unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Bauru

Faculdade de Ciências

Disciplina: Estruturas de Dados I – **ED1**
Professora: Simone das Graças Domingues Prado
e-mail: simone.prado@unesp.br

Apostila 04 - Árvores Balanceadas (AVL)

Objetivos:

⇒ Trabalhar com árvores binárias de busca balanceadas
⇒ Implementar a inserção numa árvore AVL
⇒ Implementar a remoção numa árvore AVL

Conteúdo:

1. Conceitos
2. Inserção de um nó na Árvore Balanceada
3. Remoção de um nó na Árvore Balanceada
4. Exercícios

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

1. Conceitos

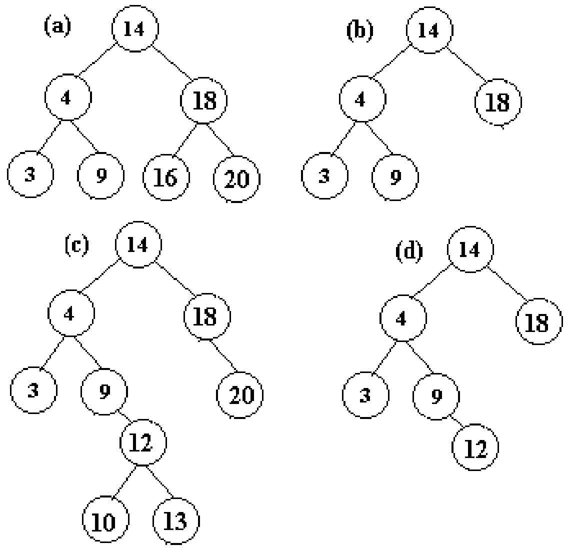
Segundo Wirth (1986,p.187):

> "Uma árvore é considerada balanceada se e somente se para qualquer nó, a **altura** de suas duas subárvores diferem de no máximo **uma unidade**."

As árvores balanceadas são também chamadas de **árvores AVL** em homenagem aos seus inventores: **Adelson-Velskii e Landis**.

Veja alguns exemplos de árvores balanceadas (a e b) e não balanceadas (c e d) na Figura 01.

Figura 01. Árvores balanceadas e não balanceadas



Analisando as quatro árvores (figura 01) nota-se que, para o nó que possui o valor 14, a diferença da altura das subárvore direita e esquerda é igual a 0 na árvore (a) e de 1 na árvore (b). Em (a) podemos ainda dizer que a árvore é perfeitamente balanceada, já que não há diferença entre as alturas da subárvore direita e esquerda.

Na árvore (c), para o nó que possui o valor 14, a altura da subárvore esquerda é 4 e a altura da subárvore direita é 2. Portanto, a diferença é 2 e conclui-se que ela não está balanceada. O mesmo ocorre com a árvore (d) para o nó que possui o valor 14. A subárvore esquerda tem altura 3 e a direita, 1 resultando numa diferença de 2.

Como o critério para balanceamento é a altura do nó, essas árvores binárias de busca balanceadas são chamadas **árvores balanceadas pela altura**. Existem outros métodos, veja em Tenenbaum et al.(1995, p.534-535). Ele cita a árvore binária balanceada por Tarjan e a árvore balanceada pelo peso.

2. Inserção de um nó na Árvore Balanceada

A inserção de um valor X exigirá mais do que verificar se este já existe na árvore e se deverá ocupar o lado direito ou esquerdo de seu pai. Deve-se, também, manter o balanceamento após a inserção.

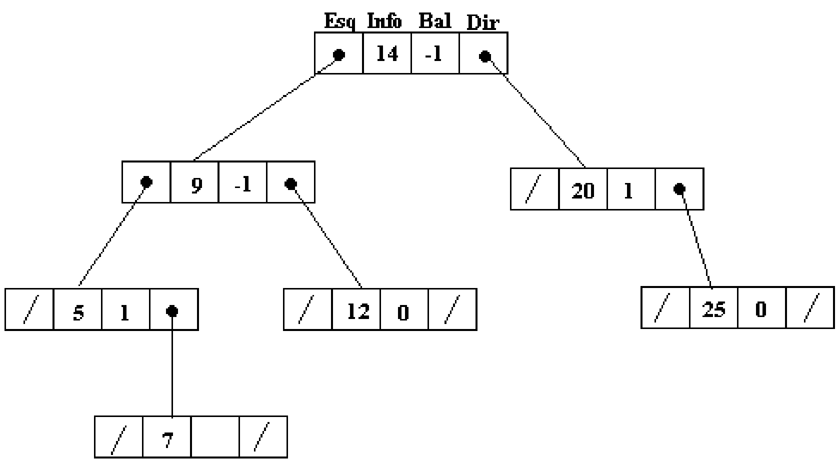
Essa manutenção da árvore balanceada, após inserção de valores, se faz por meio de rebalanceamento (ou restabelecimento da regulagem de seus nós). A ideia consiste em verificar, se a inserção desregulou algum nó, ou seja, se desbalanceou a árvore.

