Tópicos de Análise de Algoritmos

Análise amortizada

Cap 17 do CLRS

Análise amortizada

Serve para analisar uma sequência de operações ou iterações onde o pior caso individual não reflete o pior caso da sequência.

Em outras palavras, serve para melhorar análises de pior caso que baseiem-se diretamente no pior caso de uma operação/iteração e que deem uma delimitação frouxa para o tempo de pior caso da sequência.

Métodos:

- agregado
- por créditos
- potencial

Aula passada: odômetro binário

Segundo exemplo: pilha

Operações básicas: empilha, desempilha.

Segundo exemplo: pilha

Operações básicas: empilha, desempilha.

OP: operação única de acesso

OP (n)

1 ▷ exige que a pilha tenha $\geq n$ elementos
2 desempilhe n itens da pilha
3 empilhe um item na pilha

Segundo exemplo: pilha

Operações básicas: empilha, desempilha.

OP: operação única de acesso

```
OP (n)
1   ▷ exige que a pilha tenha ≥ n elementos
2   desempilhe n itens da pilha
3   empilhe um item na pilha
```

Consumo de tempo de OP(n) é $\Theta(n)$.

Segundo exemplo: pilha

Operações básicas: empilha, desempilha.

OP: operação única de acesso

```
OP (n)
1   ▷ exige que a pilha tenha ≥ n elementos
2   desempilhe n itens da pilha
3   empilhe um item na pilha
```

Consumo de tempo de OP(n) é $\Theta(n)$.

Sequência de m operações OP.

Consumo de tempo de uma operação no pior caso: $\Theta(m)$.

Qual o consumo total das m operações no pior caso?

Segundo exemplo: pilha

Consumo de tempo de uma operação no pior caso: $\Theta(m)$.

Qual o consumo das m operações no pior caso? $\Theta(m^2)$?

Segundo exemplo: pilha

Consumo de tempo de uma operação no pior caso: $\Theta(m)$.

Qual o consumo das m operações no pior caso? $\Theta(m^2)$?

Em aula...

- análise por créditos
  Quantos créditos damos para cada chamada de OP?

Segundo exemplo: pilha

Consumo de tempo de uma operação no pior caso: $\Theta(m)$.

Qual o consumo das m operações no pior caso? $\Theta(m^2)$?

Em aula...

- análise por créditos
  Quantos créditos damos para cada chamada de OP?
- análise por função potencial
  Qual seria uma boa função potencial neste caso?

Terceiro exemplo: tabelas dinâmicas

Vetor que sofre inserções. Cada inserção custa 1.

Inicialmente o vetor tem 0 posições.

Terceiro exemplo: 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.

Terceiro exemplo: 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.

Terceiro exemplo: 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.

Tabelas dinâmicas

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$$

Tabelas dinâmicas

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$$

Método agregado:

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

onde $k = \lfloor \lg n \rfloor$.

Tabelas dinâmicas

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$$

Método agregado:

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

onde $k = \lfloor \lg n \rfloor$.

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

Tabelas dinâmicas

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$$

Método agregado:

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

onde $k = \lfloor \lg n \rfloor$.

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

Custo amortizado por inserção: 3

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

Atribuímos 3 créditos por inserção:

um é usado para pagar pela inserção do item,
os outros dois são armazenados sobre o item.

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

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

Ao ocorrer uma realocação,
há 2 créditos sobre cada item novo no vetor,

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

Atribuímos 3 créditos por inserção:

- um é usado para pagar pela inserção do item,
- os outros dois são armazenados sobre o item.

Ao ocorrer uma realocação,
há 2 créditos sobre cada item novo no vetor, e isso é suficiente para pagar pela cópia de todos os itens do vetor para o novo vetor

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

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

Ao ocorrer uma realocação,
há 2 créditos sobre cada item novo no vetor, e isso é suficiente para pagar pela cópia de todos os itens do vetor para o novo vetor pois, quando o vetor está cheio, há um item novo para cada item velho.

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

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

Ao ocorrer uma realocação,
há 2 créditos sobre cada item novo no vetor, e isso é suficiente para pagar pela cópia de todos os itens do vetor para o novo vetor pois, quando o vetor está cheio, há um item novo para cada item velho.

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

Análise por créditos

Chame de velho um item que já estava no vetor
no momento da última realocação do vetor,
e de novos os itens inseridos após a última realocação.

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

Ao ocorrer uma realocação,
há 2 créditos sobre cada item novo no vetor, e isso é suficiente para pagar pela cópia de todos os itens do vetor para o novo vetor pois, quando o vetor está cheio, há um item novo para cada item velho.

Que função potencial você usaria neste caso?

