Dredd - Juiz Online

Minutos Restantes: 32568

Usuário: Lucas Antonio Lopes Neves

Notas:
Q1: ?
Q2: ?
Q3: ?
Total: 0

Exercícios sobre Hashes

Prova Aberta Até: 24/10/2018 11:32:41

Número Máximo de Tentativas: 6

Atenuação da Nota por Tentativa: 2%

Instruções para a prova: Exercícios sobre hashes, heaps e torneios. Pode ser acessada de casa.

Questão 1: Tabela Hash (encadeamento) - Código a completar

O código fornecido implementa uma tabela hash com tratamento de colisão por encadeamento, sem os métodos de inserção, alteração e remoção. Sua tarefa é corrigir e completar o código fornecido, fazendo-o funcionar adequadamente. O código será verificado com um dado número de operações que já estão na função principal.

Entradas:

1. Quantidade de valores a serem inseridos
2. chaves e valores a serem inseridos, um par por linha (strings sem espaço)
3. Quantidade de valores a serem alterados
4. chaves e valores a serem alterados, um par por linha (strings sem espaço)
5. Quantidade de valores a serem removidos
6. chaves (strings sem espaço) a serem removidas, uma por linha
7. Quantidade de valores a serem consultados
8. chaves (strings sem espaço) a serem consultadas, uma por linha

Saída:

1. As inserções, alterações, remoções e consultas produzem saídas específicas, determinadas em outros lugares.
2. A saída do método de depuração `percorre`.

Exemplo de Entrada:

```
5
joukim Joaquim
jose Silva
selvagem Bicicleta
agosto Cachorro
pokemon Go
2
agosto Louco
pokemon Niantic
1
jose
3
joukim
pokemon
jose
```

Exemplo de Saída:

```
Joaquim
Niantic
```

```
NAO ENCONTRADO!

0:[selvagem/Bicicleta]->NULL  1:NULL  2:NULL  3:NULL  4:NULL
5:[agosto/Louco]->[pokemon/Niantic]->NULL  6:NULL  7:NULL  8:NULL
9:[joukim/Joaquim]->NULL
```

Exemplo de Entrada:

```
7
joukim Joaquim
selvagem Bicicleta
agosto Cachorro
pokemon Go
farofa Cartoon
pimentao Verde
vou Passar
2
agosto Louco
pokimon Niantic
1
agosto
3
joukim
pokemon
agosto
```

Exemplo de Saída:

```
ERRO NA ALTERACAO!
Joaquim
Go
NAO ENCONTRADO!

0:[selvagem/Bicicleta]->NULL  1:NULL  2:NULL  3:NULL
4:[vou/Passar]->NULL  5:[pokemon/Go]->[pimentao/Verde]->NULL  6:NULL
7:[farofa/Cartoon]->NULL  8:NULL  9:[joukim/Joaquim]->NULL
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Choose File No file chosen Enviar Resposta

Questão 2: Hash com endereçamento aberto

Implemente uma tabela hash com tratamento de colisão por endereçamento aberto. Use o código fornecido. A implementação deve seguir a seguinte estratégia:

- A tabela hash é criada para uma determinada capacidade. A capacidade não pode ser alterada e tentativas de inserção além da capacidade falham.
- Os espaços de armazenamento podem estar em um de três estados; 1) VAZIO, 2) REMOVIDO, 3) COM VALOR. Todos os espaços começam VAZIOS e depois de alterados nunca mais voltam a ser VAZIOS.
- Em caso de colisão, usa-se a posição consecutiva e assim sucessivamente. A posição que sucede a última é a primeira.
- A busca de itens na tabela hash termina ao encontrar um espaço VAZIO.
- A tabela hash armazena pares chave/valor, onde a chave é do tipo texto e o valor é de tipo indeterminado. A chave vazia (string vazia) é especial e não pode ser associada a valores. Não é permitido criar outros valores especiais

para chave, esperando que ninguém nunca use um determinado valor como chave.

Com relação ao tratamento de erros, o código deve ser a prova de programadores descuidados. Mandar remover de estrutura vazia, por exemplo, não pode passar despercebido, mas a decisão sobre o que fazer não pode ser tomada na classe Hash. A função main é a responsável por determinar o que será feito. Em todos os casos de erro, a função main vai simplesmente escrever "ERRO" na saída padrão. A implementação dada usa manipulação de exceções, mas você pode mudar para códigos de erros caso não tenha prática com manipulação de exceções.

A implementação dada usa ponteiros para determinar os 3 estados de uma posição do armazenamento o ponteiro especial REMOVIDO já está implementado. Atenção na hora de desalocar memória. É permitido alterar a forma como os 3 estados são implementados, mas você terá que alterar coisas que já estão prontas.

Existe um método pronto para escrever a posição, chave e valor de todos os valores da tabela hash. Esse método mostra posições VAZIAS, REMOVIDAS e ocupadas com sintaxes distintas. Se você quiser mudar a forma como os 3 estados são implementados deveria mudar este método sem alterar a saída que ele produz. A função hash que mapeia chaves (texto) em posições na tabela também está pronta e não pode ser alterada.

A recuperação do valor de uma chave deve ser eficiente: não basta encontrar um valor, é necessário atender a estratégia acima.

Entradas:

A parte de interface do programa já está implementada. Inicialmente é lido um valor para a capacidade da tabela hash. Em seguida, o programa recebe comandos e os executa. Cada comando produz uma saída específica. Os comandos são:

- A letra i, seguida de uma chave (palavra) e um valor (inteiro), para inserir uma chave/valor na estrutura.
- A letra r, seguida de uma chave (palavra) para remover uma chave/valor da estrutura.
- A letra c, seguida de uma chave (palavra) para consultar o valor de uma chave (o valor é escrito).
- A letra d, para debugar (escrever todas as informações armazenadas em formato específico.
- A letra f, para finalizar a execução do programa.

Exemplo de entrada e saída juntos:

A chave "dois" é mapeada na posição 0, a chave "quatro" na posição 1 e a chave "cinco" na posição 0.

