

Estrutura de Dados

Prof. Dr. Gedson Faria

Prof.^a Dr.^a Graziela Santos de Araújo

Prof. Dr. Jonathan de Andrade Silva



Módulo 1 - Hash e Heap

Unidade 1 - Tabelas de Dispersão: Hash



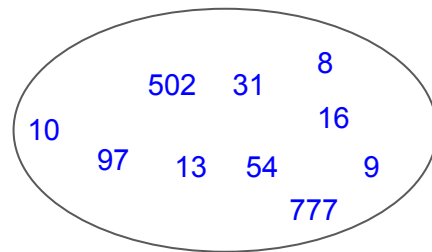
Conceitos e Definições

PARTE 1



Motivação

- Imagine que gostaríamos de armazenar **um conjunto** de valores **inteiros** na memória do computador.
 - Ex: 10 valores inteiros que variam de 0 até 999 (**dado**).
 - Estratégias:
 - Usar vetor com 10 ou 1000 posições (**chave**)? (Algoritmos 1)
 - Lista encadeada? (Algoritmos 2)



Valores inteiros de entrada (**dados**)

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 10 posições (N=10).
 - Vamos considerar a posição como **chave**;
 - $0 \leq \text{chave} \leq 9$ e $0 \leq \text{dado} \leq 999$.

tem o dado 54?



0	1	2	3	4	5	6	7	8	9
10	502	31	97	8	16	777	9	13	54

10 502 31 8
 97 13 54 16
 777 9

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 10 posições.
 - Vamos considerar a posição como **chave**;
 - $0 \leq \text{chave} \leq 9$ e $0 \leq \text{dado} \leq 999$.

tem o dado 54?

0	1	2	3	4	5	6	7	8	9
10	502	31	97	8	16	777	9	13	54

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 10 posições.
 - Vamos considerar a posição como **chave**;
 - $0 \leq \text{chave} \leq 9$ e $0 \leq \text{dado} \leq 999$.

tem o dado 54?

0	1	2	3	4	5	6	7	8	9
10	502	31	97	8	16	777	9	13	54

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 10 posições.
 - Vamos considerar a posição como **chave**;
 - $0 \leq \text{chave} \leq 9$ e $0 \leq \text{dado} \leq 999$.

tem o dado 54?

0	1	2	3	4	5	6	7	8	9
10	502	31	97	8	16	777	9	13	54

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 10 posições.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor: $O(N)$
 - Só funciona para até 10 valores (fixo)!

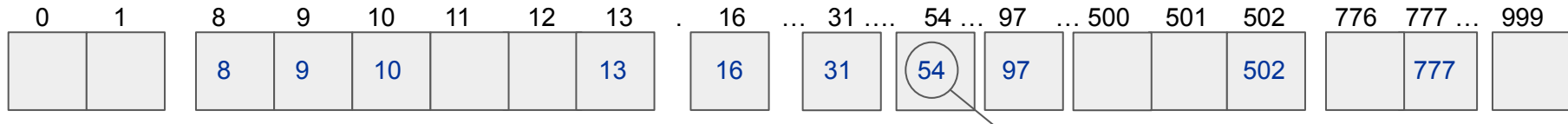
tem o dado 54?

0	1	2	3	4	5	6	7	8	9
10	502	31	97	8	16	777	9	13	54

Estrutura de Dados Lineares

- Estratégia: Usar vetor com 1000 posições (N=1000).
 - $0 \leq \text{chave} \leq 999$ e $0 \leq \text{dado} \leq 999$
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor: $O(1)$, rápido!
 - Porém.... o armazenamento....fica caro \$\$!

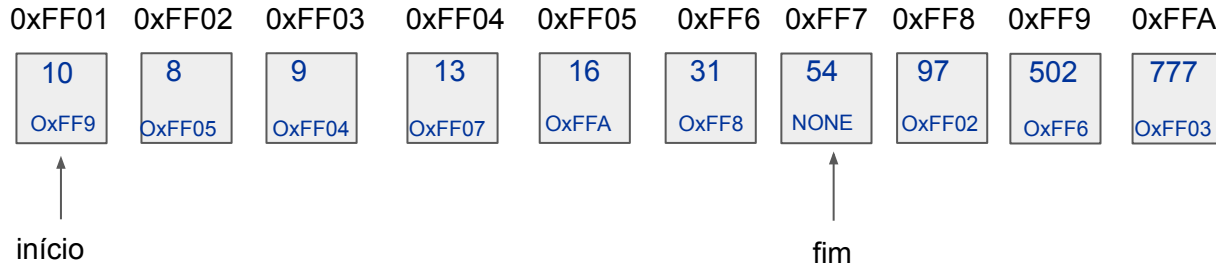
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

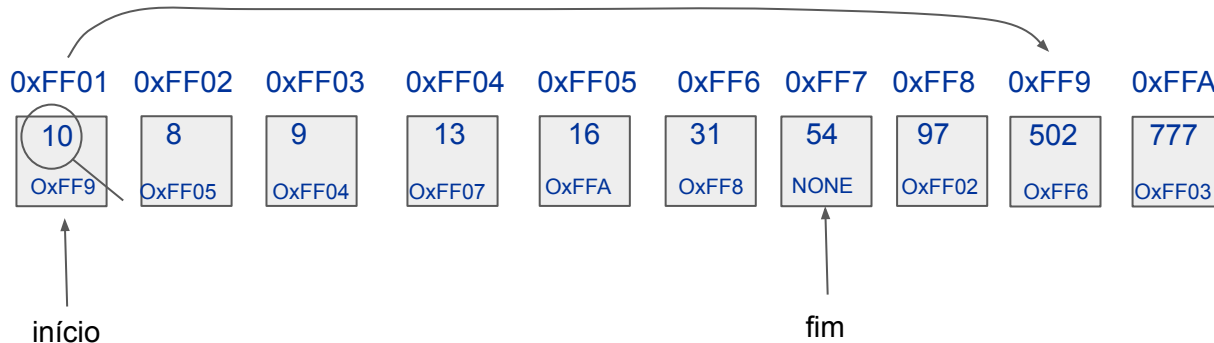
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

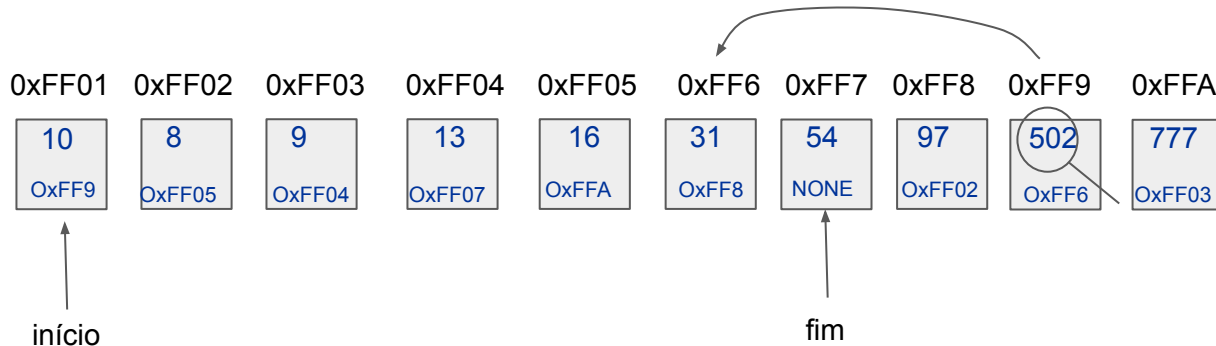
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

