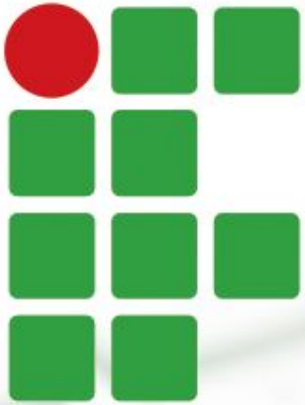


INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados 2

- Hash Tables -

Prof. MSc. Adriano Antunes Prates

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

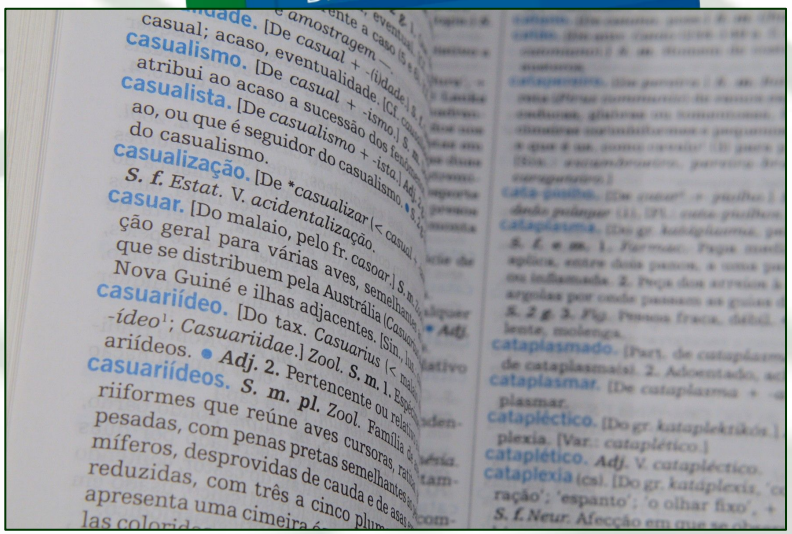
Imagine...

- Você possui um dicionário e precisa consultar o significado de algumas palavras...

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

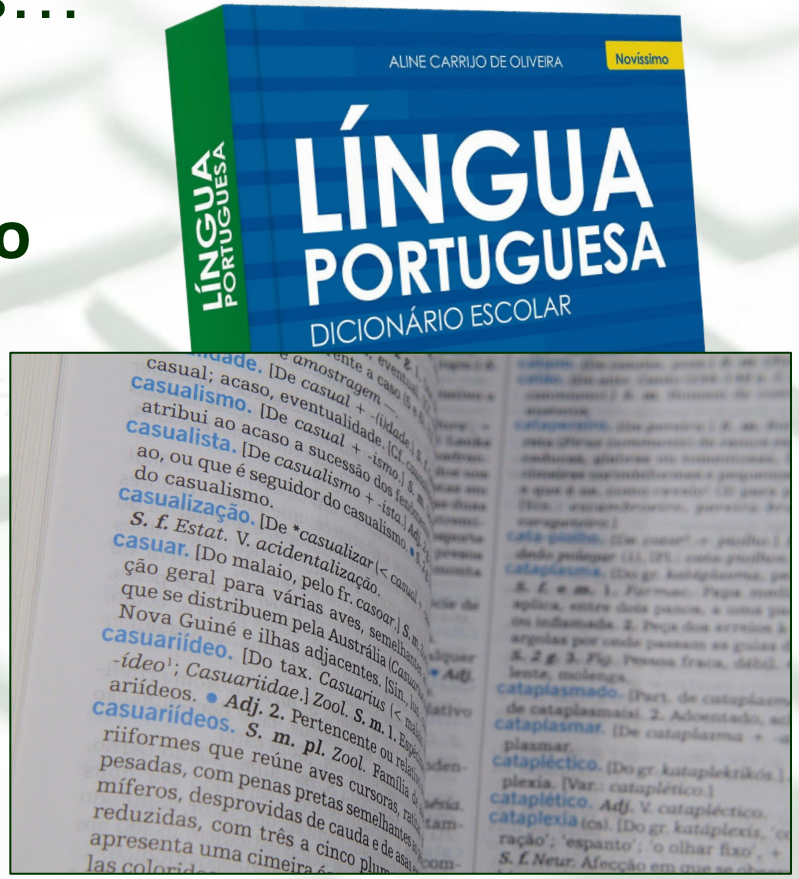
- Você possui um dicionário e precisa consultar o significado de algumas palavras...
- Para cada palavra, você **simplesmente abre o dicionário e já encontra o verbete exatamente na página aberta**!



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Você possui um dicionário e precisa consultar o significado de algumas palavras...
- Para cada palavra, você **simplesmente abre o dicionário e já encontra o verbete exatamente na página aberta**!

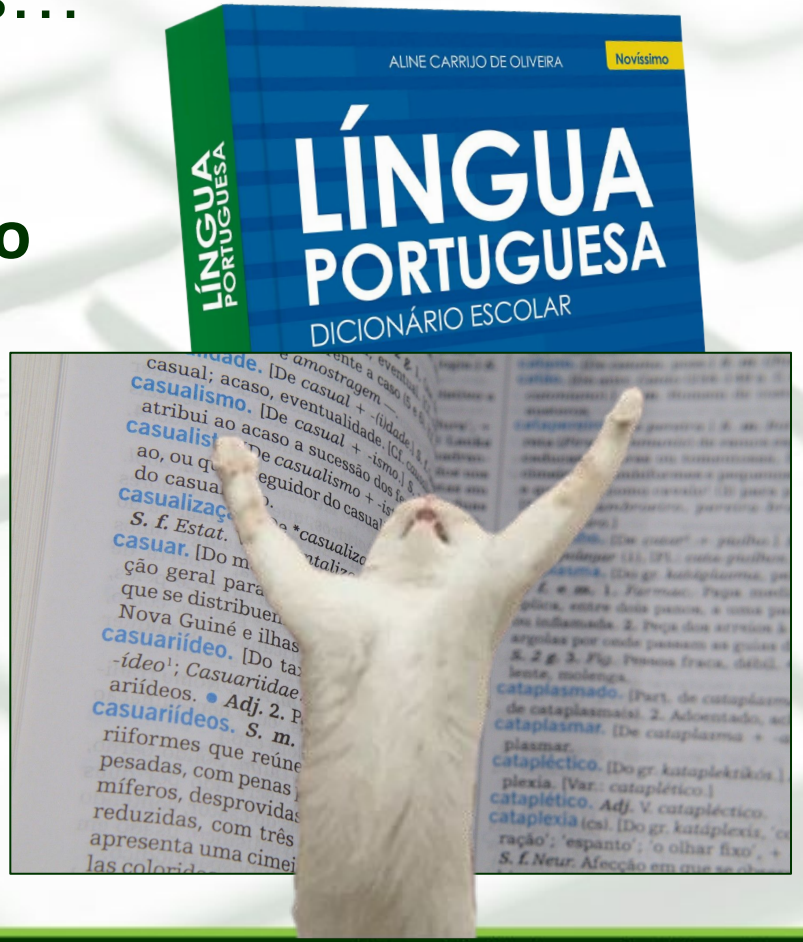


Qual seria a **Ordem de Complexidade** desta "bruxaria"?

