MAC0323 Algoritmos e Estruturas de Dados II

Edição 2020 – 2

AULA 2

Análise amortizada

Fonte: https://www.europosters.pt/telas/

CLRS 17

Contador binário

Incrementa de 1 o número binário representado por $\mathtt{a}[0 \mathinner{.\,.} \mathtt{k}{-}1]$.

Entrada:

$\mathtt{k}{-}1$		3	2	1	0	
0	1	0	1	1	1	a

Contador binário

Incrementa de 1 o número binário representado por $\mathtt{a}[0\,.\,.\,\mathtt{k}{-}1]$.

Entrada:

$\mathtt{k}{-}1$		3	2	1	0	
0	1	0	1	1	1	a

Saída:

$\mathtt{k}{-}1$		3	2	1	0	
0	1	1	0	0	0	a

Contador binário

Incrementa de 1 o número binário representado por $a[0\,..\,k-1]$.

Entrada:

$k-1$		3	2	1	0	
0	1	0	1	1	1	a

Saída:

$k-1$		3	2	1	0	
0	1	1	0	0	0	a

INCREMENT (a, k)

$i \leftarrow 0$
enquanto $i < k$ **e** $a[i] = 1$ **faça**
    $a[i] \leftarrow 0$
    $i \leftarrow i + 1$
se $i < k$
    então $a[i] \leftarrow 1$

Consumo de tempo

linha	consumo de todas as execuções da linha

Consumo de tempo

linha	consumo de todas as execuções da linha
1	$\Theta(1)$
2	$O(\mathtt{k})$
3	$O(\mathtt{k})$
4	$O(\mathtt{k})$
5	$\Theta(1)$
6	$O(1)$

Consumo de tempo

linha	consumo de todas as execuções da linha
1	$\Theta(1)$
2	$O(k)$
3	$O(k)$
4	$O(k)$
5	$\Theta(1)$
6	$O(1)$
total	$O(k) + \Theta(1) = O(k)$

“Custo” $=$ consumo de tempo

$=$ número de bits alterados $= O(k)$

Sequência de $\mathtt{n}$ chamadas

a começa zerado.

$$\underbrace{\textsc{Incr}\quad \textsc{Incr}\quad \cdots\quad \textsc{Incr}\quad \textsc{Incr}\quad \textsc{Incr}}_{\mathtt{n}}$$

Consumo de tempo é $\mathrm{O}(\mathtt{nk})$.

Sequência de $\mathtt{n}$ chamadas

$\mathtt{a}$ começa zerado.

$$\underbrace{\textsc{Incr}\quad \textsc{Incr}\quad \cdots\quad \textsc{Incr}\quad \textsc{Incr}\quad \textsc{Incr}}_{\mathtt{n}}$$

Consumo de tempo é $\mathrm{O}(\mathtt{n}\mathtt{k})$.

EXAGERO!

Exemplo

$n = 16$

a					
5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

$k = 6$

a					
5	4	3	2	1	0
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0

Exemplo

$\mathtt{n} = 16$

a					
5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

$\mathtt{k} = 6$

a					
5	4	3	2	1	0
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0

$\mathtt{a}[0]$ muda $\mathtt{n}$ vezes

Exemplo

$n = 16$ | $k = 6$

a							a					
5	4	3	2	1	0		5	4	3	2	1	0
0	0	0	0	0	0		0	0	1	0	0	0
0	0	0	0	0	1		0	0	1	0	0	1
0	0	0	0	1	0		0	0	1	0	1	0
0	0	0	0	1	1		0	0	1	0	1	1
0	0	0	1	0	0		0	0	1	1	0	0
0	0	0	1	0	1		0	0	1	1	0	1
0	0	0	1	1	0		0	0	1	1	1	0
0	0	0	1	1	1		0	0	1	1	1	1
							0	1	0	0	0	0

$a[0]$	muda	n	vezes
$a[1]$	"	$\lfloor n/2 \rfloor$	"

