INSTITUTO FEDERAL DE ALAGOAS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO
Estruturas de Dados
Prof. Ricardo Rubens
ricardo (arroba) ifal.edu.br

Lista de Exercícios 09 (Hashing)

1. Implemente o método `__len__` para uma classe `HashTable` que retorne o número de itens na tabela hash.

2. Implemente o método `__contains__` para verificar se uma chave está presente em uma tabela hash.

3. Considere o trecho de código abaixo. Qual será o valor retornado pela função `hash_val` para `chave = 25`?

```python
def hash_val(chave, tamanho):
    return chave % tamanho

tamanho = 10
print(hash_val(25, tamanho))
```

4. Qual técnica de rehash está sendo implementada no código a seguir?

```python
def rehash(indice, tamanho, tentativa):
    return (indice + tentativa ** 2) % tamanho
```

5. Qual o resultado da execução do código abaixo?

```python
tabela = [None] * 7
chave = 20
indice = chave % len(tabela)
tabela[indice] = 'Valor'
print(tabela)
```

6. Explique o conceito de "hashing" e como ele é utilizado para organizar dados em uma tabela hash.

7. Quais são os principais métodos de resolução de colisão em tabelas hash e como cada um funciona?

8. Como o fator de carga influencia o desempenho de uma tabela hash?

9. Em que situações o redimensionamento automático de uma tabela hash é necessário?

10. Quais são as vantagens e desvantagens do encadeamento comparado ao *open address* aberto?

11. Explique como o método de quadratic probing funciona para resolver colisões e como ele difere da linear probing.

12. Na implementação de Tabela Hash, o tamanho inicial da tabela foi definido como 113. Se a tabela ficar cheia, é necessário aumentar o tamanho. Reimplemente o método `put` para que a tabela redimensione-se automaticamente quando o fator de carga atingir um valor predeterminado (você pode definir o valor com base em sua avaliação de carga versus desempenho).