Análise por função potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais 1} \\ i & \text{se } i \text{ é potência de 2 mais 1.} \end{cases}$$

Chame de velho um item que já estava no vetor T
no momento da última realocação do vetor T,
e de novos os itens inseridos após a última realocação.

Método potencial:

Que função potencial você usaria neste caso?

Tome $\Phi(T)$ como duas vezes o número de elementos novos em T.

Análise por função potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais 1} \\ i & \text{se } i \text{ é potência de 2 mais 1.} \end{cases}$$

Chame de velho um item que já estava no vetor T
no momento da última realocação do vetor T,
e de novos os itens inseridos após a última realocação.

Método potencial:

Que função potencial você usaria neste caso?

Tome $\Phi(T)$ como duas vezes o número de elementos novos em T.

T_i: estado do vetor T imediatamente após a inserção i

Note que $\Phi(T_0) = 0$ e $\Phi(T_i) \geq 0$.

Custo por inserção e função potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais 1} \\ i & \text{se } i \text{ é potência de 2 mais 1.} \end{cases}$$

Chame de velho um item que já estava no vetor T
no momento da última realocação do vetor T,
e de novos os itens inseridos após a última realocação.

Método potencial:

Tome $\Phi(T) = 2(\mathrm{num}(T) - \mathrm{size}(T)/2)$
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Custo por inserção e função potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais 1} \\ i & \text{se } i \text{ é potência de 2 mais 1.} \end{cases}$$

Chame de velho um item que já estava no vetor T
no momento da última realocação do vetor T,
e de novos os itens inseridos após a última realocação.

Método potencial:

Tome $\Phi(T) = 2(\mathrm{num}(T) - \mathrm{size}(T)/2) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Custo por inserção e função potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais 1} \\ i & \text{se } i \text{ é potência de 2 mais 1.} \end{cases}$$

Chame de velho um item que já estava no vetor T
no momento da última realocação do vetor T,
e de novos os itens inseridos após a última realocação.

Método potencial:

Tome $\Phi(T) = 2(\mathrm{num}(T) - \mathrm{size}(T)/2) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Vamos calcular $\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$.

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Vamos calcular $\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$.

Dois casos:

- i não é potência de 2 mais um.

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Vamos calcular $\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$.

Dois casos:

- i não é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1$ e $\mathrm{size}(T_i) = \mathrm{size}(T_{i-1})$.

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Vamos calcular $\hat{c}_i = c_i + \Phi(T_i) - \Phi(T_{i-1})$.

Dois casos:

- i não é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1$ e $\mathrm{size}(T_i) = \mathrm{size}(T_{i-1})$.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) \\ &= 1 + (2\,\mathrm{num}(T_i) - \mathrm{size}(T_i)) - (2\,\mathrm{num}(T_{i-1}) - \mathrm{size}(T_{i-1})) \\ &= 1 + 2 + 0 \;=\; 3. \end{aligned}$$

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1 = c_i$ e
  $\mathrm{size}(T_i) = 2\,\mathrm{size}(T_{i-1}) = 2\,\mathrm{num}(T_{i-1})$

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1 = c_i$ e
  $\mathrm{size}(T_i) = 2\,\mathrm{size}(T_{i-1}) = 2\,\mathrm{num}(T_{i-1}) = 2(c_i - 1)$.

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1 = c_i$ e
  $\mathrm{size}(T_i) = 2\,\mathrm{size}(T_{i-1}) = 2\,\mathrm{num}(T_{i-1}) = 2(c_i - 1)$.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) \\ &= c_i + (2\,\mathrm{num}(T_i) - \mathrm{size}(T_i)) - (2\,\mathrm{num}(T_{i-1}) - \mathrm{size}(T_{i-1})) \\ &= c_i + (2c_i - 2(c_i - 1)) \end{aligned}$$

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1 = c_i$ e
  $\mathrm{size}(T_i) = 2\,\mathrm{size}(T_{i-1}) = 2\,\mathrm{num}(T_{i-1}) = 2(c_i - 1)$.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) \\ &= c_i + (2\,\mathrm{num}(T_i) - \mathrm{size}(T_i)) - (2\,\mathrm{num}(T_{i-1}) - \mathrm{size}(T_{i-1})) \\ &= c_i + (2c_i - 2(c_i - 1)) - (2(c_i - 1) - (c_i - 1)) \end{aligned}$$

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$, $c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$

Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um.
  $\mathrm{num}(T_i) = \mathrm{num}(T_{i-1}) + 1 = c_i$ e
  $\mathrm{size}(T_i) = 2\,\mathrm{size}(T_{i-1}) = 2\,\mathrm{num}(T_{i-1}) = 2(c_i - 1)$.

