Aula 22: Árvores chanfradas

Splay trees

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

Download me from `http://DavidDeharbe.github.io`.

PPgSC

Introdução

Análise amortizada
- Método agregado
- Método do contador
- Método do potencial

Arranjos dinâmicos

Árvores chanfradas

1. Avaliação de estruturas de dados: análise amortizada
   - Contador binário
   - Arranjos dinâmicos

Análise amortizada

- Medição da complexidade ao longo de n operações
- Custo de execução, no pior caso: $T(n)$
- Custo amortizado: $T(n)/n$
- Exemplos
  1. contador binário
  2. arranjo dinâmico

PPgSC

Método agregado

- Considere uma série de n operações em alguma estrutura de dados, n qualquer.
- Determina o custo $T(n)$ desta série de operações.
- O custo médio de cada operação é $T(n)/n$.
- Exemplo: contador binário

Método agregado: um exemplo

INCREMENT(A)

```
    // A é um arranjo de k bits
1   i = 0
2   while i < length[A] and A[i] = 1
3         A[i] = 0
4         i = i + 1
5   if i < length[A]
6         A[i] = 1
```

- O tamanho da entrada é k;
- O custo da execução da INCREMENT é proporcional ao número de bits invertidos.
- No pior caso, a complexidade é proporcional a k;
- Logo, o custo de uma sequência de n operações é $O(n \times k)$.
- Podemos prover uma análise mais precisa?

Método agregado: um exemplo

Em uma série de n execuções de INCREMENT:

- $A[0]$ é invertido n vezes;
- $A[1]$ é invertido $n/2$ vezes;
- $A[2]$ é invertido $n/4$ vezes;
- $A[k-1]$ é invertido $n/2^{k-1}$ vezes;

O custo total para n operações é

$$\begin{aligned} T(n) &= \sum_{i=1}^{\lfloor \lg n \rfloor} n/2^{i-1} \\ T(n) &< \sum_{i=1}^{\infty} n/2^{i-1} \\ T(n) &< n \times \sum_{i=1}^{\infty} 1/2^{i-1} = 2n \end{aligned}$$

Logo o custo amortizado de INCREMENT é $T(n)/n = 2 \in \Theta(1)$: é constante.

Método do contador

Quanto pagar para executar n operações?

- cada operação tem um custo real
- a cada operação é atribuído um preço ($\approx$ custo amortizado individual)
- os preços cobrados devem cobrir os custos reais
- se o preço é maior que o custo:
  - crédito é associado a elementos da estrutura de dados
- se o custo é maior que o preço
  - a diferença deve poder ser paga com os créditos disponíveis
- custo amortizado total: soma dos preços cobrados

PPgSC

Critérios de definição do custo

- o custo amortizado total deve ser uma cota superior do custo real
  - a estrutura de dados não “empresta”
- o crédito total associado aos elementos nunca pode ser negativo

Método do contador: o contador binário

- a unidade de custo é inverter um bit
- preço para $0 \rightarrow 1$: 2
  - 1 é o custo
  - sobra $2 - 1 = 1$ de crédito, associado ao bit setado a 1
- preço para $1 \rightarrow 0$: 0
  - pago com o crédito

Método do contador: o contador binário

```
INCREMENT(A)
    // A é um arranjo de k bits
1   i = 0
2   while i < length[A] and A[i] = 1
3       A[i] = 0
4       i = i + 1
5   if i < length[A]
6       A[i] = 1
```

- O custo do laço é pago usando os créditos dos bits setados a 1.
- no máximo um único bit é setado a 1: custo 1, mais 1 que fica como crédito
- logo, o custo de uma execução da operação INCREMENT é 2
- o custo de n execuções da operação INCREMENT é $O(n)$.

Método do potencial

- crédito $\Longrightarrow$ potencial para executar futuras operações
- estrutura de dados inicia em um estado D_0
- n operações são executadas,
  - custo real $c_1, \ldots c_i, \ldots c_n$
  - levando aos estados $D_1, \ldots D_i, \ldots D_n$
- A função Φ associa um número real Φ_i (potencial) ao estado D_i
- Custo amortizado $a_i = c_i + \Phi_i - \Phi_{i-1}$
- Custo amortizado total

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} c_i + \Phi_i - \Phi_{i-1} = \sum_{i=1}^{n} c_i + \Phi_n - \Phi_0$$

- se $\forall i \cdot \Phi_i \geq \Phi_0$, então $\sum_{i=1}^{n} a_i$ é uma cota superior do custo total real.

Método do potencial

Relação com o método do contador

- $\Phi_0 = 0$
- mostrar que $\Phi_i \geq 0$, $\forall i$.
- $\Phi_i > \Phi_{i-1} \approx$ crédito (potencial aumenta)
- $\Phi_i < \Phi_{i-1} \approx$ débito (potencial diminui)

Método do potencial: o contador binário

- a i-ésima chamada de INCREMENT zera t_i bits
- custo real $c_i = t_i + 1$ (zera i, seta 1)
- potencial: $\Phi_i =$ quantidade de bits setados a 1
- naturalmente, $\forall i \cdot \Phi_i \geq 0$
- logo $\Phi_i \leq \Phi_{i-1} - t_i + 1$
- $\Phi_i - \Phi_{i-1} \leq 1 - t_i$
- custo amortizado de uma operação

$$a_i = c_i + \Phi_i - \Phi_{i-1} \leq (t_i + 1) - (1 - t_i) = 2$$

- custo amortizado de n operações $O(n)$

PPgSC

Arranjos dinâmicos

- contêiner tabela, cuja capacidade adapta-se ao tamanho da coleção
  - inserção e remoção: modelo pilha
- inserção em uma tabela está cheia
  - aloca uma nova tabela de capacidade maior
  - copia o conteúdo da tabela original para a nova tabela
  - libera o espaço ocupado pela tabela original
  - insere o novo elemento
- política de expansão: dobra o tamanho
- remoção, quando o fator de carga fica baixo
  - aloca uma nova tabela de capacidade menor
  - copia o conteúdo da tabela original para a nova tabela
  - libera o espaço ocupado pela tabela original
  - remove o elemento
- política de contração: divide a capacidade por dois quando fator de carga chega a 1/4.

PPgSC

Implementação: dados

- $T.table$ tabela com os elementos
- $T.num$ quantidade de elementos armazenados na tabela
- $T.size$ capacidade máxima

Inserção

```
INSERT(T, k)
 1  if T.size == 0 // alocação inicial
 2        T.table = ALLOC-TABLE(1)
 3        T.size = 1
 4  elseif T.num == T.size // expansão
 5        tab = T.table
 6        T.table = ALLOC-TABLE(T.size × 2)
 7        for i = 1 to T.size
 8              T.table[i] = tab[i]
 9        T.size = T.size × 2
10        FREE-TABLE(tab)
11  T.num = T.num + 1 // inserção
12  T.table[T.num] = k
```

Análise

Cenário: n inserções em uma tabela inicialmente vazia

- custo da operação i
  - sem expansão: 1
  - com expansão: i
- Pior caso $O(n^2)$
- Melhoria desta análise com análise amortizada

PPgSC

Método agregado

- Há expansão quando i é uma potência de 2.
- Custo total:

$$\begin{aligned}\sum_{i=1}^{n} c_i &\leq n + \sum_{j=0}^{\lfloor \log_2 n \rfloor} 2^j \\ &< n + 2n \\ &= 3n\end{aligned}$$

Método do contador

- Custo 1 para atribuir uma posição de $T.table$
- Preço de uma operação de inserção: 3
- Pagamento de uma operação de inserção, sem expansão:
  - 1 vai para a inserção (custo real) $i = T.size/2 + j$
  - sobra 2 de crédito
  - 1 crédito na posição de inserção i
  - 1 crédito para um outro elemento da tabela j
- Pagamento da expansão:
  - todos os itens tem um crédito
  - o crédito paga a expansão

PPgSC

Método do potencial

- Após cada expansão: $\Phi = 0$
- $\Phi \uparrow$ quando a tabela é preenchida
- até a tabela ser preenchida, e Φ é gasto na expansão
- $\Phi(T) = 2 \times T.num - T.size$
  - Inicialmente, e após cada expansão $\Phi(T) = 0$
  - Antes de uma expansão $\Phi(T) = 2 \times T.num - T.size = T.size$
  - A cada momento $\Phi(T) \geq 0$
  - Logo, a soma dos custos amortizados é uma cota superior dos custos reais.

