Árvore AVL

Estrutura de Dados Avançada — QXD0115

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

1º semestre/2023

Introdução

- Contexto: Temos um conjunto de n chaves $S = \{s_1, s_2, \ldots, s_n\}$ com probabilidade de acesso idênticas entre si.

Introdução

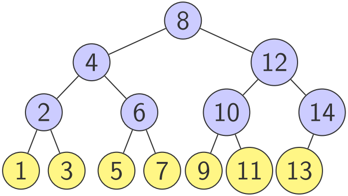
- Contexto: Temos um conjunto de n chaves $S = \{s_1, s_2, \ldots, s_n\}$ com probabilidade de acesso idênticas entre si.
- Dentre TODAS as árvores binárias de busca com n nós, as árvores binárias de busca completas são aquelas que minimizam o número de comparações efetuadas no pior caso para uma busca com chaves de probabilidades de ocorrência idênticas.
  - Uma árvore binária completa com $n > 0$ nós tem altura $h = 1 + \lfloor \lg n \rfloor$.

Introdução

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

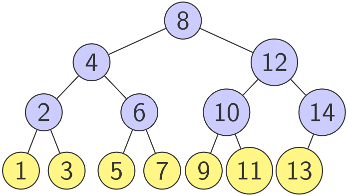
- Contexto: Temos um conjunto de n chaves $S = \{s_1, s_2, \ldots, s_n\}$ com probabilidade de acesso idênticas entre si.
- Dentre TODAS as árvores binárias de busca com n nós, as árvores binárias de busca completas são aquelas que minimizam o número de comparações efetuadas no pior caso para uma busca com chaves de probabilidades de ocorrência idênticas.
  - Uma árvore binária completa com $n > 0$ nós tem altura $h = 1 + \lfloor \lg n \rfloor$.
- Alguns questionamentos:
  - Seria possível manter a árvore sempre completa após consecutivas remoções ou inclusões?
  - Quanto custa isso? Vale o esforço?

Um exemplo ruim para o restabelecimento de árvores completas

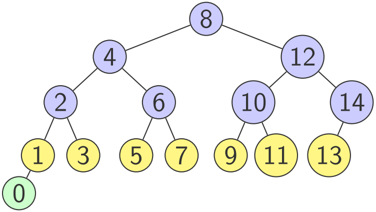


Vamos incluir 0

Um exemplo ruim para o restabelecimento de árvores completas

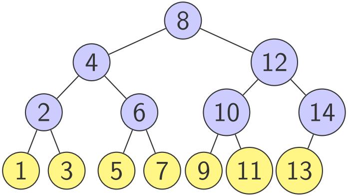


Vamos incluir 0

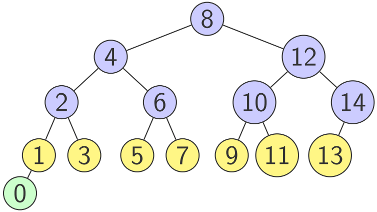


Não é mais completa

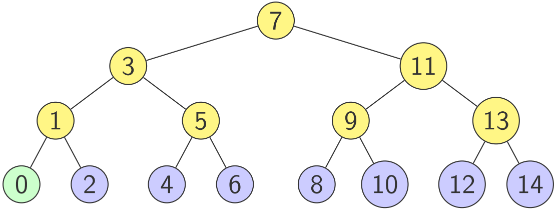
Um exemplo ruim para o restabelecimento de árvores completas



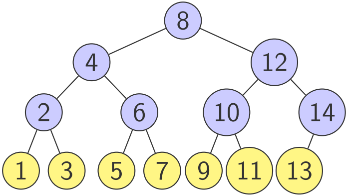
Vamos incluir 0



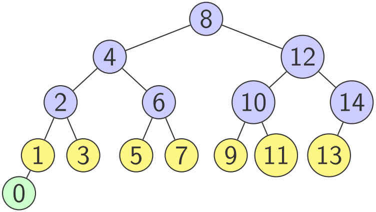
Não é mais completa



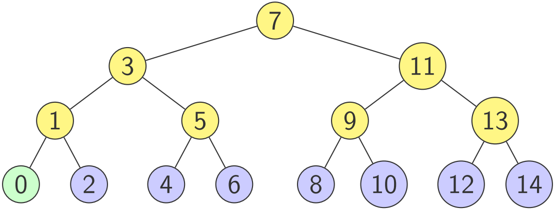
Um exemplo ruim para o restabelecimento de árvores completas



Vamos incluir 0



Não é mais completa



O algoritmo de restabelecimento requer, pelo menos, $\Omega(n)$ passos.
Árvores completas não são recomendadas para aplicações dinâmicas.

Alternativa — Árvores Balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore binária é balanceada se sua altura é da ordem de $O(\lg n)$ e, além disso, esta propriedade se estende a todas as suas subárvores:

Alternativa — Árvores Balanceadas

- Uma árvore binária é balanceada se sua altura é da ordem de $O(\lg n)$ e, além disso, esta propriedade se estende a todas as suas subárvores:
  - Cada subárvore que contém m nós deve possuir altura $O(\lg m)$.

Alternativa — Árvores Balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore binária é balanceada se sua altura é da ordem de $O(\lg n)$ e, além disso, esta propriedade se estende a todas as suas subárvores:
  - Cada subárvore que contém m nós deve possuir altura $O(\lg m)$.

- Nossa esperança: Como a forma de uma árvore balanceada é menos rígida que a de uma completa, torna-se “mais fácil” o seu rebalanceamento.

Árvore AVL

Árvore AVL

- Primeira árvore binária de busca a garantir tempo de execução $O(\log n)$ para inserção, busca e remoção no pior caso.
- Criada pelos soviéticos Adelson Vělsky e Landis, em 1962.

Vělsky

Landis

Árvore AVL

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore binária é do tipo AVL se, para todo nó v da árvore, temos que o valor absoluto da diferença entre as alturas da subárvore direita de v e da subárvore esquerda de v é no máximo 1.

Árvore AVL

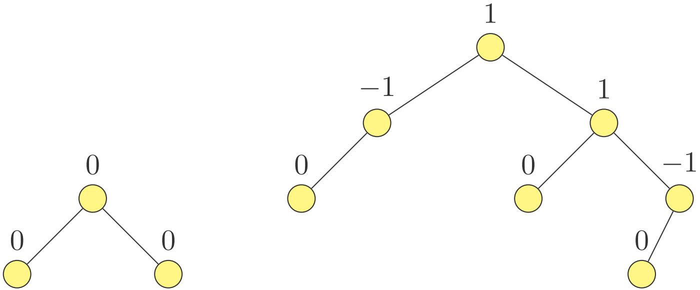
- Uma árvore binária é do tipo AVL se, para todo nó v da árvore, temos que o valor absoluto da diferença entre as alturas da subárvore direita de v e da subárvore esquerda de v é no máximo 1.
  - Um nó v que satisfaz essa propriedade é dito regulado; caso contrário, v é dito desregulado. Uma árvore que contém nó desregulado é dita desregulada.

Árvore AVL

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore binária é do tipo AVL se, para todo nó v da árvore, temos que o valor absoluto da diferença entre as alturas da subárvore direita de v e da subárvore esquerda de v é no máximo 1.
  - Um nó v que satisfaz essa propriedade é dito regulado; caso contrário, v é dito desregulado. Uma árvore que contém nó desregulado é dita desregulada.
- Fator de balanceamento $(fb(v))$: a diferença entre as alturas direita e esquerda de v, ou seja, $fb(v) = h_D(v) - h_E(v)$.

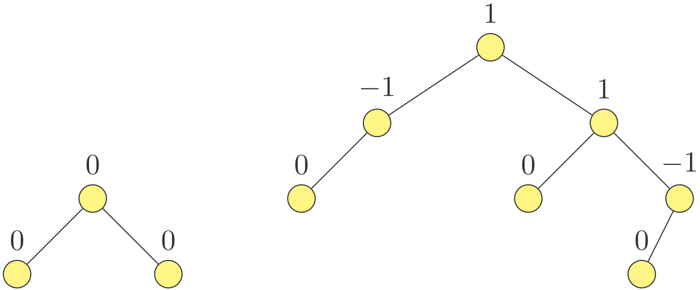
Exemplos de árvores binárias AVL



- Todos os nós nestas duas árvores estão regulados.