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Você possui um dicionário e precisa consultar o significado de algumas palavras...
- Para cada palavra, você **simplesmente abre o dicionário e já encontra o verbete exatamente na página aberta**!



Qual seria a **Ordem de Complexidade** desta "bruxaria"?

O(1) == Ordem de Complexidade Constante

O Santo Graal das Estruturas de Dados

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Vocês diriam...

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Vocês diriam...

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Há sim uma forma disso ser possível!!!

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Há sim uma forma disso ser possível!!!

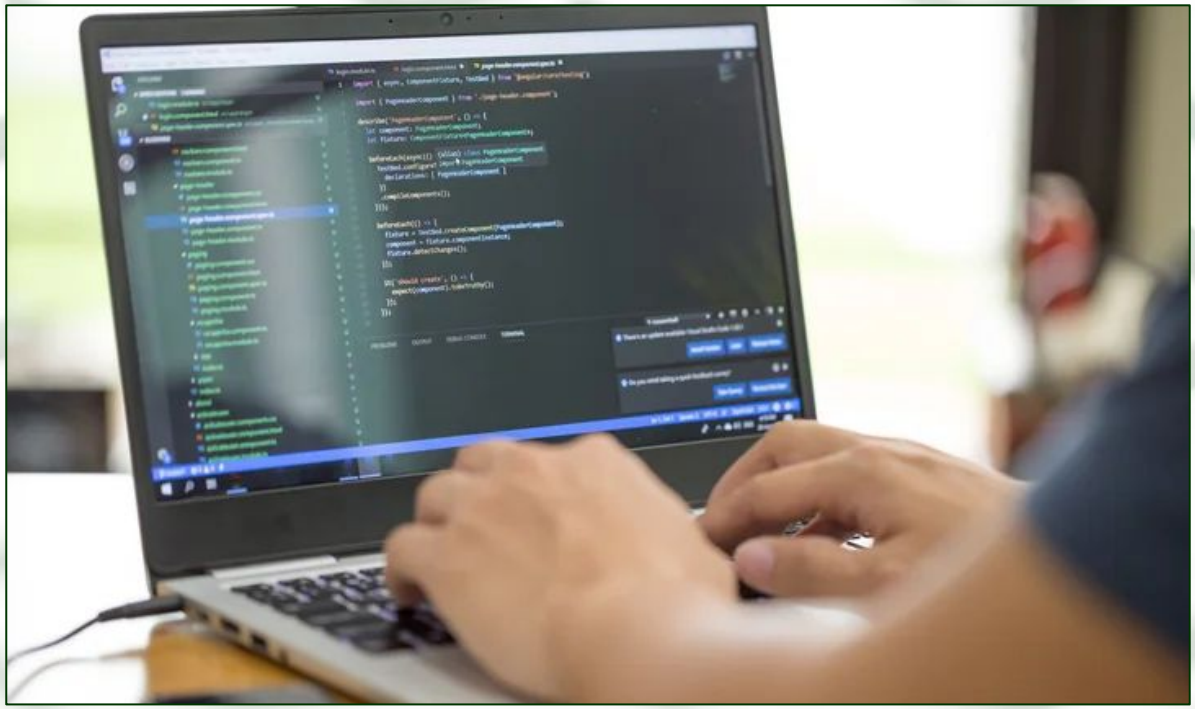
Conhecer a página de todos os verbetes do dicionário ¯_(ツ)_/¯

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Imagine...

- Ok... Para um ser humano seria complicado!
- Mas estamos falando de:

ESTRUTURAS COMPUTACIONAIS



Tabelas Hash

- **Tabelas Hash** - *Tabelas de Dispersão ou Tabelas de Espalhamento*: são Estruturas de Dados que permitem a Inserção e Consulta de Dados em Ordem de Complexidade próxima à **O(1) => Tempo Constante**!
- Um exemplo de implementação de Tabela Hash muito conhecido e utilizado é a estrutura de **dicionários (dict) existente na Linguagem Python** (também implementado na maioria das linguagens de alto nível, podendo ter outros nomes).
- Vamos observar…

Tabelas Hash

- Exemplo de **Dicionário em Python**
  - Internamente, é **uma *Tabela Hash* em operação**...

```
contatos = {}

contatos['Adriano'] = 991972046
contatos['João'] = 992823651
contatos['Maria'] = 999256324
contatos['Lúcia'] = 991523697
contatos['Antônio'] = 998586325

print('Telefone de Maria:',contatos['Maria'])
print('Telefone de Adriano:',contatos['Adriano'])
```

Tabelas Hash

- Exemplo de **Dicionário em Python**
  - Internamente, é **uma *Tabela Hash* em operação**...

```
contatos = {}
```

Cria/Inicializa um objeto Dicionário

```
contatos['Adriano'] = 991972046
contatos['João'] = 992823651
contatos['Maria'] = 999256324
contatos['Lúcia'] = 991523697
contatos['Antônio'] = 998586325

print('Telefone de Maria:',contatos['Maria'])
print('Telefone de Adriano:',contatos['Adriano'])
```

Tabelas Hash

- Exemplo de **Dicionário em Python**
  - Internamente, é **uma *Tabela Hash* em operação**…

```
contatos = {}
```

Cria/Inicializa um objeto Dicionário

```
contatos['Adriano'] = 991972046
contatos['João'] = 992823651
contatos['Maria'] = 999256324
contatos['Lúcia'] = 991523697
contatos['Antônio'] = 998586325
```

Insere dados no dicionário.