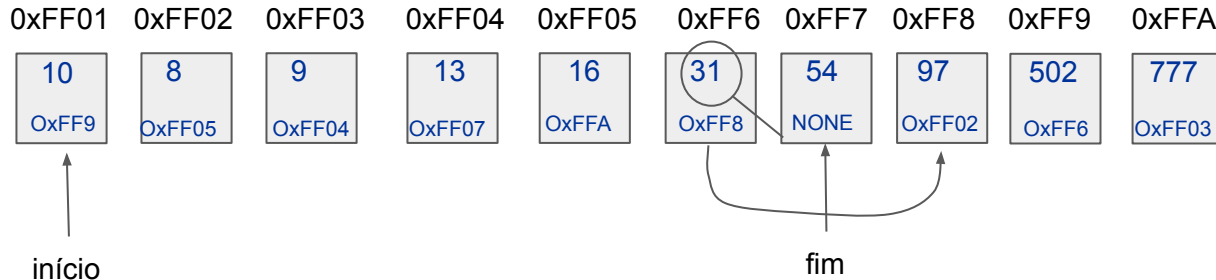
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

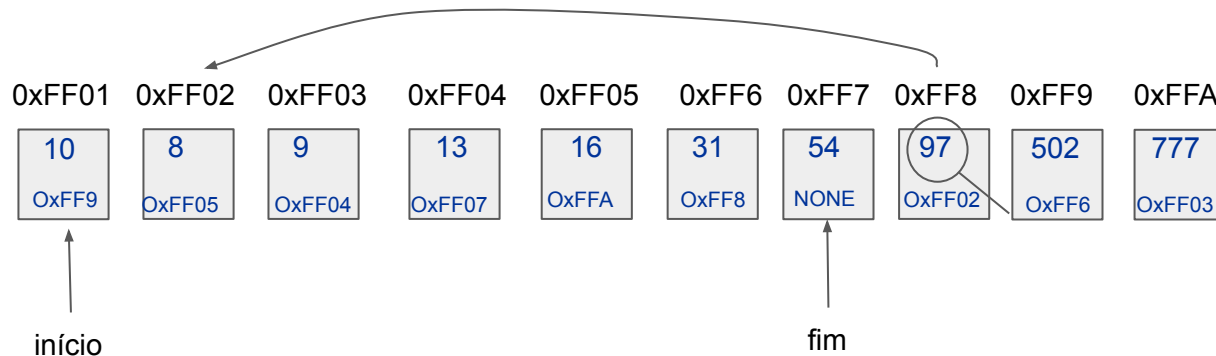
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

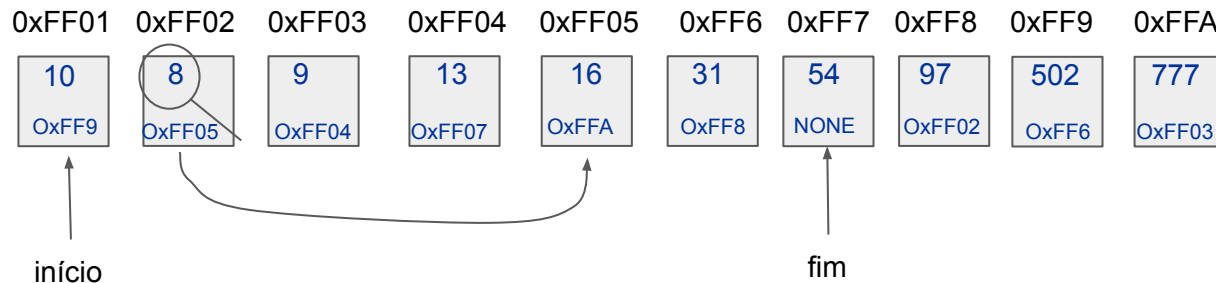
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

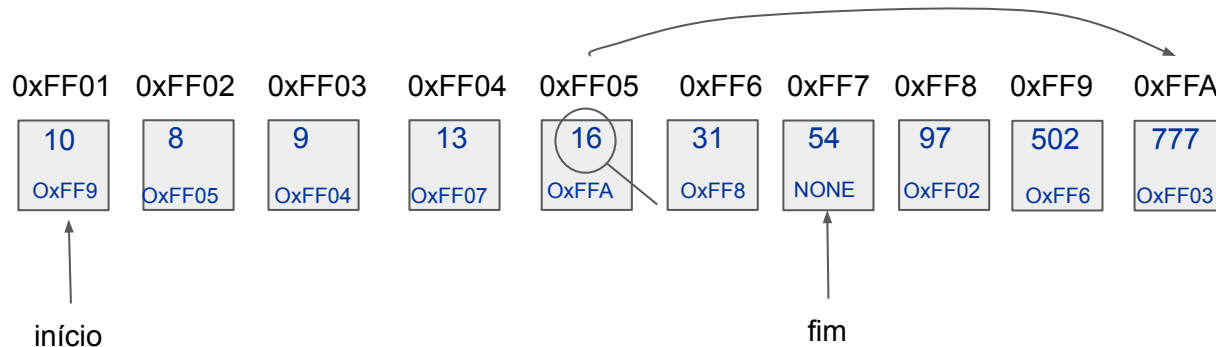
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

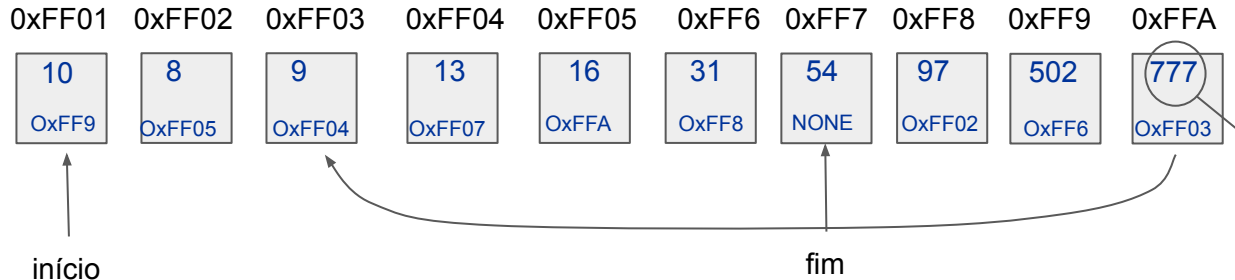
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

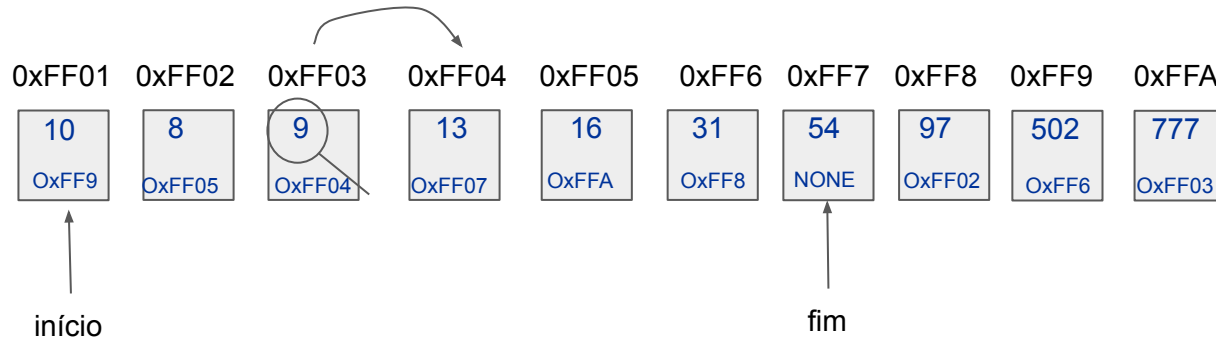
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

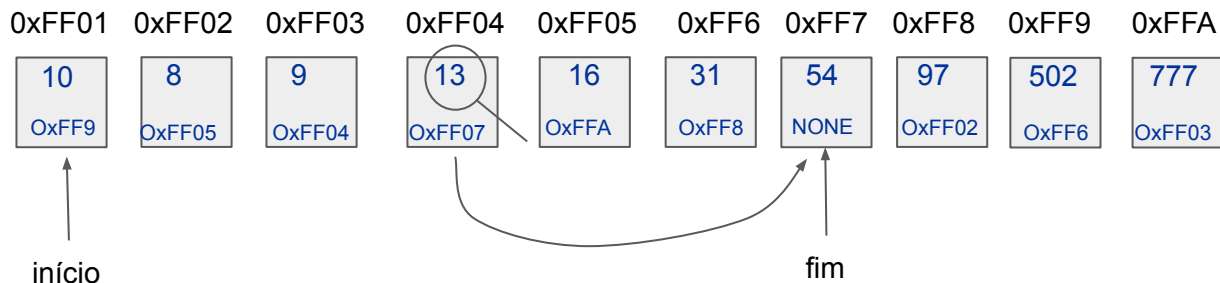
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

