Árboles balanceados

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados
  - datos en orden inverso

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados
  - datos en orden inverso
  - llaves pequeñas y grandes alternadas

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados
  - datos en orden inverso
  - llaves pequeñas y grandes alternadas
- Afortunadamente para la búsqueda por BST existen técnicas generales de balanceo de árboles que permiten garantizar que el peor caso no ocurra jamás.

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados
  - datos en orden inverso
  - llaves pequeñas y grandes alternadas
- Afortunadamente para la búsqueda por BST existen técnicas generales de balanceo de árboles que permiten garantizar que el peor caso no ocurra jamás.
- Las operaciones de transformación son simples pero costosas en tiempo.

Árboles balanceados

- Los algoritmos en árboles binarios de búsqueda dan buenos resultados en el caso promedio pero el desempeño es malo en el peor caso:
  - datos ordenados
  - datos en orden inverso
  - llaves pequeñas y grandes alternadas
- Afortunadamente para la búsqueda por BST existen técnicas generales de balanceo de árboles que permiten garantizar que el peor caso no ocurra jamás.
- Las operaciones de transformación son simples pero costosas en tiempo.
- Permiten hacer el árbol lo más regular posible respecto a un parámetro que depende en general de su altura.

Árboles balanceados

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.
- Existen varias familias de árboles equilibrados: los árboles AVL, árboles rojo-negro, árboles 2-3 (también conocidos como a-b) ...

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.
- Existen varias familias de árboles equilibrados: los árboles AVL, árboles rojo-negro, árboles 2-3 (también conocidos como a-b) ...
- La operación para re-equilibrar un árbol se conoce como rotación.

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.
- Existen varias familias de árboles equilibrados: los árboles AVL, árboles rojo-negro, árboles 2-3 (también conocidos como a-b) ...
- La operación para re-equilibrar un árbol se conoce como rotación.
- La rotación se puede aplicar a todos los árboles binarios.

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.
- Existen varias familias de árboles equilibrados: los árboles AVL, árboles rojo-negro, árboles 2-3 (también conocidos como a-b) ...
- La operación para re-equilibrar un árbol se conoce como rotación.
- La rotación se puede aplicar a todos los árboles binarios.
- Una rotación cambia la estructura de un árbol binario de búsqueda sin interferir con el orden de sus elementos.

Árboles balanceados

- Una familia de árboles que satisfacen una tal condición de regularidad se llama familia de árboles equilibrados.
- Existen varias familias de árboles equilibrados: los árboles AVL, árboles rojo-negro, árboles 2-3 (también conocidos como a-b) ...
- La operación para re-equilibrar un árbol se conoce como rotación.
- La rotación se puede aplicar a todos los árboles binarios.
- Una rotación cambia la estructura de un árbol binario de búsqueda sin interferir con el orden de sus elementos.
- Hay dos tipos de rotaciones: hacia la derecha y hacia la izquierda.

Árboles AVL

Árboles AVL

- Introducidos por Adelson-Velskii y Landis en 1962.

Árboles AVL

- Introducidos por Adelson-Velskii y Landis en 1962.
- Constituyen una familia de BST equilibrados en altura.

Árboles AVL

- Introducidos por Adelson-Velskii y Landis en 1962.
- Constituyen una familia de BST equilibrados en altura.
- Un BST es un árbol AVL si, para todo nodo del árbol, las alturas de sus sub-árboles izquierdo y derecho difieren a lo más por 1.

Árboles AVL

- Introducidos por Adelson-Velskii y Landis en 1962.
- Constituyen una familia de BST equilibrados en altura.
- Un BST es un árbol AVL si, para todo nodo del árbol, las alturas de sus sub-árboles izquierdo y derecho difieren a lo más por 1.
- Por convención, la altura de un árbol vacío es -1 y la altura de una hoja es 0 (ambos son árboles AVL).

Árboles AVL

Pongamos $\delta(A) = 0$ para un árbol vacío y $\delta(A) = h(A_i) - h(A_d)$ en el caso general.

Árboles AVL

Pongamos $\delta(A) = 0$ para un árbol vacío y $\delta(A) = h(A_i) - h(A_d)$ en el caso general.

$\delta(A)$ se llama equilibrio de A.

la notación $\delta(x)$ denota el equilibro de un árbol donde x es la raíz.

Árboles AVL

Pongamos $\delta(A) = 0$ para un árbol vacío y $\delta(A) = h(A_i) - h(A_d)$ en el caso general.

$\delta(A)$ se llama equilibrio de A.

la notación $\delta(x)$ denota el equilibro de un árbol donde x es la raíz.

Definición: Un árbol binario de búsqueda es AVL si, para todo nodo x del árbol, $\delta(x) \in \{-1, 0, 1\}$.

Árboles AVL

Pongamos $\delta(A) = 0$ para un árbol vacío y $\delta(A) = h(A_i) - h(A_d)$ en el caso general.

$\delta(A)$ se llama equilibrio de A.

la notación $\delta(x)$ denota el equilibro de un árbol donde x es la raíz.

Definición: Un árbol binario de búsqueda es AVL si, para todo nodo x del árbol, $\delta(x) \in \{-1, 0, 1\}$.

La propiedad fundamental de los árboles AVL es que podemos acotar su altura en función del log del número de nodos en el árbol:

Árboles AVL

Árboles AVL

- Implementación análoga a los árboles binarios.

Árboles AVL

- Implementación análoga a los árboles binarios.
- Es necesario agregar un campo que contenga la altura del árbol cuya raíz es el nodo actual.

Árboles AVL

- Implementación análoga a los árboles binarios.
- Es necesario agregar un campo que contenga la altura del árbol cuya raíz es el nodo actual.
- Esta modificación convierte las operaciones de inserción y supresión más complicadas

Árboles AVL

- Implementación análoga a los árboles binarios.
- Es necesario agregar un campo que contenga la altura del árbol cuya raíz es el nodo actual.
- Esta modificación convierte las operaciones de inserción y supresión más complicadas
  - hay que actualizar las alturas cada vez.

Operaciones en árboles AVL: **inserción**

Operaciones en árboles AVL: **inserción**

- Se realiza de la misma manera que en un BST:

Operaciones en árboles AVL: **inserción**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar la hoja donde se deba insertar el nuevo nodo.

Operaciones en árboles AVL: **inserción**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar la hoja donde se deba insertar el nuevo nodo.
  - subir en el árbol para actualizar el valor de la altura de cada nodo (actualizar por un solo camino).

