Tabela Hash

Estrutura de Dados Avançada — QXD0015

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

1º semestre/2024

Introdução

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Contexto: Temos um conjunto de objetos com chaves associadas e possivelmente muitos outros dados. Precisamos realizar buscas de forma muito rápida por esses objetos a partir dos valores das chaves.

Introdução

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Contexto: Temos um conjunto de objetos com chaves associadas e possivelmente muitos outros dados. Precisamos realizar buscas de forma muito rápida por esses objetos a partir dos valores das chaves.

- Tipos abstratos de dados que fornecem apenas as operações de inserção, busca e remoção são chamados de **dicionários** ou **mapas**.

Introdução

- Aplicação: Queremos carregar um dicionário da língua portuguesa na memória do computador.

Introdução

- Aplicação: Queremos carregar um dicionário da língua portuguesa na memória do computador.
    - Operações de inserção e busca serão frequentemente realizadas.

Introdução

- Aplicação: Queremos carregar um dicionário da língua portuguesa na memória do computador.
  - Operações de inserção e busca serão frequentemente realizadas.
  - Operações de remoção podem vir a ser realizadas e gostaríamos que fossem eficientes.

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- `paralelepípedo:` `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- **paralelepípedo:** `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$

Introdução

Universidade Federal do Ceará
Campus Quixadá

Cada entrada do dicionário é composta por um par (verbete, descrição).

- **paralelepípedo:** `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$
  - inserir uma nova palavra leva $O(1)$

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- **paralelepípedo:** `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$
  - inserir uma nova palavra leva $O(1)$
- Vetor ordenado - busca em $O(\lg n)$

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- `paralelepípedo:` `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$
  - inserir uma nova palavra leva $O(1)$
- Vetor ordenado - busca em $O(\lg n)$
  - inserir/remover uma nova palavra leva $O(n)$

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- **paralelepípedo:** `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$
  - inserir uma nova palavra leva $O(1)$
- Vetor ordenado - busca em $O(\lg n)$
  - inserir/remover uma nova palavra leva $O(n)$
- Árvore balanceada - busca/inserção/remoção em $O(\lg n)$

Introdução

Cada entrada do dicionário é composta por um par (verbete, descrição).

- **paralelepípedo:** `Hexaedro cujas faces, opostas e paralelas entre si, são paralelogramos.`

Primeiras opções:

- Vetor não ordenado - busca/remoção em $O(n)$
  - inserir uma nova palavra leva $O(1)$
- Vetor ordenado - busca em $O(\lg n)$
  - inserir/remover uma nova palavra leva $O(n)$
- Árvore balanceada - busca/inserção/remoção em $O(\lg n)$

Vamos ver outras possibilidades

Caso Simples: Tabela de acesso direto

Tabela de acesso direto

- Suponha que uma aplicação precisa de uma estrutura de dados na qual cada elemento tem uma chave com valor no conjunto $U = \{0, 1, \ldots, m-1\}$, em que m não é muito grande.
  - Supomos que não existem chaves repetidas.

Tabela de acesso direto

- Suponha que uma aplicação precisa de uma estrutura de dados na qual cada elemento tem uma chave com valor no conjunto $U = \{0, 1, \ldots, m-1\}$, em que m não é muito grande.
  - Supomos que não existem chaves repetidas.

- Podemos representar essa estrutura como um vetor $T[0..m-1]$ em que cada posição corresponde a uma chave do conjunto U.

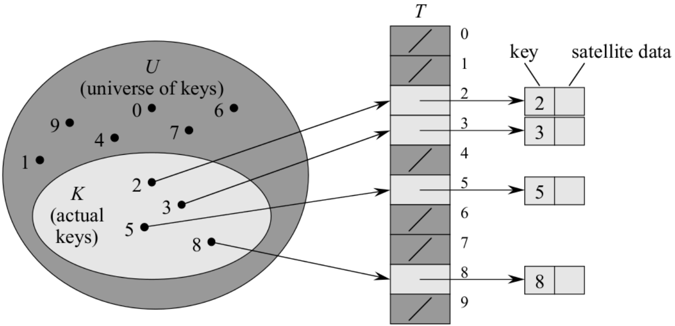


Tabela de acesso direto — Limitações

- Se o universo de possíveis chaves U for grande, então armazenar o vetor T de tamanho $|U|$ pode ser impraticável.
  - A memória é limitada.

Tabela de acesso direto — Limitações

- Se o universo de possíveis chaves U for grande, então armazenar o vetor T de tamanho $|U|$ pode ser impraticável.
    - A memória é limitada.

- O número de chaves armazenadas pode ser muito pequeno quando comparado ao tamanho do conjunto U.
    - Espaço será desperdiçado.

Tabela de acesso direto — Limitações

- Se o universo de possíveis chaves U for grande, então armazenar o vetor T de tamanho $|U|$ pode ser impraticável.
  - A memória é limitada.

- O número de chaves armazenadas pode ser muito pequeno quando comparado ao tamanho do conjunto U.
  - Espaço será desperdiçado.

- Quando o conjunto K de chaves é muito pequeno em relação ao universo U, gostaríamos de poder armazenar as chaves em uma tabela de tamanho $\Theta(|K|)$ e ao mesmo tempo manter o benefício de realizar busca, inserção e remoção em tempo $O(1)$.

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Tabela Hash (Tabela de dispersão)

Tabela hash

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.

Tabela hash

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.

Tabela hash

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.
- Tempo médio para inserção, busca e remoção: $O(1)$

Tabela hash

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.
- Tempo médio para inserção, busca e remoção: $O(1)$
- Tempo $O(n)$ no pior caso.

Tabela hash

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.
- Tempo médio para inserção, busca e remoção: $O(1)$
- Tempo $O(n)$ no pior caso.

Tabela hash

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Estrutura de dados onde as posições dos objetos armazenados são calculadas através de uma função que visa distribuir os elementos aleatoriamente ao longo de um vetor.
- Tempo médio para inserção, busca e remoção: $O(1)$
- Tempo $O(n)$ no pior caso.
- Usada em situações onde precisa-se apenas de operações de inserir, buscar e remover. Não se pode fazer caminhamento ordenado.

Componentes de uma tabela de dispersão

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

(1) Função de hashing

- As chaves nem sempre são valores numéricos. Por exemplo, as chaves podem consistir em nomes de pessoas.
  - Solução: criar uma função de hashing para mapear cada chave em uma posição i do vetor $T[0..m-1]$, com $0 \leq i \leq m-1$.

Componentes de uma tabela de dispersão

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

(1) Função de hashing

- As chaves nem sempre são valores numéricos. Por exemplo, as chaves podem consistir em nomes de pessoas.
  - Solução: criar uma função de hashing para mapear cada chave em uma posição i do vetor $T[0..m-1]$, com $0 \leq i \leq m-1$.

(2) Tratamento de colisão

- Duas chaves podem ser mapeadas no mesmo slot.
- Neste caso, teremos uma Colisão: duas ou mais chaves são mapeadas para a mesma posição da tabela.
- Devemos tratar uma colisão quando ela ocorrer.