PPgSC

Método do potencial

- n_i: número de elementos após operação i
- s_i: capacidade após a operação i
- inicialmente $n_0 = s_0 = 0$, $\Phi_0 = 0$
- $n_i = n_{i-1} + 1$

PPgSC

Método do potencial

se a operação i não tem expansão:

- $s_i = s_{i-1}$ e

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot n_i - s_i) + (2 \cdot n_{i-1} - s_{i-1}) \\ &= 1 + (2 \cdot n_i - s_i) + (2 \cdot (n_i - 1) - s_i) \\ &= 3 \end{aligned}$$

Método do potencial

se a operação i tem expansão:

- $s_i = 2 \times s_{i-1}$, $s_{i-1} = n_{i-1} = n_i - 1$ e

$$
\begin{aligned}
a_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= n_i + (2 \cdot n_i - s_i) - (2 \cdot n_{i-1} - s_{i-1}) \\
&= n_i + (2 \cdot n_i - 2 \cdot (n_i - 1)) - (2 \cdot (n_i - 1) - (n_i - 1)) \\
&= 3
\end{aligned}
$$

Remoção com contração de capacidade

- expansão dobra capacidade
- contração ocorre quando capacidade chega a $1/4$
- por quê não escolher $1/2$?

REMOVE(T)

```
 1  if T.num == 0
 2        T.size = 0
 3        FREE-TABLE(T.table)
 4  elseif T.num ≠ 0 and T.num * 4 < T.size // contração
 5        tab = T.table
 6        T.table = ALLOC-TABLE(T.size/2)
 7        for i = 1 to T.num
 8              T.table[i] = tab[i]
 9        T.size = T.size/2
10        FREE-TABLE(tab)
11  T.num = T.num − 1 // remoção
```

PPgSC

Análise com o método do potencial

- $\Phi = 0$ logo após contração ou expansão
- Fator de carga $\alpha(T) = T.num/T.size$, se T não for vazia, 1 se for.
- $T.num = \alpha(T) \cdot T.size$
- Função potencial

$$\Phi(T) = \begin{cases} 2 \cdot T.num - T.size & \text{se } \alpha T \geq 1/2 \\ T.size/2 - T.num & \text{se } 1/4 \leq \alpha T < 1/2 \end{cases}$$

A função potencial nunca é negativa: obteremos uma cota superior do custo real

PPgSC

Análise com o método do potencial

Justificativa intuitiva

- antes de uma expansão:
  - $\alpha(T) = 1$
  - $T.num = T.size$,
  - $\Phi(T) = T.num$
  - permite cópia de $T.num$ elementos.
- antes de uma contração:
  - $\alpha(T) = 1/4$
  - $T.num \cdot 4 = T.size$,
  - $\Phi(T) = T.size/2 - T.num = T.num$
  - permite cópia de $T.num$ elementos.

PPgSC

Análise com o método do potencial

Notações

s_i : capacidade após operação i

n_i : número de elementos após operação i

Φ_i : potencial após operação i

α_i : fator de carga após operação i

Inicialmente: $s_0 = n_0 = \Phi_0 = 0, \alpha_0 = 1$

Análise com o método do potencial

Inserção

A operação i é uma inserção:

- já fizemos a análise no caso $\alpha_{i-1} \geq 1/2$: $a_i = 3$
- se $\alpha_{i-1} < 1/2$, não há expansão
- se $\alpha_i \geq 1/2$:

$$
\begin{aligned}
a_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (2 \cdot n_i - s_i) - (s_{i-1}/2 - n_{i-1}) \\
&= 1 + (2 \cdot (n_{i-1} + 1) - s_{i-1}) - (s_{i-1}/2 - n_{i-1}) \\
&= 3 \cdot n_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&= 3 \cdot \alpha_{i-1} \cdot s_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&< \frac{3}{2} \cdot s_{i-1} - \frac{3}{2} \cdot s_{i-1} + 3 \\
&= 3
\end{aligned}
$$

PPgSC

Análise com o método do potencial

Inserção

A operação i é uma inserção:

- já fizemos a análise no caso $\alpha_{i-1} \geq 1/2$: $a_i = 3$
- se $\alpha_{i-1} < 1/2$, não há expansão
- se $\alpha_i < 1/2$:

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= 1 + (s_i/2 - n_i) - (s_i/2 - (n_i - 1)) \\ &= 0 \end{aligned}$$

O custo amortizado de uma inserção é 3 no máximo.

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} < 1/2$,
  - sem contração: $s_i = s_{i-1}$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= 1 + (s_i/2 - (n_{i-1} - 1)) - (s_i/2 - n_{i-1}) \\ &= 2 \end{aligned}$$

  - com contração: $n_{i-1} = n_i + 1 = s_i/2 = s_{i-1}/4$ e $c_i = n_i + 1$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= n_i + 1 + (s_i/2 - n_i) - (s_{i-1}/2 - n_{i-1}) \\ &= n_{i-1} + (n_{i-1} - (n_{i-1} - 1)) - (2 \cdot n_{i-1} - n_{i-1}) \\ &= 1 \end{aligned}$$

Análise com o método do potencial

Remoção

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} \geq 1/2$,
- não há contração: $s_i = s_{i-1}$, $n_i = n_{i-1} - 1$
- se $\alpha_i \geq 1/2$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (2 \cdot n_i - s_i) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= 1 + (2 \cdot (n_{i-1} - 1) - s_{i-1}) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= -1 \end{aligned}$$

PPgSC

$$n_i = n_{i-1} - 1$$

- se $\alpha_{i-1} \geq 1/2$,
- não há contração: $s_i = s_{i-1}$, $n_i = n_{i-1} - 1$
- se $\alpha_i < 1/2$

$$\begin{aligned} a_i &= c_i + \Phi_i - \Phi_{i-1} \\ &= 1 + (s_i/2 - n_i) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= 1 + (s_{i-1}/2 - (n_{i-1} - 1)) - (2 \cdot n_{i-1} - s_{i-1}) \\ &= 2 + \frac{3}{2} \cdot s_{i-1} - 3 \cdot n_{i-1} \\ &= 2 + \frac{3}{2} \cdot s_{i-1} - 3 \cdot \alpha_{i-1} \cdot s_{i-1} \\ &\leq 2 \end{aligned}$$

O custo amortizado de uma remoção é 2 no máximo.

- inserção tem custo amortizado $O(1)$
- remoção tem custo amortizado $O(1)$

Arranjos dinâmicos

Exercício

1. Projetar uma estrutura de dados para arranjos dinâmicos, que tem apenas operação de inserção, tal que esta tenha complexidade constante, *no pior caso*.
   - hipótese: supõe-se que o custo de uma alocação dinâmica é $\Theta(1)$.
   - dica: pode inspirar-se na aplicação do método do contado desta aula.
2. Estender a estrutura projetada com uma operação de remoção do último elemento (POP-BACK), tal que tanto a inserção quanto a remoção tenham complexidade constante, no pior caso.

Árvores chanfradas

splay trees

- árvores binárias de busca
- auto-ajustáveis
- não necessitam de atributos adicionais
- complexidade amortizada $O(\log n)$

PPgSC

Árvores chanfradas

- uma única operação básica: chanfrar
  - $\textsc{Splay}(T, k)$: chanfra a árvore T com relação ao valor k
  - no término da operação k está na raiz (ou o maior valor $< k$, ou o menor valor $> k$
- busca, inserção e remoção são todas realizadas após chanfrar a árvore

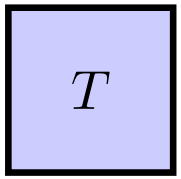
Realização das operações e o chanframento

- uma única operação básica: chanfrar
  - $\textsc{Splay}(T, k)$: chanfra a árvore T com relação ao valor k
  - busca a chave k
  - remaneja T tal que k fica na raiz (ou o maior valor $< k$, ou o menor valor $> k$
- busca, inserção e remoção são todas realizadas após chanfrar a árvore
  - $\textsc{Search}(T, k)$: $\textsc{Splay}(T, k)$ e consulta a raiz
  - $\textsc{Insert}(T, k)$: $\textsc{Splay}(T, k)$ e insere k na raiz
  - $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.
  - $O(1)$ aplicações de $\textsc{Splay}$ + $O(1)$ operações.

