"... o anti-intelectualismo tem sido uma constante ... alimentado pela falsa noção de que a democracia significa que 'minha ignorância é tão boa quanto o seu conhecimento' " (Isaac Asimov).

Endereçamento Aberto

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Endereçamento aberto

Ideia: usar a própria Tabela Hash para tratar colisões, sem a necessidade de uma lista encadeada.

Cada *slot* vai conter um elemento, ou *NULO*.

Inserir

Calcular a “primeira opção” de *slot* para inserir o elemento.

Se *slot* vazio, inserir.

Senão

Calcular a “segunda opção” de *slot* para inserir o elemento.

Se *slot* vazio, inserir.

...

Continuar o processo até encontrar um *slot* vazio ou até concluir que a Tabela está cheia.

Buscar

Calcular a “primeira opção” de *slot* onde a chave *k* pode estar.

Se a chave se encontra na posição, retornar o conteúdo.

Senão

Calcular a “segunda opção” de *slot* onde a chave *k* pode estar.

Se a chave se encontra na posição, retornar o conteúdo.

...

Continuar o processo até encontrar a chave, ou até encontrar NULO (chave não encontrada).

Vantagens e Desvantagens

Quais vantagens e desvantagens quando comparado com o método de encadeamento?

Vantagens e Desvantagens

Quais vantagens e desvantagens quando comparado com o método de encadeamento?

+ Evitar o uso de ponteiros.
+ Economizar memória e talvez simplificar o problema.

    A memória economizada pode ser usada para criar uma Tabela Hash maior, mitigando colisões.

- Quando a Tabela está cheia, a única opção é construir uma Tabela maior (*realloc*).
- As operações de inserção podem custar *O(n)* no pior caso.

Sondar

Para inserir um item é necessário **sondar** (***probe***) os *slots* da Tabela.

Continuar sondagem até encontrar uma posição vazia.

A função de hash possui dois parâmetros:

h(k,i)

Onde:

k é a chave, e *i* é um número de sondagem, que inicia em 0.

Exemplo

Verificar se *h(k,0)* está vazio.

Se não estiver, verificar *h(k,1)*.

Se não estiver, verificar *h(k,2)*.

...

Se não estiver, verificar *h(k,m-1)*.

Propriedades

Além das propriedades discutidas em aulas passadas, qual propriedade extra é fundamental na função *h(k,i)*?

Propriedades

Além das propriedades discutidas em aulas passadas, qual propriedade extra é fundamental na função *h(k,i)*?

Precisamos garantir que todas as posições da Tabela são acessíveis ao modificar o valor de *i*.

E pelo bem da eficiência $h(k,i) \neq h(k,j), \forall i,j < m, i \neq j$

Propriedades

Além das propriedades discutidas em aulas passadas, qual propriedade extra é fundamental na função *h(k,i)*?

Precisamos garantir que todas as posições da Tabela são acessíveis ao modificar o valor de *i*.

E pelo bem da eficiência $h(k,i) \neq h(k,j), \forall i,j < m, i \neq j$

Ao executar a função usando *i=0, 1, ..., m-1* , garantimos que passamos por todos os *m* slots.

Faça você mesmo

Considerando a função de hash *h(k,i)*, e um objeto *x*, onde *x.k* é a chave.

Como podem ser implementadas as seguintes funções?

```
inserir(T,x)
```

```
buscar(T,k)
```

Inserir

```
inserir(T,x)
    i = 0
    faça
        q = h(x.k,i)
        se T[q] == NULO
            T[q] = x
            retorne q
        i = i + 1
    enquanto i < T.m
    erro "Tabela Cheia"
```

Buscar

```
buscar(T,k)
    i = 0
    q = h(k,i)
    enquanto T[q] != NULO E i < T.m
        se T[q].k == k
            retorne q
        i = i + 1
        q = h(k,i)
    retorne NULO
```

Excluir

Como pode ficar o algoritmo de exclusão? Quais as dificuldades extras?

```
excluir(T,q)
```

Excluir

Como pode ficar o algoritmo de exclusão? Quais as dificuldades extras?

```
excluir(T,q)
    T[q] = NULO
```

