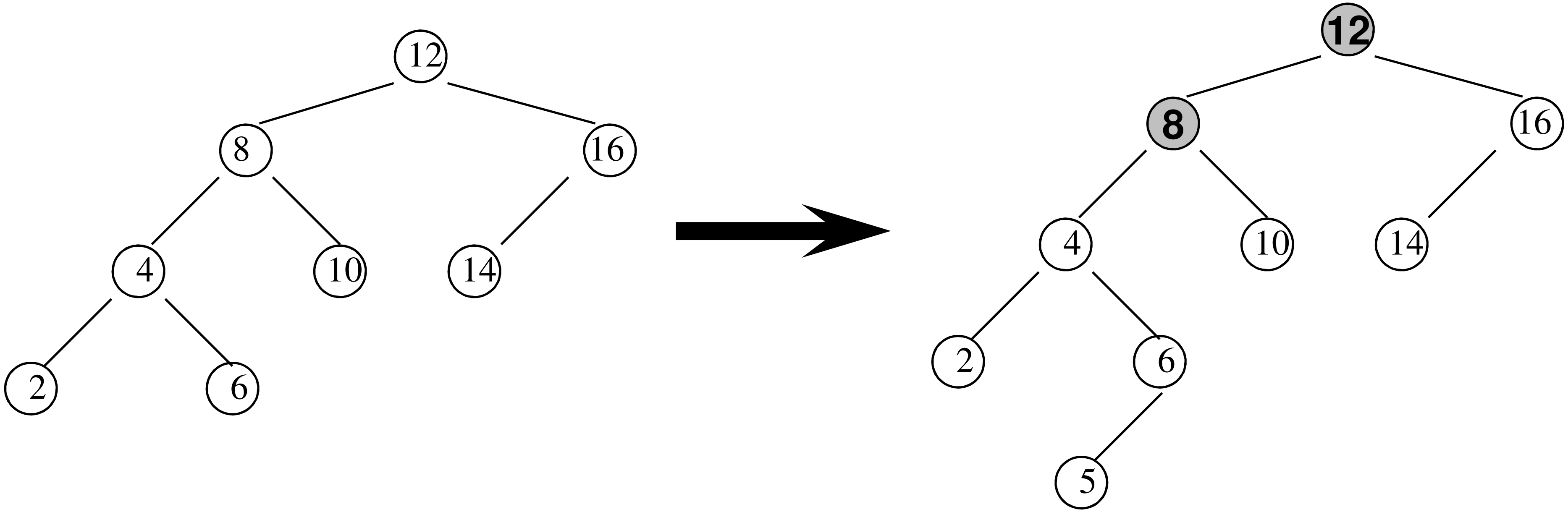
Árvores balanceadas

- Uma árvore binária de busca não garante acesso em tempo logarítmico.
  - Inserções ou eliminações podem desbalanceá-la.
  - Pior caso: a árvore degenera em lista ligada, onde a busca passa a gastar tempo linear.
- Balanceamento das árvores binárias de busca:
  - Evitam esses casos degenerados.
  - Garantem tempo logarítmico para todas as operações.
  - Requerem algoritmos mais elaborados para inserção e eliminação.
  - De modo geral, os nós das árvores balanceadas armazenam mais informações.
- Dois conhecidos modelos: árvores AVL e vermelho-preto.

Árvores AVL

- Autores: Adelson-Velskii e Landis (1962)
- Exigências para as sub-árvores de cada nó:
  - Diferença de alturas não pode exceder 1
  - É simples de manter
  - Garante altura logarítmica para a árvore
- <u>Definição</u>: uma árvore AVL é uma árvore binária de busca em cujos nós as alturas das sub-árvores diferem no máximo de uma unidade.