Exemplo

$n = 16$

5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

$k = 6$

5	4	3	2	1	0
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0

$a[0]$	muda	n	vezes
$a[1]$	"	$\lfloor n/2 \rfloor$	"
$a[2]$	"	$\lfloor n/4 \rfloor$	"

Exemplo

$\mathtt{n} = 16$

a					
5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1

$\mathtt{k} = 6$

a					
5	4	3	2	1	0
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0

$\mathtt{a}[0]$	muda	$\mathtt{n}$	vezes
$\mathtt{a}[1]$	"	$\lfloor \mathtt{n}/2 \rfloor$	"
$\mathtt{a}[2]$	"	$\lfloor \mathtt{n}/4 \rfloor$	"
$\mathtt{a}[3]$	"	$\lfloor \mathtt{n}/8 \rfloor$	"

Análise agregada

Custo total:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor$$

Análise agregada

Custo total:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor \; < \; n \sum_{i=0}^{\infty} \frac{1}{2^i}$$

Análise agregada

Custo total:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor \; < \; n \sum_{i=0}^{\infty} \frac{1}{2^i} \; = \; 2n \; = \; \Theta(n)$$

Análise agregada

Custo total:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor \; < \; n \sum_{i=0}^{\infty} \frac{1}{2^i} \; = \; 2n \; = \; \Theta(n)$$

Custo amortizado ($=$ custo médio)
de uma operação:

$$\frac{2n}{n} \; = \; \Theta(1)$$

Análise agregada

Custo total:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n = \Theta(n)$$

Custo amortizado (= custo médio)
de uma operação:

$$\frac{2n}{n} = \Theta(1)$$

Este foi o **método agregado** de análise:
soma os custos de todas as operações para
determinar o custo amortizado de cada operação.

Custo amortizado

O **custo amortizado** de uma operação é
o custo médio da operação quando considerada
em uma sequência de operações do ADT.

Conclusões

$\boxed{\text{O consumo de tempo de uma sequência de } \mathtt{n} \text{ execuções do algoritmo } \textsc{Increment} \text{ é } \Theta(\mathtt{n}).}$

Conclusões

O consumo de tempo de uma sequência de $\mathtt{n}$ execuções do algoritmo INCREMENT é $\Theta(\mathtt{n})$.

O consumo de tempo amortizado do algoritmo INCREMENT é $\Theta(1)$.

Método de análise contábil

a começa zerado.

Pague $2 para mudar a[i] de $0 \to 1$

$0 para mudar a[i] de $1 \to 0$

Método de análise contábil

a começa zerado.

Pague $\$2$ para mudar a[i] de $0 \to 1$

$\$0$ para mudar a[i] de $1 \to 0$

a[i] muda de $0 \to 1$ $\begin{cases} \$1 \text{ é pago pela operação} \\ \$1 \text{ é colocado na poupança.} \end{cases}$

a[i] muda de $1 \to 0$:
paga com poupança do i-ésimo bit.

Método de análise contábil

a começa zerado.

Pague $2 para mudar a[i] de $0 \to 1$

$0 para mudar a[i] de $1 \to 0$

a[i] muda de $0 \to 1$ $\begin{cases} \$1 \text{ é pago pela operação} \\ \$1 \text{ é colocado na poupança.} \end{cases}$

a[i] muda de $1 \to 0$:
paga com poupança do i-ésimo bit.

Custo amortizado por chamada de INCREMENT:
$\leq$ $2 (no máximo uma mudança $0 \to 1$ é feita).

