Árvore Binária de Busca Balanceada (AVL)

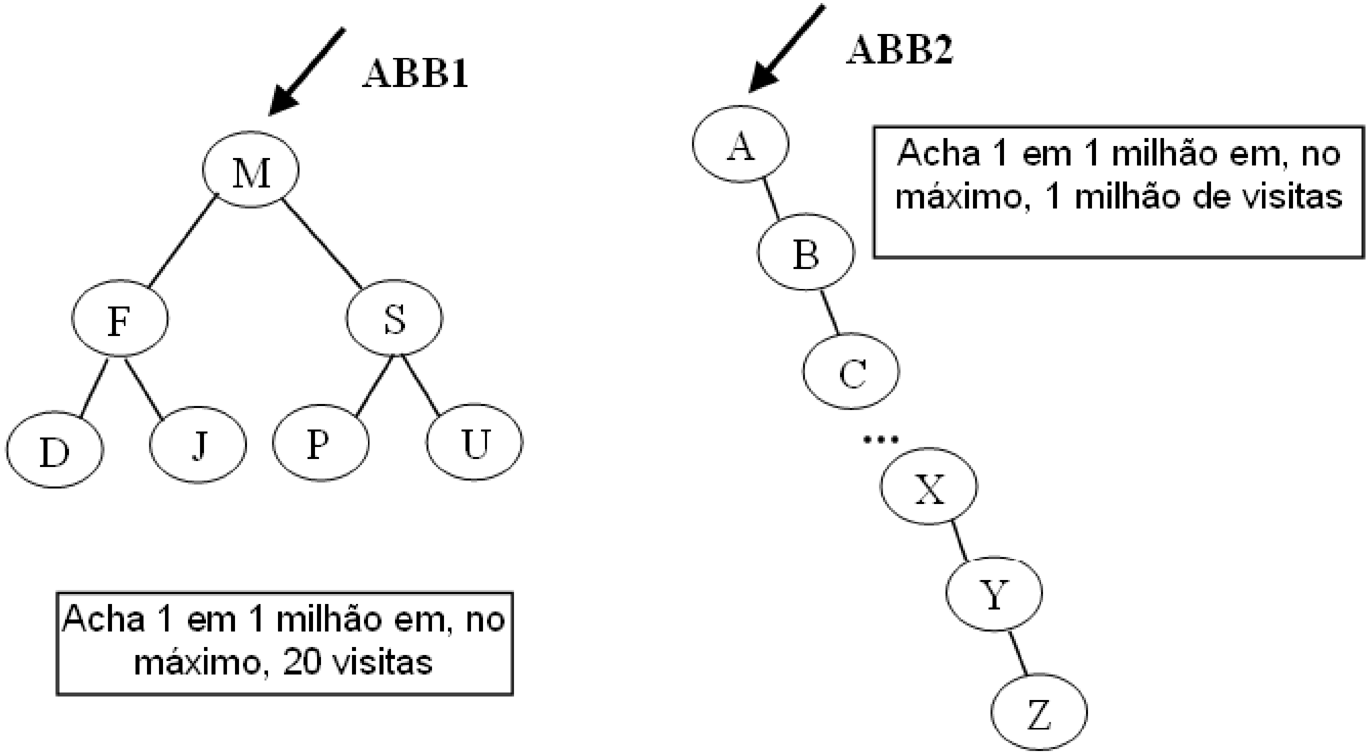
Objetivos

- Entender o conceito de balanceamento, e sua importância para a eficiência das árvores binárias de busca;
- Entender a lógica do processo de balanceamento de uma árvore binária de busca;
- Desenvolver habilidade para adaptar a lógica do balanceamento a novas situações, e para a elaboração de algoritmos sobre árvores binárias de busca balanceadas.

Exercícios

a) Dada uma ABB inicialmente vazia, insira (desenhe) os seguintes elementos (nessa ordem): M, F, S, D, J, P, U.

b) Dada uma ABB inicialmente vazia, insira (desenhe) os seguintes elementos (nessa ordem): A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.



As árvores ABB1 e ABB2 ilustram um problema com relação a árvores binárias de busca: dependendo da ordem de inserção dos elementos, as árvores podem ficar equilibradas, como a ABB1, ou totalmente desequilibradas, como a ABB2. Por árvore “equilibrada” entendemos uma árvore com os tamanhos de suas sub-árvores esquerda e direita equivalentes.

ABB

- Problema:
  - A árvore binária pode gerar para uma estrutura próxima a uma lista ligada, tanto no algoritmo de inserção quanto no algoritmo de remoção, e o tempo de acesso deixa de ser logarítmico;
- Solução:
  - Árvore AVL: Árvore Binária de Busca Balanceada

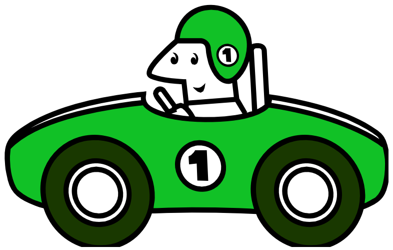
Definição

- **Definição de ABBs Balanceadas**
  - Uma árvore binária de busca é dita balanceada (ou ABBB, ou ainda árvore AVL) se e somente se para cada nó as alturas de suas sub-árvores diferem de, no máximo, 1.

- **Estratégia**: alterar os algoritmos de inserir e remover de modo a manter a ABB sempre balanceada.

Lógica de Balanceamento

- Os algoritmos sobre árvores binárias de busca precisam ser ajustados para primeiramente **monitorar** o balanceamento da árvore, verificando se em um dado momento a árvore está ou não balanceada. E precisam também ser adaptados para tomar atitudes corretivas, caso a árvore estiver se tornando desbalanceada.



Monitoramento do Balanceamento

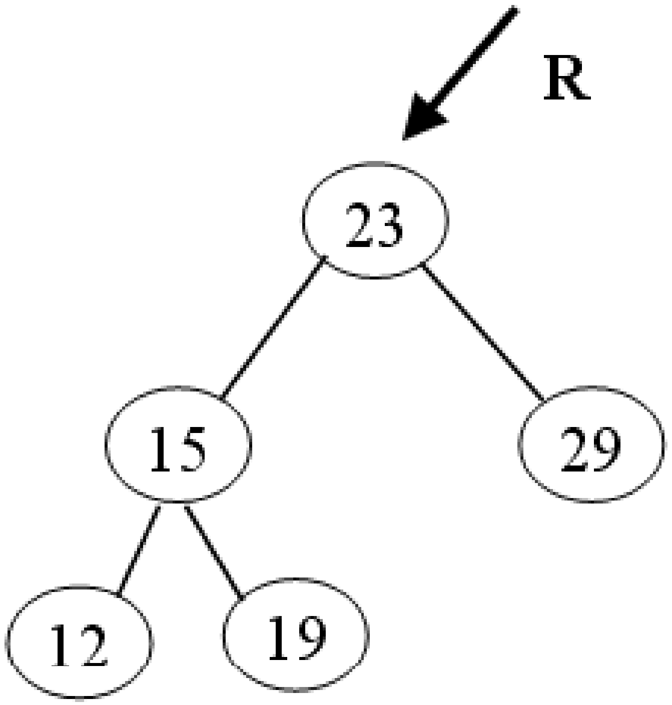
- Seja uma árvore R com sub-árvores E (esquerda) e D (direita), com alturas He e Hd, respectivamente. Se um novo elemento é inserido em E, causando um aumento na altura He, três casos podem ocorrer.

antes da inserção	conseqüências
se He = Hd	E e D passarão a ter alturas diferentes (E será maior em 1), mas o critério de balanceamento não é violado (diferença das alturas pode ser no máximo 1).
se He < Hd (em 1)	E e D passarão a ter alturas iguais, e o critério não é violado.
se He > Hd (em 1)	E passará a ser maior que D em 2, o que viola o critério => a árvore precisa ser ajustada.

Exercício

- Verifique se a inserção cada uma das chaves seguintes causaria desbalanceamento:
  - 25
  - 40
  - 13
  - 17
  - 21

causam desbalanceamento

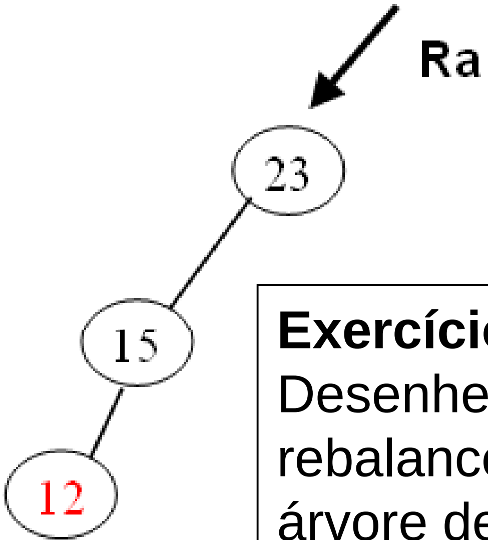


Rebalanceamento

- Quando uma árvore é desbalanceada, precisamos ajustá-la para que volte a ser balanceada. Chamamos a esse processo: rebalanceamento

Rebalanceamento – Casos do Algoritmo Insere

- **Caso 1: Rotação Simples Esquerda-Esquerda do Insere**

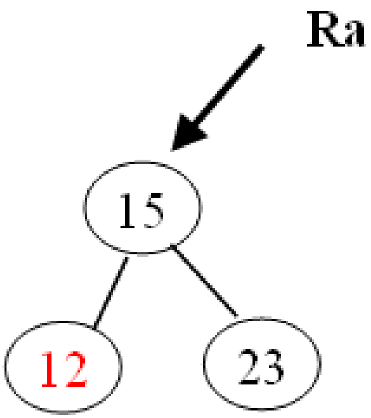


A chave 12 acabou de ser inserida, o que causou desbalanceamento na árvore

Como essa árvore pode ser rebalanceada?

Exercício. Nova Árvore
Desenhe uma nova árvore, resultante do rebalanceamento da árvore da Figura 20.2. A nova árvore deve ser uma árvore binária de busca, e deve ser balanceada.

Árvore Rebalanceada

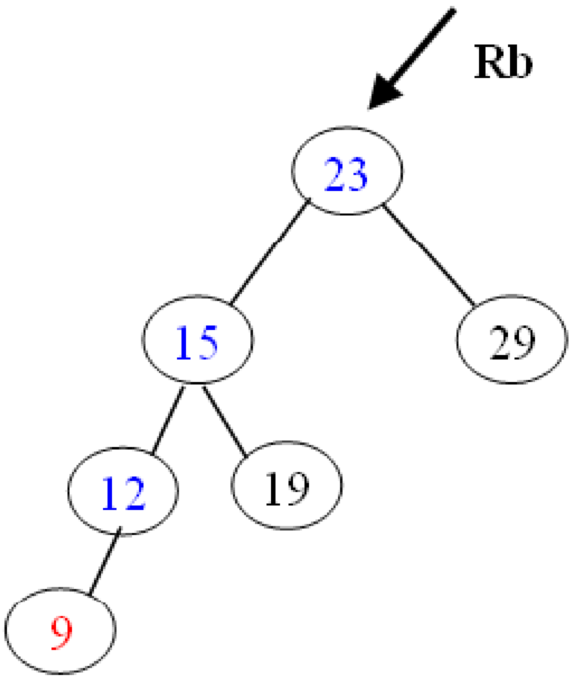


Existe outra maneira de balancear essa árvore com 3 chaves?

Não!

Outro Exemplo

- rotação simples esquerda-esquerda
  - Chave 9 acabou de ser inserida

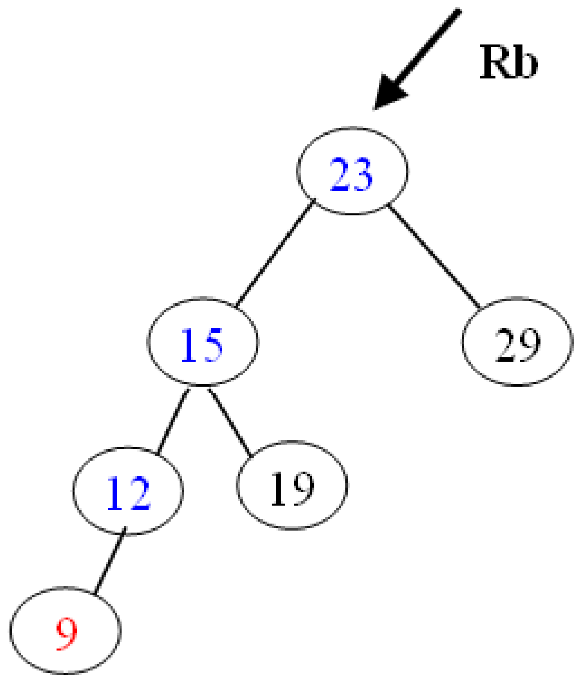


Exercício. Desenhe uma Nova Árvore
Desenhe uma nova árvore, resultante do rebalanceamento da árvore da Figura ao lado. A nova árvore deve ser uma árvore binária de busca, e deve ser balanceada.

Passo 1

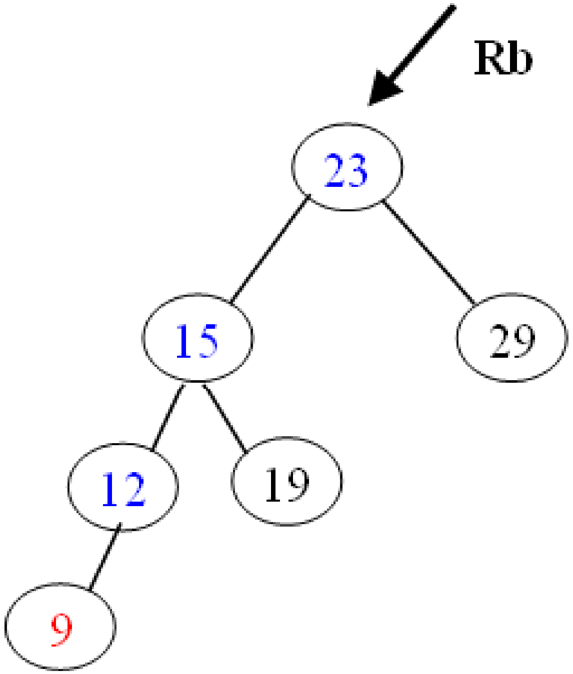
- Note que as chaves 15, 12 e 23 estão em posição idêntica à posição da solução do primeiro exercício (árvore com apenas essas 3 chaves).

- **Como Identificar as 3 Chaves Principais?**
  - Para identificar essas 3 chaves, primeiramente responda à seguinte pergunta: a partir de qual chave foi detectado o desbalanceamento?



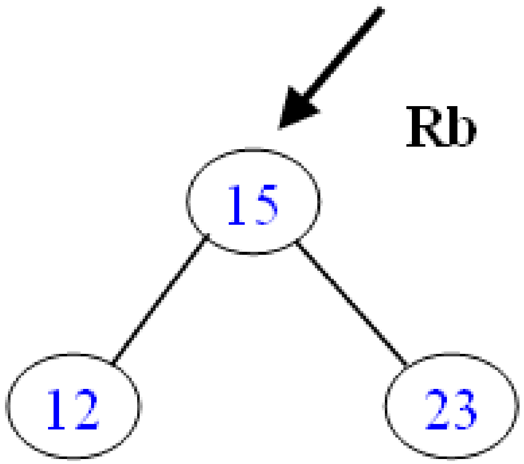
- O desbalanceamento foi detectado na chave 23:
  - A sub-árvore direita de 23 tem tamanho 1;
  - A sub-árvore esquerda de 23 tem tamanho 3
  - 3 menos 1 = 2, o que quebra o critério de balanceamento
- Se mais que um nó estiver desbalanceado, escolha o nó “mais em baixo”, ou seja, mais próximo da posição de inserção.

- Após identificar em qual nó ocorreu o desbalanceamento, para identificar as outras 2 chaves principais, siga, na árvore, a partir do nó desbalanceado, em direção à chave que acabou de ser inserida e causou o desbalanceamento.
- O caminho será para a esquerda, e depois para a esquerda novamente. É por isso que o nome desse primeiro caso de rebalanceamento é *Rotação Simples Esquerda-Esquerda.*
- Ao seguir para à esquerda-esquerda, passará pelos 2 nós que devem se juntar ao nó desbalanceado. São esses os 3 nós principais do processo de rebalanceamento.



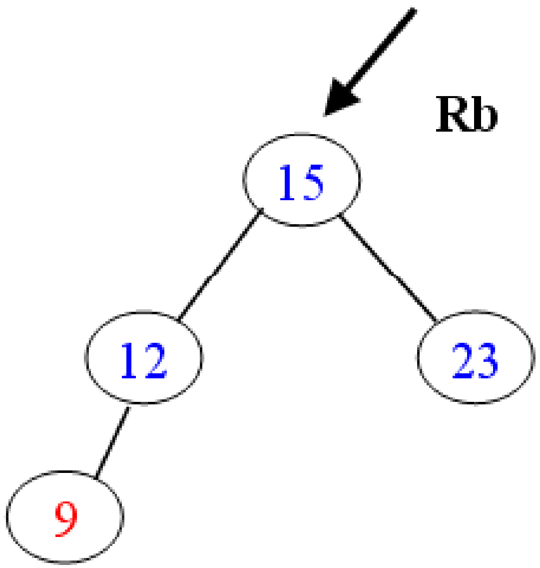
Posicionando as 3 chaves Principais

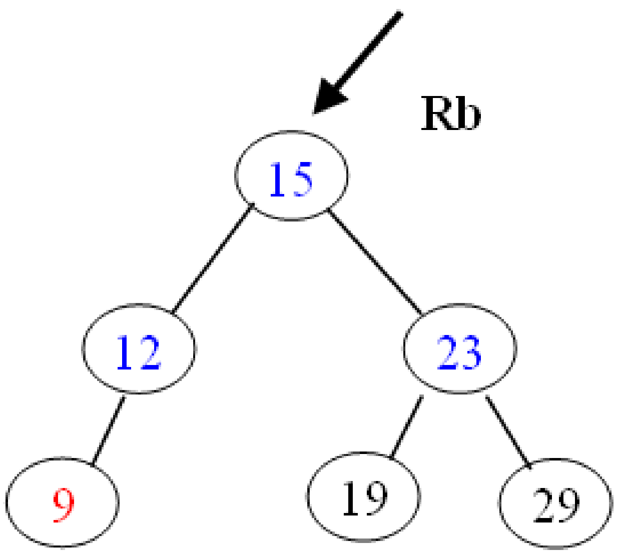
- Uma vez identificadas as 3 chaves principais, basta posiciona-las na única posição possível que mantém o critério de árvore binária de busca, e também o critério de balanceamento.