ÁRVORES AVL



Inserção do 5

Desbalanceamento

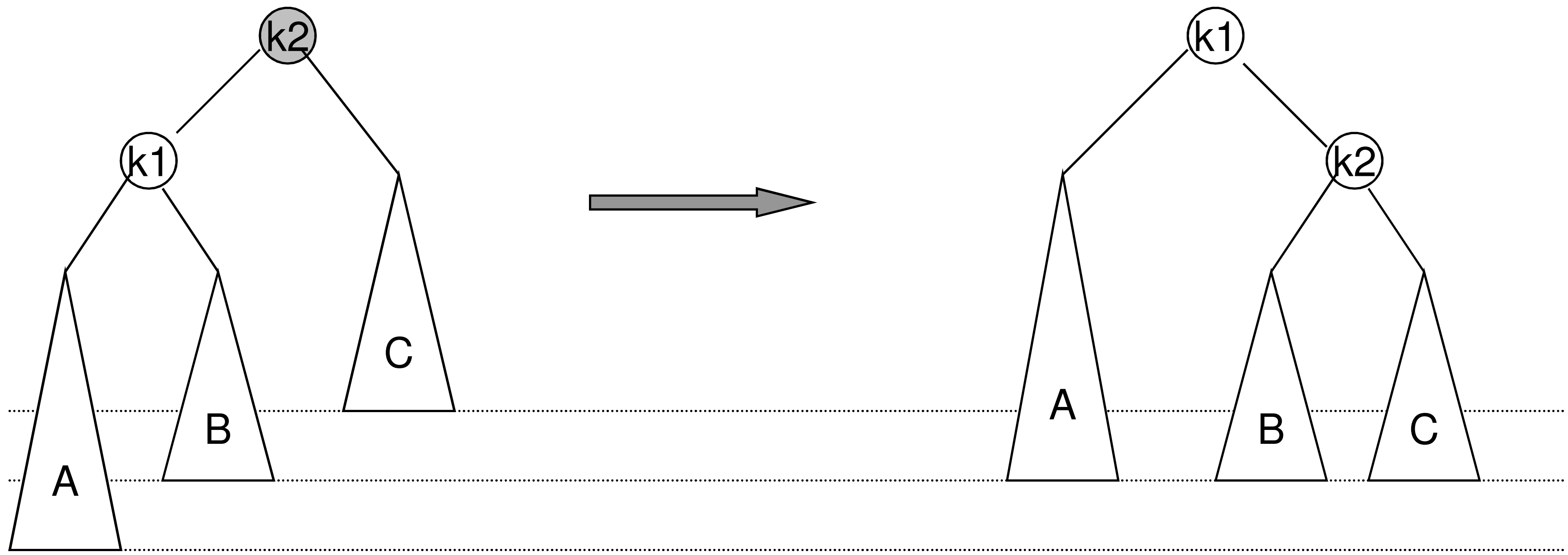
Após cada inserção ou eliminação, é necessário verificar o balanceamento de todos os nós da árvore

INSERÇÃO EM ÁRVORES AVL

- Após uma inserção, somente podem ficar desbalanceados os nós do caminho da raiz até esse novo nó.
- Nesse caminho, é preciso encontrar o nó mais profundo no qual ocorreu desbalanceamento.
  - Basta rebalancear esse nó!
- Supondo que X seja esse nó, possíveis casos a serem analisados:

