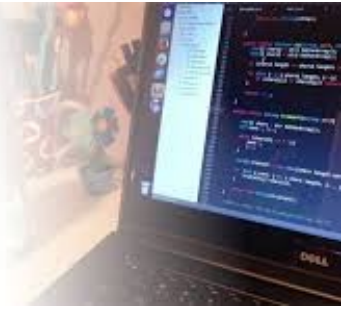
Tecnologia em Análise e Desenvolvimento de Sistemas - TADS

Estrutura de Dados

Prof. ***Luciano Vargas Gonçalves***
E-mail: luciano.goncalves@riogrande.ifrs.edu.br

tads
Análise e Desenvolvimento de Sistemas
INSTITUTO FEDERAL
Rio Grande do Sul
Campus Rio Grande



Sumário

Estrutura de Dados

- Tabela Hashing

Sumário

- **Estrutura de Dados**
  - **Objetivo:**
    - **Armazenar e recuperar informações em estruturas de dados;**
    - **Métodos de recuperação de informação:**
      - **Busca exaustiva (sequencial)**
        - **Percorrer toda a lista, baseado em processo de repetição e teste.**
      - **Tabelas Hash**

Busca exaustiva

- Exemplo de um Vetor de nomes:
  - Inserção desordenada (ordem de chegada)
  - Busca sequencial, comparando chave por chave.

Família	1	2	3	4	5	6
	José Maria	Leila	Artur	Jolinda	Gisela	Alciene

```
Família[1] = "José Maria"
Família[3] = "Artur"
Família[2] = "Leila"
```

Em qual posição está "Alciene" ?

Busca exaustiva

- Exemplo de um Vetor de nomes:
  - Novas inserções geram mais complexidade ao sistema;
  - O sistema precisaria ser ordenado e reordenado a cada inserção (Ou inserções precisam ser ordenadas)

Família	1	2	3	4	5	6
	José Maria	Leila	Artur	Jolinda	Gisela	Alciene

```
Família[1] = "José Maria"
Família[3] = "Artur"
Família[2] = "Leila"
```

Em qual posição está "Alciene" ?

Tabela Hashing (Tabela de Dispersão)

- ***Contextualização***
  - Veremos agora, o método de pesquisa conhecido como ***hashing*** (tabela de dispersão) ou método de cálculo de endereço.
  - Na média dos casos, é possível encontrar a chave (informação) com apenas UMA OPERAÇÃO de LEITURA.
  - ***Realiza a Inserção Ordenada e a Busca mais rápida pelo cálculo do endereço.***

Tabela Hashing

- ***Tabela HASHING***
  - Os índices (posições) dos elementos são definidos pelo cálculo da chave (hash);
  - Utiliza uma função matemática (Hash)
  - Facilita a inserção, ordenação e recuperação (busca);

Tabela Hashing

- **Hash**
  - A ideia central do Hash é **utilizar uma função (ex: matemática)**, aplicada sobre parte da informação (chave), para retornar o índice onde a informação deve ou deveria estar armazenada.
  - **Esta função que mapeia a chave para um índice de um arranjo é chamada de Função de Hashing.**
  - ***A estrutura de dados Hash é comumente chamada de Tabela Hash.***
    - ***Índices e valores mapeados (função)***