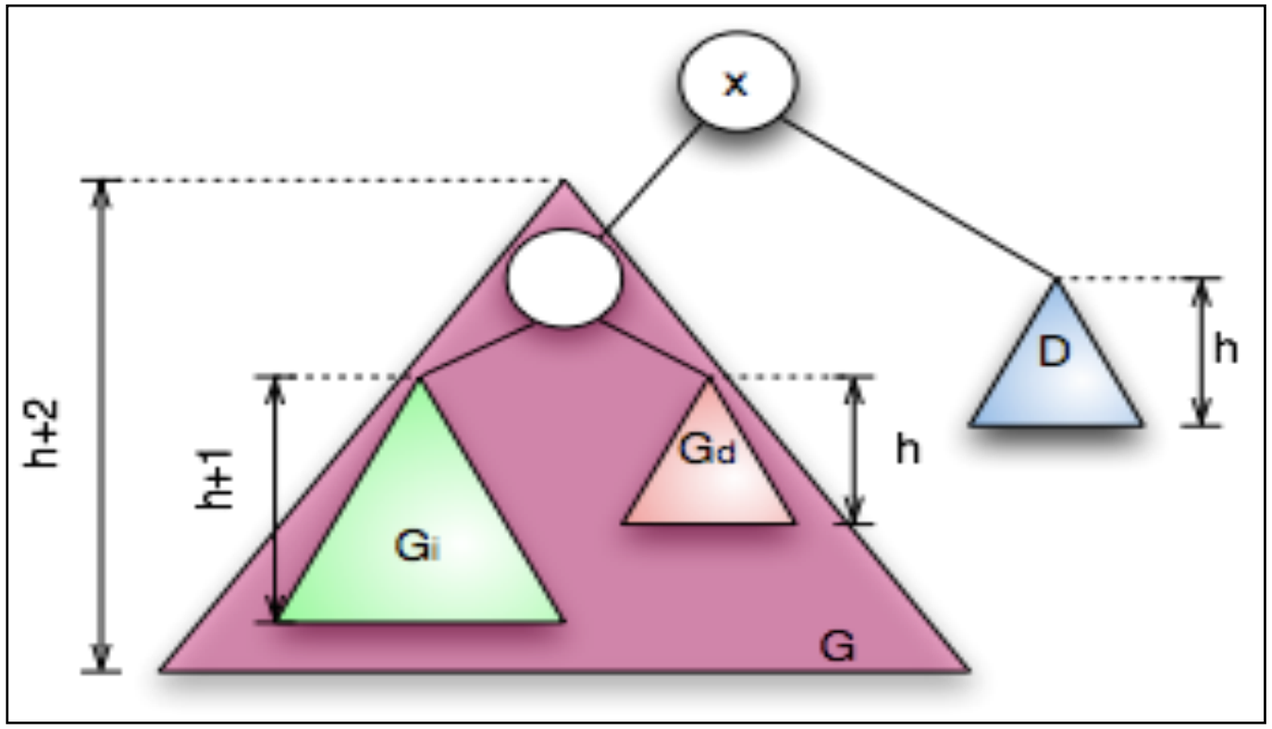
Operaciones en árboles AVL: **inserción**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar la hoja donde se deba insertar el nuevo nodo.
  - subir en el árbol para actualizar el valor de la altura de cada nodo (actualizar por un solo camino).
- La operación de inserción puede desequilibrar el árbol: el árbol resultante no es AVL.

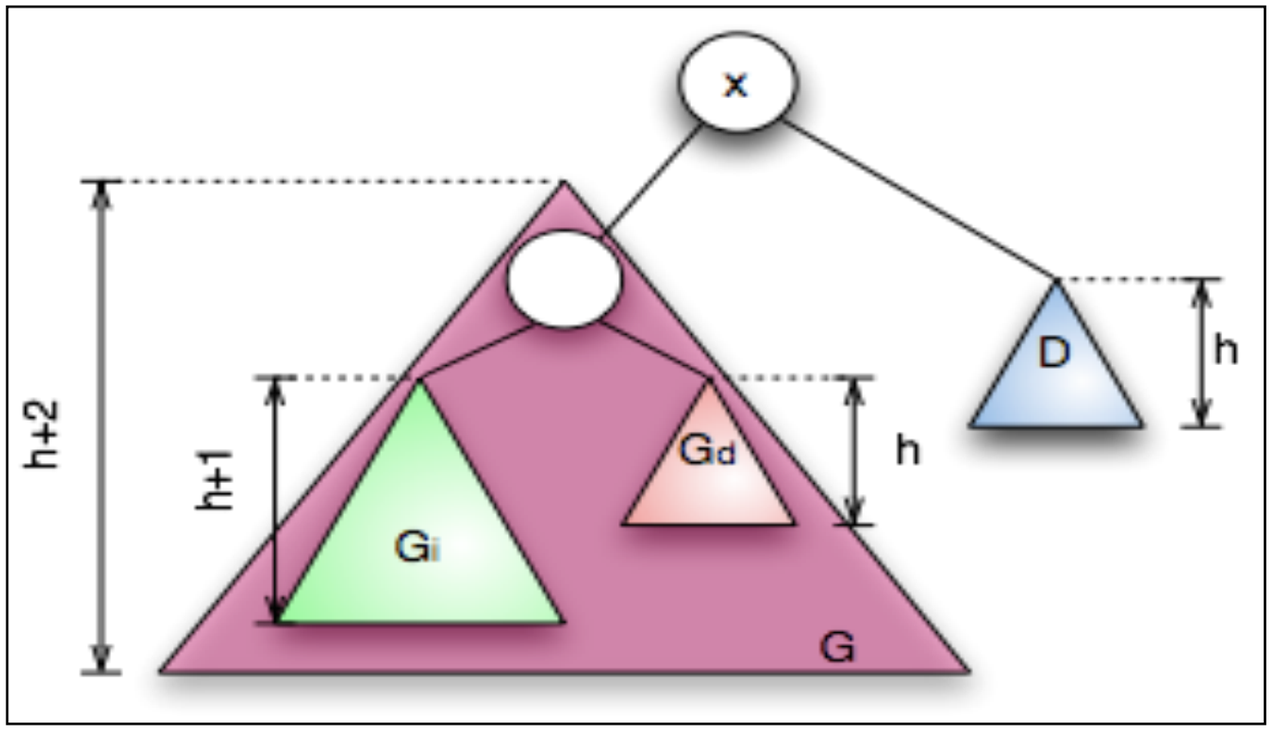
Operaciones en árboles AVL: **inserción**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar la hoja donde se deba insertar el nuevo nodo.
  - subir en el árbol para actualizar el valor de la altura de cada nodo (actualizar por un solo camino).
- La operación de inserción puede desequilibrar el árbol: el árbol resultante no es AVL.
- Para restablecer esta propiedad es necesario re-equilibrar el árbol por medio de rotaciones a través del camino que lleva de la hoja donde se realizó la inserción a la raiz.