Posicionando as Demais Chaves

- Para posicionar as demais chaves da árvore – 9, 19 e 29 - é preciso seguir o critério que define uma Arvore Binária de Busca (chaves menores a esquerda da raiz, chaves maiores à direita).
- Onde colocaremos o 9? E o 19? E o 29?

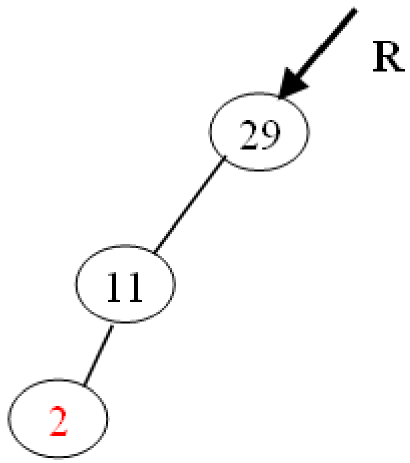




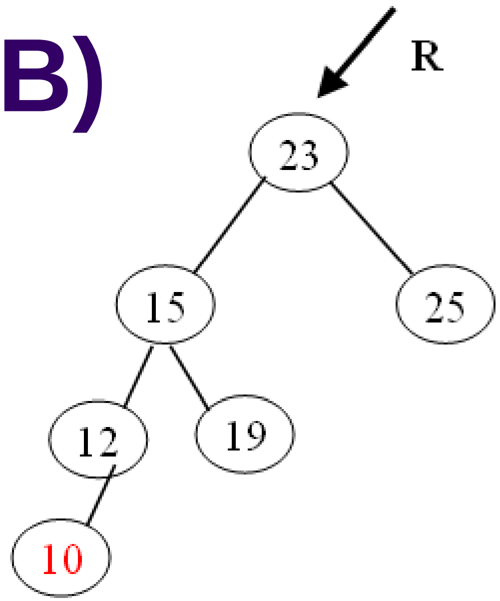
Exercício

- **Desenhe a Nova Árvore EE do Insere**
  - Desenhe a nova árvore, resultante do rebalanceamento da árvore das figuras fornecidas. Considere que a chave que acabou de ser inserida está indicada em vermelho. A nova árvore deve ser uma árvore binária de busca, e deve ser balanceada. Identifique a chave onde ocorreu o desbalanceamento, identifique as 3 chaves principais, posicione-as, e então posicione as demais chaves.

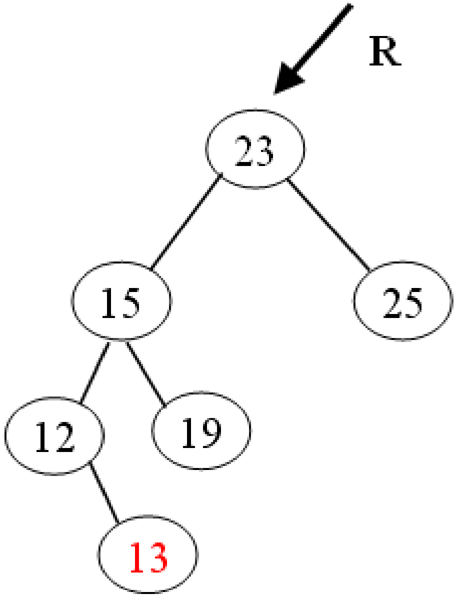
A)



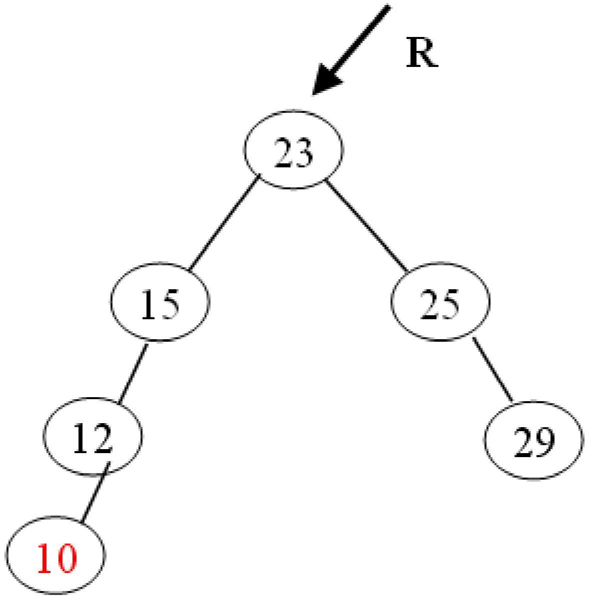
B)



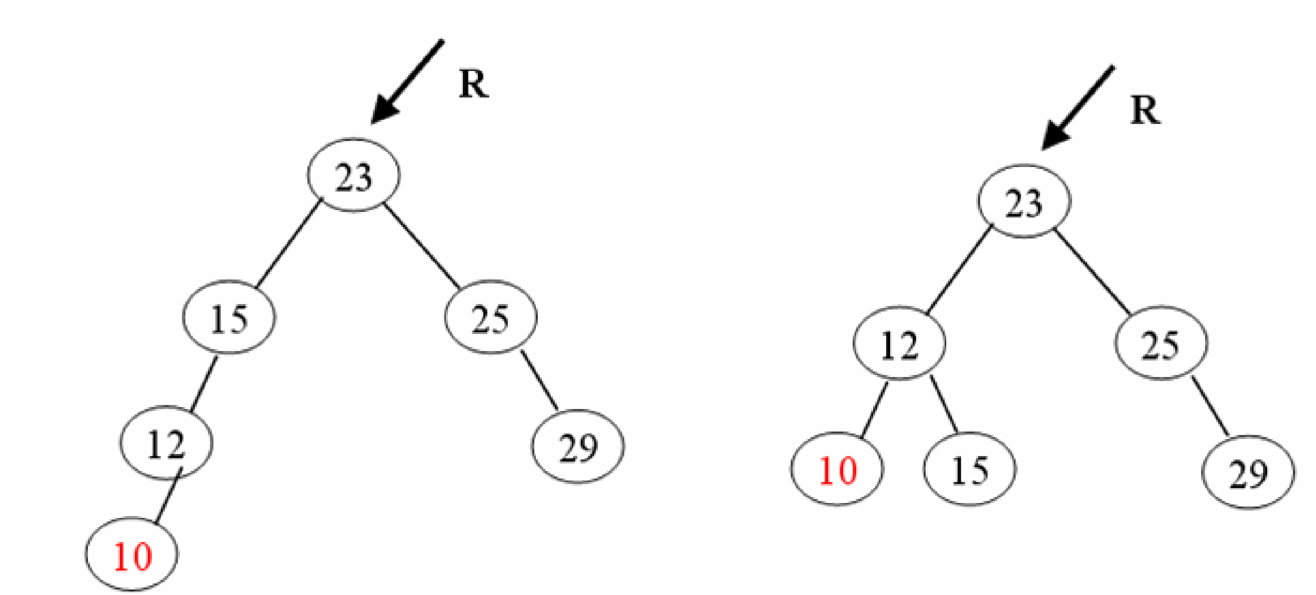
c



d)

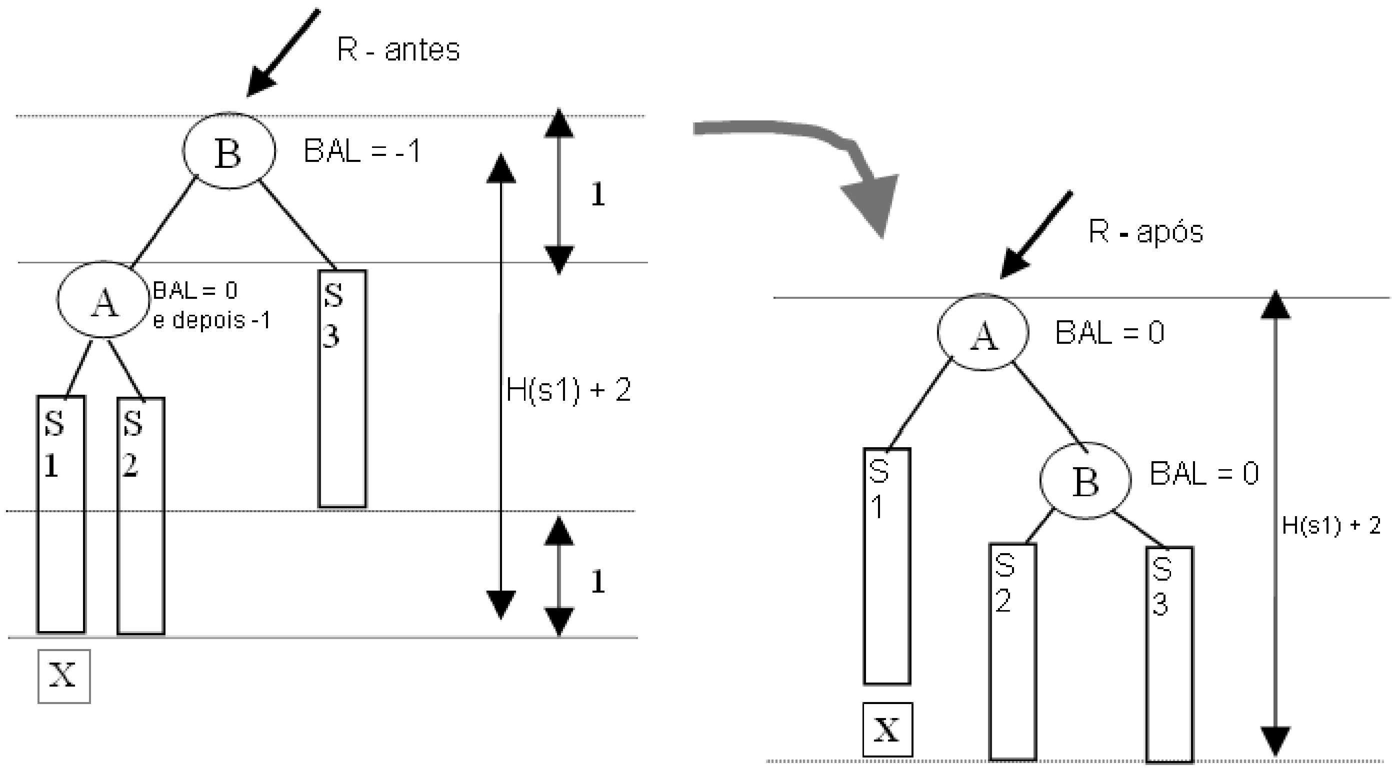


- A) 11 na raiz
- B) 15 na raiz
- C) 15 na raiz
- D)

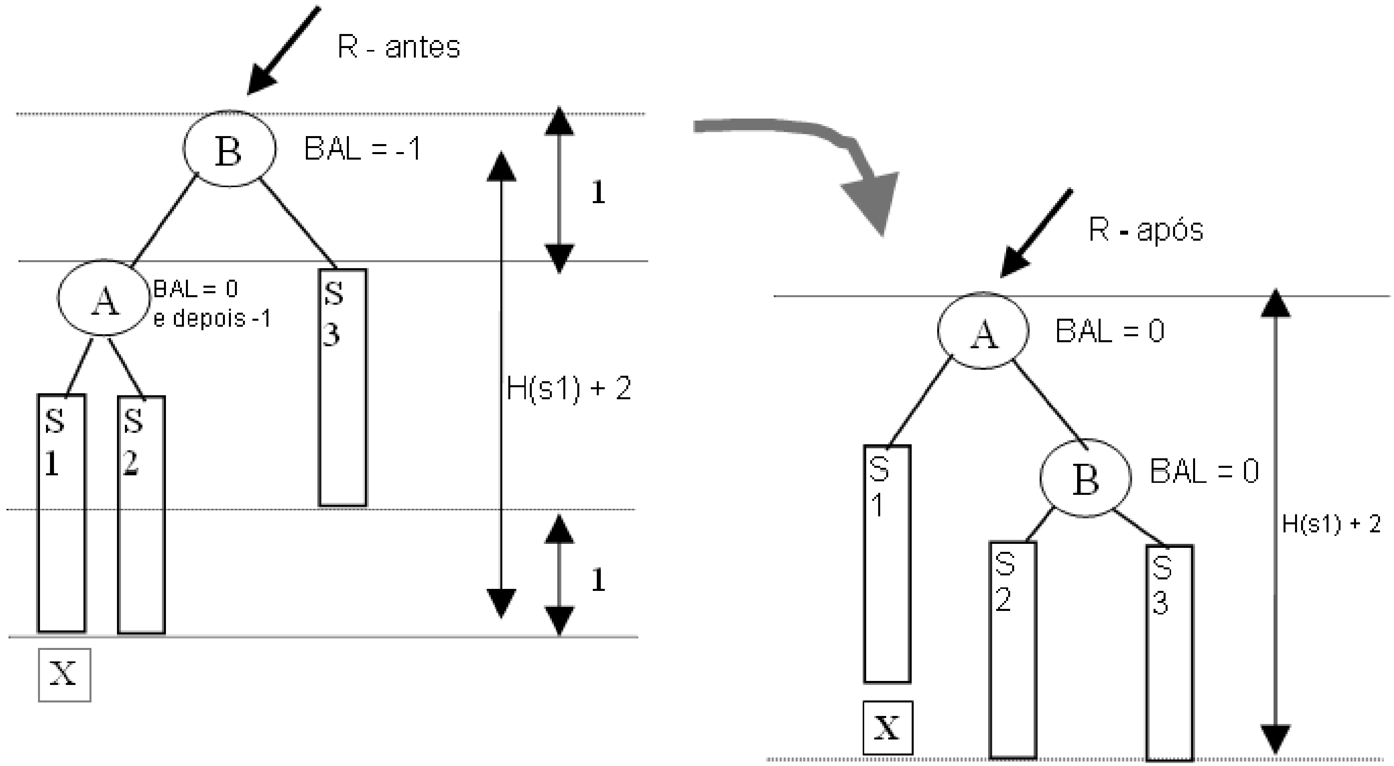


Generalização do Caso 1: Rotação Simples Esquerda-Esquerda (Insere)

- Definição de novo campo BAL = Hd – He
- Para atender o critério de balanceamento, o campo BAL poderá assumir os valores -1, 0 e +1.
- Ajustamos o BAL da árvore de baixo para cima.

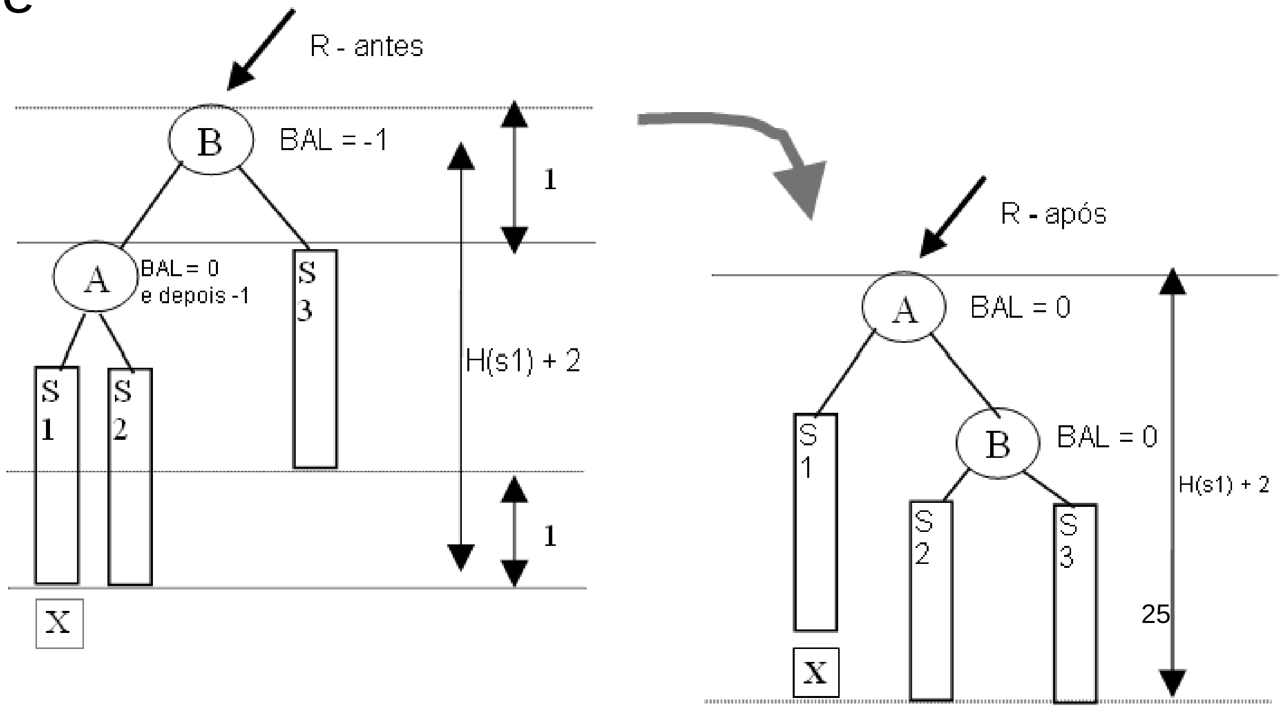


- Após o rebalanceamento, a altura total da árvore é a mesma altura que a árvore tinha antes da inserção da chave X, ou seja, H(S1) + 2.