Tabela Hashing

- **Exemplo: Mapear CPF em Posições de um Vetor**

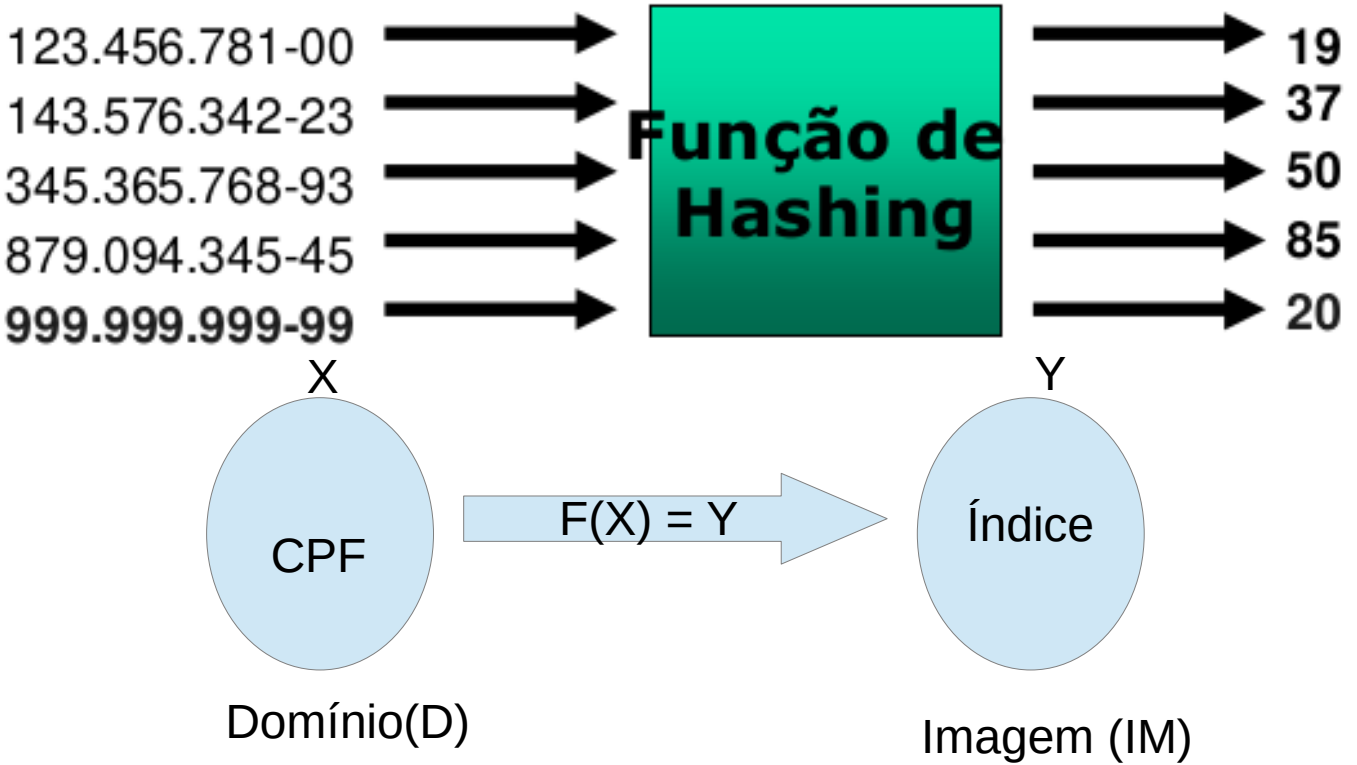


Tabela Hashing

- **Exemplo: Mapear CPF em Posições de um Vetor**

Tabela Hash

19	123.456.781-00; Fausto Silva; Av. Canal. Nº 45.
20	
...	
37	143.576.342-23; Carla Perez; Rua Celso Oliva. Nº 27.
...	
50	345.365.768-93; Gugu Liberato; Av. Atlântica. S/N.
...	
85	879.094.345-45 ; Hebe Camargo; Rua B. Nº 100.
...	

Tabela Hash

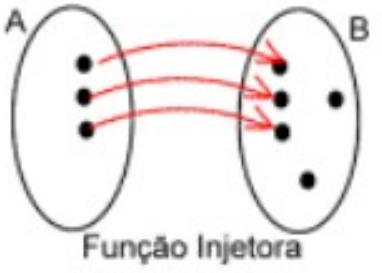
- Em Computação a **Tabela Hash** é uma estrutura de dados especial, que armazena as informações desejadas associando chaves de pesquisa a estas informações.
  - Objetivo: a partir de uma chave, fazer uma busca rápida e obter o valor desejado.
  - A ideia central por trás da construção de uma Tabela Hash é identificar, na chave de busca, quais as partes que são significativas.