Tratamento de colisão por encadeamento exterior

Encadeamento Exterior

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca

boca

bolo

bela

bala

dia

escola

gratuito

ilha

0	NULL
1	NULL
2	NULL
3	NULL
4	NULL
5	NULL
6	NULL
7	NULL
8	NULL

Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca

boca

bolo

bela

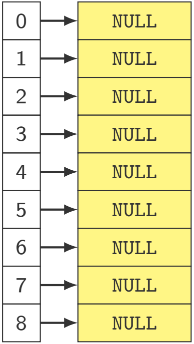
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca

boca

bolo

bela

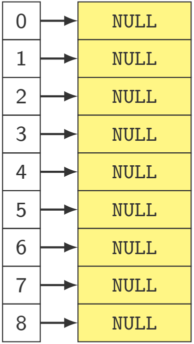
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca

bolo

bela

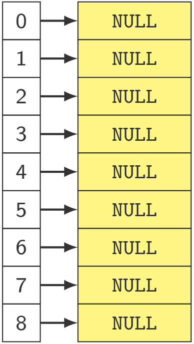
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca ⇝ `h("broca") = 3`

boca

bolo

bela

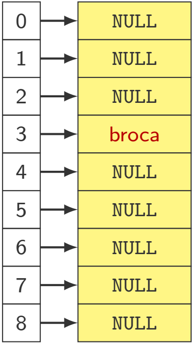
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo

bela

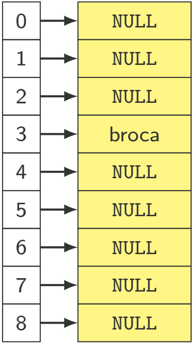
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo

bela

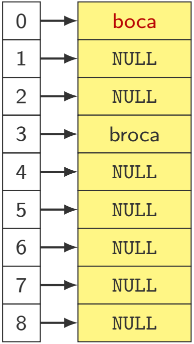
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela

bala

dia

escola

gratuito

ilha

0	boca
1	NULL
2	NULL
3	broca
4	NULL
5	NULL
6	NULL
7	NULL
8	NULL

Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela

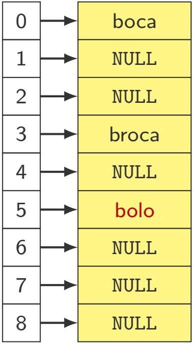
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

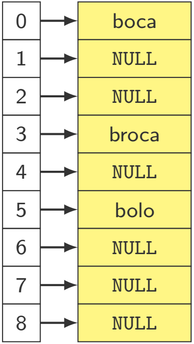
bala

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

bala

dia

escola

gratuito

ilha

0 $\rightarrow$	boca
1 $\rightarrow$	NULL
2 $\rightarrow$	bela
3 $\rightarrow$	broca
4 $\rightarrow$	NULL
5 $\rightarrow$	bolo
6 $\rightarrow$	NULL
7 $\rightarrow$	NULL
8 $\rightarrow$	NULL

Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

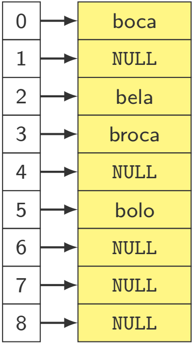
bala $\rightsquigarrow$ `h("bala") = 0`

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

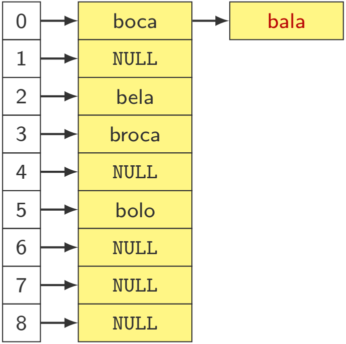
bala $\rightsquigarrow$ `h("bala") = 0`

dia

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

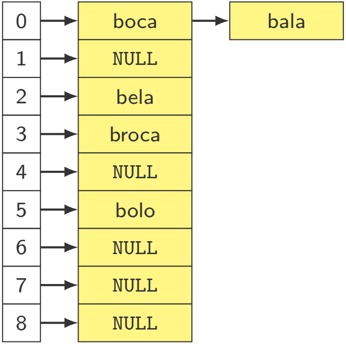
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

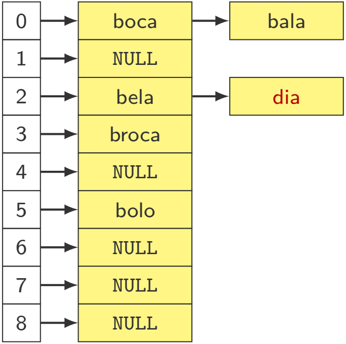
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

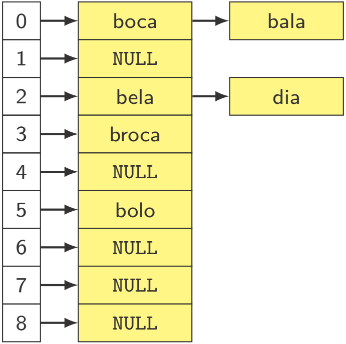
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

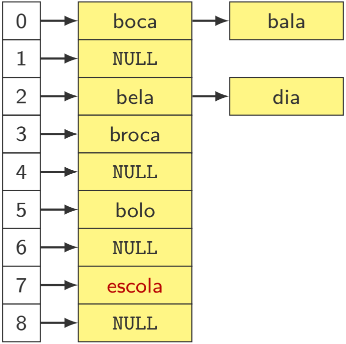
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

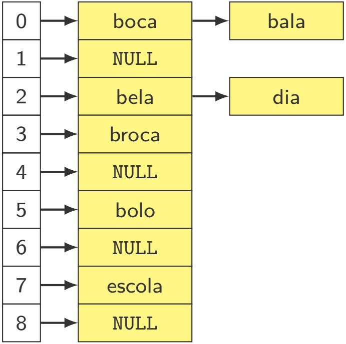
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito $\rightsquigarrow$ `h("gratuito") = 0`

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

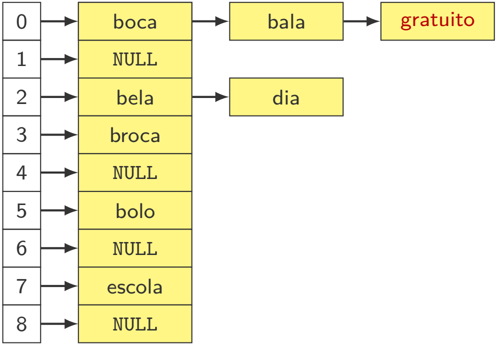
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito $\rightsquigarrow$ `h("gratuito") = 0`

ilha



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

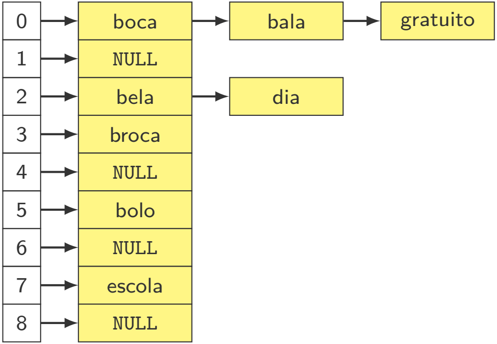
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito $\rightsquigarrow$ `h("gratuito") = 0`

ilha $\rightsquigarrow$ `h("ilha") = 6`



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Encadeamento Exterior

broca $\rightsquigarrow$ `h("broca") = 3`

boca $\rightsquigarrow$ `h("boca") = 0`

bolo $\rightsquigarrow$ `h("bolo") = 5`

bela $\rightsquigarrow$ `h("bela") = 2`

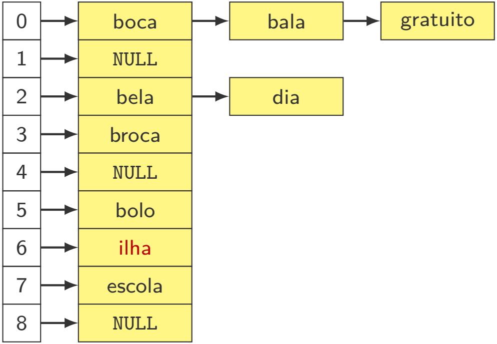
bala $\rightsquigarrow$ `h("bala") = 0`

dia $\rightsquigarrow$ `h("dia") = 2`

escola $\rightsquigarrow$ `h("escola") = 7`

gratuito $\rightsquigarrow$ `h("gratuito") = 0`

ilha $\rightsquigarrow$ `h("ilha") = 6`



Ideia:

- Dada uma tabela com $M = 9$ slots, usamos uma função hash para associar cada chave a um número inteiro (entre 0 e 8)

Complexidade do caso médio para tabela hash com encadeamento exterior

Algumas definições necessárias

- **Definição:** O fator de carga de uma tabela hash é o valor $\alpha = n/M$, onde n é o número de chaves armazenadas e M é o tamanho da tabela.

Algumas definições necessárias

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- **Definição:** O fator de carga de uma tabela hash é o valor $\alpha = n/M$, onde n é o número de chaves armazenadas e M é o tamanho da tabela.

Algumas definições necessárias

- **Definição:** O fator de carga de uma tabela hash é o valor $\alpha = n/M$, onde n é o número de chaves armazenadas e M é o tamanho da tabela.

- **Definição:** Uma função de hashing h é uniforme se a probabilidade de que $h(x)$ seja igual a k é $1/M$ para toda chave x e todos os endereços $k \in \{0, \ldots, M-1\}$.