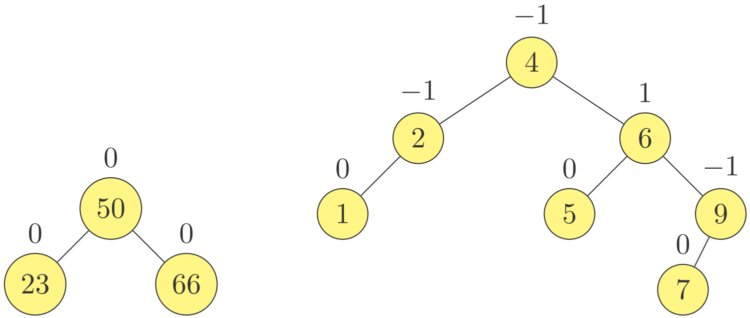
Exemplos de árvores binárias AVL

Universidade Federal do Ceará
Campus Quixadá



- Todos os nós nestas duas árvores estão regulados.
- Ou seja, para todo nó v, temos $|h_D(v) - h_E(v)| \leq 1$.

Exemplos de árvores binárias de busca AVL



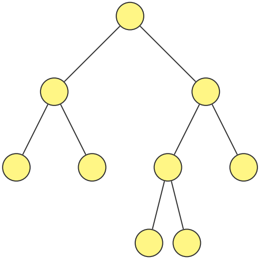
- Além de possuirem a propriedade AVL, essas árvores possuem a propriedade de serem binárias de busca.

Árvore completa $\times$ Árvore AVL

Fato: Toda árvore completa é AVL, mas nem toda árvore AVL é completa. $\square$

Árvore completa × Árvore AVL

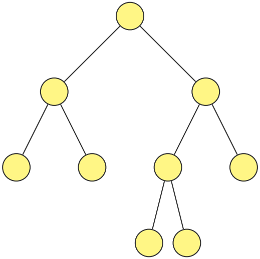
Fato: Toda árvore completa é AVL, mas nem toda árvore AVL é completa. □



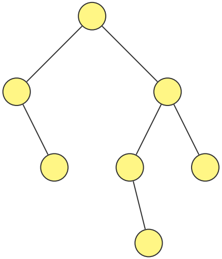
Árvore completa

Árvore completa × Árvore AVL

Fato: Toda árvore completa é AVL, mas nem toda árvore AVL é completa. □



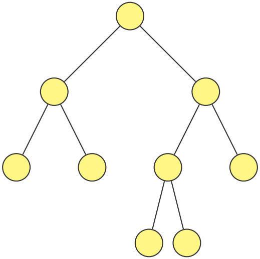
Árvore completa



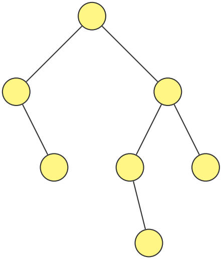
É AVL mas não é completa. Por quê?

Árvore completa × Árvore AVL

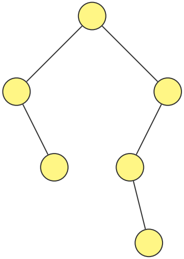
Fato: Toda árvore completa é AVL, mas nem toda árvore AVL é completa. □



Árvore completa



É AVL mas não é completa. Por quê?



Não é completa e nem AVL. Por quê?

Prova do balanceamento

Balanceamento de árvores AVL

Como provar que toda árvore AVL é balanceada?

Balanceamento de árvores AVL

Como provar que toda árvore AVL é balanceada?

- Ideia 1: Considerar uma árvore AVL com n nós e determinar o valor máximo de sua altura h.

Balanceamento de árvores AVL

Como provar que toda árvore AVL é balanceada?

- Ideia 1: Considerar uma árvore AVL com n nós e determinar o valor máximo de sua altura h.

Ou, equivalentemente:

- Ideia 2: Fixar h e determinar o valor mínimo do número n de nós.
  - Vamos seguir por esta linha de raciocínio.

Balanceamento de árvores AVL

Como provar que toda árvore AVL é balanceada?

- Ideia 1: Considerar uma árvore AVL com n nós e determinar o valor máximo de sua altura h.

Ou, equivalentemente:

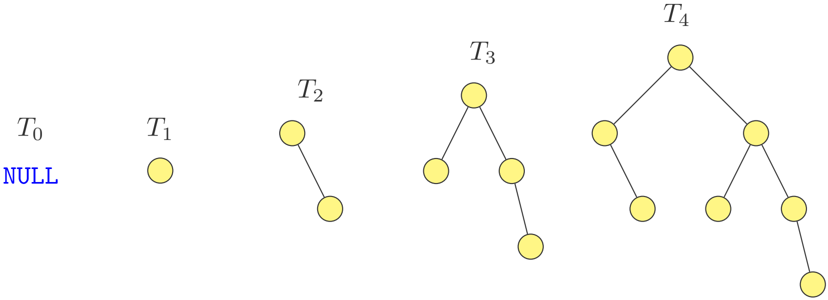
- Ideia 2: Fixar h e determinar o valor mínimo do número n de nós.
  - Vamos seguir por esta linha de raciocínio.

Problema

Dada uma árvore AVL de altura h, qual seria o valor mínimo possível para n?

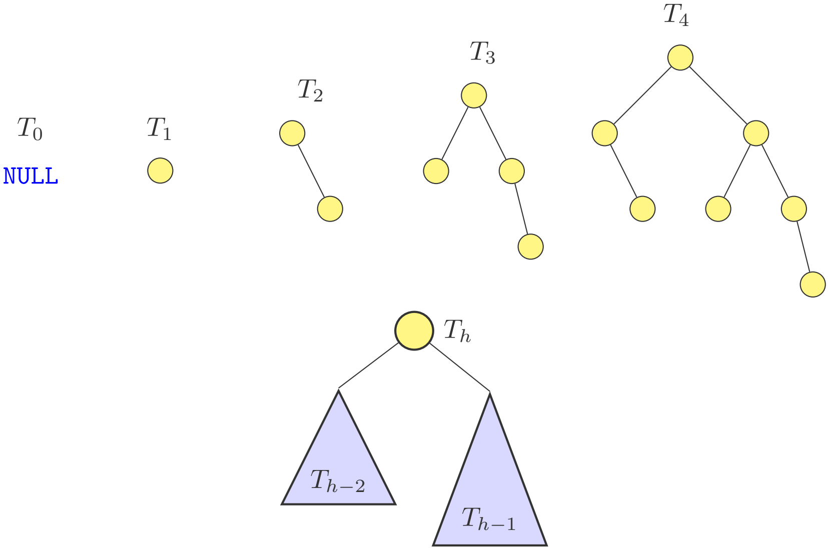
Árvores de Fibonacci

- Árvores AVL com os piores fatores de balanceamento.



Árvores de Fibonacci

- Árvores AVL com os piores fatores de balanceamento.



Número de nós — Árvores de Fibonacci

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Número de nós de T_h:

$$N(T_h) = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ 1 + N(T_{h-1}) + N(T_{h-2}) & \text{se } h > 1. \end{cases}$$

Número de nós — Árvores de Fibonacci

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Número de nós de T_h:

$$N(T_h) = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ 1 + N(T_{h-1}) + N(T_{h-2}) & \text{se } h > 1. \end{cases}$$

- A fórmula acima, lembra a fórmula do h-ésimo termo da sequência de Fibonacci:

$$F_h = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ F_{h-1} + F_{h-2} & \text{se } h > 1. \end{cases}$$

Número de nós — Árvores de Fibonacci

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Número de nós de T_h:

$$N(T_h) = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ 1 + N(T_{h-1}) + N(T_{h-2}) & \text{se } h > 1. \end{cases}$$

- A fórmula acima, lembra a fórmula do h-ésimo termo da sequência de Fibonacci:

$$F_h = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ F_{h-1} + F_{h-2} & \text{se } h > 1. \end{cases}$$

- Logo, $N(T_h) \geq F_h$, para todo $h \geq 0$.

