# Análise amortizada

CLRS 17

# Análise amortizada

Propósito:
Analisar sequência de operações ou iterações onde
o pior caso individual não reflete o pior caso da sequência.

Consequência:
Melhora análises de pior caso que baseiem-se diretamente
no pior caso de uma operação/iteração e dão delimitação
frouxa para o tempo de pior caso da sequência.

Métodos:

- agregado
- por créditos
- potencial

# Odômetro binário

Considere um contador binário, inicialmente zerado, representado em vetor $A[0\,..\,n-1]$, onde cada $A[i] \in \{0, 1\}$.

Operação: incrementa.

# Odômetro binário

Considere um contador binário, inicialmente zerado, representado em vetor $A[0\,..\,n-1]$, onde cada $A[i] \in \{0,1\}$.

Operação: incrementa.

```
INCREMENTA (A, n)
1  i ← 0
2  enquanto i < n e A[i] = 1 faça
3      A[i] ← 0
4      i ← i + 1
5  se i < n
6      então A[i] ← 1
```

Consumo de tempo no pior caso: $\Theta(n)$

# Odômetro binário

| $i$ | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |
| 16 | 0 | 0 | 0 | 0 |

# Método agregado

Custo da volta completa é proporcional
ao número de vezes que os bits são alterados.

# Método agregado

Custo da volta completa é proporcional
ao número de vezes que os bits são alterados.

- bit 0 muda $2^n$ vezes
- bit 1 muda $2^{n-1}$ vezes
- bit 2 muda $2^{n-2}$ vezes
- ...
- bit $n-2$ muda 4 vezes
- bit $n-1$ muda 2 vezes

Total de alterações de bits: $\sum_{i=1}^{n} 2^i < 2 \cdot 2^n$.

Custo da volta completa: $\Theta(2^n)$.

Custo *amortizado* por INCREMENTA: $\Theta(1)$.

# Análise por créditos

Atribuímos 2 créditos por INCREMENTA.

```
INCREMENTA (A, n)
1  i ← 0
2  enquanto i < n e A[i] = 1 faça
3      A[i] ← 0
4      i ← i + 1
5  se i < n
6      então A[i] ← 1
```

Um é usado para pagar pela alteração da linha 6.

O outro fica armazenado sobre o bit alterado na linha 6.

O número de créditos armazenados em cada instante é
o número de bits que valem 1, logo é sempre não negativo.

# Análise por créditos

Atribuímos 2 créditos por INCREMENTA.

```
INCREMENTA (A, n)
1  i ← 0
2  enquanto i < n e A[i] = 1 faça
3      A[i] ← 0
4      i ← i + 1
5  se i < n
6      então A[i] ← 1
```

Um é usado para pagar pela alteração da linha 6.

O outro fica armazenado sobre o bit alterado na linha 6.

O número de créditos armazenados em cada instante é o número de bits que valem 1, logo é sempre não negativo.

Custo amortizado por INCREMENTA: 2

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

Note que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

Note que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

Note que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA.

Note que $c_i \leq 1 + t_i$ onde $t_i$ é o número de bits 1 consecutivos no final do contador $A$.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

Note que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA.

Note que $c_i \leq 1 + t_i$ onde $t_i$ é o número de
bits 1 consecutivos no final do contador $A$.

Note que $\phi(A_i) - \phi(A_{i-1}) \leq 1 - t_i$, pois $\phi(A_i) \leq \phi(A_{i-1}) + 1 - t_i$

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.

Note que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA.

Note que $c_i \leq 1 + t_i$ onde $t_i$ é o número de
bits 1 consecutivos no final do contador $A$.

Note que $\phi(A_i) - \phi(A_{i-1}) \leq 1 - t_i$, pois $\phi(A_i) \leq \phi(A_{i-1}) + 1 - t_i$

Seja $\hat{c}_i = c_i + \phi(A_i) - \phi(A_{i-1}) \leq (1 + t_i) + (1 - t_i) = 2$.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.
Temos que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA
e $t_i$ é o número de bits 1 consecutivos no final do contador $A$.
Temos que $c_i \leq 1 + t_i$.

Seja $\hat{c}_i = c_i + \phi(A_i) - \phi(A_{i-1}) \leq (1 + t_i) + (1 - t_i) = 2$.