Operaciones en árboles AVL: **balanceo**

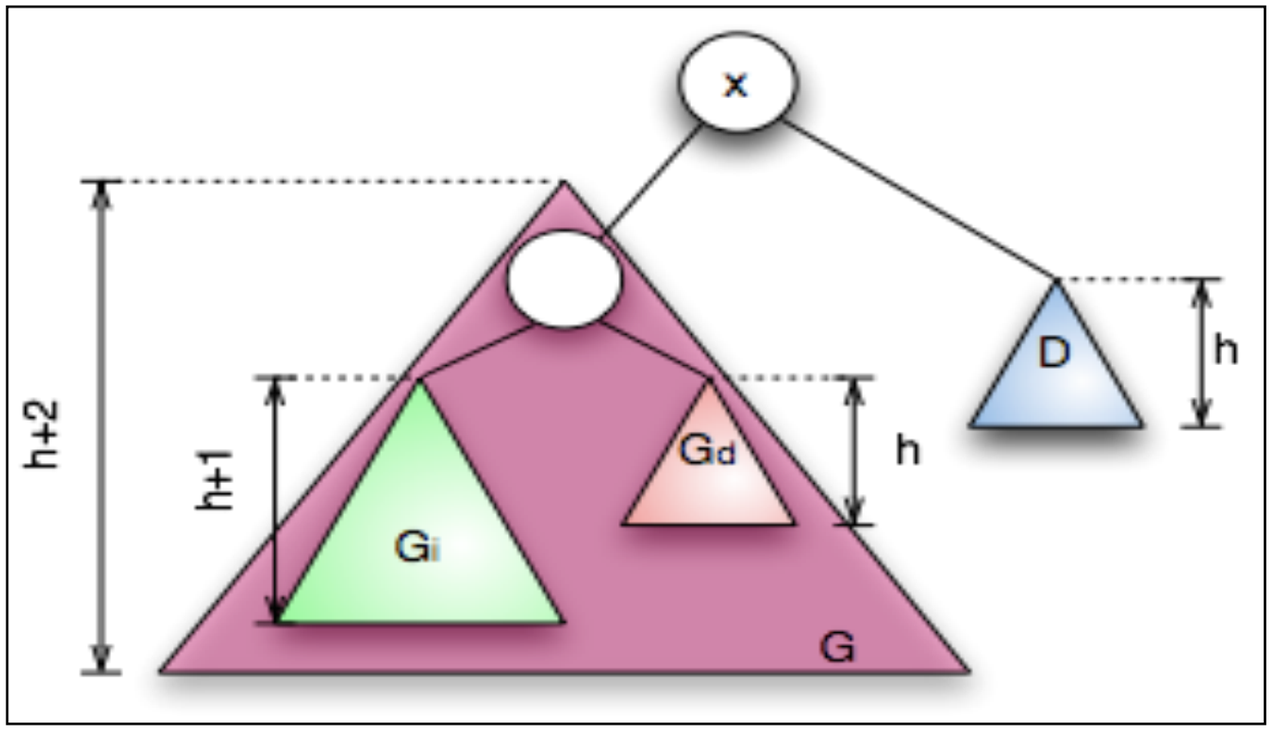


Operaciones en árboles AVL: **balanceo**



$$\delta(x) = h(G) - h(D) = 2$$

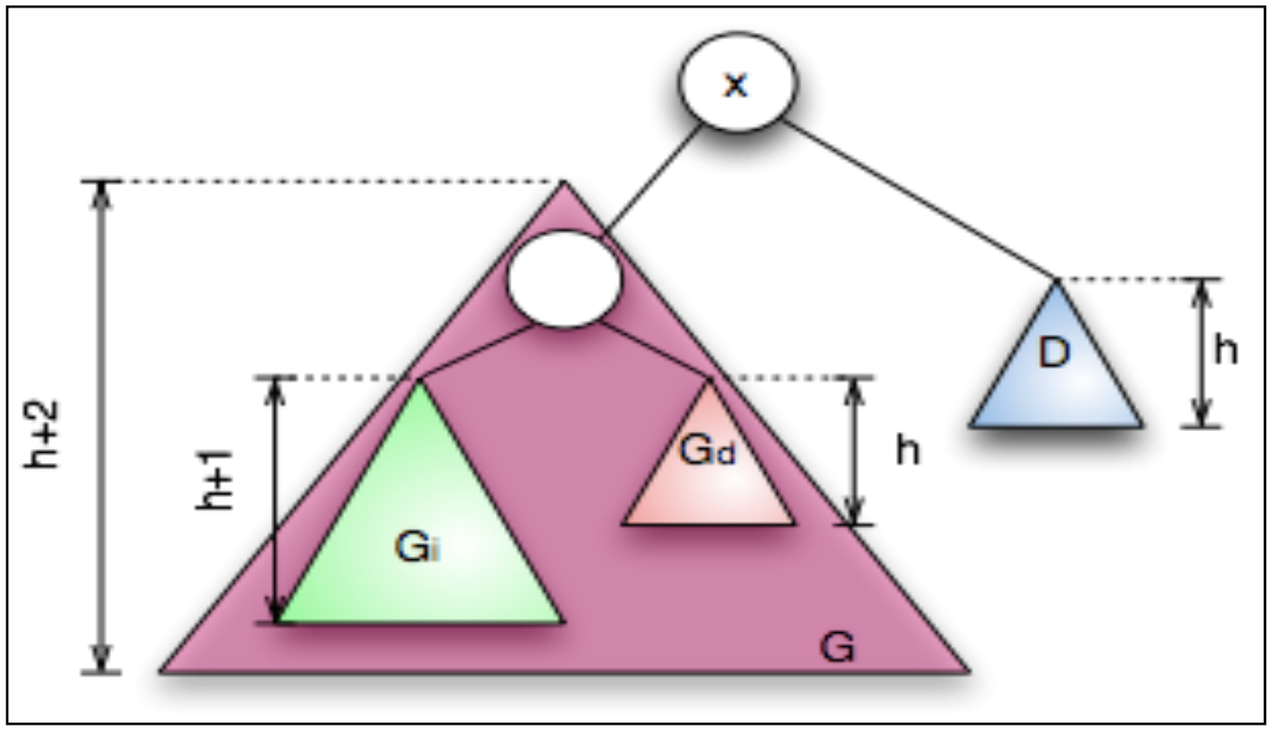
Operaciones en árboles AVL: **balanceo**