*A **Inserção é Dinâmica** e possui **tempo de complexidade constante***

```
print('Telefone de Maria:',contatos['Maria'])
print('Telefone de Adriano:',contatos['Adriano'])
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- Exemplo de **Dicionário em Python**
  - Internamente, é **uma *Tabela Hash* em operação**...

```
contatos = {}
```

Cria/Inicializa um objeto Dicionário

```
contatos['Adriano'] = 991972046
contatos['João'] = 992823651
contatos['Maria'] = 999256324
contatos['Lúcia'] = 991523697
contatos['Antônio'] = 998586325
```

Insere dados no dicionário.
*A **Inserção é Dinâmica** e possui **tempo de complexidade constante****

```
print('Telefone de Maria:',contatos['Maria'])
print('Telefone de Adriano:',contatos['Adriano'])
```

*Consulta dados no dicionário. **Tempo de complexidade também é constante****

Tabelas Hash

- O que acontece é que Python (*e outras linguagens*) nos entrega uma Interface de Programação muito simples e intuitiva…

Mas como é possível entregar isso?

Como funciona esse mecanismo de Inserção e Consulta tão eficientes?

```
times = {}

times['Atlético'] = {'mascote': 'Galo', 'brasileiros': 3, 'libertadores': 1}
times['Cruzeiro'] = {'mascote': 'Raposa', 'brasileiros': 4, 'libertadores': 2}
times['Palmeiras'] = {'mascote': 'Porco', 'brasileiros': 12, 'libertadores': 3}

consulta = input('Por quem deseja procurar? ')
print(consulta,'\t',
          times[consulta]['mascote'],'\t',
          times[consulta]['brasileiros'],'BR \t',
          times[consulta]['libertadores'],'LIB')
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- O que acontece é que Python (*e outras linguagens*) nos entrega uma Interface de Programação muito simples e intuitiva…

Mas como é possível entregar isso?

Como funciona esse mecanismo de Inserção e Consulta tão eficientes?

```
times = {}

times['Atlético'] = {'mascote': 'Galo', 'brasileiros': 3, 'libertadores': 1}
times['Cruzeiro'] = {'mascote': 'Raposa', 'brasileiros': 4, 'libertadores': 2}
times['Palmeiras'] = {'mascote': 'Porco', 'brasileiros': 12, 'libertadores': 3}

consulta = input('Por quem deseja procurar? ')
print(consulta,'\t',
          times[consulta]['mascote'],'\t',
          times[consulta]['brasileiros'],'BR \t',
          times[consulta]['libertadores'],'LIB')
```