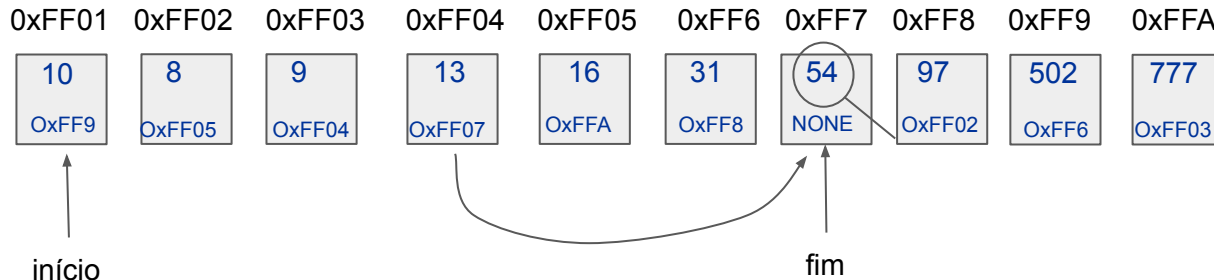
tem o dado 54?



Estrutura de Dados Lineares

- Estratégia: Lista encadeada.
 - Custo da **inserção**: $O(1)$
 - Custo de **buscar** um valor $O(N)$
 - Armazenamento sob demanda!

tem o dado 54?



Estrutura de Dados Lineares

- Resumo:
 - Melhoramos o custo da busca, mas pioramos o custo de armazenamento;
 - Resolvemos o problema de tamanho fixo, porém o custo da busca ainda está $O(N)$;
 - Tem como melhorar o armazenamento e ter custo $O(1)$ na busca?

Tabela de Dispersão (*Hash Table*)

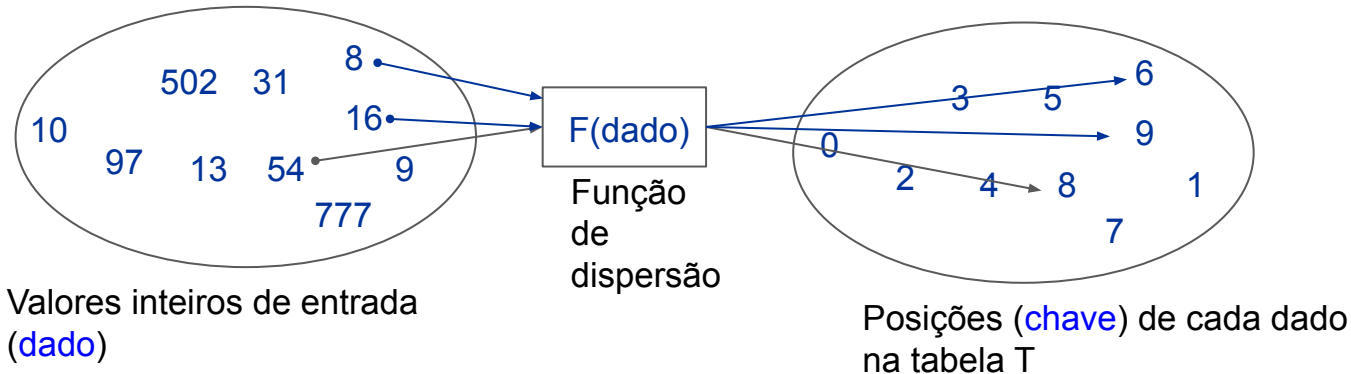
- Encontrar no dado algo significativo para determinar sua posição (chave);
- Método de pesquisa com o objetivo de melhorar o custo da busca de um dado;
- Se baseia na importante característica dos vetores que permitem acesso direto aos dados;
 - Ideia: associar (função de dispersão) cada chave a um valor na tabela (memória).

Tabela de Dispersão (definições)

- Vamos definir então as variáveis da tabela de dispersão:
 - **d** (dado), podendo ser de qualquer tipo: string, float, objeto, matriz, etc.;
 - **k** (chave/posição na tabela) valores inteiros de 0 até **M-1**;
 - **T** (vetor/tabela) e **M** é o tamanho da tabela (idealmente um número primo);
 - **v** (valor/dado) armazenado na tabela;
 - **F** (função de dispersão).

Tabela de Dispersão (*Hash Table*)

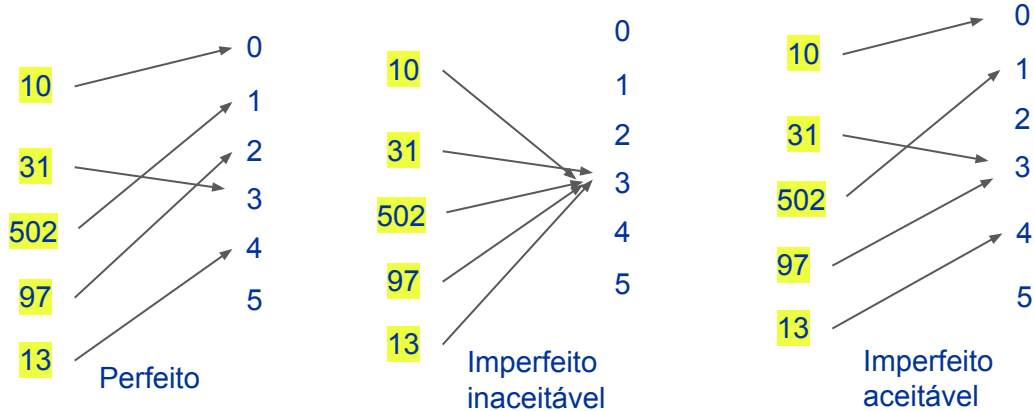
- Ideia: associar (função de dispersão) cada chave a um valor na tabela (memória). $F(d)=k$; $T[k] = d$.



T	k	
10	0	onde está o 54?
9	1	
97	2	$F(54)=8$
	...	$T[8]=54$
54	8	
16	9	

Tabela de Dispersão (*Hash Table*)

- Podemos ter funções perfeitas e imperfeitas;
- Difícil encontrar funções perfeitas em prática.



Função de Dispersão

- O que se espera de uma **boa** função de dispersão?
 - Seja **rápido calcular** a chave para um dado e **rápido acessar** a chave na Tabela, custo $O(1)$;
 - Distribua (disperse) igualmente os dados na tabela;
 - Capture no dado alguma informação que seja relevante para obter uma **chave única**;

Função de Dispersão

- O que seria ruim para uma função de dispersão?
 - Custo alto de cálculo $> O(N)$;
 - Para cada chave ter associado mais de um dado (problema conhecido como colisão).

Funções de Dispersão

- Mas como funciona para strings?
 - Associar um valor para cada caracter da string (Tabela ASCII);
 - Somar os valores de cada caractere para obter o dado numérico.

Caracter	Valor
A	65
L	76
O	79
S	83
U	85

Funções de Dispersão (Strings)

Caracter (c)	Valor
A	65
L	76
O	79
S	83
U	85

onde inserir LUA?

$$F(LUA) = F(76+85+65) = F(226) = 3$$

onde inserir SOL?

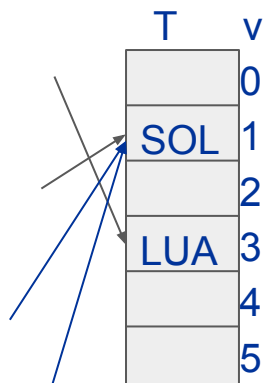
$$F(SOL) = F(83+79+76) = F(238) = 1$$

onde inserir LOS?

$$F(LOS) = F(76+79+83) = F(238) = 1$$

onde inserir OSL?

$$F(OSL) = F(79+83+76) = F(238) = 1$$



Como resolver as colisões?

