AULA 2

Análise amortizada

CLRS 17

Contador binário

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

INCREMENT $(A, \mathtt{k})$
1 $\mathtt{i} \leftarrow 0$
2 **enquanto** $\mathtt{i} < \mathtt{k}$ **e** $A[\mathtt{i}] = 1$ **faça**
3 $\quad A[\mathtt{i}] \leftarrow 0$
4 $\quad \mathtt{i} \leftarrow \mathtt{i} + 1$
5 **se** $\mathtt{i} < \mathtt{k}$
6 $\quad$ **então** $A[\mathtt{i}] \leftarrow 1$

Contador binário

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

INCREMENT $(A, \mathtt{k})$
1 $\mathtt{i} \leftarrow 0$
2 **enquanto** $\mathtt{i} < \mathtt{k}$ **e** $A[\mathtt{i}] = 1$ **faça**
3 $\quad A[\mathtt{i}] \leftarrow 0$
4 $\quad \mathtt{i} \leftarrow \mathtt{i} + 1$
5 **se** $\mathtt{i} < \mathtt{k}$
6 $\quad$ **então** $A[\mathtt{i}] \leftarrow 1$

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 $A[0\,.\,.\,\mathtt{k}{-}1]$.

INCREMENT $(A, \mathtt{k})$
1 $\mathtt{i} \leftarrow 0$
2 **enquanto** $\mathtt{i} < \mathtt{k}$ **e** $A[\mathtt{i}] = 1$ **faça**
3 $\quad A[\mathtt{i}] \leftarrow 0$
4 $\quad \mathtt{i} \leftarrow \mathtt{i} + 1$
5 **se** $\mathtt{i} < \mathtt{k}$
6 $\quad$ **então** $A[\mathtt{i}] \leftarrow 1$

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

Consumo de tempo

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

"Custo" =
consumo de tempo = número de bits alterados
$= \mathrm{O}(k)$

Sequência de n chamadas

A começa zerado.

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

Consumo de tempo é $O(nk)$

Sequência de n chamadas

A começa zerado.

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

Consumo de tempo é $O(nk)$

EXAGERO!

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

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

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

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

$A[0]$	muda	$\mathtt{n}$	vezes
$A[1]$	"	$\lfloor \mathtt{n}/2 \rfloor$	"

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

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

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
$A[1]$	"	$\lfloor n/2 \rfloor$	"
$A[2]$	"	$\lfloor n/4 \rfloor$	"
$A[3]$	"	$\lfloor n/8 \rfloor$	"

Análise agregada

Custo total:

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

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

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

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 $0 \to 1$ $\begin{cases} \text{\$1 é pago pela operação} \\ \text{\$1 é colocado na poupança.} \end{cases}$

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

Método de análise contábil

A começa zerado.

Pague $2 para mudar $A[\mathtt{i}]$ de $0 \to 1$

$0 para mudar $A[\mathtt{i}]$ de $1 \to 0$

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

$A[\mathtt{i}]$ 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

A começa zerado.

Pague $2 para mudar $A[\mathtt{i}]$ de $0 \to 1$

$0 para mudar $A[\mathtt{i}]$ de $1 \to 0$

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

$A[\mathtt{i}]$ muda de $1 \to 0$:
paga com poupança do i-ésimo bit.

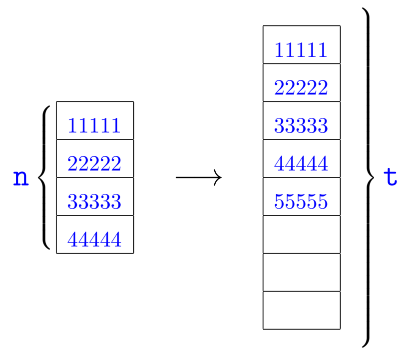
Custo amortizado por chamada de INCREMENT: $\leq$ $2 (no máximo uma mudança $0 \to 1$ é feita).

Como $ armazenado nunca é negativo, uma sequência de n chamadas de INCREMENT custa

Tabelas dinâmicas

CLRS 17

Tabelas dinâmicas



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

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

Inserção

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

1 **se** $\mathtt{t}[\mathtt{T}] = 0$
2 **então** aloque *tabela*[T] com 1 posição
3 $\mathtt{t}[\mathtt{T}] \leftarrow 1$
4 **se** $\mathtt{n}[\mathtt{T}] = \mathtt{t}[\mathtt{T}]$
5 **então** aloque *nova-tabela* com $2\,\mathtt{t}[\mathtt{T}]$ pos.
6 insira itens da *tabela*[T] na *nova-tabela*
7 $\mathtt{t}[\textit{nova-tabela}] \leftarrow 2\,\mathtt{t}[\mathtt{T}]$
8 libere *tabela*[T]
9 *tabela*[T] $\leftarrow$ *nova-tabela*
10 insira x na *tabela*[T]
11 $\mathtt{n}[\mathtt{T}] \leftarrow \mathtt{n}[\mathtt{T}] + 1$

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

Sequência de m TABLE-INSERTs

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

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

Sequência de m TABLE-INSERTs

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

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

Custo real da i^{a} operação:

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

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

Sequência de m TABLE-INSERTs

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

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

Custo real da i^{a} operação:

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

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

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

Sequência de m TABLE-INSERTs

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

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

Custo real da i^{a} operação:

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

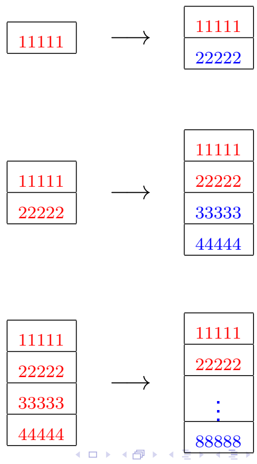
$$\begin{aligned} \text{onde } \mathtt{n}_i &= \text{valor de } \mathtt{n}[\mathtt{T}] \text{ depois da } i^{\text{a}} \text{ operação} \\ &= i. \end{aligned}$$

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

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

Exemplo

$\mathtt{n}[\mathtt{T}]$ (operação)	$\mathtt{t}[\mathtt{T}]$	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:

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

onde $k = \lfloor \lg(m-1) \rfloor$

Custo amortizado:

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

Conclusões

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

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

Método de análise agregada

- m operações consomem tempo $T(m)$.

Método de análise agregada

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

Método de análise agregada

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

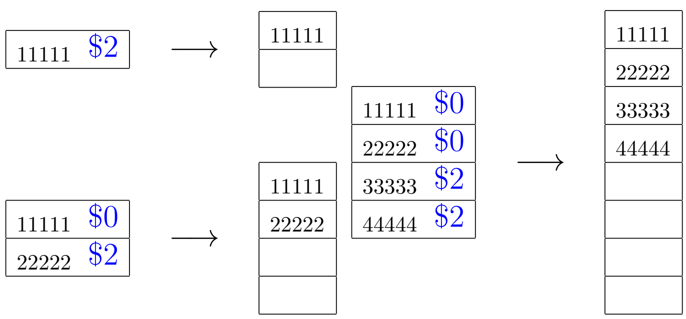
```
    credito ← credito + 3
se t[T] = 0
    então aloque tabela[T] com 1 posição
          t[T] ← 1
se n[T] = t[T]
    então aloque nova-tabela com 2 t[T] pos.
          insira itens da tabela[T] na nova-tabela
          custo ← custo + n[T]
          libere tabela[T]
          tabela[T] ← nova-tabela
          t[T] ← 2 t[T]
insira x na tabela[T]
n[T] ← n[T] + 1
custo ← 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 tabe

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

Sequência de m chamadas de TABLE-INSERT.

Como $ armazenado nunca é negativo,

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

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 m execuções do algoritmos TABLE-INSERT é $\Theta(m)$.

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

Sequencia de INSERT e DELETE

Sequência de operações TABLE-INSERT e TABLE-DELETE

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

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]

```
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

$$\underbrace{\underbrace{\mathsf{I}\ \cdots\ \mathsf{I\ \ I}}_{m/2}\ \mathsf{I\ \ D\ \ D\ \ I\ \ I\ \ D\ \ D\ \ I\ \ I\ \ D\ \ D}}_{m}$$

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

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

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 m operações

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

T_i = estado de T depois da i^{a} operação.

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

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

onde n_i = valor de n[T] depois da i^{a} operação

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

Sequência de m operações

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

T_i = estado de T depois da i^{a} operação.

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

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

onde n_i = valor de n[T] depois da i^{a} operação
e t_i = valor de t[T] depois da i^{a} operação

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

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

Conclusões

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

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

Pilhas redimensionáveis

Pilha (= stack) e sua API (PF)
1.3 Bags, Queues, and Stacks (SW)

Pilhas redimensionáveis

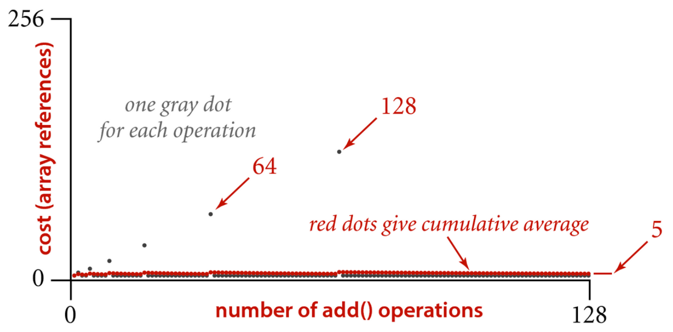
Considere a implementação de saco (Bag) em vetor com redimensionamento.

O custo amortizado da operação add() é muito baixo, pois cada ocorrência de uma execução lenta de add() é precedida por muitas ocorrências de execuções rápidas.

Pilhas redimensionáveis

`public class`	`Stack<Item>`	
	`Stack()`	construtor cria uma pilha de Items vazia
`void`	`push(Item item)`	insere item nesta pilha
`Item`	`pop()`	remove o Item mais recente desta pilha
`boolean`	`isEmpty()`	esta pilha está vazia?
`int`	`size()`	número de Items nesta pilha

Pilhas redimensionáveis



Amortized cost of adding to a RandomBag