PPgSC

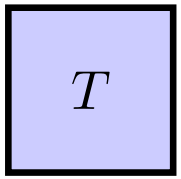
Busca

- $\textsc{Search}(T, x)$: $\textsc{Splay}(T, x)$ e consulta a raiz



Busca

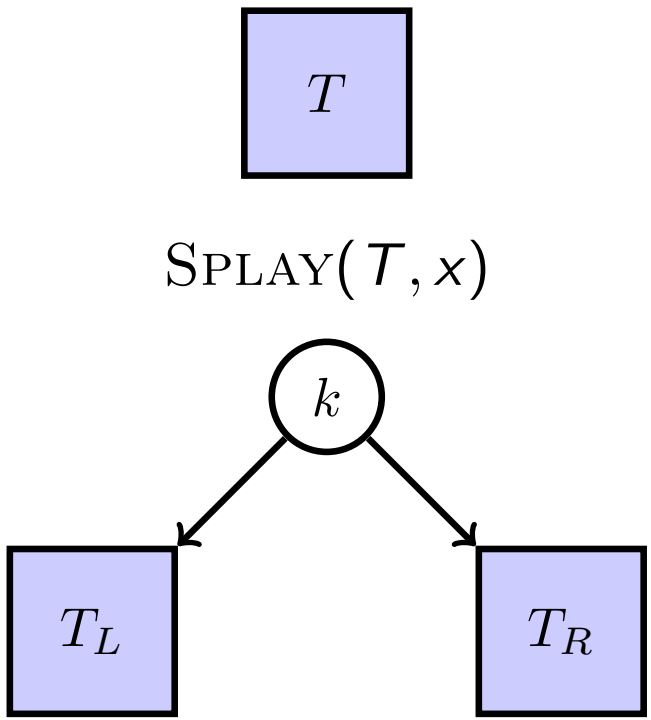
- $\textsc{Search}(T, x)$: $\textsc{Splay}(T, x)$ e consulta a raiz



$\textsc{Splay}(T, x)$

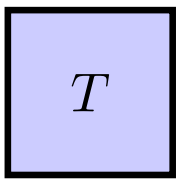
Busca

- $\textsc{Search}(T, x)$: $\textsc{Splay}(T, x)$ e consulta a raiz



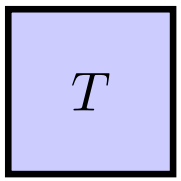
Inserção

- INSERT(T, k): SPLAY(T, k) e insere k na raiz



Inserção

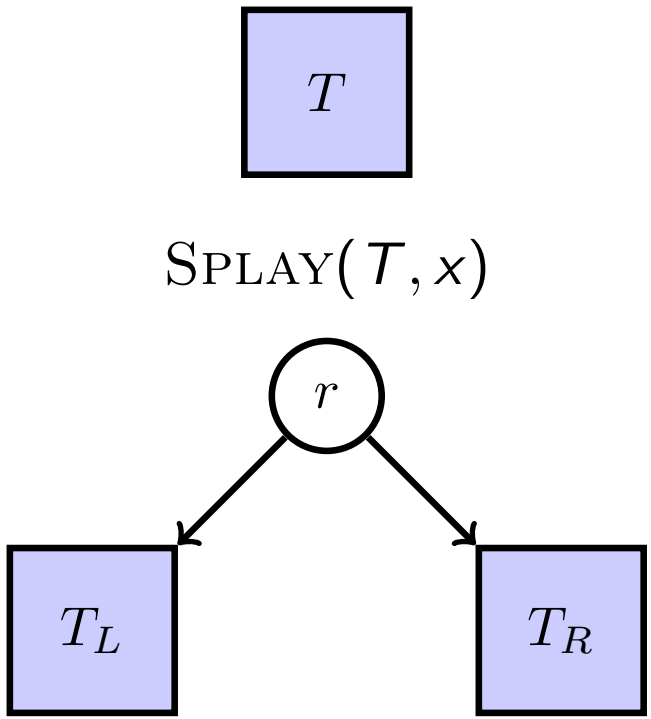
- INSERT(T, k): SPLAY(T, k) e insere k na raiz



SPLAY(T, x)

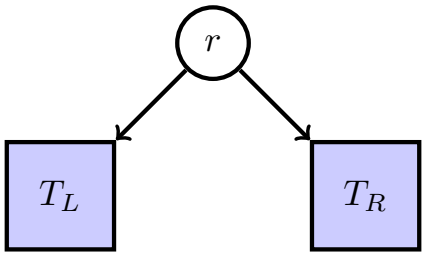
Inserção

- INSERT(T, k): SPLAY(T, k) e insere k na raiz



Inserção

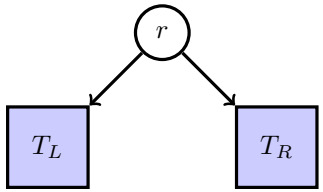
- INSERT(T, k): SPLAY(T, k) e insere k na raiz



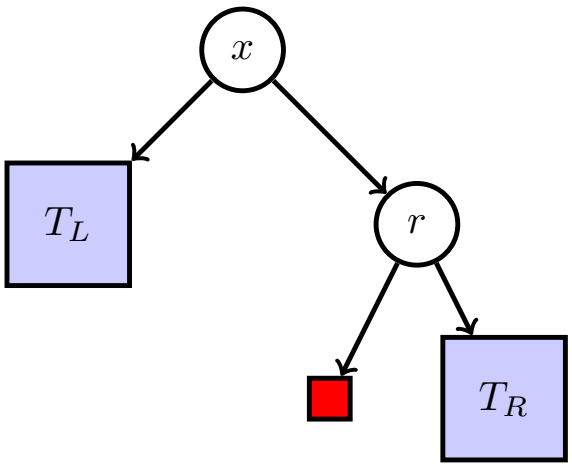
$$n.val = x;$$
$$n.left = T.root.left;$$
$$T.root.left = \text{NIL};$$
$$n.right = T.root;$$
$$T.root = n$$

Inserção

- INSERT(T, k): SPLAY(T, k) e insere k na raiz

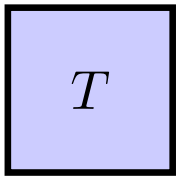


$$n.val = x;$$
$$n.left = T.root.left;$$
$$T.root.left = \text{NIL};$$
$$n.right = T.root;$$
$$T.root = n$$



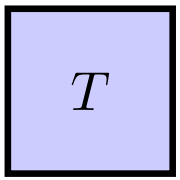
Remoção

- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



Remoção

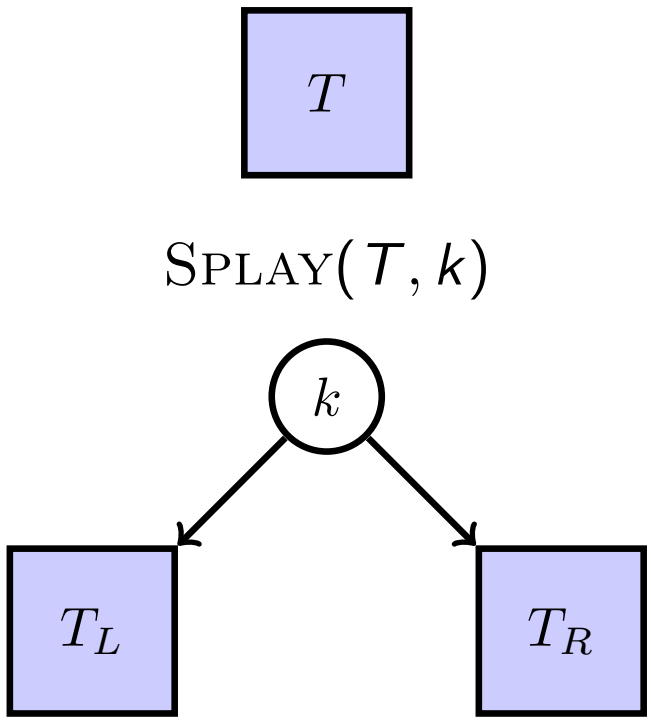
- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