Função Hash

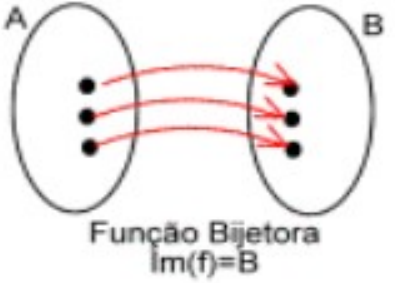
- A **Função de Hashing** é a responsável por gerar **um índice (Hash Code) a partir de uma determinada chave.**
  - O ideal é que a função forneça **índices únicos** para o conjunto das chaves de entrada possíveis.
  - A função de Hashing é extremamente importante, pois ela é responsável **por distribuir as informações pela Tabela Hash.**
  - A implementação da função de Hashing tem influência direta na eficiência das operações sobre o Hash.

Hashing Perfeito

- **Cada Chave gera uma única posição no vetor:**
  - Para quaisquer chaves x e y diferentes e pertencentes A, a função utilizada fornece **saídas diferentes em B;**
  - **Função ótima, observação:**
    - **Depende da Chave;**
    - **Difícil de ser alcançada;**



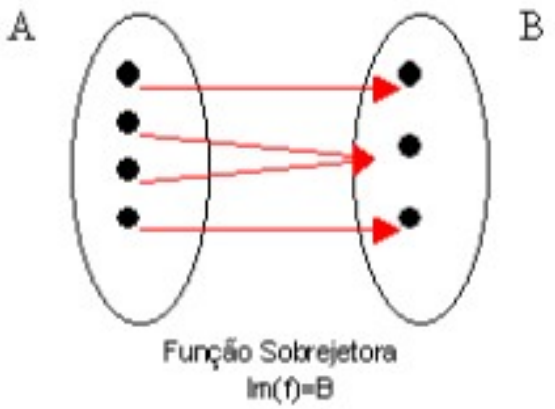
Toda chave A tem apenas um valor correspondente em B



Toda chave A tem apenas um valor correspondente em B, não sobram elementos em B

Hashing ImPerfeito

- **Diferentes chaves podem gera uma único valor de código HASH (Colisão):**
  - Existe chaves x e y diferentes e pertencentes a A, onde a função Hash utilizada fornece **saídas iguais;**
  - **Função Hash comum ou simples:**
    - **Problema complexo;**



Hashing ImPerfeito

- **Exemplo :**
  - Suponha que queiramos armazenar as seguintes chaves: K={C, H, A, V, E, S} em um vetor de P = 7 posições (0..6) conforme a seguinte função:
    - F (k) = k % P
      - K = (código ASCII) e P é o número de elementos, % é a função mod.
        - Forma de mapear 7 elementos diferentes em um vetor de 7 posições
        - Faz a divisão do código ASC pelo tamanho do vetor, e pega o resto da divisão
        - **Tabela ASCII:**
          - **C (67); H (72); A (65); V (86); E (69) e S (83).**

Hashing ImPerfeito

- **Exemplo :**
  - **Código ASCII: C (67); H (72); A (65); V (86); E (69) e S (83).**

chave	K = ord(chave)	$i_1 = h(K)$
C	67	4
H	72	2
A	65	2
V	86	2
E	69	6
S	83	6

Colisão
hash iguais

Tabela HASH para os Elementos de CHAVES

Hashing ImPerfeito

- ***Hashing Perfeito ou quase perfeito:***
  - São encontradas apenas onde a colisão não é tolerável
    - Nas funções de hashing de criptografia;
  - Quando conhecemos previamente o conteúdo a ser armazenado na tabela.
- Nas ***Tabelas Hash comuns a colisão é apenas indesejável,*** diminuindo o desempenho do sistema.
  - Assertiva do sistemas na recuperação de informações;
  - Colisão tolerável;

Tratamento de Colisão

- Por causa das colisões, muitas Tabelas Hash são aliadas com algumas outras estruturas de dados para solucionar os problemas de colisão, são elas:
  - Listas Encadeadas (LSE e LDE);
  - Árvores Balanceadas e dentro da própria tabela.