```
5
i dois 2
i quatro 4
i cinco 5
d
[0/dois/2] [1/quatro/4] [2/cinco/5]
f
```

Exemplo de entrada e saída juntos:

As chaves "cinco", "seis" e "dois" são mapeadas na posição 0, a chave "quatro" na posição 1.

```
5
i cinco 5
i quatro 4
i seis 6
d
[0/cinco/5] [1/quatro/4] [2/seis/6] [3] [4]
r quatro
d
[0/cinco/5] [1/removido] [2/seis/6] [3] [4]
i dois 2
```

```
d
[0/cinco/5] [1/dois/2] [2/seis/6] [3] [4]
r cinco
d
[0/removido] [1/dois/2] [2/seis/6] [3] [4]
r sete
ERRO
c seis
6
c dois
2
f
```


Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Choose File No file chosen Enviar Resposta

Questão 3: Hash com endereçamento aberto e redimensionamento

Implemente uma tabela hash com tratamento de colisão por endereçamento aberto. Use o código fornecido. A implementação deve seguir a seguinte estratégia:

- A tabela hash é criada para uma determinada capacidade. Toda vez que a inserção encontrar estrutura cheia, a capacidade aumenta de forma a permitir o armazenamento de cinco novos elementos.
- Os espaços de armazenamento podem estar em um de três estados; 1) VAZIO, 2) REMOVIDO, 3) COM VALOR. Todos os espaços começam VAZIOS e depois de alterados nunca mais voltam a ser VAZIOS.
- Em caso de colisão, usa-se a posição consecutiva e assim sucessivamente. A posição que sucede a última é a primeira.
- A busca de itens na tabela hash termina ao encontrar um espaço VAZIO.
- A tabela hash armazena pares chave/valor, onde a chave é do tipo texto e o valor é de tipo indeterminado. A chave vazia (string vazia) é especial e não pode ser associada a valores. Não é permitido criar outros valores especiais para chave, esperando que ninguém nunca use um determinado valor como chave.

Com relação ao tratamento de erros, o código deve ser a prova de programadores descuidados. Mandar remover de estrutura vazia, por exemplo, não pode passar despercebido, mas a decisão sobre o que fazer não pode ser tomada na classe Hash. A função main é a responsável por determinar o que será feito. Em todos os casos de erro, a função main vai simplesmente escrever "ERRO" na saída padrão. A implementação dada usa manipulação de exceções, mas você pode mudar para códigos de erros caso não tenha prática com manipulação de exceções.

A implementação dada usa ponteiros para determinar os 3 estados de uma posição do armazenamento o ponteiro especial REMOVIDO já está implementado. Atenção na hora de desalocar memória. É permitido alterar a forma como os 3 estados são implementados, mas você terá que alterar coisas que já estão prontas.

Existe um método pronto para escrever a posição, chave e valor de todos os valores da tabela hash. Esse método mostra posições VAZIAS, REMOVIDAS e ocupadas com sintaxes distintas. Se você quiser mudar a forma como os 3 estados são implementados deveria mudar este método sem alterar a saída que ele produz. A função hash que mapeia chaves (texto) em posições na tabela também está pronta e não pode ser alterada.

A recuperação do valor de uma chave deve ser eficiente: não basta encontrar um valor, é necessário atender a estratégia acima.

Entradas:

A parte de interface do programa já está implementada. Inicialmente é lido um valor para a capacidade da tabela hash. Em seguida, o programa recebe comandos e os executa. Cada comando produz uma saída específica. Os comandos são:

- A letra i, seguida de uma chave (palavra) e um valor (inteiro), para inserir uma chave/valor na estrutura.
- A letra r, seguida de uma chave (palavra) para remover uma chave/valor da estrutura.
- A letra c, seguida de uma chave (palavra) para consultar o valor de uma chave (o valor é escrito).
- A letra d, para debugar (escrever todas as informações armazenadas em formato específico.
- A letra f, para finalizar a execução do programa.

Exemplo de entrada e saída juntos:

```
3
i um 1
i dois 2
i tres 3
d
[0/dois/2] [1/tres/3] [2/um/1]
i quatro 4
d
[0] [1/um/1] [2] [3] [4] [5/tres/3] [6/dois/2] [7/quatro/4]
f
```

Exemplo de entrada e saída juntos:

```
3
i quatro 4
i tres 3
i dois 2
d
[0/quatro/4] [1/tres/3] [2/dois/2]
r dois
r tres
d
[0/quatro/4] [1/removido] [2/removido]
i um 1
d
[0/quatro/4] [1/removido] [2/um/1]
i cinco 5
d
[0/quatro/4] [1/cinco/5] [2/um/1]
i seis 6
d
[0] [1/um/1] [2] [3/cinco/5] [4] [5/quatro/4] [6/seis/6] [7]
f
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Choose File No file chosen Enviar Resposta

de Renato R. R. de Oliveira.

Minutos Restantes:
32568

Usuário:
Lucas Antonio Lopes Neves

Notas:
Q1: ?
Q2: ?
Q3: ?
Total: 0