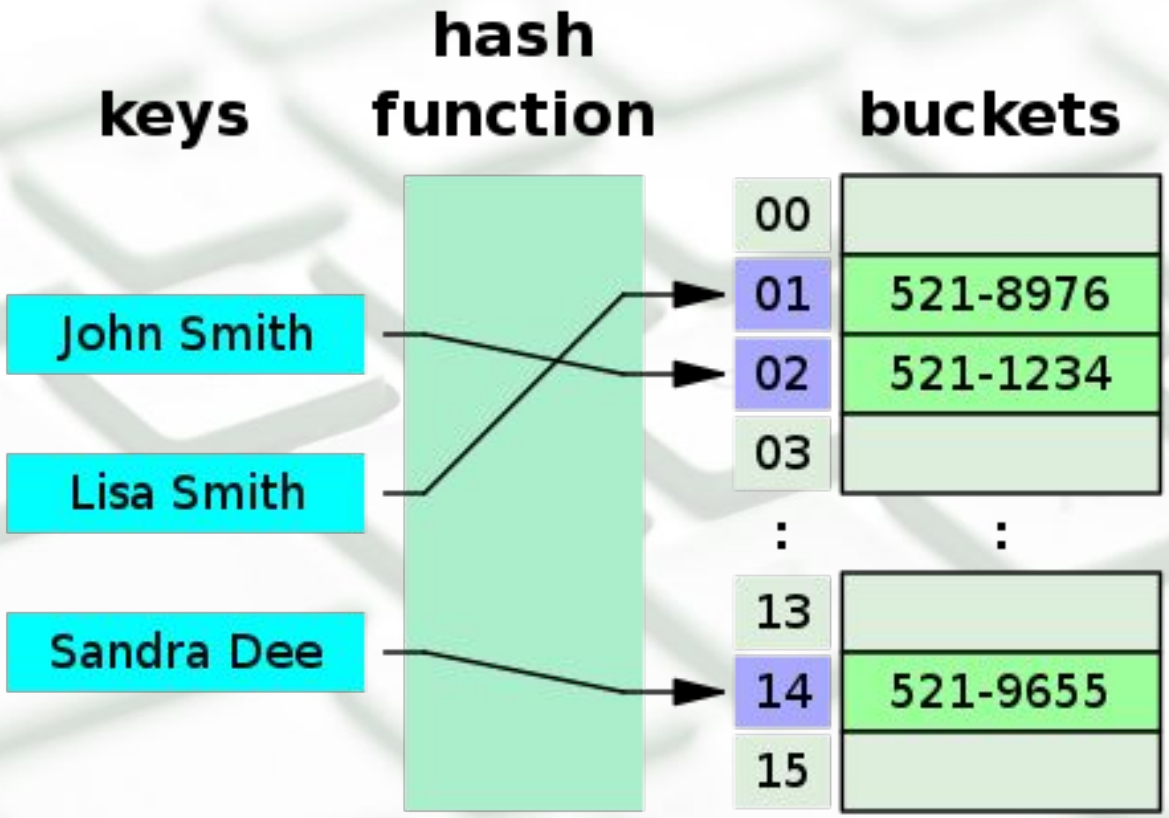
Acesso Direto
O(1)

```
Por quem deseja procurar? Palmeiras
Palmeiras      Porco    12 BR    3 LIB
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- Estrutura Básica de uma **Tabela Hash**



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

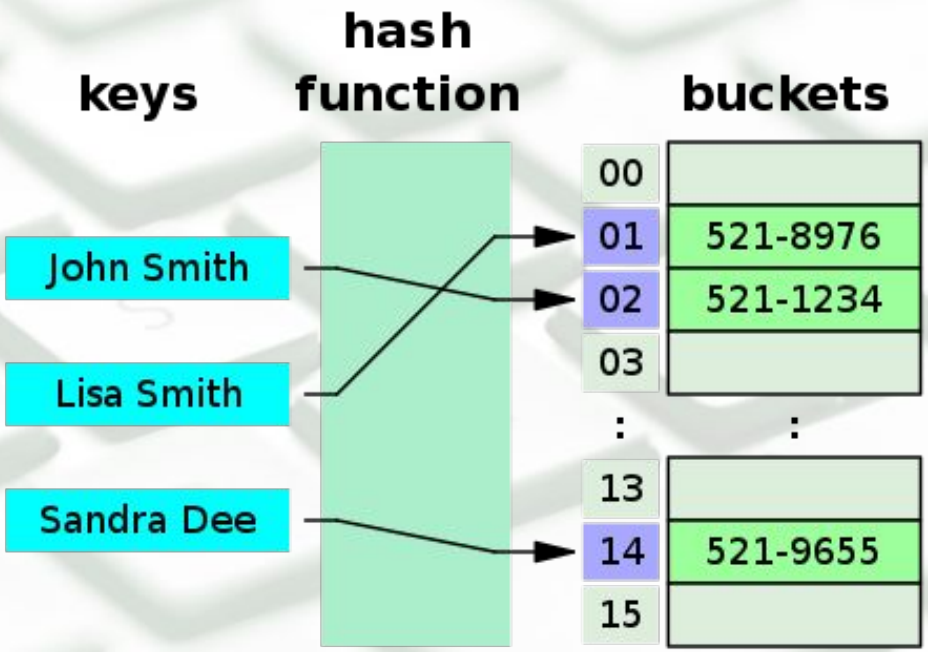
Tabelas Hash

- Estrutura Básica de uma **Tabela Hash**

Hash Function (Função Resumo)

É uma **função unidirecional**, que mapeia qualquer informação de tamanho variável (**chave**) para um **valor de tamanho fixo, num intervalo definido**.

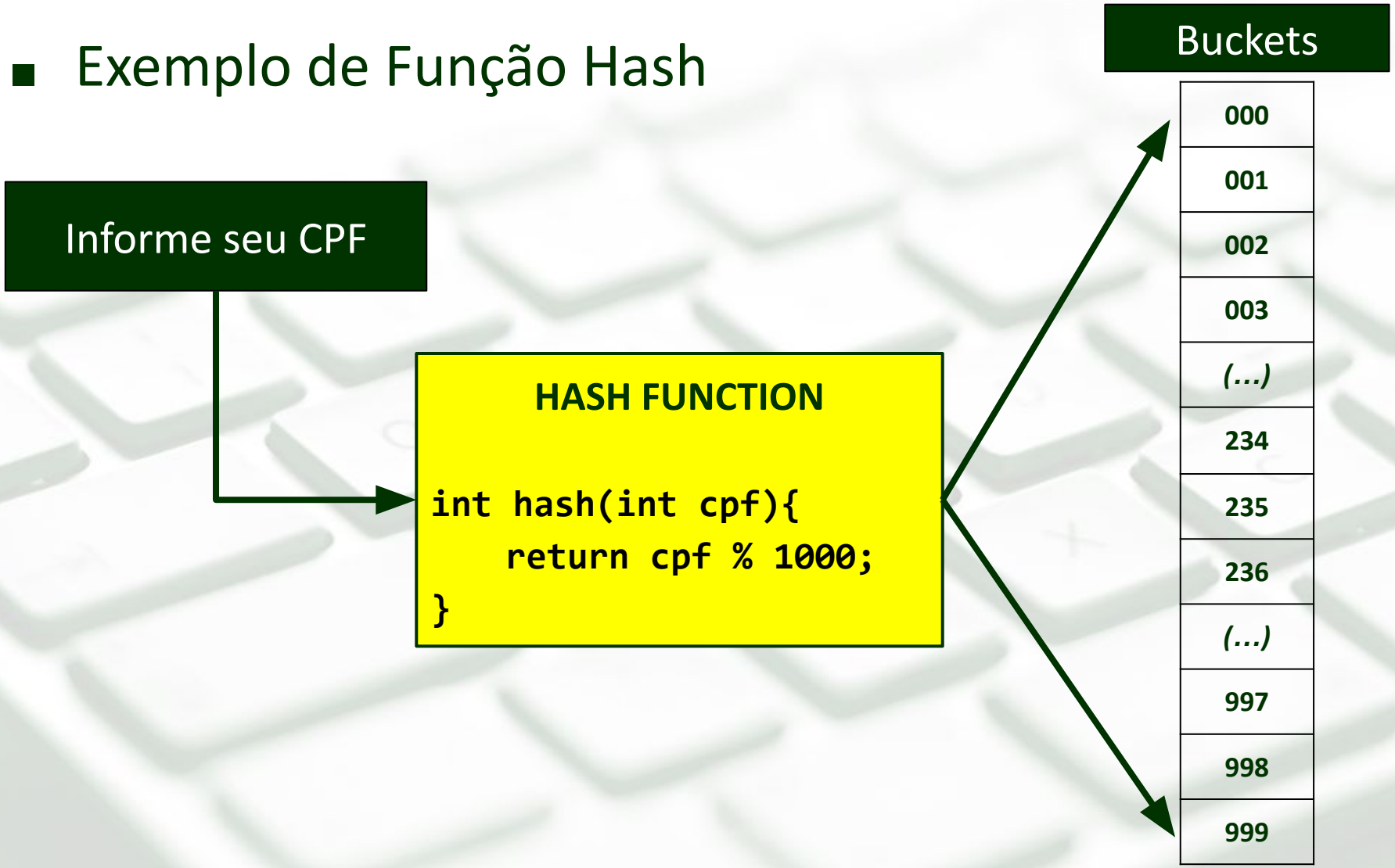
Também é chamado de "Algoritmo de Dispersão", pois a ideia é espalhar os dados de forma a facilitar a busca futura.



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

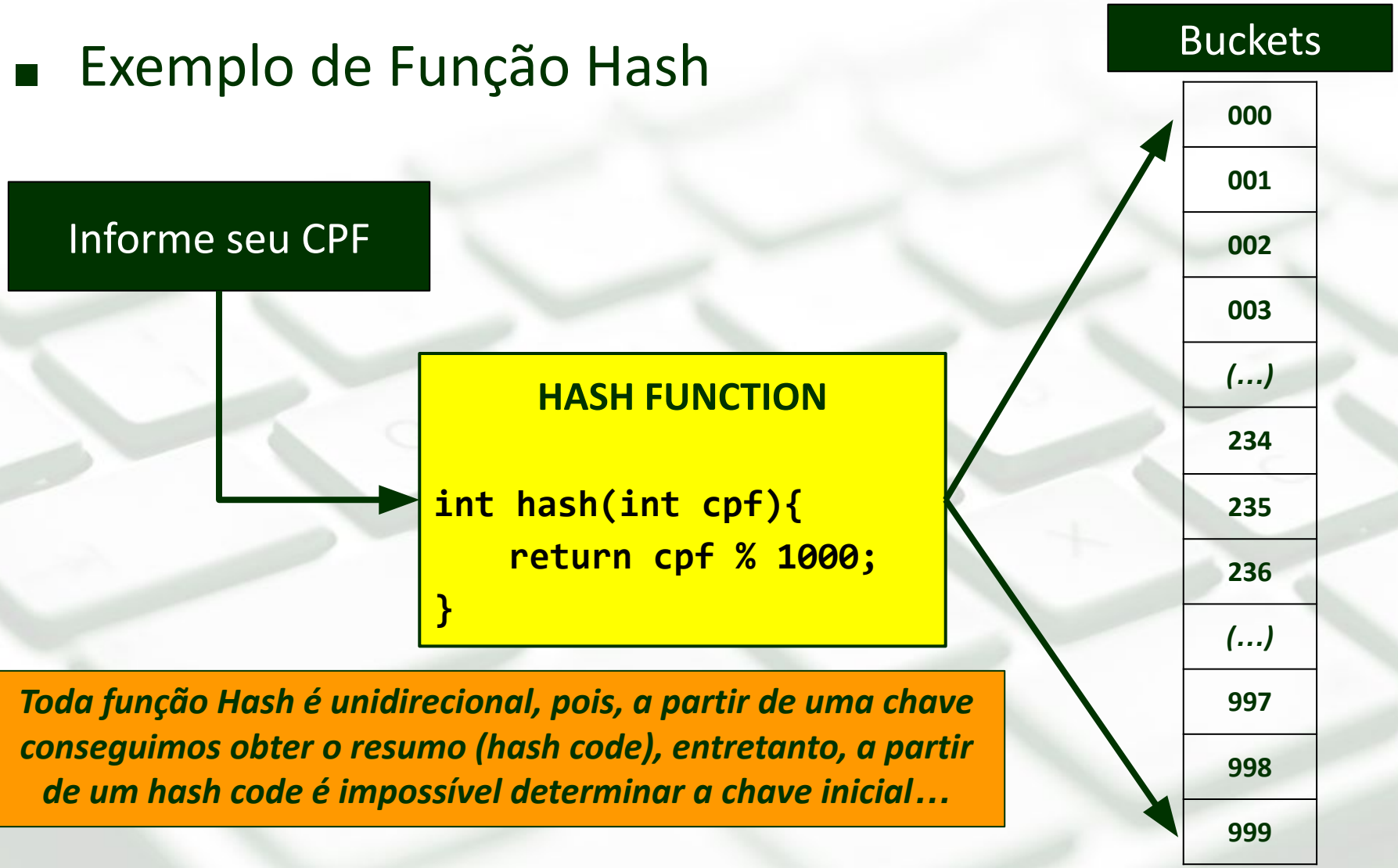
Tabelas Hash

- Exemplo de Função Hash



Tabelas Hash

- Exemplo de Função Hash

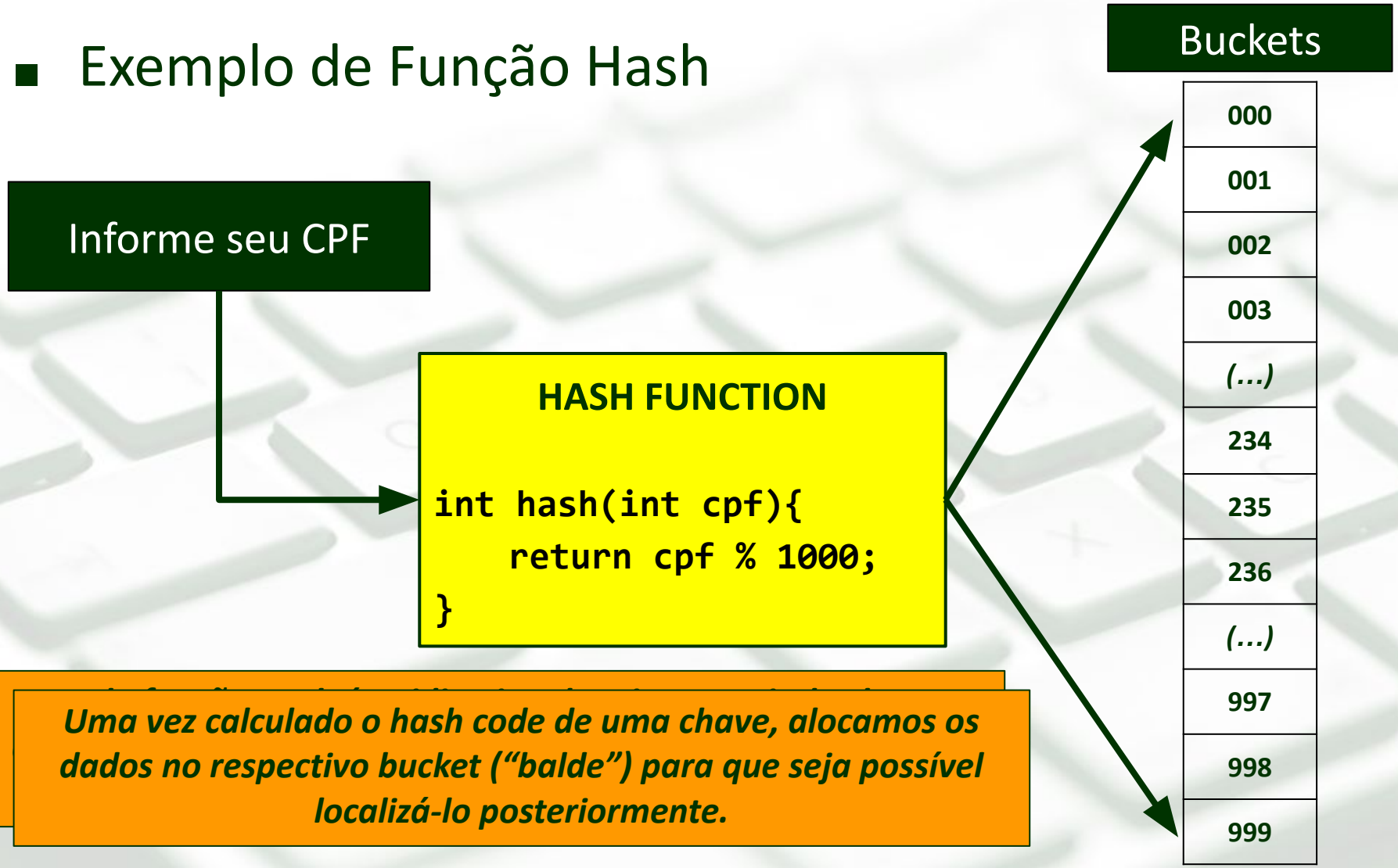


Toda função Hash é unidirecional, pois, a partir de uma chave conseguimos obter o resumo (hash code), entretanto, a partir de um hash code é impossível determinar a chave inicial…