Número de nós — Árvores de Fibonacci

Fato: Dada uma árvore AVL T de altura h, temos que:
$N(T) \geq N(T_h) \geq F_h$.

Número de nós — Árvores de Fibonacci

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Fato: Dada uma árvore AVL T de altura h, temos que:
$N(T) \geq N(T_h) \geq F_h$.

Fórmula do h-ésimo termo da sequência de Fibonacci:

$$F_h = \begin{cases} 0 & \text{se } h = 0; \\ 1 & \text{se } h = 1; \\ F_{h-1} + F_{h-2} & \text{se } h > 1. \end{cases}$$

Fato: Para $h > 1$, o h-ésimo termo da sequência de Fibonacci é dado por:

$$F_h = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^h - \left(\frac{1-\sqrt{5}}{2} \right)^h \right]$$

Prova $h = O(\lg n)$

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Se T é uma árvore AVL com altura h e com n nós, então $h = O(\lg n)$.

Prova: Do slide anterior, temos que $n \geq F_h = \frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^h - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^h$.

Como $h > 0$, temos que $\frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^h < 1$.

Portanto, $n \geq F_h > \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^h - 1$.

Fazendo $a = \frac{1+\sqrt{5}}{2}$, tem-se $n > \frac{1}{\sqrt{5}}a^h - 1$. O que implica, $n + 1 > \frac{a^h}{\sqrt{5}}$.
Aplicando logaritmo na base a em ambos os lados, temos:

$$\begin{aligned}
\log_a(n+1) &> \log_a a^h - \log_a \sqrt{5} \\
\log_a(n+1) &> h - \log_a \sqrt{5} \\
h &< \log_a(n+1) + \log_a \sqrt{5} \\
h &< \frac{1}{\log_2 a}\log_2(n+1) + \log_a \sqrt{5} \quad \text{(mudança de base)} \\
h &< 1.44 \cdot \log_2(n+1) + 1.67 = O(\lg n).
\end{aligned}$$

Inserção em árvores AVL

Inserção em Árvores AVL

- Ideia: Após cada inserção, verificar se algum nó p se encontra desregulado.
  - Em caso positivo, aplicar transformações apropriadas para regulá-lo.

Inserção em Árvores AVL

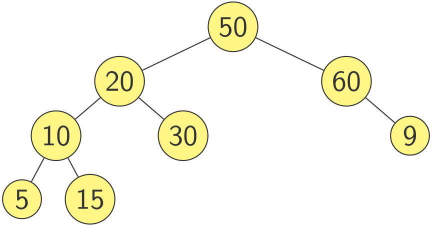
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Ideia: Após cada inserção, verificar se algum nó p se encontra desregulado.
  - Em caso positivo, aplicar transformações apropriadas para regulá-lo.
- Pergunta: Após a inserção de um nó, quais nós podem ter se tornado desregulados?

Inserção em Árvores AVL

- Ideia: Após cada inserção, verificar se algum nó p se encontra desregulado.
  - Em caso positivo, aplicar transformações apropriadas para regulá-lo.

- Pergunta: Após a inserção de um nó, quais nós podem ter se tornado desregulados?
  Exemplo: Inserir 3 na árvore abaixo.



Inserção em Árvores AVL

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

Inserção em Árvores AVL

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.

Inserção em Árvores AVL

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
  - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

Inserção em Árvores AVL

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
  - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

- Usaremos basicamente quatro tipos de rotações:

Inserção em Árvores AVL

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
  - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

- Usaremos basicamente quatro tipos de rotações:
  - Rotação esquerda

Inserção em Árvores AVL

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
    - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

- Usaremos basicamente quatro tipos de rotações:
    - Rotação esquerda
    - Rotação direita

Inserção em Árvores AVL

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
  - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

- Usaremos basicamente quatro tipos de rotações:
  - Rotação esquerda
  - Rotação direita
  - Rotação dupla à esquerda

Inserção em Árvores AVL

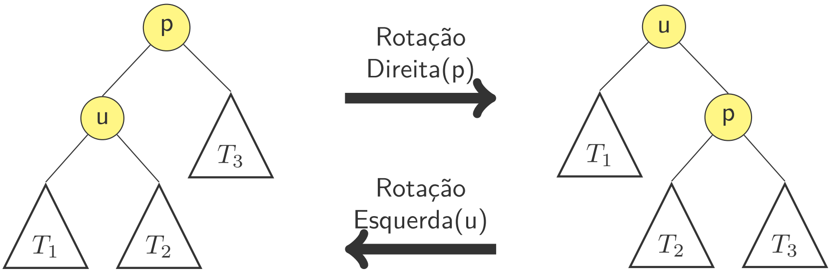
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Em alguns casos, a inserção de um novo nó x na árvore AVL pode vir a modificar o fator de balanceamento de algum nó no caminho que vai de x até a raiz.

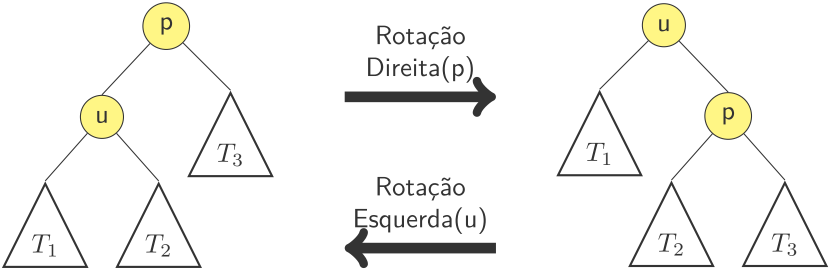
- No caso em que algum nó v neste caminho ficar desregulado, uma operação de regulagem do nó v deve ser realizada a fim de regular o nó.
  - A essa operação de regulagem do nó v chamaremos de rotação do nó v.

- Usaremos basicamente quatro tipos de rotações:
  - Rotação esquerda
  - Rotação direita
  - Rotação dupla à esquerda
  - Rotação dupla à direita

