Estruturas de Dados II

Árvores AVL

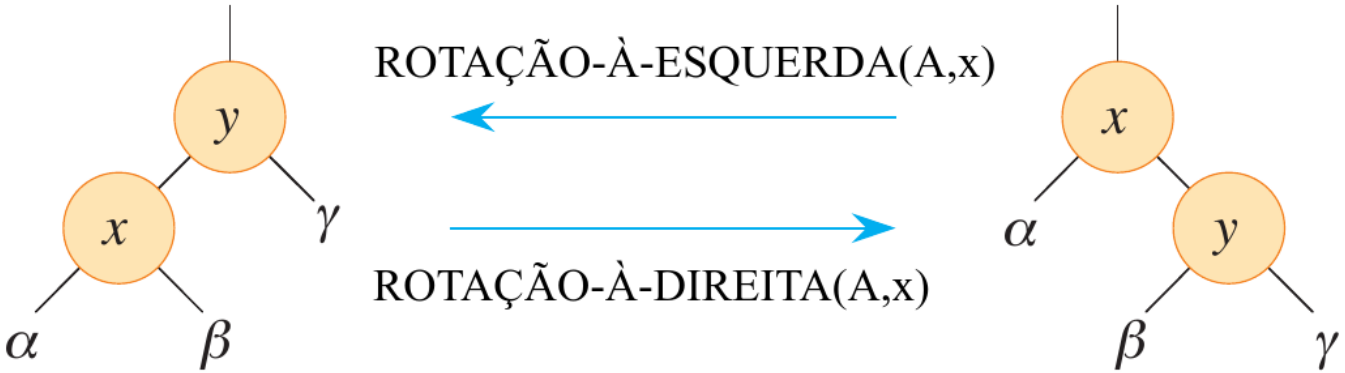
Prof. Bruno Azevedo

Instituto Federal de São Paulo

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus Catanduva

Revisão: Rotações

- O balanceamento em uma árvore AVL é feito através de rotações.
- Uma rotação é uma operação que preserva a propriedade da Árvore Binária de Busca.
- A figura abaixo mostra os dois tipos de rotações utilizados: rotações à esquerda e rotações à direita.

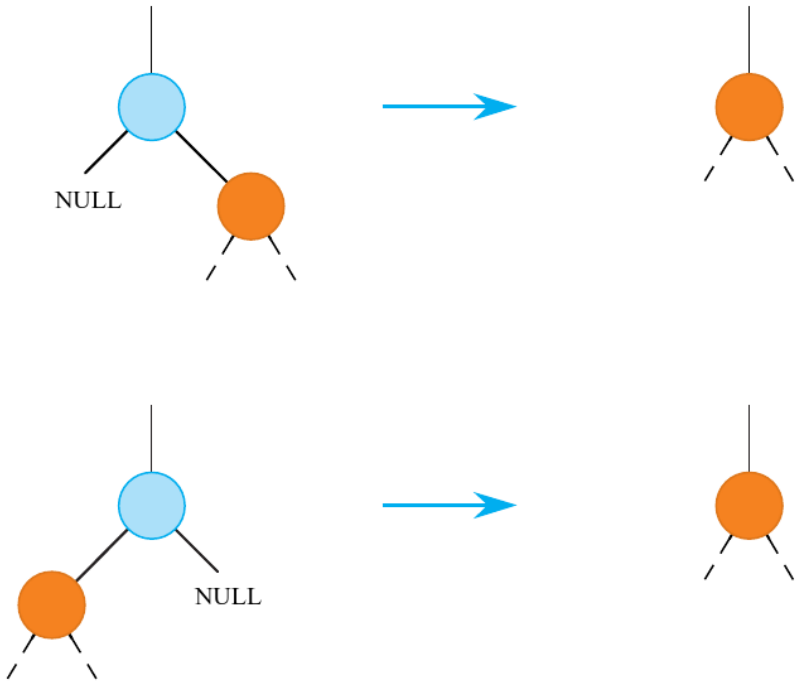


Revisão: Deleção em Árvores Binárias de Busca

- A **deleção** de um nó em uma Árvore AVL é idêntica à deleção da Árvore Binária de Busca básica.
- Vamos relembrar.

Revisão: Deleção em Árvores Binárias de Busca

- Caso 1: Se o nó não possuir filhos, basta deletar o nó.
- Caso 2: Se o nó possuir apenas um filho, este tomará o lugar do nó deletado.

