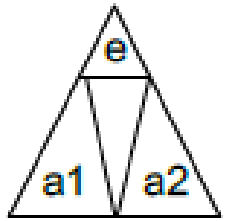
Árvore AVL

Prof. D.Sc. Saulo Ribeiro

4.9.1. El TAD AVL

En el TAD AVL sólo existen una constructora (crear un árbol AVL vacío) y dos modificadoras (insertar y eliminar un elemento), que garanticen las condiciones de balanceo que debe cumplir el árbol. La analizadora de búsqueda es igual a la de árboles binarios ordenados.

TAD AVL[TipoAVL]



{ inv: a1 y a2 son disyuntos, todos los elementos de a1 son menores que e,

todos los elementos de a2 son mayores que e, a1 y a2 son ordenados,

| altura (a1) - altura (a2) | ≤ 1, a1 y a2 son AVL }

Constructora:

- inicAVL: → AVL

Modificadoras:

- insAVL: AVL x TipoAVL → AVL
- elimAVL: AVL x TipoAVL → AVL

```
AVL inicAVL ( void )
/* Crea un árbol AVL vacío */


{ post: inicAVL = ∅ }
```

```
AVL insAVL ( AVL a, TipoAVL elem )
/* Adiciona un elemento a un árbol AVL */


{ pre: a = A, elem ∉ a }
{ post: insAVL = A ∪ { elem } }
```

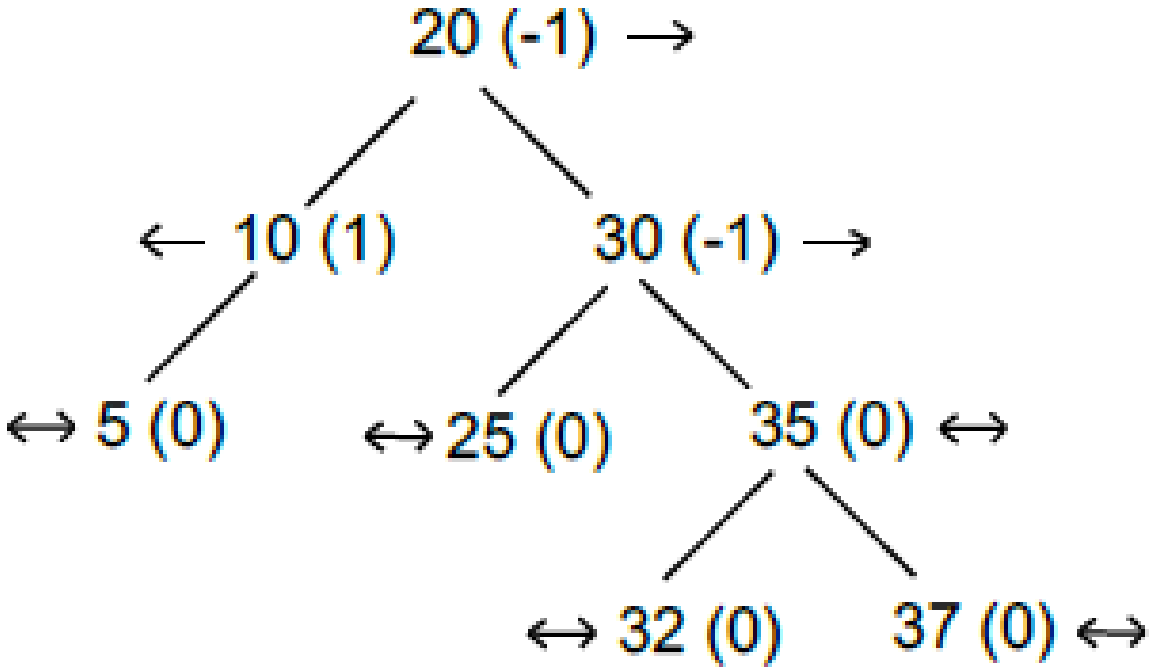
```
AVL elimAVL ( AVL a, TipoAVL elem )
/* Elimina un elemento de un árbol AVL */

{ pre: a = A, elem ∈ a }
{ post: elimAVL = A - { elem } }
```

4.9.2. Estructuras de Datos

La implementación de las operaciones del TAD AVL se ilustra utilizando el esquema de representación de árboles sencillamente encadenados, extendido con un campo en cada nodo (balan) para incluir un indicador del estado de balanceo, que corresponde a la diferencia de altura de sus subárboles (izq - der). Este nuevo campo es utilizado por las modificadoras para dirigir la estrategia de rebalanceo de un árbol. Las convenciones gráficas son las siguientes:

- balan = 0 (↔): ambos subárboles tienen la misma altura
- balan = -1 (→): el subárbol derecho excede en 1 la altura del izquierdo
- balan = 1 (←): el subárbol izquierdo excede en 1 la altura del derecho

Árvore AVL



Árvore AVL: Inserção

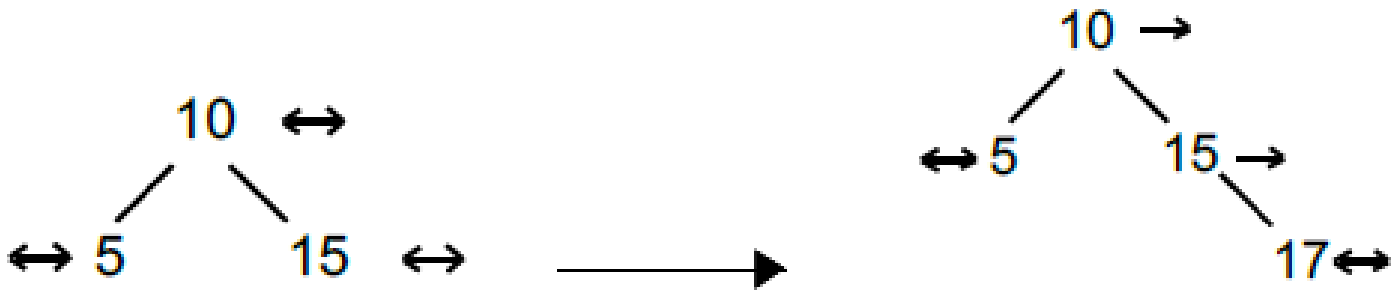
1. Adicionar o elemento numa árvore binária de busca
2. Restabelecer o balanceamento da árvore usando rotações