Como o balanço de cada nó deverá ser sabido a cada momento, essa informação estará dentro do nó. Veja a nova definição da árvore:

```
typedef struct no_arvore{
            int info;
            int balanco;
            struct no_arvore* esquerdo;
            struct no_arvore* direito;
            } *def_arvore;
```

Na figura 02 veja uma árvore balanceada com os nós já representando o tipo definido.

Figura 02. Árvore balanceada com os balanços em cada nó



O cálculo do balanço é feito pela seguinte fórmula:

$$\textbf{Balanço (p)} = H_D(p) - H_E(p)$$

Onde:

- H_D (p) é a altura da subárvore direita do nó p e
- H_E (p) é a altura da subárvore esquerda do nó p.

Nessa árvore, ao verificar o balanço em cada nó, percebe-se que os valores podem ser 1, 0 ou -1.

Observe a Figura 02.
Se a inserção de um elemento cair:

- à esquerda do nó que contem a informação 5 (Balanço(5) = 1), o balanço após a inserção será 0.
- à direita ou esquerda do nó que contem a informação 12 (Balanço(12)=0), o balanço será 1 ou -1 e o balanço do nó que contem a informação 9 (Balanço(9)=-1) passará para zero.
- à esquerda do nó que contem a informação 20 (Balanço(20) = 1), o balanço após a inserção será 0.

Portanto, não haverá desbalanceamento da árvore.

Se a inserção de um elemento cair nas situações abaixo, haverá um desbalanceamento.

- abaixo do nó que contem a informação 7 (os nós 9 e 14 ficam desbalanceados)
- abaixo do nó que contem a informação 25 (o nó 20 fica desbalanceado)

Constata-se facilmente que a árvore se torna desbalanceada apenas se o nó recém-inserido é um descendente esquerdo de um nó que tinha anteriormente um balanceamento –1 (já que a subárvore esquerda já era maior e ficou maior ainda). O mesmo acontece se o nó recém-inserido é um descendente direito de um nó que tinha anteriormente um balanceamento de 1.