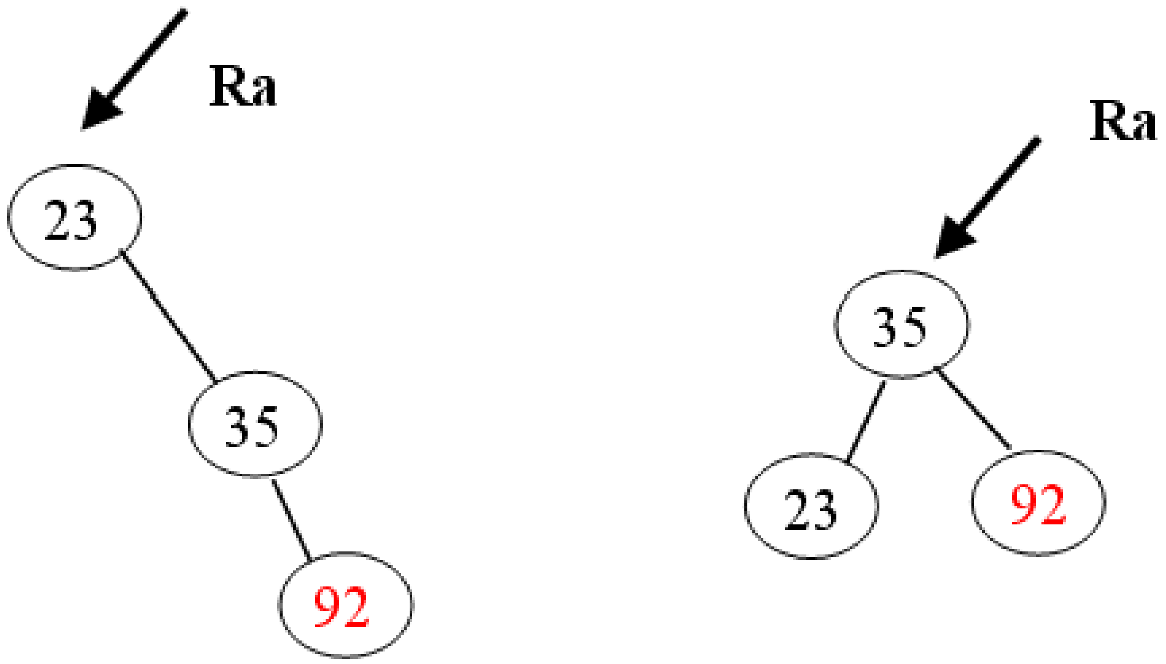


Algoritmo de Rotação Simples Esquerda-Esquerda

filho = esq(R) { A }
esq(R) = dir(filho) { S2 }
dir(filho) = R { B }
BAL(R) = 0
BAL(filho) = 0
mudou-altura = false
R = filho { A }

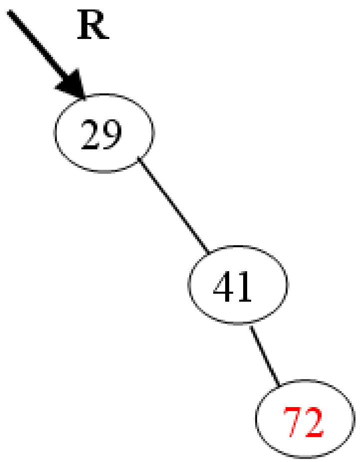


Caso 1´: Rotação Simples Direita-Direita (Insere)

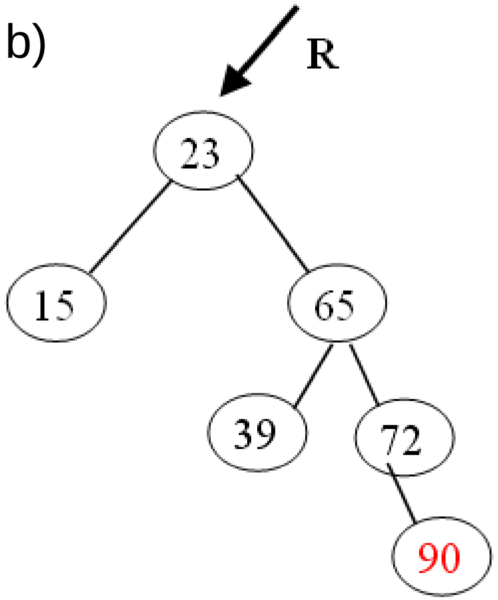


- Desenhe a nova árvore, resultante do rebalanceamento da árvore das figuras fornecidas.

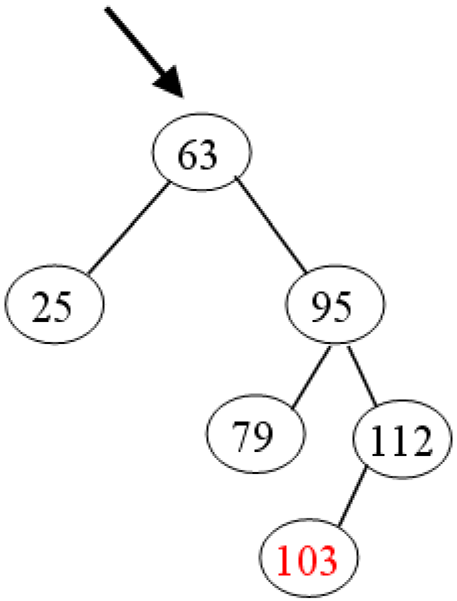
a)



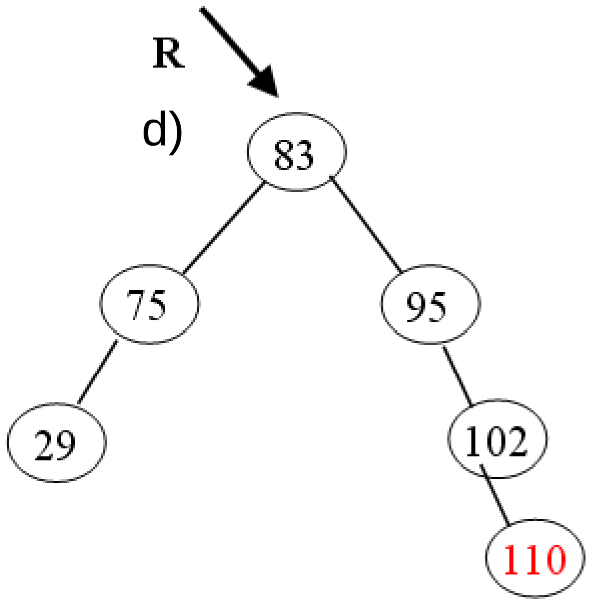
b)



c)



d)

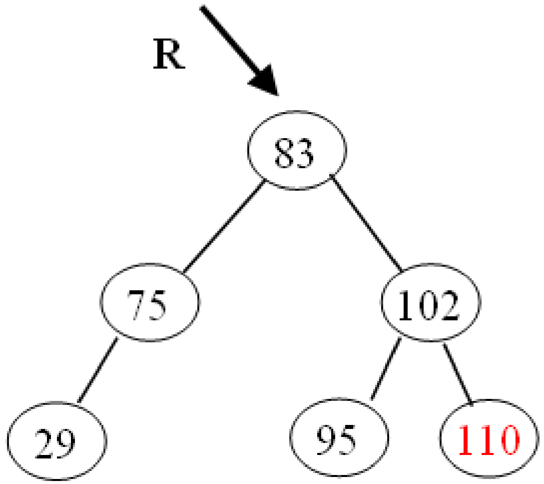


A) 41 na raiz

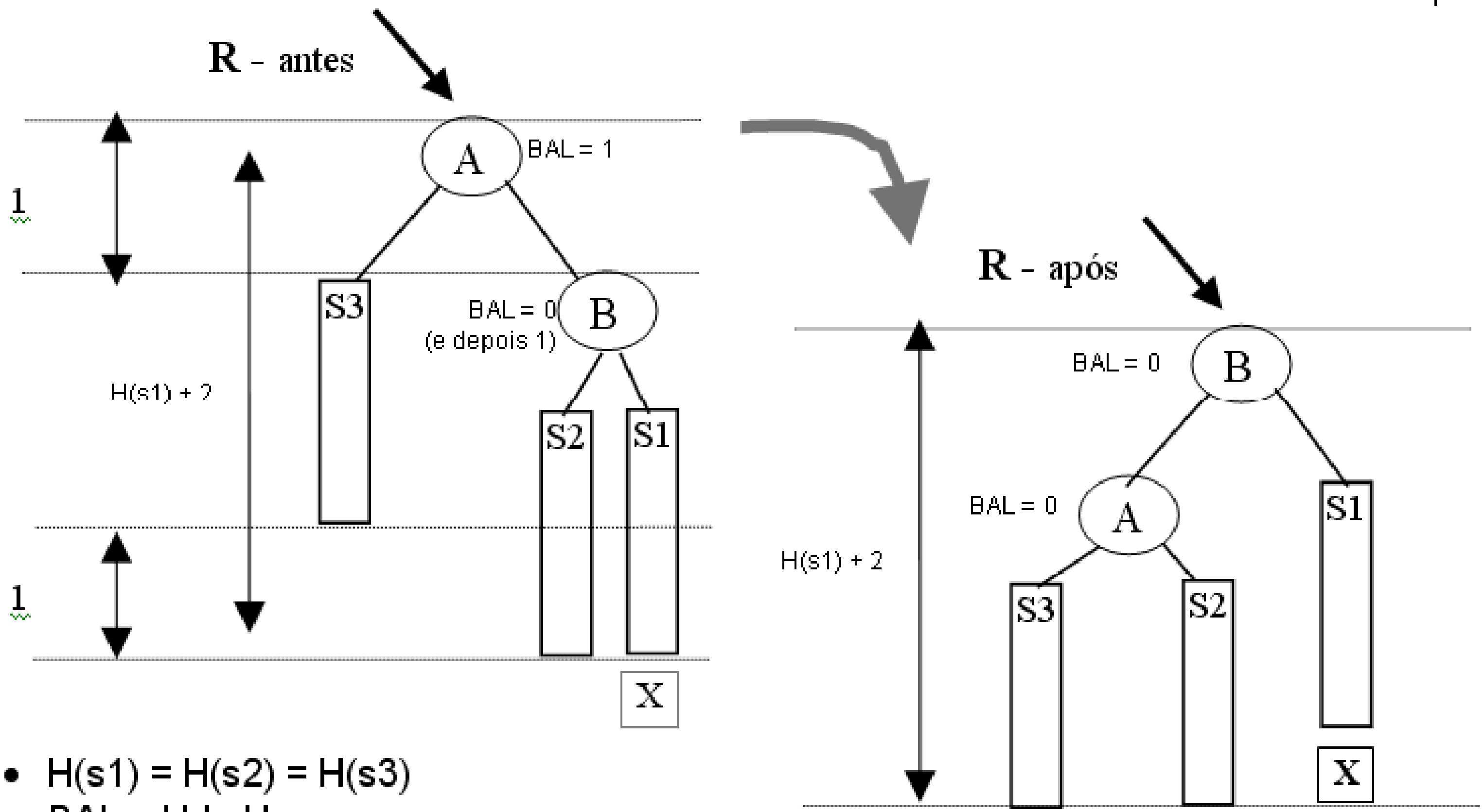
B) 65 na raiz

C) 95 na raiz

D)



Generalização do Caso 1´: Rotação Simples Direita-Direita do Insere



- H(s1) = H(s2) = H(s3)
- BAL = Hd - He
- inserindo X é preciso rebalancear
- após o rebalanceamento, H(R) é o mesmo de antes da inserção de X, e os BAL são = 0

Exercício: faça o algoritmo para a Rotação Simples DD.

O algoritmo para a Rotação Simples DD.

filho = dir(R) { B }

dir(R) = esq(filho) { S2 }

esq(filho) = R { A }

BAL(R) = 0

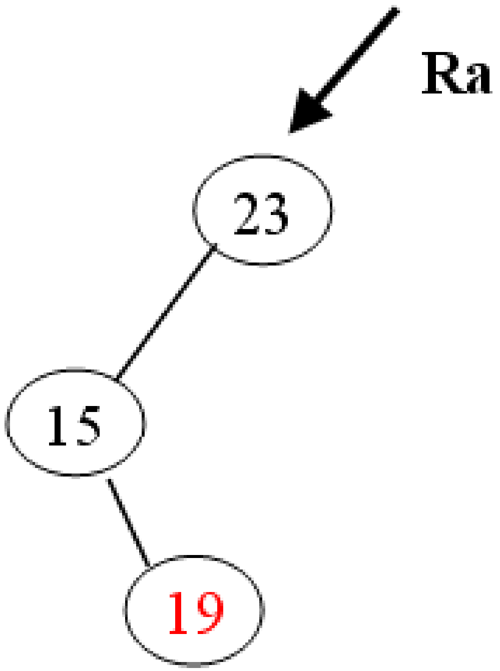
BAL(filho) = 0

mudou-altura = false

R = filho

Caso 2: Rotação Dupla Esquerda-Direita (ED do Insere)

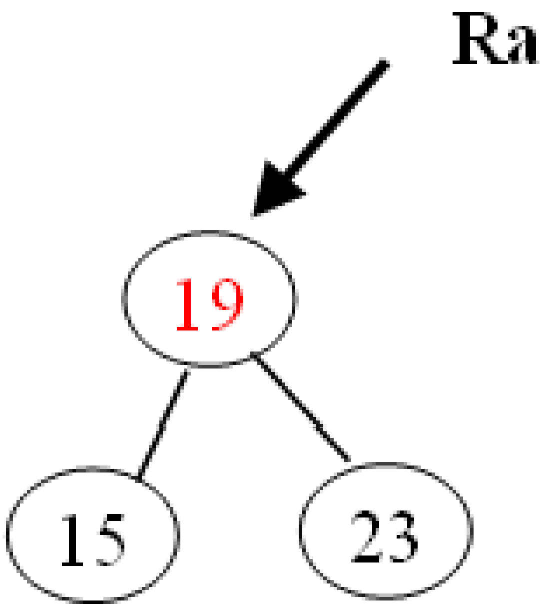
- Um exemplo do caso 2 de rebalanceamento: rotação dupla esquerda-direira (ED).



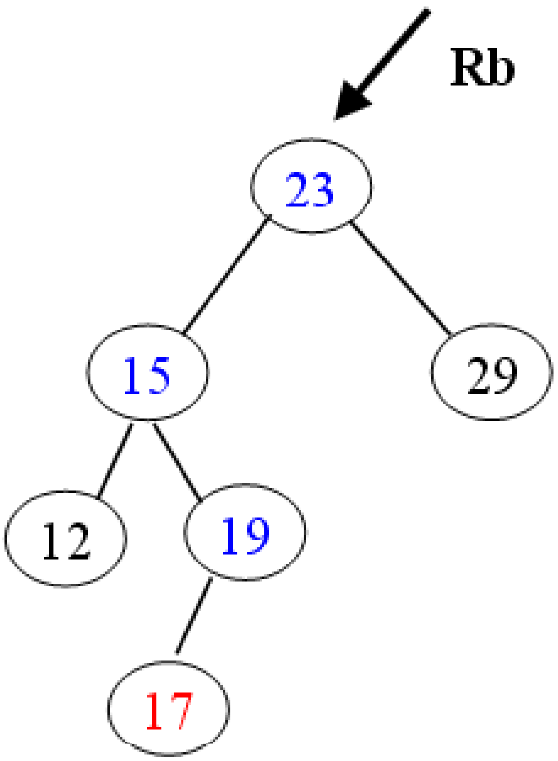
A chave 19 acabou de ser inserida, o que causou desbalanceamento na árvore. A altura da sub-árvore esquerda (de 23) é 2, e a altura da sub-árvore direita é zero. A diferença das alturas acaba sendo 2, o que quebra o critério de balanceamento. Como essa árvore pode ser rebalanceada?

Exercício: desenhe a nova árvore balanceada

Árvore Rebalanceada



Segundo exemplo da rotação dupla esquerda-direita

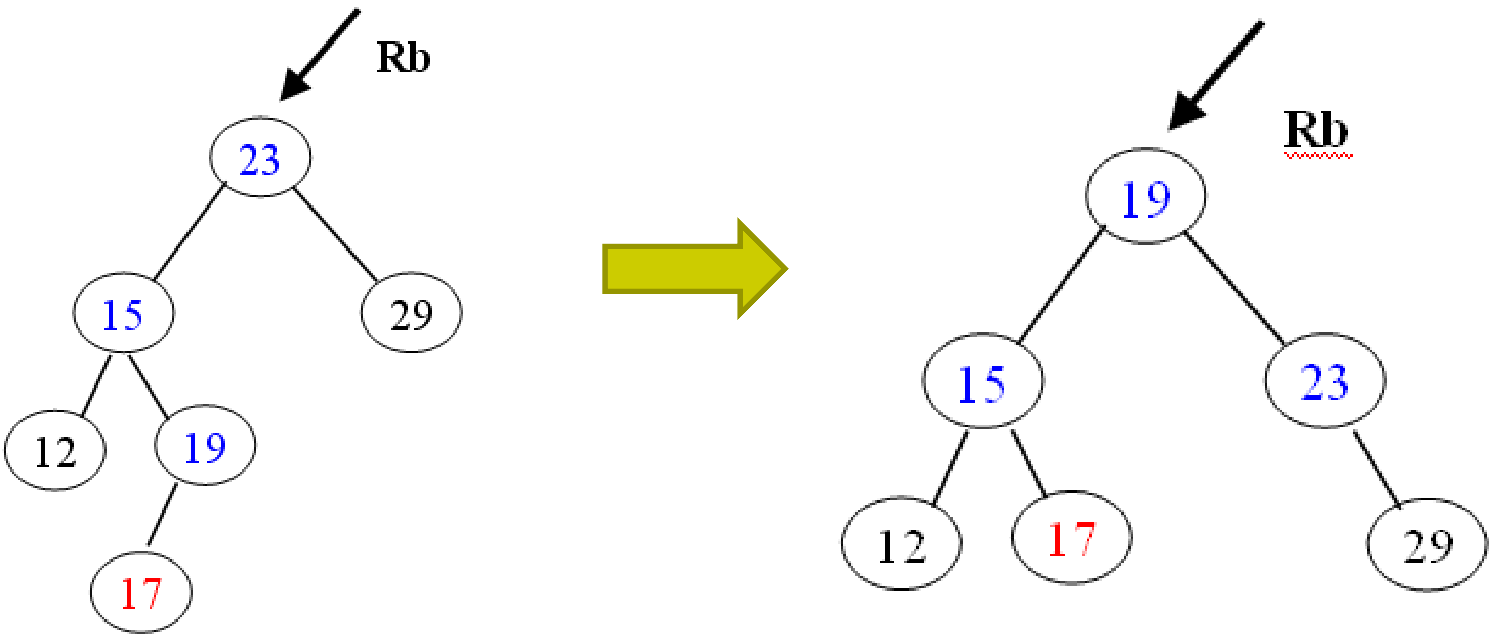


A chave de valor 17, em vermelho, acabou de ser inserida, causando desbalanceamento.