Excluir

Como pode ficar o algoritmo de exclusão? Quais as dificuldades extras?

```
excluir(T,q)
    T[q] = NULO
```

O algoritmo de busca deixará de funcionar. Um elemento de chave *k* que precisou ser alocado em outra posição quando *q* estava ocupado não será acessível.

```
buscar(T,k)
    i = 0
    q = h(k,i)
    enquanto T[q] != NULO E i < T.m E
        se T[q].k = k
            retorne q
        i = i + 1
        q = h(k,i)
    retorne NULO
```

Excluir

Uma possível solução é marcar as posições excluídas com um valor especial DELETADO.

```
excluir(T,q)
    T[q] = DELETADO
```

Problemas?

```
buscar(T,k)
    i = 0
    q = h(k,i)
    enquanto T[q] != NULO E i < T.m E
        se T[q].k = k
            retorne q
        i = i + 1
        q = h(k,i)
    retorne NULO
```

Excluir

Uma possível solução é marcar as posições excluídas com um valor especial DELETADO.

```
excluir(T,q)
    T[q] = DELETADO
```

Problemas?

O algoritmo de busca deixa de depender do fator de carga α.

Pode ser necessário passar por todas *m* posições da tabela para concluir que a chave não existe.

Mesmo em uma **Tabela Vazia!**

Excluir

Uma possível solução é marcar as posições excluídas com um valor especial DELETADO.

```
excluir(T,q)
    T[q] = DELETADO
```

Problemas?

O algoritmo de busca deixa de depender do fator de carga α.

Pode ser necessário passar por todas *m* posições da tabela para concluir que a chave não existe.

Mesmo em uma **Tabela Vazia!**

Por conta disso, quando as exclusões são frequentes, é comum a utilização de **encadeamento**.

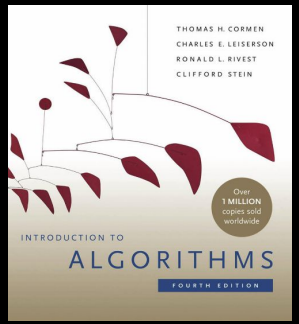
Função de Hash

Idealmente, a função *h(k,i)* deve ser uma função de hash de permutação uniforme e independente.

Também chamada de função de hash uniforme.

A sequência de *slots* sondados deve ser qualquer uma das *m!* permutações das *[0, 1, ..., m-1]* posições, onde todas as permutações possíveis têm a mesma probabilidade de acontecer.

Veja detalhes em Cormen et al. (2022).



Hash Duplo

Um hash duplo é uma aproximação comum para uma função de hash uniforme.

Mas não é uniforme, já que pode gerar no máximo m^2 sequências de sondagem distintas, ao invés de $m!$.

Forma geral:

$$h(k, i) = (h_1(k) + ih_2(k)) \ mod \ m$$

Onde $h_1(k)$ e $h_2(k)$ são funções de **hash auxiliares**.

Hash Duplo

$$h(k, i) = (h_1(k) + ih_2(k))\ mod\ m$$

Para garantir que todos os slots são acessíveis, $h_2(k)$ e m devem ser primos relativos.

Não existe um inteiro maior do que 1 que divida simultâneamente $h_2(k)$ e m.

Hash Duplo

$$h(k, i) = (h_1(k) + ih_2(k)) \ mod \ m$$

Para garantir que todos os slots são acessíveis, $h_2(k)$ e m devem ser primos relativos.

Não existe um inteiro maior do que 1 que divida simultâneamente $h_2(k)$ e m.

Algumas formas de se garantir isso:

- Fazer com que m seja uma potência de 2, e h_2 sempre produzir números ímpares.
- Fazer com que m seja um número primo, e h_2 sempre produzir um inteiro positivo menor que m.

Hash Duplo

Fazer com que m seja um número primo, e h_2 produza um inteiro positivo menor que m.

Exemplo: $h(k,i) = (h_1(k) + ih_2(k))\ mod\ m$

$h_1(k) = k\ mod\ m$

$h_2(k) = 1 + (k\ mod\ m')$

Onde *m'* é um valor um pouco menor que *m*.