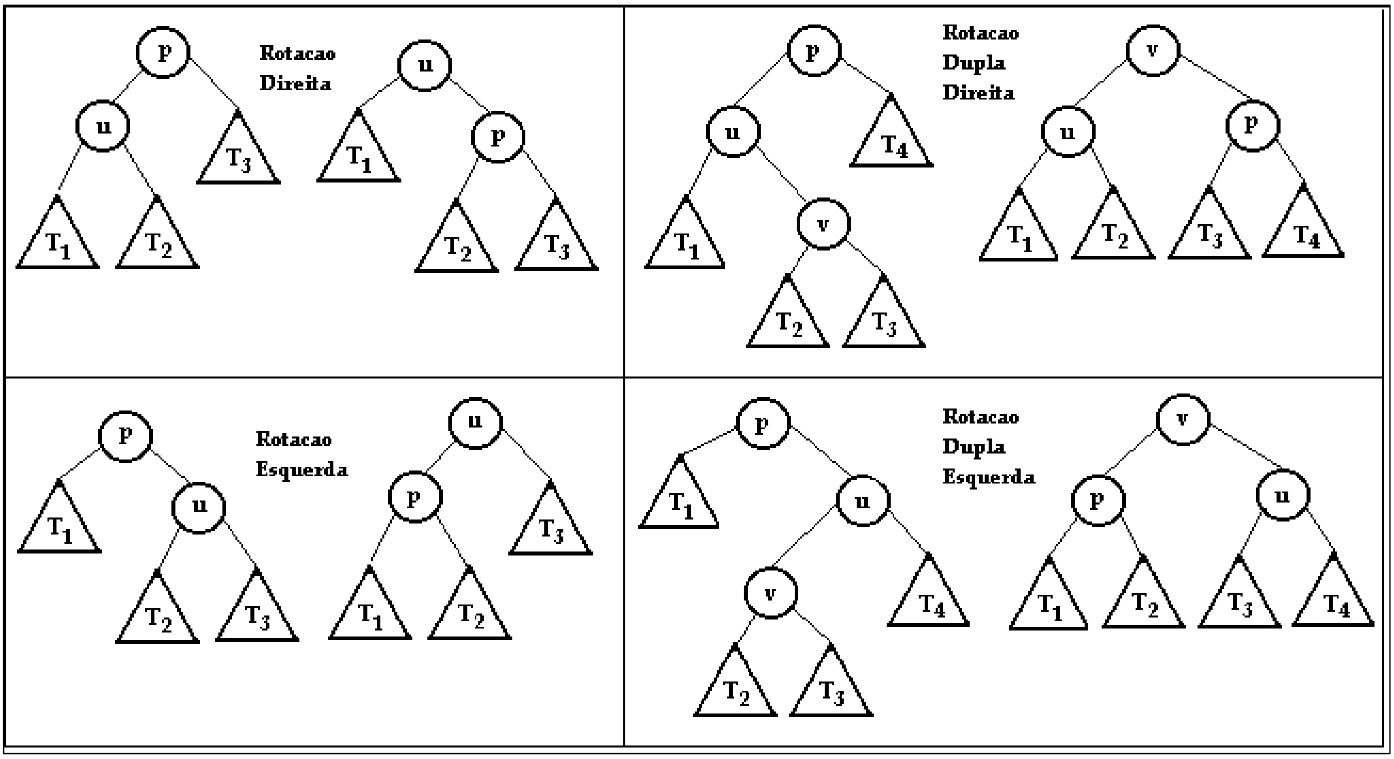
Para manter a árvore balanceada, é necessário fazer uma transformação na árvore de forma que a árvore continue sendo uma árvore binária de busca e balanceada. Para tal são usadas quatro transformações (Figura 3):

- rotação direita,
- rotação esquerda,
- rotação dupla direita e
- rotação dupla esquerda.

As subárvores T1 a T4 que aparecem na Figura 03 podem ser vazias ou não. O nó p é chamado raiz da transformação.

Swarcfiter & Markenzon (1994,p.134-137) provam que essas quatro transformações cobrem todas as possibilidades para o balanceamento da árvore de busca após uma inserção.

Figura 03. As transformações: (a)rotação direita; (b) rotação esquerda;
(c) rotação dupla direita; (d) rotação dupla esquerda



Veja como implementar a árvore balanceada pela altura. A ideia geral, a seguir, foi baseada em Swarcfiter & Markenzon (1994,p.139-144):

(1) Verifica se x (o valor a ser inserido) está na árvore
(2) Se sim,
  (2.1) encerra.
(3) Se não,
  (3.1) Insere o valor x no local correto na árvore binária
  (3.2) Verifica se surgiu algum nó desregulado
  (3.3) Se sim,
    (3.3.1) faça o rebalanceamento
    (3.3.2) encerra.
  (3.4) Se não encerra.

Para verificar se o nó está desregulado temos que calcular o balanço para cada nó. O nó p só está regulado se **-1 ≤ balanço (p) ≤ 1.**

Suponha que q será inserido abaixo do nó p.
- Se **q** pertencer à subárvore esquerda de **p** e essa inserção aumentar a altura dessa subárvore, então deve-se subtrair 1 do balanço(**p**).

- Se **q** ao ser inserido na subárvore direita aumentar a altura da subárvore, deve-se somar 1 ao balanço (**p**).
- Se o valor de balanço (**p**) ultrapassar a faixa [-1,1], então **p** ficou desregulado após a inserção de **q,** ou seja,

$$| H_D(\mathbf{p}) - H_E(\mathbf{p}) | = 2.$$

Veja abaixo uma análise da operação de inserção:

Caso 1: **H_D (p) < H_E (p) então H(p) = -2**

Então a <u>subárvore</u> <u>esquerda</u> <u>de</u> **p** <u>aumentou</u> ainda mais após a inserção do elemento **q.** Na figura 04 podemos ver em (a) que a inserção do elemento **q** pode ser feita à esquerda (Figura 04(b)) ou pode ser feita à direita (Figura 04(c)) dependendo do valor contido em **q**. Então a partir desse caso (H_D (**p**) < H_E (**p**)) gera-se dois subcasos:

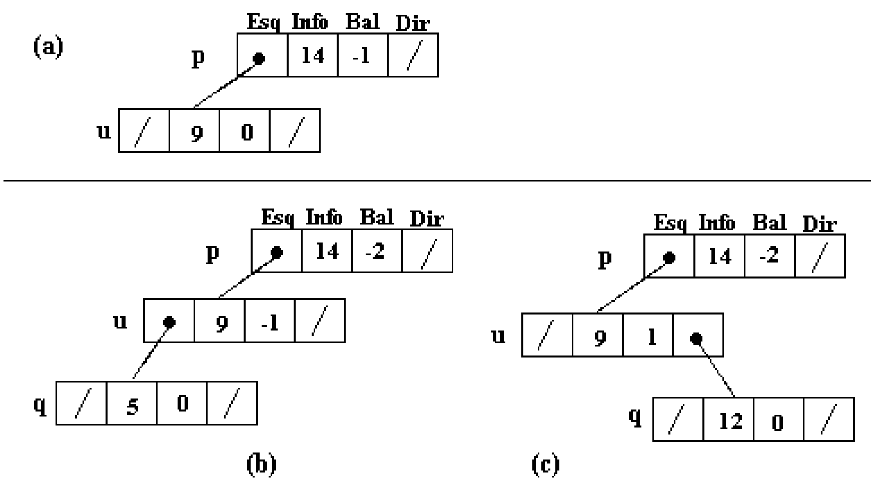


Figura 04. As possibilidades de árvore para o caso 1.

Caso 1.1: **H_D (u) < H_E (u)**, ou seja, houve inserção à esquerda de **u** (Figura04(b)) Olhando as rotações da Figura 03 percebe-se que será necessário fazer uma rotação direita para tornar a árvore balanceada. Veja Figura 05.

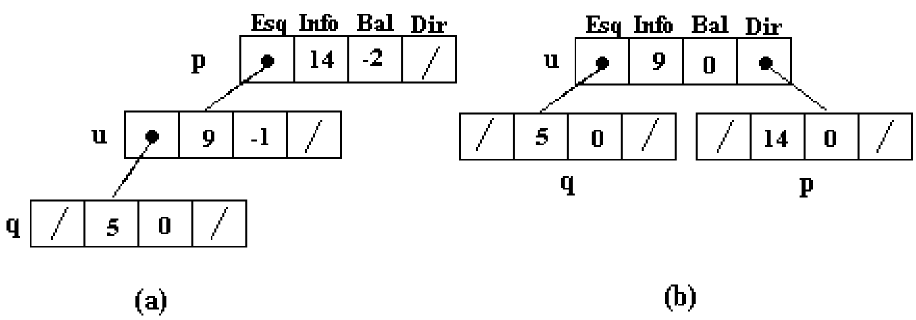


Figura 05. (a) Inserção na subárvore esquerda (b) Árvore após rotação direita.

Caso 1.2: **H_D (u) > H_E (u)**, ou seja, houve inserção à direita de **u** (Figura 04 (c)) Observando as rotações da Figura 03, deve-se fazer uma rotação dupla direita para que a árvore fique balanceada. Veja Figura 06.

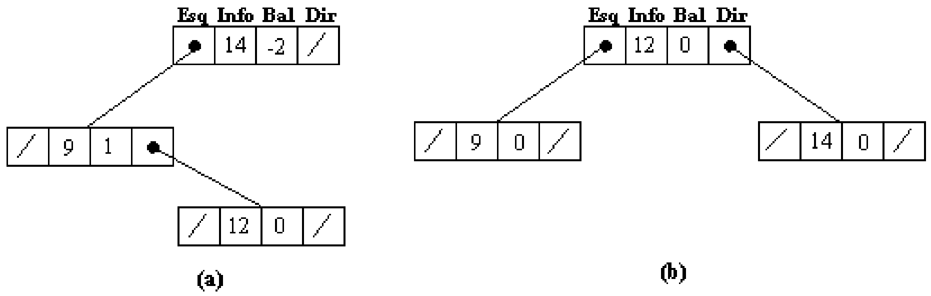


Figura 06. (a) Inserção na subárvore direita (c) Árvore após rotação dupla direita.

Caso 2: **H_D (p) > H_E (p) então H(p) = 2**

A <u>subárvore</u> <u>direita</u> <u>aumentou</u> ainda mais após a inserção do elemento **q**.