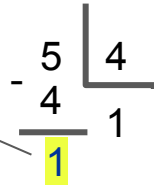
Considerar as posições **p** de cada **c**: $S(p=0)$, $O(p=1)$, $L(p=2)$ e um número primo **a**, $c \cdot a^p$ ($a=3$).

$$\begin{aligned} F(SOL) &= F(83 \cdot 3^0 + 79 \cdot 3^1 + 76 \cdot 3^2) \\ &= F(7371) = 5 \end{aligned}$$

$$\begin{aligned} F(LOS) &= F(76 \cdot 3^0 + 83 \cdot 3^1 + 79 \cdot 3^2) \\ &= F(7381) = 0 \end{aligned}$$

Criando Funções de Dispersão

- A mais clássica função de dispersão é usar a fórmula do resto da divisão (*mod* ou %) também conhecido como método da divisão, dado % **m**, **m** é um número inteiro maior que zero.
 - Ex: o resto da divisão de 5 por 4 é representado como

$$5\%4=1$$


$$F(\text{dado}) = \text{dado} \% \mathbf{m}$$

Qual deve ser o valor de **m** ?

que tal o tamanho da Tabela.

$$F(\text{dado}) = \text{dado} \% M$$

Função de Dispersão (divisão)

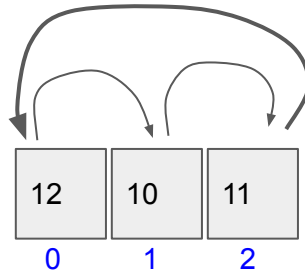
- Dada uma tabela M=7 para inserir 5 valores inteiros de 0 a 99. $F(\text{dado}) = \text{dado} \% 7$.

3	$F(3) = 3\%7 = 3$
14	$F(14) = 14\%7 = 0$
27	$F(27) = 27\%7 = 6$
65	$F(65) = 65\%7 = 2$
99	$F(99) = 99\%7 = 1$
d	

T	k
14	0
99	1
65	2
3	3
	4
	5
27	6

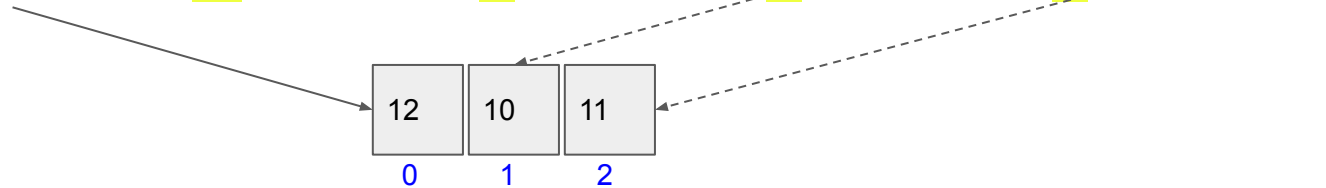
Função de Dispersão (divisão)

- Impacto da escolha de **m**: $F(\text{dado}) = \text{dado} \% m$
- A ideia é simular a caminhada em uma estrutura linear de maneira circular.



Função de Dispersão (divisão)

- Exemplo:
 - Dado o valor $m = 3$ e o valor do dado variando de 10 a 15, temos chaves variando de 0 a 2: $10\%3 = 1$; $11\%3 = 2$; $12\%3 = 0$; $13\%3 = 1$; $14\%3 = 2$; $15\%3 = 0$;...



- E se o valor de m for muito pequeno?
 - Vão ocorrer muitas colisões.

Função de Dispersão (divisão)

- Impacto da escolha de m : $F(\text{dado}) = \text{dado} \% m$
 - Vamos observar a distribuição dos dados (10,...,15) nas chaves:

$m=2$

15	14
13	12
11	10

0 1

$m=3$

15	13	14
12	10	11

0 1 2

$m=4$

		14	15
12	13	10	11

0 1 2 3

Função de Dispersão (divisão)

- Se tenho valores de 10 a 15 e uso um m (par), exemplo $m=2$, quantas colisões ocorrem?
 - Teremos **TODOS** os valores **pares** associados a chave **0** (4 colisões)
- Se tenho valores de 10 a 15 e uso um m (par), exemplo $m=4$, quantas colisões ocorrem?
 - Teremos **TODOS** os valores **pares** distribuídos nas chaves pares (**0 e 2**), 2 colisões.

Função de Dispersão (divisão)

- Se tenho valores de 10 a 15 e uso um m (ímpar/primo), exemplo $m=3$, quantas colisões ocorrem?
 - Teremos os valores pares e ímpares associados a uma mesma chave (3 colisões).

Função de Dispersão (multiplicação)

- Ideia: usar os valores fracionados de uma multiplicação para determinar a posição da chave;
- Utiliza outra constante A , tal que $0 < A < 1$;

Função de Dispersão (multiplicação)

- Fórmula: $F(d) = m \cdot (d \cdot A \% 1)$ ou $F(d) = M \cdot (d \cdot A \% 1)$;
 - O termo $\%1$ serve para pegar apenas a parte fracionária da multiplicação ($d \cdot A$).

($A=0,618$, $M=2$):

2->0

3->1

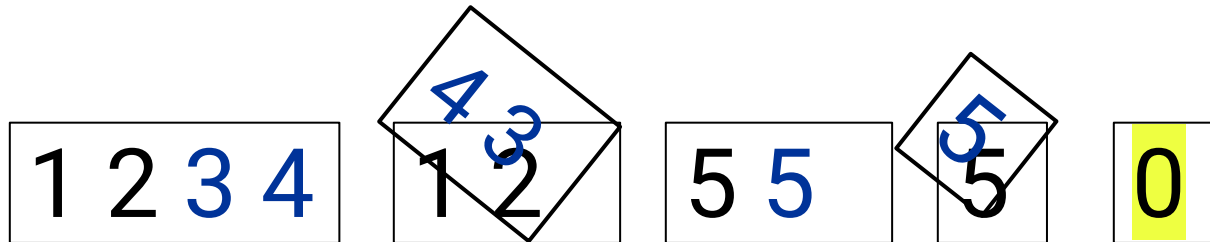
d	d*A	d*A%1	M*d*A%1	k (arredonda)
2	1,236	0,236	0,472	0
3	1,854	0,854	1,708	1

Função de Dispersão (dobra)

- Dobrar o dado como se **dobra um papel**;
- Ideia: quebrar o dado em partes (dobrar e somar) até que se tenha a posição válida (**chave**) para a tabela de tamanho M;

Função de Dispersão (dobra)

- Exemplo: encontrar a posição para o dado 1234 sendo $M=2$ (tabela com 2 posições).
 - 1234 \rightarrow unidade(1+4) unidade(2+3) = 55 $> 2 \rightarrow$ unidade(5+5) = 0 < 2



Função de Dispersão

- O desempenho é medido pelo número de colisões;
 - Pode ser estimado pelo fator de carga (**fc**): quantidade de chaves / M ;
- Em geral, as colisões ocorrem quando a distribuição das chaves é desigual ou quando a tabela tem tamanho pequeno;
- Estratégias para lidar com as colisões se fazem necessárias, vamos conhecer algumas delas.

Referências

- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro, RJ: LTC, 2010. ISBN 9788521629955.
- CORMEN, Thomas. **Algoritmos: teoria e prática**. Rio de Janeiro: GEN LTC, 2013. ISBN 9788595158092.

Tratamento de Colisões

PARTE 2



Tratamento de Colisões

- Quando dois ou mais valores estão associados a uma chave;
- Vamos conhecer duas estratégias:
 1. Sondagem por endereçamento aberto:
 - Tentar encontrar na tabela posições vizinhas disponíveis.

Tratamento de Colisões

2. Sondagem por endereçamento encadeado:
 - Utilizar outras estruturas de dados para armazenar mais de um valor em uma chave (posição da tabela) com colisão.