$\textsc{Splay}(T, k)$

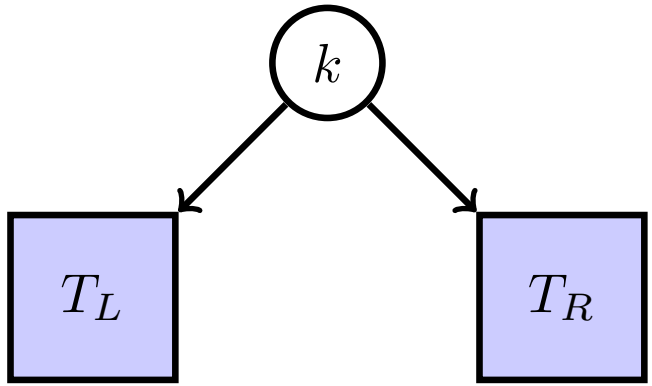
Remoção

- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



Remoção

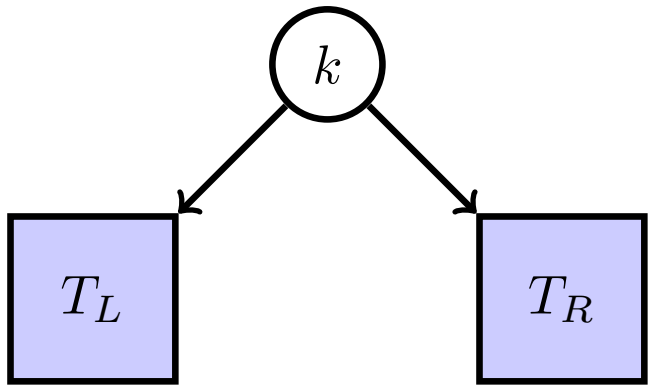
- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



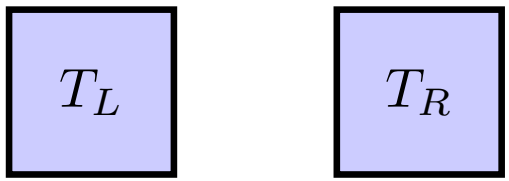
remove root: $n = T.root$; $T_L = n.left$; $T_R = n.right$; $\textsc{Free}(n)$

Remoção

- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.

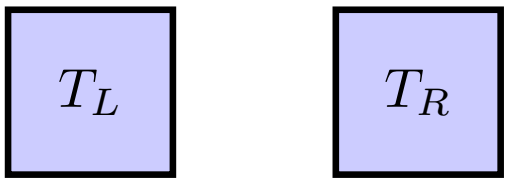


remove root: $n = T.root$; $T_L = n.left$; $T_R = n.right$; $\textsc{Free}(n)$



Remoção

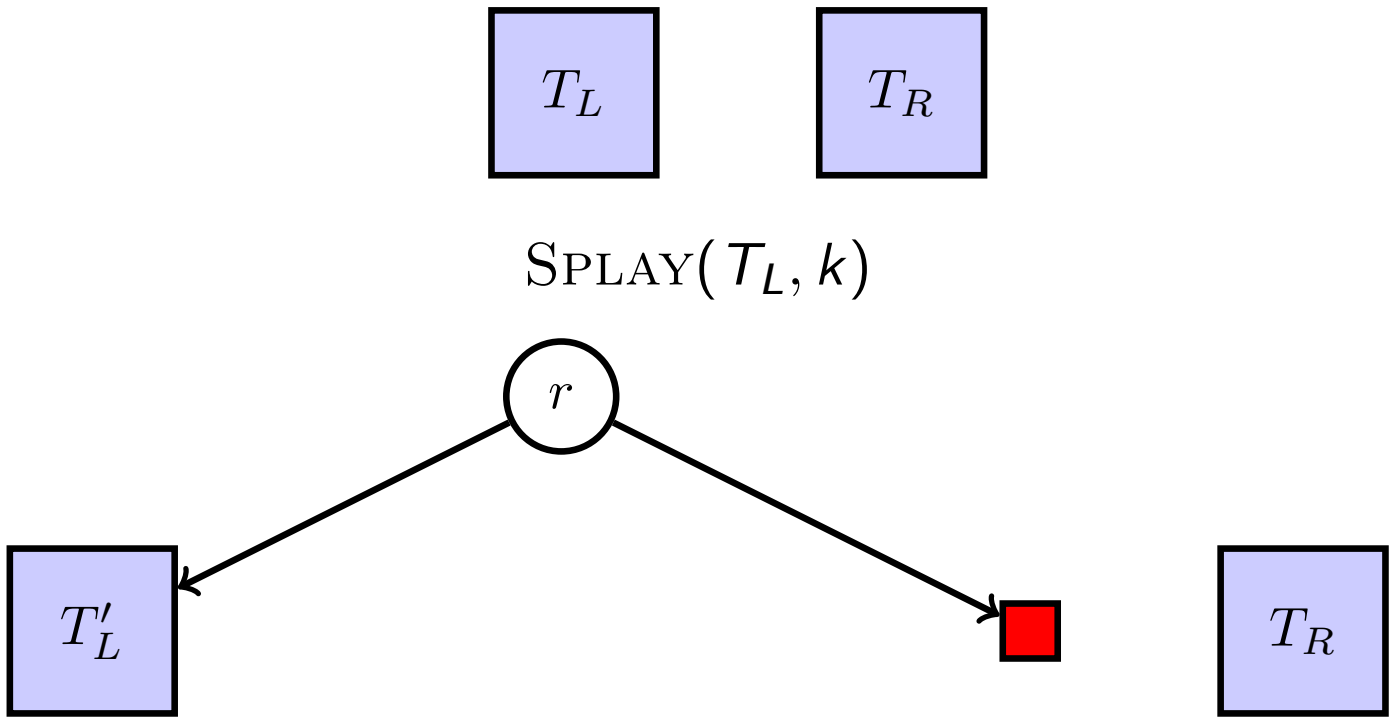
- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



