing*. A operação gerará na saída uma representação da estrutura de segundo nível associada ao índice informado.

   O formato da saída desta operação será semelhante (mas não igual) ao da operação anterior. Inicialmente será gerada a seguinte sequência: 'hashing perfeiro: segundo nivel - indice:', seguido de um espaço, seguido do número do índice. Na próxima linha, deverá ser gerada a sequência 'tamanho da tabela:', seguida de espaço, seguida do tamanho da tabela. Na próxima linha, deverá ser gerada a sequência 'parametro a:', seguida de espaço, seguido do valor do parâmetro *'a'* da função genérica utilizada para se criar o conjunto de funções de *hashing* universal (de acordo com livro de Cormen et al.), utilizado para a tabela de segundo nível. Em seguida, na próxima linha, deverá ser gerada a sequência 'parametro b:', seguida de espaço, seguido do valor do parâmetro *'b'* da função genérica utilizada para se criar o conjunto de funções de *hashing* universal (de acordo com livro de Cormen et al.), utilizado para a tabela de segundo nível. Na próxima linha, deve aparecer a sequência 'numero primo:', seguida de um espaço, seguido do número primo utilizado.

   A partir da próxima linha será apresentada a tabela em si. Somente devem aparecer os índices da tabela em que tiver havido chave associada a ela pela função de *hashing* do nível. Devem ser apresentadas as entradas em ordem crescente de índice. Para cada índice, deve-se gerar em uma linha: o número do índice, seguido de dois pontos (':'), seguido de um espaço, seguido da chave associada àquele índice.

5. **imprime estrutura global:** esta operação conterá apenas uma linha, contendo a letra 'h'.

   A saída dessa operação será da seguinte forma. inicialmente será apresentado o primeiro nível, como descrito para a operação *imprime primeiro nível da estrutura de hashing*. Em seguida, deverão ser apresentadas as estruturas de cada nível em que houver chaves, seguindo a descrição da operação *imprime segundo nível da estrutura de hashing*.

6. **cardinalidade do conjunto de funções de *hashing* universal:** esta operação conterá apenas uma linha, contendo a letra 'n'.

   Essa operação deve indicar o número total de funções que existem no conjunto de funções de *hashing* universal utilizado.

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

3 Observações

- 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.
- Lembre-se de que é assumido que a memória principal é insuficiente para armazenar todos os dados. Portanto, por exemplo, uma implementação que mantém a estrutura do arquivo em memória principal e a salva por completo no arquivo será considerada inaceitável.
- Os arquivos devem ser armazenados em formato binário.
- O programa não deve gerar nenhum caractere a mais na saída, além dos indicados acima. Em particular, o programa não deve conter menus.
- Não pode haver espaço entre linhas na saída. A saída deve apresentar os caracteres em letras minúsculas.
- Trabalho individual ou em dupla.
- Além do código fonte do programa, deve ser entregue também um relatório sucinto, indicando como a estrutura de *hashing* foi armazenada em memória secundária.

  Importante: Se não houver entrega desse relatório, a nota do trabalho pode ser **bastante** afetada, uma vez que o relatório é necessário para se entender como a implementação foi feita.
- Data de entrega: **01/05/2022**
- Linguagens de programação permitidas: C, C++, Java ou Python.
  - **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 do JDK (mais recente)

  No caso de Python, deve ser indicada a versão utilizada.

  Não serão compilados trabalhos em outros compiladores! Erros ocasionados por uso de diferentes compiladores serão considerados erros do trabalho!
- O trabalho deve ser entregue através do *moodle*.