Rotação Esquerda e Rotação Direita



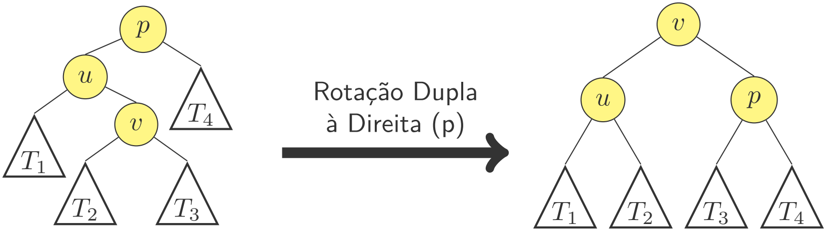
Rotação Esquerda e Rotação Direita



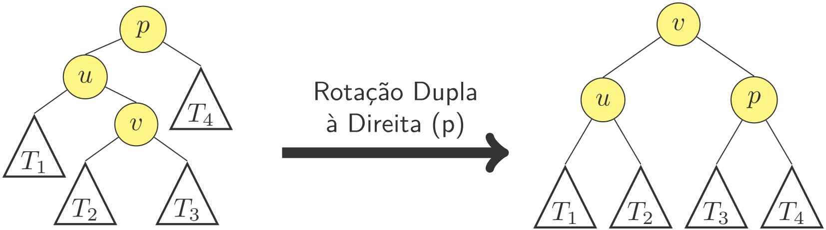
Propriedade 1

As rotações preservam a natureza da árvore como sendo binária de busca. □

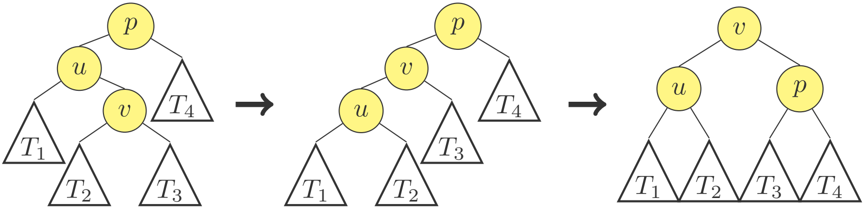
Rotação Dupla à Direita



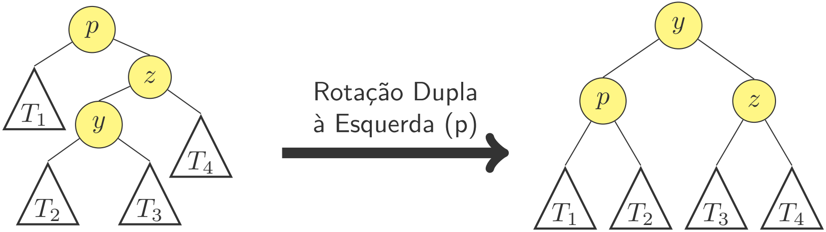
Rotação Dupla à Direita



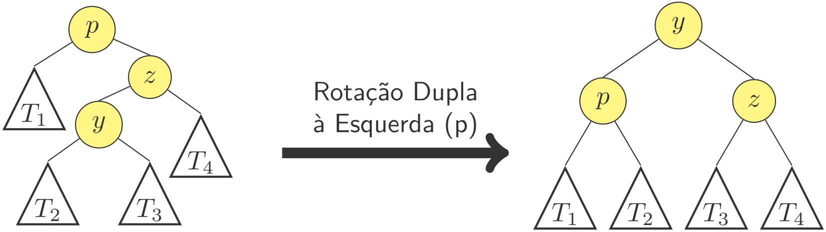
- Rotação Dupla à Direita = Rot. Esquerda(u) + Rot. Direita(p)



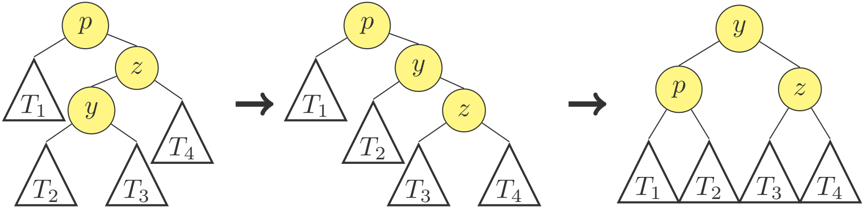
Rotação Dupla à Esquerda



Rotação Dupla à Esquerda

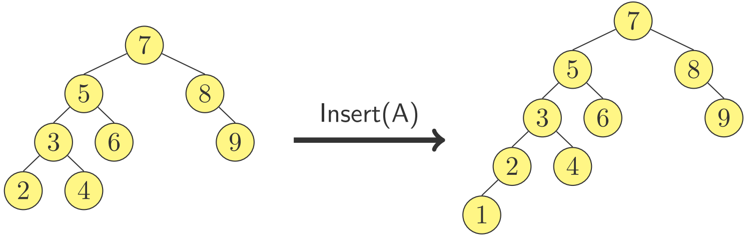


- Rotação Dupla à Esquerda = Rot. Direita(z) + Rot. Esquerda(p)



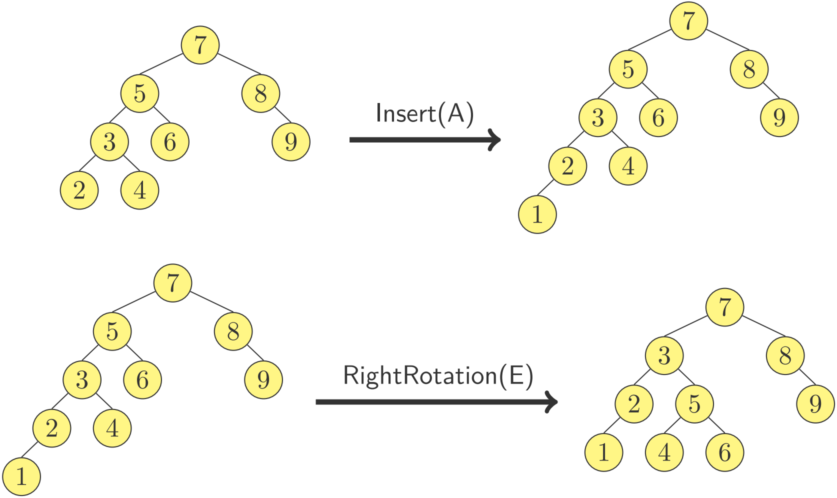
Exemplo

Inclusão da chave 1 na árvore e posterior efeito de uma rotação direita do nó 5, que tornou-se desregulado após a inclusão de 1.



Exemplo

Inclusão da chave 1 na árvore e posterior efeito de uma rotação direita do nó 5, que tornou-se desregulado após a inclusão de 1.



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Análise da operação de inserção

Análise da Inserção

- Após a inserção de um nó x na árvore, as chamadas recursivas vão se "desenrolando" e todos os vértices no caminho de x até a raiz da árvore devem ter seus fatores de balanceamento devidamente checados.

Análise da Inserção

- Após a inserção de um nó x na árvore, as chamadas recursivas vão se "desenrolando" e todos os vértices no caminho de x até a raiz da árvore devem ter seus fatores de balanceamento devidamente checados.
- Vamos provar que, uma vez que um nó p torna-se desregulado, a regulagem de p é restabelecida pela aplicação de uma das rotações AVL estudadas.

Análise da Inserção

- Suponha que o nó x acabou de ser inserido em T.

Análise da Inserção

Universidade Federal do Ceará
Campus Quixadá

- Suponha que o nó x acabou de ser inserido em T.
- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.

Análise da Inserção

- Suponha que o nó x acabou de ser inserido em T.
- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.
- Caso contrário, seja p o ancestral de x mais próximo que se tornou desregulado.

Análise da Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Suponha que o nó x acabou de ser inserido em T.
- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.
- Caso contrário, seja p o ancestral de x mais próximo que se tornou desregulado.
  - Temos que $|h_D(p) - h_E(p)| = 2$ pois T era uma árvore AVL antes da inclusão de x e, além disso, a inclusão de um nó não pode aumentar em mais de uma unidade a altura de qualquer subárvore.

Análise da Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Suponha que o nó x acabou de ser inserido em T.
- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.
- Caso contrário, seja p o ancestral de x mais próximo que se tornou desregulado.
  - Temos que $|h_D(p) - h_E(p)| = 2$ pois T era uma árvore AVL antes da inclusão de x e, além disso, a inclusão de um nó não pode aumentar em mais de uma unidade a altura de qualquer subárvore.
- Há exatamente dois casos a considerar:

Análise da Inserção

- Suponha que o nó x acabou de ser inserido em T.
- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.
- Caso contrário, seja p o ancestral de x mais próximo que se tornou desregulado.
  - Temos que $|h_D(p) - h_E(p)| = 2$ pois T era uma árvore AVL antes da inclusão de x e, além disso, a inclusão de um nó não pode aumentar em mais de uma unidade a altura de qualquer subárvore.
- Há exatamente dois casos a considerar:
  - **Caso (1):** $h_E(p) > h_D(p)$

Análise da Inserção

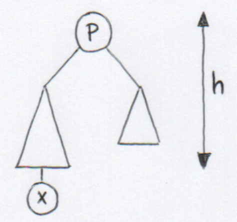
- Suponha que o nó x acabou de ser inserido em T.

- Se após a inclusão de x todos os nós mantiveram-se regulados, então a árvore manteve-se AVL e não há nada o que efetuar.

- Caso contrário, seja p o ancestral de x mais próximo que se tornou desregulado.
  - Temos que $|h_D(p) - h_E(p)| = 2$ pois T era uma árvore AVL antes da inclusão de x e, além disso, a inclusão de um nó não pode aumentar em mais de uma unidade a altura de qualquer subárvore.

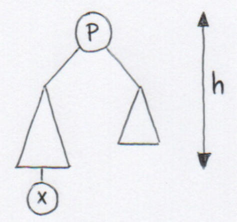
- Há exatamente dois casos a considerar:
  - **Caso (1):** $h_E(p) > h_D(p)$
  - **Caso (2):** $h_D(p) > h_E(p)$

Caso 1: $h_E(p) > h_D(p)$



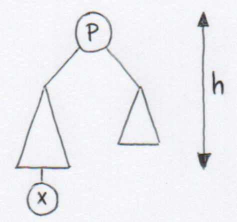
- O nó x foi inserido na subárvore esquerda de p.