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(T_i) - \Phi(T_{i-1}) \\ &= c_i + (2\,\mathrm{num}(T_i) - \mathrm{size}(T_i)) - (2\,\mathrm{num}(T_{i-1}) - \mathrm{size}(T_{i-1})) \\ &= c_i + (2c_i - 2(c_i - 1)) - (2(c_i - 1) - (c_i - 1)) \\ &= c_i + 2 - (c_i - 1) \;=\; 3. \end{aligned}$$

Tabelas dinâmicas: método potencial

Para $i = 1, 2, \ldots, n$,

$$c_i = \begin{cases} 1 & \text{se } i \text{ não é potência de 2 mais um} \\ i & \text{se } i \text{ é potência de 2 mais um.} \end{cases}$$

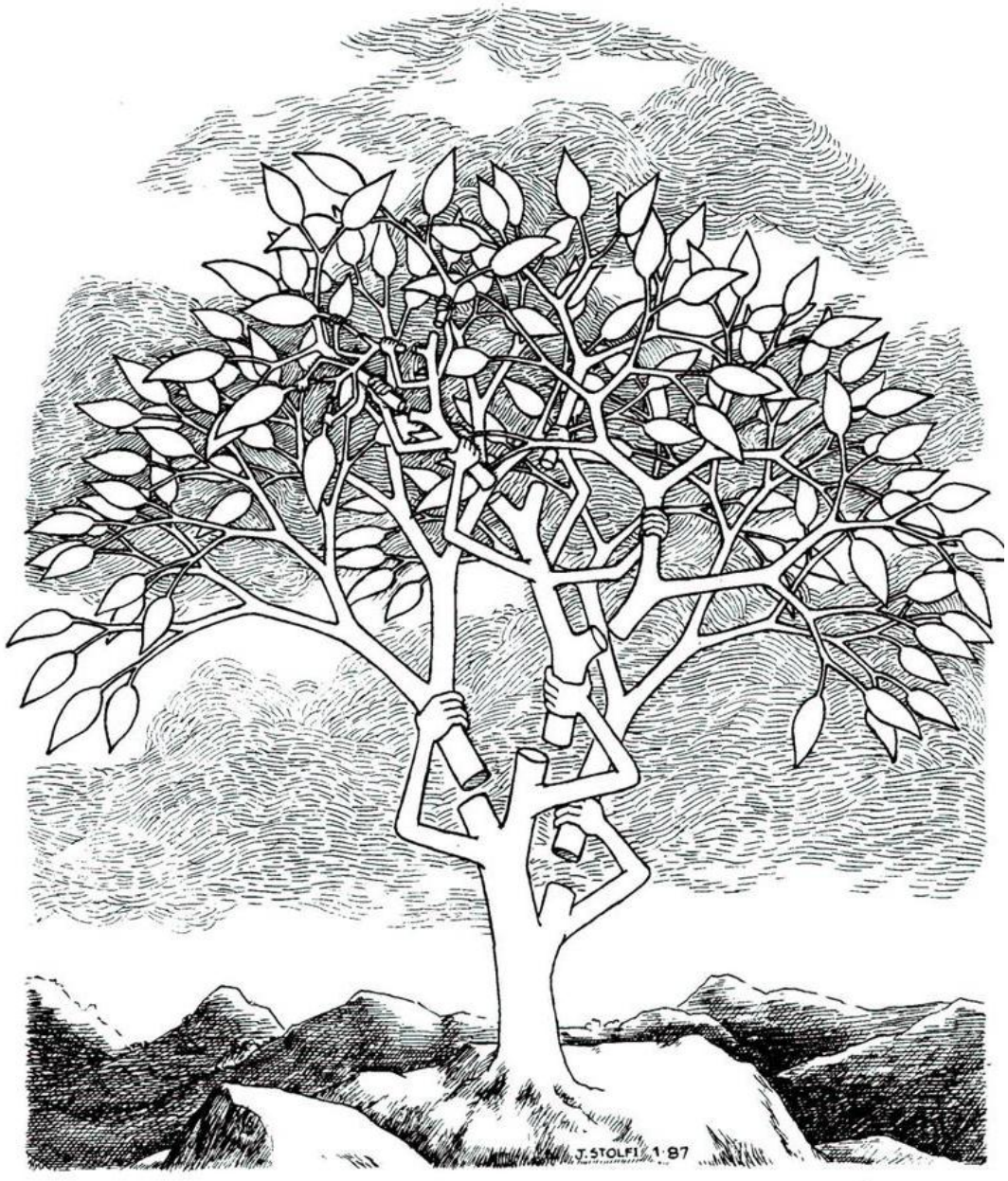
Tome $\Phi(T) = 2\,\mathrm{num}(T) - \mathrm{size}(T)$,
onde $\mathrm{num}(T)$ é o número de elementos em T e
$\mathrm{size}(T)$ é o tamanho de T.