$$\delta(x) = h(G) - h(D) = 2$$

Si $\delta(G) = h(G_i) - h(G_d) > -1$: rotación derecha sobre x

Operaciones en árboles AVL: **balanceo**

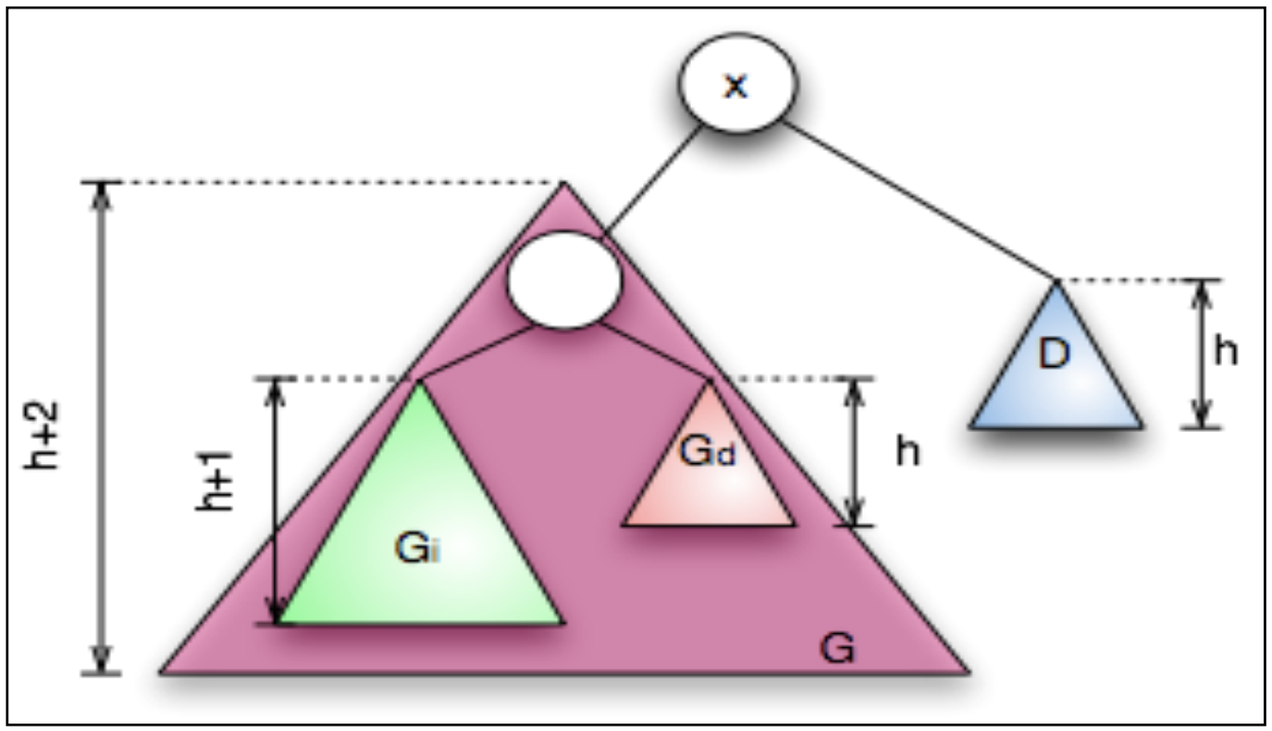


$$\delta(x) = h(G) - h(D) = 2$$

Si $\delta(G) = h(G_i) - h(G_d) > -1$: rotación derecha sobre x

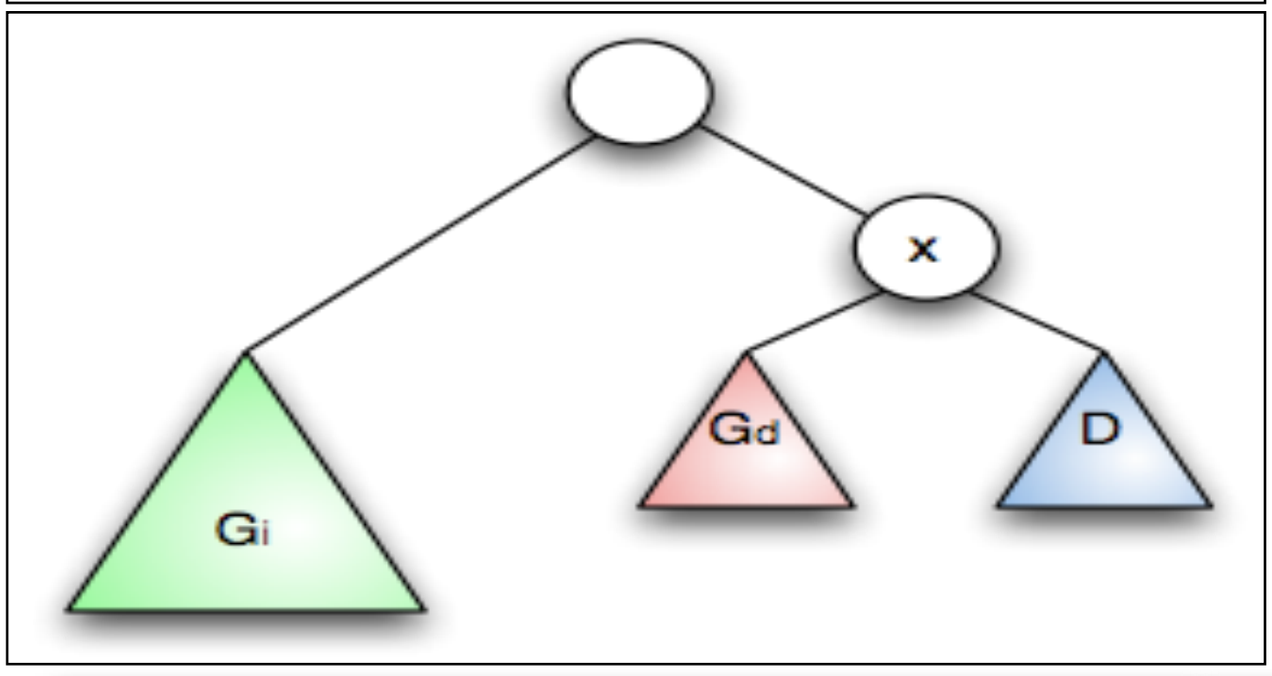
$$\delta(G) = h(G_i) - h(G_d) = 1 > -1$$