Exercício: desenhe a nova árvore balanceada

Relembre

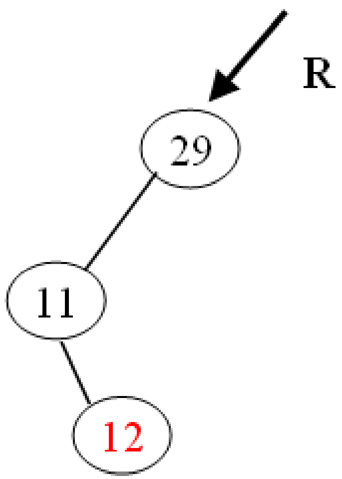
- A identificação das chaves principais:
  - Após identificar em qual chave ocorreu o desbalanceamento, para identificar as outras 2 chaves principais, siga, na árvore, em direção à chave que acabou de ser inserida e causou o desbalanceamento



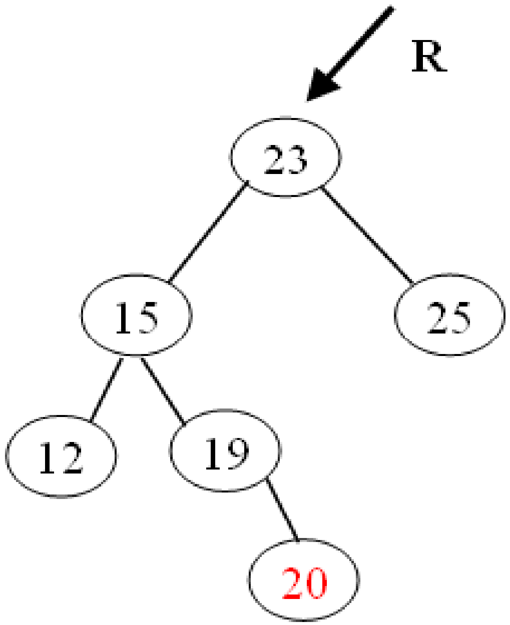
Casos da rotação dupla esquerda-direita

Exercício: Desenhe a nova árvore, resultante do rebalanceamento da árvore das figuras fornecidas. Considere que a chave que acabou de ser inserida está indicada em vermelho. A nova árvore deve ser uma árvore binária de busca, e deve ser balanceada. Identifique a chave onde ocorreu o desbalanceamento, identifique as 3 chaves principais, posicione-as, e então posicione as demais chaves.

A)

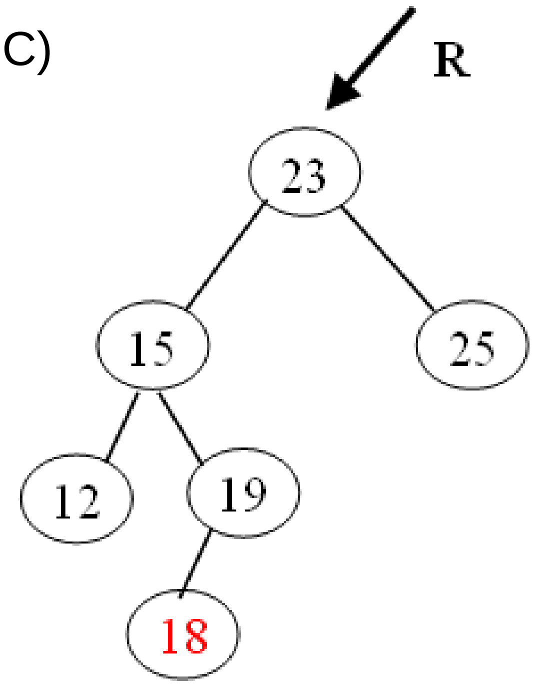


B)

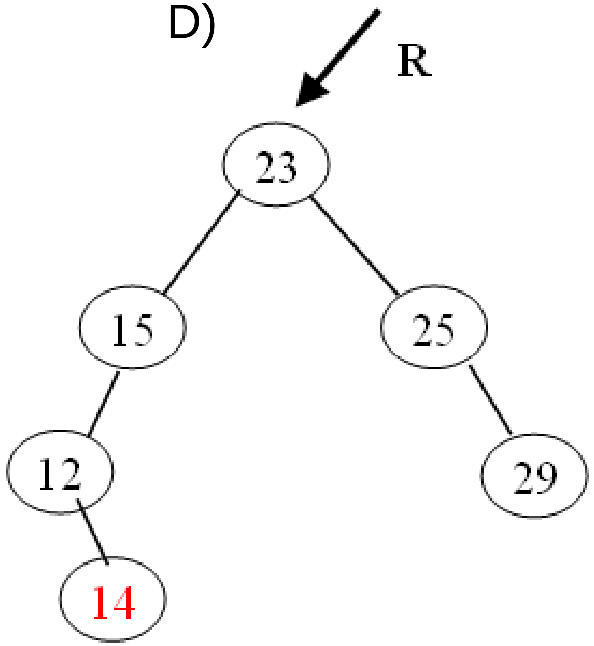


Generalização do Caso 2: Rotação Dupla Esquerda-Direita do Insere

C)



D)



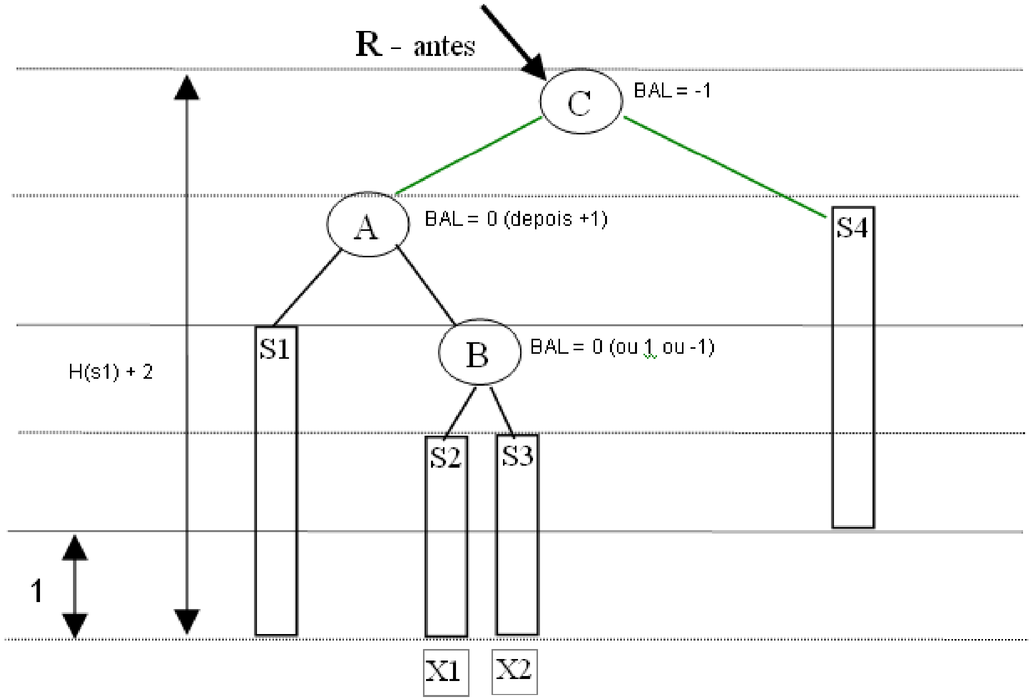
a) chave 12 na raiz.

b) chave 19 na raiz.

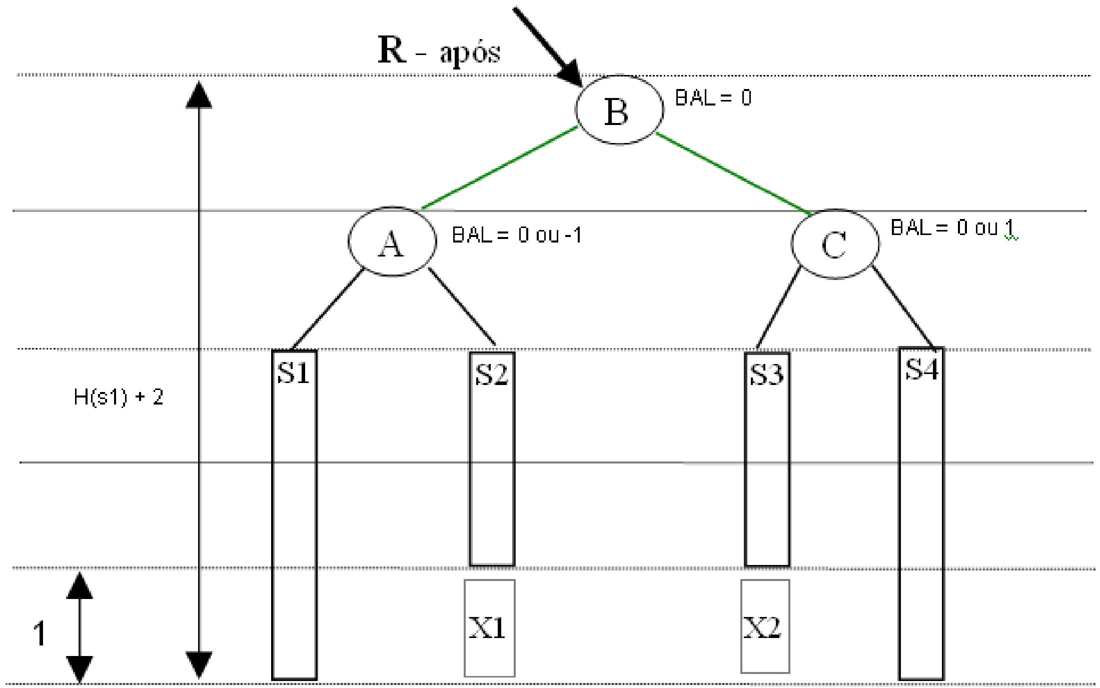
c) chave 19 na raiz.

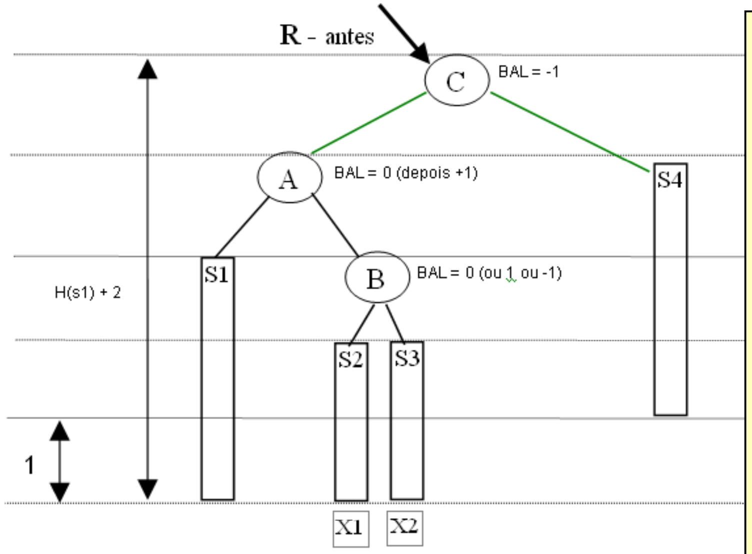
d) chave que desbalanceou foi a chave 15

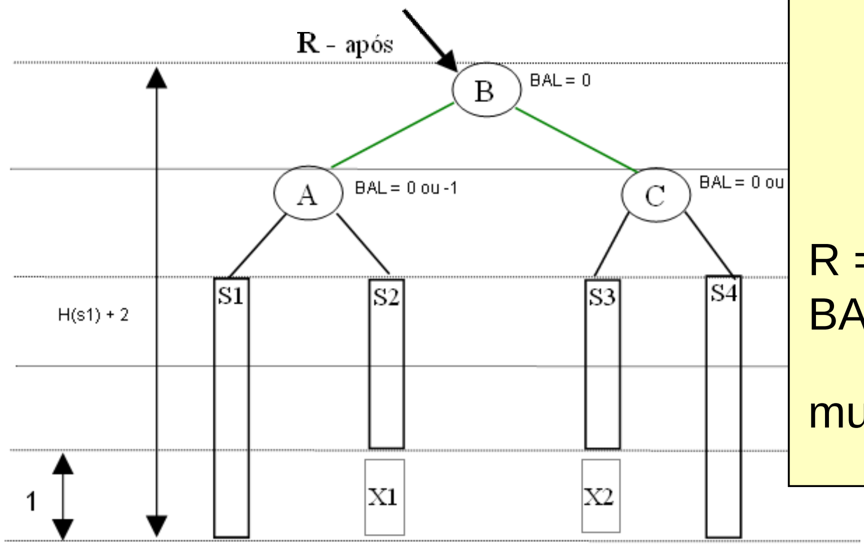
Generalizando rotação dupla Esquerda/Direita



Ou inserindo o X1 **ou** o X2.







Algoritmo para rotação dupla ED do Insere

filho = esq(R)   { A }
neto = dir(filho)   { B }
dir(filho) = esq(neto)   { S2 }
esq(neto) = filho   { A }
esq(R) = dir(neto)   { S3 }
dir(neto) = R   { C }

{ algoritmo já executou, recursivamente, em neto e em filho, ajustando seus balanceamentos, e agora está ajustando o BAL de R }

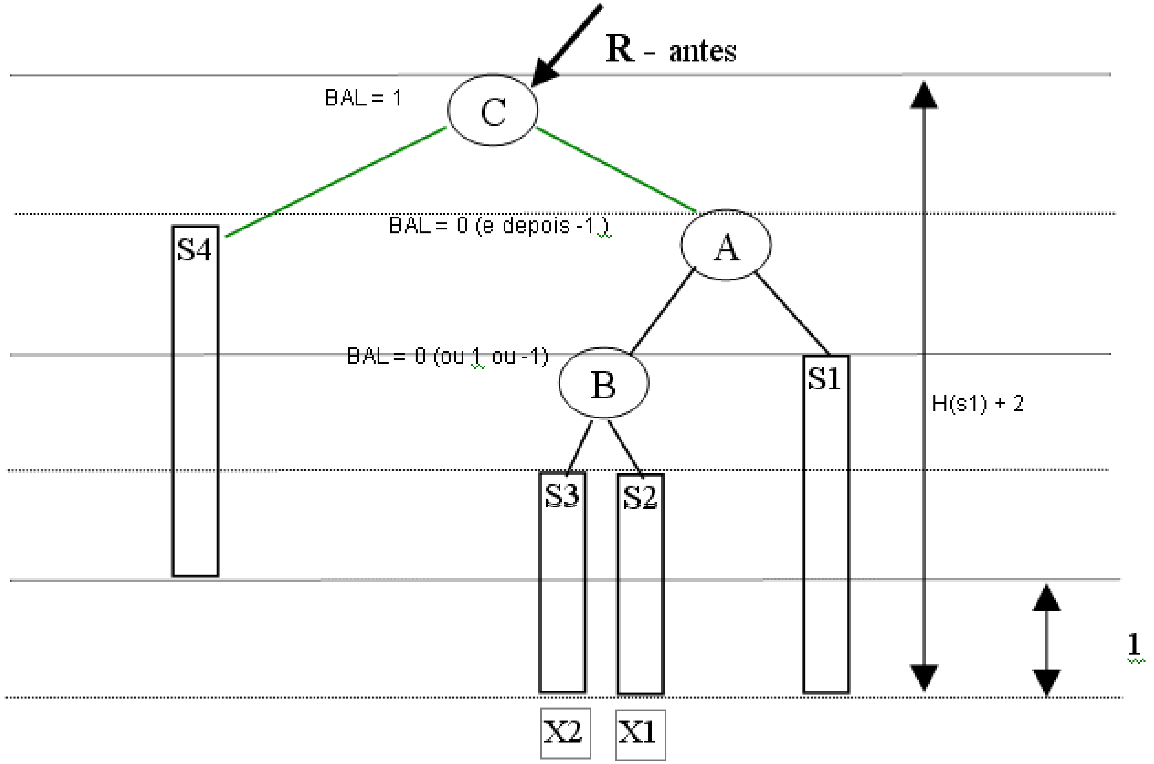
Caso BAL(neto) for
- -1 : BAL(R) = 1   { inseriu X1 }
  BAL(filho) = 0
- +1 : BAL(R) = 0   { inseriu X2 }
  BAL(filho) = -1
- 0 : BAL(R) = 0
  BAL (filho) = 0   { inseriu o próprio B }