Caso 1: $h_E(p) > h_D(p)$



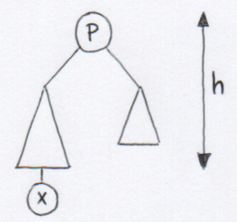
- O nó x foi inserido na subárvore esquerda de p.
- p possui o filho esquerdo u, $u \neq x$. Pois caso contrário, p não estaria desregulado.

Caso 1: $h_E(p) > h_D(p)$



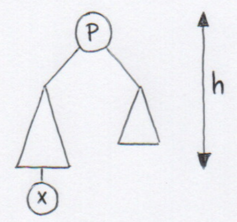
- O nó x foi inserido na subárvore esquerda de p.
- p possui o filho esquerdo u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.

Caso 1: $h_E(p) > h_D(p)$



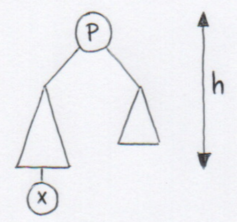
- O nó x foi inserido na subárvore esquerda de p.
- p possui o filho esquerdo u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:

Caso 1: $h_E(p) > h_D(p)$



- O nó x foi inserido na subárvore esquerda de p.
- p possui o filho esquerdo u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:
  - **Caso 1(a):** $h_E(u) > h_D(u)$.

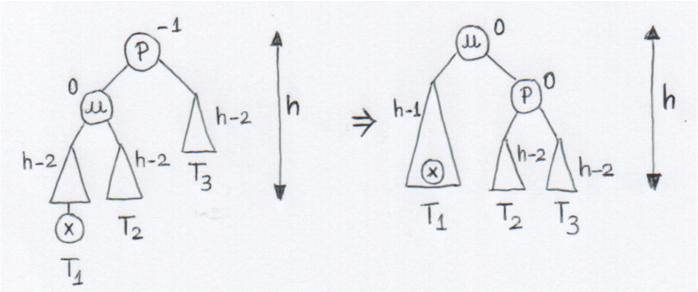
Caso 1: $h_E(p) > h_D(p)$



- O nó x foi inserido na subárvore esquerda de p.
- p possui o filho esquerdo u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:
  - **Caso 1(a):** $h_E(u) > h_D(u)$.
  - **Caso 1(b):** $h_E(u) < h_D(u)$.

Caso 1(a): $h_E(u) > h_D(u)$

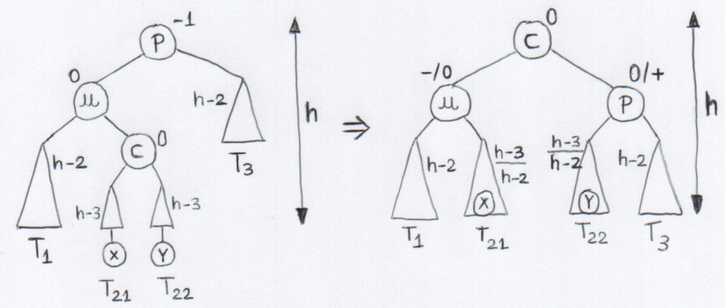
Solução: Rotação direita simples em p.



O nó x é inserido à esquerda de u.
Note que $h(T_1) = h(T_2) + 1$.
Após a rotação simples, a altura final permanece inalterada.
Nenhuma modificação futura é necessária.

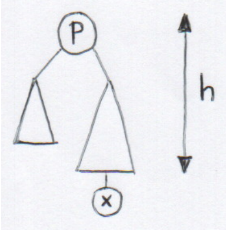
Caso 1(b): $h_E(u) < h_D(u)$

Solução: Rotação dupla direita.



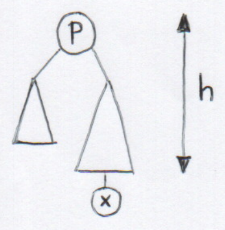
O nó inserido pode ser X ou Y.
Quando $h = 2$, as árvores T_1 e T_3 são vazias, e o nó inserido é o próprio nó C; neste caso as árvores T_{21} e T_{22} são vazias.
A altura final permanece inalterada.
Nenhuma modificação futura é necessária.

Caso 2: $h_E(p) < h_D(p)$



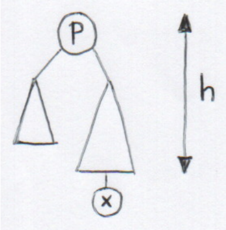
- O nó x foi inserido na subárvore direita de p.

Caso 2: $h_E(p) < h_D(p)$



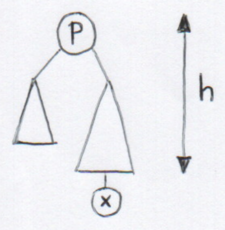
- O nó x foi inserido na subárvore direita de p.
- p possui o filho direito u, $u \neq x$. Pois caso contrário, p não estaria desregulado.

Caso 2: $h_E(p) < h_D(p)$



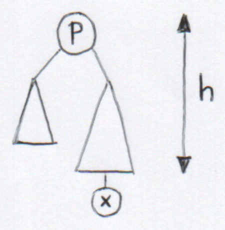
- O nó x foi inserido na subárvore direita de p.
- p possui o filho direito u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.

Caso 2: $h_E(p) < h_D(p)$



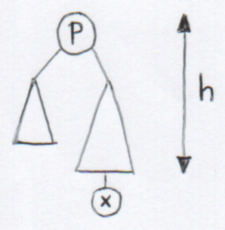
- O nó x foi inserido na subárvore direita de p.
- p possui o filho direito u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:

Caso 2: $h_E(p) < h_D(p)$



- O nó x foi inserido na subárvore direita de p.
- p possui o filho direito u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:
  - **Caso 2(a):** $h_E(u) < h_D(u)$.

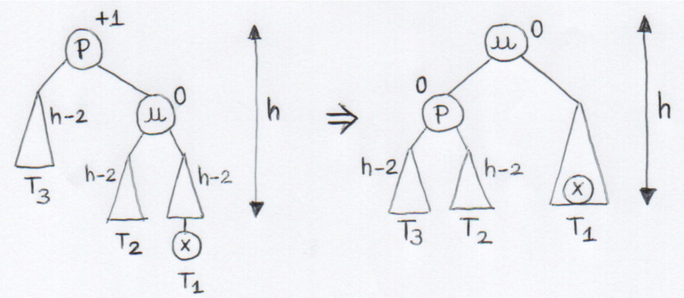
Caso 2: $h_E(p) < h_D(p)$



- O nó x foi inserido na subárvore direita de p.
- p possui o filho direito u, $u \neq x$. Pois caso contrário, p não estaria desregulado.
- Por esse mesmo motivo, sabe-se que $h_E(u) \neq h_D(u)$.
- Há dois subcasos a considerar:
  - **Caso 2(a):** $h_E(u) < h_D(u)$.
  - **Caso 2(b):** $h_E(u) > h_D(u)$.

Caso 2(a): $h_E(u) < h_D(u)$

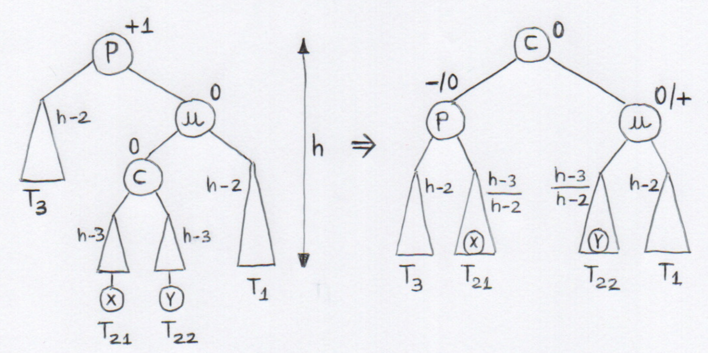
Solução: Rotação esquerda simples em p.



O nó x é inserido à direita de u.
Note que $h(T_1) = h(T_2) + 1$.
Após a rotação, a altura final permanece inalterada.
Nenhuma modificação futura é necessária.