$\textsc{Splay}(T_L, k)$

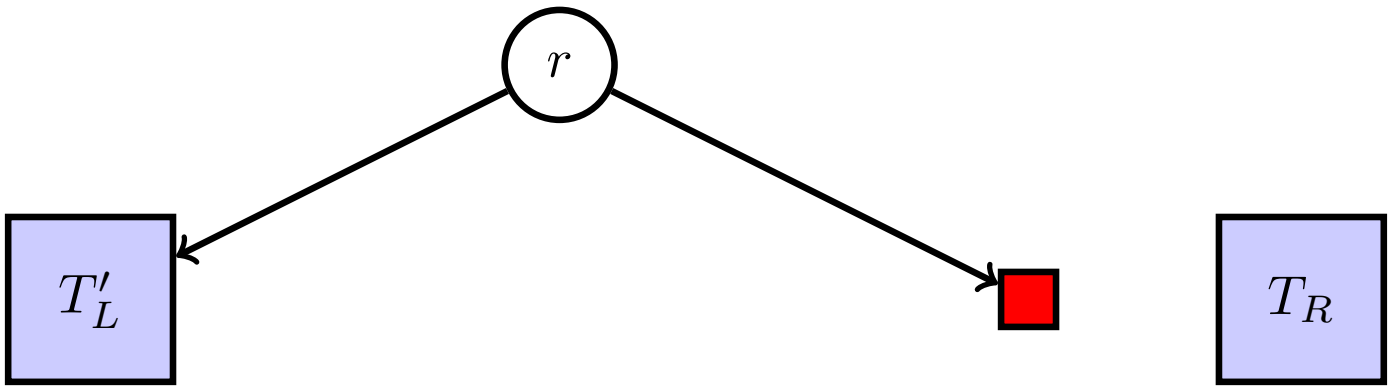
Remoção

- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



Remoção

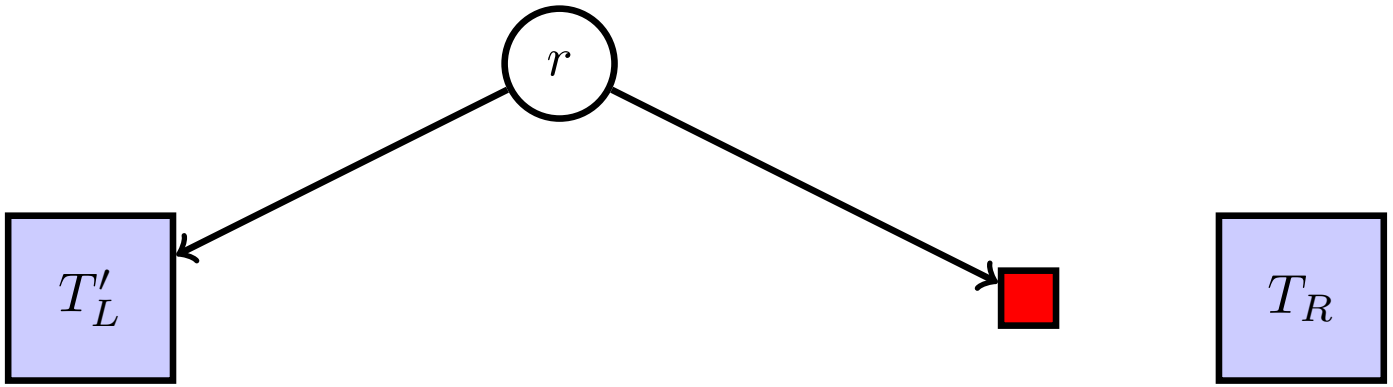
- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.



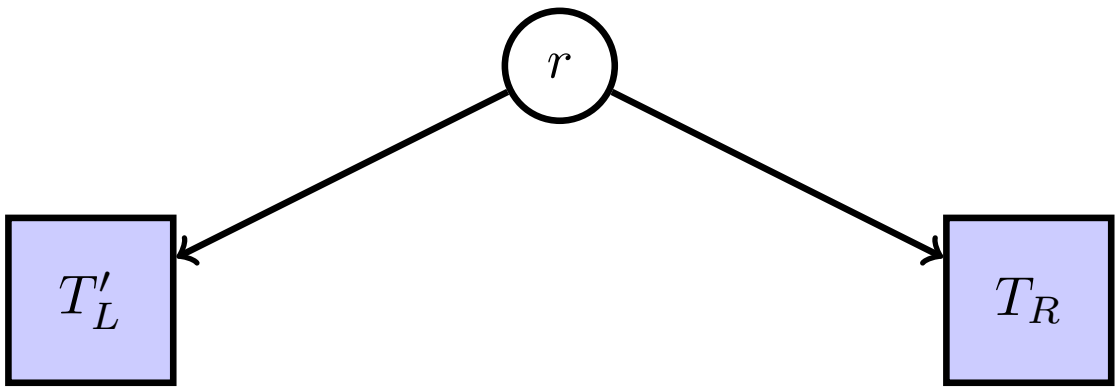
$$T_L.right = T_R$$

Remoção

- REMOVE(T, k): SPLAY(T, k), remove a raiz, T_L e T_R árvores resultantes, SPLAY(T_L, k), e torne T_R a sub-árvore da nova raiz de T_L.

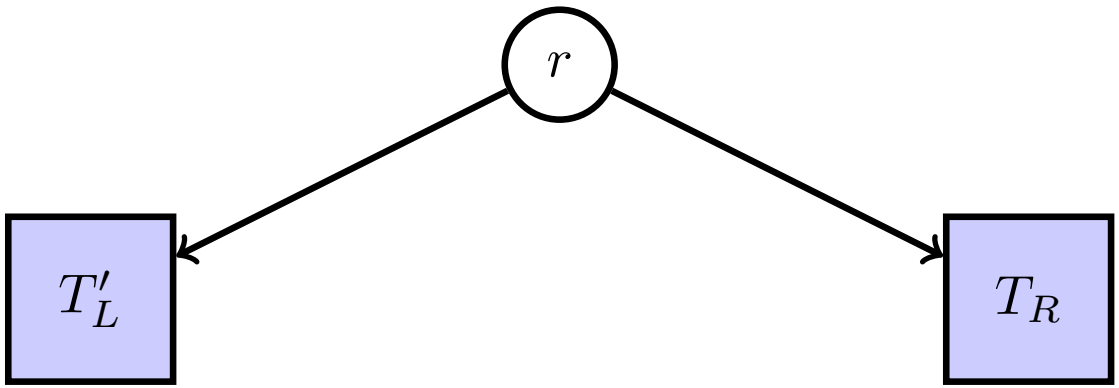


$$T_L.right = T_R$$



Remoção

- $\textsc{Remove}(T, k)$: $\textsc{Splay}(T, k)$, remove a raiz, T_L e T_R árvores resultantes, $\textsc{Splay}(T_L, k)$, e torne T_R a sub-árvore da nova raiz de T_L.

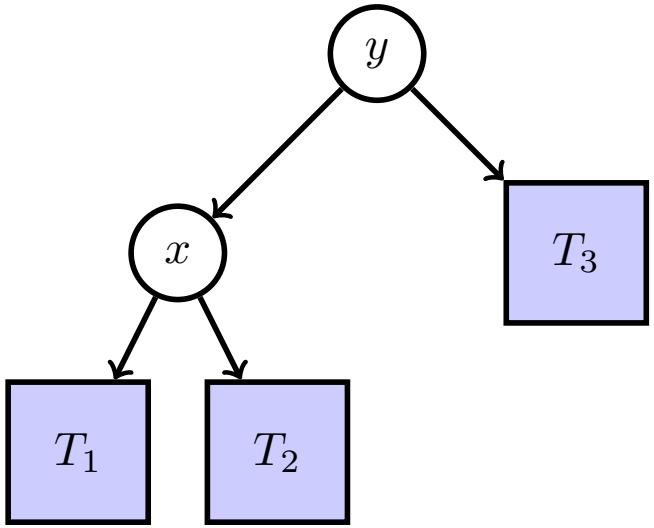


Chanfrar um valor

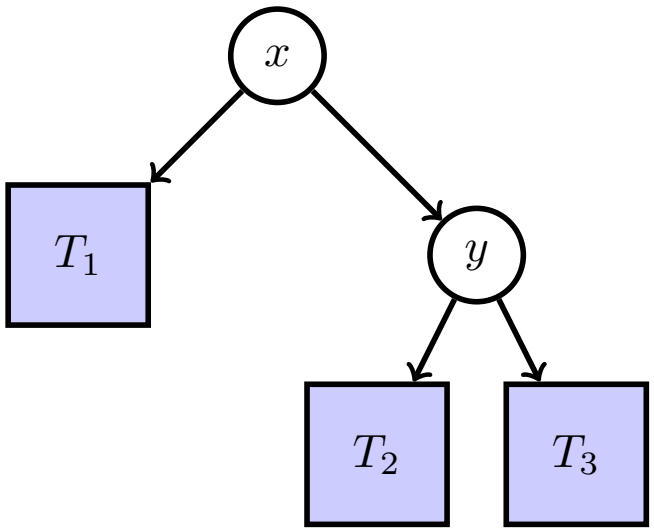
- aplicar rotações até chegar à raiz
- se x for um filho da raiz: rotação simples (caso 1)
- $x.up = y$ e $y.up = z$
  - se $x = y.left$ e $y = z.left$: rotação simples de y, rotação simples de x (caso 2)
  - se $x = y.right$ e $y = z.right$: rotação simples de y, rotação simples de x (caso 2)
  - se $x = y.right$ e $y = z.left$: rotação dupla de x (caso 3)
  - se $x = y.left$ e $y = z.right$: rotação dupla de x (caso 3)
- complexidade no pior caso $O(n)$
- análise amortizada mostra que todas as operações de chanframento tem custo amortizado $O(\log n)$