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- Exemplo de Função Hash



Tabelas Hash

- A escolha de uma boa função Hash é determinante para o melhor desempenho da Estrutura de Dados.
- Imagine uma função Hash que recebe como chave o Nome de uma Pessoa, e devolve como resultado (*hash code*) o caractere inicial deste nome.

Esta seria uma boa função hash?

Tabelas Hash

- A escolha de uma boa função Hash é determinante para o melhor desempenho da Estrutura de Dados.
- Imagine uma função Hash que recebe como chave o Nome de uma Pessoa, e devolve como resultado (*hash code*) o caractere inicial deste nome.

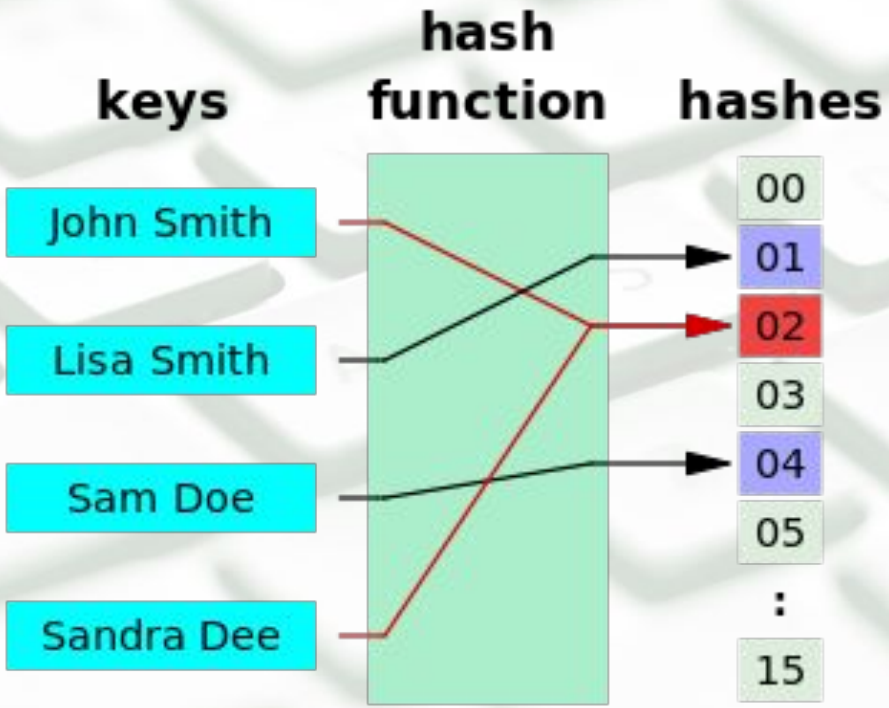
Esta seria uma boa função hash?

Qualquer função que mapeia chaves para o intervalo 0..M-1 índices serve como função de espalhamento. Entretanto, só será de fato eficiente se conseguir espalhar as chaves de maneira RAZOAVELMENTE UNIFORME.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

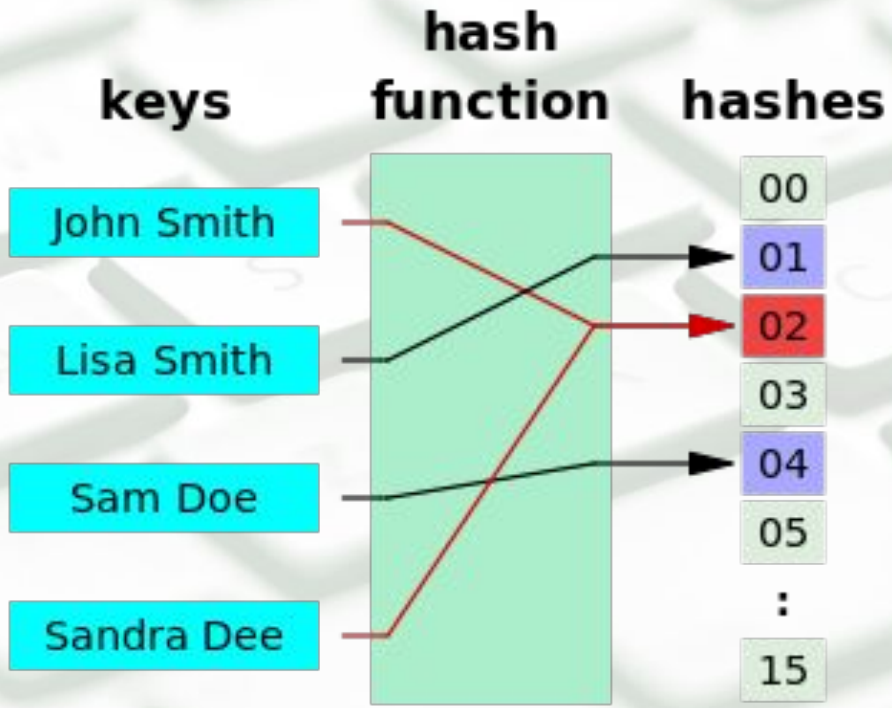
