

# Trabalho Final: Implementação de uma Tabela de Espalhamento (Tabela Hash)

## Instruções gerais

Resolver seguindo **obrigatoriamente** as regras abaixo:

- 1) Utilizar, obrigatoriamente, **lista encadeada dupla**
- 2) Implementar inserção, consulta, quantidade de elementos por chave, remoção, etc
- 3) Considerar **tratamento de colisão**. Para isto analisar a tabela hash gerada considerando as questões:
  - a) Quando deveria ser implementado tratamento de colisão?
  - b) Como poderia ser esta implementação?
  - c) Qual sua avaliação da tabela hash gerado em relação a hipótese do hashing uniforme?
  - d) Qual sua análise em relação ao histograma de frequência de cada uma das chaves da tabela hash?
- 4) Implementar ordenação dos elementos sob uma chave (**OBS**: para a ordenação deverá ser utilizado o método **quicksort** ou outro de complexidade algorítmica equivalente).
- 5) Implementar obrigatoriamente **hashing com encadeamento**.
- 6) Usar a base de dados com 100.788 nomes de brasileiros fornecida pelo professor ([acesso aqui](#))
- 7) **A quantidade de chaves (m) deverá ser igual a 53 e a função de hash será de livre escolha do estudante.** Considere para a sua escolha o referencial teórico apresentado na aula. **OBS**: não esqueça de justificar suas escolhas no **relatório final!**
- 8) Importante questão a ser respondida é: **a hipótese do hashing uniforme foi alcançada?**
- 9) Demonstrar a distribuição dos nomes em cada uma das chaves utilizando um histograma e explicar o que o histograma está apresentando.

**OBS 1:** Apresentar, além do código, pequeno relatório **descrevendo o método (metodologia)** utilizado para resolver o problema. O relatório deverá conter um resumo em torno de 500 a 800 palavras, figuras e gráficos que demonstrem a resolução e resultados obtidos (Fazer no próprio descriptivo do GitHub).

**OBS 2:** Utilizar o GitHub para entrega do trabalho final.

**OBS 2: Apresentação até 12/07, que será o último prazo (improrrogável)**

## Exemplo de função hash e do espalhamento nas chaves

A função de hash poderá, por exemplo, ser implementada pela **função modular**, considerando a letra inicial do nome convertido para inteiro (ou seja, o código ASCII do caractere em decimal). Em um exemplo onde a Tabela Hash tenha M=10, para o nome “Maria, a primeira letra “M”, convertido para ASCII decimal o valor é 77, calculando o resto da divisão por 10, a chave será “7”. Contudo, se “maria” fosse escrito em minúsculo, o valor ASCII decimal seria 109, calculando o resto da divisão por 10 a chave seria “9”. Abaixo um exemplo das chaves e seus respectivos símbolos na tabela hash:

Chave	Nomes iniciados com ...
0	F, P, Z, d, n, x
1	G, Q, e, o, y

<b>2</b>	<b>H, R, f, p, z</b>
<b>3</b>	<b>I, S, g, q</b>
<b>4</b>	<b>J, T, h, r</b>
<b>5</b>	<b>A, K, U, i, s</b>
<b>6</b>	<b>B, L, V, j, t</b>
<b>7</b>	<b>C, M, W, a, k, u</b>
<b>8</b>	<b>D, N, X, b, l, v</b>
<b>9</b>	<b>E, O, Y, c, m, w</b>

**Exemplo gráfico do tratamento de colisões utilizando lista encadeada:**

