Quiz 13 - Tabela Hash

- Entrega 26 mai em 9:16
- Pontos 100
- Perguntas 5
- Disponível 26 mai em 8:50 - 26 mai em 9:16 26 minutos
- Limite de tempo Nenhum

Este teste foi travado 26 mai em 9:16.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	6 minutos	100 de 100

Pontuação deste teste: 100 de 100
Enviado 26 mai em 9:11
Esta tentativa levou 6 minutos.

Pergunta 1
20 / 20 pts
Os profissionais que desenvolvem sistemas para a Internet devem saber empregar linguagens de programação, raciocínio lógico e estruturas de dados adequadas. Dessa forma, o uso de estruturas de dados como tabelas e dicionários são um pré-requisito indispensável dado que o principal diferencial dessas estruturas são o fato delas serem eficientes para
Correto!

- (•) pesquisar a existência de elementos contidos nas mesmas.

A alternativa está CORRETA. As estruturas de tabelas e dicionários têm como principal diferencial pesquisar a existência de elementos sendo que o custo dessa operação é $\Theta(1)$.

- ( ) recuperar a média dos valores contidos nas mesmas.
- ( ) pesquisar o maior elemento contido nas mesmas.
- ( ) recuperar a ordem em que os elementos foram inseridos nas mesmas.
- ( ) recuperar a ordem em que os elementos serão removidos das mesmas.

Pergunta 2
20 / 20 pts
As estruturas de dados são fundamentais na programação de computadores. Uma dessas estruturas tem como propriedades: ser baseada em pares do tipo chave/elemento; e, permitir o acesso aos seus elementos com custo de $\Theta(1)$. A estrutura de dados que possui tais

propriedades é conhecida como:
Correto!

- Tabela *Hash*

A afirmação é verdadeira porque as tabelas *hash* permitem o acesso a elementos com custo $\Theta(1)$ e são baseadas em pares do tipo chave/elemento.

- Árvore Binária
- Árvore Binária Balanceada
- Lista flexível
- Lista sequencial

Pergunta 3
20 / 20 pts
As tabelas *hash* são estruturas de dados importantes no desenvolvimento de sistemas computacionais. Elas armazenam dados de maneira associativa, fazendo com que cada registro de dado seja armazenado como um par chave/valor. Nesse caso, cada valor possui uma chave de identificação e essa é utilizada para encontrar os valores armazenados na tabela. Considerando tais estruturas é correto o que se afirma em
Correto!

- As coleções *HashTable* e *Dictionary* implementam os conceitos de tabela *hash* em C#.

A alternativa está CORRETA. As implementações de tabela hash em C# são *HashTable* e sua versão genérica, o *Dictionary*.

- As operações de pesquisa e encontrar o menor elemento são eficientes nas tabelas *hash* e o custo dessas operações nessa estrutura é $\Theta(1)$.
- As tabelas *hash* contém as operações de inserção e pesquisa, contudo, não possuem a operação de remoção.
- A inserção de elementos nas tabelas *hash* deve ser efetuada em ordem crescente de valores.
- As colisões primárias são um problema importante nessas estruturas e devem ser tratados utilizando listas lineares.

Pergunta 4
20 / 20 pts
O profissional da computação desenvolve soluções específicas em sistemas. Esse profissional analisa, documenta, projeta, implementa, testa e gerencia os sistemas computacionais necessários para seus clientes. Ele faz o levantamento de requisitos, atua na codificação, no levantamento de hardware e na manutenção/atualização do sistema. Uma das principais

habilidades do profissional da computação é ser capaz de analisar problemas e propor soluções algorítmicas, observando as estruturas de dados do sistema tais como listas, árvores e tabelas *hash*. O principal desafio no projeto das tabelas *hash* consiste em

Correto!

- tratar colisões.

A alternativa está CORRETA. O principal desafio no projeto das tabelas *hash* consiste em tratar colisões porque as tabelas *hash* fazem um mapeamento da chave de pesquisa em uma posição da tabela. Dois ou mais elementos podem ter a mesma chave de pesquisa, causando uma colisão.

- encontrar o menor elemento.
- ordenar os valores,
- remover valores.
- somar os valores.

Pergunta 5

20 / 20 pts

A Linguagem C# implementa a estrutura de tabela *hash* de forma nativa através da *Collection Hashtable*.

Abaixo, temos um exemplo de programa usando a classe *Hashtable*.

```
Hashtable h = new Hashtable();
string[] palavra = {"João", "Maria", "José", "Pedro",
                    "Patrícia", "Luciana", "Fernando",
                    "Juliana","Antônio", "Júlia"};

for(int i = 0; i < 10; i++){
  h.Add(i, palavra[i]);
}
h.Remove("João");

Console.WriteLine(h.Contains(1));
Console.WriteLine(h.Contains("João"));
Console.WriteLine(h.ContainsKey(1));
Console.WriteLine(h.ContainsValue("João"));
```

A respeito da classe *Hashtable* e considerando o exemplo acima, avalie as afirmações a seguir:

I) Os comandos Contains(1) e h.Contains("João") retornam *True*.

II) O comando ContainsKey(1) retorna verdadeiro.

III) O comando h.ContainsValue("João") retorna verdadeiro.

É **correto** o que se afirma em:

- I e II, apenas.
- I e III, apenas.

Correto!

- II e III, apenas.

A afirmação I é falsa porque o comando h.Remove("João") recebe como parâmetro uma chave sendo que João é o valor cuja chave é 1. O comando *Contains* também recebe uma chave como parâmetro. Logo, h.Contains(1) retorna verdadeiro e h.Contains("João"), falso.

As afirmações II e III são verdadeiras porque nossa tabela tem o par chave/valor 1/João.

- I, II e III.

Pontuação do teste: 100 de 100