Algumas definições necessárias

- **Definição:** O fator de carga de uma tabela hash é o valor $\alpha = n/M$, onde n é o número de chaves armazenadas e M é o tamanho da tabela.

- **Definição:** Uma função de hashing h é uniforme se a probabilidade de que $h(x)$ seja igual a k é $1/M$ para toda chave x e todos os endereços $k \in \{0, \ldots, M-1\}$.

- **Definição:** Seja A um algoritmo, $\{E_1, \ldots, E_m\}$ o conjunto de todas as entradas possíveis de A. Denote por t_i o número de passos efetuados por A quando a entrada for E_i. Seja p_i a probabilidade de ocorrência da entrada E_i. A complexidade do caso médio de A é dada por

$$\sum_{i=1}^{m} p_i t_i.$$

Complexidade do caso médio para da busca malsucedida

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Complexidade da busca malsucedida

Teorema 1. Numa tabela hash que utiliza função de hashing uniforme e na qual as colisões são tratadas por encadeamento exterior, o número médio de comparações efetuadas numa busca sem sucesso é igual ao fator de carga α.

Complexidade do caso médio para da busca malsucedida

Demonstração:

- Seja $N(T)$ o número médio de comparações efetuadas em uma busca sem sucesso, numa tabela hash T.

Complexidade do caso médio para da busca malsucedida

Demonstração:

- Seja $N(T)$ o número médio de comparações efetuadas em uma busca sem sucesso, numa tabela hash T.

- Como a função de hashing é uniforme, existe a mesma probabilidade $1/M$ da busca ser efetuada em qualquer uma das M listas encadeadas. Seja L_i a lista onde a busca se efetua e $|L_i|$ o seu comprimento. Então,

$$N(T) = \frac{1}{M}|L_0| + \frac{1}{M}|L_1| + \cdots + \frac{1}{M}|L_{M-1}| = \frac{1}{M}\sum_{i=0}^{M-1}|L_i|.$$

Complexidade do caso médio para da busca malsucedida

Universidade Federal do Ceará
Campus Quixadá

Demonstração:

- Seja $N(T)$ o número médio de comparações efetuadas em uma busca sem sucesso, numa tabela hash T.
- Como a função de hashing é uniforme, existe a mesma probabilidade $1/M$ da busca ser efetuada em qualquer uma das M listas encadeadas. Seja L_i a lista onde a busca se efetua e $|L_i|$ o seu comprimento. Então,

$$N(T) = \frac{1}{M}|L_0| + \frac{1}{M}|L_1| + \cdots + \frac{1}{M}|L_{M-1}| = \frac{1}{M}\sum_{i=0}^{M-1}|L_i|.$$

- Como existe um total de n elementos, $\sum_{i=0}^{M-1}|L_i| = n$. Logo,

$$N(T) = \frac{n}{M} = \alpha. \qquad \blacksquare$$

Complexidade do caso médio da busca bem sucedida

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 2. Numa tabela hash que utiliza função de hashing uniforme e na qual as colisões são tratadas por encadeamento exterior, o número médio de comparações efetuadas numa busca com sucesso é igual a $1 + \alpha/2 - 1/2M$. $\square$

Demonstração:

Complexidade do caso médio da busca bem sucedida

Teorema 2. Numa tabela hash que utiliza função de hashing uniforme e na qual as colisões são tratadas por encadeamento exterior, o número médio de comparações efetuadas numa busca com sucesso é igual a $1 + \alpha/2 - 1/2M$. $\square$

Demonstração:

- Seja T uma tabela hash e $N(T)$ o número médio de comparações efetuadas em uma busca com sucesso.

Complexidade do caso médio da busca bem sucedida

Teorema 2. Numa tabela hash que utiliza função de hashing uniforme e na qual as colisões são tratadas por encadeamento exterior, o número médio de comparações efetuadas numa busca com sucesso é igual a $1 + \alpha/2 - 1/2M$. □

Demonstração:

- Seja T uma tabela hash e $N(T)$ o número médio de comparações efetuadas em uma busca com sucesso.
- Sabe-se que a inclusão de cada chave nas listas encadeadas é realizada sempre no final da lista. Supondo a ausência de exclusões nas listas, a posição de cada chave em relação à cabeça da lista se mantém constante.

Complexidade do caso médio da busca bem sucedida

Teorema 2. Numa tabela hash que utiliza função de hashing uniforme e na qual as colisões são tratadas por encadeamento exterior, o número médio de comparações efetuadas numa busca com sucesso é igual a $1 + \alpha/2 - 1/2M$. □

Demonstração:

- Seja T uma tabela hash e $N(T)$ o número médio de comparações efetuadas em uma busca com sucesso.

- Sabe-se que a inclusão de cada chave nas listas encadeadas é realizada sempre no final da lista. Supondo a ausência de exclusões nas listas, a posição de cada chave em relação à cabeça da lista se mantém constante.

- Cada uma das n chaves tem a mesma probabilidade $\frac{1}{n}$ de ser pesquisada.

Complexidade do caso médio da busca bem sucedida

Universidade Federal do Ceará
Campus Quixadá

- Logo, o número médio de comparações para localizar uma chave x, com sucesso, localizada em uma certa lista L_i, é igual ao comprimento médio de L_i na ocasião em que a chave foi inserida em L_i, adicionado de uma unidade (correspondente à comparação final com a própria chave x).

Complexidade do caso médio da busca bem sucedida

- Logo, o número médio de comparações para localizar uma chave x, com sucesso, localizada em uma certa lista L_i, é igual ao comprimento médio de L_i na ocasião em que a chave foi inserida em L_i, adicionado de uma unidade (correspondente à comparação final com a própria chave x).

- Supondo que x tenha sido a $(j + 1)$-ésima chave a ser incluída, o comprimento médio de L_i é j/M. Logo,

Complexidade do caso médio da busca bem sucedida

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Logo, o número médio de comparações para localizar uma chave x, com sucesso, localizada em uma certa lista L_i, é igual ao comprimento médio de L_i na ocasião em que a chave foi inserida em L_i, adicionado de uma unidade (correspondente à comparação final com a própria chave x).

- Supondo que x tenha sido a $(j+1)$-ésima chave a ser incluída, o comprimento médio de L_i é j/M. Logo,

$$
\begin{aligned}
N(T) &= \frac{1}{n}\left(1 + \frac{0}{M}\right) + \frac{1}{n}\left(1 + \frac{1}{M}\right) + \cdots + \frac{1}{n}\left(1 + \frac{n-1}{M}\right) \\
&= \frac{1}{n}\sum_{j=0}^{n-1}\left(1 + \frac{j}{M}\right) = 1 + \frac{n(n-1)}{2nM} = 1 + \frac{\alpha}{2} - \frac{1}{2M}. \quad \square
\end{aligned}
$$

Interpretação dos resultados

- Se o tamanho da tabela for proporcional ao número de elementos, então temos que $n = O(M)$.

Interpretação dos resultados

- Se o tamanho da tabela for proporcional ao número de elementos, então temos que $n = O(M)$.
- Ou seja, $\alpha = n/M = O(M)/M = O(1)$.

Interpretação dos resultados

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Se o tamanho da tabela for proporcional ao número de elementos, então temos que $n = O(M)$.
- Ou seja, $\alpha = n/M = O(M)/M = O(1)$.
- Tanto a complexidade média de busca sem sucesso quanto a da busca com sucesso são constantes.

Interpretação dos resultados

- Se o tamanho da tabela for proporcional ao número de elementos, então temos que $n = O(M)$.
- Ou seja, $\alpha = n/M = O(M)/M = O(1)$.
- Tanto a complexidade média de busca sem sucesso quanto a da busca com sucesso são constantes.

A fim de garantir que as listas não se tornem muito longas, geralmente, mantém-se o invariante

$$\frac{n}{M} \leq 1$$

Para isso, pode ser necessário aumentar o tamanho da tabela e reconstruí-la.

Função de hashing

Função de hashing

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Definição

Dado um conjunto de chaves K e um inteiro positivo M, uma função de hashing é uma função $h\colon K \to \{0, 1, \ldots, M-1\}$ que idealmente satisfaz as seguintes condições:

- produz um número baixo de colisões.
- é facilmente computável.
- é uniforme: a probabilidade de que $h(x)$ seja igual ao índice i dever ser $1/M$ para todas as chaves $x \in K$ e todos os endereços $i \in \{0, \ldots, M-1\}$.