- Mas e se a Função Hash mapear duas ou mais **chaves distintas para o mesmo hash code**?



Tabelas Hash

- Mas e se a Função Hash mapear duas ou mais **chaves distintas para o mesmo hash code**?

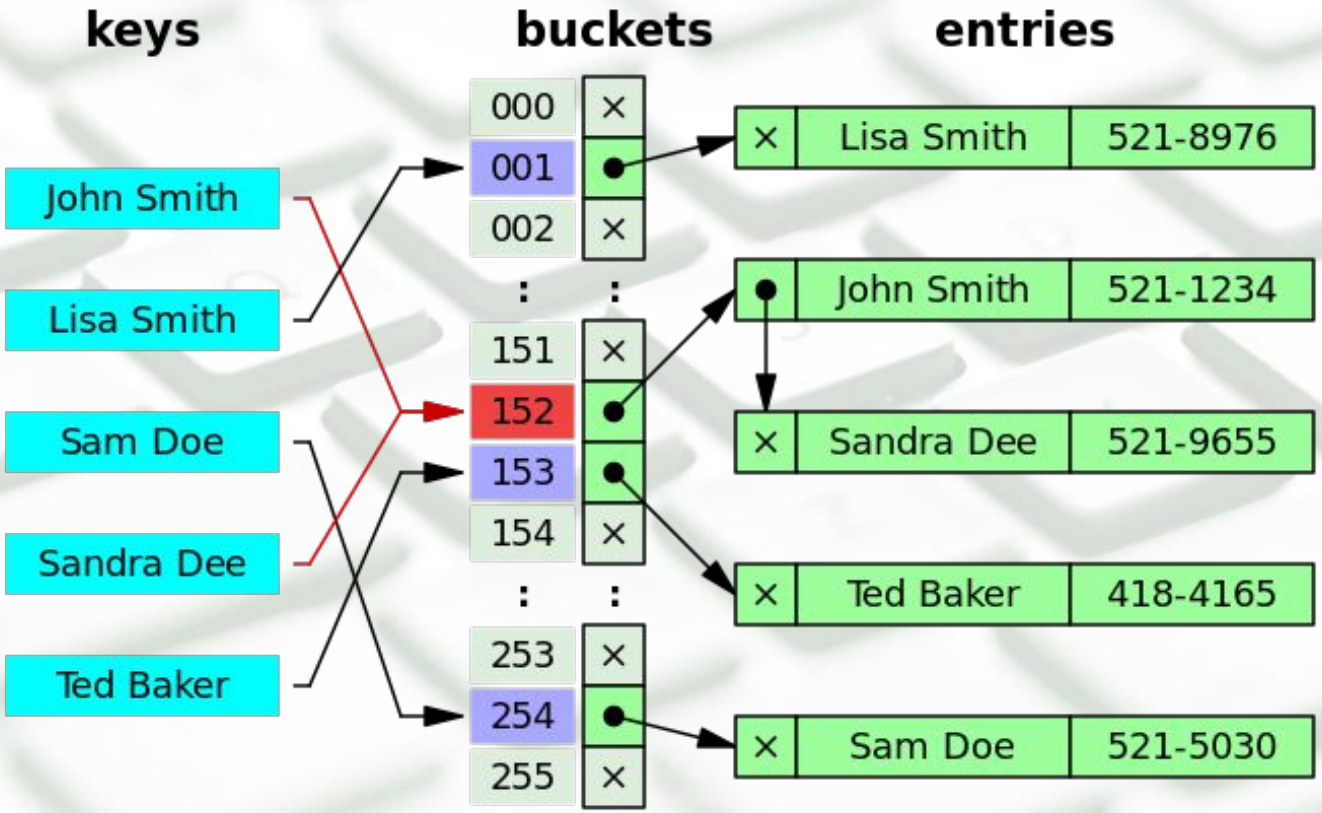
Neste caso temos uma colisão, algo comum, mas que deve ser tratado!



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- Uma forma comum para tratamento de colisões é **tornar todo bucket uma Lista Encadeada**.



Tabelas Hash

- Características de uma boa Função Hash:
  - Fácil de computar (rapidez de processamento)
  - Possuir uma estimativa do tamanho do conjunto de dados (N itens) que deverá ser mapeado.
  - Satisfazer (*ao menos aproximadamente*) a hipótese de distribuição uniforme dos N itens em M buckets.

Tabelas Hash

- Características de uma boa Função Hash:
  - Fácil de computar (rapidez de processamento)
  - Possuir uma estimativa do tamanho do conjunto de dados (N itens) que deverá ser mapeado.
  - Satisfazer (*ao menos aproximadamente*) a hipótese de distribuição uniforme dos N itens em M buckets.