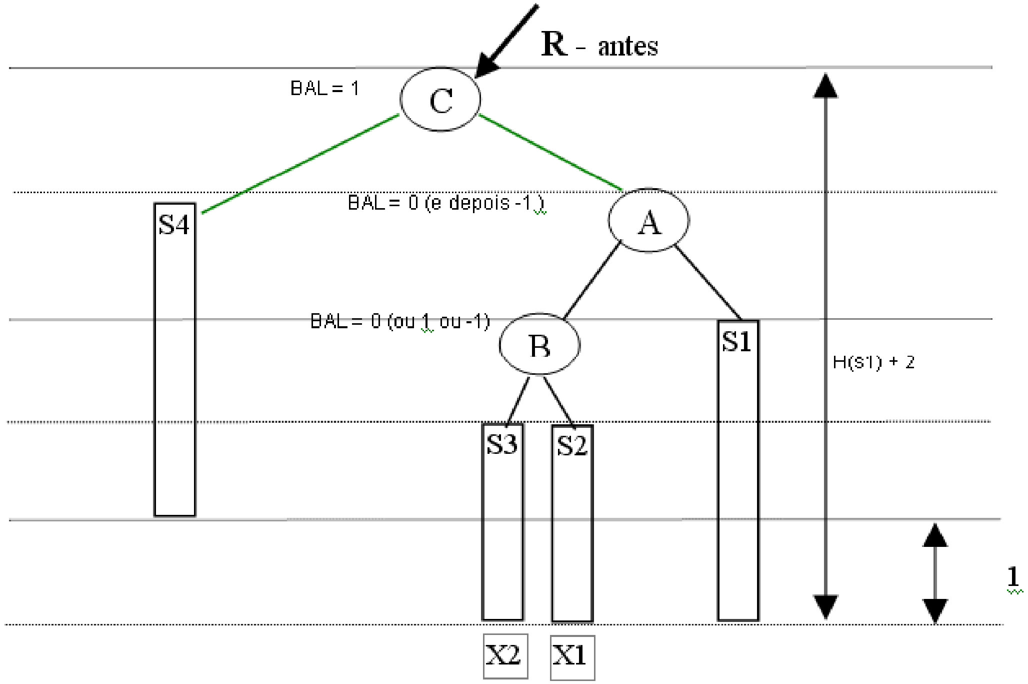
R = neto
BAL (R) = 0   { balanceamento da nova raiz é sempre 0 }
mudou-altura = false

Exercício

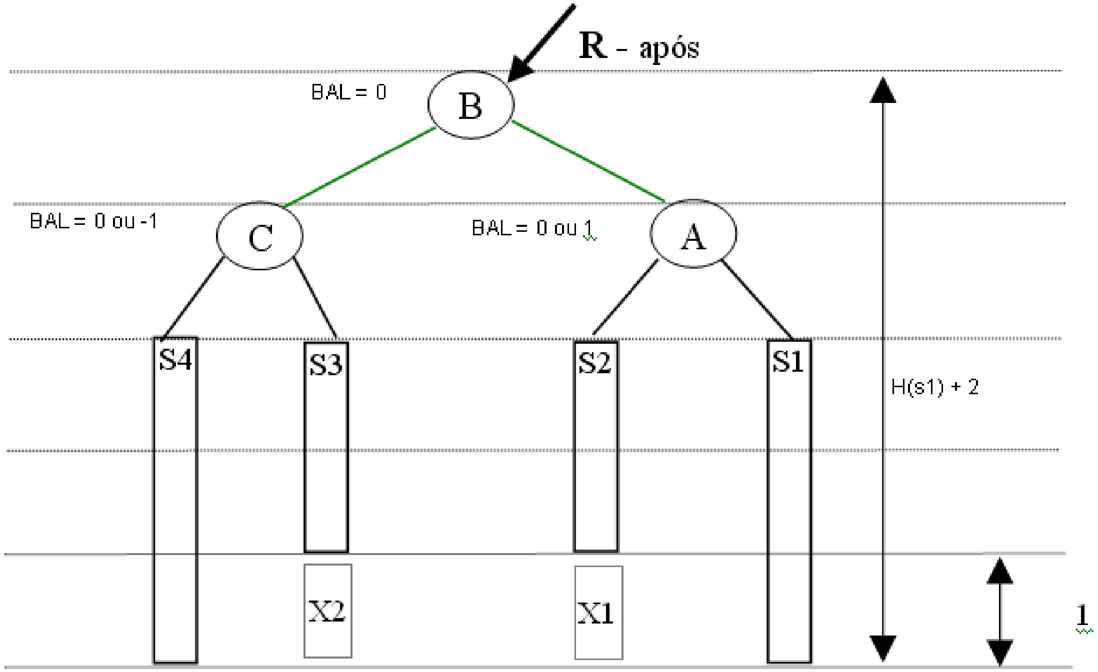
- Identifique exemplos, faça diagramas e o algoritmo de rebalanceamento para o caso DE, ou seja rotação dupla direita-esquerda, caso simétrico ao rotação dupla esquerda-direita.

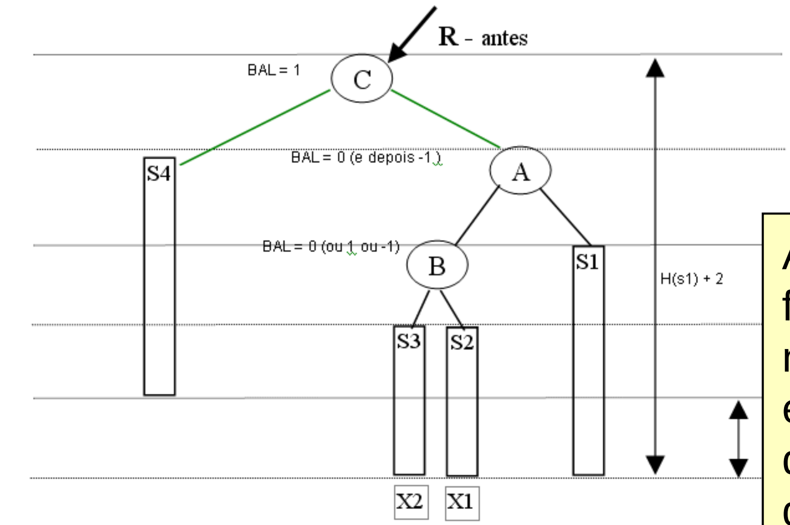
Rotação dupla direita-esquerda – Caso Simétrico



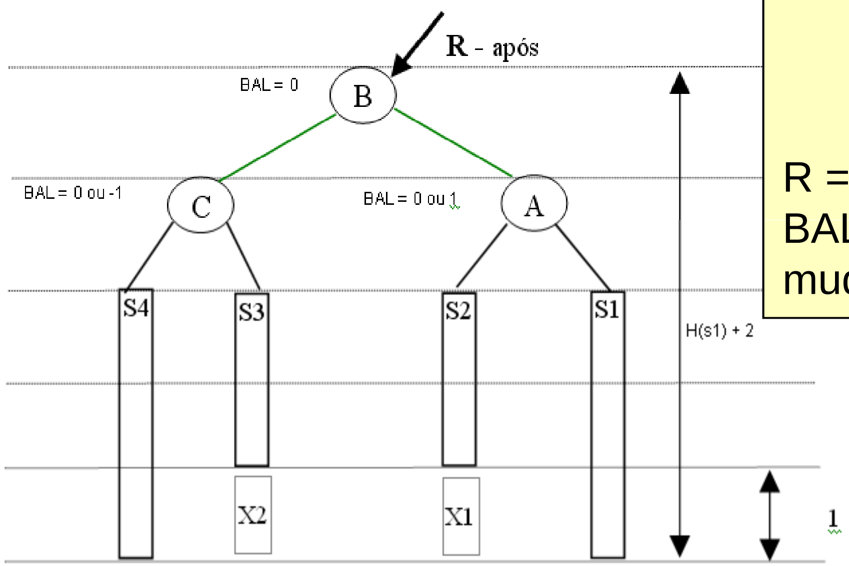


Rotação dupla direita-esquerda





Algorítmo para rotação dupla DE (Insere)
filho = dir(R) { A }
neto = esq(filho) { B }
esq(filho) = dir(neto) { S2 }
dir(neto) = filho { A }
dir(R) = esq(neto) { S3 }
esq(neto) = R { C }
caso BAL(neto) for { note que o algoritmo já executou em neto e em filho }
-1 : BAL(R) = 0 { inseriu X2 }
BAL(filho) = 1
+1 : BAL(R) = -1 { inseriu X1 }
BAL(filho) = 0
0 : BAL(R) = 0
BAL(filho) = 0{ inseriu o próprio B }
R = neto
BAL (R) = 0 { BAL da nova raiz }
mudou-altura = false



Rotação dupla direita-esquerda

Perguntas

- Suponha que para os casos estudados R seja na verdade uma sub-árvore, e que a raiz verdadeira seja outra, identificada por RV. Suponha também que, após inserir um elemento e rebalancear a sub-árvore R, a altura de R não muda (isso ocorre em todos os casos de rebalanceamento vistos até o momento).

- Pergunta 1: existe o risco de essa inserção desbalancear RV? Por que?
- Pergunta 2: terei que me preocupar em ajustar os BALs do resto da árvore (a porção entre R e RV)? Por que?

```
insere( x : char, referência R : ABBB; referência MudouAltura : booleano )
                variáveis filho, neto : ABBB;
se R == null
então           getnode( R )
                info( R ) = x
                dir( R ) = null
                esq( R ) = null
                BAL( R ) = 0
                MudouAltura = true
senão           se x < info( R )
                então           insere( x, esq( R ), MudouAltura )
                                se MudouAltura                          { se mudou altura, verifica o balanceamento }
                                então           caso BAL( R ) for igual a...
                                                  1             BAL( R ) = 0
                                                                MudouAltura = false
                                                  0             BAL( R ) =  -1              { MudouAltura continua true }
                                                 -1             filho = esq( R )            { cresceu a esquerda, que já estava.. }
                                                                se BAL( filho ) =  -1       {    ...grande => precisa rebalancear ! }
                                                                                então ROTAÇÃO SIMPLES EE (do Insere)
                                                                                senão ROTAÇÃO DUPLA ED (do Insere)
                senão           se x > info( R )
                                então           insere( x, dir( R ), MudouAltura )
                                                se MudouAltura
                                                então           caso BAL( R ) for igual a...
                                                                  -1            BAL( R ) = 0
                                                                                MudouAltura = false
                                                                   0            BAL( R ) = 1
                                                                   1            filho = dir( R )
                                                                                se BAL( filho ) = 1
                                                                                                então ROTAÇÃO SIMPLES DD (do Insere)
                                                                                                senão ROTAÇÃO DUPLA DE (do Insere)
                senão           MudouAltura = false                     { não insere pois x já está na árvore }
```

E a remoção?

- Ela desbalanceia a árvore?
- Como seria?

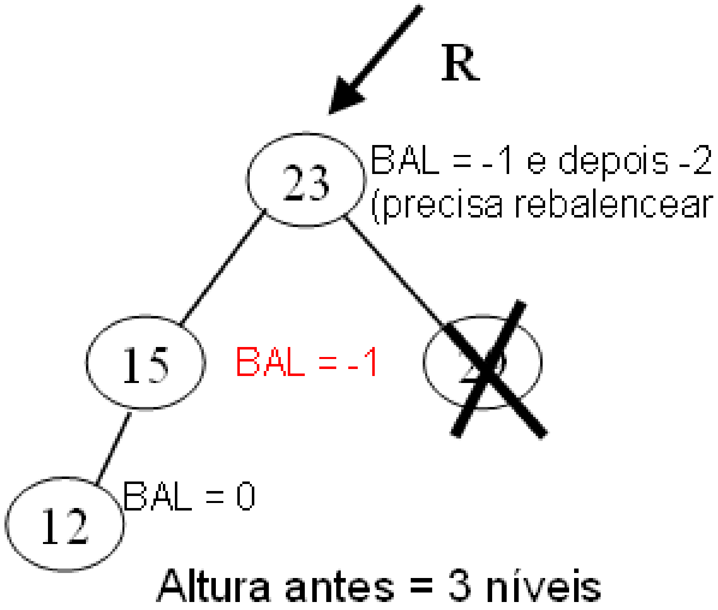
Algoritmo Remove para Árvores Balanceadas

- Os casos de rebalanceamento para o algoritimo Remove são muito parecidos com os casos utilizados na inserção: EE, DD, ED e DE.
- A movimentação dos nós será exatamente como nos casos de rebalanceamento da inserção.
- O que muda então?
  - O ajuste dos BALs será um pouco diferente, e a altura da árvore poderá mudar em alguns casos, mesmo após o rebalanceamento.

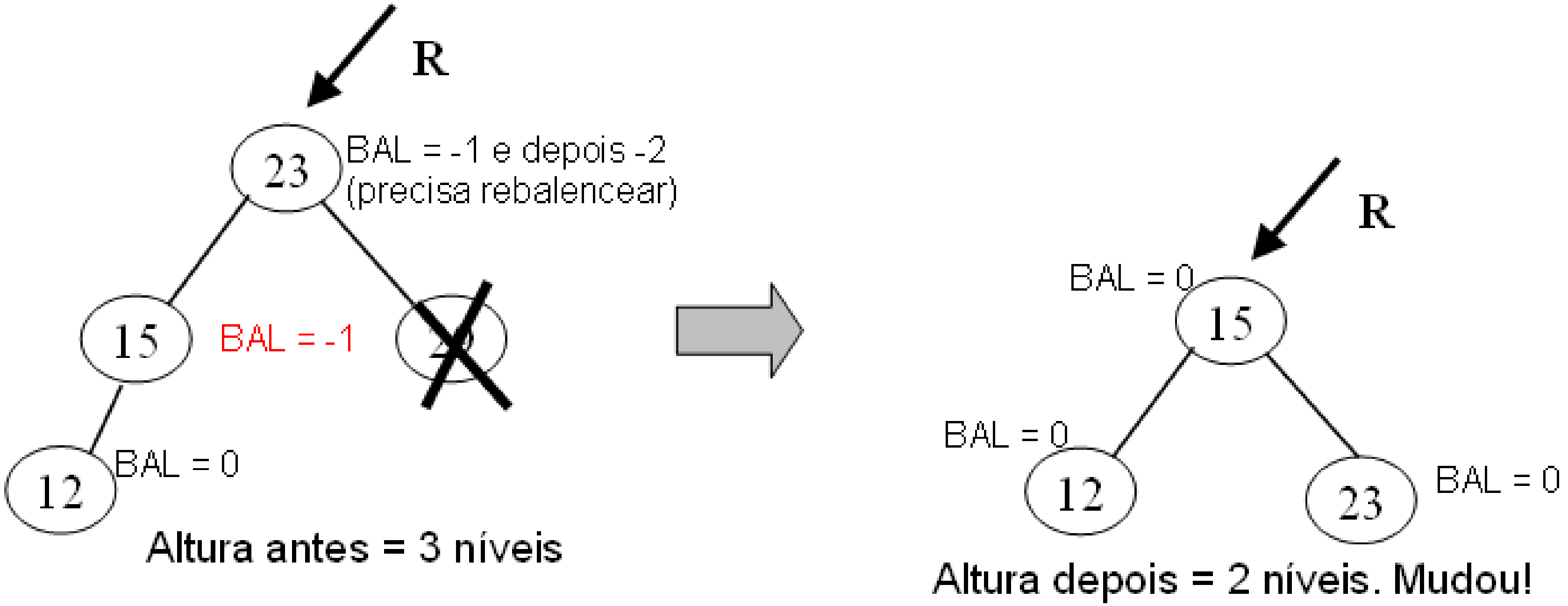
Casos de Rebalanceamento do Remove – Caso EE

- **Exercícios - Aplicar Caso EE do Remove**
  - As árvores abaixo indicam um nó que acabou de ser eliminado. Aplique o caso de rebalanceamento EE: movimente os nós, desenhe a nova árvore, e coloque o valor do BAL de cada nó antes e depois de rebalancear. Verifique se a altura da árvore mudou após a eliminação e rebalanceamento. Faça como no modelo - item (a).