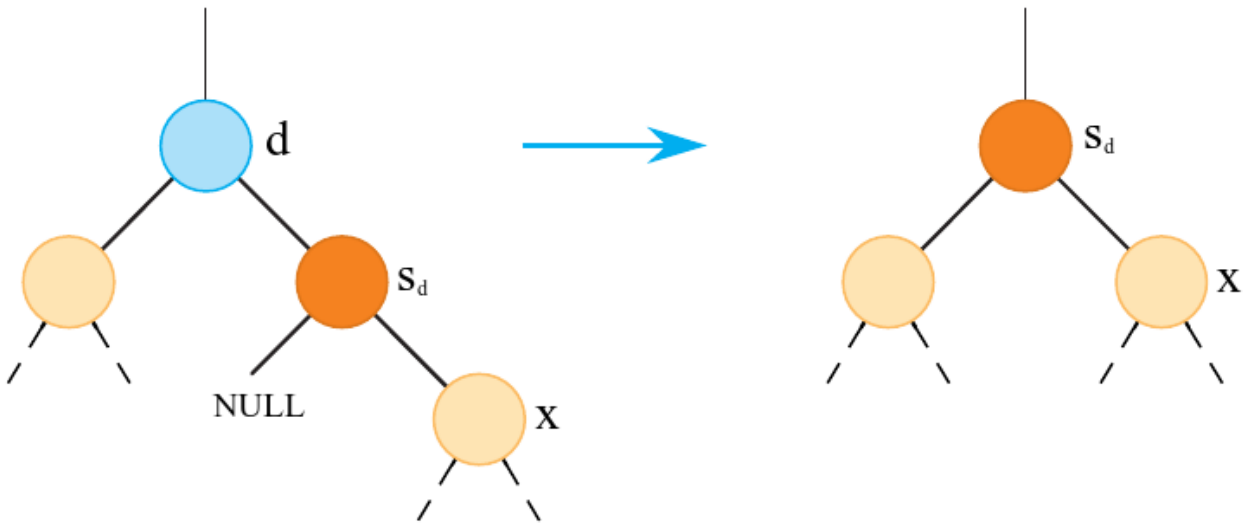


Revisão: Deleção em Árvores Binárias de Busca

- Caso 3: Se o nó possuir dois filhos, encontraremos o sucessor do nó a ser deletado, que chamaremos de $\mathtt{d}$.
- Encontraremos o sucessor de $\mathtt{d}$, que chamaremos de $\mathtt{S_d}$, que está na subárvore direita de $\mathtt{d}$. Este não possuirá filho à esquerda.
- Remova o nó $\mathtt{S_d}$ de sua posição atual e substitua $\mathtt{d}$ por $\mathtt{S_d}$ na árvore.
- Como fazer isso dependerá se $\mathtt{S_d}$ for o filho direito de $\mathtt{d}$ ou não.

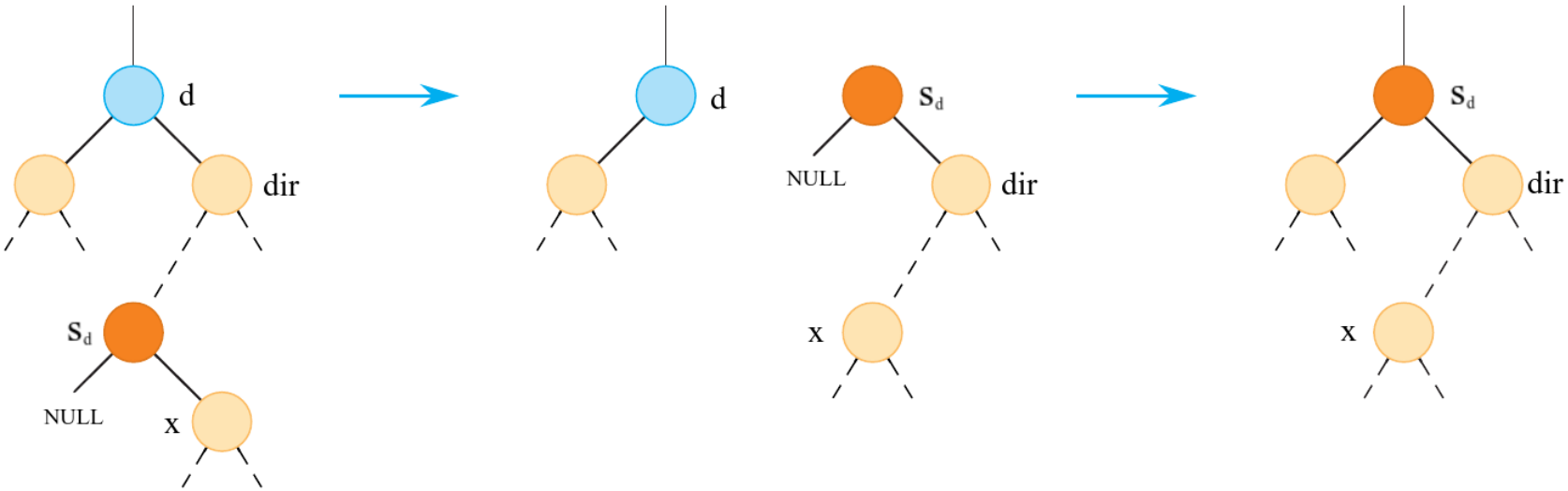
Revisão: Deleção em Árvores Binárias de Busca