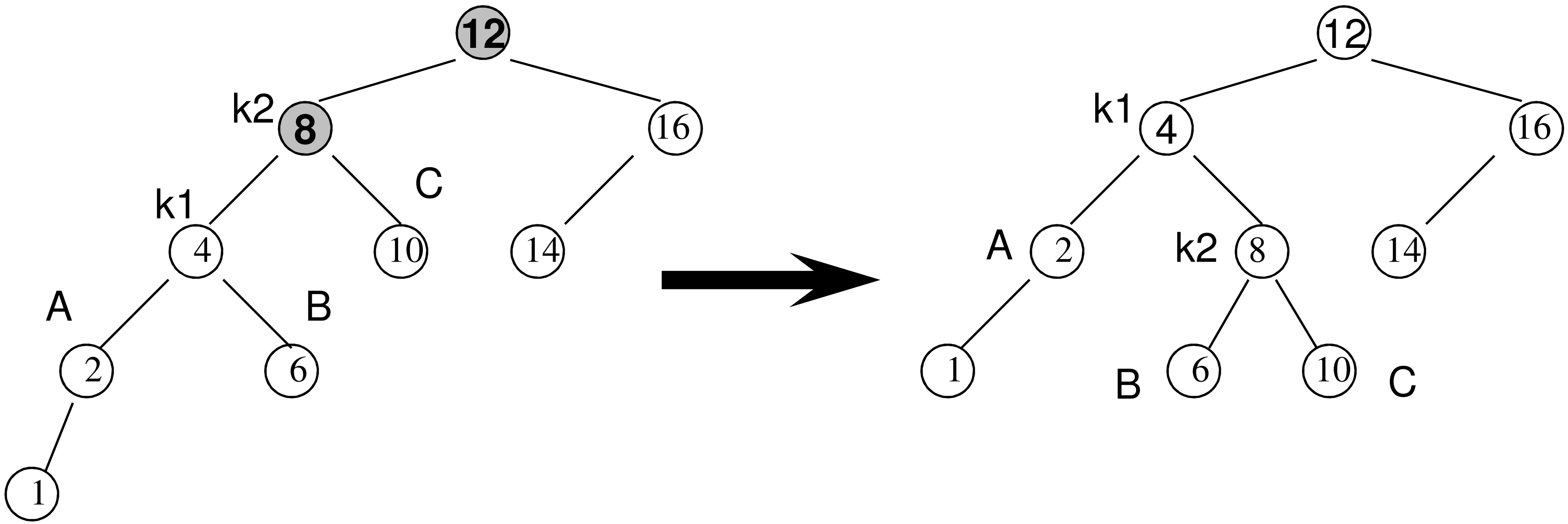
  a) árvore esquerda do filho esquerdo de X
  b) árvore direita do filho esquerdo de X
  c) árvore esquerda do filho direito de X
  d) árvore direita do filho direito de X
- Casos simétricos: a e d (caso 1); b e c (caso 2).

CASO 1: ROTAÇÃO SIMPLES



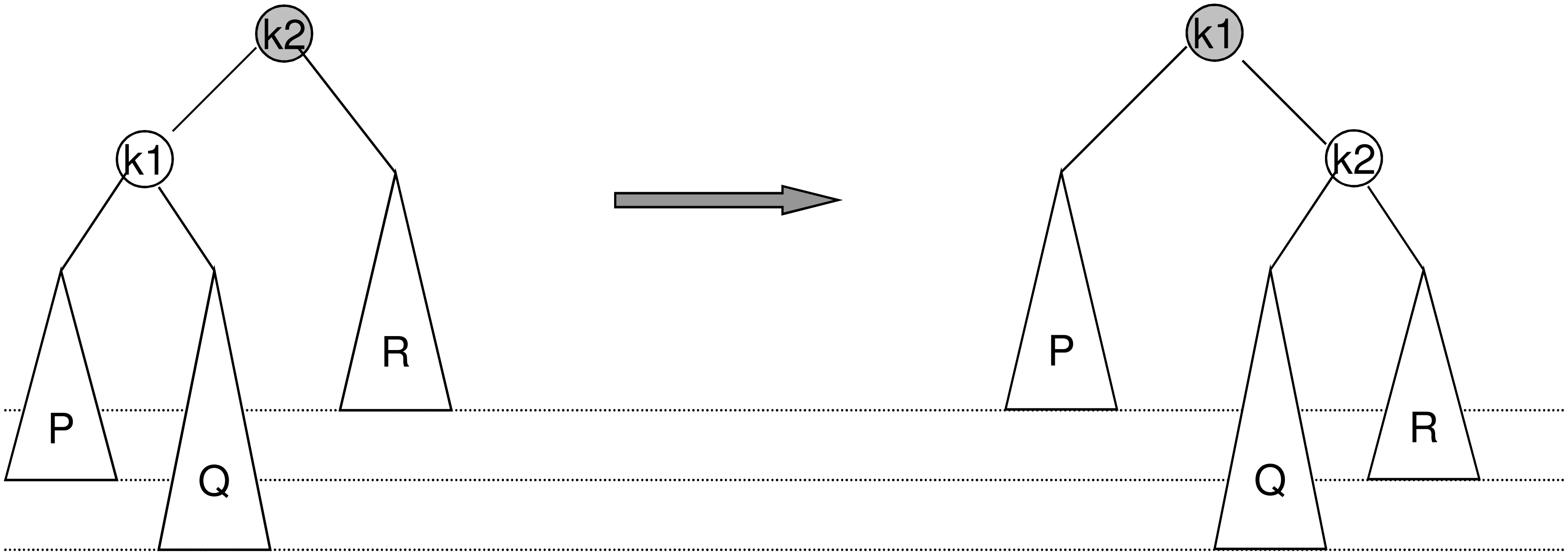
- k2 é nó mais profundo que sofreu desbalanceamento
  - Sua sub-árvore esquerda ficou 2 níveis abaixo da direita
    - B não está no mesmo nível de A (pois k2 estaria desbalanceado antes da inserção)
    - B não está no mesmo nível que C (pois k1 seria o nó mais profundo)