Tratamento de Colisão

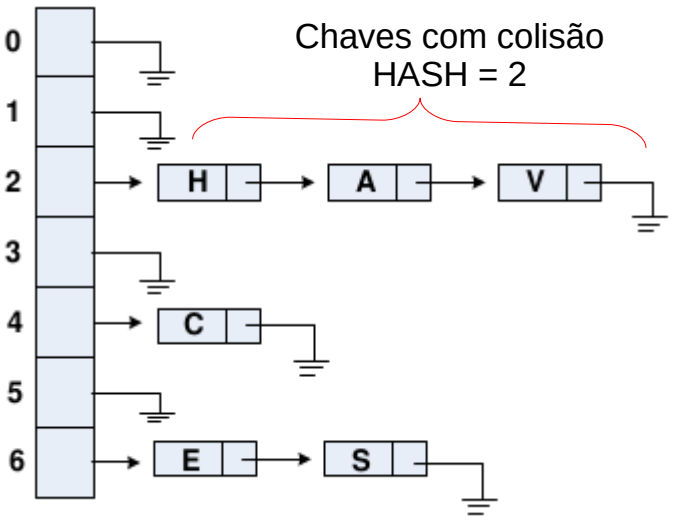
- ***Colisão:***
  - Quando duas ou mais chaves geram o mesmo endereço da Tabela Hash, dizemos que houve uma Colisão.
  - É comum ocorrer colisões, faz parte do processo.
  - Principais causas:
    - em geral o número N de **chaves possíveis é muito maior** que o número de **entradas disponíveis na tabela.**
    - não se pode garantir que as funções de hashing possuam um bom potencial de distribuição (espalhamento).
    - Função matemática ineficiente;

Tratamento de Colisão

- ***Tratamento de Colisão:***
  - Um bom método de resolução de colisões é essencial, não importando a qualidade da função de hashing.
  - Há diversas técnicas de resolução de colisão.
  - As técnicas mais comuns podem ser enquadradas em duas categorias:
    - Encadeamento.
    - Endereçamento Aberto (Rehash);

Tratamento de Colisão com Lista Encadeada

- ***Encadeamento:***
  - Característica Principal: A informação é armazenada em estruturas encadeadas fora da Tabela Hash.
  - A tabela Hash é apenas um Vetor de ponteiros para listas encadeadas com mesmo Hash:
    - Suponha que queiramos armazenar os caracteres que compõem a palavra CHAVES em um vetor de P = 7 posições (0..6) conforme a seguinte função f(k) = k(código ASCII) % P.



Tratamento de Colisão com Deslocamento

- Característica Principal:
  - A informação é armazenada na própria Tabela Hash em outra posição disponível (deslocamento).
  - Uma estratégia de solução:
    - Rehash Linear
      - Repetição da função Hash ***com deslocamento***;
        - Chave + deslocamento

Endereçamento Aberto: Rehash Linear

- Uma das alternativas mais usadas para aplicar Endereçamento Aberto é chamada de ***rehash linear.***
- A posição na tabela é dada por:
  - rh(k) = (k + i) % M,
    - onde M é a qtd de entradas na Tabela Hash
    - k é a chave
    - i é o índice de reaplicação do Hash

Endereçamento Aberto: Rehash Linear

- os caracteres da palavra “CHAVES” serão inseridos em uma tabela hash através do Rehash Linear.
- O número de entradas (M) da Tabela Hash a ser criada é 7.