Por exemplo, *m' = m-1*.

Exemplo

Considere m = 7.

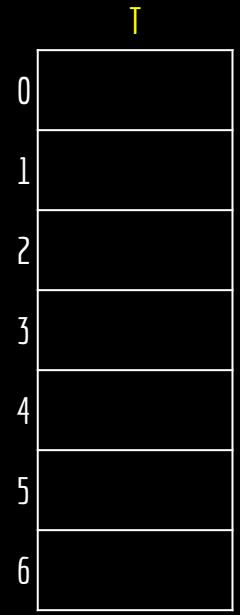
m' = m -1 = 6.

k = 27.

$$h(k,i) = (h_1(k) + ih_2(k))\ mod\ m$$

$$h_1(k) = k\ mod\ m$$

$$h_2(k) = 1 + (k\ mod\ m')$$



Exemplo

Considere m = 7.

m' = m -1 = 6.

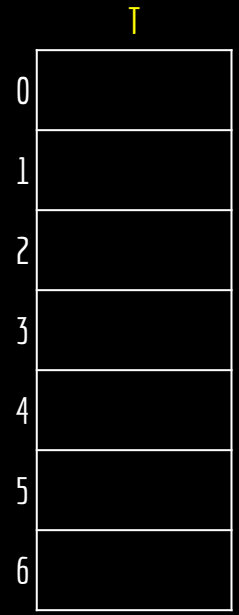
k = 27.

$$h(k,i) = (h_1(k) + ih_2(k))\ mod\ m$$

$$h_1(k) = k\ mod\ m$$

$$h_2(k) = 1 + (k\ mod\ m')$$

$h_1(k) = 6$
$h_2(k) = 4$



Exemplo

Considere m = 7.

m' = m -1 = 6.

k = 27.

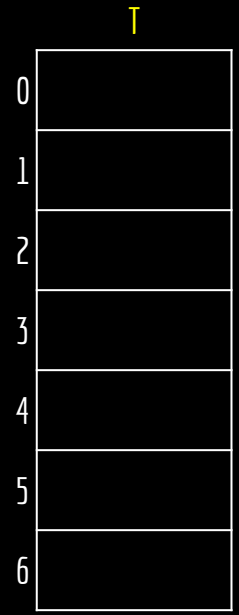
$$h(k,i) = (h_1(k) + ih_2(k))\ mod\ m$$

$$h_1(k) = k\ mod\ m$$

$$h_2(k) = 1 + (k\ mod\ m')$$

$h_1(k) = 6$
$h_2(k) = 4$

h(k,0) = 6
h(k,1) = 3
h(k,2) = 0
h(k,3) = 4
h(k,4) = 1
h(k,5) = 5
h(k,6) = 2



Sondagem Linear

Um caso especial do hash duplo é a sondagem linear.

Quando uma colisão é detectada, buscamos a posição vizinha de forma linear:

$$h(k, i) = (h_1(k) + i) \ mod \ m$$

Nesse caso, temos apenas *m sequências* distintas.

Lembre-se que, idealmente, todas as *m!* sequências possíveis deveriam ser igualmente prováveis.

Sondagem Linear

Apesar de longe de um hash uniforme, a sondagem linear tem vantagens em arquiteturas com hierarquias de memórias (registradores, caches, memórias DRAM, ...).

Por quê?

Sondagem Linear

Apesar de longe de um hash uniforme, a sondagem linear tem vantagens em arquiteturas com hierarquias de memórias (registradores, caches, memórias DRAM, ...).

A memória é geralmente carregada para a cache em blocos da memória principal para a cache.

A sondagem linear pode tirar vantagem disso.

Existe uma alta probabilidade do próximo slot a ser analisado estar na cache (*cache hit*).

Excluir com Sondagem Linear

Com sondagem linear podemos fazer melhor do que marcar os *slots* como *DELETADO* em uma exclusão.

Atenção: essa abordagem funciona apenas para a sondagem linear.