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.
Temos que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA
e $t_i$ é o número de bits 1 consecutivos no final do contador $A$.
Temos que $c_i \leq 1 + t_i$.

Seja $\hat{c}_i = c_i + \phi(A_i) - \phi(A_{i-1}) \leq (1 + t_i) + (1 - t_i) = 2$.

Então o custo da volta completa é

$$c = \sum_{i=1}^{2^n} c_i = \sum_{i=1}^{2^n} \hat{c}_i + \underbrace{\phi(A_0) - \phi(A_{2^n})}_{\leq 0} \leq \sum_{i=1}^{2^n} \hat{c}_i \leq 2 \cdot 2^n.$$

# Método do potencial

Seja $\phi(A)$ o número de bits que valem 1 em $A[0\,..\,n-1]$.

Seja $A_i$ o estado do contador $A$ após o $i$-ésimo INCREMENTA.
Temos que $\phi(A_0) = 0$ e $\phi(A_i) \geq 0$.

Seja $c_i$ o número de bits alterados no $i$-ésimo INCREMENTA
e $t_i$ é o número de bits 1 consecutivos no final do contador $A$.
Temos que $c_i \leq 1 + t_i$.

Seja $\hat{c}_i = c_i + \phi(A_i) - \phi(A_{i-1}) \leq (1 + t_i) + (1 - t_i) = 2$.

Então o custo da volta completa é

$$c \;=\; \sum_{i=1}^{2^n} c_i \;=\; \sum_{i=1}^{2^n} \hat{c}_i + \underbrace{\phi(A_0) - \phi(A_{2^n})}_{\leq 0} \;\leq\; \sum_{i=1}^{2^n} \hat{c}_i \;\leq\; 2 \cdot 2^n.$$

Custo amortizado por INCREMENTA: 2 ▷ (valor do $\hat{c}_i$)

# Outro exemplo: pilha

Pilha $S$, inicialmente vazia. CLRS 17.1

| | | |
|---|---|---|
| PUSH$(S, x)$ | empilha $x$ | custo 1 |
| POP$(S)$ | desempilha e devolve o topo | custo 1 |

# Outro exemplo: pilha

Pilha $S$, inicialmente vazia. CLRS 17.1

| | | |
|---|---|---|
| PUSH$(S, x)$ | empilha $x$ | custo 1 |
| POP$(S)$ | desempilha e devolve o topo | custo 1 |

```
MULTIPOP (S, k)
1  enquanto S não está vazia e k > 0 faça
2      POP(S)
3      k ← k − 1
```

Se $S$ tem $s$ elementos, MULTIPOP$(S, k)$ custa $\min(s, k)$.

# Outro exemplo: pilha

Pilha $S$, inicialmente vazia. CLRS 17.1

| | | |
|---|---|---|
| PUSH$(S, x)$ | empilha $x$ | custo 1 |
| POP$(S)$ | desempilha e devolve o topo | custo 1 |

```
MULTIPOP (S, k)
1  enquanto S não está vazia e k > 0 faça
2      POP(S)
3      k ← k − 1
```

Se $S$ tem $s$ elementos, MULTIPOP$(S, k)$ custa $\min(s, k)$.

Sequência de $n$ operações PUSH, POP e MULTIPOP.

Pior caso de uma operação: $O(n)$ $\Rightarrow$ total $O(n^2)$.

# Outro exemplo: pilha

Pilha $S$, inicialmente vazia. CLRS 17.1

| | | |
|---|---|---|
| PUSH$(S, x)$ | empilha $x$ | custo 1 |
| POP$(S)$ | desempilha e devolve o topo | custo 1 |

```
MULTIPOP (S, k)
1  enquanto S não está vazia e k > 0 faça
2      POP(S)
3      k ← k − 1
```

Se $S$ tem $s$ elementos, MULTIPOP$(S, k)$ custa $\min(s, k)$.

Sequência de $n$ operações PUSH, POP e MULTIPOP.

Pior caso de uma operação: $O(n)$ $\Rightarrow$ total $O(n^2)$.

Correto, mas frouxo! O total é $O(n)$.

# Pilha: método agregado

Cada elemento é desempilhado no máximo uma vez para cada vez que é empilhado.

# Pilha: método agregado

Cada elemento é desempilhado no máximo uma vez
para cada vez que é empilhado.