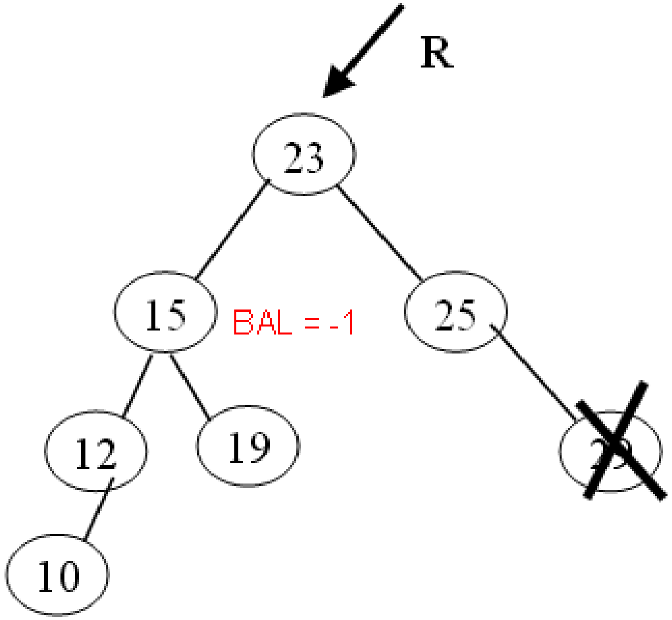
A)


R
23
BAL = -1 e depois -2
(precisa rebalencear
15
BAL = -1
20
12
BAL = 0
Altura antes = 3 níveis

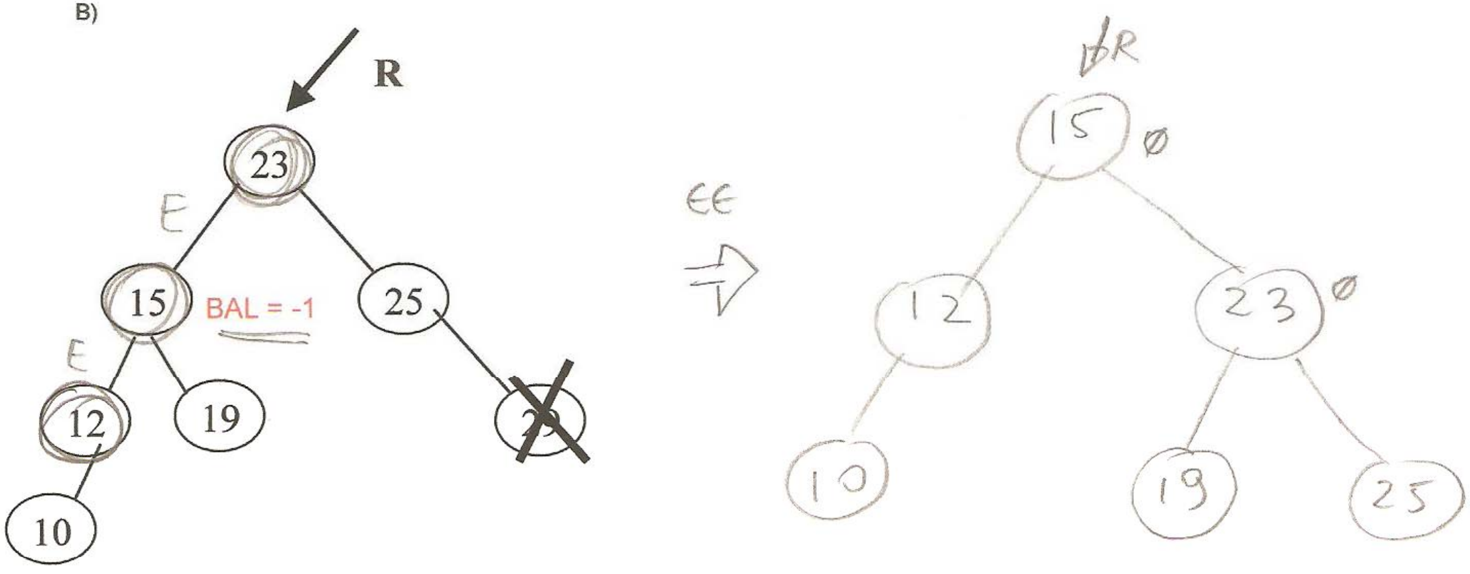
A)


R
23
BAL = -1 e depois -2
(precisa rebalencear)
15
BAL = -1
20
12
BAL = 0
Altura antes = 3 níveis
R
BAL = 0
15
BAL = 0
12
23
BAL = 0
Altura depois = 2 níveis. Mudou!

b)

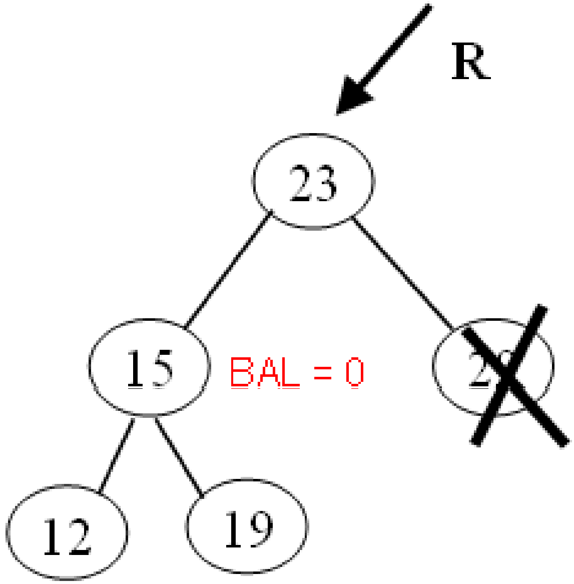


B)

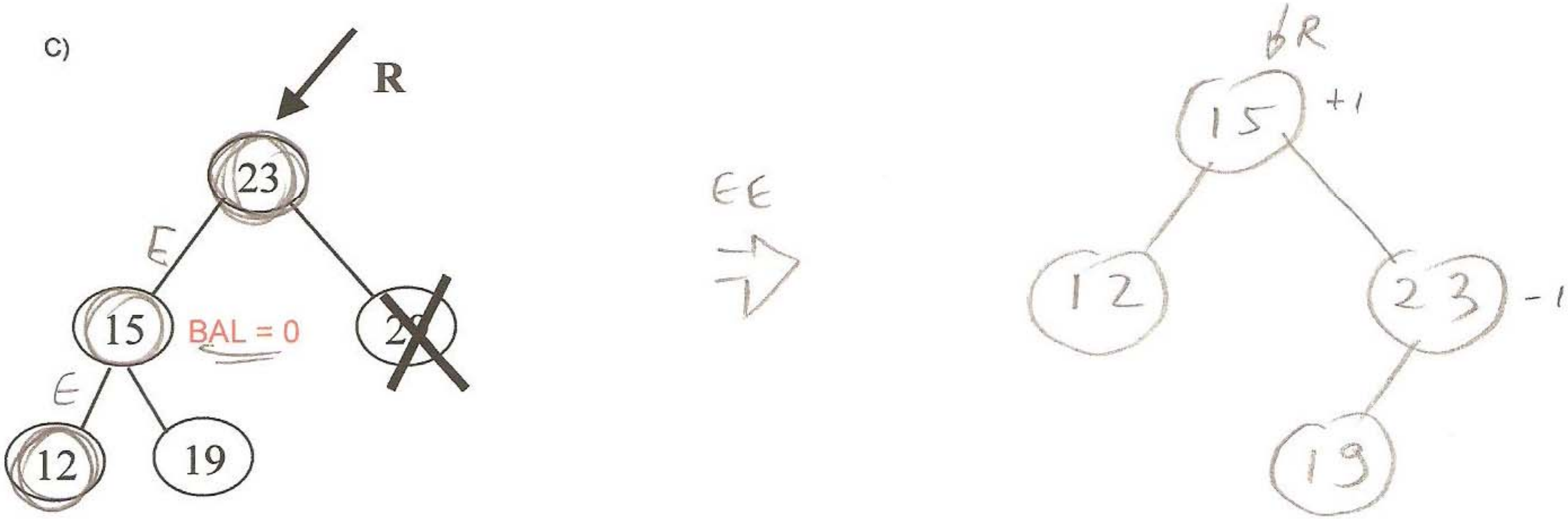


MUDOU ALTURA = TRUE

c)

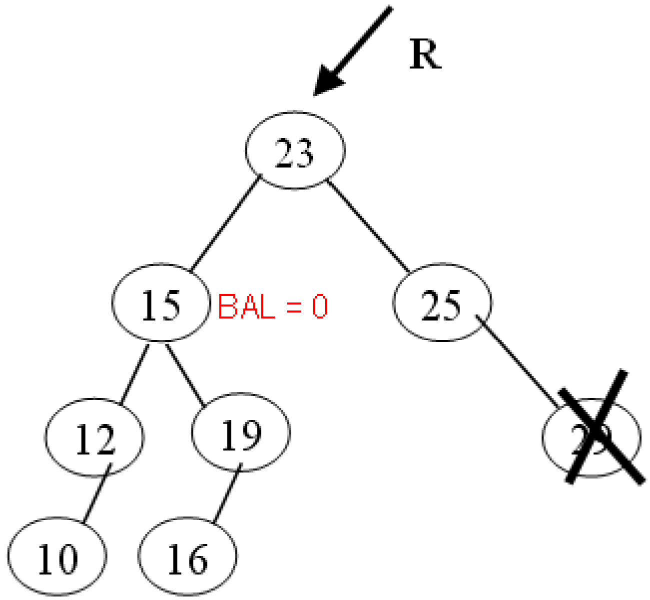


c)

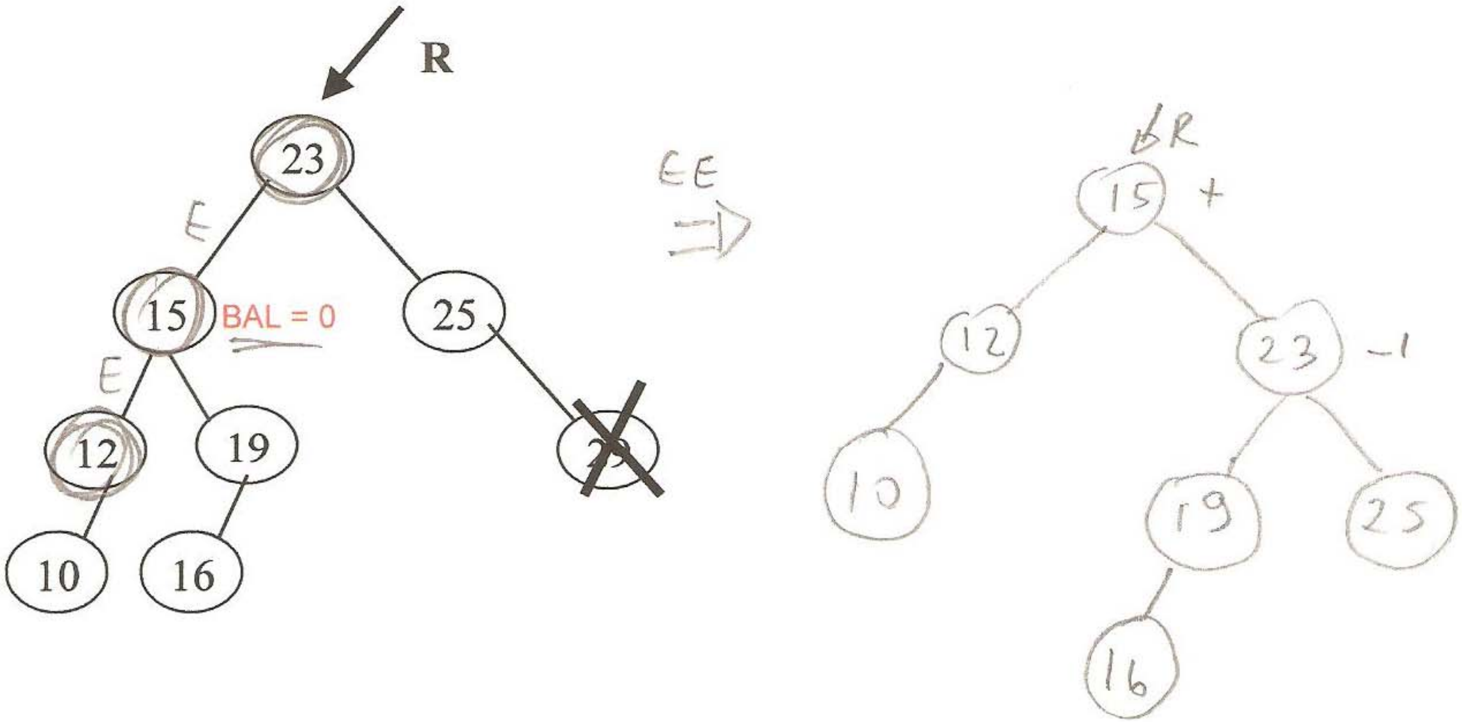


MUDOU ALTURA = FALSE

d)



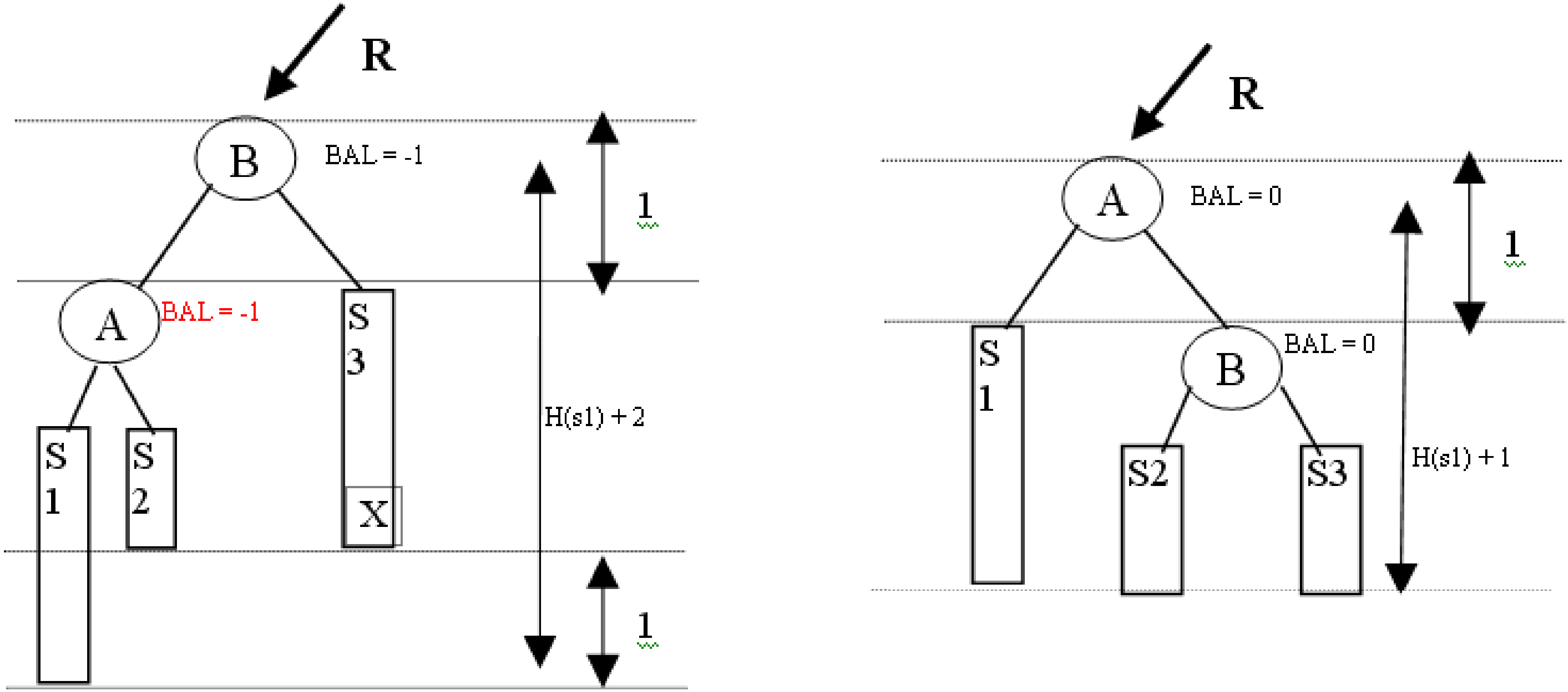
D)



Generalização do Caso EE do Remove

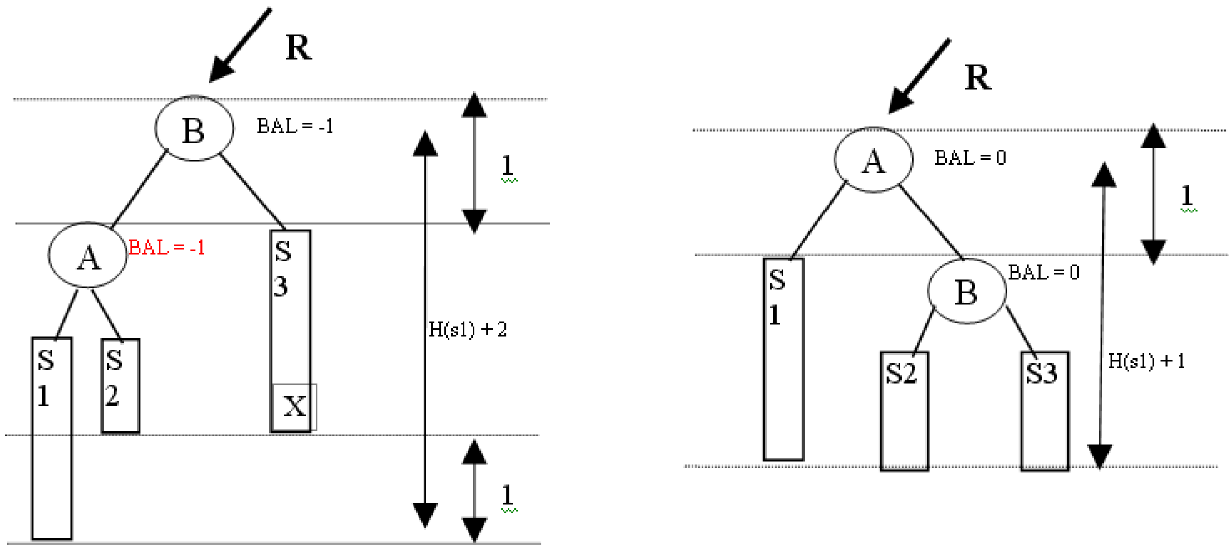
- Dividimos a generalização do caso EE do Remove em dois sub-casos.
- No item (a) e no item (b), o BAL do Filho do nó que desbalanceou (desbalanceou = 23, filho = 15) é -1.
- Já nos itens (c) e (d), o BAL do Filho (15) é zero.
- Nos casos em que o BAL do Filho é -1, a altura da árvore muda; nos casos em que o BAL do Filho é zero a altura da árvore não muda.