Dois casos:

- i não é potência de 2 mais um: $\hat{c}_i = 3$.
- i é potência de 2 mais um: $\hat{c}_i = 3$.

Conclusão: custo amortizado por inserção é 3.

Quarto exemplo



Jorge Stolfi

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

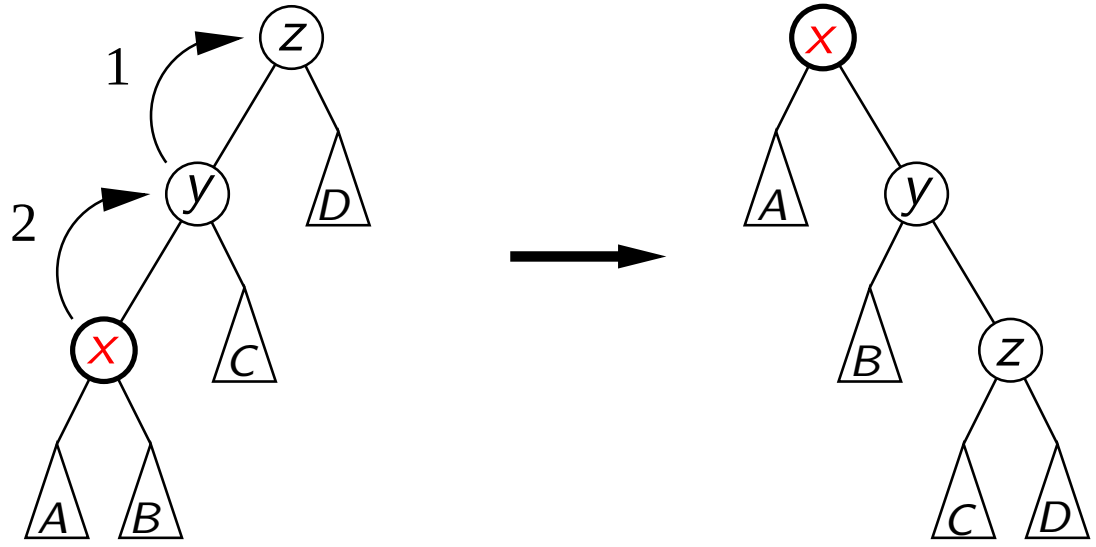
As rotações são feitas numa ordem específica, de duas em duas.

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

As rotações são feitas numa ordem específica, de duas em duas.



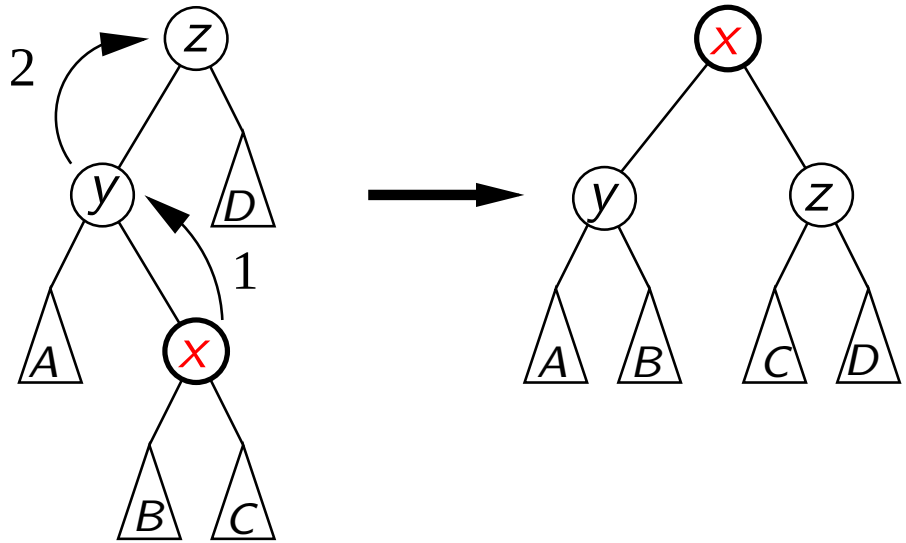
Acima, o rr splay step.

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

As rotações são feitas numa ordem específica, de duas em duas.