Função de hashing

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Na prática, é conveniente implementar uma função de hashing h como a composição de duas funções f e g.

Função de hashing

- Na prática, é conveniente implementar uma função de hashing h como a composição de duas funções f e g.
  - A função de codificação f mapeia chaves em inteiros não negativos (hash codes):

$$f\colon K \to \mathbb{Z}_{\geq 0}$$

Função de hashing

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Na prática, é conveniente implementar uma função de hashing h como a composição de duas funções f e g.
  - A função de codificação f mapeia chaves em inteiros não negativos (hash codes):

$$f\colon K \to \mathbb{Z}_{\geq 0}$$

  - A função de compressão g mapeia `hash codes` em inteiros no conjunto $\{0, 1, \ldots, M-1\}$:

$$g\colon \mathbb{Z}_{\geq 0} \to \{0, \ldots, M-1\}$$

Função de hashing

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Na prática, é conveniente implementar uma função de hashing h como a composição de duas funções f e g.
  - A função de codificação f mapeia chaves em inteiros não negativos (hash codes):
  $$f\colon K \to \mathbb{Z}_{\geq 0}$$
  - A função de compressão g mapeia `hash codes` em inteiros no conjunto $\{0, 1, \ldots, M-1\}$:
  $$g\colon \mathbb{Z}_{\geq 0} \to \{0, \ldots, M-1\}$$
- Assim, o valor de hashing $h(x)$ de uma chave $x \in K$ é dado por
$$h(x) = g(f(x)).$$

Primeira componente da função de hashing: Função de codificação

Função de codificação — Strings

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Strings estão entre os tipos mais comuns de chaves.
- Temos um conjunto de chaves K do tipo string e queremos construir uma função de codificação $f\colon K \to \mathbb{Z}^+$. Vamos supor que o tipo de retorno da função é `unsigned int`.

Função de codificação — Strings

- Strings estão entre os tipos mais comuns de chaves.
- Temos um conjunto de chaves K do tipo string e queremos construir uma função de codificação $f\colon K \to \mathbb{Z}^+$. Vamos supor que o tipo de retorno da função é `unsigned int`.
- Pergunta: Dado um valor do tipo `string`, como transformá-lo em um valor do tipo `unsigned int`?

Função de codificação — Strings

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Strings estão entre os tipos mais comuns de chaves.

- Temos um conjunto de chaves K do tipo string e queremos construir uma função de codificação $f\colon K \to \mathbb{Z}^+$. Vamos supor que o tipo de retorno da função é `unsigned int`.

- Pergunta: Dado um valor do tipo `string`, como transformá-lo em um valor do tipo `unsigned int`?
  - **Fato:** Uma string é composta por uma cadeia de caracteres. Cada caractere é um inteiro positivo cujo valor é determinado na tabela ASCII.
  - Ideia: Vamos usar o valor ASCII de cada caractere da chave para compor o valor de retorno da função.

Função de codificação — Strings

- Por exemplo, 'c' = 99, 'a' = 97 e 't' = 116, então essa função hash produziria $99 + 97 + 116 = 312$ para a string "cat".

```
size_t string_hash_ruim (const char *x, size_t len) {
   size_t sum = 0;
   for (size_t i = 0; i < len; ++i)
      sum += x[i];
   return sum;
}
```

Função de codificação — Strings

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

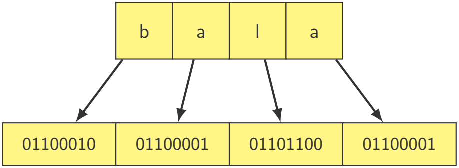
- Por exemplo, 'c' = 99, 'a' = 97 e 't' = 116, então essa função hash produziria $99 + 97 + 116 = 312$ para a string "cat".

```
size_t string_hash_ruim (const char *x, size_t len) {
   size_t sum = 0;
   for (size_t i = 0; i < len; ++i)
      sum += x[i];
   return sum;
}
```

- Essa é uma função de codificação simples, mas não é muito boa. Por exemplo, produz o mesmo valor para "act" como para "cat".

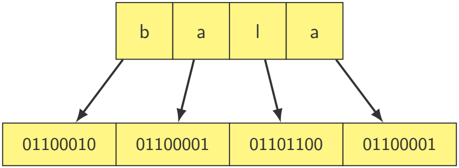
- Uma função de codificação para strings mais sofisticada deve certamente depender de todos os caracteres e da ordem deles.

Função de codificação — Strings



Como podemos calcular o número x que representa "bala"?

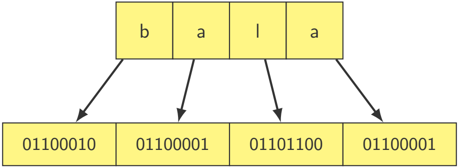
Função de codificação — Strings



Como podemos calcular o número x que representa "bala"?

- $x = \texttt{'b'} \cdot 127^3 + \texttt{'a'} \cdot 127^2 + \texttt{'l'} \cdot 127^1 + \texttt{'a'} \cdot 127^0$

Função de codificação — Strings

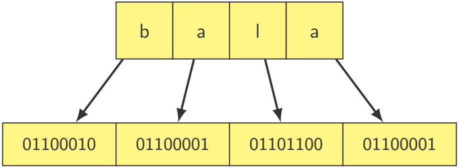


Como podemos calcular o número x que representa "bala"?

- $x = \texttt{'b'} \cdot 127^3 + \texttt{'a'} \cdot 127^2 + \texttt{'l'} \cdot 127^1 + \texttt{'a'} \cdot 127^0$

que pode ser reescrito como

Função de codificação — Strings



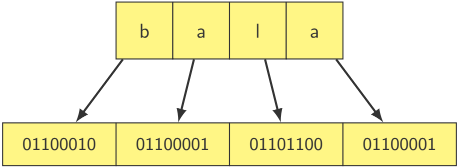
Como podemos calcular o número x que representa "bala"?

- $x = \texttt{'b'} \cdot 127^3 + \texttt{'a'} \cdot 127^2 + \texttt{'l'} \cdot 127^1 + \texttt{'a'} \cdot 127^0$

que pode ser reescrito como

- $x = (((\texttt{'b'}) \cdot 256 + \texttt{'a'}) \cdot 127 + \texttt{'l'}) \cdot 127 + \texttt{'a'}$

Função de codificação — Strings



Como podemos calcular o número x que representa “bala”?

- $x = \texttt{'b'} \cdot 127^3 + \texttt{'a'} \cdot 127^2 + \texttt{'l'} \cdot 127^1 + \texttt{'a'} \cdot 127^0$

que pode ser reescrito como

- $x = (((\texttt{'b'}) \cdot 256 + \texttt{'a'}) \cdot 127 + \texttt{'l'}) \cdot 127 + \texttt{'a'}$

Caso o valor de x fique muito grande para caber em um `unsigned int`, uma operação modulo M é executada internamente, em que M é o valor do maior `unsigned int`.

Função de codificação — Strings

```
// teste.cpp
size_t string_hash ( const char *x, size_t len ) {
    size_t code = 0, BASE = 127;
    for (size_t i = 0; i < len; ++i)
        code = ( code * BASE ) + x[i];
    return code;
}
```

Função de codificação — Strings

```
// teste.cpp
size_t string_hash ( const char *x, size_t len ) {
    size_t code = 0, BASE = 127;
    for (size_t i = 0; i < len; ++i)
        code = ( code * BASE ) + x[i];
    return code;
}
```

Exemplos com $BASE = 127$:

- "casa" $=$ 204369132
- "asac" $=$ 200560404
- "saca" $=$ 237141228

Função de codificação — `float`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Ideia: trataremos o número de ponto flutuante como uma sequência de bytes. Em C++ um `float` utiliza 32 bits (4 bytes).
- Como um `char` é armazenado em 8 bits (1 byte), podemos interpretar um `float` de 32 bits como um vetor de 4 caracteres.