EXEMPLO



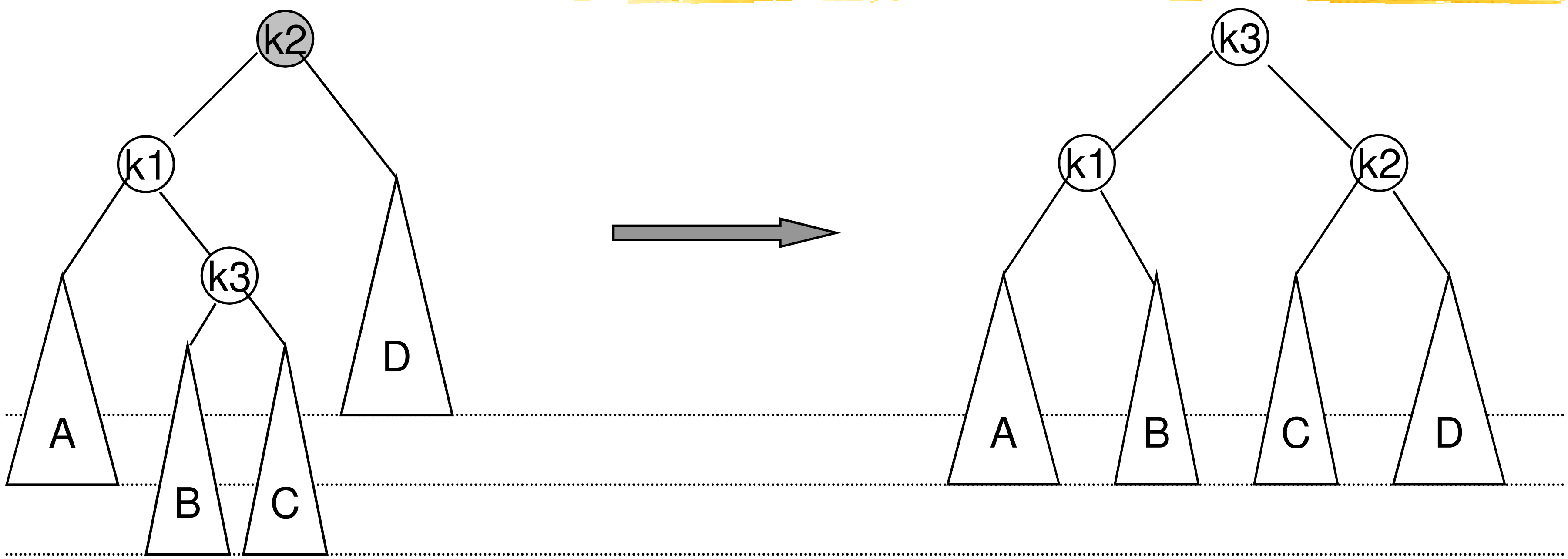
- A árvore resultante é AVL
  - k1 e k2 ficam balanceados
  - A altura da árvore resultante é igual à da árvore anterior à inserção
  - O problema é resolvido em tempo constante