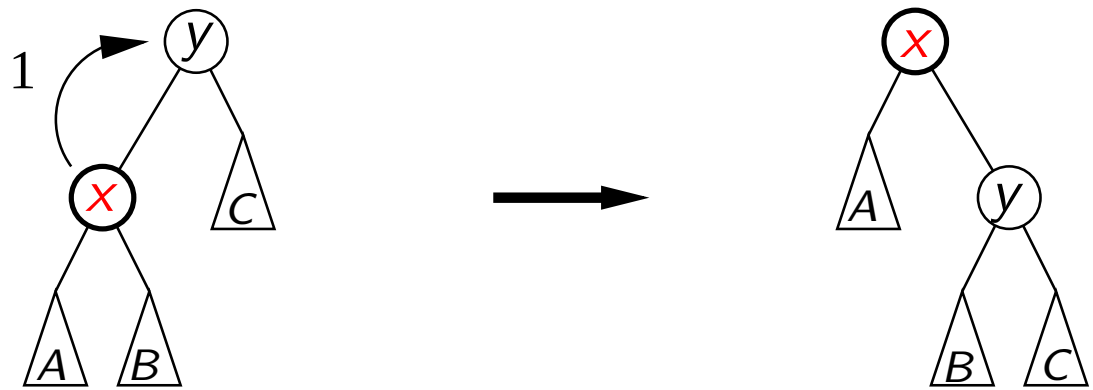
Acima, o lr splay step.

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

As rotações são feitas numa ordem específica, de duas em duas.
Eventualmente no final uma rotação simples é necessária.



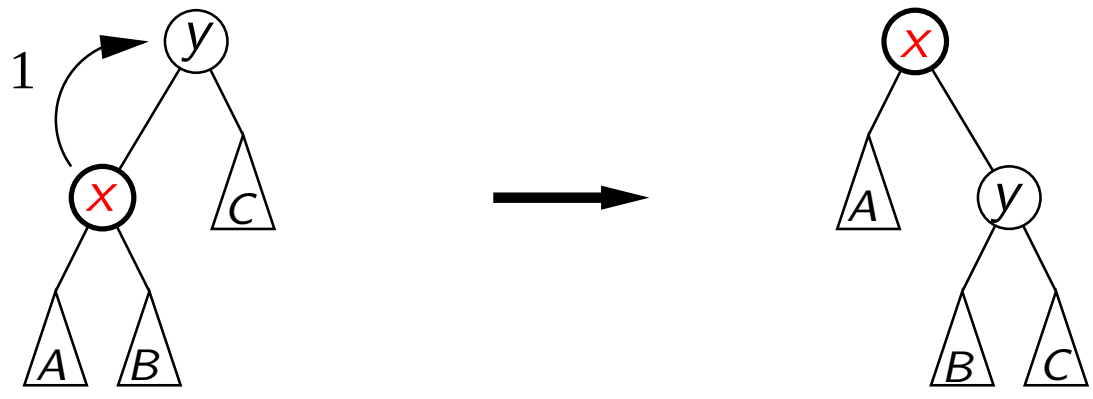
Acima, o r splay step.

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

As rotações são feitas numa ordem específica, de duas em duas.
Eventualmente no final uma rotação simples é necessária.



Acima, o r splay step.

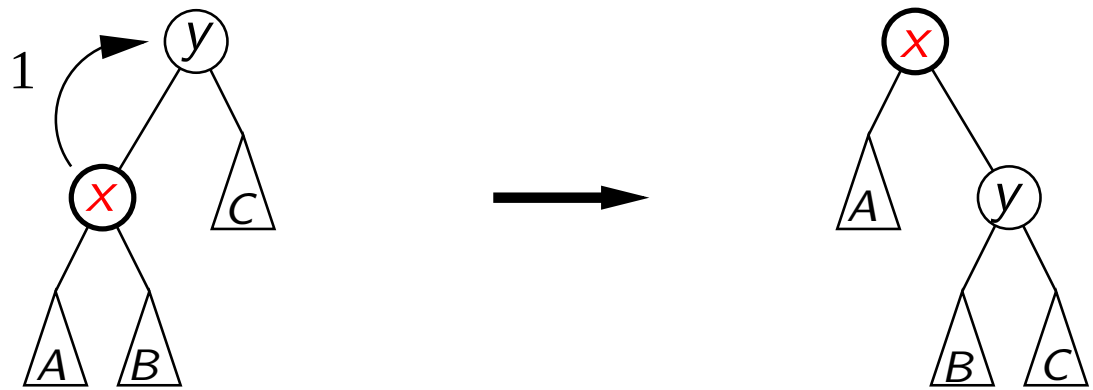
Além destes, o l splay, o rl splay e o ll splay step.

Quarto exemplo: splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

As rotações são feitas numa ordem específica, de duas em duas.
Eventualmente no final uma rotação simples é necessária.



Acima, o r splay step.

Além destes, o l splay, o rl splay e o ll splay step.

Splay steps são realizados até que x seja raiz.

Splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

Splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

Custo de pior caso do SPLAY: $\Theta(n)$,
onde n é o número de elementos na splay tree.

Splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

Custo de pior caso do SPLAY: $\Theta(n)$,
onde n é o número de elementos na splay tree.

Queremos mostrar que uma splay tree tem
comportamento mais parecido com o de uma ABBB.

Splay trees

ABB: árvore binária de busca

Operação SPLAY(x, S), onde S é uma splay tree:
quando x é acessado, move-se x para a raiz por rotações.

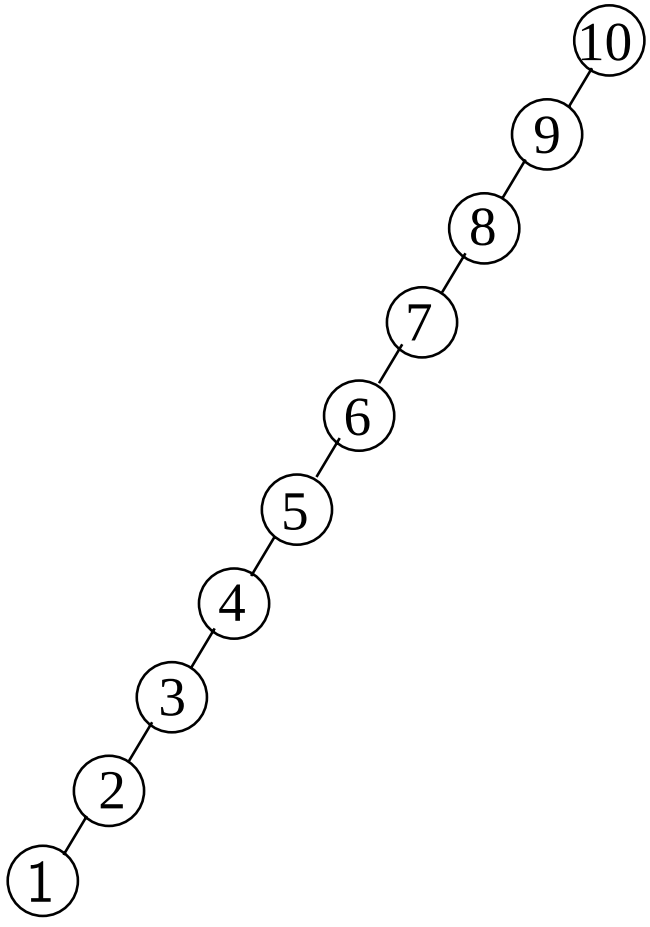
Custo de pior caso do SPLAY: $\Theta(n)$,
onde n é o número de elementos na splay tree.

Queremos mostrar que uma splay tree tem
comportamento mais parecido com o de uma ABBB.

(ABBB: ABB balanceada)