- Caso 3.1: Se S_d for o filho direito de d, então, como na ilustrado figura abaixo, substitua d por S_d, deixando o filho direito de S_d inalterado.



Revisão: Deleção em Árvores Binárias de Busca

- Caso 3.2: Caso contrário, S_d está na subárvore direita de d, mas não é o filho direito de d. Nesse caso, como ilustrado na figura abaixo, primeiro substitua S_d por seu próprio filho direito, e então substitua d por S_d.



Deleção em Árvores AVL

- Entretanto, ao contrário da Árvore Binária de Busca básica, precisamos verificar se a deleção não tornou a Árvore AVL desbalanceada.
- Para identificarmos desbalanceamentos, e decidir qual ação tomar, é necessário considerar os fatores de balanceamento dos nós envolvidos.

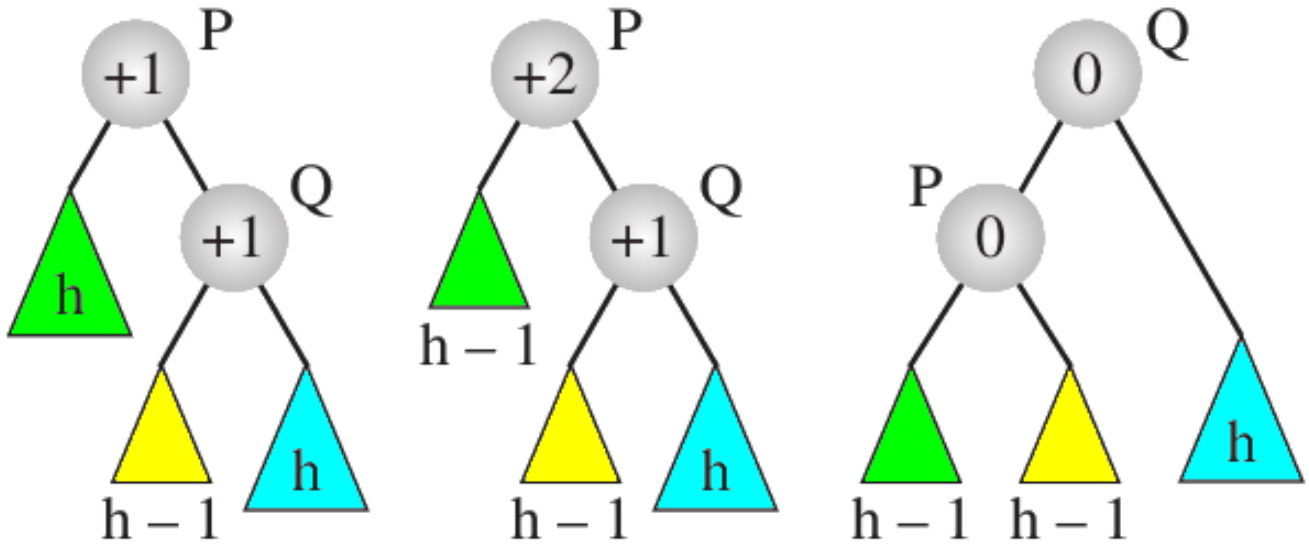
Deleção em Árvores AVL

- Será preciso tratar 10 casos gerais, mas apenas cinco distintos, devido a simetria entre os casos.
- Iremos analisar os cinco casos de deleção em uma subárvore esquerda. Os outros cinco casos são simétricos e correspondem à deleção em uma subárvore direita.
- Entretanto, observem como teremos apenas duas soluções distintas para os cinco casos. Estaremos analisando cada caso para acompanhar como ficam os fatores de balanceamento após a sua solução.