Árvore AVL: Inserção – Caso 1

Caso 1: No se dañó el balanceo al insertar el elemento

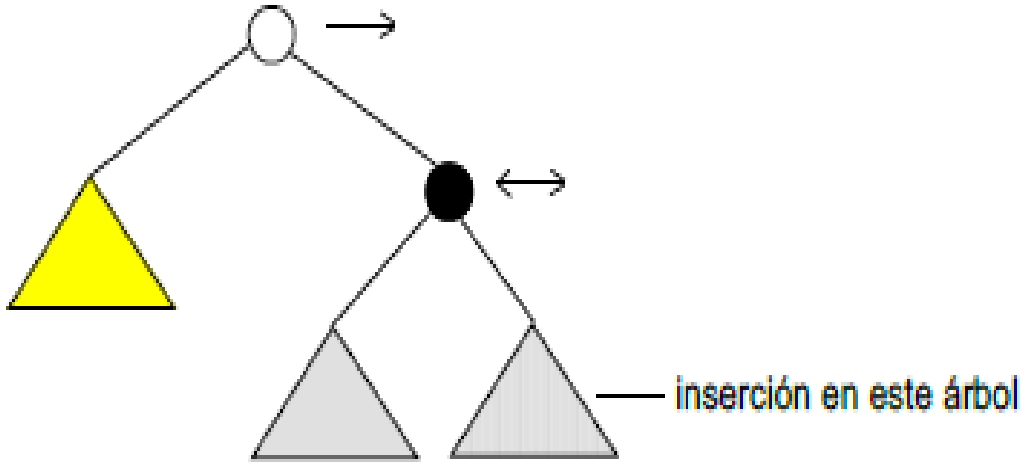
Proceso: No se debe rebalancear. Sólo se debe actualizar el indicador de balanceo en los elementos afectados

Ejemplo: Resultado de insertar el elemento 17



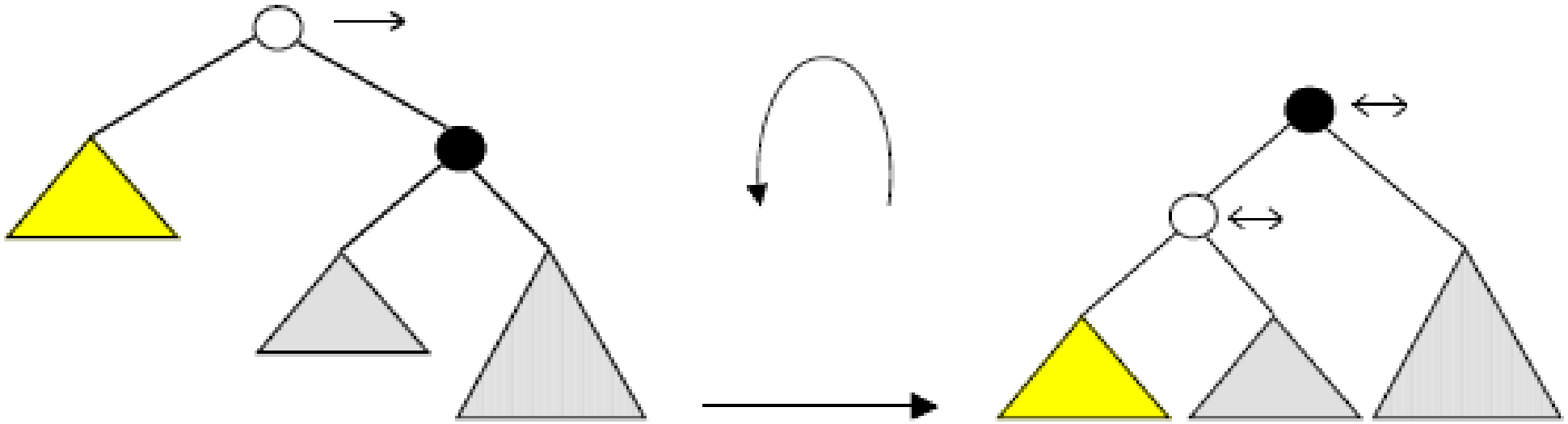
Árvore AVL: Inserção – Caso 2

Caso 2: El árbol inicial es de la forma mostrada en la figura, y se inserta el nuevo elemento en el subárbol derecho.



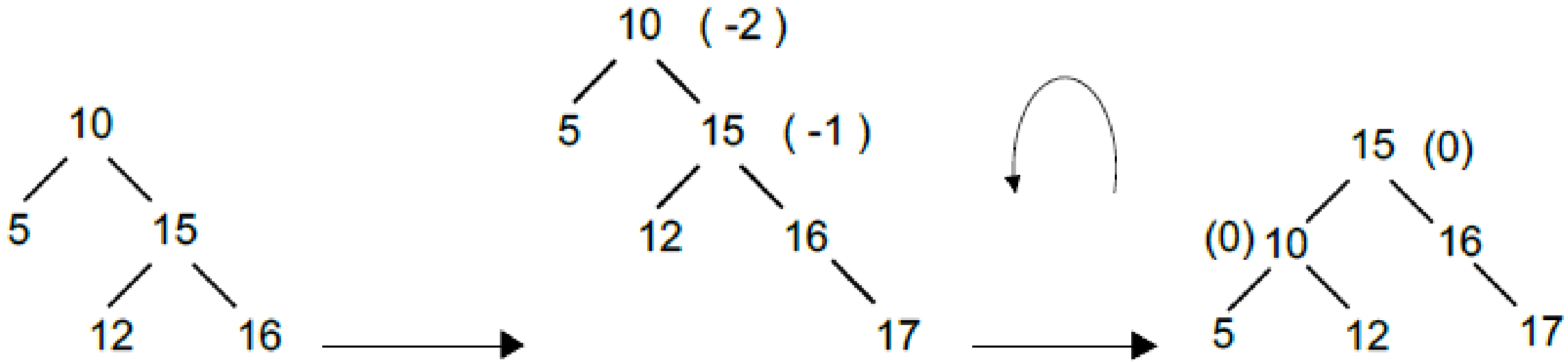
Árvore AVL: Inserção – Caso 2

Proceso: Rebalancear el árbol con una rotación a la izquierda, de la siguiente manera:



Árvore AVL: Inserção – Caso 2

Ejemplo: Resultado de insertar el elemento 17



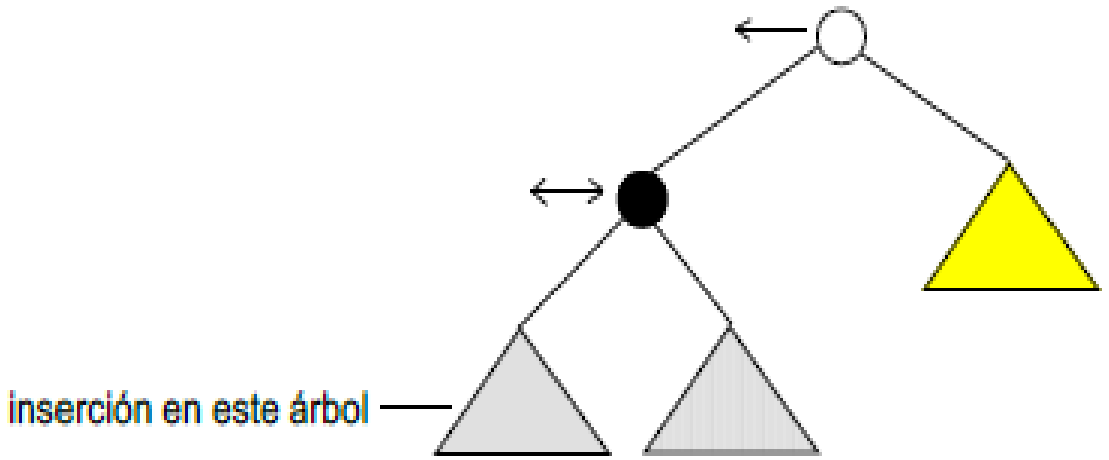
Árvore AVL: Inserção – Caso 2 - rotacaoEsq

Rutina para la rotación a la izquierda (no recalcula el factor de balance):

```
AVL roteIzq( AVL a )
{       AVL temp = a->der;
        a->der = temp->izq;
        temp->izq = a;
        return temp;
}
```

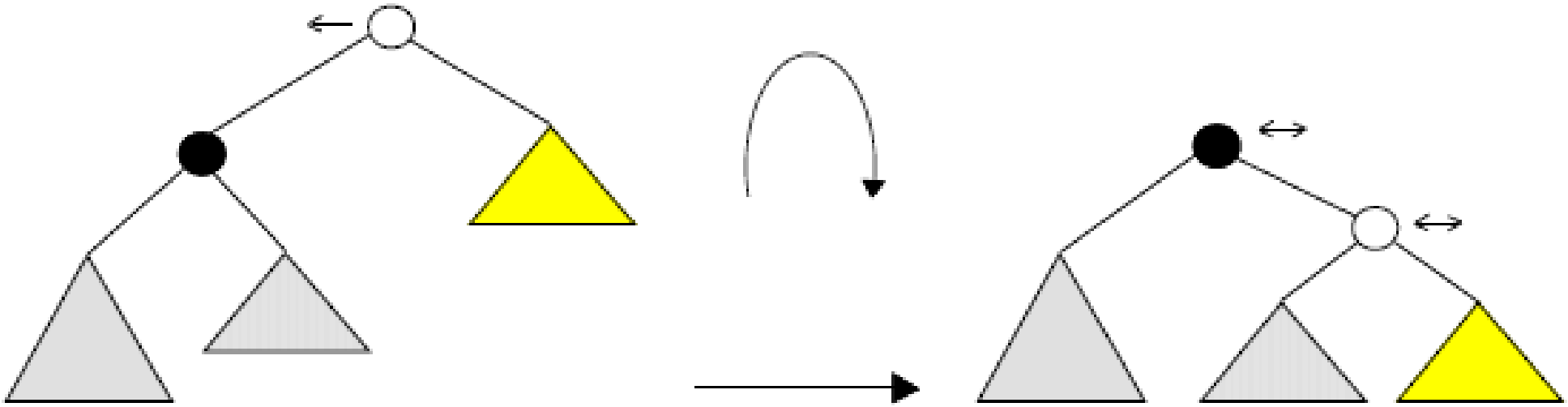
Árvore AVL: Inserção – Caso 3

Caso 3: El árbol inicial es de la forma mostrada en la figura, y se inserta el nuevo elemento en el subárbol izquierdo. Es un caso simétrico al anterior.