EE do Remove Quando BAL do Filho = -1



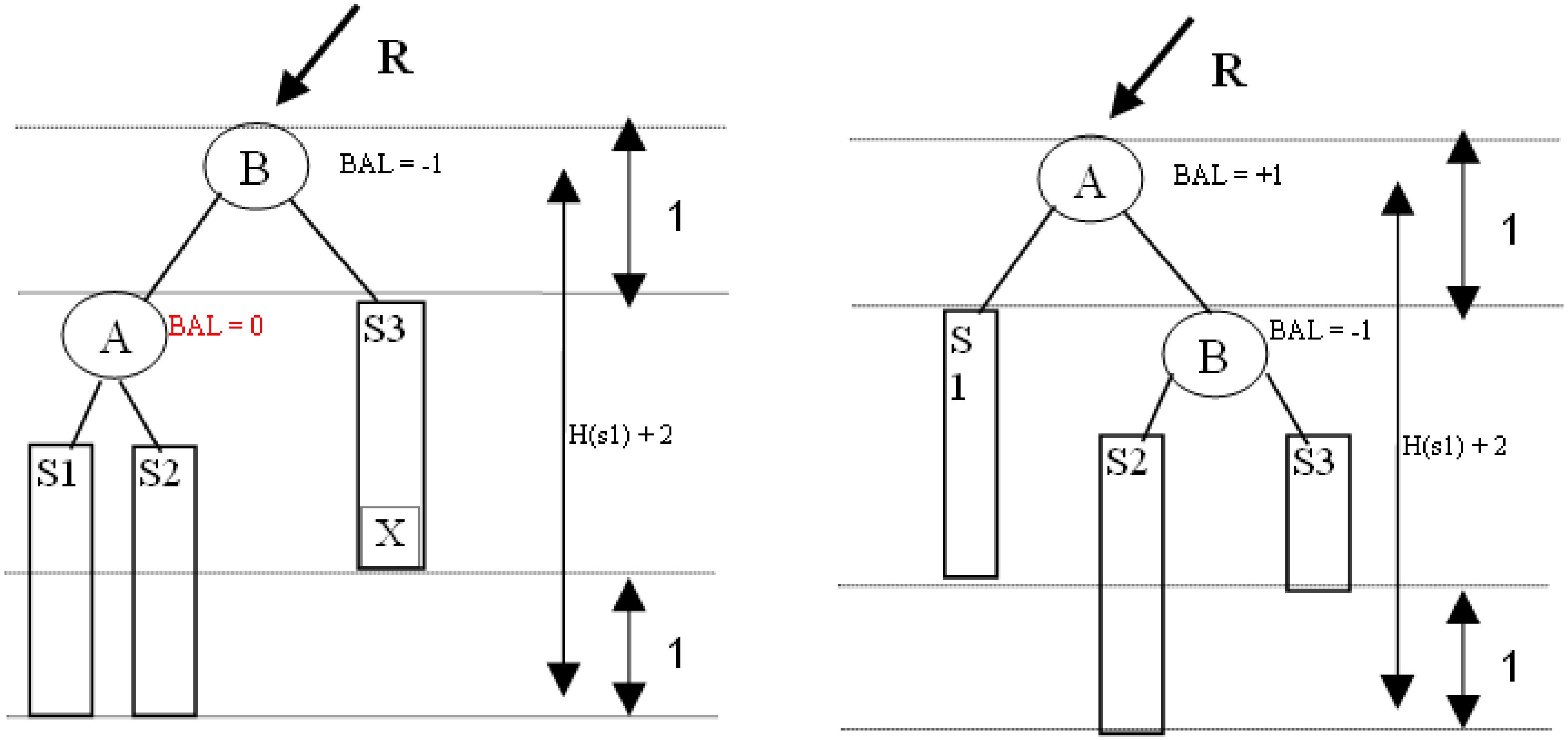
H(s1) = H(s3) = H(s2)+1 // BAL = Hd – He // eliminando X é preciso rebalancear

ALGORITMO { EE Remove quando bal(filho) = -1 }
Filho =esq(R)
SE Bal(Filho) = -1
ENTÃO { início }
Esq(R) =dir (filho)
Dir (filho) =R
Bal (filho) =0
Bal (R) =0
{ mudou altura continua true }
{ fim }



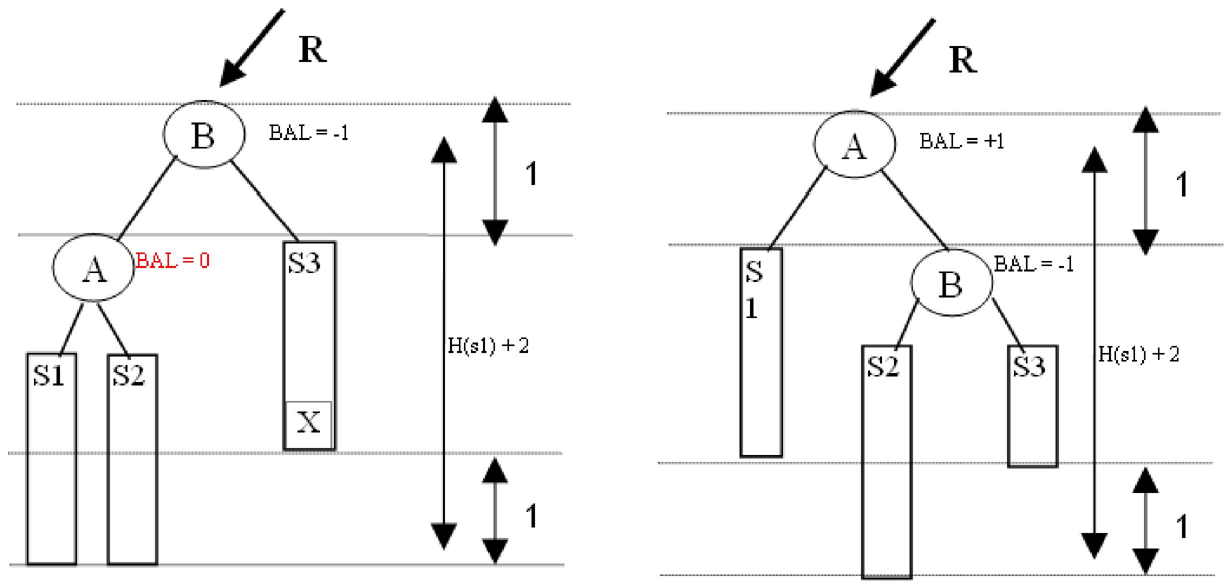
H(s1) = H(s3) = H(s2)+1 // BAL = Hd – He // eliminando X é preciso rebalancear

EE do Remove Quando BAL do Filho = 0



H(s1) = H(s3) = H(s2) // BAL = Hd – He // eliminando X é preciso rebalancear

ALGORITMO { EE Remove quando bal(filho) = 0 }
Filho =esq(R)
SE Bal(Filho) = 0
ENTÃO { início }
Esq(R) =dir (filho)
Dir (filho) =R
Bal (filho) =+1
Bal (R) =-1
MudouAltura =false
{ fim }



H(s1) = H(s3) = H(s2) // BAL = Hd – He // eliminando X é preciso rebalancear

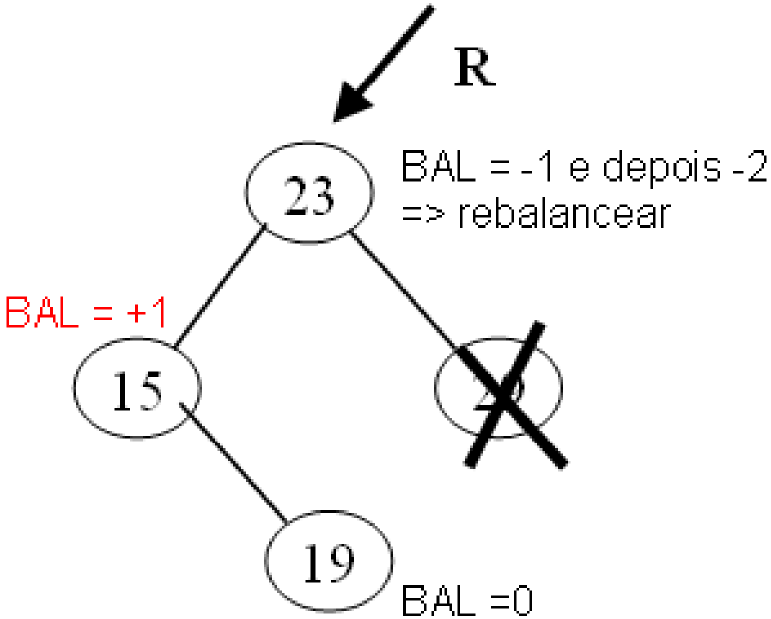
Casos de Balanceamento do Remove – Caso ED

- Quando o BAL do filho for +1, aplicaremos o caso ED do remove.

Casos de Rebalanceamento do Remove – Caso ED

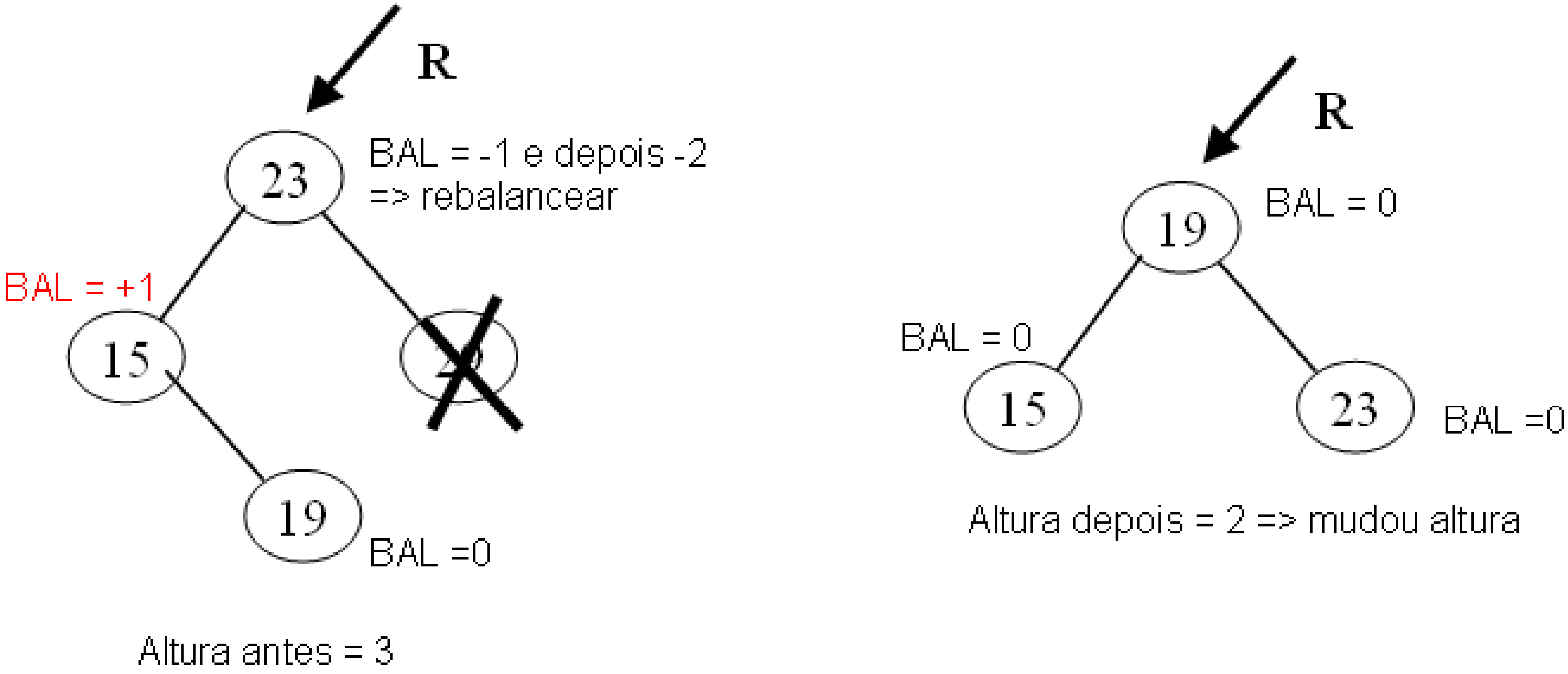
- **Exercício.** As árvores a seguir indicam um nó que acabou de ser eliminado. Aplique o caso de rebalanceamento ED: movimente os nós, desenhe a nova árvore, e coloque o valor do BAL de cada nó antes e depois de rebalancear. Verifique se a altura da árvore mudou após a eliminação e rebalanceamento.

A)

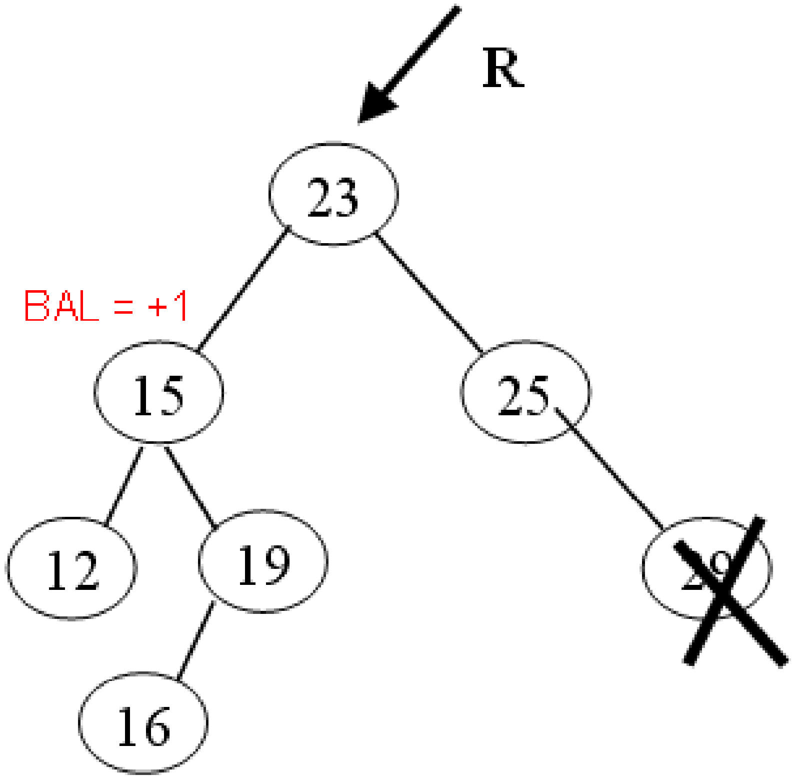


Altura antes = 3

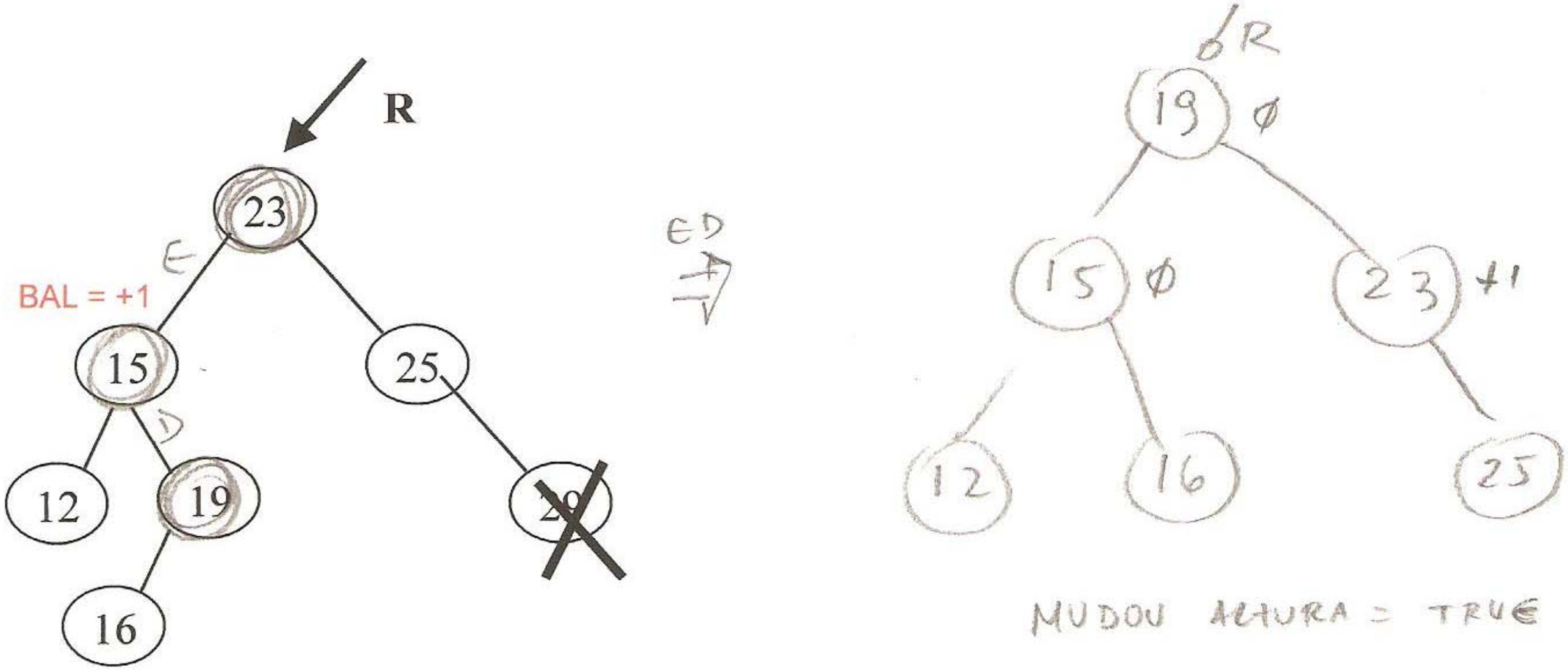
A)


R
23
BAL = -1 e depois -2
=> rebalancear
BAL = +1
15
19
BAL =0
Altura antes = 3
R
19
BAL = 0
BAL = 0
15
23
BAL =0
Altura depois = 2 => mudou altura

B)

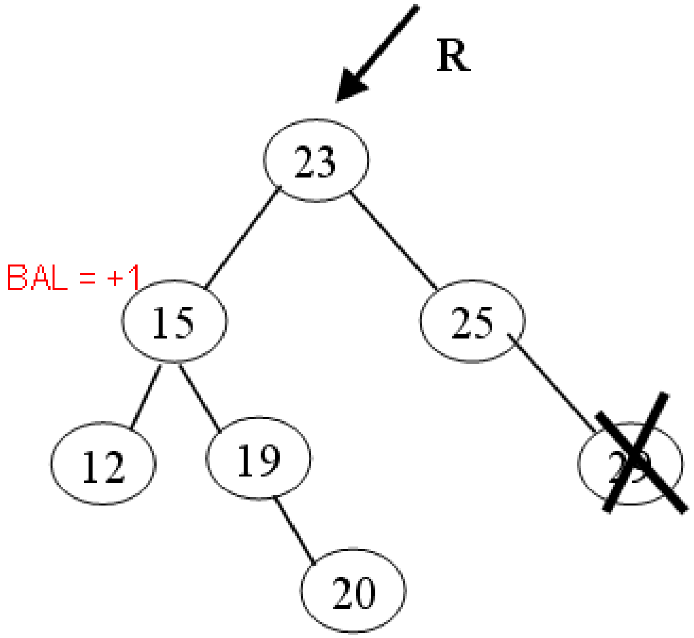


3)