Endereçamento aberto

- Distribuir os valores diretamente na tabela;
- Encontrar (**sondagem**) em posições vizinhas daquela que houve colisão alguma posição disponível.

Endereçamento aberto

- Algumas formas de fazer a sondagem:
 - Sondagem linear
 - Caminhar sequencialmente por toda a tabela;
 - Sondagem quadrática
 - Caminhar em saltos (quadráticos) por toda a tabela;
 - Sondagem dupla
 - Utiliza uma segunda função de dispersão.

Sondagem Linear

- Caso ocorra a colisão em alguma posição, incrementar sequencialmente (linear) a posição até encontrar uma posição não ocupada.

Sondagem Linear

- A ideia é que valores parecidos tendem a ficarem próximos (agrupamento);

12	$F(12) = 12\%5 = 2$	
13	$F(13) = 13\%5 = 3$	
14	$F(14) = 14\%5 = 4$	
17	$F(17) = 17\%5 = 2$	
d	$F(17) = (17+1)\%5 = 3$	17
	$F(17) = (17+2)\%5 = 4$	17
	$F(17) = (17+3)\%5 = 0$	17

T	k
17	0
	1
12	2
13	3
14	4

E agora para buscar o 17?

$F(17) = 17\%5 = 2$, não tem o 17!

Será que não tem? ou teve colisão?

Vamos percorrer toda a tabela!

Custo da busca quando há colisão

$O(M)$

Sondagem Linear (análise)

- Quando há uma colisão, temos que percorrer (no pior caso) a tabela inteira para encontrar uma posição disponível.

Sondagem Linear (análise)

- À medida que o fator de carga ($fc=N/M$) tende a 1, as colisões aumentam.

12
13
14
17

d

$$F(12) = 12\%5 = 2; \quad fc=0/5=0$$

$$F(13) = 13\%5 = 3; \quad fc=1/5=0.2$$

$$F(14) = 14\%5 = 4; \quad fc=2/5=0.4$$

$$F(17) = 17\%5 = 2; \quad fc=3/5=0.6$$

T k	
17	0
	1
12	2
13	3
14	4

$$F(17) = (17+1)\%5 = 3$$

$$F(17) = (17+2)\%5 = 4$$

$$F(17) = (17+3)\%5 = 0;$$

$$fc=4/5=0.8$$

Sondagem Linear (análise)

- Visualizando exemplo em [VISUALGO](#);
- Vamos:
 - Criar uma tabela $M=5$ vazia `Create(5,0)` com LP (Linear Probing);
 - Aplicar inserção (`Insert(v)`), busca (`Search(v)`) e remoção (`Remove(v)`) de um ou conjunto de valores v ;
 - Valores em $v = [12, 13, 14, \text{e } 17]$.

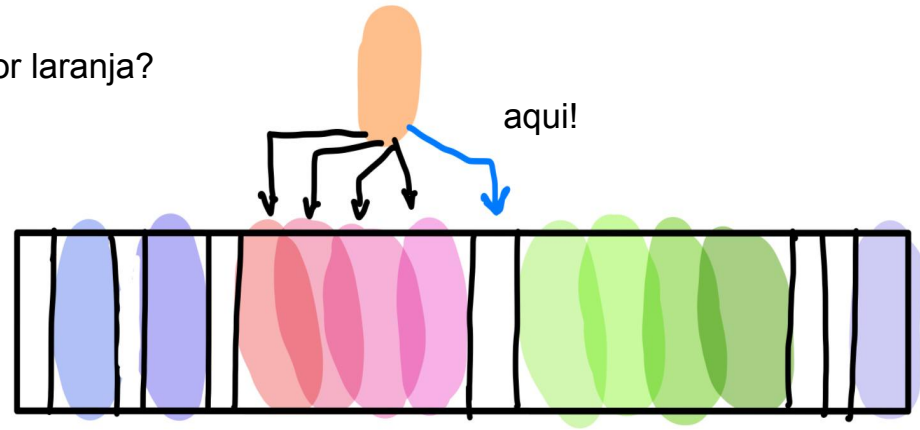
Sondagem Linear (análise)

- A sondagem permite formar "agrupamentos" colocando valores parecidos em posições próximas;

Sondagem Linear (análise)

- Esses agrupamentos, se muito grandes, tendem a ocupar muitos espaços na tabela necessitando que a sondagem caminhe toda a tabela para encontrar uma posição disponível.

Onde inserir a cor laranja?



Sondagem Quadrática

- Talvez possamos acelerar a sondagem linear se percorrermos em saltos largos (quadráticos);
- A ideia da sondagem quadrática é quebrar os “agrupamentos grandes” de valores parecidos espalhando-os pela tabela;

Sondagem Quadrática

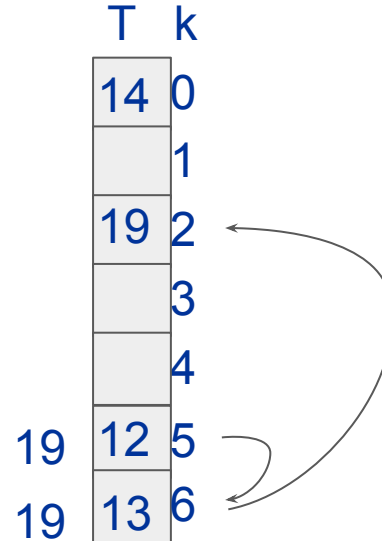
- Ao invés de incrementar **uma unidade** a cada passo, como na sondagem linear, incrementa com o **quadrado** do passo;
 - $(\text{dado} + \text{passo}) \% M \rightarrow (\text{dado} + \text{passo}^2) \% M$
- O tamanho da tabela **não pode ser par**, pois acarreta numa má distribuição dos valores pares.

Sondagem Quadrática

- Exemplo para os dados 12, 13, 14 e 19 e $M=7$

12	$F(12) = 12\%7 = 5$
13	$F(13) = 13\%7 = 6$
14	$F(14) = 14\%7 = 0$
19	$F(19) = 19\%7 = 5;$
	$F(19) = (19+1^2)\%7 = 6$
d	$F(19) = (19+2^2)\%7 = 2;$

	T	k
	14	0
		1
	19	2
		3
		4
19	12	5
19	13	6



Sondagem Dupla

- Utilizar uma segunda função de dispersão (F_2) para resolver as colisões
 - $F(d) = (F_1(d) + \text{passo} * F_2(d)) \% M$

Sondagem Dupla

- Exemplo: $F_1(d) = d \% 7$ e $F_2(d) = 1 + d \% 5$, passo=0, M=7:

$$F(12) = (12 \% 7 + 0 * (1 + 12 \% 5)) \% 7 = 5$$

$$F(19) = (19 \% 7 + 1 * (1 + 19 \% 5)) \% 7 = 3$$

$$F(13) = (13 \% 7 + 0 * (1 + 13 \% 5)) \% 7 = 6$$

$$F(14) = (14 \% 7 + 0 * (1 + 14 \% 5)) \% 7 = 0$$

$$F(19) = (19 \% 7 + 0 * (1 + 19 \% 5)) \% 7 = 5$$

0	1	2	3	4	5	6
14			19		12	13

Sondagem Dupla

- Na sondagem linear ou quadrática F_2 pode ser considerada uma **leve** alteração de F_1 que considera o incremento do passo. Em algumas situações pode levar a ambos F_2 e F_1 produzirem valores idênticos dependendo do padrão das chaves.

Sondagem Dupla

- A sondagem dupla é mais robusta que a linear e a quadrática, pelo fato de F_2 ser uma função com padrão diferente de distribuição das chaves do que F_1 . Ex: F_1 lida com valores pares e F_2 com valores ímpares.

Sondagem Dupla

- Comparando o número de colisões entre os 3 métodos de sondagem para os dados: [12,13,14,16 e 19] na tabela com $M=7$, teríamos colisão com $d=19$:
 - Sondagem linear: 3 colisões para encontrar a chave $k=0$;
 - Sondagem quadrática: muitas colisões, pois, a tentativa quadrática só vai gerar $k=\{0,2,5,6\}$ (posições ocupadas);
 - Sondagem dupla: 1 colisão e encontra $k=3$.