Logo o número de chamadas de POP
(inclusive as feitas dentro de MULTIPOP)
é no máximo o número de chamadas de PUSH, que é $\leq n$.

# Pilha: método agregado

Cada elemento é desempilhado no máximo uma vez
para cada vez que é empilhado.

Logo o número de chamadas de POP
(inclusive as feitas dentro de MULTIPOP)
é no máximo o número de chamadas de PUSH, que é $\leq n$.

Custo total das $n$ operações: $O(n)$.

Custo *amortizado* por operação: $O(n)/n = O(1)$.

# Pilha: análise por créditos

Atribuímos créditos a cada operação:

| operação | custo real | créditos |
|---|---|---|
| PUSH | 1 | 2 |
| POP | 1 | 0 |
| MULTIPOP | $\min(s, k)$ | 0 |

# Pilha: análise por créditos

Atribuímos créditos a cada operação:

| operação | custo real | créditos |
|---|---|---|
| PUSH | 1 | 2 |
| POP | 1 | 0 |
| MULTIPOP | $\min(s, k)$ | 0 |

PUSH: um crédito paga o próprio PUSH;
o outro fica armazenado sobre o elemento empilhado.

# Pilha: análise por créditos

Atribuímos créditos a cada operação:

| operação | custo real | créditos |
|---|---|---|
| PUSH | 1 | 2 |
| POP | 1 | 0 |
| MULTIPOP | $\min(s, k)$ | 0 |

PUSH: um crédito paga o próprio PUSH;
o outro fica armazenado sobre o elemento empilhado.

Cada POP (avulso ou dentro de MULTIPOP) é pago
pelo crédito armazenado sobre o elemento desempilhado.

# Pilha: análise por créditos

Atribuímos créditos a cada operação:

| operação | custo real | créditos |
|---|---|---|
| PUSH | 1 | 2 |
| POP | 1 | 0 |
| MULTIPOP | $\min(s, k)$ | 0 |

PUSH: um crédito paga o próprio PUSH;
o outro fica armazenado sobre o elemento empilhado.

Cada POP (avulso ou dentro de MULTIPOP) é pago
pelo crédito armazenado sobre o elemento desempilhado.

O número de créditos armazenados é o número de elementos na pilha, logo é sempre não negativo.

# Pilha: análise por créditos

Atribuímos créditos a cada operação:

| operação | custo real | créditos |
|---|---|---|
| PUSH | 1 | 2 |
| POP | 1 | 0 |
| MULTIPOP | $\min(s, k)$ | 0 |

PUSH: um crédito paga o próprio PUSH;
o outro fica armazenado sobre o elemento empilhado.

Cada POP (avulso ou dentro de MULTIPOP) é pago
pelo crédito armazenado sobre o elemento desempilhado.

O número de créditos armazenados é o número de elementos na pilha, logo é sempre não negativo.

Custo total: $\leq 2n$. Custo amortizado por operação: $O(1)$.

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

PUSH: $\hat{c}_i = 1 + 1 = 2$

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

PUSH: $\hat{c}_i = 1 + 1 = 2$
POP: $\hat{c}_i = 1 - 1 = 0$

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

PUSH: $\hat{c}_i = 1 + 1 = 2$
POP: $\hat{c}_i = 1 - 1 = 0$
MULTIPOP$(S, k)$, com $k' = \min(s, k)$: $\hat{c}_i = k' - k' = 0$

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

PUSH: $\hat{c}_i = 1 + 1 = 2$
POP: $\hat{c}_i = 1 - 1 = 0$
MULTIPOP$(S, k)$, com $k' = \min(s, k)$: $\hat{c}_i = k' - k' = 0$

Então o custo das $n$ operações é

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i + \underbrace{\phi(S_0) - \phi(S_n)}_{\leq 0} \leq \sum_{i=1}^{n} \hat{c}_i \leq 2n.$$

# Pilha: método do potencial

Seja $\phi(S)$ o número de elementos na pilha $S$.

Seja $S_i$ a pilha após a $i$-ésima operação.
Temos que $\phi(S_0) = 0$ e $\phi(S_i) \geq 0$.

Seja $\hat{c}_i = c_i + \phi(S_i) - \phi(S_{i-1})$ e $s = \phi(S_{i-1})$.