Deleção em Árvores AVL

- Caso 1.1: Deletamos um nó da subárvore esquerda do nó P.
- Este nó possui um fator de balanceamento de $+1$.
- Portanto, a deleção de um nó da sua subárvore esquerda resultará em P aumentar seu fator de balanceamento para $+2$.

$\Rightarrow$ Sua árvore direita possui um fator de balanceamento de $+1$.

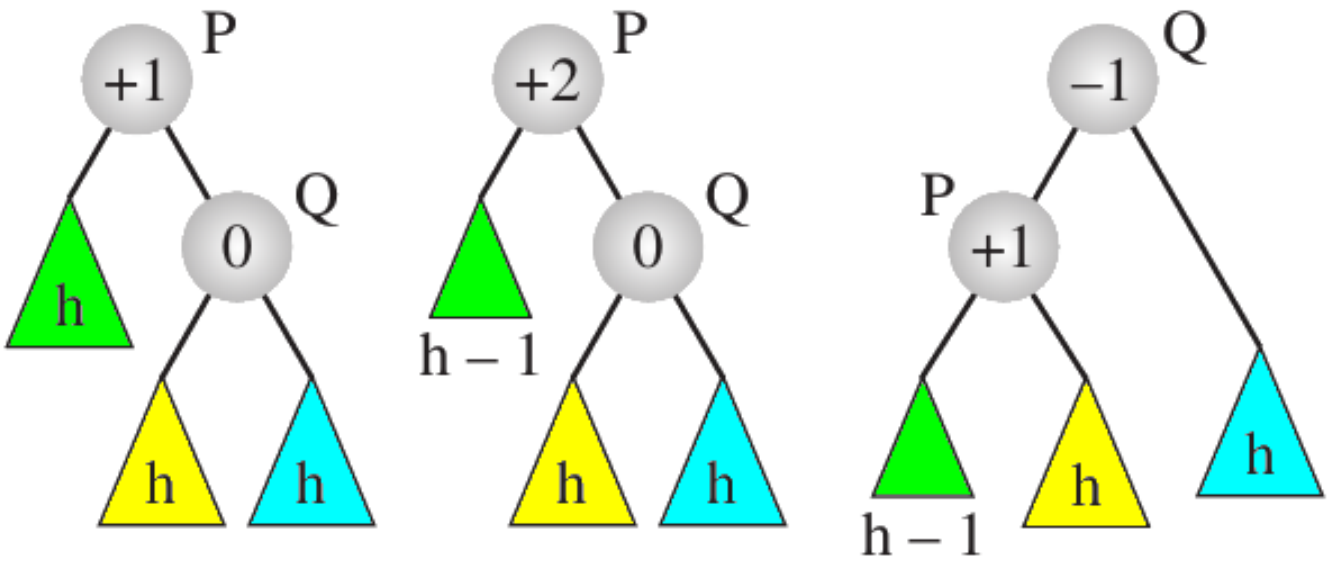


- Este desbalanceamento é solucionado através de uma rotação para à esquerda.

Deleção em Árvores AVL

- Caso 1.2: Deletamos um nó da subárvore esquerda do nó P.
- Este nó possui um fator de balanceamento de $+1$.

$\Rightarrow$ Sua árvore direita possui um fator de balanceamento de 0.



- Como no Caso 1, este desbalanceamento é solucionado através de uma rotação para à esquerda.