Na figura 07(a) a inserção do elemento **q** pode ser feita à esquerda (Figura 07(b)) ou pode ser feita a direita (Figura 07(c)) dependendo do valor contido em **q**. Então surgem os sub-casos:

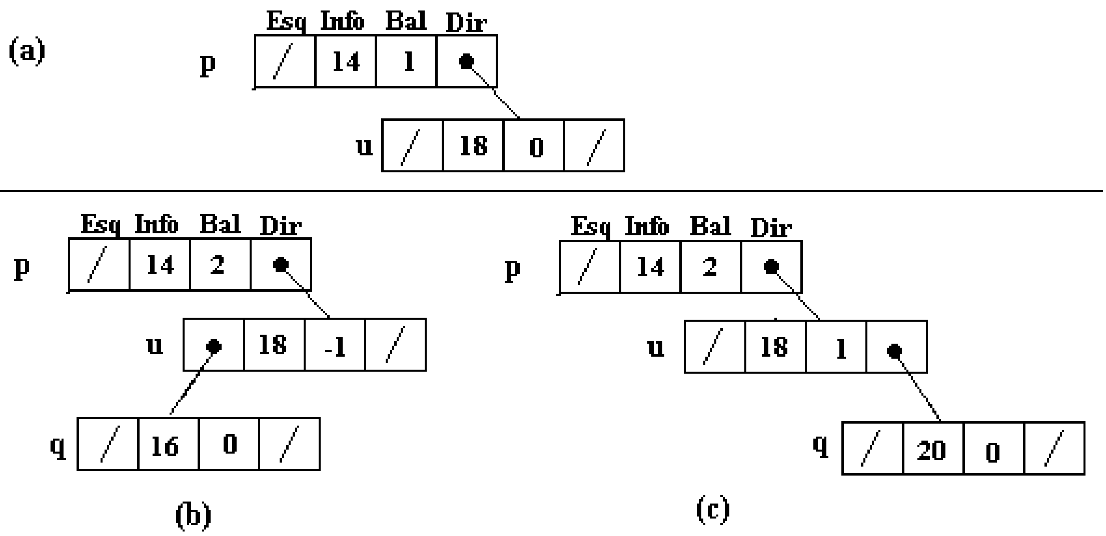
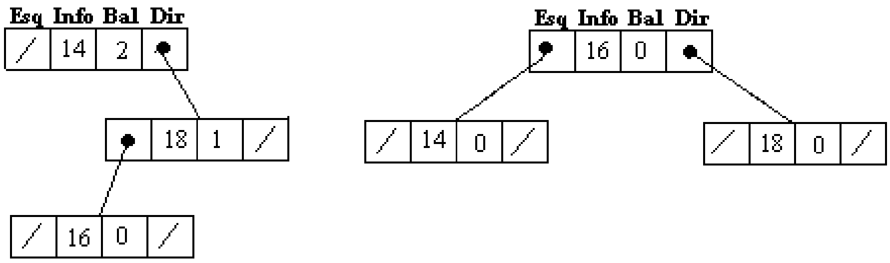


Figura 07. As possibilidades de árvore para o caso 2.

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

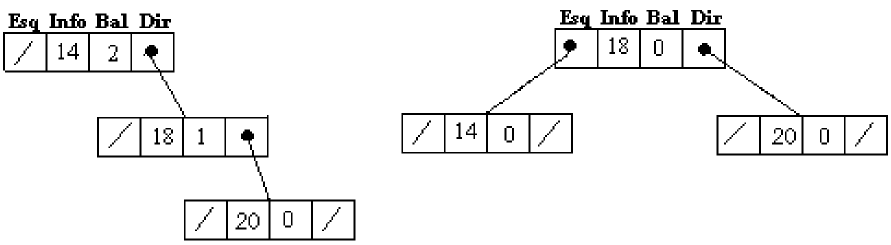
Caso 2.1: **H $_D$ (u) < H $_E$ (u)**, ou seja, houve inserção à esquerda de **u** (Figura07(b)) Observando as rotações da Figura 03 percebe-se que será necessário fazer uma rotação dupla esquerda para tornar a árvore balanceada. Veja Figura 08.



(a) (b)

Figura 08. (a) Inserção na subárvore esquerda (c) Árvore após rotação dupla esquerda.

Caso 2.2: **H $_D$ (u) > H $_E$ (u)**, ou seja, houve inserção à direita de **u** (Figura07(c)) Observando as rotações da Figura 03, deve-se fazer uma rotação esquerda para que a árvore fique balanceada. Veja Figura 09.