Função de codificação — `float`

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Ideia: trataremos o número de ponto flutuante como uma sequência de bytes. Em C++ um `float` utiliza 32 bits (4 bytes).
- Como um `char` é armazenado em 8 bits (1 byte), podemos interpretar um `float` de 32 bits como um vetor de 4 caracteres.
- C++ fornece uma operação de casting `reinterpret_cast`, para conversão entre tipos não relacionados.
    - Esse casting trata valores como uma sequência de bits e não tenta converter de maneira inteligente o significado de um valor para outro.

Função de codificação — `float`

```
size_t float_hash ( const float& f )
{
    const char *p = reinterpret_cast<const char*>(&f);
    return string_hash( p, sizeof(f) );
}
```

Função de codificação — `float`

```
size_t float_hash ( const float& f )
{
    const char *p = reinterpret_cast<const char*>(&f);
    return string_hash( p, sizeof(f) );
}
```

Exemplos usando `hash_float`:

- $23.564 = 34846198$
- $87.6 = 105279891$
- $89.096 = 80667418$
- $67.8 = 18446744073498939962$
- $3.23413 = 18446744073701287409$

Função de codificação genérica


```cpp
#include <iostream> // generic.cpp
#include <string>
#include <type_traits>
```

```cpp
template<typename T>
size_t generic_hash(const T& v) {
    if constexpr (std::is_same<T, std::string>::value) {
        // trata std::string
        return string_hash(v.c_str(), v.size());
    } else {
        // trata outros tipos genericamente
        const char* p = reinterpret_cast<const char*>(&v);
        return string_hash(p, sizeof(v));
    }
}
```

- **if constexpr** é um condicional em tempo de compilação disponível desde o C++17. Ele garante que apenas o ramo relevante do código seja compilado. Ele garante que o primeiro bloco só será compilado se for passado um `std::string`.

Template de classe std::hash

std::hash

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- O C++ possui o cabeçalho <`functional`> no qual está definido o template `std::hash` da seguinte maneira:

```
namespace std {
  template< typename T > class hash;
}
```

std::hash

- O C++ possui o cabeçalho `<functional>` no qual está definido o template `std::hash` da seguinte maneira:

```
namespace std {
  template< typename T > class hash;
}
```

- Em `<functional>` há especializações desse template para todos os tipos primitivos, como `int`, `float`, `double`, etc. e também para `std::string`.

- Em resumo: um objeto da classe `std::hash<T>` é um objeto sem estado que implementa o operador `()`. Esse operador recebe como parâmetro um valor do tipo `T` e retorna seu `hash` code como `size_t`.

std::hash — Exemplo de uso

```cpp
#include <iostream> //hash01.cpp
#include <string>
#include <functional> // para std::hash
using namespace std;

int main () {
  string name { "Ana Almeida" };
  hash<string> ff;
  cout << name << " = " << ff(name) << endl;

  int i = 24, j = -24;
  hash<int> hint;
  cout << i << " = " << hint( i ) << endl;
  cout << j << " = " << hint( j ) << endl;

  double d = 23.45, e = 24.35;
  cout << d << " = " << hash<double>()( d ) << endl;
  cout << e << " = " << hash<double>()( e ) << endl;
  return 0;
}
```

std::hash — Tipos definidos pelo usuário

Também podemos especializar o template `std::hash` para definir hash codes para tipos que nós mesmos definirmos.

```cpp
#include <iostream> // hash02.cpp
#include <string>
#include <functional>

struct Name {
   std::string first;
   std::string last;
};

namespace std {
   template <>
   class hash< Name > {
      public:
         size_t operator()( const Name & p ) const {
            auto n1 = std::hash<string>()(p.first);
            auto n2 = std::hash<string>()(p.last);
            return n1 ^ n2; // XOR operation
         }
   };
}
```

std::hash — Tipos definidos pelo usuário (cont.)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
int main () {
  Name p1 {"Ana", "Almeida" };
  Name p2 {"Aan", "Ameidal" };

  size_t key1 = std::hash<Name>()( p1 );
  size_t key2 = std::hash<Name>()( p2 );

  std::cout << key1 << std::endl;
  std::cout << key2 << std::endl;

  return 0;
}
```

Função de codificação como uma classe própria

```cpp
#include <iostream> // hash03.cpp
#include <functional>

struct Name {
    std::string first;
    std::string last;
};

class hashName {
public:
    size_t operator()( const Name &name ) const {
        auto n1 = std::hash<std::string>()(name.first);
        auto n2 = std::hash<std::string>()(name.last);
        return n1 ^ n2;
    }
};

int main () {
  Name p3 { "Pedro", "Paulo" };
  size_t key3 = hashName()( p3 );
  std::cout << key3 << std::endl;
  return 0;
}
```

Segunda componente da função de hashing: Função de compressão

Recapitulando funções hash...

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Implementamos uma função hash h como a composição de duas funções f e g.
  - A função de codificação f mapeia chaves em inteiros não negativos (hash codes):
  $$f\colon K \to \mathbb{Z}^+$$
  - A função de compressão g mapeia hash codes em inteiros no conjunto $\{0, 1, \ldots, M-1\}$:
  $$g\colon \mathbb{Z}^+ \to \{0, \ldots, M-1\}$$

Recapitulando funções hash...

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Implementamos uma função hash h como a composição de duas funções f e g.
  - A função de codificação f mapeia chaves em inteiros não negativos (hash codes):
$$f\colon K \to \mathbb{Z}^+$$
  - A função de compressão g mapeia hash codes em inteiros no conjunto $\{0, 1, \ldots, M-1\}$:
$$g\colon \mathbb{Z}^+ \to \{0, \ldots, M-1\}$$

Veremos a seguir como implementar uma função de compressão usando o método da divisão.

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$

- Ideia: usar a função módulo para obter o resto da divisão de x pelo tamanho M da tabela hash.

$$g(x) = x \textbf{ mod } M$$

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$

- Ideia: usar a função módulo para obter o resto da divisão de x pelo tamanho M da tabela hash.

$$g(x) = x \textbf{ mod } M$$

Exemplo: supondo hash("bala") $= 1.650.551.905$, temos

$$g(\text{hash(“bala”)}) = 1.650.551.905 \textbf{ mod } 1783 = 277$$

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$

- Ideia: usar a função módulo para obter o resto da divisão de x pelo tamanho M da tabela hash.

$$g(x) = x \textbf{ mod } M$$

Exemplo: supondo hash(“bala”) $= 1.650.551.905$, temos

$$g(\text{hash(“bala”)}) = 1.650.551.905 \textbf{ mod } 1783 = 277$$

Escolhendo M:

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$

- Ideia: usar a função módulo para obter o resto da divisão de x pelo tamanho M da tabela hash.

$$g(x) = x \textbf{ mod } M$$

Exemplo: supondo hash(“bala”) $= 1.650.551.905$, temos

$$g(\text{hash(“bala”)}) = 1.650.551.905 \textbf{ mod } 1783 = 277$$

Escolhendo M:

- escolher M como uma potência de 2 não é uma boa ideia:

Método da divisão

- **Objetivo:** Construir função g que mapeia qualquer inteiro x não negativo no conjunto $\{0, 1, \ldots, M-1\}$
- Ideia: usar a função módulo para obter o resto da divisão de x pelo tamanho M da tabela hash.

$$g(x) = x \ \textbf{mod}\ M$$

Exemplo: supondo hash(“bala”) $= 1.650.551.905$, temos

$$g(\text{hash(“bala”)}) = 1.650.551.905 \ \textbf{mod}\ 1783 = 277$$

Escolhendo M:

- escolher M como uma potência de 2 não é uma boa ideia:
  - considera apenas os bits menos significativos. Exemplo:

Método da divisão — M como potência de 2

- Suponha $M = 2^j$ e a chave armazenada numa palavra de memória de 16 bits.

Método da divisão — M como potência de 2

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Suponha $M = 2^j$ e a chave armazenada numa palavra de memória de 16 bits.
- Se $j = 5$, a função $g(x)$ produzirá endereços que resultam dos últimos cinco bits da chave, isto é, $g(x)$ não levará nunca em consideração os dígitos mais significativos de x.

Método da divisão — M como potência de 2

- Suponha $M = 2^j$ e a chave armazenada numa palavra de memória de 16 bits.
- Se $j = 5$, a função $g(x)$ produzirá endereços que resultam dos últimos cinco bits da chave, isto é, $g(x)$ não levará nunca em consideração os dígitos mais significativos de x.

x	$g(x) = x \mod 32$	binário(x)
16838	6	1000001110**00110**
5758	30	10110011**11110**
17515	11	1000100011**01011**
31051	11	1111001010**01011**
5627	27	10101111**11011**
23010	2	1011001111**00010**
7419	27	11100111**11011**
16212	20	111111010**10100**
4086	22	1111111**10110**

Método da divisão — M como potência de 2

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Suponha $M = 2^j$ e a chave armazenada numa palavra de memória de 16 bits.
- Se $j = 5$, a função $g(x)$ produzirá endereços que resultam dos últimos cinco bits da chave, isto é, $g(x)$ não levará nunca em consideração os dígitos mais significativos de x.

x	g$(x) = x \mod 32$	binário(x)
16838	6	100000111**00110**
5758	30	10110011**11110**
17515	11	100010001**01011**
31051	11	111100101**01011**
5627	27	10101111**11011**
23010	2	101100111**00010**
7419	27	11100111**11011**
16212	20	1111110101**10100**
4086	22	1111111**10110**

- Outra escolha ruim: Se M for par, $g(x)$ será par quando x for par e será ímpar caso contrário.

Escolhendo o tamanho M da tabela hash

- Dica: normalmente escolhemos M como um número primo.

Escolhendo o tamanho M da tabela hash

Universidade Federal do Ceará
Campus Quixadá

- Dica: normalmente escolhemos M como um número primo.
- Dica do Sedgewick: Escolha uma potência de 2 que esteja próxima do valor desejado de M. Depois, adote para M o número primo que esteja logo abaixo da potência escolhida.

Escolhendo o tamanho M da tabela hash

- Dica: normalmente escolhemos M como um número primo.
- Dica do Sedgewick: Escolha uma potência de 2 que esteja próxima do valor desejado de M. Depois, adote para M o número primo que esteja logo abaixo da potência escolhida.

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

Método da divisão — Implementação

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
#include <iostream> // hashFunction02a.cpp
#include <iomanip>
#include <functional>
using namespace std;

// funcao de codificacao + uncao de compressao
size_t hash_code( const float& chave, size_t tableSize )
{
    return std::hash<float>()(chave) % tableSize;
}

int main()
{
    for(int i = 1; i <= 10; ++i)
    {
        cout << "hash_code(" << setw(3) << right << i * 0.5
             << ") = " << hash_code(i * 0.5, 251) << endl;
    }
}
```