Operaciones en árboles AVL: **balanceo**



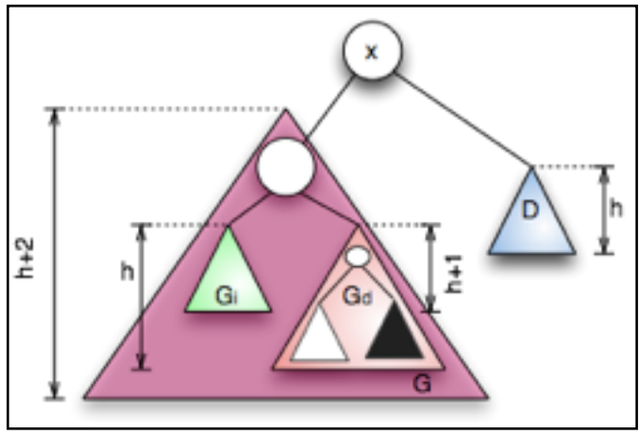
$$\delta(x) = h(G) - h(D) = 2$$

Si $\delta(G) = h(G_i) - h(G_d) > -1$: rotación derecha sobre x



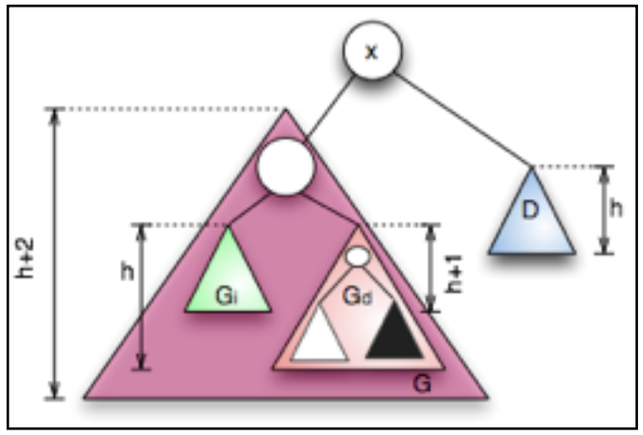
$$\delta(G) = h(G_i) - h(G_d) = 1 > -1$$

Operaciones en árboles AVL: **balanceo**



$$\delta(x) = h(G) - h(D) = 2$$

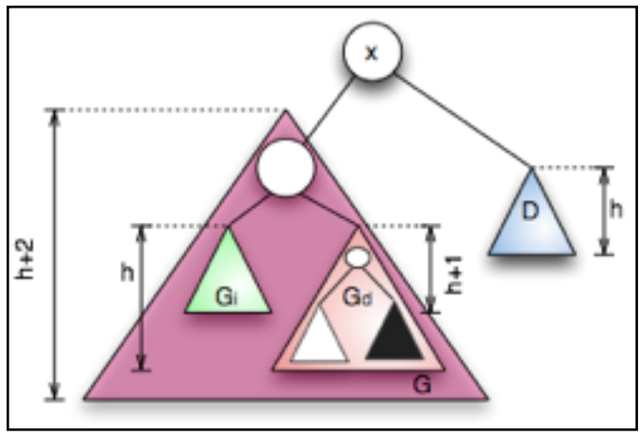
Operaciones en árboles AVL: **balanceo**



$$\delta(x) = h(G) - h(D) = 2$$

$$\delta(G) = h(G_i) - h(G_d) = -1 = -1$$

Operaciones en árboles AVL: **balanceo**

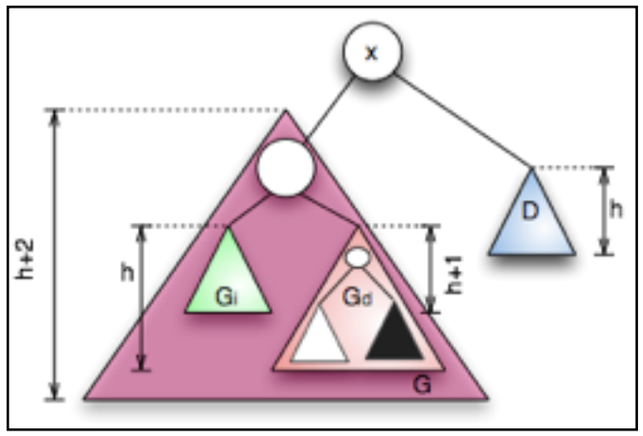


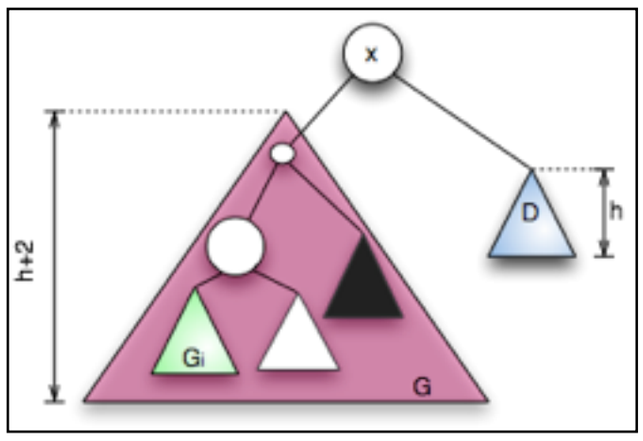
$$\delta(x) = h(G) - h(D) = 2$$

$$\delta(G) = h(G_i) - h(G_d) = -1 = -1$$

Si $\delta(G) = -1$: doble rotación, empezando por una rotación izquierda sobre G para hacer $\delta(G) > -1$ y luego rotación derecha sobre x.

Operaciones en árboles AVL: **balanceo**



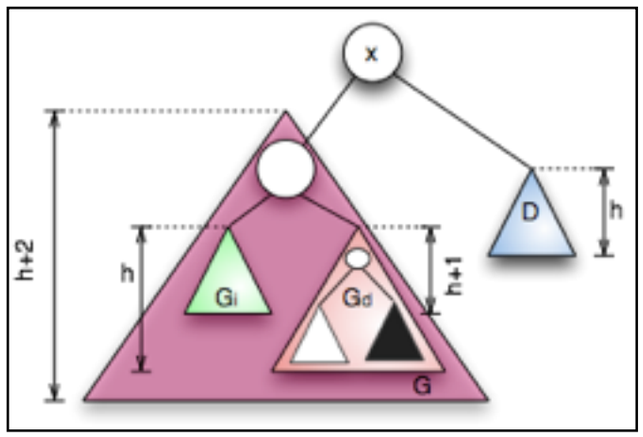


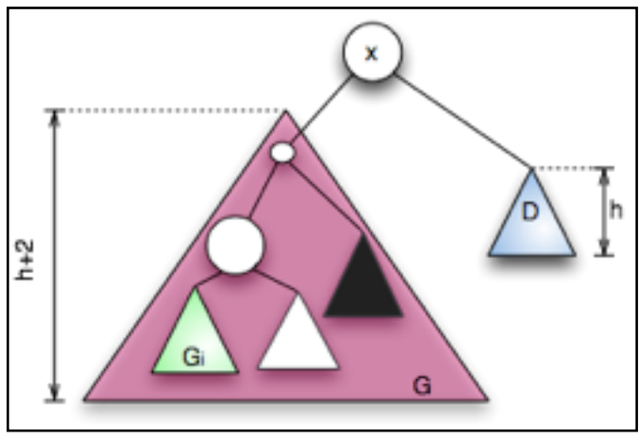
$$\delta(x) = h(G) - h(D) = 2$$

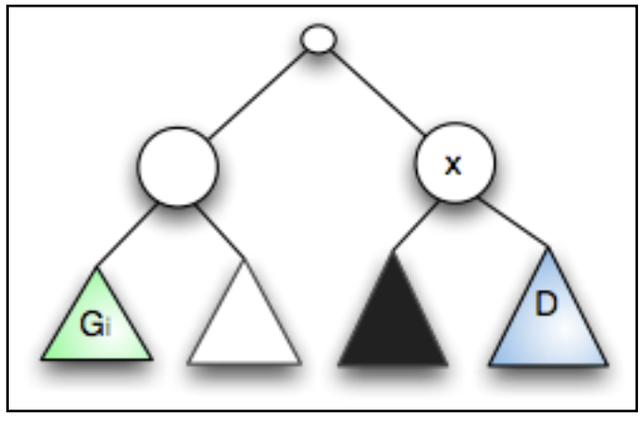
$$\delta(G) = h(G_i) - h(G_d) = -1 = -1$$