Resumo

- Em geral os métodos de sondagem tem, no pior caso, a mesma complexidade nas operações de inserção, $O(1)$, e busca, $O(M)$.
 - Entretanto, a frequência de colisões é menor na sondagem dupla.

Resumo

- Podemos escrever as funções como $(0 \leq \text{passo} \leq M-1)$:

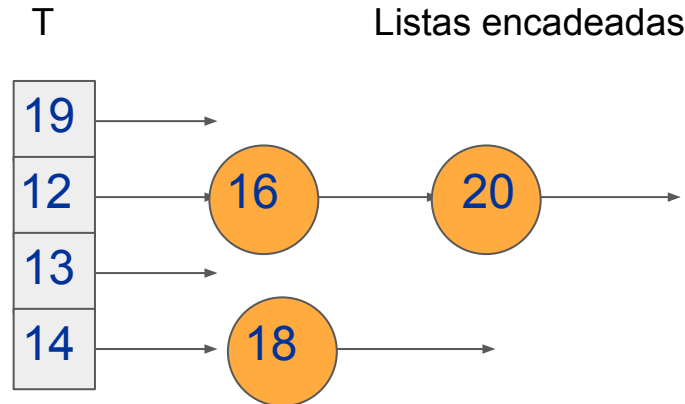
Linear	$F(d) = (F_1(d) + \text{passo} * 1) \% M$
Quadrático	$F(d) = (F_1(d) + \text{passo} * \text{passo}) \% M$
Duplo	$F(d) = (F_1(d) + \text{passo} * F_2(d)) \% M$

Endereçamento encadeado

- Utilizar uma segunda estrutura de dados para armazenar as colisões em uma posição da tabela;

Endereçamento encadeado

- A ideia é armazenar dados/valores “parecidos” em uma estrutura de dados, por exemplo, uma lista encadeada, na posição de colisão.



Endereçamento encadeado

- O custo de inserir um elemento, tanto na Tabela quanto na Lista, continua com custo $O(1)$.

Endereçamento encadeado

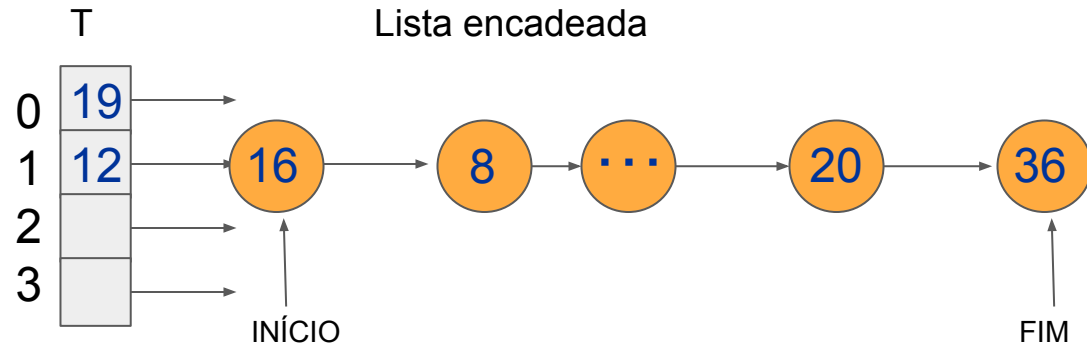
- Porém, o custo da busca na Tabela e na Lista são diferentes:
 - Na tabela $O(1)$ e na Lista $O(N)$, (N =tamanho da lista) no pior caso.

Onde está o 20?

Na tabela:

$F(20) = 1$, custo $O(1)$

Na lista: percorrer de INÍCIO até FIM, custo $O(N)$



Endereçamento encadeado

- Mais simples de lidar com colisões ao invés de criar funções de dispersão mais elaboradas;
- Ideal para quando não se sabe como será a distribuição das chaves na tabela (dinâmica);
- Tamanho mais flexível do que no endereçamento aberto que considera o tamanho fixo da tabela.

Endereçamento encadeado

- Porém:
 - Custo adicional de memória (Tabela + Listas)
 - Lista pode ficar muito grande
- Tem como reduzir o custo da busca na estrutura de dados?
 - Usar estruturas de dados mais sofisticadas ao invés de uma lista encadeada, por exemplo **árvore binária** (veremos nessa disciplina).

Referências

- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro, RJ: LTC, 2010. ISBN 9788521629955.
- CORMEN, Thomas. **Algoritmos: teoria e prática**. Rio de Janeiro: GEN LTC, 2013. ISBN 9788595158092.

Implementação em Python

PARTE 3



Tabela de Dispersão (dicionário)

- Podemos representar a tabela por meio da estrutura de dados em Python, conhecida como **dicionário**;
- Precisamos do par: <chave:valor> ou <k:v>, ambos k e v podem ser de qualquer tipo (int,string...);
 - Exemplo, tabela com M=3:
 - tabela = {k₁:v₁, k₂:v₂, k₃:v₃}
 - tabela = {'A':0, 'B':0, 'C':0}
 - tabela['B']=1
 - tabela = dict(k₁=v₁, k₂=v₂, k₃=v₃)
 - k deve ter padrão de nome de variável.

k	T	k	T
A	0	A	0
B	0	B	1
C	0	C	0

Tabela de Dispersão (dicionário)

- Vamos criar uma tabela com $M=5$ para $d=[12,13,14,17]$;
 - Criar a tabela e inicializar as posições com vazio:
 - (o que seria vazio? 0, -1, None,...); vamos usar None.
 - Criar a função de dispersão:
 - vamos usar o método da divisão: $F(\text{dado}) = \text{dado} \% M$;
 - Para cada dado, aplicar F para encontrar a chave e inserir o valor/dado na tabela.
 - Antes de inserir deve-se identificar a colisão, mas como?
 - Se $\text{tabela}[\text{chave}] == \text{? None}$

Criar a Tabela de Dispersão

- Vamos criar uma tabela com $M=5$;

```
1 tabela={} # Tabela sem nenhum campo
2 M=5      # Tamanho da tabela
3 for chave in range(M): #Associando chave aos valores de 0...M-1
4     tabela[chave]=None #Atribuindo None em cada posição da tabela
5
6 print(tabela) #Imprimindo a tabela toda
```

Global frame		dict	
tabela		0	None
M	5	1	None
chave	4	2	None
		3	None
		4	None

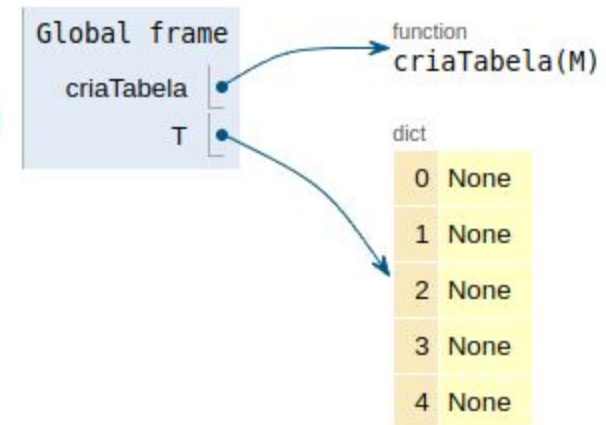
[Código](#)

Criar a Tabela de Dispersão

- Vamos colocar o código de criação da tabela dentro de uma função chamada **criaTabela(M)**;

```

1 def criaTabela(M): #criando a função criaTabela com parâmetro M
2     tabela={} # Tabela sem nenhum campo
3     for chave in range(M):
4         tabela[chave]=None
5     return tabela #Retornando a tabela criada
6
7 T = criaTabela(5) #chamando criaTabela(M=5)
    
```