Caso 2(b): $h_E(u) > h_D(u)$

Solução: Rotação dupla direita.



O nó inserido pode ser X ou Y.
Quando $h = 2$, as árvores T_1 e T_3 são vazias, e o nó inserido é o próprio nó C; neste caso as árvores T_{21} e T_{22} são vazias.
A altura final permanece inalterada.
Nenhuma modificação futura é necessária.

Propriedades das rotações

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Propriedade 1

As rotações preservam a natureza da árvore como sendo binária de busca. $\square$

Propriedades das rotações

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Propriedade 1

As rotações preservam a natureza da árvore como sendo binária de busca. $\square$

Propriedade 2

Uma vez que um nó p torna-se desregulado, a regulagem de p é restabelecida pela aplicação de uma das 4 rotações vistas. $\square$

Propriedades das rotações

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Propriedade 1

As rotações preservam a natureza da árvore como sendo binária de busca. $\square$

Propriedade 2

Uma vez que um nó p torna-se desregulado, a regulagem de p é restabelecida pela aplicação de uma das 4 rotações vistas. $\square$

Propriedade 3

Dado um nó p desregulado, uma rotação apropriada de p assegura a regulagem de TODOS os nós ancestrais de p. $\square$

Atividade

Mostre o passo-a-passo da inserção das chaves 1,2,3,4,5,6,7 em uma árvore AVL inicialmente vazia. Em cada passo, ilustre o valor do `fator de balanceamento` de cada nó, assim como as rotações realizadas.

Implementação da inserção

Como determinar o fator de balanço de um nó?

Como verificar se algum nó v de T se tornou desregulado após a inclusão?

- Basta calcular as alturas de suas subárvores e subtrair uma da outra.

Como determinar o fator de balanço de um nó?

Como verificar se algum nó v de T se tornou desregulado após a inclusão?

- Basta calcular as alturas de suas subárvores e subtrair uma da outra.
- Precisamos fazer isso mantendo o tempo de inserção em $O(\log n)$.
    - É possível?

Como determinar o fator de balanço de um nó?

Universidade Federal do Ceará
Campus Quixadá

Como verificar se algum nó v de T se tornou desregulado após a inclusão?

- Basta calcular as alturas de suas subárvores e subtrair uma da outra.
- Precisamos fazer isso mantendo o tempo de inserção em $O(\log n)$.
  - É possível?

Ideia: Cada nó v da árvore terá um campo adicional chamado `height` que guardará a altura da árvore enraizada em v.

- Assim, não será preciso percorrer a árvore enraizada em v.
- Poderemos calcular o fator de balanceamento do nó v em tempo constante $O(1)$.

Como a altura de cada nó é determinada?

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Cada nó da árvore possui o campo `height`, que guarda sua altura.
- Assim que um nó p é inserido na árvore ele é um nó folha.
  - A altura de p é igual a 1 logo após sua inserção.
  - Fazendo `p->height = 1` assim que o nó p é inserido, determinamos sua altura em tempo $O(1)$.

Como a altura de cada nó é determinada?

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Cada nó da árvore possui o campo `height`, que guarda sua altura.
- Assim que um nó p é inserido na árvore ele é um nó folha.
  - A altura de p é igual a 1 logo após sua inserção.
  - Fazendo `p->height = 1` assim que o nó p é inserido, determinamos sua altura em tempo $O(1)$.
- Observação: A partir deste momento, os únicos nós da árvore que podem ter alturas modificadas são os nós no caminho de p até a raiz.

Como a altura de cada nó é determinada?

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Cada nó da árvore possui o campo `height`, que guarda sua altura.
- Assim que um nó p é inserido na árvore ele é um nó folha.
    - A altura de p é igual a 1 logo após sua inserção.
    - Fazendo `p->height = 1` assim que o nó p é inserido, determinamos sua altura em tempo $O(1)$.
- Observação: A partir deste momento, os únicos nós da árvore que podem ter alturas modificadas são os nós no caminho de p até a raiz.
    - Todos eles devem ser verificados e ter seus campos `height` corretamente atualizados.

Como a altura de cada nó é determinada?

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Cada nó da árvore possui o campo `height`, que guarda sua altura.
- Assim que um nó p é inserido na árvore ele é um nó folha.
  - A altura de p é igual a 1 logo após sua inserção.
  - Fazendo `p->height = 1` assim que o nó p é inserido, determinamos sua altura em tempo $O(1)$.
- Observação: A partir deste momento, os únicos nós da árvore que podem ter alturas modificadas são os nós no caminho de p até a raiz.
  - Todos eles devem ser verificados e ter seus campos `height` corretamente atualizados.
  - Existem $O(\log n)$ destes nós e essa atualização pode ser feita em tempo constante a medida que as chamadas recursivas "se desenrolam".

Arquivo `node.h`

Arquivo node.h

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
#ifndef NODE_H
#define NODE_H

struct Node {
    // atributos
    int key;
    int height;
    Node *left;
    Node *right;

    // Construtor
    Node (int key, int height, Node *left, Node *right)
        : key(key), height(height), left(left), right(right)
    {
    }
};

#endif
```

Arquivo `avl.h` (com código inicial)

Arquivo avl.h (com código inicial)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```cpp
#ifndef AVL_H
#define AVL_H
#include "node.h"

class avl_tree {
public:
    avl_tree() = default;
    avl_tree(const avl_tree& t) = delete;
    avl_tree& operator=(const avl_tree& t) = delete;
    void add(int key);
    ~avl_tree();

private:
    Node *root {nullptr};
    int height(Node *node);
    int balance(Node *node);
    Node* rightRotation(Node *p);
    Node* leftRotation(Node *p);
    Node* add(Node *p, int key);
    Node* fixup_node(Node *p, int key);
};

#endif
```

Determinando fator de balanceamento de um nó

Calculamos o fator de balanceamento de um nó v através da subtração das alturas das subárvores esquerda e direita do nó:

Determinando fator de balanceamento de um nó

Calculamos o fator de balanceamento de um nó v através da subtração das alturas das subárvores esquerda e direita do nó:

```
int avl_tree::height(Node *node) {
    return (node == nullptr) ? 0 : node->height;
}
```

Determinando fator de balanceamento de um nó

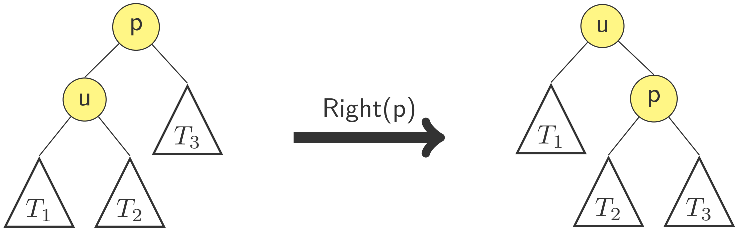
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Calculamos o fator de balanceamento de um nó v através da subtração das alturas das subárvores esquerda e direita do nó:

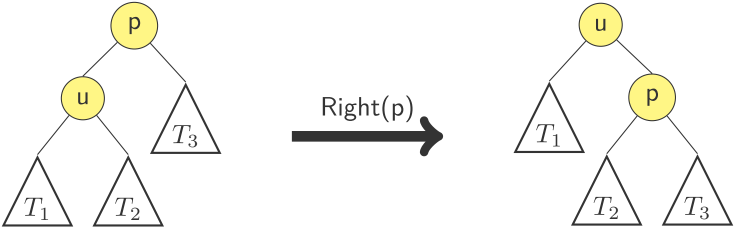
```
int avl_tree::height(Node *node) {
    return (node == nullptr) ? 0 : node->height;
}
```

```
int avl_tree::balance(Node *node) {
    return height(node->right) - height(node->left);
}
```