Chanfrar um valor — caso 1

se x for um filho da raiz: rotação simples



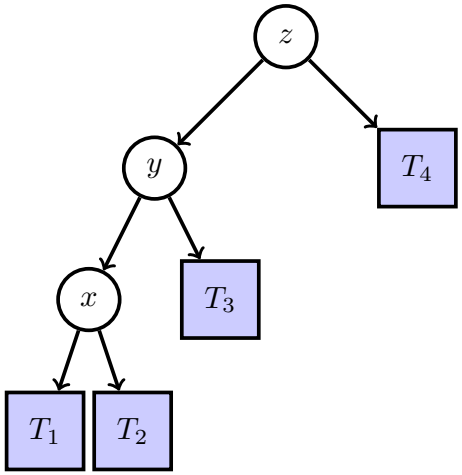
ROTATE-SIMPLE-RIGHT(x, y)



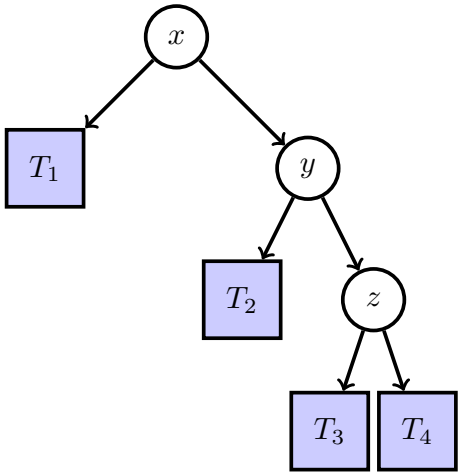
PPgSC

Chanfrar um valor — caso 2

$x = y.\mathit{left}$ e $y = z.\mathit{left}$ (e simétrico)

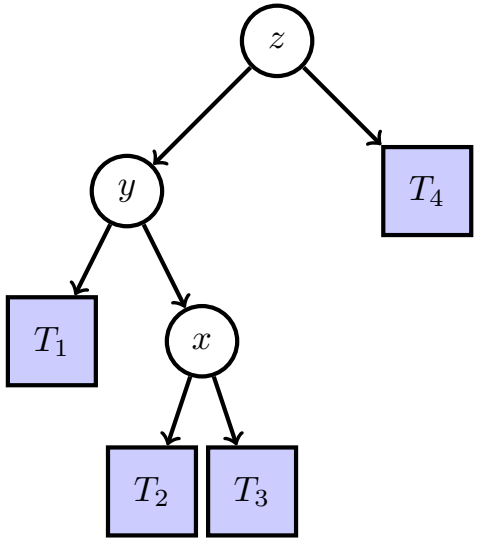


ROTATE-SIMPLE-RIGHT(y, z)
ROTATE-SIMPLE-RIGHT(x, y)

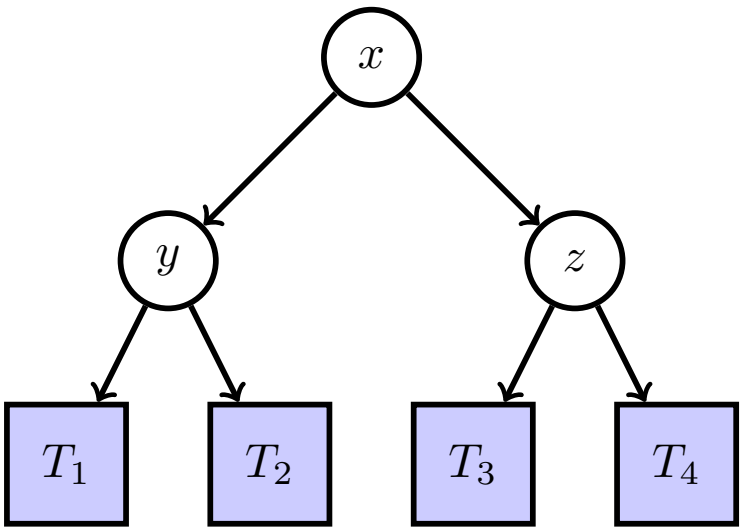


Chanfrar um valor — caso 3

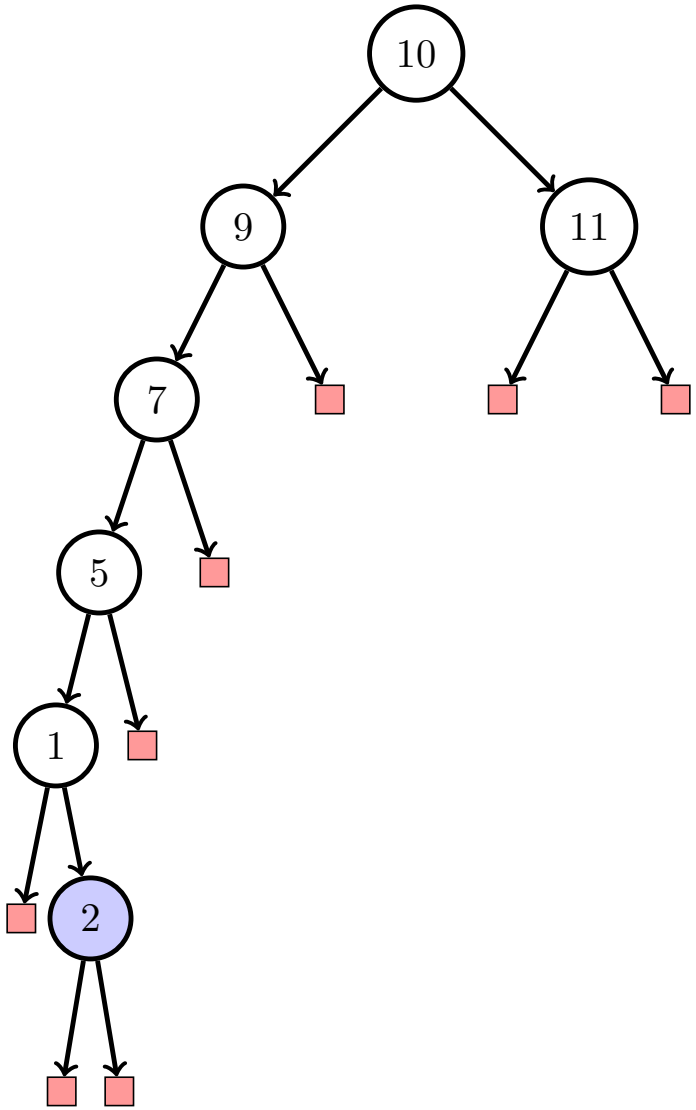
$x = y.\mathit{right}$ e $y = z.\mathit{left}$