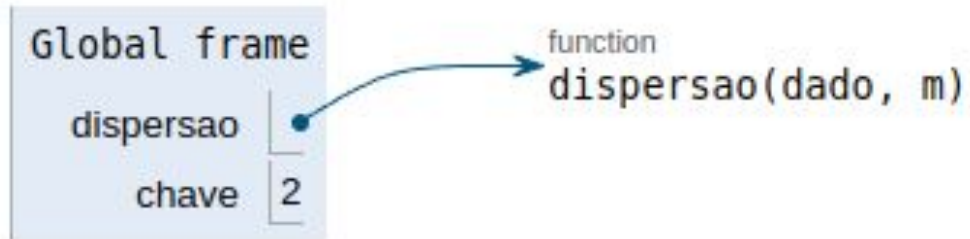


[Código](#)

Criar a Função Dispersão

- Vamos criar uma função chamada `dispersao(dado,m)`;
 - função de dispersão será o método da divisão.

```
1 def dispersao(dado,m): #Criando a função dispersão
2     k = dado % m      #Calculando a chave
3     return k          #Retornando a chave
4
5 chave = dispersao(12,5)#Chamando a função
```



[Código](#)

Verificar colisão e inserir na tabela

- Vamos criar uma função chamada `inserir(dado, chave, tabela)`;
 - temos que verificar se está “vazia” `tabela[chave]==None`;
 - caso contrário imprimir que houve colisão.

```

1 def inserir(dado,chave,tabela):#Função para inserir o dado na tabela[chave]
2     if tabela[chave]==None: #Verificando se está desocupada
3         tabela[chave]=dado  #Inserindo na tabela
4     else:
5         print('colisão d=',dado,' k=',chave) #Imprimindo que houve colisão

```

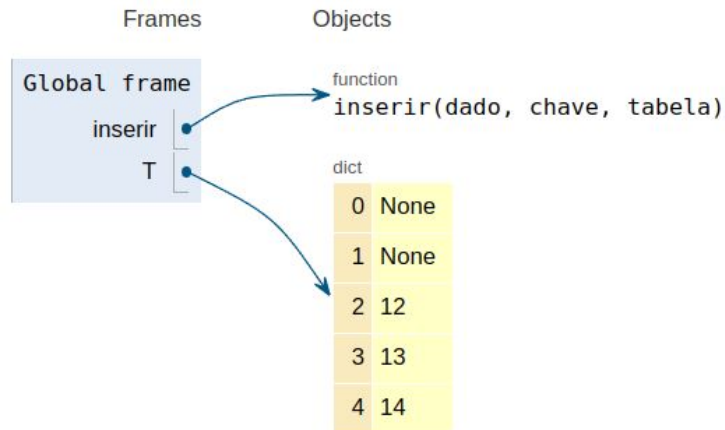
Verificar colisão e inserir na tabela

```

1 def inserir(dado,chave,tabela):#Função para inserir o dado na tabela[chave]
2     if tabela[chave]==None: #Verificando se está desocupada
3         tabela[chave]=dado #Inserindo na tabela
4     else:
5         print('colisão d=',dado,' k=',chave) #Imprimindo que houve colisão
6 T={0:None,1:None,2:12,3:13,4:14} #Tabela com valores já inseridos
7 inserir(17,2,T) #Inserindo o dado 17 (colisão)

```

colisão d= 17 k= 2



[Código](#)

Verificar colisão e inserir na tabela

- Agora é só combinar os códigos e ter a implementação completa da inserção na tabela de dispersão com o método da divisão.
- Podemos até melhorar a organização do código utilizando orientação a objeto para **criar uma classe** associada à tabela de dispersão e definir as funções para criar, inserir, buscar, remover e imprimir a tabela de dispersão.

Implementação com classe

```

1  class HashTable:
2      def __init__(self,M):
3          #função de inicialização para criar a tabela de tamanho M
4          self.M=M
5          self.T={}
6          #Criar a tabela...
7          pass
8
9      def dispersao(self,d):
10         #Criando a função dispersão
11         #...
12         pass
13
14     def inserir(self,d):
15         #Criando a função de inserção...
16         #....
17         self.dispersao(d)
18         #....
19         pass
20
21 tabela=HashTable(5)
22 tabela.inserir(10)

```

[Código](#)

Tratamento de Colisão (Endereçamento Aberto)

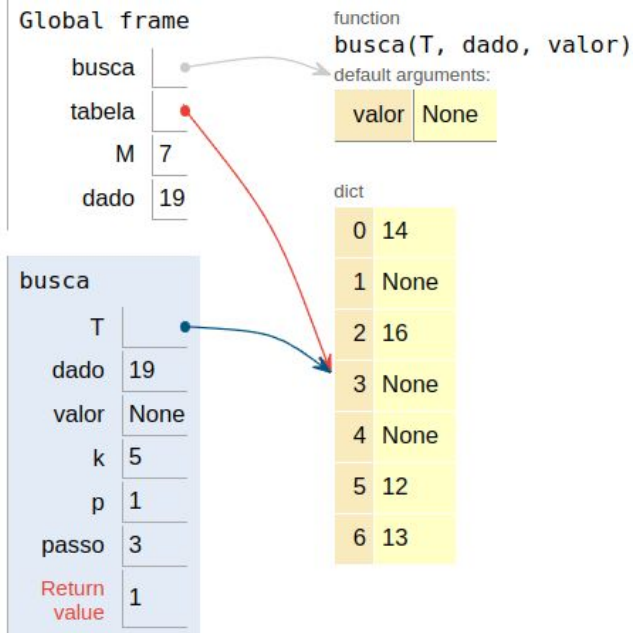
- Com as sondagens vamos ter que **percorrer** a tabela **até** encontrar uma posição disponível (“vazia”);
- Aplicado em todas as operações (ex: inserção, remoção e busca);
- Precisamos definir a estratégia de sondagem e ajustar:
 - A função de dispersão;
 - A função de inserção (busca ou remoção) que na ocorrência de colisão tem que continuar **buscando** uma posição vazia (inserir) ou o dado (busca e remoção).

Sondagem Linear (inserção)

```

1 def busca(T,dado,valor=None):
2     k = dado % M #calculando a chave (resto da divisão)
3     p=k
4     passo=0
5     while tabela[p]!=valor and passo<M:
6         #incremento a tentativa (passo)
7         passo=passo+1
8         #Recalculando a chave para o passo
9         p=(k+passo*1)%M
10    if tabela[p]==valor:
11        return p
12    else:
13        return None
14
15    #Tabela com 3 valores já inseridos 12,13,14 e 16
16    tabela={0:14,1:None,2:16,3:None,4:None,5:12,6:13}
17    #Inicializando M
18    M=len(tabela)
19    #Considerando inserir o dado=19
20    dado=19
21
22    p=busca(tabela,dado) #buscar um valor = (dado ou vazio)
23    #Se terminei o laço e a posição está disponível...
24    if p!=None:
25        tabela[p]=dado #insere dado na tabela[chave]

```



[Código](#)

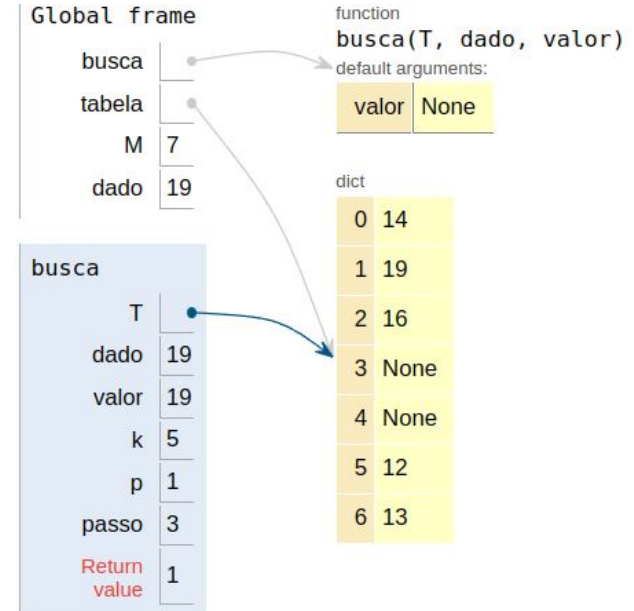
Sondagem Linear (remoção)

```

1 def busca(T,dado,valor=None):
2     k = dado % M #calculando a chave (resto da divisão)
3     p=k
4     passo=0
5     while tabela[p]!=valor and passo<M:
6         #incremento a tentativa (passo)
7         passo=passo+1
8         #Recalculando a chave para o passo
9         p=(k+passo*1)%M
10    if tabela[p]==valor:
11        return p
12    else:
13        return None
14
15    #Tabela com 5 valores já inseridos 12,13,14,16 e 19
16    tabela={0:14,1:19,2:16,3:None,4:None,5:12,6:13}
17    #Inicializando M
18    M=len(tabela)
19    #Considerando remover o dado=19
20    dado=19
21
22    p=busca(tabela,dado,dado) #buscar um valor = (dado ou vazio)
23    #Se terminei o laço e a posição está disponível...
24    if p!=None:
25        tabela[p]=None #Remover o dado na tabela[chave]