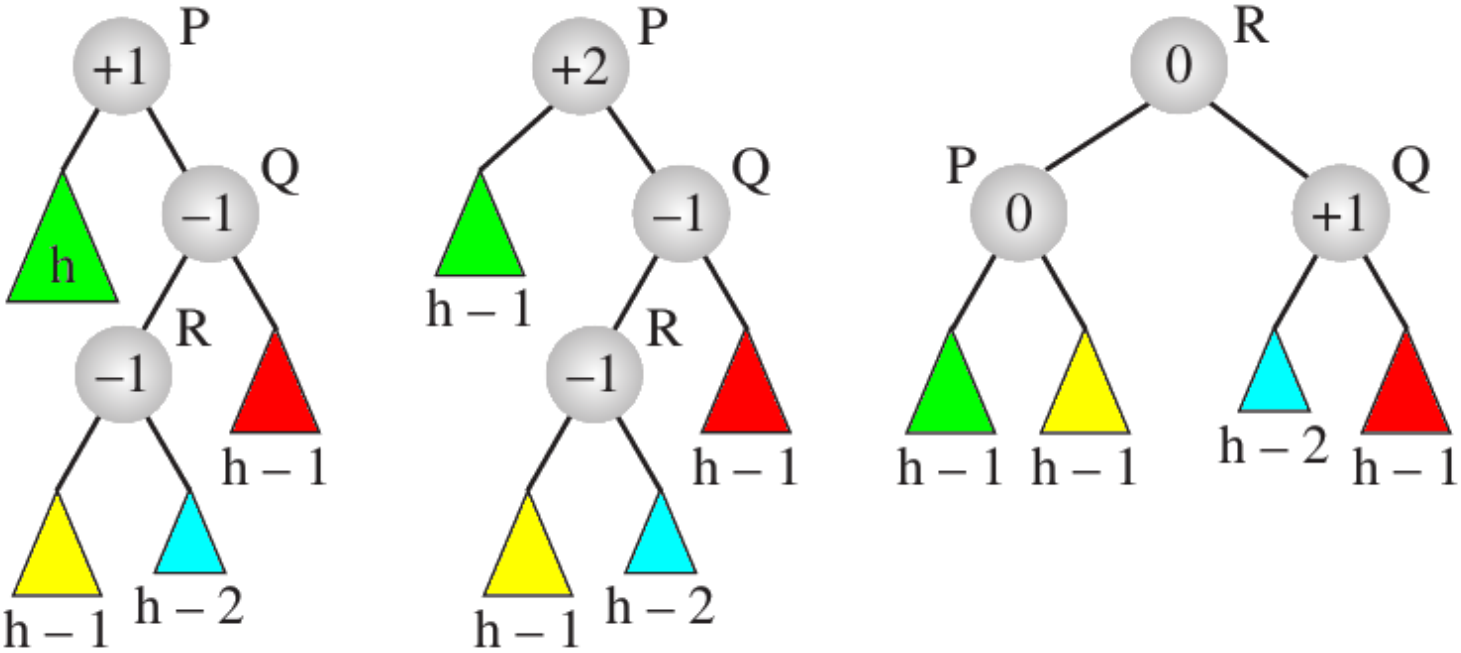
Deleção em Árvores AVL

- Esses dois primeiros casos podem ser tratados juntos na implementação após identificar o fator de balanceamento do nó Q.
- Se o fator de balanceamento de Q for -1, temos outros três casos.
- Vamos analisar estes três casos.

