Dredd - Juiz Online

Principal | Perfil | Minhas Provas | Sair

Minutos Restantes: 10910

Usuário: Lucas Antonio Lopes Neves

Notas:
Q1: 100
Q2: 42.5
Q3: ?
Q4: 98
Q5: 68.6
Q6: 100
Q7: 98
Q8: ?
Total: 63

Exercícios sobre Hashes, Heaps e Torneios

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.

É permitido supor que não vão tentar inserir chaves repetidas na tabela hash.

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

Última tentativa realizada em: 01/10/2018 22:12:21

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

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

Última tentativa realizada em: 16/10/2018 21:31:21

Tentativas: 1 de 6

Nota (0 a 100): 42.5

Status ou Justificativa de Nota: A quantidade de dados escritos pelo programa é diferente da quantidade de dados esperados. O programa não resolve todas as instâncias do problema.

Ver Código da Última Tentativa

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

Questão 4: Tabela hash - Dicionário

Utilize uma tabela hash para implementar um dicionário de capacidade para 23 palavras, sua tabela terá o tratamento de colisões por encadeamento. Seu programa deve ser capaz de armazenar e buscar por uma palavra e seu significado. Caso seja feita uma busca por uma palavra inexistente em seu dicionário, o programa imprimirá NULL no lugar do significado da palavra. Seu programa deve permitir quantas buscas forem desejadas pelo usuário, sendo -1 a palavra que representa a intenção de termino do programa. **OBS:** Sua função que faz o hash será da seguinte forma: tamanho_da_palavra mod 23.

Entradas:

1. Quantidade de palavras a serem inseridas.
2. Palavra.
3. Significado da Palavra.
4. Palavras buscadas.

Saídas:

1. [Palavra_buscada] => Significado da palavra buscada.

Exemplo de Entrada:

```
5
Casa Edifício de formatos e tamanhos variados, ger. de um ou dois andares, quase sempre destinado
Aniversário Diz-se de ou dia em que se completa um ou mais anos em que se deu determinado aconteci
Carta Mensagem, manuscrita ou impressa, a uma pessoa ou a uma organização, para comunicar-lhe algo
Exonerado  Libertar ou libertar-se de uma obrigação ou de um dever.
Concomitantemente Que acompanha ou coexiste. Que acontece ou se faz ao mesmo tempo.
Carta
Casa
-1
```

Exemplo de Saída:

```
[Carta] => Mensagem, manuscrita ou impressa, a uma pessoa ou a uma organização, para comunicar-lhe
[Casa] => Edifício de formatos e tamanhos variados, ger. de um ou dois andares, quase sempre desti
```

Peso: 1

Última tentativa realizada em: 13/10/2018 10:34:29

Tentativas: 2 de 6

Nota (0 a 100): 98

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

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 5: Tabela Hash – Aniversários

Sua empresa passará a presentear seus funcionários com um presente aleatório no dia de seus respectivos aniversários. Para isso ela necessita de um sistema que seja possível cadastrar os funcionários, sendo necessários armazenar seu nome e dia, mês e ano de seu aniversario, e então dada uma data saber quantos presentes necessitam ser preparados. Sua tarefa é desenvolver esse programa utilizando uma tabela hash, de forma que são cadastrados vários funcionários e então é fornecida uma data (dia/mês), com isso o programa deve exibir a quantidade de presentes necessários naquele dia.

A tabela hash deve ser criada com uma quantidade de posições fornecida no início da execução. Após escrever a quantidade de presentes necessários, escreva a porcentagem de posições da tabela que estão usadas na forma de um número entre 0 e 1. Use, a seu critério, o código fornecido, que já tem pronto o tipo funcionário e outras coisas. Não é necessário usar os mesmos métodos que estão no programa, mas os métodos que não forem usados devem ser apagados.

A mesma função hash que está no programa deve ser usada: (dia * mes - 1) *mod* limite (onde *mod* é a operação de resto da divisão).

Entradas:

1. Quantidade de posições que a tabela hash deve ter.
2. Quantidade de funcionários a serem cadastrados.
3. Nome de cada funcionário (uma palavra só) com sua data de nascimento (3 inteiros, não é necessário verificar se a data é válida).
4. Uma data qualquer (dois inteiros para o dia e mês).

Saída:

1. Quantidade de aniversariantes naquele dia.
2. Porcentagem de posições usadas na tabela (número real entre 0 e 1).

Exemplo de Entrada:

```
4
5
João 10 05 1985
Maria 25 09 1976
Marcos 03 01 1992
Paulo 06 07 1993
Pedro 25 09 1996
25 09
```

Exemplo de Saída:

```
2 0.75
```

Peso: 1

Última tentativa realizada em: 04/10/2018 11:39:49

Tentativas: 2 de 6

Nota (0 a 100): 68.6

Status ou Justificativa de Nota: O programa não resolve todas as instâncias do problema.

Ver Código da Última Tentativa

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 6: Heap - Construção de MinHeap passo a passo

Implemente uma classe MinHeap que tem um construtor que recebe um vetor de elementos. Durante a criação do MinHeap, alguns elementos deverão ser reposicionados para que passem a ter as propriedades de um min heap. O construtor (ou método usado no construtor) deverá escrever passo a passo como fica o armazenamento após a *correção descendo* de cada posição.

Note que não é para escrever os valores do heap após cada troca, os valores do heap devem ser escritos após todas as trocas (zero ou mais) realizadas para a correção de uma posição.