Caso 2



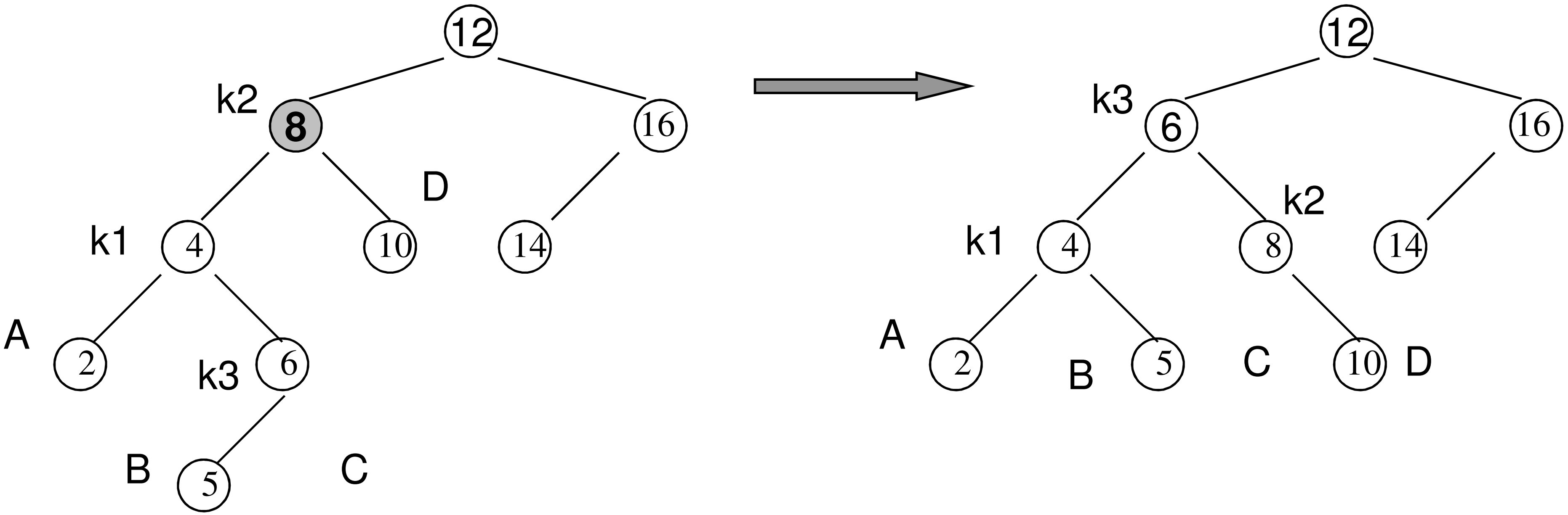
- Uma rotação simples não resolveria o desbalanceamento
  - A sub-árvore Q, que está a 2 níveis de diferença de R, passaria a estar a 2 níveis de diferença de P

Caso 2: rotação dupla



- Uma (e somente uma) das sub-árvores B ou C está 2 níveis abaixo de D
  - k3 ficaria na raiz
  - As novas posições de k1, k2 e das sub-árvores respeitam a ordenação
  - A altura da árvore resultante é igual à da árvore anterior à inserção