Método de análise contábil

$\texttt{a}$ começa zerado.

$$\begin{array}{lll} \text{Pague} & \$2 \text{ para mudar } \texttt{a}[\texttt{i}] \text{ de } 0 \to 1 \\ & \$0 \text{ para mudar } \texttt{a}[\texttt{i}] \text{ de } 1 \to 0 \end{array}$$

Como \$ armazenado nunca é negativo, uma sequência de $\texttt{n}$ chamadas de INCREMENT custa

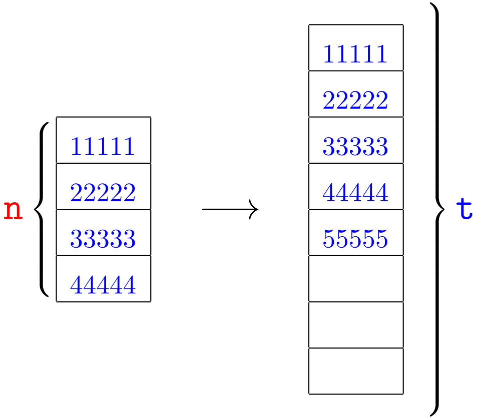
$$\begin{aligned} \text{soma custos reais} &\leq \text{soma custos amortizados} \\ &= 2\texttt{n} \\ &= \mathrm{O}(\texttt{n}) \end{aligned}$$

Tabelas dinâmicas

Fonte: https://twitter.com/MinionPostDoc

CLRS 17

Tabelas dinâmicas



$\mathtt{n}[\mathtt{T}]=$ número de itens $\qquad \mathtt{t}[\mathtt{T}]=$ tamanho de $\mathtt{T}$

Inicialmente $\mathtt{n}[\mathtt{T}] = \mathtt{t}[\mathtt{T}] = 0$.

Inserção

TABLE-INSERT $(\mathtt{T}, x)$ $\triangleright$ Insere x na tabela $\mathtt{T}$

se $\mathtt{t}[\mathtt{T}] = 0$
  então aloque $tabela[\mathtt{T}]$ com 1 posição
    $\mathtt{t}[\mathtt{T}] \leftarrow 1$

Inserção

TABLE-INSERT $(\mathtt{T}, x)$ $\quad \triangleright$ Insere x na tabela $\mathtt{T}$

se $\mathtt{t}[\mathtt{T}] = 0$

  então aloque $\mathit{tabela}[\mathtt{T}]$ com 1 posição

    $\mathtt{t}[\mathtt{T}] \leftarrow 1$

se $\mathtt{n}[\mathtt{T}] = \mathtt{t}[\mathtt{T}]$

  então aloque $\mathit{nova\text{-}tabela}$ com $2\,\mathtt{t}[\mathtt{T}]$ pos.

    insira itens da $\mathit{tabela}[\mathtt{T}]$ na $\mathit{nova\text{-}tabela}$

    $\mathtt{t}[\mathit{nova\text{-}tabela}] \leftarrow 2\,\mathtt{t}[\mathtt{T}]$

    libere $\mathit{tabela}[\mathtt{T}]$

    $\mathit{tabela}[\mathtt{T}] \leftarrow \mathit{nova\text{-}tabela}$

Inserção

TABLE-INSERT $(\mathtt{T}, x)$ $\triangleright$ Insere x na tabela $\mathtt{T}$

1 **se** $\mathtt{t}[\mathtt{T}] = 0$

2 **então** aloque $\mathit{tabela}[\mathtt{T}]$ com 1 posição

3 $\mathtt{t}[\mathtt{T}] \leftarrow 1$

4 **se** $\mathtt{n}[\mathtt{T}] = \mathtt{t}[\mathtt{T}]$

5 **então** aloque $\mathit{nova\text{-}tabela}$ com $2\,\mathtt{t}[\mathtt{T}]$ pos.

6 insira itens da $\mathit{tabela}[\mathtt{T}]$ na $\mathit{nova\text{-}tabela}$

7 $\mathtt{t}[\mathit{nova\text{-}tabela}] \leftarrow 2\,\mathtt{t}[\mathtt{T}]$

8 libere $\mathit{tabela}[\mathtt{T}]$

9 $\mathit{tabela}[\mathtt{T}] \leftarrow \mathit{nova\text{-}tabela}$

10 insira x na $\mathit{tabela}[\mathtt{T}]$

11 $\mathtt{n}[\mathtt{T}] \leftarrow \mathtt{n}[\mathtt{T}] + 1$

Custo $=$ número de inserções elementares (linhas 6 e 10)