Splay trees: pior caso

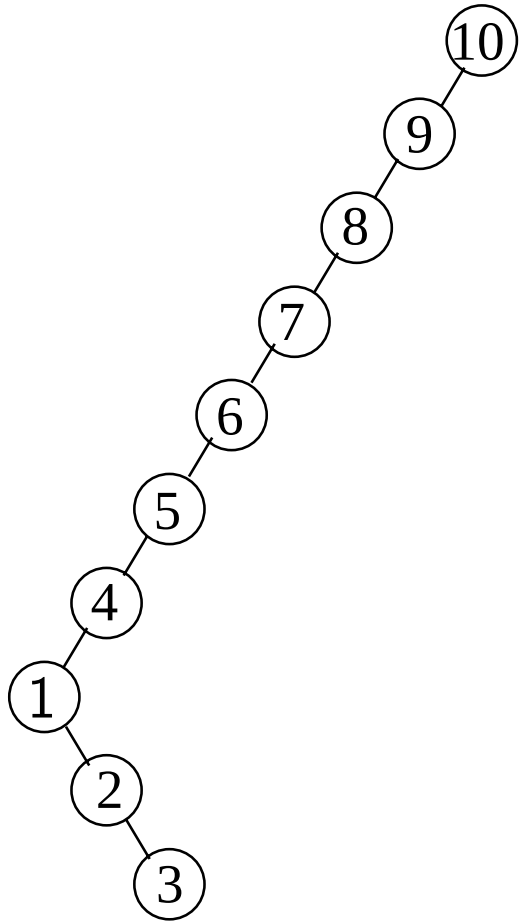
SPLAY$(1, S)$



Splay trees: pior caso

SPLAY$(1, S)$

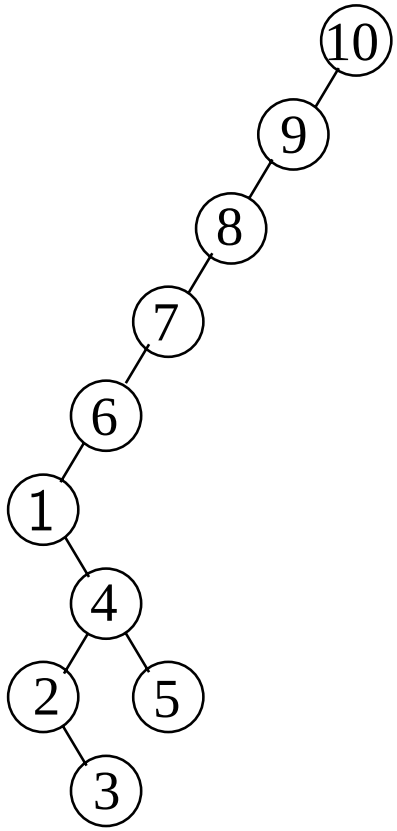
primeiro splay step



Splay trees: pior caso

SPLAY$(1, S)$

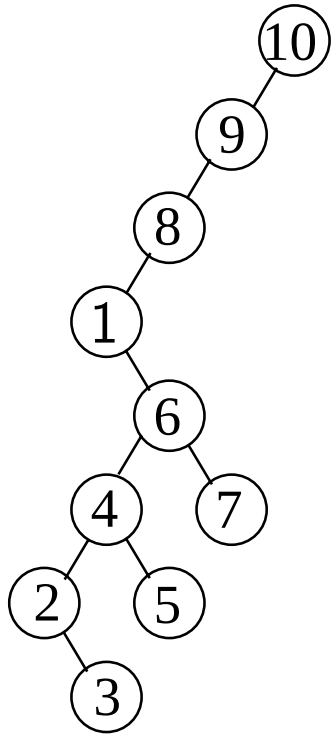
segundo splay step



Splay trees: pior caso

SPLAY$(1, S)$

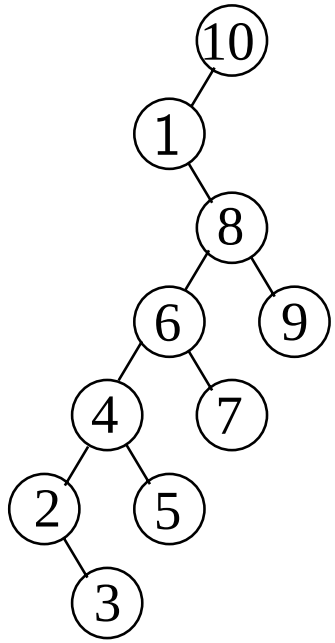
terceiro splay step



Splay trees: pior caso

SPLAY$(1, S)$

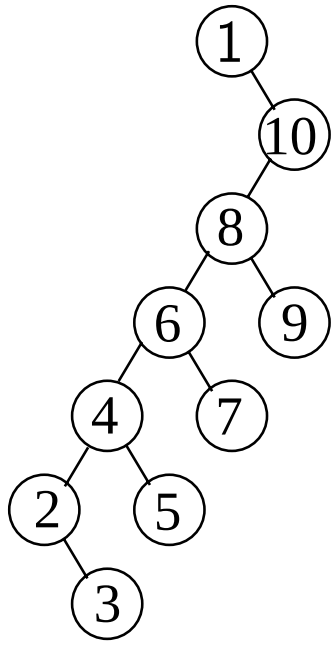
quarto splay step



Splay trees: pior caso

SPLAY$(1, S)$

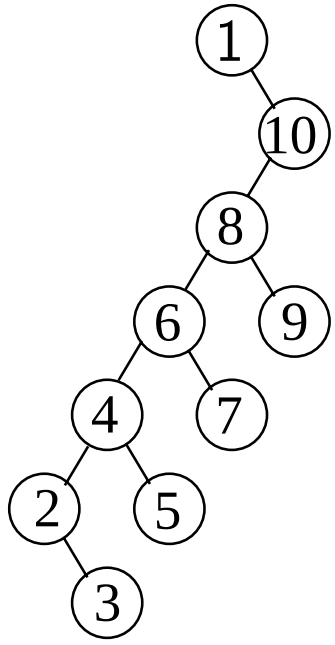
quinto splay step



Splay trees: pior caso

SPLAY$(1, S)$

quinto splay step



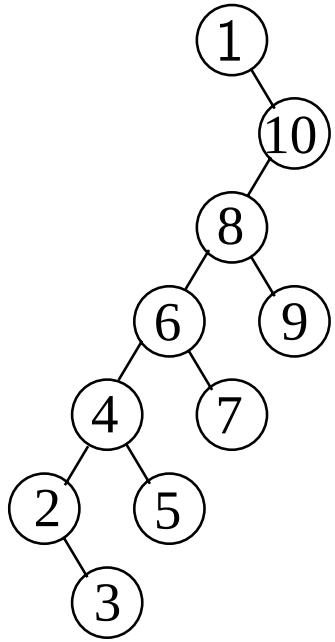
Total de rotações: 9

Em geral, esse caso é $\Theta(n)$, onde n é o número de nós.