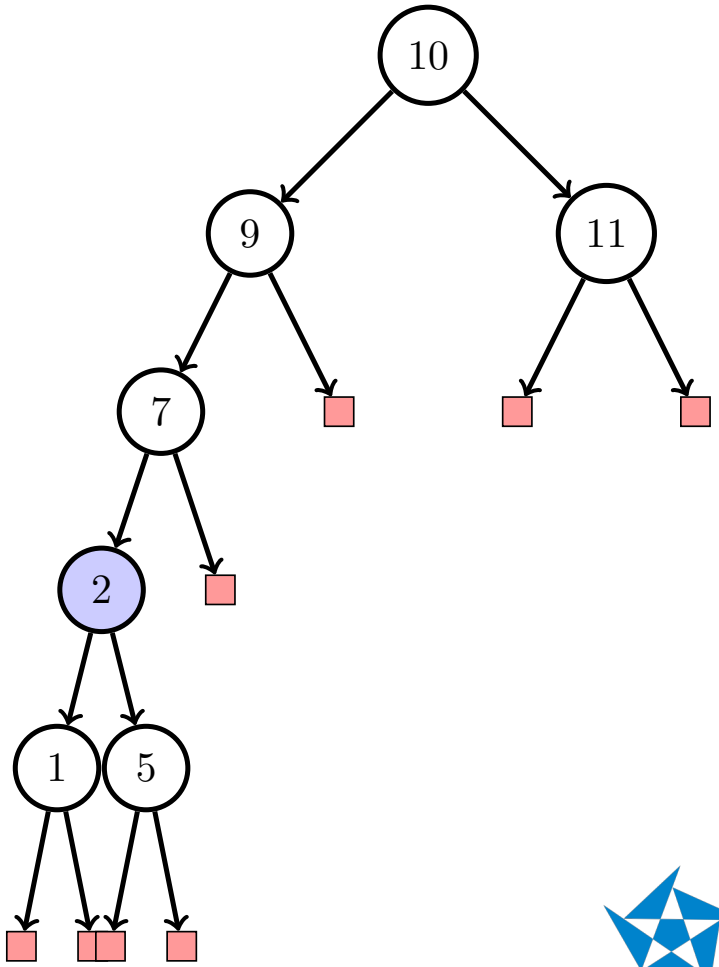
ROTATE-DOUBLE-RIGHT(x, y, z)