Sequência de $\mathtt{m}$ Table-Inserts

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\text{ op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\text{ op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\text{ op}} \mathtt{T}_{\mathtt{m}}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Sequência de $\mathtt{m}$ Table-Inserts

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\ \text{op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\ \text{op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\ \text{op}} \mathtt{T}_{\mathtt{m}}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Custo real da i^{a} operação:

$$c_i = \begin{cases} 1 & \text{se há espaço} \\ \mathtt{n}_i & \text{se tabela cheia,} \end{cases}$$

onde $\mathtt{n}_i = $ valor de $\mathtt{n}[\mathtt{T}]$ depois da i^{a} operação
$= i$.

Sequência de $\mathtt{m}$ Table-Inserts

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\text{ op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\text{ op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\text{ op}} \mathtt{T}_{\mathtt{m}}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Custo real da i^{a} operação:

$$c_i = \begin{cases} 1 & \text{se há espaço} \\ \mathtt{n}_i & \text{se tabela cheia,} \end{cases}$$

onde $\mathtt{n}_i$ = valor de $\mathtt{n}[\mathtt{T}]$ depois da i^{a} operação
= i.

Custo de uma operação = $\mathrm{O}(\mathtt{m})$.

Sequência de $\mathtt{m}$ Table-Inserts

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\text{ op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\text{ op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\text{ op}} \mathtt{T}_{\mathtt{m}}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Custo real da i^{a} operação:

$$c_i = \begin{cases} 1 & \text{se há espaço} \\ \mathtt{n}_i & \text{se tabela cheia,} \end{cases}$$

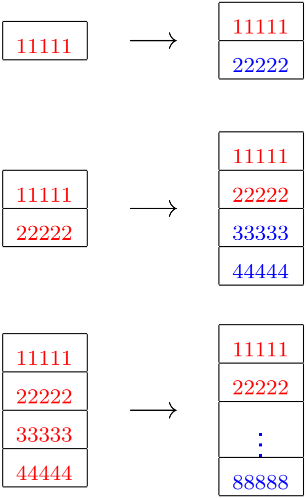
onde $\mathtt{n}_i$ = valor de $\mathtt{n}[\mathtt{T}]$ depois da i^{a} operação
= i.

Custo de uma operação = $\mathrm{O}(\mathtt{m})$.

Custo das $\mathtt{m}$ operações = $\mathrm{O}(\mathtt{m}^2)$. Exagero!

Exemplo

n[T] (operação)	t[T]	custo
1	1	1
2	2	1+1
3	4	1+2
4	4	1
5	8	1+4
6	8	1
7	8	1
8	8	1
9	16	1+8
10	16	1
16	16	1
17	32	1+16
33	64	1+32



Custo amortizado

Custo total: Para $k = \lfloor \lg(\mathtt{m} - 1) \rfloor$,

$$\sum_{i=1}^{\mathtt{m}} c_i \;=\; \mathtt{m} + \sum_{i=0}^{k} 2^i$$

Custo amortizado

Custo total: Para $k = \lfloor \lg(\mathtt{m} - 1) \rfloor$,

$$\sum_{i=1}^{\mathtt{m}} c_i \;=\; \mathtt{m} + \sum_{i=0}^{k} 2^i \;=\; \mathtt{m} + 2^{k+1} - 1$$

Custo amortizado

Custo total: Para $k = \lfloor \lg(\mathtt{m} - 1) \rfloor$,

$$\sum_{i=1}^{\mathtt{m}} c_i \;=\; \mathtt{m} + \sum_{i=0}^{k} 2^i \;=\; \mathtt{m} + 2^{k+1} - 1 \;<\; \mathtt{m} + 2\mathtt{m} - 1 \;<\; 3\mathtt{m}.$$

Custo amortizado:

$$\frac{3\mathtt{m}}{\mathtt{m}} \;=\; 3 \;=\; \Theta(1)$$

Conclusões

O custo de uma sequência de $\mathtt{m}$ execuções do algoritmo TABLE-INSERT é $\Theta(\mathtt{m})$.