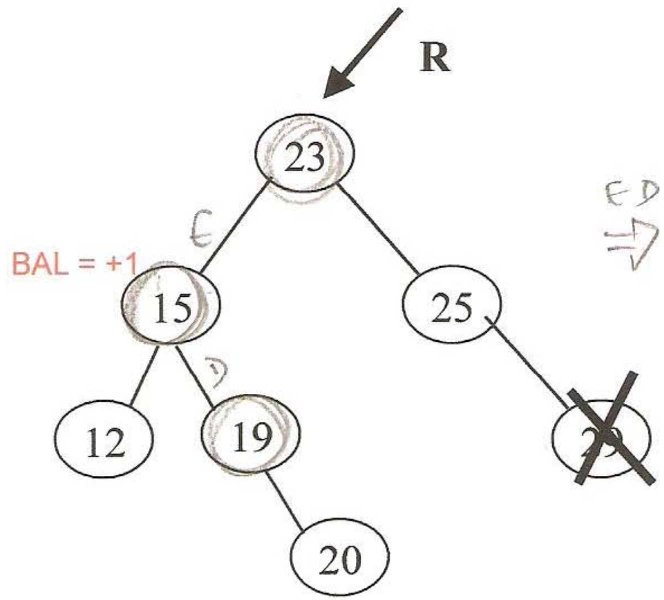


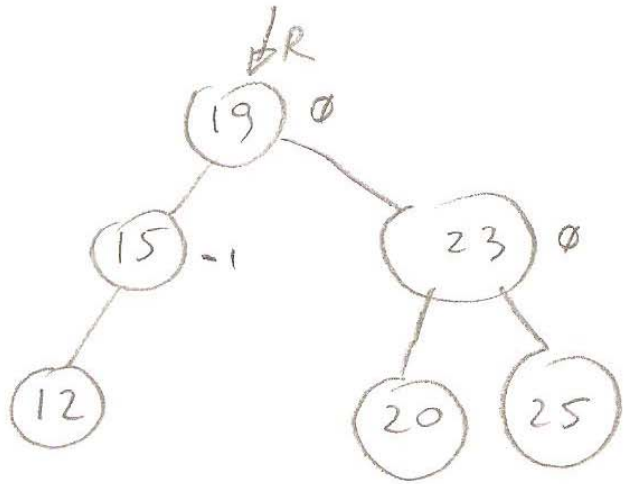
MUDOU ALTURA = TRUE

C)



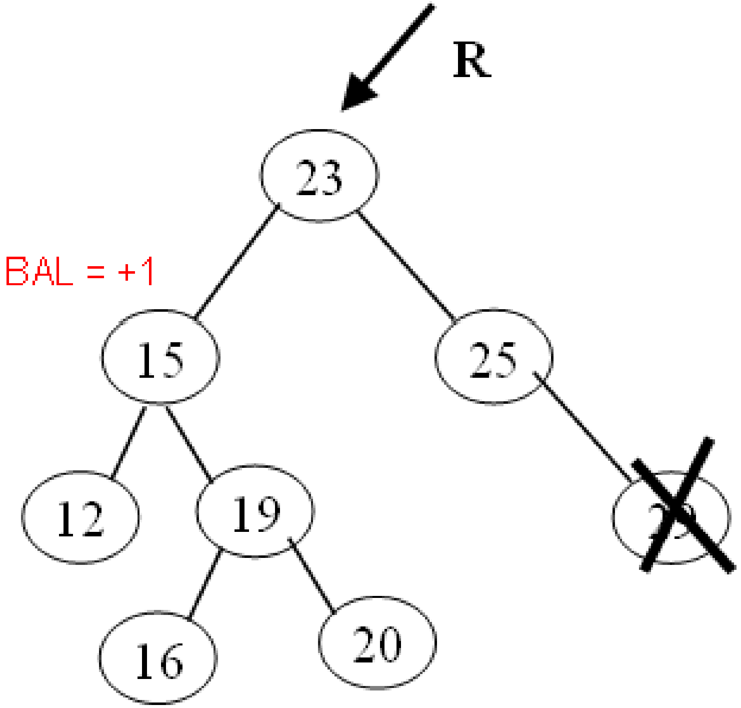
c)



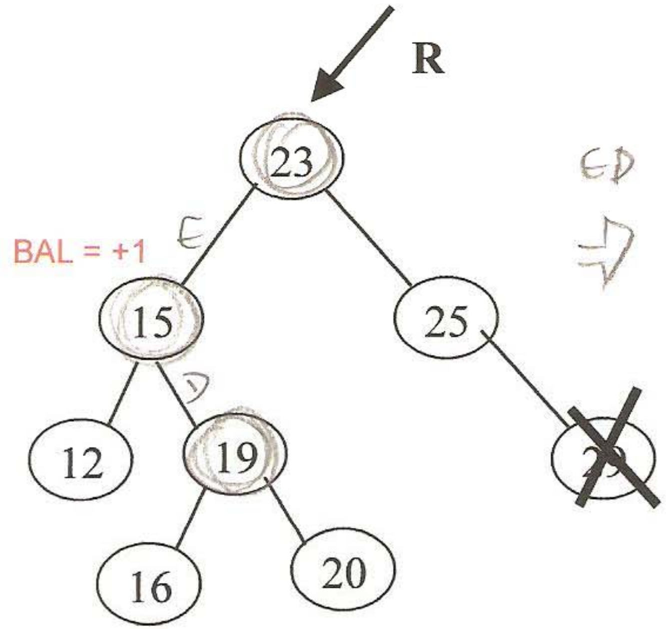


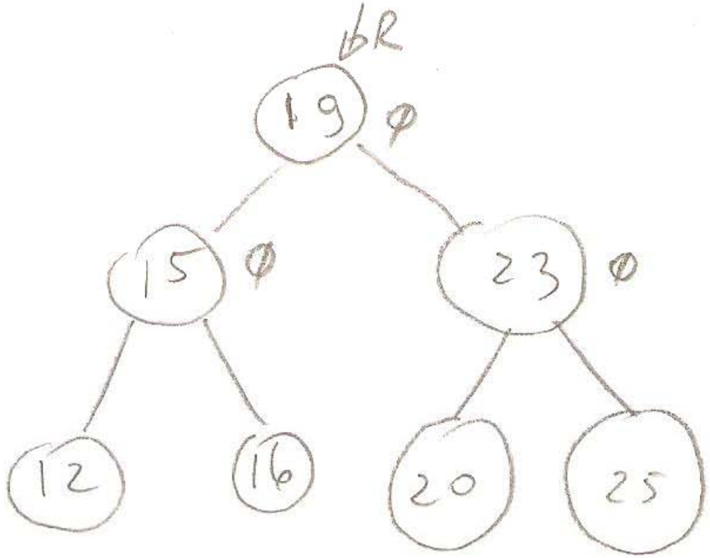
MUDOU ALTURA = TRUE

D)



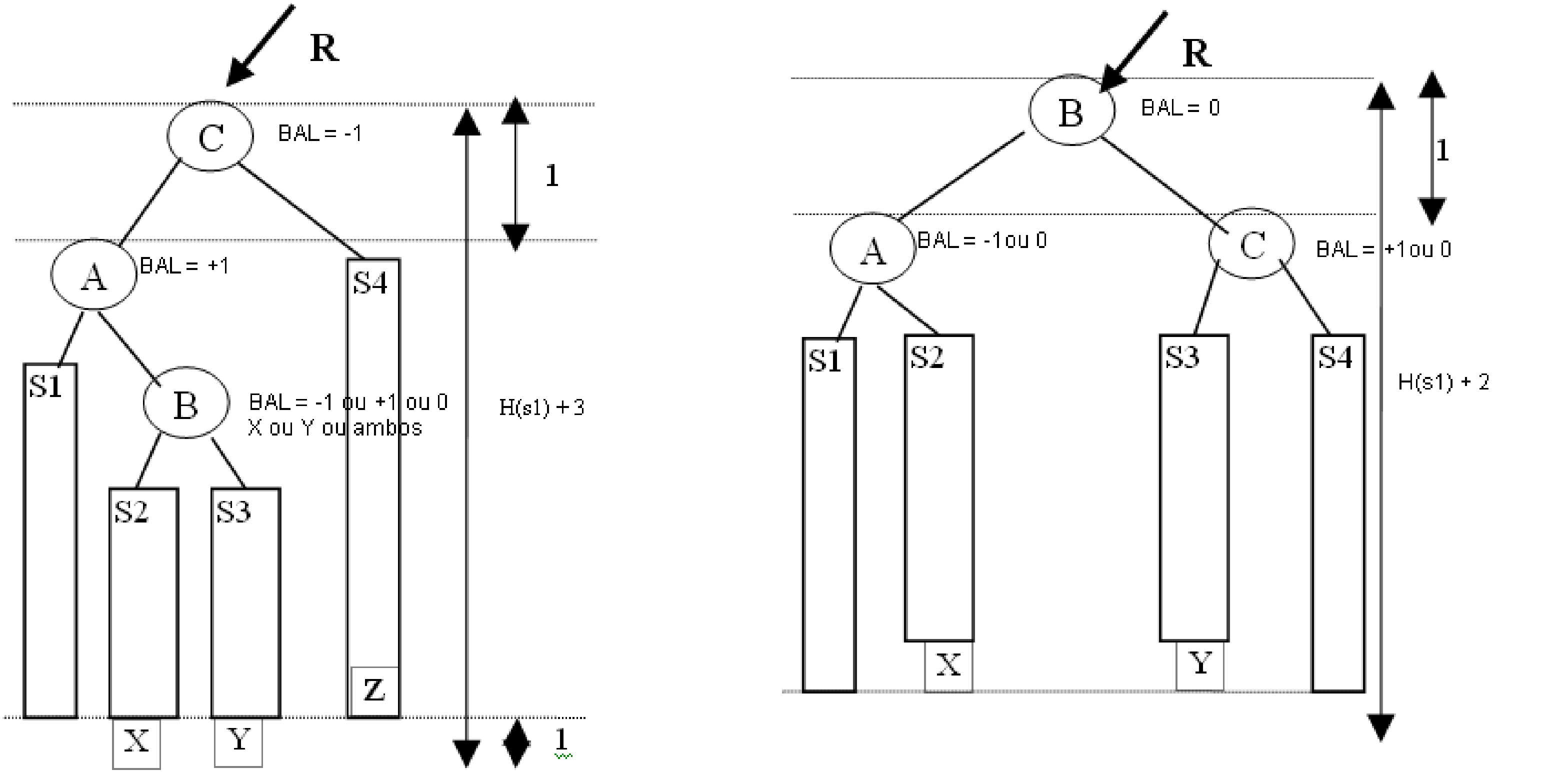
)





MUDOU ALTURA = TRUE

Rebalanceamento ED do Remove



s4=s1+1

Estão na árvore X ou Y ou ambos ➔ s1=s2 OU s1 = s3 OU (S1 = S2 = S3)

Eliminando Z é preciso rebalancear

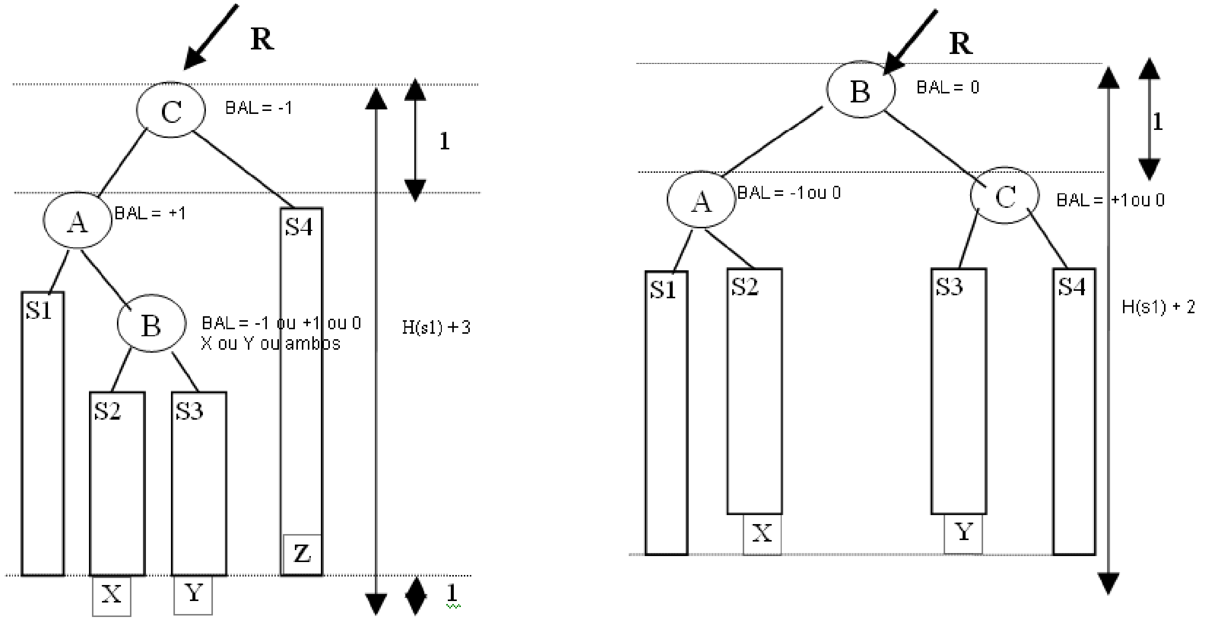
ALGORITMO { ED do Remove }
Filho =esq(R)
Neto =dir (Filho)
Esq(r) =Dir (Neto)
Dir (Neto) =R
Dir (Filho) =esq (Neto)
Esq (Neto) =Filho
Caso Bal (neto) for

0 : bal (R) =0
bal (filho) =0
bal (neto) =0

+1 bal (r) =0
bal (filho) =-1
bal (neto) =0

-1 bal (r) =+1
bal (filho) =0
bal (neto) =0

MudouAltura =True
R =Neto



s4=s1+1
Estão na árvore X ou Y ou ambos ➔ s1=s2 OU s1 = s3 OU (S1 = S2 = S3)
Eliminando Z é preciso rebalancear

Exercícios

1. Fizemos os casos EE e ED do remove. Os casos DD e DE são simétricos. Faça os casos DD e DE do Remove. Identifique casos – exemplo, faça os diagramas de generalização, e os algoritmos.
2. Ajuste o algoritmo remove para Árvore Binária de Busca, garantindo agora que a árvore permanecerá balanceada. Não se esqueça de que em cada nó agora temos um campo BAL. Não se esqueça também que é importante saber se, em uma operação de remoção, a altura da árvore mudou.
3. Implemente uma Árvore Binária de Busca Balanceada.

```
remove( chave tipo  char;  referência R  tipo  ABBB;  referência MudouAltura tipo  booleano)
            variável aux  tipo  ABBB
if raiz == nil
então       MudouAltura = falso
senão       se info( raiz ) == chave
            então       aux = R                                                      { remove efetivamente }
                        se esq( R ) == nil AND dir( R ) == nil                       { caso 1: nenhum filho }
                        então       freenode( aux )
                                    R = nil
                                    MudouAltura = true
                        senão       se dir( R ) <> nil AND esq( R ) <> nil           { caso 3: com dois filhos }
                                    então       info( R ) = maior(  esq( R )  )
                                                remove(  info( R ), esq( R ), MudouAltura )
                                                se MudouAltura
                                                então       {BALANCEAMENTO, ELIMINANDO DA ESQUERDA}
                                    senão       se esq( R ) == nil                               { caso 2: 1 único
filho }
                                                então
                                                            R = dir( R )                  { puxa o filho direito }
                                                senão       R = esq( R )                    { puxa o esquerdo }
                                                freenode( aux )
                                                MudouAltura = true
            senão       se chave > info( raiz )
                        então       remove( chave, dir( raiz ), MudouAltura)
                                    {MONITORAR BALANCEAMENTO, ELIMINANDO DA DIREITA}
                        senão        remove( chave, esq( raiz ), MudouAltura)
                                    { MONITORAR BALANCEAMENTO, ELIMINANDO DA ESQUERDA}
```

```
{ MONITORAR BALANCEAMENTO, ELIMINANDO DA ESQUERDA}
se MudouAltura
então     caso BAL( R ) for igual a...
           -1        BAL( R ) = 0                  { MudouAltura continua true }
            0        BAL( R ) =  1
                     MudouAltura = false
            1        filho = dir( R )              { diminuiu a esquerda; a direita já estava
grande... }
                     se BAL( filho ) ==  -1
                     então ROTAÇÃO DUPLA DE (Remove)
                     senão ROTAÇÃO SIMPLES DD (remove) // tratar sub-casos bal(filho)
== 0 e bal(filho) == 1
{ MONITORAR BALANCEAMENTO, ELIMINANDO DA DIREITA}
se MudouAltura
então     caso BAL( R ) for igual a...
            1        BAL( R ) = 0                  { MudouAltura continua true }
            0        BAL( R ) =  -1
                     MudouAltura = false
           -1        filho = esq( R )     { diminuiu a direita; a esquerda já estava grande.. }
                     se BAL( filho ) ==  1
                     então ROTAÇÃO DUPLA ED (Remove)
                     senão ROTAÇÃO SIMPLES EE (Remove)  // tratar sub-casos bal(filho)
== 0 e bal(filho) == -1
```

```
função maior( A  tipo  ABB ) resultado char
while dir( A ) <> nil do
      A = dir( A )
resultado = info( A )
```