Si $\delta(G) = -1$: doble rotación, empezando por una rotación izquierda sobre G para hacer $\delta(G) > -1$ y luego rotación derecha sobre x.

Operaciones en árboles AVL: **balanceo**





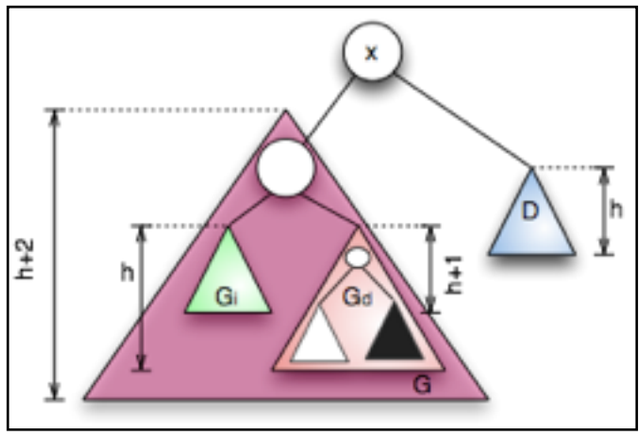


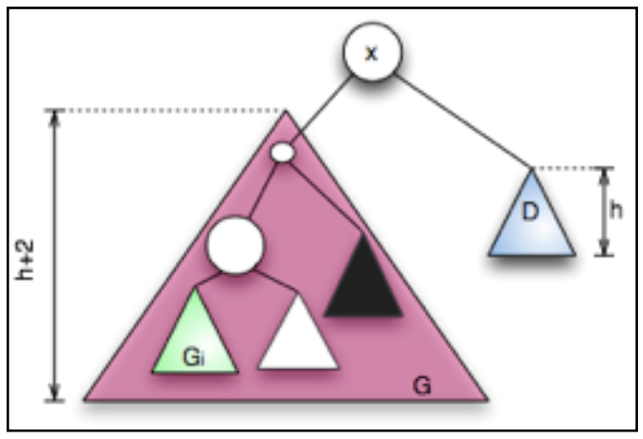
$$\delta(x) = h(G) - h(D) = 2$$

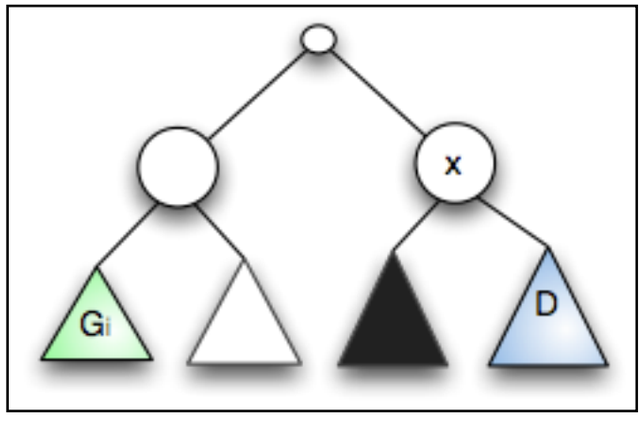
$$\delta(G) = h(G_i) - h(G_d) = -1 = -1$$

Si $\delta(G) = -1$: doble rotación, empezando por una rotación izquierda sobre G para hacer $\delta(G) > -1$ y luego rotación derecha sobre x.

Operaciones en árboles AVL: **balanceo**







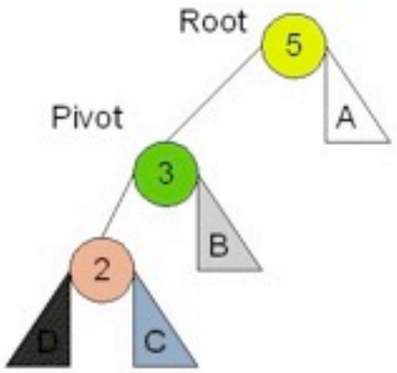
$$\delta(x) = h(G) - h(D) = 2$$

$$\delta(G) = h(G_i) - h(G_d) = -1 = -1$$

Si $\delta(G) = -1$: doble rotación, empezando por una rotación izquierda sobre G para hacer $\delta(G) > -1$ y luego rotación derecha sobre x.

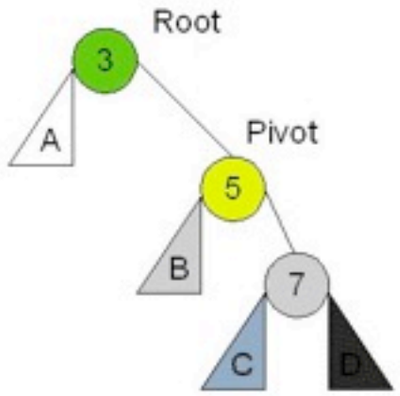
Si D es más alto basta proceder de forma simétrica.