Implementação da tabela hash com tratamento de colisão por encadeamento exterior

Detalhes de implementação

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Implementaremos a tabela hash como um template de classe chamado **HashTable**. O template terá dois parâmetros: `T` para a chave e `V` para o valor associado à chave.
  - Dentro da classe, esses dois valores serão representados como um tipo composto. Para isso, usaremos o tipo `std::pair` padrão do C++.

Detalhes de implementação

- Implementaremos a tabela hash como um template de classe chamado **HashTable**. O template terá dois parâmetros: `T` para a chave e `V` para o valor associado à chave.
  - Dentro da classe, esses dois valores serão representados como um tipo composto. Para isso, usaremos o tipo `std::pair` padrão do C++.

- A tabela hash com tratamento de colisão por encadeamento exterior consiste em um vetor $T[0..M-1]$ com M slots, onde cada slot é uma lista encadeada contendo as chaves mapeadas naquele slot.
  - Não vamos programar do zero.
  - Vamos usar um `std::vector` em que cada elemento é uma lista `std::list` de elementos do tipo `std::pair<T,V>`

Classe `std::pair`

- `std::pair`: Esta classe acopla um par de valores, que podem ser de diferentes tipos (T1 e T2). Está definido no cabeçalho `<utility>`.

Classe `std::pair`

- `std::pair`: Esta classe acopla um par de valores, que podem ser de diferentes tipos (T1 e T2). Está definido no cabeçalho `<utility>`.
- O primeiro elemento é acessado pelo atributo público '`first`' e o segundo elemento é acessado pelo atributo público '`second`' e a ordem é fixa `(first, second)`.

Classe `std::pair`

- `std::pair`: Esta classe acopla um par de valores, que podem ser de diferentes tipos (T1 e T2). Está definido no cabeçalho `<utility>`.
- O primeiro elemento é acessado pelo atributo público '`first`' e o segundo elemento é acessado pelo atributo público '`second`' e a ordem é fixa `(first, second)`.
- `std::pair` fornece uma maneira de armazenar dois objetos heterogêneos como uma única unidade. O par pode ser atribuído, copiado e comparado.
- Um template de função útil que vamos usar é a função `std::make_pair`. Esta função recebe como argumento dois valores dos tipos T1 e T2 e retorna um `std::pair<T1, T2>`

Classe std::pair — Exemplo

```cpp
// pair.cpp
#include <utility>      // std::pair
#include <iostream>     // std::cout
using std::cout;

int main () {
  std::pair <int,int> bar(3,4);
  std::pair <int,int> foo;
  foo = std::make_pair (10,20);

  cout << "foo: " << foo.first;
  cout << ", " << foo.second << '\n';

  cout << "bar: " << bar.first;
  cout << ", " << bar.second << '\n';

  return 0;
}
```

Outra técnica de tratamento de colisão: Endereçamento aberto

Endereçamento aberto

- Existem vários métodos para armazenar N chaves em uma tabela de tamanho $M > N$, os quais utilizam os slots vazios na própria tabela para resolver as colisões. Esses métodos são chamados endereçamento aberto (open addressing)

Endereçamento aberto

- Existem vários métodos para armazenar N chaves em uma tabela de tamanho $M > N$, os quais utilizam os slots vazios na própria tabela para resolver as colisões. Esses métodos são chamados endereçamento aberto (open addressing)

- Características:
  - evita percorrer usando ponteiros e alocação e desalocação de memória nas inserções e remoções.

Endereçamento aberto

Universidade Federal do Ceará
Campus Quixadá

- Existem vários métodos para armazenar N chaves em uma tabela de tamanho $M > N$, os quais utilizam os slots vazios na própria tabela para resolver as colisões. Esses métodos são chamados endereçamento aberto (open addressing)

- Características:
  - evita percorrer usando ponteiros e alocação e desalocação de memória nas inserções e remoções.
  - se a tabela encher, uma alternativa é criar uma tabela maior
    - e mudar a função de hashing

Endereçamento aberto

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Existem vários métodos para armazenar N chaves em uma tabela de tamanho $M > N$, os quais utilizam os slots vazios na própria tabela para resolver as colisões. Esses métodos são chamados endereçamento aberto (open addressing)

- Características:
  - evita percorrer usando ponteiros e alocação e desalocação de memória nas inserções e remoções.
  - se a tabela encher, uma alternativa é criar uma tabela maior
    - e mudar a função de hashing
  - No endereçamento aberto, a tabela pode ser preenchida até ficar cheia. Uma consequência disso é que o fator de carga $\alpha = \frac{n}{M}$ nunca é maior do que 1.

Endereçamento aberto — Inserção

- Para executar a inserção usando endereçamento aberto, examinamos sucessivamente a tabela hash (**sondamos**) até encontrar uma posição vazia na qual inserir a chave.

Endereçamento aberto — Inserção

- Para executar a inserção usando endereçamento aberto, examinamos sucessivamente a tabela hash (**sondamos**) até encontrar uma posição vazia na qual inserir a chave.

Endereçamento aberto — Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Para executar a inserção usando endereçamento aberto, examinamos sucessivamente a tabela hash (**sondamos**) até encontrar uma posição vazia na qual inserir a chave.

- **Exemplo:** queremos inserir a chave maria neste array. O problema é que não sabemos de antemão quais slots estão vazios. Qual abordagem ingênua poderíamos usar?

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	
8	

Endereçamento aberto — Inserção

- **Ideia:** Em vez de seguir a ordem de sondagem $0, 1, \ldots, m-1$ (o que exige o tempo de busca $O(n)$), fazemos com que a sequência de posições sondadas dependa da chave que está sendo inserida.