EXEMPLO



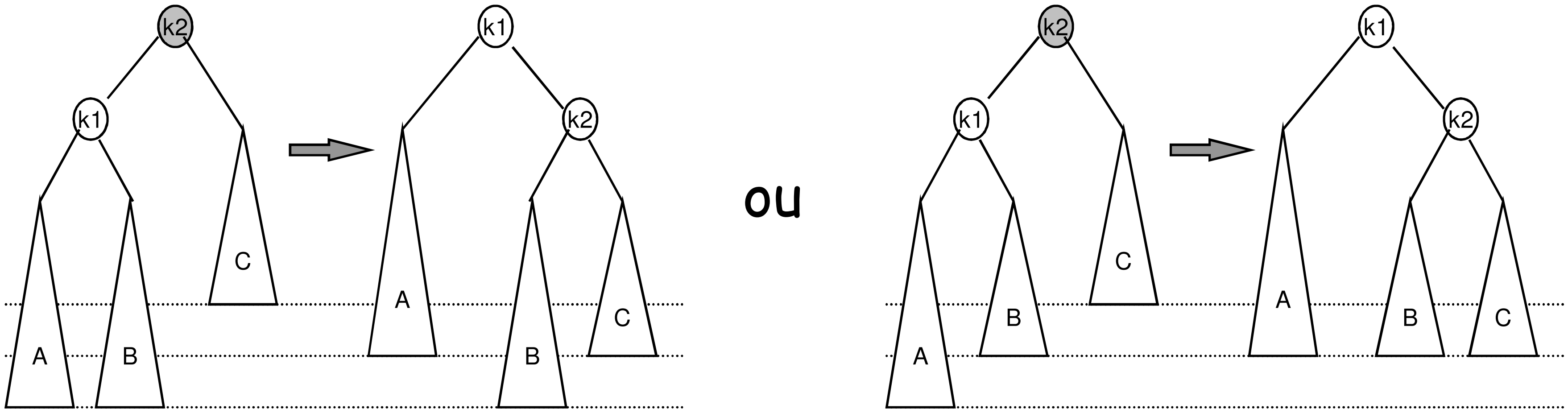
- Essa rotação dupla corresponde a 2 rotações simples
  - Entre k1 e k3
  - Entre k2 e k3
- Também pode ser feita em tempo constante

Eliminações em árvores AVL

- A eliminação de um nó numa árvore AVL é, inicialmente, análoga à que ocorre numa árvore binária de busca:
  - Se for folha, basta eliminá-la.
  - Se tiver um único filho, ele ficará em sua posição.
  - Se tiver dois filhos, elimina-se o nó mais à esquerda da sua sub-árvore à direita, cujo valor passará a ser armazenado em seu lugar.
- É fácil perceber que essa técnica pode provocar desbalanceamentos na árvore AVL.
- O rebalanceamento começará no nó mais profundo que, após a eliminação, perdeu a propriedade AVL.
- Do modo semelhante às inserções, será preciso verificar seis possíveis casos, simétricos dois a dois.

CASO 1: ROTAÇÃO SIMPLES

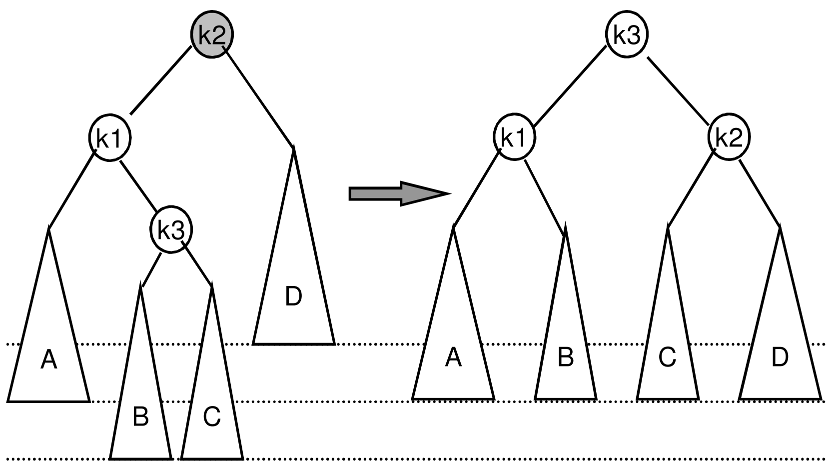
- Nos esquemas abaixo, como a eliminação ocorreu na sub-árvore C, bastará realizar uma rotação simples:



- No segundo esquema, a sub-árvore resultante diminuiu de altura (uma unidade).
  - Por isso, também será preciso realizar um eventual rebalanceamento no pai de k1. Isso pode continuar até a raiz...
- Há também outros dois casos simétricos, onde C é inicialmente a sub-árvore esquerda de k2.

Caso 2: rotação dupla

- No esquema abaixo, B ou C podem ter altura menor (uma unidade). Como a eliminação ocorreu na sub-árvore D, será preciso realizar uma rotação dupla:



- A sub-árvore resultante diminuiu de altura (uma unidade).
  - Por isso, também será preciso realizar um eventual rebalanceamento no pai de k3. Isso pode continuar até a raiz...
- Há um caso simétrico, onde inicialmente D é a sub-árvore esquerda de k2 e k3 é filho esquerdo de k1.

Árvores AVL

- Nas árvores AVL, inserções gastam tempo constante, e eliminações gastam, no pior caso, tempo proporcional à altura da árvore.
- Exercício:
  - Implementar uma sequência de inserções e eliminações numa árvore AVL, e depois imprimir percursos nessa árvore.
- <u>Dica</u>: em cada nó, será preciso manter um inteiro (-1, 0 ou 1) que indica a diferença entre as alturas das suas sub-árvores.

ALTURA DE UMA ÁRVORE AVL

- Seja $n(h)$ o número mínimo de nós de uma árvore AVL com altura h.
- Sabemos que $n(0)=1$ e $n(1)=2$.
- Para $h>1$, essa árvore AVL mínima será formada pela raiz, por uma sub-árvore de altura $h-1$ e por outra sub-árvore de altura $h-2$.
- Portanto, $n(h) = 1 + n(h-1) + n(h-2)$, para $h>1$.
- Como $n(h-1) > n(h-2)$, sabemos que $n(h) > 2.n(h-2)$.
- Repetindo:
  - $n(h) > 2.n(h-2) > 2.(2.n(h-2-2)) = 4.n(h-4)$
  - $n(h) > 4.n(h-4) > 4.(2.n(h-4-2)) = 8.n(h-6)$
  - Generalizando: $n(h) > 2^i.n(h-2i)$, para $i>0$.

ALTURA DE UMA ÁRVORE AVL

- $n(h) > 2^i.n(h-2i)$, para $i>0$.
- Consideremos o caso $h = 2i+1$, ou seja, $i = (h-1)/2$:
  - $n(h) > 2^{(h-1)/2}.n(2i+1-2i)$
  - $n(h) > 2.2^{(h-1)/2}$, pois $n(1) = 2$
  - $n(h) > 2^{(h+1)/2}$
  - $\lg n(h) > (h+1)/2$
  - $h < 2.\lg n(h) - 1$
- $h = O(\log n)$, ou seja, a altura de uma árvore AVL é de ordem logarítmica em relação ao seu número de nós.
- Portanto, os algoritmos de busca e de eliminação na árvore AVL são logarítmicos!

Altura de uma árvore AVL

- Por outro lado, é fácil verificar que $n(h) = F(h+3)-1$, onde $F(h)$ é o h-ésimo número de Fibonacci.

- Mais precisamente, sabe-se que $h < 1,44.\lg(n+2)$, onde h é a altura e n é o número de nós de uma árvore AVL, ou seja, o pior caso tem um fator multiplicativo pequeno.