Left Left Case



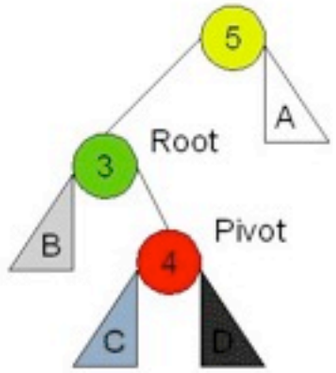
Right Rotation

Right Right Case



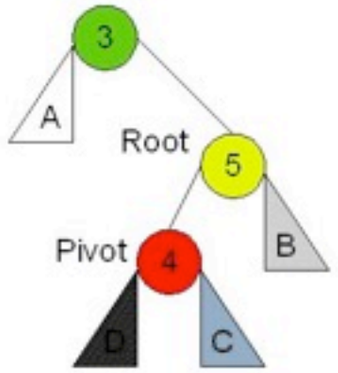
Left Rotation

Left Right Case

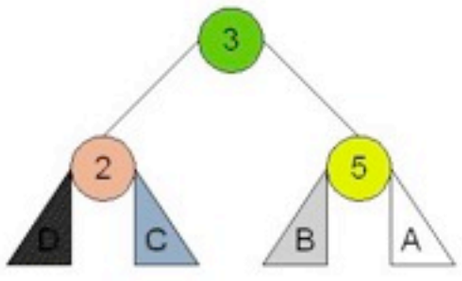


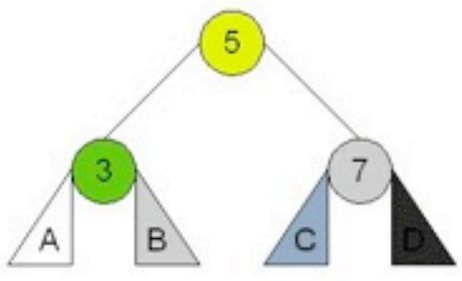
Left Rotation

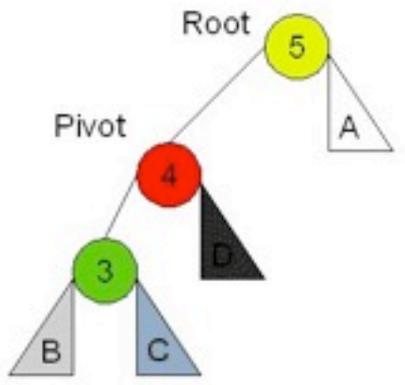
Right Left Case



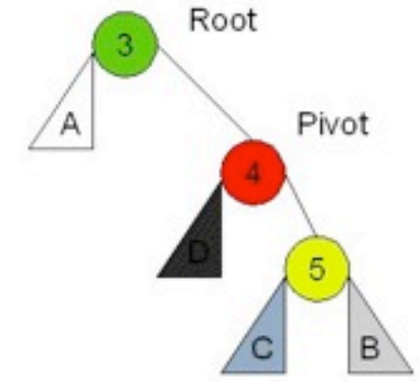
Right Rotation



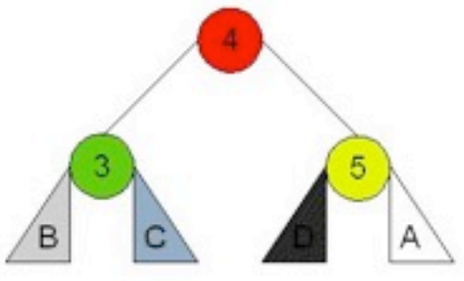


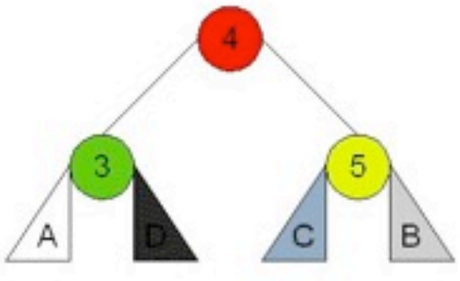


Right Rotation



Left Rotation





Operaciones en árboles AVL: **balanceo**

Operaciones en árboles AVL: **balanceo**

- Después de una inserción en un árbol AVL basta con hacer una o dos rotaciones para re-equilibrar el árbol.

Operaciones en árboles AVL: **balanceo**

- Después de una inserción en un árbol AVL basta con hacer una o dos rotaciones para re-equilibrar el árbol.
- La operación de inserción/balance de un AVL con n nodos se realiza en un tiempo $O(\log_2(n))$.

Operaciones en árboles AVL: **eliminación**

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.
  - si se trata de una hoja, la eliminamos.

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.
  - si se trata de una hoja, la eliminamos.
  - si no, remplazamos el nodo por su nodo sucesor y suprimimos el sucesor.

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.
  - si se trata de una hoja, la eliminamos.
  - si no, remplazamos el nodo por su nodo sucesor y suprimimos el sucesor.
- Para re-equilibrar realizamos rotaciones y dobles rotaciones a lo largo del camino que lleva de la hoja donde se realizó la eliminación a la raíz.

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.
  - si se trata de una hoja, la eliminamos.
  - si no, remplazamos el nodo por su nodo sucesor y suprimimos el sucesor.
- Para re-equilibrar realizamos rotaciones y dobles rotaciones a lo largo del camino que lleva de la hoja donde se realizó la eliminación a la raíz.
- El número de rotaciones o dobles rotaciones necesarias es menor o igual a la altura del árbol (se realiza a lo más, una por nivel)

Operaciones en árboles AVL: **eliminación**

- Se realiza de la misma manera que en un BST:
  - bajamos en el árbol a partir de la raíz para buscar el nodo que contiene la llave a suprimir.
  - si se trata de una hoja, la eliminamos.
  - si no, remplazamos el nodo por su nodo sucesor y suprimimos el sucesor.
- Para re-equilibrar realizamos rotaciones y dobles rotaciones a lo largo del camino que lleva de la hoja donde se realizó la eliminación a la raíz.
- El número de rotaciones o dobles rotaciones necesarias es menor o igual a la altura del árbol (se realiza a lo más, una por nivel)
- La operación de eliminación/balance de un AVL con n nodos se realiza en un tiempo $O(\log_2(n))$.

Árboles AVL

Árboles AVL

- Árboles binarios de búsqueda que verifican una propiedad de equilibrio suplementaria.

Árboles AVL

- Árboles binarios de búsqueda que verifican una propiedad de equilibrio suplementaria.
- El mantenimiento de esta propiedad no aumenta el costo de las operaciones de búsqueda, inserción o supresión en el árbol.

Árboles AVL

- Árboles binarios de búsqueda que verifican una propiedad de equilibrio suplementaria.
- El mantenimiento de esta propiedad no aumenta el costo de las operaciones de búsqueda, inserción o supresión en el árbol.
- Permite garantizar que la altura del árbol AVL permanezca logarítmica a su número de nodos.

Árboles AVL

- Árboles binarios de búsqueda que verifican una propiedad de equilibrio suplementaria.
- El mantenimiento de esta propiedad no aumenta el costo de las operaciones de búsqueda, inserción o supresión en el árbol.
- Permite garantizar que la altura del árbol AVL permanezca logarítmica a su número de nodos.
- Estructura mas compleja de implementar que un BST pero solo ofrece ventajas.