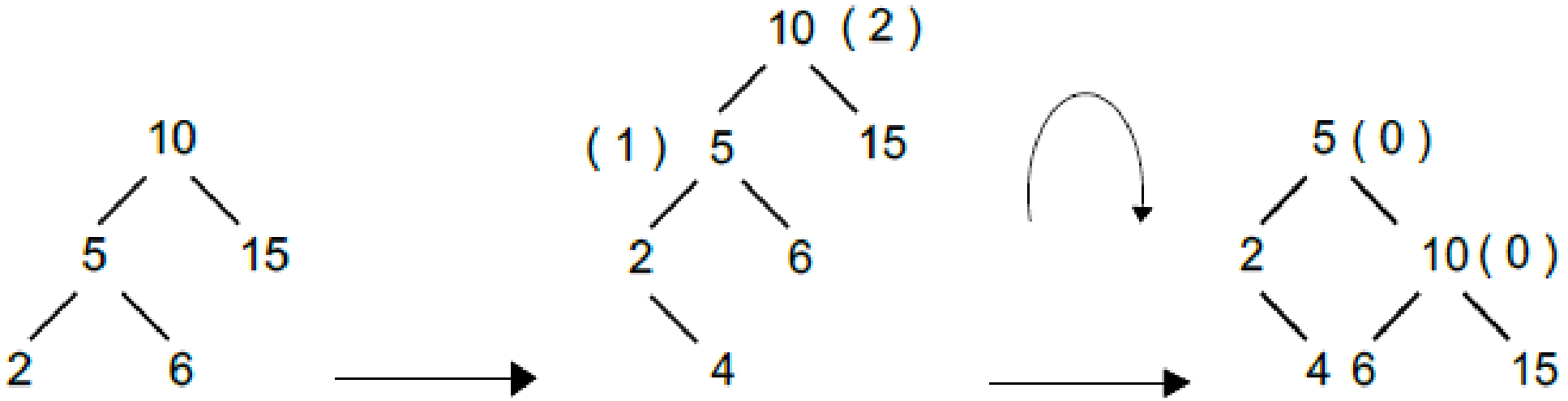
Árvore AVL: Inserção – Caso 3

Proceso: Rebalancear el árbol con una rotación a la derecha, de la siguiente manera:



Árvore AVL: Inserção – Caso 3

Ejemplo: Resultado de insertar el elemento 4



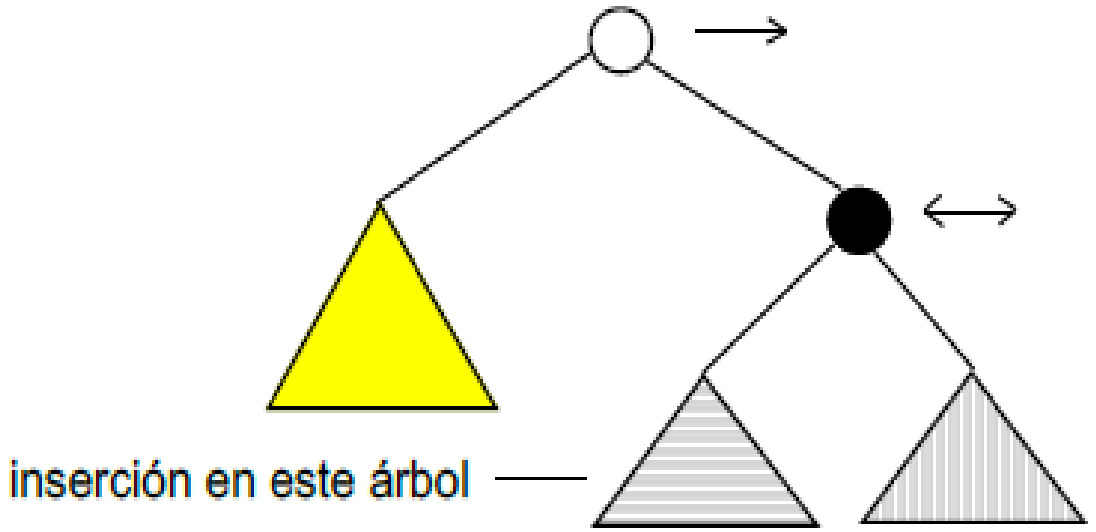
Árvore AVL: Inserção – Caso 3 - rotacaoDir

Rutina para la rotación a la derecha (no recalcula el factor de balance):

```
AVL roteDer( AVL a )
{       AVL temp = a->izq;
        a->izq = temp->der;
        temp->der = a;
        return temp;
}
```

Árvore AVL: Inserção – Caso 3 - rotacaoDir

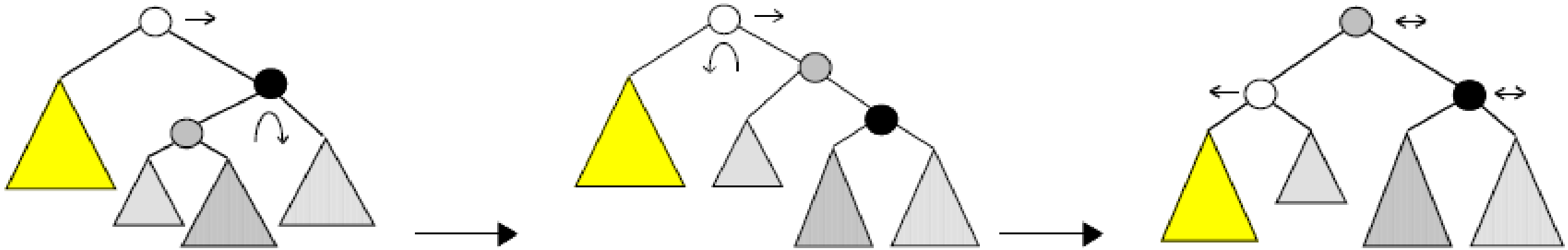
Caso 4: El árbol inicial es de la forma mostrada en la figura, y se inserta el nuevo elemento en un subárbol interno.