```
#define M 50
#define HASH(key)(key % M)
```

Tabelas Hash

- Características de uma boa Função Hash:
  - Fácil de computar (rapidez de processamento)
  - Possuir uma estimativa do tamanho do conjunto de dados (N itens) que deverá ser mapeado.
  - Satisfazer (*ao menos aproximadamente*) a hipótese de distribuição uniforme dos N itens em M buckets.

```
#define M 50
#define HASH(key)(key % M)
```

A complexidade de acesso seria, em média, O(N/M)

Tabelas Hash

- Características de uma boa Função Hash:
  - Fácil de computar (rapidez de processamento)
  - Possuir uma estimativa do tamanho do conjunto de dados (N itens) que deverá ser mapeado.
  - Satisfazer (*ao menos aproximadamente*) a hipótese de distribuição uniforme dos N itens em M buckets.

```
#define M 50
#define HASH(key)(key % M)
```

Se N for 100 vezes o valor de M?

A complexidade de acesso seria, em média, O(N/M)

Tabelas Hash

- Características de uma boa Função Hash:
  - Fácil de computar (rapidez de processamento)
  - Possuir uma estimativa do tamanho do conjunto de dados (N itens) que deverá ser mapeado.
  - Satisfazer (*ao menos aproximadamente*) a hipótese de distribuição uniforme dos N itens em M buckets.