Rotação Direita

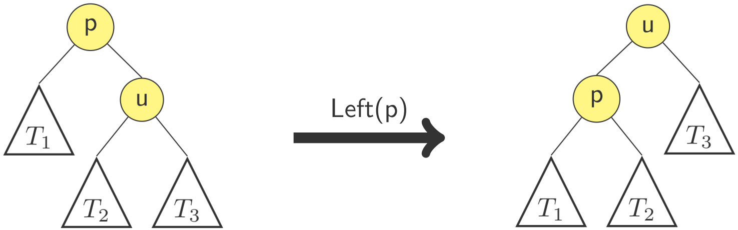


Rotação Direita

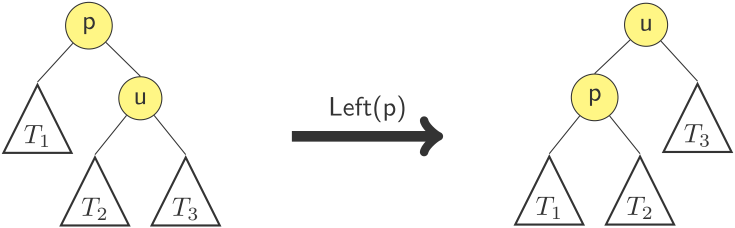



```
Node* avl_tree::rightRotation(Node *p) {
    Node *u = p->left;
    p->left = u->right;
    u->right = p;
    // atualiza altura dos nodes
    p->height = 1 + max(height(p->left),height(p->right));
    u->height = 1 + max(height(u->left),height(u->right));
    return u; // nova raiz
}
```

Rotação Esquerda



Rotação Esquerda




```
Node* avl_tree::leftRotation(Node *p) {
    Node *u = p->right;
    p->right = u->left;
    u->left = p;
    // atualiza altura dos nodes
    p->height = 1 + max(height(p->left),height(p->right));
    u->height = 1 + max(height(u->left),height(u->right));
    return u; // nova raiz
}
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Função pública:

```
void avl_tree::add(int key) {
    root = add(root, key);
}
```

Inserção

Função privada:

```
Node* avl_tree::add(Node *p, int key) {
    if(p == nullptr) // subarvore vazia
        return new Node(key, 1, nullptr, nullptr);
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Função privada:

```
Node* avl_tree::add(Node *p, int key) {
    if(p == nullptr) // subarvore vazia
        return new Node(key, 1, nullptr, nullptr);
    if(key == p->key) // chave repetida
        return p;
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Função privada:

```
Node* avl_tree::add(Node *p, int key) {
    if(p == nullptr) // subarvore vazia
        return new Node(key, 1, nullptr, nullptr);
    if(key == p->key) // chave repetida
        return p;
    if(key < p->key)
        p->left = add(p->left, key);
    else
        p->right = add(p->right, key);
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Função privada:

```
Node* avl_tree::add(Node *p, int key) {
    if(p == nullptr) // subarvore vazia
        return new Node(key, 1, nullptr, nullptr);
    if(key == p->key) // chave repetida
        return p;
    if(key < p->key)
        p->left = add(p->left, key);
    else
        p->right = add(p->right, key);

    p = fixup_node(p, key); // regula o node p

    return p;
}
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
    // Caso 1(a): rotacao direita
    if(bal < -1 && key < p->left->key)
        return rightRotation(p);
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
    // Caso 1(a): rotacao direita
    if(bal < -1 && key < p->left->key)
        return rightRotation(p);
    // Caso 1(b): rotacao dupla direita
    else if(bal < -1 && key > p->left->key) {
        p->left = leftRotation(p->left);
        return rightRotation(p);
    }
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
    // Caso 1(a): rotacao direita
    if(bal < -1 && key < p->left->key)
        return rightRotation(p);
    // Caso 1(b): rotacao dupla direita
    else if(bal < -1 && key > p->left->key) {
        p->left = leftRotation(p->left);
        return rightRotation(p);
    }
    // Caso 2(a): rotacao esquerda
    else if(bal > 1 && key > p->right->key)
        return leftRotation(p);
```

Inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
    // Caso 1(a): rotacao direita
    if(bal < -1 && key < p->left->key)
        return rightRotation(p);
    // Caso 1(b): rotacao dupla direita
    else if(bal < -1 && key > p->left->key) {
        p->left = leftRotation(p->left);
        return rightRotation(p);
    }
    // Caso 2(a): rotacao esquerda
    else if(bal > 1 && key > p->right->key)
        return leftRotation(p);
    // Caso 2(b): rotacao dupla esquerda
    else if(bal > 1 && key < p->right->key) {
        p->right = rightRotation(p->right);
        return leftRotation(p);
    }
```

Inserção