```

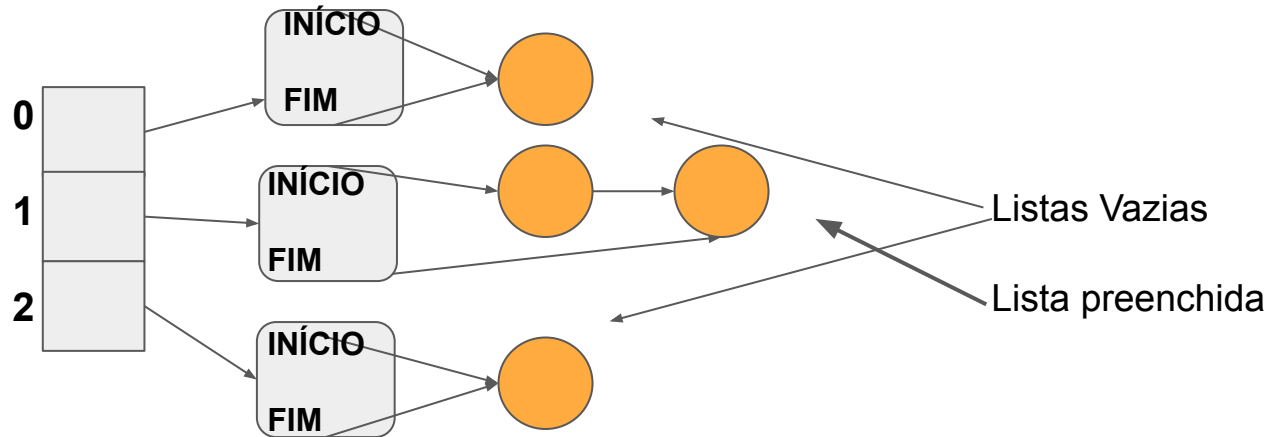
Ajustes



[Código](#)

Tratamento de Colisão (Endereçamento Encadeado)

- Vamos implementar, além da tabela, uma segunda estrutura de dados (Lista ligada):
 - 2 ponteiros (início e fim) e nó cabeça;



Tratamento de Colisão (Endereçamento Encadeado)

```

1 class no:#Criando a classe que representa um nó da lista
2     def __init__(self,dado):
3         self.d=dado #Campo para armazenar o valor
4         self.prox=None #Campo para armazenar o endereço do próximo nó
5
6 class listaL:# Classe que cria a Lista Ligada
7     def __init__(self): #Inicializando a classe
8         self.inicio = no(None)
9         self.fim = self.inicio
10        self.N = 0
11
12    def inserir(self,dado):#Inserindo na Lista Ligada
13        d = no(dado)
14        if self.inicio.prox==None: #Se tiver só o nó inicial (cabeça)
15            self.inicio.prox = d
16        else: #Se tiver mais nós acessar o fim
17            self.fim.prox = d
18        self.fim = d

```

[Código](#)

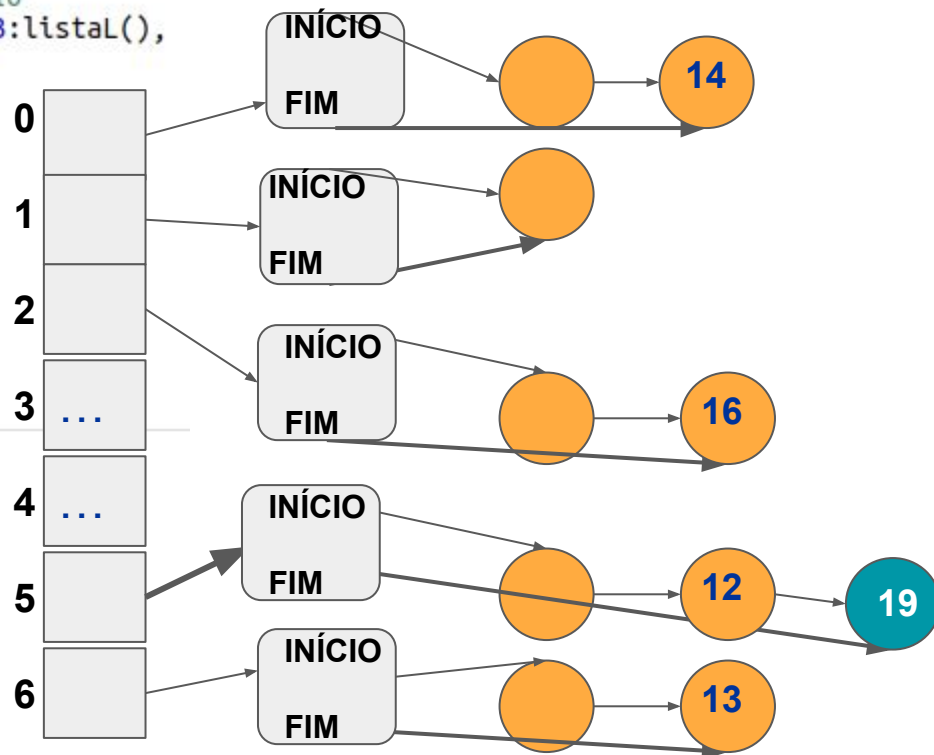
Tratamento de Colisão (Endereçamento Encadeado)

```
21 #Tabela com 4 valores já inseridos 12,13,14,16
22 tabela={0:listaL(), 1:listaL(), 2:listaL(), 3:listaL(),
23 4:listaL(), 5:listaL(), 6:listaL()}
24
25 tabela[0].inserir(14)
26 tabela[2].inserir(16)
27 tabela[5].inserir(12)
28 tabela[6].inserir(13)
29
30 M=len(tabela) #Inicializando M
31 dado=19 #Considerando inserir o dado=19
32
33 k=dado % M #
34 tabela[k].inserir(dado)
```

19

k=5

Código



Resumo (implementações)

- Nas técnicas de sondagem temos que percorrer a tabela para, por exemplo, buscar um valor, custo $O(M)$;
- Já na técnica encadeada temos que percorrer uma lista para buscar um valor, $O(L)$, onde L é o tamanho da lista;
- O ideal é que L tenha um tamanho médio L/N , indicando uma distribuição equilibrada das N chaves nas listas;
- Se a maior lista na tabela tem tamanho $L = M$, temos que o custo da busca fica, no pior caso, igual às técnicas de sondagem, ou seja, $O(L)=O(M)$.

Referências

- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de dados e seus algoritmos**. 3. ed. Rio de Janeiro, RJ: LTC, 2010. ISBN 9788521629955.
- CORMEN, Thomas. **Algoritmos: teoria e prática**. Rio de Janeiro: GEN LTC, 2013. ISBN 9788595158092.

Licenciamento



Respeitadas as formas de citação formal de autores de acordo com as normas da ABNT NBR 6023 (2018), a não ser que esteja indicado de outra forma, todo material desta apresentação está licenciado sob uma [Licença Creative Commons - Atribuição 4.0 Internacional](#).