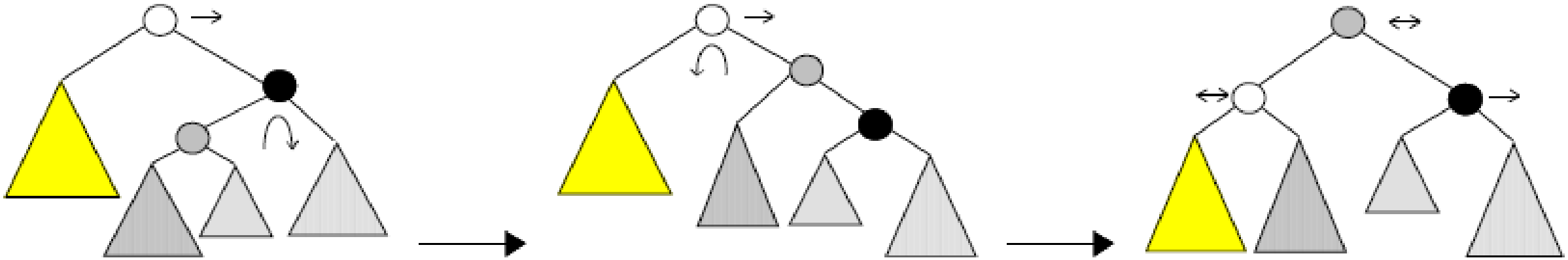
Árvore AVL: Inserção – Caso 4 (dois subcasos)

Proceso: Rebalancear el árbol con una doble rotación derecha-izquierda (subárbol derecho una rotación a la derecha, y, luego, el árbol completo una rotación a la izquierda). Pueden darse dos casos distintos, con respecto al subárbol del subárbol interno que se desbalancea, pero ambos se resuelven con la misma estrategia de rotación, de la siguiente manera:

Árvore AVL: Inserção – Caso 4 (dois subcasos)

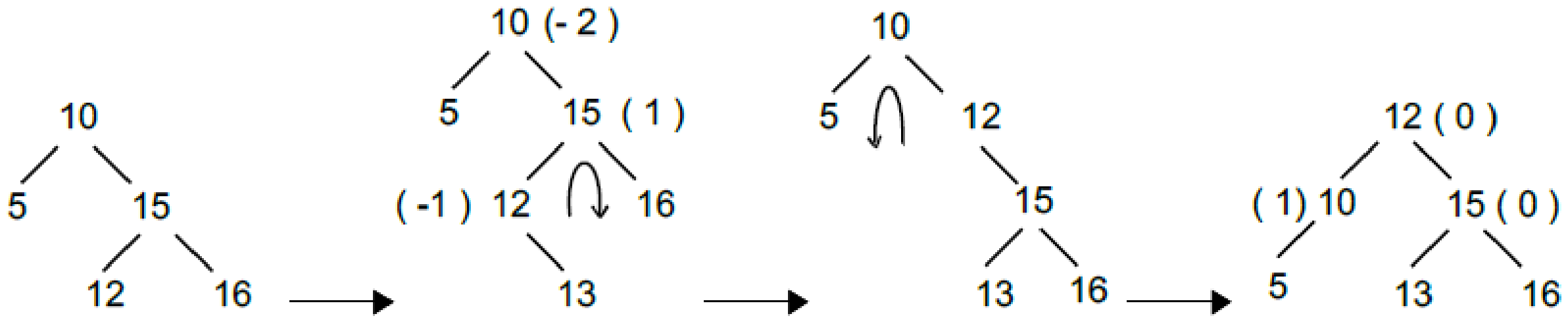


Árvore AVL: Inserção – Caso 4 (dois subcasos)



Árvore AVL: Inserção – Caso 4 (dois subcasos)

Ejemplo: Resultado de insertar el elemento 13



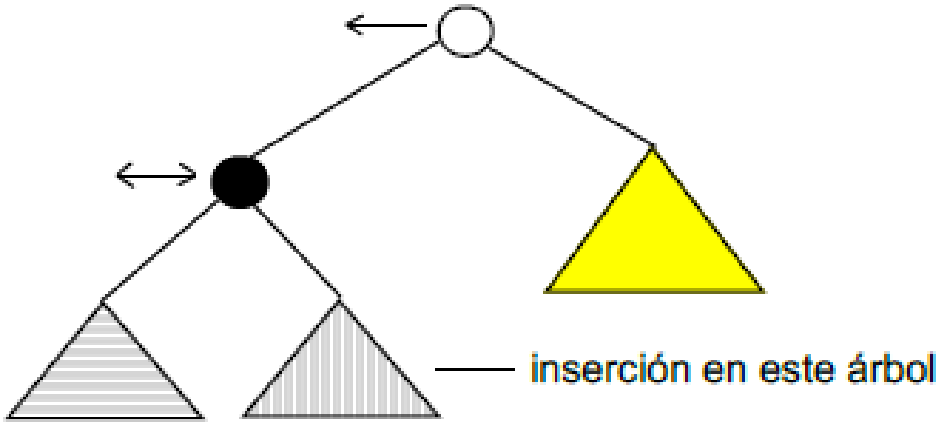
Árvore AVL: Inserção – Caso 4 (dois subcasos) rotacaoDirEsq

Rutina para la doble rotación derecha-izquierda (no recalcula el factor de balance):

```
AVL roteDerIzq( AVL a )
{       a->der = roteDer( a->der );
        return roteIzq( a );
}
```

Árvore AVL: Inserção – Caso 5 (dois subcasos)

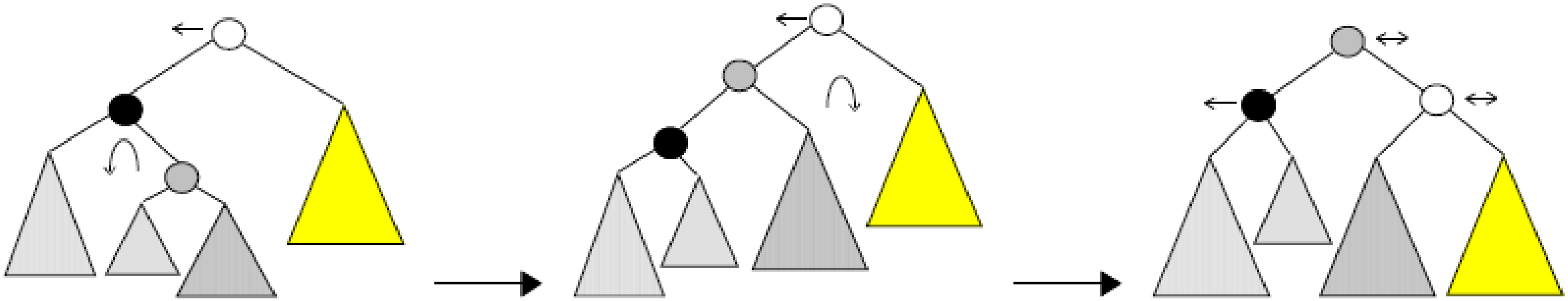
Caso 5: El árbol inicial es de la forma mostrada en la figura, y se inserta el nuevo elemento en un subárbol interno. Es un caso simétrico al anterior:



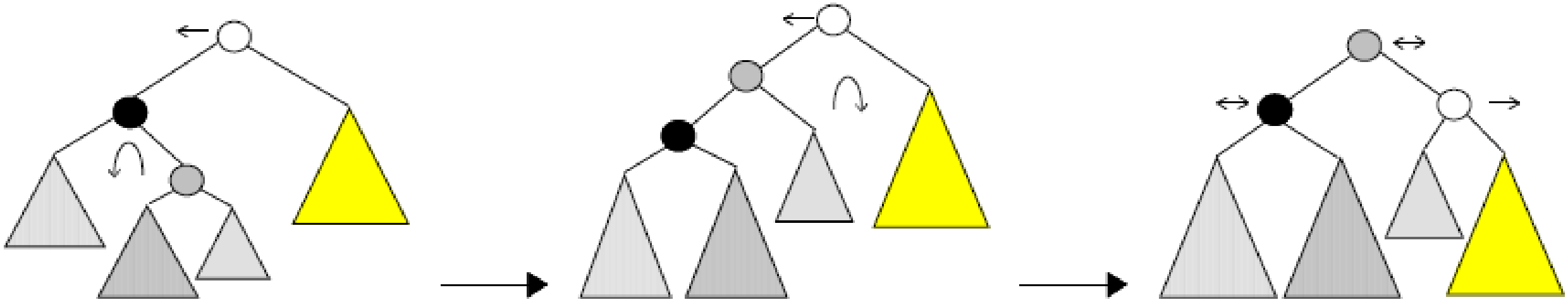
Árvore AVL: Inserção – Caso 5 (dois subcasos)

Proceso: Rebalancear el árbol con una doble rotación izquierda-derecha (subárbol izquierdo una rotación a la izquierda, y, luego, el árbol completo una rotación a la derecha). Pueden darse dos casos distintos, con respecto al subárbol del subárbol interno que se desbalancea, pero ambos se resuelven con la misma estrategia de rotación, de la siguiente manera:

Árvore AVL: Inserção – Caso 5 (dois subcasos)

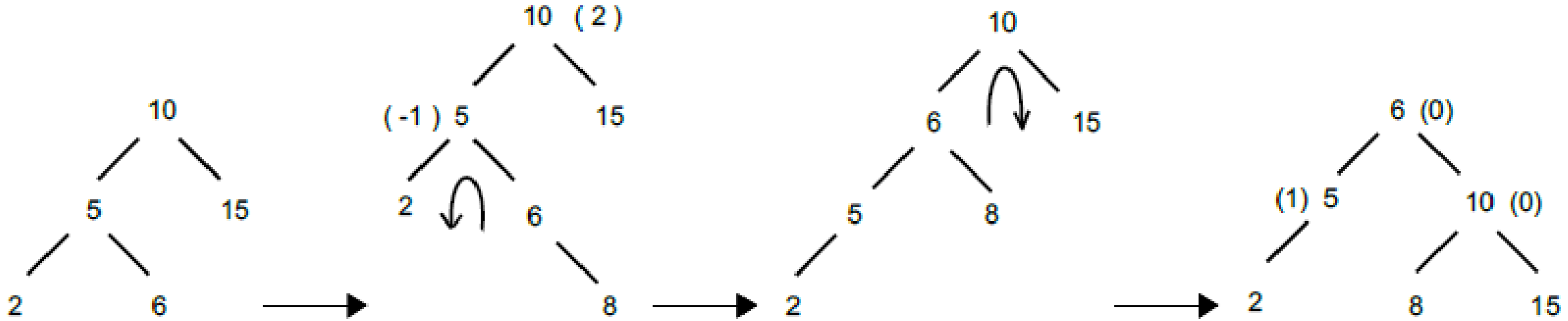


Árvore AVL: Inserção – Caso 5 (dois subcasos)



Árvore AVL: Inserção – Caso 5 (dois subcasos)

Ejemplo: rebalanceo del árbol al insertar el 8



Árvore AVL: Inserção – Caso 5 (dois subcasos) rotacaoEsqDir

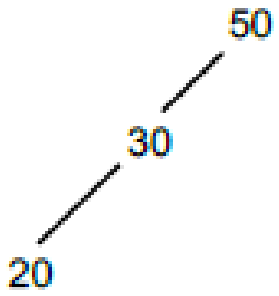
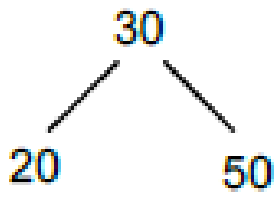
Rutina para la doble rotación izquierda-derecha (no recalcula el factor de balance):

```
static AVL roteIzqDer( AVL a )
{       a->izq = roteIzq( a->izq );
        return roteDer( a );
}
```