Deleção em Árvores AVL

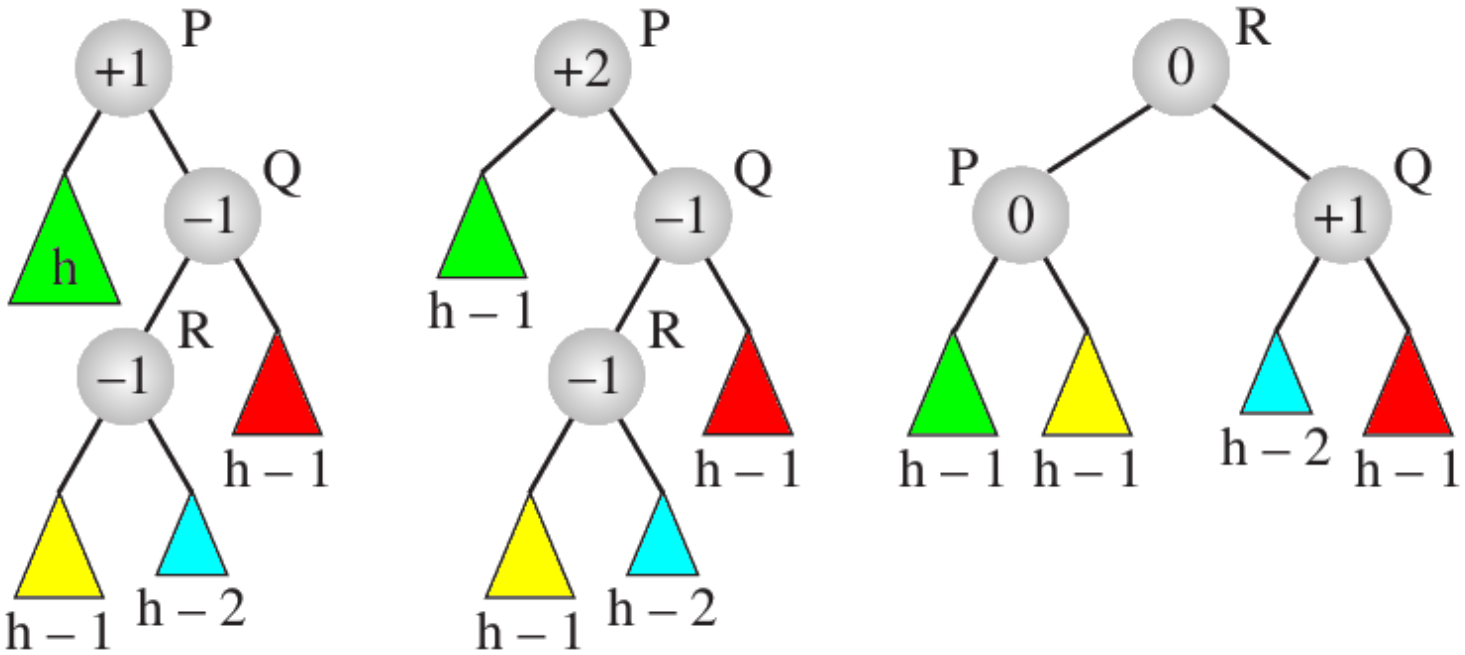
- Caso 2.1: Deletamos um nó da subárvore esquerda do nó P.
- Este nó possui um fator de balanceamento de $+1$.
- Sua subárvore direita (raiz em Q) possui um fator de balanceamento de -1.

$\Rightarrow$ A subárvore esquerda de Q (raiz em R) possui um fator de balanceamento de -1.



Deleção em Árvores AVL

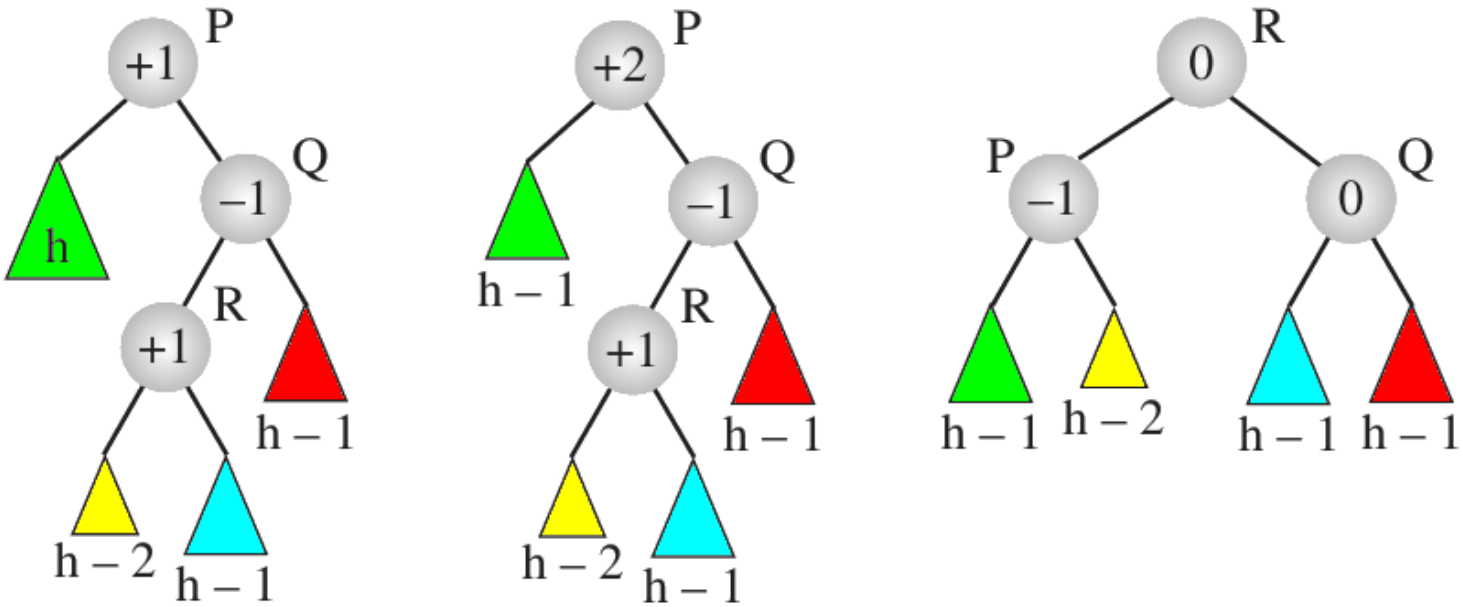
- Para solucionar este desbalanceamento, rotacionamos Q para a direita e, em seguida, rotacionamos P para a esquerda.