(a) (b)

Figura 09. (a) Inserção na subárvore direita (c) Árvore após rotação simples direita.

Assim, a regulagem de p é restabelecida pela aplicação de uma transformação apropriada.

Resumo:

H(**p**) = -1	H(**u**) = 0	Inserção a esquerda de u (H(**p**) = -2, H(**u**) = -1)	Rotação Simples a Direita
	H(**u**) = 0	Inserção a direita de u (H(**p**) = -2, H(**u**) = 1)	Rotação Dupla a Direita
H(**p**) = 1	H(**u**) = 0	Inserção a esquerda de u (H(**p**) = 2, H(**u**) = -1)	Rotação Dupla a Esquerda
	H(**u**) = 0	Inserção a direita de u (H(**p**) = 2, H(**u**) = 1)	Rotação Simples a Esquerda

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

3. Remoção de um nó na Árvore Balanceada

Quando remover um nó numa árvore AVL, é necessário fazer as operações como foram feitas nas árvores binárias de busca e, depois, verificar se a árvore ficou desbalanceada.

A árvore pode ficar desbalanceada quando um nó à esquerda for removido e a remoção desse elemento fizer com que a subárvore esquerda encolha, fazendo com que o lado direito da árvore fique maior, e assim o balanço da árvore passe a valer 2. Quando isso ocorrer tem-se de fazer rotações à esquerda (simples ou dupla) para balancear a árvore novamente. As rotações já foram mostradas na figura 03.

Caso 1: $\mathbf{H_D(p) > H_E(p)}$ **então** $\mathbf{H(p) = 2}$

Então a <u>subárvore esquerda de</u> **p** <u>encolheu</u> e por conseqüência a <u>subárvore direita aumentou</u> ainda mais após a remoção do elemento **q**.

Caso 1.1: $\mathbf{H_D(u) \geq H_E(u)}$, ou seja, $\mathbf{H(u) \geq 0}$ **=> rotação esquerda** (Figura 10)

Caso 1.2: $\mathbf{H_D(u) < H_E(u)}$, ou seja, $\mathbf{H(u) < 0}$ **=> rotação dupla esquerda** (Figura 11)

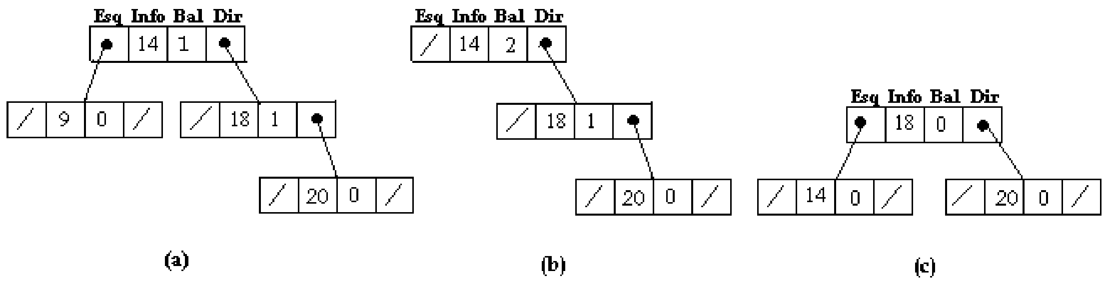


Figura 10. (a) Árvore de onde vai ser removido o nó 9 (b) Árvore após remoção (c) Árvore após rotação esquerda.

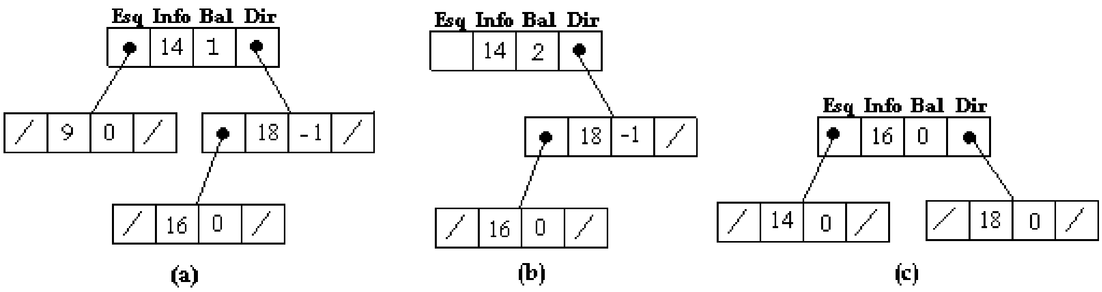


Figura 11. (a) Árvore de onde vai ser removido o nó 9 (b) Árvore após remoção (c) Árvore após rotação dupla esquerda.