```
#define M 50
#define HASH(key)(key % M)
```

Porque 50 e não 500???

A complexidade de acesso seria, em média, O(N/M)

Tabelas Hash

- Sedgewick sugere escolher M da seguinte maneira.

- Defina uma potência de 2 que esteja próxima do tamanho de itens (N).
- Adote para M o número primo que esteja logo abaixo da potência escolhida.

P.Ex… Para uma Hash Table de 10k itens, faça ***HASH(Key % 8191)***

k	2^k	M
7	128	127
8	256	251
9	512	509
10	1024	1021
11	2048	2039
12	4096	4093
13	8192	8191
14	16384	16381
15	32768	32749
16	65536	65521
17	131072	131071
18	262144	262139

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Tabelas Hash

- Ordem de Complexidade de **Hash Tables**

Data Structure	Time Complexity					
	Average			Worst		
	Search	Insertion	Deletion	Search	Insertion	Deletion
Stack	Θ(n)	Θ(1)	Θ(1)	O(n)	O(1)	O(1)
Queue	Θ(n)	Θ(1)	Θ(1)	O(n)	O(1)	O(1)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)
Hash Table	Θ(1)	Θ(1)	Θ(1)	O(n)	O(n)	O(n)

Θ(1)

Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

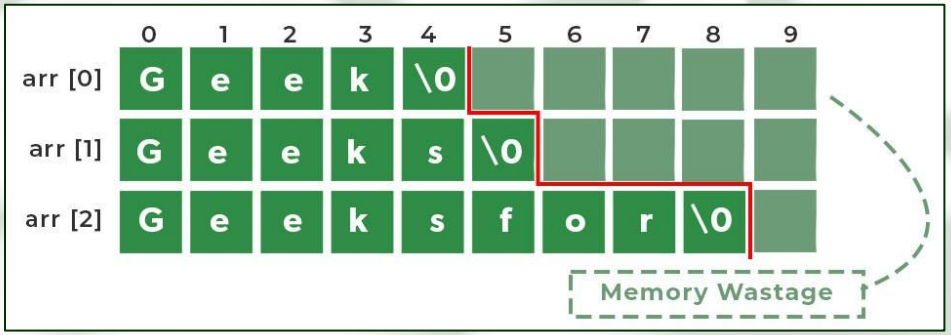
INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

- Imagine resolver este problema aplicando uma **Estrutura de Dados do tipo**:

ARRAYs

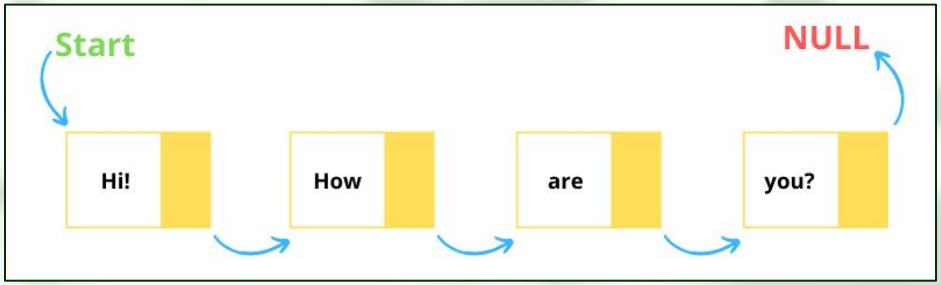


Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

- Imagine resolver este problema aplicando uma **Estrutura de Dados do tipo**:

LISTA ENCADEADA

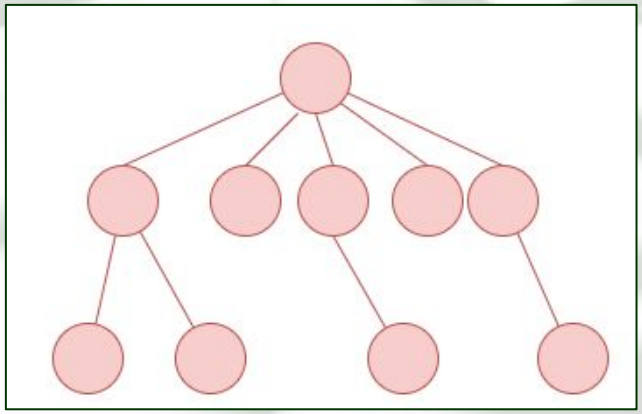


Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

- Imagine resolver este problema aplicando uma **Estrutura de Dados do tipo**:

ÁRVORE



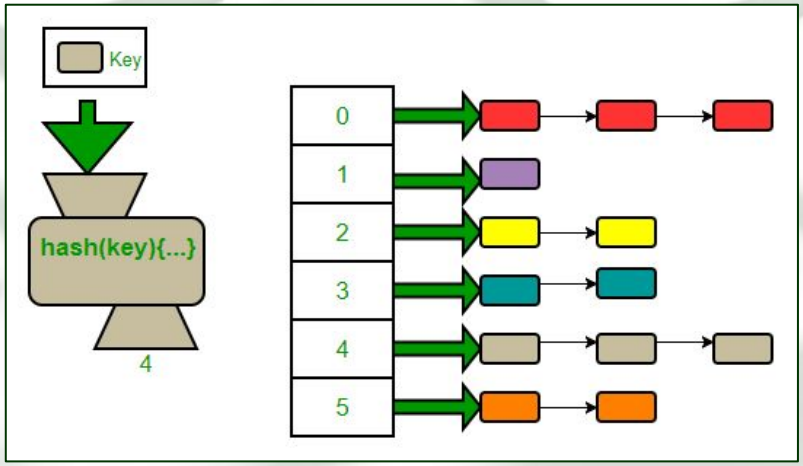
INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

- Imagine resolver este problema aplicando uma **Estrutura de Dados do tipo**:

HASH TABLE



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Problema

- Abstraia a solução para o seguinte problema…
- *Dado um arquivo de texto com inúmeras páginas, desenvolva um programa que liste* ***quantas vezes cada palavra aparece no texto.***

Se você foi capaz de compreender as virtudes, mas também as implicações técnicas e práticas de cada alternativa…

PARABÉNS!!!

Você aprendeu algo sobre Estruturas de Dados

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora Codar!

- Aproveite a Biblioteca de **Listas Genéricas** criada anteriormente, e use-a para criar uma nova Lib.

Hash Table

`HashTable ht = new(HashTable)` <Criar uma nova Hash Table>	`ht->print()` <Listar todos objetos da Hash Table>
`ht->insert(Object)` <Adicionar Objeto na Hash Table>	`ht->remove(key)` <Remover objeto de chave key>
`ht->search(key)` <Retornar Object* de chave key>	`ht->clear()` <Limpar/Excluir todos Objetos da H.T.>

Referências

- Hash Tables
- Tabelas de Espalhamento
- Funções Hash