Conclusões

O custo de uma sequência de $\mathtt{m}$ execuções do algoritmo TABLE-INSERT é $\Theta(\mathtt{m})$.

O custo amortizado do algoritmo TABLE-INSERT é $\Theta(1)$.

Método de análise agregada

- $\mathtt{m}$ operações consomem tempo $T(\mathtt{m})$.

Método de análise agregada

- $\mathtt{m}$ operações consomem tempo $T(\mathtt{m})$.
- custo médio de cada operação é $T(\mathtt{m})/\mathtt{m}$.
- custo amortizado de cada operação é $T(\mathtt{m})/\mathtt{m}$.

Método de análise agregada

- $\mathtt{m}$ operações consomem tempo $T(\mathtt{m})$.
- custo médio de cada operação é $T(\mathtt{m})/\mathtt{m}$.
- custo amortizado de cada operação é $T(\mathtt{m})/\mathtt{m}$.
- defeito: no caso de mais de um tipo de operação, o custo de cada tipo não é determinado separadamente.

Método de análise contábil

TABLE-INSERT (T, x)

$\quad$ *credito* $\leftarrow$ *credito* $+ 3$
se $\mathtt{t}[\mathrm{T}] = 0$
$\quad$ **então** aloque *tabela*$[\mathrm{T}]$ com 1 posição
$\qquad$ $\mathtt{t}[\mathrm{T}] \leftarrow 1$
se $\mathtt{n}[\mathrm{T}] = \mathtt{t}[\mathrm{T}]$
$\quad$ **então** aloque *nova-tabela* com $2\,\mathtt{t}[\mathrm{T}]$ pos.
$\qquad$ insira itens da *tabela*$[\mathrm{T}]$ na *nova-tabela*

$\qquad$ *custo* $\leftarrow$ *custo* $+ \mathtt{n}[\mathrm{T}]$
$\qquad$ libere *tabela*$[\mathrm{T}]$
$\qquad$ *tabela*$[\mathrm{T}] \leftarrow$ *nova-tabela*
$\qquad$ $\mathtt{t}[\mathrm{T}] \leftarrow 2\,\mathtt{t}[\mathrm{T}]$
insira x na *tabela*$[\mathrm{T}]$

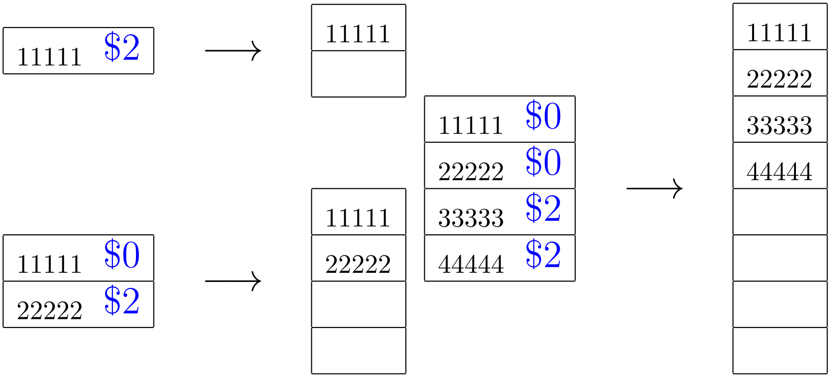
$\mathtt{n}[\mathrm{T}] \leftarrow \mathtt{n}[\mathrm{T}] + 1$
custo $\leftarrow$ *custo* $+ 1$

Método de análise contábil

Invariante: soma créditos $\geq$ soma custos reais

n[T]	t[T]	custo	crédito	saldo
1	1	1	3	2
2	2	$1+1$	3	3
3	4	$1+2$	3	3
4	4	1	3	5
5	8	$1+4$	3	3
6	8	1	3	5
7	8	1	3	7
8	8	1	3	9
9	16	$1+8$	3	3
10	16	1	3	5
16	16	1	3	17
17	32	$1+16$	3	3

Método de análise contábil



Método de análise contábil

Pague $\$1$ para inserir um novo elemento
Guarde $\$1$ para eventualmente mover o novo elemento
Guarde $\$1$ para mover um elemento que já está na tabela

Custo amortizado

por chamada de TABLE-INSERT: $\leq \$3$

Sequência de $\mathtt{m}$ chamadas de TABLE-INSERT.