Endereçamento aberto — Inserção

- Para determinar quais serão as posições a sondar, estendemos a função de hashing para incluir o número da sondagem (a partir de 0) como uma segunda entrada. Assim, a função de hashing se torna:

$$h\colon U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\}.$$

Endereçamento aberto — Inserção

- Para determinar quais serão as posições a sondar, estendemos a função de hashing para incluir o número da sondagem (a partir de 0) como uma segunda entrada. Assim, a função de hashing se torna:

$$h\colon U \times \{0, 1, \ldots, m-1\} \;\to\; \{0, 1, \ldots, m-1\}.$$

- Com endereçamento aberto, exigimos que, para toda chave k, a **sequência de sondagem**

$$\langle h(k,0), h(k,1), \ldots, h(k,m-1) \rangle$$

seja uma permutação de $\langle 0, 1, \ldots, m-1 \rangle$, de modo que toda posição da tabela hash seja eventualmente considerada uma posição para uma nova chave, à medida que a tabela é preenchida.

Universidade Federal do Ceará
Campus Quixadá

Sondagem Linear

Endereçamento aberto com sondagem linear

broca

boca

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	
4	
5	
6	
7	
8	

Inserindo:

Endereçamento aberto com sondagem linear

broca ⇝ `h("broca",0) = 3`

boca

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo

bela

bala

dia

escola

gratuito

ilha

0	
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca ⇝ `h("broca",0) = 3`

boca ⇝ `h("boca",0) = 0`

bolo

bela

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala

dia

escola

gratuito

ilha

0	boca
1	
2	
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala

dia

escola

gratuito

ilha

0	boca
1	
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos

Endereçamento aberto com sondagem linear

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia

escola

gratuito

ilha

0	boca
1	
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia

escola

gratuito

ilha

0	boca
1	
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca ⇝ `h("broca",0) = 3`

boca ⇝ `h("boca",0) = 0`

bolo ⇝ `h("bolo",0) = 5`

bela ⇝ `h("bela",0) = 2`

bala ⇝ `h("bala",0) = 0`

dia ⇝ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca ⇝ `h("broca",0) = 3`

boca ⇝ `h("boca",0) = 0`

bolo ⇝ `h("bolo",0) = 5`

bela ⇝ `h("bela",0) = 2`

bala ⇝ `h("bala",0) = 0`

dia ⇝ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha $\rightsquigarrow$ `h("ilha",0) = 6`

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha $\rightsquigarrow$ `h("ilha",0) = 6`

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha $\rightsquigarrow$ `h("ilha",0) = 6`

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha $\rightsquigarrow$ `h("ilha",0) = 6`

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Endereçamento aberto com sondagem linear

Universidade Federal do Ceará
Campus Quixadá

broca $\rightsquigarrow$ `h("broca",0) = 3`

boca $\rightsquigarrow$ `h("boca",0) = 0`

bolo $\rightsquigarrow$ `h("bolo",0) = 5`

bela $\rightsquigarrow$ `h("bela",0) = 2`

bala $\rightsquigarrow$ `h("bala",0) = 0`

dia $\rightsquigarrow$ `h("dia",0) = 2`

escola $\rightsquigarrow$ `h("escola",0) = 7`

gratuito $\rightsquigarrow$ `h("gratuito",0) = 0`

ilha $\rightsquigarrow$ `h("ilha",0) = 6`

0	boca
1	bala
2	bela
3	broca
4	dia
5	bolo
6	gratuito
7	escola
8	ilha

Inserindo:

- procuramos posição
- se houver espaço, guardamos
- se não houver espaço, procuramos a próxima posição livre (módulo M)

Sondagem linear (linear probing)

- Dada uma *função hash* comum $h' : U \rightarrow \{0, 1, \ldots, m-1\}$, à qual nos referimos como uma **função hash auxiliar**, o método de **sondagem linear** usa a função hash

$$h(k, i) = (h'(k) + i) \bmod m, \text{ para } i = 0, 1, \ldots, m-1.$$

Sondagem linear (linear probing)

- Dada uma *função hash* comum $h' : U \to \{0, 1, \ldots, m-1\}$, à qual nos referimos como uma **função hash auxiliar**, o método de **sondagem linear** usa a função hash

$$h(k, i) = (h'(k) + i) \bmod m, \text{ para } i = 0, 1, \ldots, m-1.$$

- Obs.1: Note que $\langle h(k,0), h(k,1), \ldots, h(k, m-1)\rangle$ é uma permutação de $\langle 0, 1, \ldots, m-1\rangle$.

- Obs.2: Como a sondagem inicial determina toda a sequência de sondagem, há somente m sequências de sondagem distintas.

Sondagem linear (linear probing)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Dada uma *função hash* comum $h' : U \to \{0, 1, \ldots, m-1\}$, à qual nos referimos como uma **função hash auxiliar**, o método de **sondagem linear** usa a função hash

$$h(k, i) = (h'(k) + i) \bmod m, \text{ para } i = 0, 1, \ldots, m-1.$$

- Obs.1: Note que $\langle h(k,0), h(k,1), \ldots, h(k, m-1) \rangle$ é uma permutação de $\langle 0, 1, \ldots, m-1 \rangle$.

- Obs.2: Como a sondagem inicial determina toda a sequência de sondagem, há somente m sequências de sondagem distintas.

- A sondagem linear sofre de um problema conhecido como **agrupamento primário** (primary clustering): longas sequências de posições ocupadas se acumulam, aumentando o tempo médio de busca.

Implementação da Tabela

Endereçamento aberto — Implementação da Tabela

Universidade Federal do Ceará
Campus Quixadá

- Como os elementos serão agora guardados na própria tabela, precisamos saber quando uma posição da tabela está vazia (disponível) e quando ela contém um elemento válido (não está disponível).

Endereçamento aberto — Implementação da Tabela

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Como os elementos serão agora guardados na própria tabela, precisamos saber quando uma posição da tabela está vazia (disponível) e quando ela contém um elemento válido (não está disponível).

- Para isso, podemos supor que cada slot j da tabela $T[0..M-1]$ contém um objeto $T[j]$ que possui os seguintes campos:
  - `status:` indica se o slot está ou não disponível. Pode assumir os valores `EMPTY`, `ACTIVE` e `DELETED`.
  - `key:` guarda a chave.
  - `value:` guarda o valor associado à chave.

Endereçamento aberto — Implementação da Tabela

- No início, quando a tabela é criada, o campo `status` de todos os objetos são `EMPTY`.

Endereçamento aberto — Implementação da Tabela

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- No início, quando a tabela é criada, o campo `status` de todos os objetos são `EMPTY`.
- À medida que pares chave-valor são inseridos ou removidos, esses campos vão recebendo os valores `ACTIVE` ou `DELETED`, respectivamente, e nunca mais voltam a ser `EMPTY` enquanto a tabela se manter do mesmo tamanho.

Endereçamento aberto — Implementação da Tabela

Universidade Federal do Ceará
Campus Quixadá

- No início, quando a tabela é criada, o campo `status` de todos os objetos são `EMPTY`.
- À medida que pares chave-valor são inseridos ou removidos, esses campos vão recebendo os valores `ACTIVE` ou `DELETED`, respectivamente, e nunca mais voltam a ser `EMPTY` enquanto a tabela se manter do mesmo tamanho.
- Quando a tabela encher, uma operação de rehashing é executada, uma nova tabela maior é criada (com todos os status `EMPTY`) e os elementos da tabela antiga são re-inseridos na tabela nova.

Endereçamento aberto — Inserção

- O procedimento HASH-INSERT tem como entrada uma tabela hash $T[0..M-1]$ de tamanho M, uma chave k e seu valor v associado.
- Devolve o índice de T onde a chave k é armazenada ou sinaliza um erro porque a tabela já está cheia.

Endereçamento aberto — Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- O procedimento HASH-INSERT tem como entrada uma tabela hash $T[0..M-1]$ de tamanho M, uma chave k e seu valor v associado.
- Devolve o índice de T onde a chave k é armazenada ou sinaliza um erro porque a tabela já está cheia.

```
HASH-INSERT(T, k, v)
    i = 0
    do
        j = h(k,i)
        if(T[j].status == ACTIVE and T[j].key == k)
            break
        if( T[j].status != ACTIVE )
            T[j].key = k
            T[j].value = v
            T[j].status = ACTIVE
            return j
        i = i + 1
    while( i < M )
    error "hash table overflow or repeated key"
```

Busca em endereçamento aberto

Universidade Federal do Ceará
Campus Quixadá

Como fazer uma busca com endereçamento aberto?

Busca em endereçamento aberto

Como fazer uma busca com endereçamento aberto?