PUSH: $\hat{c}_i = 1 + 1 = 2$
POP: $\hat{c}_i = 1 - 1 = 0$
MULTIPOP$(S, k)$, com $k' = \min(s, k)$: $\hat{c}_i = k' - k' = 0$

Então o custo das $n$ operações é

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i + \underbrace{\phi(S_0) - \phi(S_n)}_{\leq 0} \leq \sum_{i=1}^{n} \hat{c}_i \leq 2n.$$

Custo amortizado por operação: $O(1)$.

# Tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

# Tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

Inicialmente o vetor tem 0 posições.

# Tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

Inicialmente o vetor tem 0 posições.

Na primeira inserção, um vetor com uma posição é alocado, e o item em questão é inserido.

# Tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

Inicialmente o vetor tem 0 posições.

Na primeira inserção, um vetor com uma posição é alocado,
e o item em questão é inserido.

A cada inserção em que o vetor está cheio,
antes da inserção propriamente dita,
um vetor do dobro do tamanho é alocado,
o vetor anterior é copiado para o novo vetor
e depois é desalocado.

# Tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

Inicialmente o vetor tem 0 posições.

Na primeira inserção, um vetor com uma posição é alocado,
e o item em questão é inserido.

A cada inserção em que o vetor está cheio,
antes da inserção propriamente dita,
um vetor do dobro do tamanho é alocado,
o vetor anterior é copiado para o novo vetor
e depois é desalocado.

O custo no pior caso de uma inserção é alto,
pois pode haver uma realocação (de custo zero)
e subsequente reinserções.

# Custo por inserção

Seja $c_i$ o custo da $i$-ésima inserção, para $i = 1, 2, \ldots, n$:

$$c_i = \begin{cases} i & \text{se } i - 1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$$

# Custo por inserção

Seja $c_i$ o custo da $i$-ésima inserção, para $i = 1, 2, \ldots, n$:

$$c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$$

Método agregado:

$$\sum_{i=1}^{n} c_i = n + (1 + 2 + 2^2 + \cdots + 2^k)$$

onde $k = \lfloor \lg(n-1) \rfloor$ (para $n \geq 2$).

# Custo por inserção

Seja $c_i$ o custo da $i$-ésima inserção, para $i = 1, 2, \ldots, n$:

$$c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$$

Método agregado:

$$\sum_{i=1}^{n} c_i = n + (1 + 2 + 2^2 + \cdots + 2^k)$$

onde $k = \lfloor \lg(n-1) \rfloor$ (para $n \geq 2$).

Logo $\sum_{i=1}^{n} c_i = n + 2^{k+1} - 1 < n + 2n - 1 < 3n$.

# Custo por inserção

Seja $c_i$ o custo da $i$-ésima inserção, para $i = 1, 2, \ldots, n$:

$$c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$$

Método agregado:

$$\sum_{i=1}^{n} c_i = n + (1 + 2 + 2^2 + \cdots + 2^k)$$

onde $k = \lfloor \lg(n-1) \rfloor$ (para $n \geq 2$).

Logo $\sum_{i=1}^{n} c_i = n + 2^{k+1} - 1 < n + 2n - 1 < 3n$.

Custo amortizado por inserção: 3

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

Atribuímos 3 créditos por inserção:
- um paga pela inserção do item,
- os outros dois são armazenados sobre o item.

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

Atribuímos 3 créditos por inserção:
- um paga pela inserção do item,
- os outros dois são armazenados sobre o item.

Ao ocorrer uma realocação,
há dois créditos sobre cada item novo no vetor.

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

Atribuímos 3 créditos por inserção:
um paga pela inserção do item,
os outros dois são armazenados sobre o item.

Ao ocorrer uma realocação,
há dois créditos sobre cada item novo no vetor.

Isso paga cópia de todos os itens para o novo vetor pois,
quando vetor está cheio, há um item novo para cada velho.

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

Atribuímos 3 créditos por inserção:
um paga pela inserção do item,
os outros dois são armazenados sobre o item.

Ao ocorrer uma realocação,
há dois créditos sobre cada item novo no vetor.

Isso paga cópia de todos os itens para o novo vetor pois,
quando vetor está cheio, há um item novo para cada velho.

Em outras palavras, o segundo crédito paga cópia do item
na primeira realocação que acontecer após sua inserção,

# Análise por créditos

Item velho: já estava no vetor durante a última realocação
Item novo: item inserido após a última realocação.