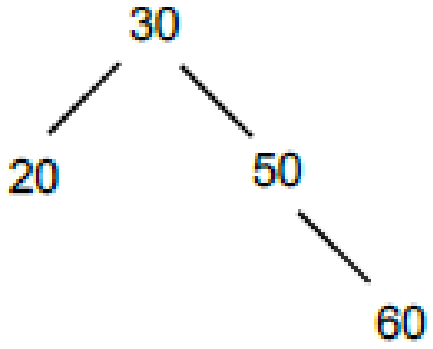
Árvore AVL: Inserção

- Inserir na sequência: 50, 30, 20, 60, 70, 55, 57, 58

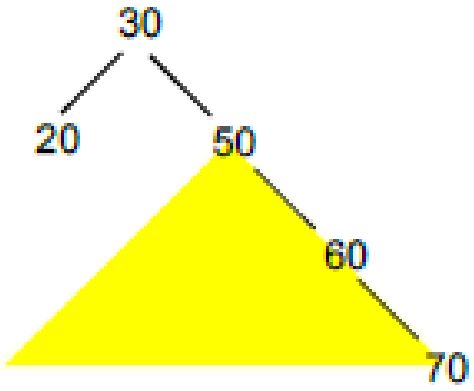
Árvore AVL: Inserção

1. Insertar el 50 - Caso 1	**2.** Insertar el 30 - Caso 1
	
3. Insertar el 20 - Caso 3 (rotación derecha)	
	

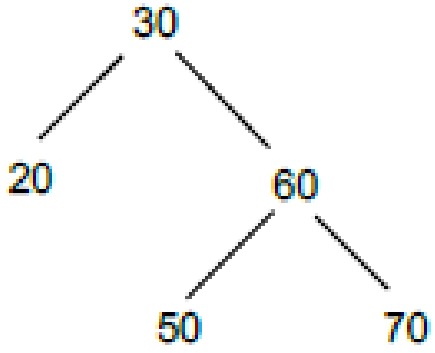
4. Insertar el 60 - Caso 1

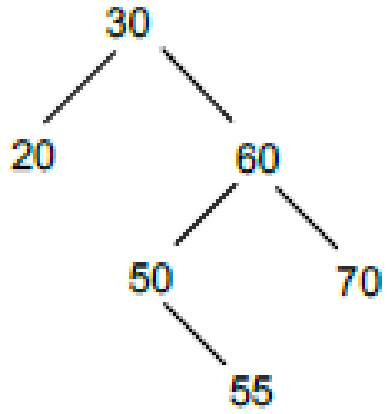


5. Insertar el 70 - Caso 2 (rotación izquierda del subárbol derecho, mínimo subárbol desbalanceado)

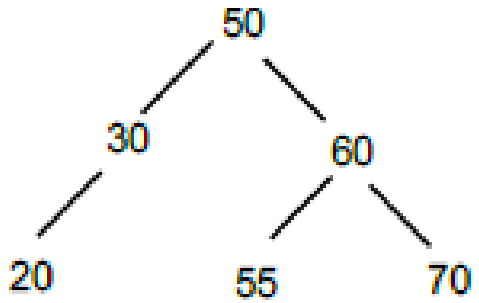
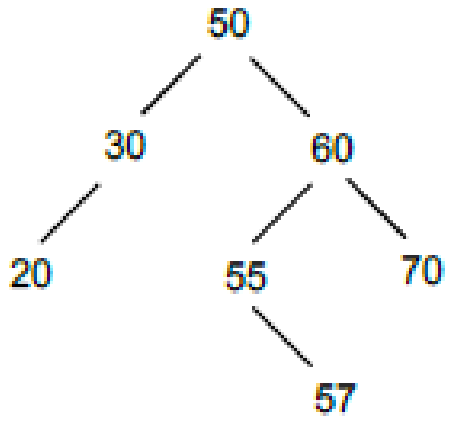