Dado que

$$q = h(k, i) = (h_1(k) + i) \ mod \ m$$

Desejamos a inversa de *h(k,i)*, tal que

$$g(k, q) = i$$

Excluir com Sondagem Linear

Com sondagem linear podemos fazer melhor do que marcar os *slots* como *DELETADO* em uma exclusão.

Atenção: essa abordagem funciona apenas para a sondagem linear.

Dado que

$$q = h(k, i) = (h_1(k) + i) \ mod \ m$$

Desejamos a inversa de *h(k,i)*, tal que

$$g(k, q) = i$$

A função é definida como

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Algoritmo

```
excluirSondagemLinear(T,q)
    enquanto verdade //loop infinito
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

T

0	
1	
2	82
3	43
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q
3
```

T

0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'
3    3
```

T

0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'
3    4
```

T

0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
3    4    74
```

T

0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
3    4    74
```

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
3    4    74
```

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
3    4    74
```

```
g(k',q) = 9
g(k',q') = 0
```

Quando 74 foi inserido, a iteração inicial era menor que a iteração de onde ele se encontra atualmente? Se sim, isso foi por conta de uma colisão!

$$i = g(k, q) = (q - h_1(k))\ mod\ m$$

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
3    5    74
```

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
3    5    93
```

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
3    5    93
```

	T
0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k))\ mod\ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
3    5    93
```

```
g(k',q) = 0
g(k',q') = 2
```

$$i = g(k,q) = (q - h_1(k))\ mod\ m$$

T

0	
1	
2	82
3	
4	74
5	93
6	92
7	
8	18
9	38

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
3	5	93

	T
0	
1	
2	82
3	93
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
5    5    93
```

	T
0	
1	
2	82
3	93
4	74
5	93
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k))\ mod\ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	5	93

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
5    5    93
```

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

```
excluirSondagemLinear(T,3)
q    q'   k'
5    6    93
```

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	6	92

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	6	92

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k)$ = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	6	92

g(k',q) = 3
g(k',q') = 4

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k))\ mod\ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	7	92

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

m = 10
$h_1(k) = k \bmod 10$
Inseridos na tabela:
74,43,93,18,82,38,92

excluirSondagemLinear(T,3)

q	q'	k'
5	7	NULO

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
5    7    NULO
```

T

0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Teste de Mesa

```
excluirSondagemLinear(T,q)
    enquanto verdade
        T[q] = NULO
        q' = q
        repita
            q' = (q' + 1) mod m
            k' = T[q']
            se k' == NULO
                retorne
        enquanto g(k',q) ≥ g(k',q')
        T[q] = k'
        q = q'
```

```
m = 10
h1(k) = k mod 10
Inseridos na tabela:
74,43,93,18,82,38,92
```

```
excluirSondagemLinear(T,3)
q    q'   k'
5    7    NULO
```

	T
0	
1	
2	82
3	93
4	74
5	
6	92
7	
8	18
9	38

$$i = g(k, q) = (q - h_1(k)) \ mod \ m$$

Dica

Cuidado com a operação de **módulo**.

Em C, a operação % é o resto da divisão.

Dá na mesma que a operação módulo para números Naturais.

Difere para valores negativos! **Pesquise!**

Exemplo:

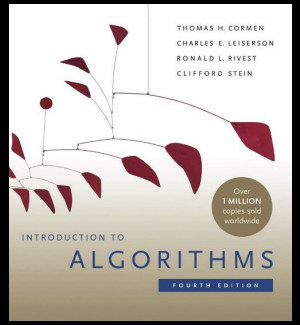
- 5 mod 3 = 1
-5 % 3 = -2 <- Resposta do C. Resto da divisão.

Exercícios

1. Implemente os algoritmos discutidos em aula em C.

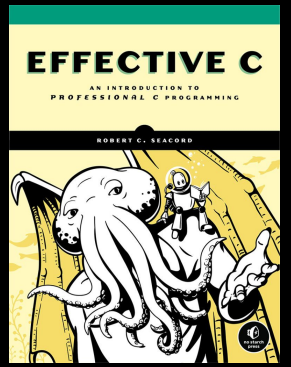
Referências

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 4a ed. 2022.



Estrutura de Dados e Algoritmos em C++. A. Drozdek. 4a ed. 2016.

Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.



Licença