A árvore pode ficar desbalanceada quando um nó a direita for removido e a remoção desse elemento fizer com que a subárvore direita encolha, fazendo com que o lado esquerdo da árvore fique maior, e assim o balanço da árvore passe a valer -2. Quando isso ocorrer faz-se rotações à direita (simples ou dupla) para balancear a árvore novamente. As rotações já foram dadas conforme figura 3

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

Caso 2: **H_D (p) < H_E (p) => H(p) = -2**

Então a <u>subárvore</u> <u>direita</u> <u>de</u> **p** <u>encolheu</u> e por conseqüência a <u>subárvore</u> <u>esquerda</u> <u>aumentou</u> ainda mais após a remoção do elemento **q**.

Caso 2.1: **H_D(u) > H_E(u)**, ou seja, **H(u) > 0 => rotação dupla direita** (Figura 12)

Caso 2.2: **H_D(u) ≤ H_E(u)**, ou seja, **H(u) ≤ 0 => rotação direita** (Figura 13)

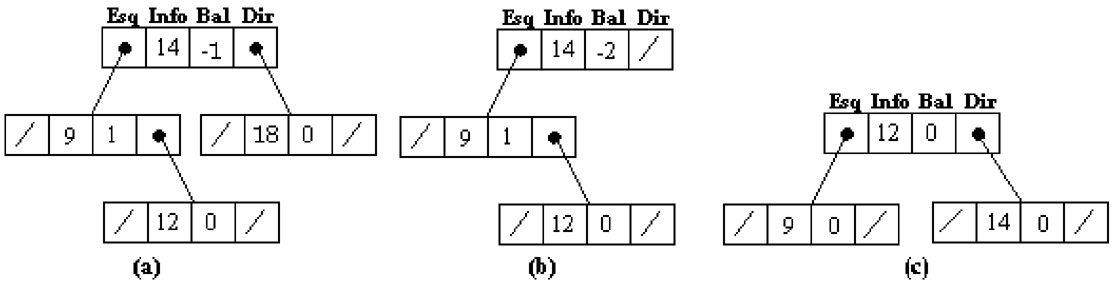


Figura 12. (a) Árvore de onde vai ser removido o nó 18 (b) Árvore após remoção (c) Árvore após rotação dupla direita.

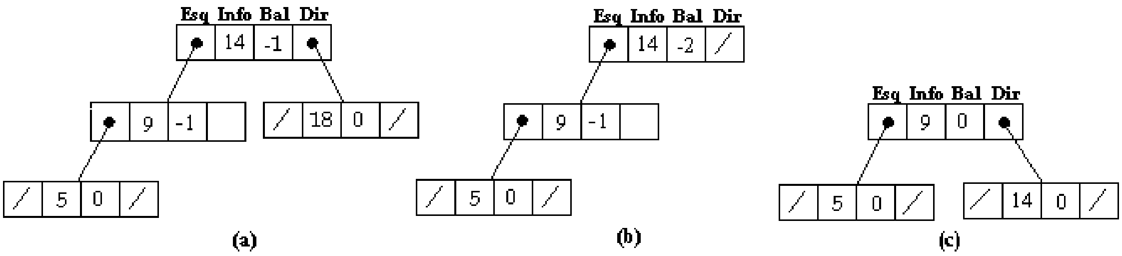


Figura 13. (a) Árvore de onde vai ser removido o nó 18 (b) Árvore após remoção (c) Árvore após rotação direita.

Resumo:

H(**p**) = -1	H(**u**) = 1	Remoção a direita de p (H(**p**) = -2, H(**u**) = 1)	Rotação Dupla a Direita
	H(**u**) = -1	Remoção a direita de p (H(**p**) = -2, H(**u**) = -1)	Rotação Simples a Direita
H(**p**) = 1	H(**u**) = 1	Remoção a esquerda de p (H(**p**) = 2, H(**u**) = 1)	Rotação Simples a Esquerda
	H(**u**) = -1	Remoção a esquerda de p (H(**p**) = 2, H(**u**) = -1)	Rotação Dupla a Esquerda