6. Insertar el 55 - Caso 4 (rotación derecha izquierda)



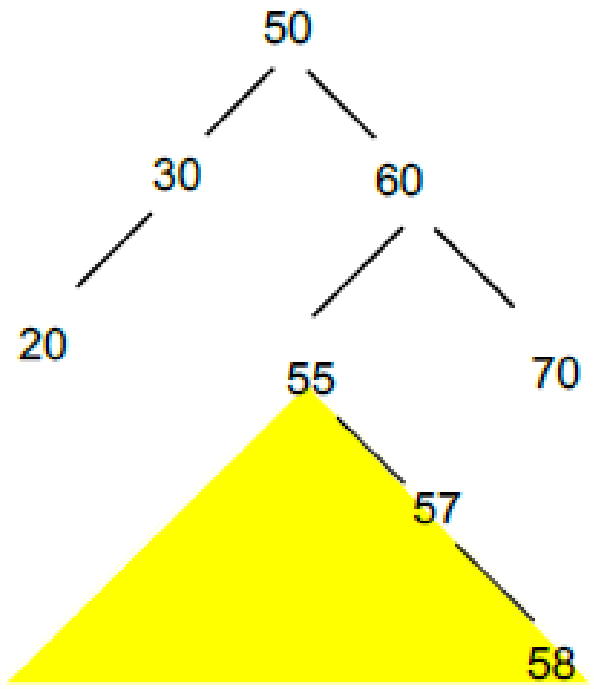


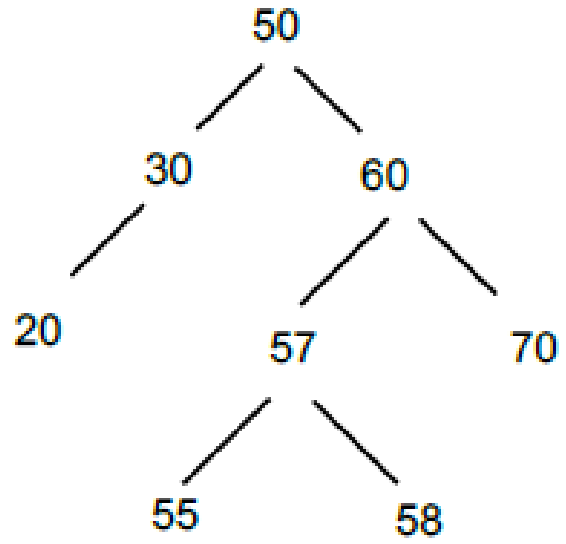
Árvore AVL: Inserção

	7. Insertar el 57 - Caso 1
	

Árvore AVL: Inserção

8. Insertar el 58 - Caso 4 (rotación derecha-izquierda del mínimo subárbol desbalanceado)





La rutina completa, para balancear el subárbol derecho de un árbol AVL (el balanceo del izquierdo es equivalente), establece a cuál de los dos casos posibles de desbalanceo corresponde (2 ó 4) y llama la rutina respectiva, actualizando luego los indicadores que lo requieran.

```
/* pre: se ha insertado un elemento en el subárbol derecho de a y se ha desbalanceado */
/* post: retorna un árbol AVL con todos los elementos de a */

AVL balanceaDer ( AVL a )
{   if( a->der->balan == DER )
    {     /* Caso 2 */
          a->balan = a->der->balan = BAL;
          a = roteIzq( a );
    }
    else
    {     /* Caso 4 */
           switch( a->der->izq->balan )
          { case IZQ:     a->balan = BAL;
                          a->der->balan = DER;
                          break;
            case BAL:     a->balan = a->der->balan = BAL;
                          break;
             case DER:    a->balan = IZQ;
                          a->der->balan = BAL;
                          break;
          }
          a->der->izq->balan = BAL;
          a = roteDerIzq( a );
    }
    return a;
}
```

Árvore AVL: Inserção

La operación de inserción en un árbol AVL se basa en varias rutinas, las cuales se presentan a continuación, y utilizan los algoritmos de rebalanceo descritos en cada caso. La primera rutina (insAVL) corresponde a la operación de inserción en un árbol AVL; crea el nodo que se va a adicionar y abre espacio para el manejo de un parámetro adicional, que se va a manejar durante todo el proceso de inserción, para informar si su altura aumentó después de alguna modificación estructural del árbol.

```
AVL insAVL( AVL a, TipoAVL elem )
{   AVL p = ( AVL )malloc( sizeof( TAVL ) );
    int masAlto;
    p->izq = p->der = NULL;
    p->info = elem;
    p->balan = BAL;
    return insertar( a, p, &masAlto );
}
```

Árvore AVL: Inserção

La rutina insertar recibe un árbol AVL y un nodo con el elemento que se quiere agregar, y retorna funcionalmente un árbol AVL con el nuevo nodo agregado a la estructura inicial. Informa, en un parámetro por referencia, si la altura del árbol resultante es mayor que la altura del árbol inicial. La rutina considera tres casos principales:

- el árbol es vacío: retorna el nodo apuntado por p
- el nuevo elemento es menor que la raíz: inserta el nodo apuntado por p en el subárbol izquierdo, y, si la altura del árbol aumenta, lo rebalancea según el factor de balance de la raíz
- el nuevo elemento es mayor que la raíz: inserta el nodo apuntado por p en el subárbol derecho, y, si la altura del árbol aumenta, lo rebalancea según el factor de balance de la raíz

Árvore AVL: Inserção

```
/* pre: p->info ∉ a, a = A */
/* post:  insertar = A ∪ p, *masAlto = altura( a ) > altura( A ) */

AVL insertar( AVL a , Nodo AVL *p, int *masAlto )
{   if( a == NULL )
    {     *masAlto = TRUE;
          a = p;
    }
    else if( a->info > p->info )
    {     a->izq = insertar( a->izq, p, masAlto );
          if( *masAlto )
          switch( a->balan )
          { case IZQ:     *masAlto = FALSE;
                          a = balanceaIzq( a );
                          break;
            case BAL:     a->balan = IZQ;
                          break;
            case DER:     *masAlto = FALSE;
                          a->balan = BAL;
                          break;
          }
    }
    else
    {     a->der = insertar( a->der, p, masAlto );
          if( *masAlto )
          switch( a->balan )
          { case IZQ:     *masAlto = FALSE;
                          a->balan = BAL;
                          break;
            case BAL:     a->balan = DER;
                          break;
            case DER:     *masAlto = FALSE;
                          a = balanceaDer( a );
          }
    }
    return a;
}
```

Árvore AVL: Balanceamento

Las funciones balanDer y balanIzq tienen una estructura similar, y se encargan de reestablecer el balanceo de un árbol al cual se le ha eliminado un elemento de uno de los subárboles, y, por esta razón, el otro subárbol está generando un desbalanceo. Al entrar a la rutina, el parametro *menosAlto es TRUE. A continuación se presenta una de estas rutinas.

Árvore AVL: Balanceai

```
AVL balanIzq( AVL a, int *menosAlto )
{   switch( a->balan )
    { case IZQ:     if( a->izq->balan != DER )
                    {     a = roteDer( a );
                          if( a->balan == BAL )
                          {     a->balan = DER;
                                a->der->balan = IZQ;
                                *menosAlto = FALSE;
                          }
                          else
                                a->balan = a->der->balan = BAL;
                    }
                    else
                    {     a = roteIzqDer( a );
                          a->der->balan = ( a->balan == IZQ ) ? DER : BAL;
                          a->izq->balan = ( a->balan == DER ) ? IZQ : BAL;
                          a->balan = BAL;
                    }
                    break;
      case BAL:     a->balan = IZQ;
                    *menosAlto = FALSE;
                    break;
      case DER:     a->balan = BAL;
                    break;
    }
    return a;
}
```

Árvore AVL: Exercícios

Muestre el proceso de creación de un árbol AVL para la siguiente secuencia de elementos: identifique el caso de desbalanceo al cual se llega después de cada inserción y aplique la solución correspondiente:

a-) 10 - 8 - 5 - 20 - 30 - 25

b-) 25 - 20 - 30 - 22 - 12 - 27 - 32 - 28 - 26 - 29

c-) 5 - 10 - 15 - 20 - 25 - 30 - 35 - 40 - 45

d-) 100 - 90 - 80 - 70 - 60 - 50 - 40 - 30 - 20 - 10