```
Node* avl_tree::fixup_node(Node *p, int key) {
    // atualiza altura deste node ancestral p
    p->height = 1 + max(height(p->left),height(p->right));
    // obtem balanco de p
    int bal = balance(p);
    // Caso 1(a): rotacao direita
    if(bal < -1 && key < p->left->key)
        return rightRotation(p);
    // Caso 1(b): rotacao dupla direita
    else if(bal < -1 && key > p->left->key) {
        p->left = leftRotation(p->left);
        return rightRotation(p);
    }
    // Caso 2(a): rotacao esquerda
    else if(bal > 1 && key > p->right->key)
        return leftRotation(p);
    // Caso 2(b): rotacao dupla esquerda
    else if(bal > 1 && key < p->right->key) {
        p->right = rightRotation(p->right);
        return leftRotation(p);
    }
    return p;
}
```

Remoção

Remoção em árvores AVL

- A remoção também pode ser feita em $O(\log n)$.
- Após a exclusão da chave, verificamos se a árvore se tornou desregulada.
- Assim como na inserção, os nós a serem examinados pertencem ao caminho da raiz até uma de suas folhas.
- Ao contrário da inserção, agora o número de rotações necessárias para a regulagem da árvore pode atingir $O(\log n)$.

Algoritmo de remoção em árvores AVL

1. Fazemos uma busca pelo nó a ser removido.

2. Se o nó encontrado for nulo, então a árvore é vazia ou a chave não existe na árvore. Não há o que remover neste caso.

3. Caso contrário, uma vez encontrado o nó x com a chave desejada, tratamos de removê-lo da árvore. Há somente dois casos a considerar:
   (a) Se o nó x não tiver filho direito, então o seu filho esquerdo (seja ele vazio ou não) assume o papel de x e o nó x é liberado.
      - Todos os ancestrais de x devem ter suas alturas atualizadas e devem ser regulados, caso necessário, por meio de rotação apropriada.

   (b) Se o nó x tiver filho direito, trocamos a chave de x com a chave do seu nó sucessor e o nó sucessor é liberado.
      - Todos os nós que estiverem no caminho do antigo pai do sucessor de x até a raiz da árvore devem ter suas alturas atualizadas e devem ser regulados, caso necessário, por meio de rotação apropriada.

Análide do balanceamento na remoção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Os únicos nós que podem ter se tornado desregulados após a remoção de x ou de seu sucessor são todos os ancestrais do nó removido.

- Analisaremos os casos que podem influenciar no fator de balanceamento de um nó p quando um nó x é removido do lado esquerdo de p.

Análide do balanceamento na remoção

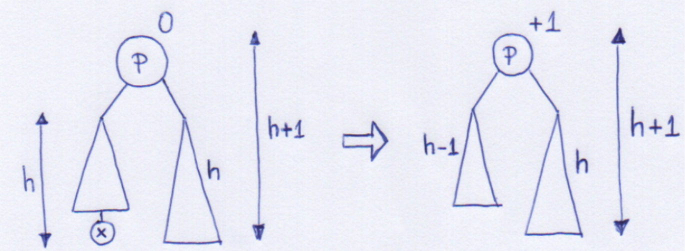
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Os únicos nós que podem ter se tornado desregulados após a remoção de x ou de seu sucessor são todos os ancestrais do nó removido.
- Analisaremos os casos que podem influenciar no fator de balanceamento de um nó p quando um nó x é removido do lado esquerdo de p.

Atenção: Os casos em que o nó x é removido do lado direito de p são simétricos aos apresentados nestes slides, e sua análise e exame será deixada como exercício para casa.

Análide do balanceamento na remoção

Caso 1: As alturas das subárvores do nó p eram iguais.

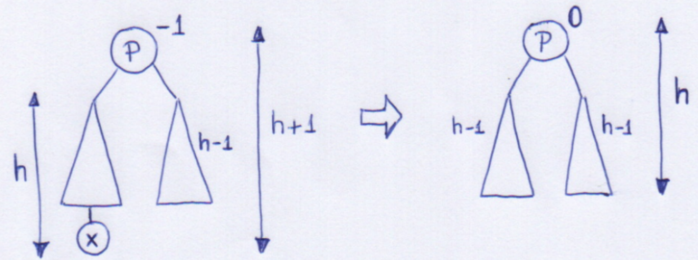


A altura permanece inalterada.

Nenhuma regulagem é necessária aqui, e nem será futuramente.

Análide do balanceamento na remoção

Caso 2: Remoção da subárvore mais alta de p.

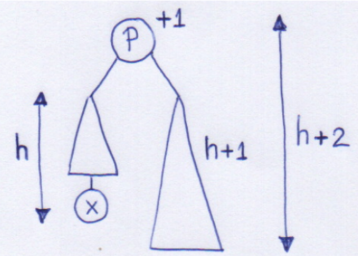


A altura diminui.
Nenhuma regulagem é necessária aqui. Porém, algum ancestral de p pode ter se tornado desregulado.

Análide do balanceamento na remoção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

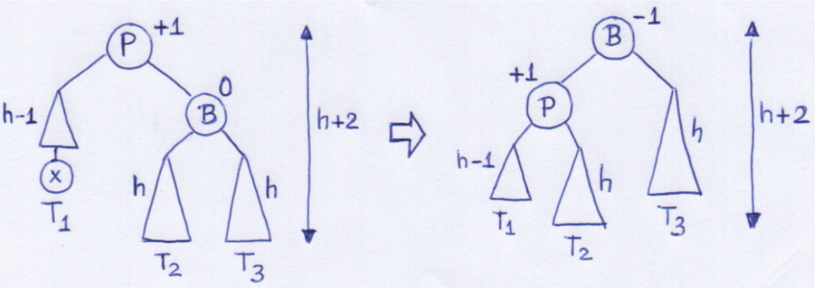
Caso 3: Remoção da subárvore mais baixa de p.



Há três subcasos a considerar, dependendo do fator de balanceamento do filho direito da raiz p, que pode ser 0, $+1$ ou -1.

Análide do balanceamento na remoção

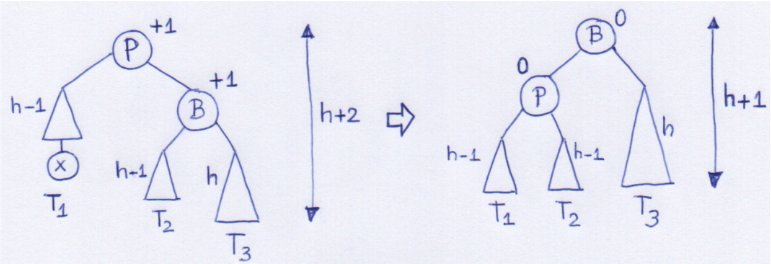
Caso 3(a): Filho direito de p tem balanço $= 0$.
Solução: Rotação esquerda.



A altura permanece inalterada.
Nenhuma regulagem acontecerá mais.

Análide do balanceamento na remoção

Caso 3(b): Filho direito de p tem balanço $= +1$.
Solução: Rotação esquerda.

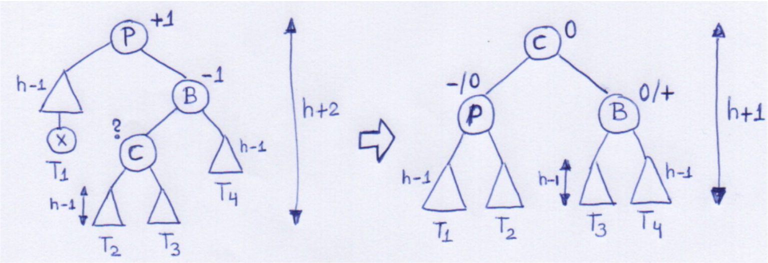


A altura diminui (algum ancestral pode ter se tornado desregulado).

Análide do balanceamento na remoção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

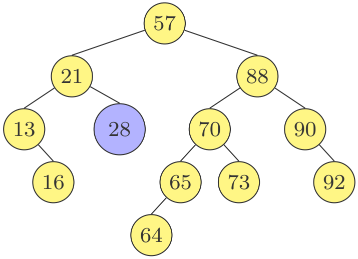
Caso 3(c): Filho direito de p tem balanço $= -1$.
Solução: Rotação esquerda dupla.



A altura diminui (algum ancestral pode ter se tornado desregulado).

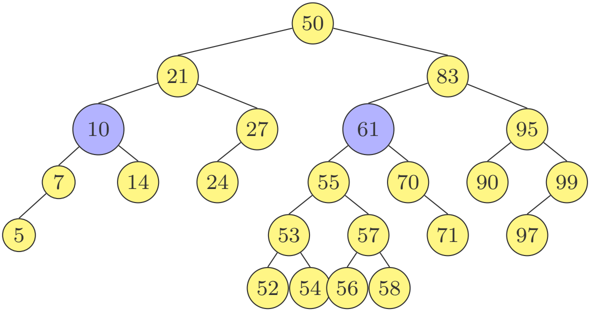
Remoção — Exercício 1

Excluir o nó 28 e fazer as regulagens para manter a árvore AVL.



Remoção — Exercício 2

Excluir a chave 61 e, depois, a chave 10 da árvore abaixo e fazer as regulagens necessárias.



Remoção — Função Pública

```
void avl_tree::remove(int key) {
    root = remove(root, key);
}
```

Remoção – Função privada que busca o nó


```
Node* avl_tree::remove(Node *node, int key) {
    if(node == nullptr) // node nao encontrado
        return nullptr;
```

Remoção – Função privada que busca o nó

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::remove(Node *node, int key) {
    if(node == nullptr) // node nao encontrado
        return nullptr;
    if(key < node->key)
        node->left = remove(node->left, key);
    else if(key > node->key)
        node->right = remove(node->right, key);
```

Remoção – Função privada que busca o nó

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::remove(Node *node, int key) {
    if(node == nullptr) // node nao encontrado
        return nullptr;
    if(key < node->key)
        node->left = remove(node->left, key);
    else if(key > node->key)
        node->right = remove(node->right, key);
    // encontramos no node
    else if(node->right == nullptr) { // sem filho direito
        Node *child = node->left;
        delete node;
        return child;
    }
    else // tem filho direito: troca pelo sucessor
        node->right = remove_successor(node, node->right);
```

Remoção – Função privada que busca o nó

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
Node* avl_tree::remove(Node *node, int key) {
    if(node == nullptr) // node nao encontrado
        return nullptr;
    if(key < node->key)
        node->left = remove(node->left, key);
    else if(key > node->key)
        node->right = remove(node->right, key);
    // encontramos no node
    else if(node->right == nullptr) { // sem filho direito
        Node *child = node->left;
        delete node;
        return child;
    }
    else // tem filho direito: troca pelo sucessor
        node->right = remove_successor(node, node->right);

    // Atualiza a altura do node e regula o node
    node = fixup_deletion(node);
    return node;
}
```

Remoção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Função privada que remove o sucessor

```
Node* avl_tree::remove_successor(Node *root, Node *node) {
    if(node->left != nullptr)
        node->left = remove_successor(root, node->left);
    else {
        root->key = node->key;
        Node *aux = node->right;
        delete node;
        return aux;
    }
    // Atualiza a altura do node e regula o node
    node = fixup_deletion(node);
    return node;
}
```

Remoção — Função privada de rebalanceamento

```cpp
Node* avl_tree::fixup_deletion(Node *node) {
    node->height =
        1 + max(height(node->left),height(node->right));

    int bal = balance(node);

    // node pode estar desregulado, ha 4 casos a considerar
    if(bal > 1 && balance(node->right) >= 0) {
        return leftRotation(node);
    }
    else if(bal > 1 && balance(node->right) < 0) {
        node->right = rightRotation(node->right);
        return leftRotation(node);
    }
    else if(bal < -1 && balance(node->left) <= 0) {
        return rightRotation(node);
    }
    else if(bal < -1 && balance(node->left) > 0) {
        node->left = leftRotation(node->left);
        return rightRotation(node);
    }
    return node;
}
```

FIM