Depois disso, o maior custo de um SPLAY nesta árvore é 6.

Splay trees: mais um exemplo

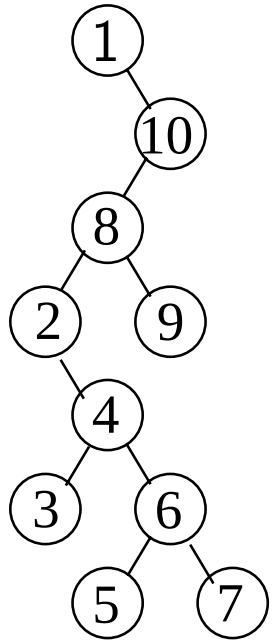
SPLAY$(2, S)$



Splay trees: mais um exemplo

SPLAY$(2, S)$

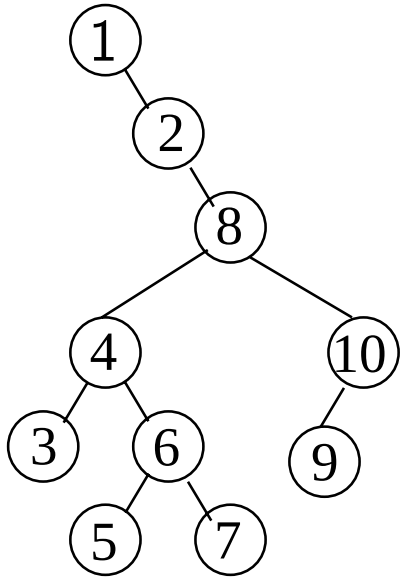
primeiro splay step



Splay trees: mais um exemplo

SPLAY$(2, S)$

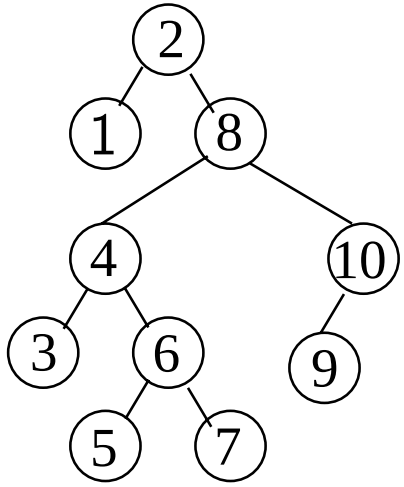
segundo splay step



Splay trees: mais um exemplo

SPLAY$(2, S)$

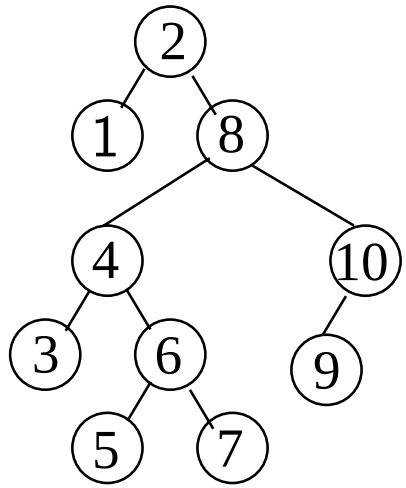
terceiro splay step



Splay trees: mais um exemplo

SPLAY$(2, S)$

terceiro splay step



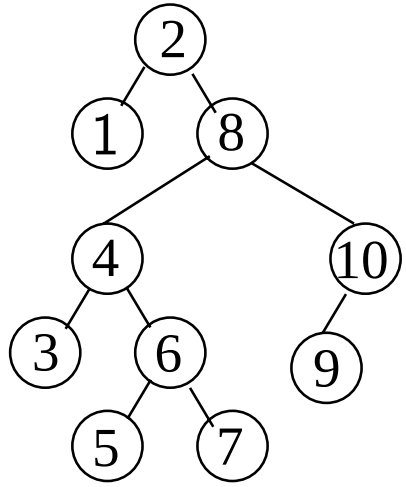
Total de rotações: 5

Árvore bem mais balanceada após estes dois SPLAYs.

Splay trees: mais um exemplo

SPLAY$(2, S)$

terceiro splay step



Total de rotações: 5

Árvore bem mais balanceada após estes dois SPLAYs.

Custo: número de rotações.

Aula que vem

Análise da política MTF (move to front) em listas.

Análise do consumo de tempo das splay trees.