Atribuímos 3 créditos por inserção:
um paga pela inserção do item,
os outros dois são armazenados sobre o item.

Ao ocorrer uma realocação,
há dois créditos sobre cada item novo no vetor.

Isso paga cópia de todos os itens para o novo vetor pois,
quando vetor está cheio, há um item novo para cada velho.

Em outras palavras, o segundo crédito paga cópia do item
na primeira realocação que acontecer após sua inserção,
e o terceiro paga cópia de um item velho nesta mesma realocação.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Note que $n_i = n_{i-1} + 1$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Note que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i - 1$ não é potência de 2, então $c_i = 1$ e $s_i = s_{i-1}$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)

$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.

Seja $\phi(T_i) = 2n_i - s_i$.

Como $T_0$ é vazia, $n_0 = s_0 = 0$, e portanto $\phi(T_0) = 0$.

Como não há remoção, $n_i \geq s_i/2$. Logo $\phi(T_i) \geq 0$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Note que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i - 1$ não é potência de 2, então $c_i = 1$ e $s_i = s_{i-1}$.
Assim $\hat{c}_i = 1 + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) = 1 + 2 = 3$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i - 1$ é potência de 2, então ...

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i-1$ é potência de 2, então ...
$c_i = i, \quad s_i = 2s_{i-1} \quad$ e $\quad s_{i-1} = n_{i-1} = i-1$.

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i-1$ é potência de 2, então ...
$c_i = i$, $\quad s_i = 2s_{i-1}$ $\quad$ e $\quad s_{i-1} = n_{i-1} = i-1$.

Assim

$$\hat{c}_i \quad = \quad i + (2n_i - s_i) - (2n_{i-1} - s_{i-1})$$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i-1$ é potência de 2, então ...
$c_i = i, \quad s_i = 2s_{i-1} \quad$ e $\quad s_{i-1} = n_{i-1} = i-1$.

Assim

$$\begin{aligned} \hat{c}_i &= i + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \\ &= i + 2 + s_{i-1} - s_i \end{aligned}$$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i-1$ é potência de 2, então ...
$c_i = i$, $\quad s_i = 2s_{i-1}$ $\quad$ e $\quad s_{i-1} = n_{i-1} = i-1$.

Assim

$$\begin{aligned} \hat{c}_i &= i + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \\ &= i + 2 + s_{i-1} - s_i \\ &= i + 2 + s_{i-1} - 2s_{i-1} \;=\; i + 2 - s_{i-1} \end{aligned}$$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i > 1$ e $i-1$ é potência de 2, então ...
$c_i = i$, $\quad s_i = 2s_{i-1}$ $\quad$ e $\quad s_{i-1} = n_{i-1} = i-1$.

Assim

$$\begin{aligned} \hat{c}_i &= i + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \\ &= i + 2 + s_{i-1} - s_i \\ &= i + 2 + s_{i-1} - 2s_{i-1} = i + 2 - s_{i-1} \\ &= i + 2 - (i-1) = 3. \end{aligned}$$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i = 1$, então ...
$c_i = 1$, $s_i = 1$, $s_{i-1} = 0$,
$n_i = 1$ e $n_{i-1} = 0$

# Método do potencial

$T_i$: tabela após a $i$-ésima inserção ($T_0$: tabela inicial, vazia)
$n_i$: número de itens na tabela $T_i$ $\quad$ $s_i$: tamanho da tabela $T_i$.
$\phi(T_i) = 2n_i - s_i$.

Lembre-se que $c_i = \begin{cases} i & \text{se } i-1 \text{ é potência de 2} \\ 1 & \text{caso contrário} \end{cases}$

Custo amortizado: $\hat{c}_i = c_i + \phi(T_i) - \phi(T_{i-1})$

Lembre-se que $n_i = n_{i-1} + 1$.

Se $i = 1$, então ...
$c_i = 1$, $\quad s_i = 1$, $\quad s_{i-1} = 0$,
$n_i = 1$ $\quad$ e $\quad n_{i-1} = 0$

Assim

$$\begin{aligned} \hat{c}_i &= i + (2n_i - s_i) - (2n_{i-1} - s_{i-1}) \\ &= 1 + (2-1) - (0-0) = 2 \end{aligned}$$