Deleção em Árvores AVL

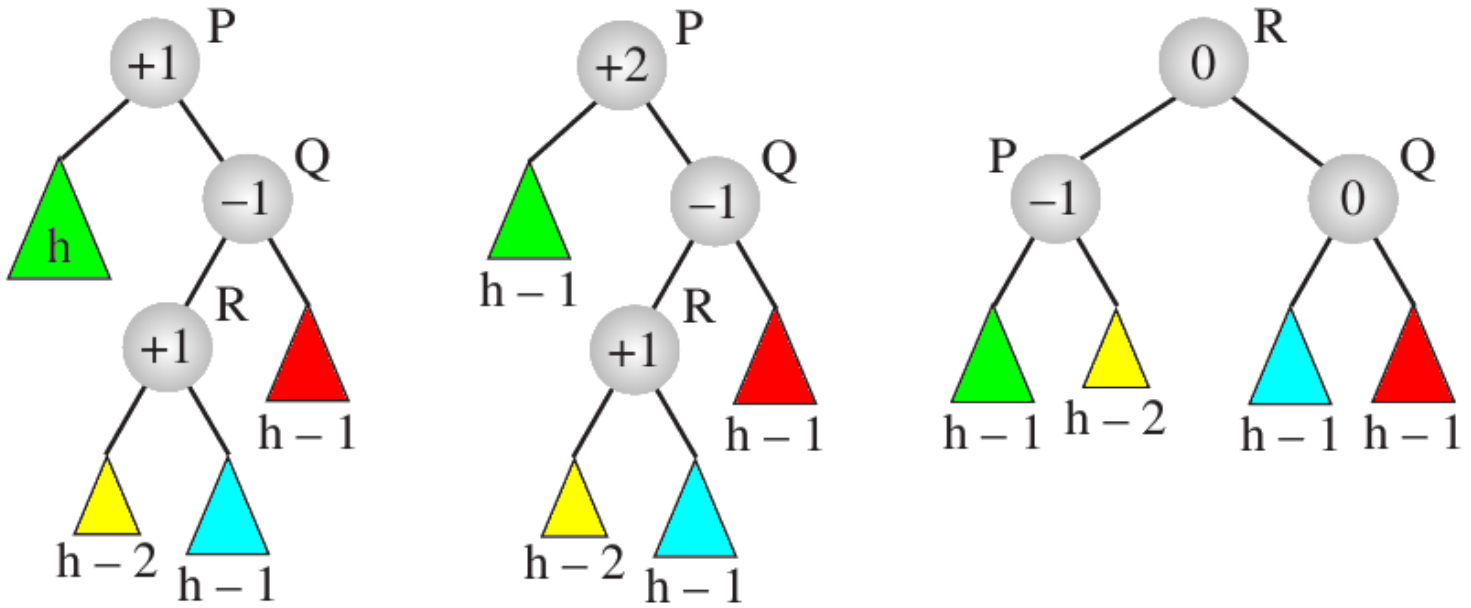
- Caso 2.2: Deletamos um nó da subárvore esquerda do nó P.
- Este nó possui um fator de balanceamento de $+1$.
- Sua subárvore direita (raiz em Q) possui um fator de balanceamento de -1.

$\Rightarrow$ A subárvore esquerda de Q (raiz em R) possui um fator de balanceamento de $+1$.