Como \$ armazenado nunca é negativo,

$$\begin{aligned}\text{soma custos reais} &\leq \text{soma custos amortizados} \\ &= 3\mathtt{m} \\ &= \mathrm{O}(\mathtt{m})\end{aligned}$$

Método de análise contábil

- cada operação paga seu custo real

Método de análise contábil

- cada operação paga seu custo real
- cada operação recebe um certo número de créditos (chute de custo amortizado)

Método de análise contábil

- cada operação paga seu custo real
- cada operação recebe um certo número de créditos (chute de custo amortizado)
- balanço nunca pode ser negativo

$$\text{soma créditos} \;\geq\; \text{soma custos reais}$$

créditos não usados são guardados
para pagar operações futuras.

Método de análise contábil

- cada operação paga seu custo real
- cada operação recebe um certo número de créditos (chute de custo amortizado)
- balanço nunca pode ser negativo

  $$\text{soma créditos} \;\geq\; \text{soma custos reais}$$

  créditos não usados são guardados
  para pagar operações futuras.
- custo amortizado de cada tipo de operação pode ser determinado separadamente

Conclusões

O custo de uma sequência de $\mathtt{m}$ execuções do algoritmo TABLE-INSERT é $\Theta(\mathtt{m})$.

Conclusões

O custo de uma sequência de $\mathtt{m}$ execuções do algoritmo TABLE-INSERT é $\Theta(\mathtt{m})$.

O custo amortizado do algoritmo TABLE-INSERT é $\Theta(1)$.

Sequência de Insert e Delete

Sequência de operações Table-Insert e
Table-Delete

$$\underbrace{\text{I I D I I D I D D I D D I I}}_{m}$$

Custo total de uma sequência
de Table-Insert e Table-Delete?

Remoção

Remove um elemento x da tabela T

```
TABLE-DELETE (T, x)      ▷ supõe x na tabela[T]
    remova x da tabela[T]
    n[T] ← n[T] − 1
```

Remoção

Remove um elemento x da tabela T

```
TABLE-DELETE (T, x)    ▷ supõe x na tabela[T]
    remova x da tabela[T]
    n[T] ← n[T] − 1
    se n[T] < t[T]/2    ▷ tabela está "vazia"?
        então aloque nova-tabela com t[T]/2 pos.
              insira itens da tabela[T] na nova-tabela
              t[nova-tabela] ← t[T]/2
              n[nova-tabela] ← n[T]
              libere tabela[T]
              tabela[T] ← nova-tabela
```

Remoção

Remove um elemento x da tabela T

```
TABLE-DELETE (T, x)      ▷ supõe x na tabela[T]
1    remova x da tabela[T]
2    n[T] ← n[T] − 1
3    se n[T] < t[T]/2      ▷ tabela está "vazia"?
4         então aloque nova-tabela com t[T]/2 pos.
5               insira itens da tabela[T] na nova-tabela
6               t[nova-tabela] ← t[T]/2
7               n[nova-tabela] ← n[T]
8               libere tabela[T]
9               tabela[T] ← nova-tabela
```

Custo = número de remoções
e inserções elementares (linhas 1 e 5)

Sequência de Insert e Delete

Sequência de operações Table-Insert e
Table-Delete

$$\underbrace{\underbrace{\text{I} \quad \cdots \quad \text{I} \quad \text{I}}_{\mathtt{m}/2} \quad \text{I} \quad \text{D} \quad \text{D} \quad \text{I} \quad \text{I} \quad \text{D} \quad \text{D} \quad \text{I} \quad \text{I} \quad \text{D} \quad \text{D}}_{\mathtt{m}}$$