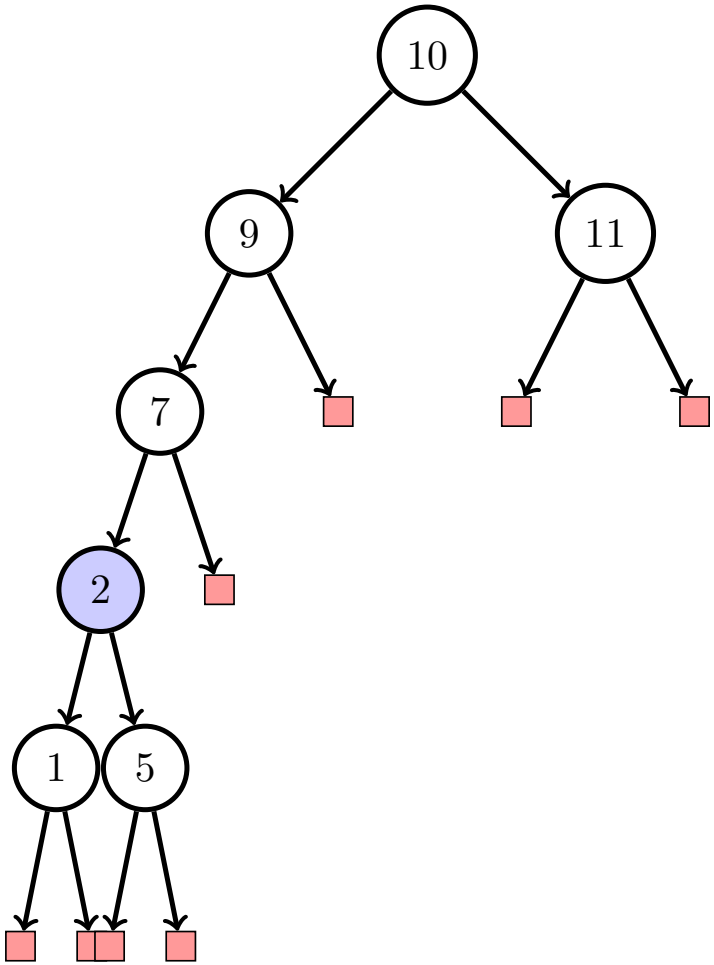
Chanfrar um valor: exemplo



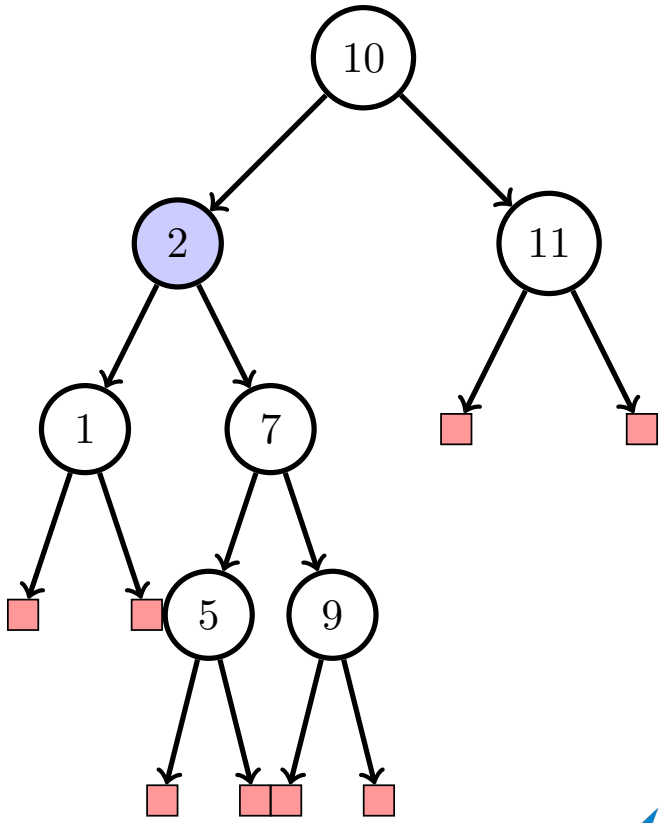
Caso 3



Chanfrar um valor: exemplo

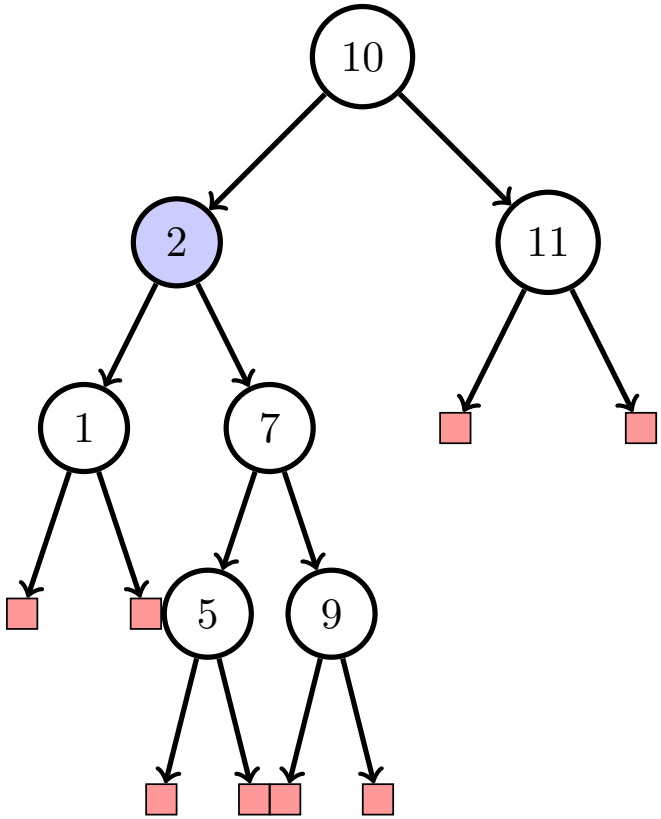


Caso 2

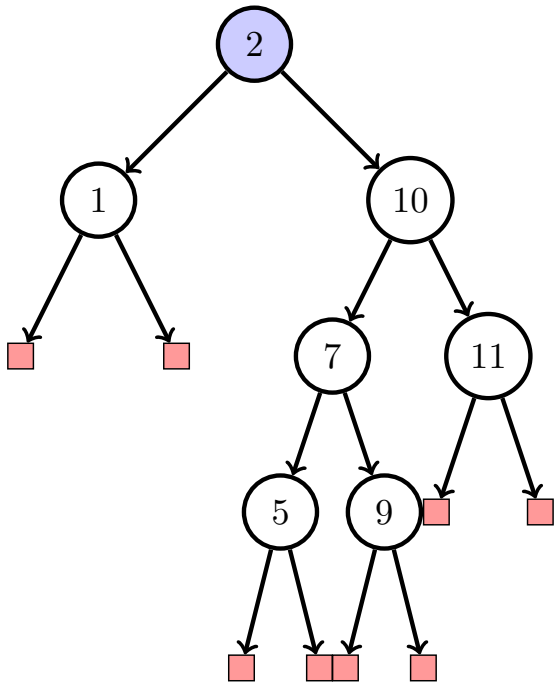


PPgSC

Chanfrar um valor: exemplo



Caso 1



Interlúdio matemático

Seja a, b dois números reais quaisquer

$$
\begin{aligned}
0 &\leq (a-b)^2 = a^2 - 2ab + b^2 \\
4ab &\leq a^2 + 2ab + b^2 \\
ab &\leq \frac{(a+b)^2}{4} \\
ab^{\frac{1}{2}} &\leq \frac{a+b}{2} \\
\log(ab^{\frac{1}{2}}) &\leq \log\frac{a+b}{2} \\
\log(a^{\frac{1}{2}}) + \log(b^{\frac{1}{2}}) &\leq \log\frac{a+b}{2} \\
\frac{1}{2} \times \log a + \frac{1}{2} \times \log b &\leq \log\frac{a+b}{2} \\
\frac{(\log a + \log b)}{2} &\leq \log\frac{a+b}{2}
\end{aligned}
$$

PPgSC

Análise amortizada: método do potencial

- seja $T_i(x)$ a sub-árvore enraizada em x após a operação i.
- $|T|$: número de nós em T
- o potencial fica distribuído entre os nós ($\phi_i = \sum_x \gamma_i(x)$)
- invariante: $\gamma_i(x) = \lfloor \log |T_i(x)| \rfloor$
- a complexidade amortizada de SPLAY é $O(\log n)$
  $T(x)$: sub-árvore enraizada em x
- propriedade: o custo de SPLAY(T, x) e manter o invariante é menor ou igual a $3 \times (\gamma(T) - \gamma(x)) + O(1)$. (vamos mostrar isto depois)
- logo SPLAY custa no máximo $3\lfloor \log n \rfloor + O(1) \in |\Theta(\log n)$.
- para pagar inserção, busca ou remoção, basta pagar $O(\log n)$.

Análise amortizada: método do potencial

- $\gamma(x)$ e $\gamma'(x)$: potencial em x antes e depois das rotações nos casos 1, 2 e 3
- SPLAY cai $k \geq 0$ vezes nos casos 2 e 3 e possivelmente uma vez no caso 1.
- vamos mostrar que o custo em cada caso é
  - caso 1: $\gamma'(x) - \gamma(x) + O(1)$
  - caso 2: $3 \times (\gamma'(x) - \gamma(x))$
  - caso 3: $3 \times (\gamma'(x) - \gamma(x))$
- somando, obtemos $3(\gamma(T) - \gamma(x)) + O(1)$

PPgSC

Caso 1

$y = x.up, y.up = \text{NIL}$

γ': potencial depois, γ: potencial antes

- os únicos nós que alteram seu potencial são x e y,
- logo $\phi' - \phi = (\gamma'(x) + \gamma'(y)) - (\gamma(x) + \gamma(y))$
- $\gamma'(x) = \gamma(y)$, $\gamma'(y) \leq \gamma'(x)$, $\gamma(x) \leq \gamma'(x)$
- logo

$$\begin{aligned} \gamma'(x) + \gamma'(y) - \gamma(x) - \gamma(y) &= \gamma'(y) - \gamma(x) \\ &\leq \gamma'(x) - \gamma(x) \\ &\leq 3 \times (\gamma'(x) - \gamma(x)) \end{aligned}$$

- é necessária apenas uma rotação:

$$a = c + \phi' - \phi \leq 1 + \gamma'(x) - \gamma(x)$$

Caso 2

$y = x.up, x = y.left, z = y.up, y = z.left$

- temos: $\gamma'(x) = \gamma(z)$, $\gamma'(y) \leq \gamma'(x)$, $\gamma'(z) \leq \gamma'(x)$, $\gamma(y) \geq \gamma(x)$
- logo

$$\begin{aligned} a &= c + \phi'(x) - \phi(x) \\ &= 2 + \gamma'(x) - \gamma(x) + \gamma'(y) - \gamma(y) + \gamma'(z) - \gamma(z) \\ &= 2 + (\gamma'(x) - \gamma(z)) + \gamma'(y) + \gamma'(z) - \gamma(x) - \gamma(y) \\ &= 2 + \gamma'(y) + \gamma'(z) - \gamma(x)) - \gamma(y) \\ &\leq 2 + \gamma'(x) + \gamma'(z) - \gamma(x) - \gamma(x) \\ &= 2 + \gamma'(x) + \gamma'(z) - 2 \times \gamma(x) \end{aligned}$$

- Vamos substituir $\gamma'(z)$ nesta expressão...

Caso 2 (continuação)

$$\frac{\log a + \log b}{2} \leq \log\left(\frac{a+b}{2}\right)$$

$$\begin{aligned}\gamma(x) + \gamma'(z) &= \log|T(x)| + \log|T'(z)| \\ &\leq 2 \times \log\left(\frac{|T(x)| + |T'(z)|}{2}\right)\end{aligned}$$

- observe que $|T(x)| + |T'(z)| \leq |T'(x)|$

$$\begin{aligned}\gamma(x) + \gamma'(z) &\leq 2 \times \log\left(\frac{|T'(x)|}{2}\right) \\ &= 2 \times \log|T'(x)| - 2 \times \log 2 \\ &= 2 \times \gamma'(x) - 2\end{aligned}$$

Logo:

$$\gamma'(z) \leq 2 \times \gamma'(x) - 2 - \gamma(x)$$

Caso 2 (continuação)

Temos:

$$\begin{aligned} a &= c + \phi'(x) - \phi(x) \\ &\leq \quad 2 + \gamma'(x) + \gamma'(z) - 2 \times \gamma(x) \end{aligned}$$

e:

$$\gamma'(z) \quad \leq \quad 2 \times \gamma'(x) - 2 - \gamma(x).$$

Logo, podemos prosseguir:

$$\begin{aligned} a &\leq 2 + \gamma'(x) + \gamma'(z) - 2 \times \gamma(x) \\ &\leq 2 + \gamma'(x) + 2 \times \gamma'(x) - 2 - \gamma(x) - 2 \times \gamma(x) \\ &= 3 \times \gamma'(x) - 3 \times \gamma(x) \\ &= 3 \times (\gamma'(x) - \gamma(x)) \end{aligned}$$

Caso 3

- Análogo ao caso 2
- Lemas úteis:
  1. $\gamma'(x) = \gamma(z)$;
  2. $\gamma(y) \geq \gamma(x)$;
  3. $\gamma'(y) + \gamma'(z) \leq 2 \times \gamma'(x) - 2$.
  4. $\gamma(x) \leq \gamma'(x)$.

$$
\begin{aligned}
a &= c + [\gamma'(x) + \gamma'(y) + \gamma'(z)] - [\gamma(x) + \gamma(y) + \gamma(z)] \\
&= 2 + \gamma'(y) + \gamma'(z) - \gamma(x) - \gamma(y) \quad \text{por 1)} \\
&\leq 2 + \gamma'(y) + \gamma'(z) - 2 \times \gamma(x) \quad \text{por 2)} \\
&\leq 2 + 2 \times \gamma'(x) - 2 - 2 \times \gamma(x) \quad \text{por 3)} \\
&= 2 \times (\gamma'(x) - \gamma(x)) \\
&\leq 3 \times (\gamma'(x) - \gamma(x))
\end{aligned}
$$

Conclusões

- Análise amortizada pode fornecer resultados mais apurados
- outro exemplo: árvores rubro-negras
  - inserção: $O(\log n)$
  - mas o custo amortizado de m inserções em uma árvore de n nós é $O(n + m)$

PPgSC