- Basta simular a inserção:
  - Calcule a função de hashing
  - Percorra a tabela em sequência procurando pela chave
  - Se encontrar a chave, devolva o valor associado à chave
  - Se encontrar um slot vazio (`status == EMPTY`), então lance uma exceção, indicando que não encontrou

Busca em tabela de endereçamento aberto

Algoritmo:

```
HASH-SEARCH(T, k)
    i = 0
    do
        j = h(k, i)
        if( T[j].status == ACTIVE and T[j].key == k )
            return T[j].value
        i = i + 1
    while( T[j].status != EMPTY and i < M )
    error "key not found"
```

Remoção em endereçamento aberto

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- **Ideia:** o campo `status` de cada objeto da tabela suporta um valor chamado `DELETED` indicando que aquele elemento foi removido.
- Assim, para remover o elemento da tabela, basta atribuir o valor `DELETED` ao campo `status` do objeto.

Remoção em endereçamento aberto

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Algoritmo:

```
HASH-DELETE(T, k)
    i = 0
    do
        j = h(k, i)
        if( T[j].status == ACTIVE and T[j].key == k )
            T[j].status = DELETED
            return true
        i = i + 1
    while( T[j].status != EMPTY and i < M )
    return false
```

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Sondagem por hashing duplo

Sondagem por hashing duplo

É semelhante à sondagem linear, só que agora nós usamos duas funções de hashing comuns, denominadas $hash_1$ e a $hash_2$:

Sondagem por hashing duplo

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

É semelhante à sondagem linear, só que agora nós usamos duas funções de hashing comuns, denominadas $hash_1$ e a $hash_2$:

- Quando detectamos conflito, ao invés de dar um pulo de 1, damos um pulo $h(k, i)$ calculado a partir de uma segunda função de hashing. Isto é,

$$h(k, i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

Sondagem por hashing duplo

É semelhante à sondagem linear, só que agora nós usamos duas funções de hashing comuns, denominadas $hash_1$ e a $hash_2$:

- Quando detectamos conflito, ao invés de dar um pulo de 1, damos um pulo $h(k,i)$ calculado a partir de uma segunda função de hashing. Isto é,

$$h(k,i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

Sequência de sondagem:

- $h(k,0) = hash_1(k) \bmod M$
- $h(k,1) = (hash_1(k) + hash_2(k)) \bmod M$
- $h(k,2) = (hash_1(k) + 2 \cdot hash_2(k)) \bmod M$
- $\ldots$

Cuidados a se tomar com o hashing duplo

Dada a função de hashing duplo:

$$h(k, i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

- $hash_2(k)$ nunca pode ser zero
- $hash_2(k)$ precisa ser co-primo com M.
  - garante que toda a tabela seja pesquisada.
  - dois números a e b são co-primos se não têm nenhum divisor em comum além de 1

Cuidados a se tomar com o hashing duplo (cont.)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

$$h(k, i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

Exemplos de escolhas para M e $hash_2$:

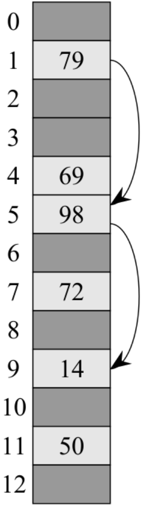
(i) Escolha M como uma potência de 2 e faça que $hash_2(k)$ seja sempre ímpar

(ii) Escolha M como um número primo e faça com que $hash_2(k) < M$. Por exemplo, escolhendo M primo, podemos fazer

- $hash_1(k) = k \mod M$
- $hash_2(k) = 1 + (k \mod m')$

onde m' é ligeiramente menor que M (por exemplo $M - 1$ ou um primo menor que M)

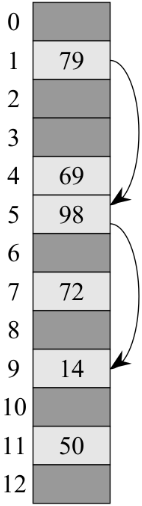
Hashing duplo — Exemplo



$$h(k, i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

- Tabela hash de tamanho $M = 13$ com $hash_1(k) = k \bmod 13$ e $hash_2(k) = 1 + (k \bmod 11)$.

Hashing duplo — Exemplo



$$h(k,i) = (hash_1(k) + i \cdot hash_2(k)) \mod M$$

- Tabela hash de tamanho $M = 13$ com $hash_1(k) = k \bmod 13$ e $hash_2(k) = 1 + (k \bmod 11)$.
- Como $14 \equiv 1 \pmod{13}$ e $14 \equiv 3 \pmod{11}$, inserimos a chave 14 na posição vazia 9, após examinar as posições 1 e 5 e verificarmos que elas já estão ocupadas.

Sondagem linear e Hashing duplo[1]

Sondagem linear - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.5	2.0	3.0	5.5
sem sucesso	2.5	5.0	8.5	55.5

[1] Baseado em Sedgewick, R. Algorithms in C, third edition, Addison-Wesley. 1998.

Sondagem linear e Hashing duplo[1]

Sondagem linear - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.5	2.0	3.0	5.5
sem sucesso	2.5	5.0	8.5	55.5

Hashing duplo - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.4	1.6	1.8	2.6
sem sucesso	1.5	2.0	3.0	5.5

[1] Baseado em Sedgewick, R. Algorithms in C, third edition, Addison-Wesley. 1998.

Sondagem linear e Hashing duplo[1]

Sondagem linear - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.5	2.0	3.0	5.5
sem sucesso	2.5	5.0	8.5	55.5

Hashing duplo - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.4	1.6	1.8	2.6
sem sucesso	1.5	2.0	3.0	5.5

De qualquer forma, é muito importante não deixar a tabela encher muito:

[1] Baseado em Sedgewick, R. Algorithms in C, third edition, Addison-Wesley. 1998.

Sondagem linear e Hashing duplo[1]

Sondagem linear - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.5	2.0	3.0	5.5
sem sucesso	2.5	5.0	8.5	55.5

Hashing duplo - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.4	1.6	1.8	2.6
sem sucesso	1.5	2.0	3.0	5.5

De qualquer forma, é muito importante não deixar a tabela encher muito:

- Você pode aumentar o tamanho da tabela dinamicamente

[1] Baseado em Sedgewick, R. Algorithms in C, third edition, Addison-Wesley. 1998.

Sondagem linear e Hashing duplo[1]

Universidade Federal do Ceará
Campus Quixadá

Sondagem linear - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.5	2.0	3.0	5.5
sem sucesso	2.5	5.0	8.5	55.5

Hashing duplo - tempo de busca médio

n/M	$1/2$	$2/3$	$3/4$	$9/10$
com sucesso	1.4	1.6	1.8	2.6
sem sucesso	1.5	2.0	3.0	5.5

De qualquer forma, é muito importante não deixar a tabela encher muito:

- Você pode aumentar o tamanho da tabela dinamicamente
- Porém, precisa fazer um rehash de cada elemento para a nova tabela

[1] Baseado em Sedgewick, R. Algorithms in C, third edition, Addison-Wesley. 1998.

Conclusão

Hashing é uma boa estrutura de dados para

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

- Sondagem linear é o mais rápido se a tabela for esparsa

Conclusão

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

- Sondagem linear é o mais rápido se a tabela for esparsa
- Hashing duplo usa melhor a memória

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

- Sondagem linear é o mais rápido se a tabela for esparsa
- Hashing duplo usa melhor a memória
    - mas gasta mais tempo para computar a segunda função de hash

Conclusão

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

- Sondagem linear é o mais rápido se a tabela for esparsa
- Hashing duplo usa melhor a memória
  - mas gasta mais tempo para computar a segunda função de hash
- Encadeamento exterior é mais fácil de implementar

Conclusão

Hashing é uma boa estrutura de dados para

- inserir, remover e buscar dados pela sua chave rapidamente
- com uma boa função de hashing, essas operações levam tempo $O(1)$
- mas não é boa se quisermos fazer operação relacionadas a ordem das chaves

Escolhendo a implementação:

- Sondagem linear é o mais rápido se a tabela for esparsa
- Hashing duplo usa melhor a memória
  - mas gasta mais tempo para computar a segunda função de hash
- Encadeamento exterior é mais fácil de implementar
  - Usa memória a mais para os ponteiros

FIM