Se $\mathtt{m} = 4\,k$, então o custo total da sequência:

$$\Theta(2\,k) + k\,\Theta(2\,k) = \Theta\left(\frac{\mathtt{m}}{2}\right) + \frac{\mathtt{m}}{4}\,\Theta\left(\frac{\mathtt{m}}{2}\right) = \Theta(\mathtt{m}^2)$$

Remoção

Remove um elemento x da tabela T

```
TABLE-DELETE (T, x)    ▷ supõe x na tabela[T]
1   remova x da tabela[T]
2   n[T] ← n[T] − 1
3   se n[T] < t[T]/4    ▷ tabela está "vazia"?
4       então aloque nova-tabela com t[T]/2 pos.
5           insira itens da tabela[T] na nova-tabela
6           t[nova-tabela] ← t[T]/2
7           n[nova-tabela] ← n[T]
8           libere tabela[T]
9           tabela[T] ← nova-tabela
```

Custo = número de remoções
e inserções elementares (linhas 1 e 5)

Sequência de $\mathtt{m}$ operações

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\ \text{op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\ \text{op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\ \text{op}} \mathtt{T}_\mathtt{m}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Custo real da i^{a} operação se for TABLE-INSERT:

$$c_i = \begin{cases} 1 & \text{se há espaço} \\ \mathtt{n}_i & \text{se tabela cheia} \end{cases}$$

onde $\mathtt{n}_i$ = valor de $\mathtt{n}[\mathtt{T}]$ depois da i^{a} operação

Custo de uma operação $= \mathrm{O}(\mathtt{m})$

Sequência de $\mathtt{m}$ operações

$$\mathtt{T}_0 \xrightarrow{1^{\text{a}}\text{ op}} \mathtt{T}_1 \xrightarrow{2^{\text{a}}\text{ op}} \mathtt{T}_2 \longrightarrow \cdots \xrightarrow{\mathtt{m}^{\text{a}}\text{ op}} \mathtt{T}_{\mathtt{m}}$$

$\mathtt{T}_i$ = estado de $\mathtt{T}$ depois da i^{a} operação.

Custo real da i^{a} operação se for TABLE-DELETE:

$$c_i = \begin{cases} 1 & \text{se } \mathtt{n}_{i-1} > \mathtt{t}_{i-1}/4 \\ 1 + \mathtt{n}_i & \text{se } \mathtt{n}_{i-1} = \mathtt{t}_{i-1}/4 \end{cases}$$

onde $\mathtt{n}_i$ = valor de $\mathtt{n}[\mathtt{T}]$ depois da i^{a} operação e
$\mathtt{t}_i$ = valor de $\mathtt{t}[\mathtt{T}]$ depois da i^{a} operação

Custo de uma operação $= \mathrm{O}(\mathtt{m})$

Custo das $\mathtt{m}$ operações $= \mathrm{O}(\mathtt{m}^2)$ Exagero!

Conclusões

O custo de uma sequência de $\mathtt{m}$ execuções dos algoritmos TABLE-INSERT e TABLE-DELETE é $\Theta(\mathtt{m})$.

O custo amortizado dos algoritmos TABLE-INSERT e TABLE-DELETE é $\Theta(1)$.

Class ArrayList

https://docs.oracle.com/.../util/ArrayList.html

"... Each ArrayList instance has a capacity. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayList, its capacity grows automatically. The details of the growth policy are not specified beyond the fact that adding an element has constant amortized time cost. ..."

Listas em Python

"... CPython's lists are really variable-length arrays, ... The implementation uses a contiguous array of references to other objects, ...

This makes indexing a list `a[i]` *an operation whose cost is independent of the size of the list or the value of the index.*

When items are appended or inserted, the array of references is resized. Some cleverness is applied to improve ...; when the array must be grown, some extra space is allocated so the next few times don't require an actual resize."

Veja Design and History FAQ e Laurent Luce's Blog.