Departamento de Computação
Av. Engº Luiz Edmundo Carrijo Coube 14-01 Vargem Limpa Bauru-SP CEP: 17033-360
Fone: (14) 3103-6034 Fax: (14) 3103-6079 email: dcogeral@fc.unesp.br

4. Exercícios:

1. Sejam as sequencias de números:
   a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
   b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
   c) 10, 20, 30, 40, 35, 25, 15, 5, 7 12, 17, 27, 37, 22, 02
   d) 14, 15, 4, 9, 7, 18, 3, 5, 16, 20, 17, 19, 10, 11, 2, 1, 0, 12, 13
   e) 50, 30, 20, 70, 80, 35, 33, 34, 90
   f) 10, 5, 7, 25, 20, 3, 1, 2, 6, 4
   g) 11, 26, 27, 32, 57, 66, 74, 78, 79, 94, 95, 96
   h) 10, 5, 8, 25, 20, 3, 1, 6,7, 2
   i) 50, 70, 90, 40, 35, 80, 85, 97, 95, 99

   I. Faça a inserção dos números de cada sequencia em uma árvore AVL, sem usar o computador. Mostre balanço de cada nó e as rotações que foram feitas
   II. Faça a remoção das raízes de cada árvore AVL gerada no item anterior. Retire sempre a raiz até chegar numa árvore nula.

2. Usando árvores AVL que guardam números em seus nós, faça rotinas que:
a) encontre o menor elemento da árvore
b) encontre o maior elemento da árvore
c) dado um elemento pertence a árvore, descubra quem é seu pai e seu irmão.

3. Transformar uma árvore busca binária em uma AVL. Para isso tem-se que percorrer a primeira árvore segundo algum critério e inserir os valores armazenados em seus nós na árvore AVL. Implemente essa conversão usando os percursos pré-ordem, em-ordem e pós-ordem.

4. Construa uma rotina que dadas duas árvores AVL verifique se elas são semelhantes.

5. Crie uma rotina que calcule a altura de uma árvore AVL.

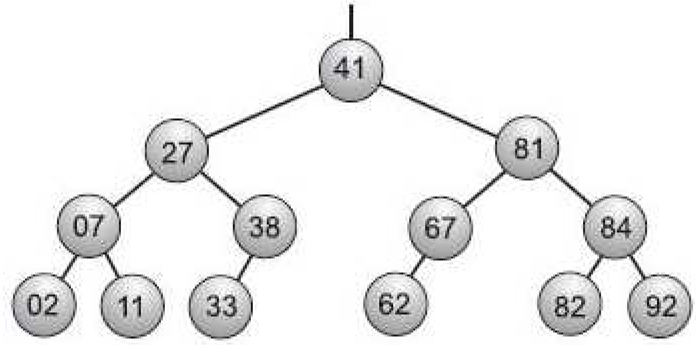
6. Crie uma rotina para que dada uma árvore binária de busca, verifique se ela é uma AVL.

7. A profundidade (= depth) de um nó x em uma árvore binária com raiz R é a distância entre x e R. Mais precisamente, a profundidade de x é o comprimento do (único) caminho que vai de R até x. Escreva uma função que determine a profundidade de um nó em relação à raiz da árvore.

8. (POSCOMP, 2007, Q23) Seja T uma árvore AVL vazia. Supondo que os elementos 5, 10, 11, 7, 9, 3 e 6 sejam inseridos nessa ordem em T, indique a sequência abaixo que corresponde a um percurso de T em pós-ordem:
   (a) 3, 5, 6, 7, 9, 10 e 11.
   (b) 7, 5, 3, 6, 10, 9 e 11.
   (c) 9, 10, 7, 6, 11, 5 e 3.
   (d) 11, 10, 9, 7, 6, 5 e 3.
   (e) 3, 6, 5, 9, 11, 10 e 7.

9. (Concurso) Uma árvore AVL é uma estrutura de dados muito usada para armazenar dados em memória. Ela possui algumas propriedades que fazem com que sua altura tenha uma relação muito específica com o número de elementos nela armazenados. Para uma folha, cuja altura é igual a um, tem-se uma árvore AVL com 6 nós. Qual é a altura máxima que esta árvore pode ter?

10. (Concurso) Suponha a seguinte árvore AVL:



A inserção do elemento 30 nesta árvore:

a) Aumenta a profundidade da árvore após uma rotação
b) Provoca uma rotação a direita
c) Deixa os nós 02 e 07 no mesmo nível
d) Altera a raiz da árvore
e) Torna o nó 33 pai do nó 27