chave	K = ord(chave)
C	67
H	72
A	65
V	86
E	69
S	83

Endereçamento Aberto: Rehash Linear

- Obs.:
  - h(k) = k mod 7
  - rh (k) = (k+1) mod 7
  - Ocorre um deslocamento do elemento

Repetição do Hash

chave	k=ord(chave)	i_1=h(k)	i_2 =rh(i_1)	i_3 =rh(i_2)	i_4 =rh(i_3)
c	67	4	-	-	-
h	72	2	-	-	-
a	65	2	3	-	-
v	86	2	3	4	5
e	69	6	-	-	-
s	83	6	0	-	-

Endereçamento Aberto: Rehash Linear

- Obs.:
  - h(k) = k mod 7
  - rh (k) = (k+1) mod 7
  - Ocorre um deslocamento

Repetição do Hash

chave	k=ord(chave)	i_1=h(k)	i_2 =rh(i_1)	i_3 =rh(i_2)	i_4 =rh(i_3)
c	67	4	-	-	-
h	72	2	-	-	-
a	65	2	3	-	-
v	86	2	3	4	5
e	69	6	-	-	-
s	83	6	0	-	-

0	s
1	
2	h
3	a
4	c
5	v
6	e

Exemplo:

Implementar um Hash para Nomes:

- Vamos usar a função “soma de código ASC” do Primeiro nome;
- Implementar um vetor de 20 posições;
- Mostrar o vetor após as inserções;
  - Em caso de colisões inserir no vizinho mais a direita (Rehash);

Exemplo:

- Dez nomes para criar um tabela HASH
  - Função Hash = soma dos códigos ASCII%11

```
int main(){

    char tabelaHash [15][50];
    char nomes[][50] = {"Joaquim","Pedro","Carlos","Maria","Ana","Felipe","Tais","Rui","Fernando","Zeca"};
```

```
0 - Joaquim
1 - Pedro
2 - Zeca
3 - Felipe
4 -
5 - Tais
6 - Maria
7 - Carlos
8 - Ana
9 - Rui
10 - Fernando
11 -
12 -
13 -
14 -
```

Tabela Hash

Exemplo:

- Colisões e Rehash
  - Deslocamento para direita

```
Joaquim e 0
Pedro e 0
colisão!! Pedro
Carlos e 7
Maria e 6
Ana e 8
Felipe e 3
Tais e 5
Rui e 7
colisão!! Rui
Fernando e 10
Zeca e 2
```

```
0 - Joaquim
1 - Pedro
2 - Zeca
3 - Felipe
4 -
5 - Tais
6 - Maria
7 - Carlos
8 - Ana
9 - Rui
10 - Fernando
11 -
12 -
13 -
14 -
```

Tabela Hash

Exemplo:

- Recuperar um registro
  - Deslocamento

```
nome =Maria posicao= 6 Tabela=Maria

Rui 7

Deslocamento 1

Deslocamento 2

nome =Rui posicao= 9 Tabela=Rui
```

Não encontrado

Encontrado

```
0 - Joaquim
1 - Pedro
2 - Zeca
3 - Felipe
4 -
5 - Tais
6 - Maria
7 - Carlos
8 - Ana
9 - Rui
10 - Fernando
11 -
12 -
13 -
14 -
```

Tabela Hash

Exemplo

Tabela Hash com vetor e Lista Simplesmente Encadeada

Posição	Marca
0	Ford
1	Fiat
2	
3	Volkwagen
..	Peugeot
10	

Lista Simplesmente Encadeada

0	1	2	3	...	4	5
Ka	Focus	Focus		Belina	Fusion	

Compara as Placas (inserção a direita)

Inserir registro de carros (Marca, Modelo, Placa e Ano)

Fazer a inserção baseada em HASH
e a recuperação dos registros com Hash