Deleção em Árvores AVL

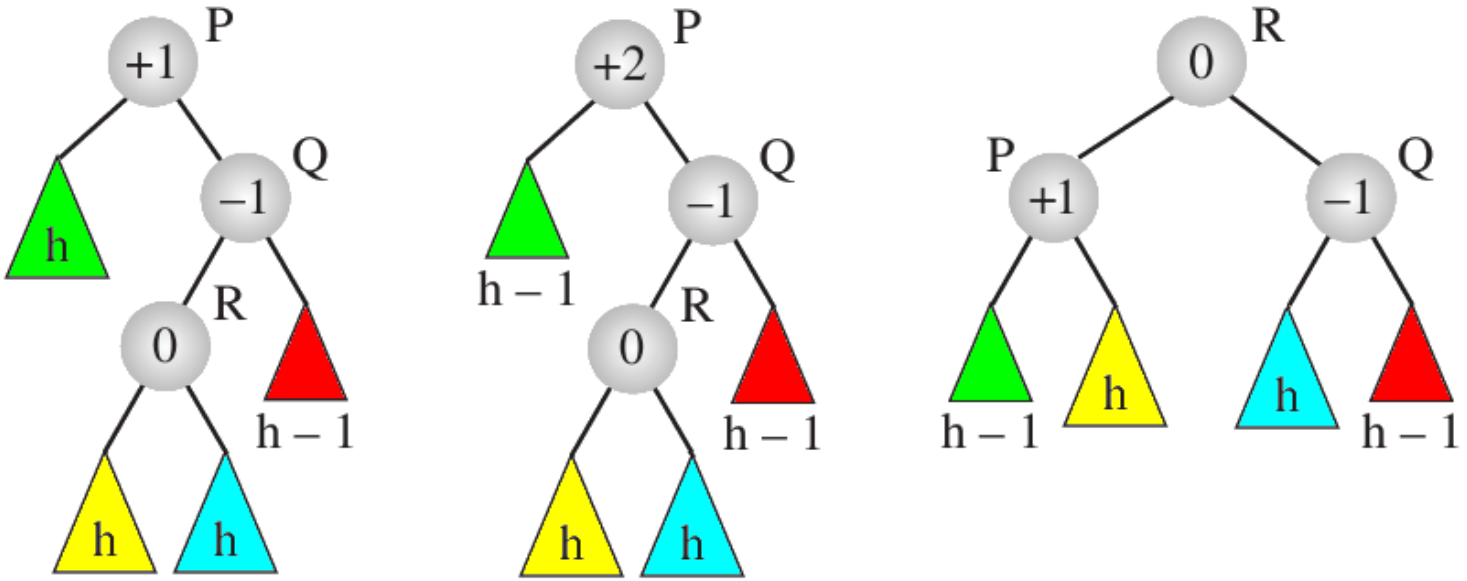
- Como no caso anterior, para solucionar este desbalanceamento, basta rotacionarmos Q para a direita e, em seguida, rotacionar P para a esquerda.



Deleção em Árvores AVL

- Caso 2.3: Deletamos um nó da subárvore esquerda do nó P.
- Este nó possui um fator de balanceamento de $+1$.
- Sua subárvore direita (raiz em Q) possui um fator de balanceamento de -1.

$\Rightarrow$ A subárvore esquerda de Q (raiz em R) possui um fator de balanceamento de 0.



- Para solucionar este desbalanceamento, é necessário rotacionar Q para a direita e, em seguida, rotacionar P para a esquerda.

Deleção em Árvores AVL

- Esses três casos (2.1, 2.2 e 2.3) podem ser tratados juntos na implementação após identificar o fator de balanceamento do nó R.
- Os outros cinco casos são simétricos e se referem a deleção de um nó na subárvore direita do nó P.
- Portanto, podem ser tratados de modo análogo.

Deleção em Árvores AVL

- Na inserção de um nó em uma Árvore AVL, se o balanceamento de P é perturbado e depois restaurado, não é preciso fazer ajustes nos predecessores de P.
- **Isso não é válido para a deleção de um nó.**
- É necessário propagar o ajuste de balanceamento para os predecessores do nó que foi deletado.
- Isso ocorre porque, após a deleção de um nó, o balanceamento da árvore pode ser afetado em diferentes níveis, e os fatores de balanceamento precisam ser recalculados para garantir que a árvore continue balanceada.

Exercícios

1. Qual é a complexidade computacional, no pior caso, do tempo para deletar um nó em uma Árvore AVL? Em que situação ocorre o pior caso?
2. Descrevam os cinco casos simétricos não analisados nos slides anteriores. Descrevam suas soluções.
3. Implemente em C/C++ a operação de deleção balanceada da Árvore AVL. Adicione esse código aos códigos da aula anterior. Dica: utilize uma abordagem recursiva.