Implemente a função principal que cria um heap a partir do vetor.

Entradas:

1. A quantidade de elementos a serem lidos.
2. Os elementos (números inteiros) a serem armazenados no heap.

Saídas:

Para cada posição a ser corrigida:

1. A posição, seguida de dois pontos;
2. Todos os elementos armazenados após a reorganização iniciada na posição em questão.

Note a ausência de espaço antes do dois pontos e a presença do espaço depois, como em qualquer sinal de pontuação.

Exemplo de Entrada:

```
6
9 8 7 5 -1 -3
```

Exemplo de Saída:

```
2: 9 8 -3 5 -1 7
1: 9 -1 -3 5 8 7
0: -3 -1 7 5 8 9
```

Peso: 1

Última tentativa realizada em: 11/10/2018 11:26:23

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

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 7: Heap - Heapsort

Utilizando como base uma implementação de MaxHeap, implemente o heapsort. O algoritmo heapsort é um algoritmo de ordenação generalista baseado em uma árvore heap, ou seja, ele constrói um heap a partir de um vetor desordenado para

ordenar seus elementos.

O programa deve ler 15 elementos e criar um heap com eles. Depois, deve inserir mais 5 elementos no heap. Em seguida, deve escrever os elementos em ordem decrescente.

Entradas:

1. Quinze elementos a serem inseridos no vetor.
2. Cinco elementos adicionais a serem inseridos no heap construído com o vetor lido anteriormente

Saídas:

1. MaxHeap construído a partir do vetor inicial.
2. MaxHeap anterior, acrescido dos cinco elementos adicionais.
3. Elementos ordenados, em ordem decrescente, por heapsort.

Exemplo de Entrada:

```
6 1 3 48 97 0 55 74 31 1 2 3 4 5 6
35 69 82 10 20
```

Exemplo de Saída:

```
97 74 55 48 2 4 6 6 31 1 1 3 0 5 3
97 82 55 74 20 4 6 48 69 2 1 3 0 5 3 6 35 31 10 1
97 82 74 69 55 48 35 31 20 10 6 6 5 4 3 3 2 1 1 0
```

Peso: 1

Última tentativa realizada em: 11/10/2018 10:37:05

Tentativas: 2 de 6

Nota (0 a 100): 98

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

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 8: Torneio

Um torneio é uma árvore estritamente binária na qual cada nó não folha contém uma cópia do maior elemento entre seus dois filhos. O conteúdo das folhas de um torneio determina o conteúdo de todos os seus nós. Um torneio com n folhas representa um conjunto de n elementos.

Implemente um torneio usando estratégia similar a um *MaxHeap*, com métodos para Inserir e Remover valores. O código fornecido contém um projeto com métodos e atributos, além de um programa de interface para inserir e remover. Os métodos auxiliares são sugestão de implementação.

Como um torneio é uma árvore estritamente binária, cheia, implementada num vetor de capacidade limitada, a estratégia é ajustar o tamanho indicado pelo usuário. Por exemplo, se usuário quiser um torneio com capacidade para 13 valores, será criado um torneiro para 16 valores (menor potência de 2 maior ou igual à tamanho desejado pelo usuário). Quinze espaços para nós intermediários existirão para suportar os 16 espaços para folhas.

Os espaços não usados receberão um valor especial; nesta implementação, eles recebem o menor valor representável. Se vários valores com esse valor especial existem, então nós intermediários também terão esse valor especial. Como esse valor especial pode ser muito cheio de algarismos, o programa o mostra como ## (duas cerquilhas).

A inserção procura uma folha com valor especial e substitui a primeira posição encontrada.

Entradas:

O programa lê a quantidade de valores a guardar no torneio, e ajusta esse valor. Depois lé vários comandos para inserir, remover, escrever ou terminar a execução.

1. A quantidade de valores a armazenar no torneio (inteiro maior que zero).
2. Vários comandos (letras minúsculas) com seus argumentos.

Os comandos são:

- f - para finalizar a execução do programa.
- i - para inserir valor no torneio, seguido do valor (inteiro) a inserir.
- r - para remover valor do torneio. O maior valor será removido.
- e - para escrever o conteúdo do torneio.

Saídas:

1. O comando para remover escreve o valor removido, que é sempre o maior valor do torneio.
2. O comando para escrever, escreve o torneio num formato especial e já está pronto. Ele substitui o valor especial por ## e escreve uma barra vertical entre os nós intermediários e as folhas, para facilitar a depuração.

Exemplo de entrada e saída juntos:

```
5
i 15
e
[15 15 ## 15 ## ## ## | 15]
i 13
e
[15 15 ## 15 ## ## ## | 15 13]
i 18
e
[18 18 ## 15 18 ## ## | 15 13 18]
i 21
e
[21 21 ## 15 21 ## ## | 15 13 18 21]
i 11
e
[21 21 11 15 21 11 ## | 15 13 18 21 11]
r
21
e
[18 18 11 15 18 11 ## | 15 13 18 ## 11]
r
18
e
[15 15 11 15 ## 11 ## | 15 13 ## ## 11]
i 14
e
[15 15 11 15 14 11 ## | 15 13 14 ## 11]
i 20
e
[20 20 11 15 20 11 ## | 15 13 14 20 11]
r
20
r
15
e
[14 14 11 13 14 11 ## | ## 13 14 ## 11]
r
14
r
13
e
[11 ## 11 ## ## 11 ## | ## ## ## ## 11]
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