El apuntador *this*, para saber quién es quien

```
class clase {
  public:
   clase() {}
   void EresTu(clase& c) {
      if(&c == this) cout << "Sí, soy yo." << endl;
      else cout << "No, no soy yo." << endl;
   }
};

int main() {
   clase c1, c2;

   c1.EresTu(c2);
   c1.EresTu(c1);

   return 0;
}
```

El apuntador *this* para diferenciar variables

```
struct X {
private:
   int a;
public:
   void Set_a(int a) {
      // The 'this' pointer is used to retrieve 'xobj.a'
      // hidden by the automatic variable 'a'
      this->a = a;
   }
    void Print_a() { cout << "a = " << a << endl; }
};
```

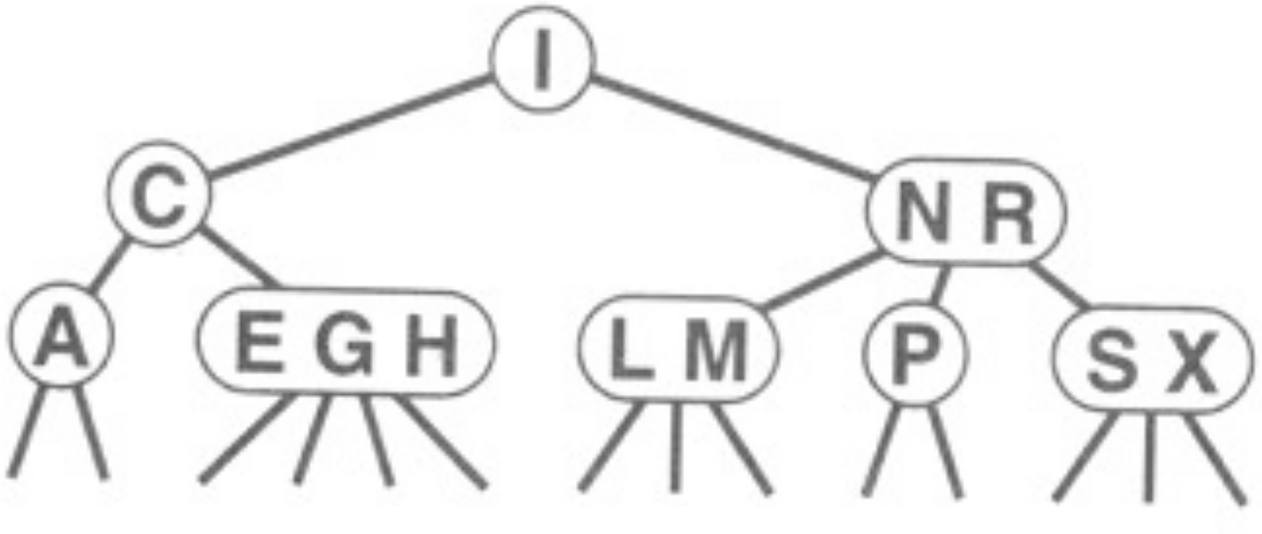
El apuntador *this* para usar funciones que requieren un apuntador

```
struct X {
private:
   int a;
public:
   void reset(X *obj) {
      obj->a = -1;
   }

   void empieza(void){
      reset(this);
      limpiaMemoria(this);
   }

};
```

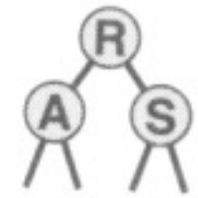
Árboles Top-down 2-3-4



O es un arbol vacio o tiene 3 tipos de nodos: 2-node con 1 llave y dos hijos, 3-node con 2 llaves y 3 hijos o 4-node con 3 llaves y 4 hijos. El orden de los nodos hijos es intuitivo (el hijo de enmedio es menor que el padre derecho y mayor que el padre izquierdo).

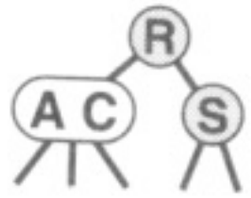
- La distancia de la raíz a cada apuntador nulo es siempre la misma.

Árboles Top-down 2-3-4

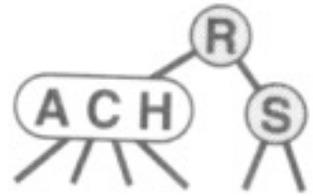


Inserción:

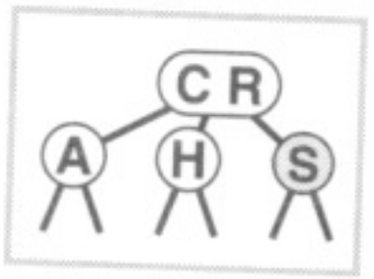
-Convertir de 2-nodo a un 3-node.

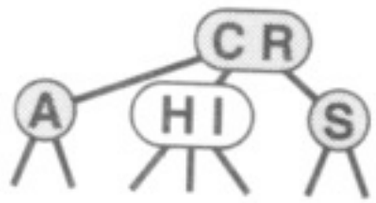


-Convertir de 3-nodo a un 4-node.

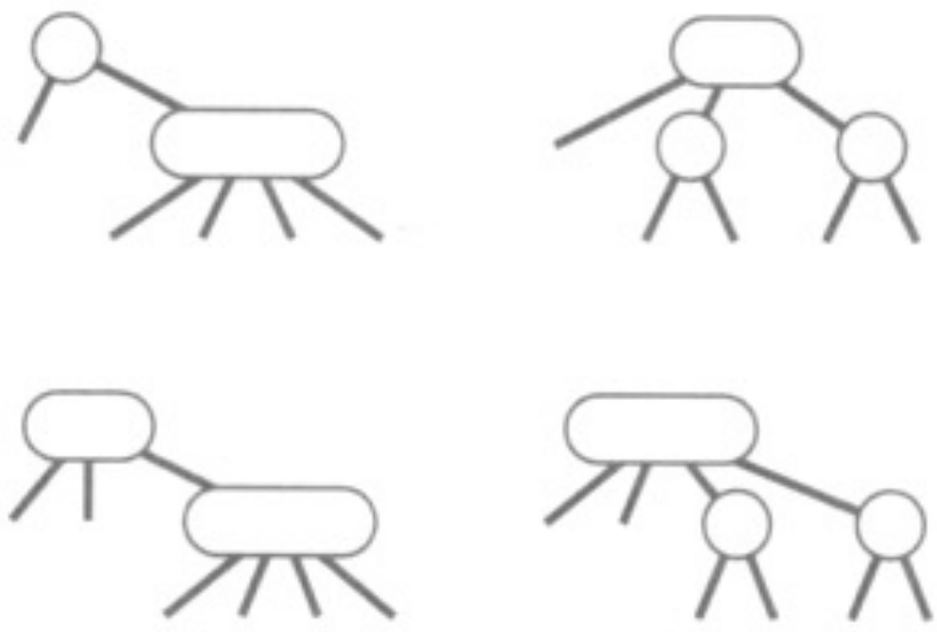


-Partir un 4-node pasando la llave de en medio al padre (convirtiendolo en 3-node) y luego covirtiendo un 2-node en 3-node





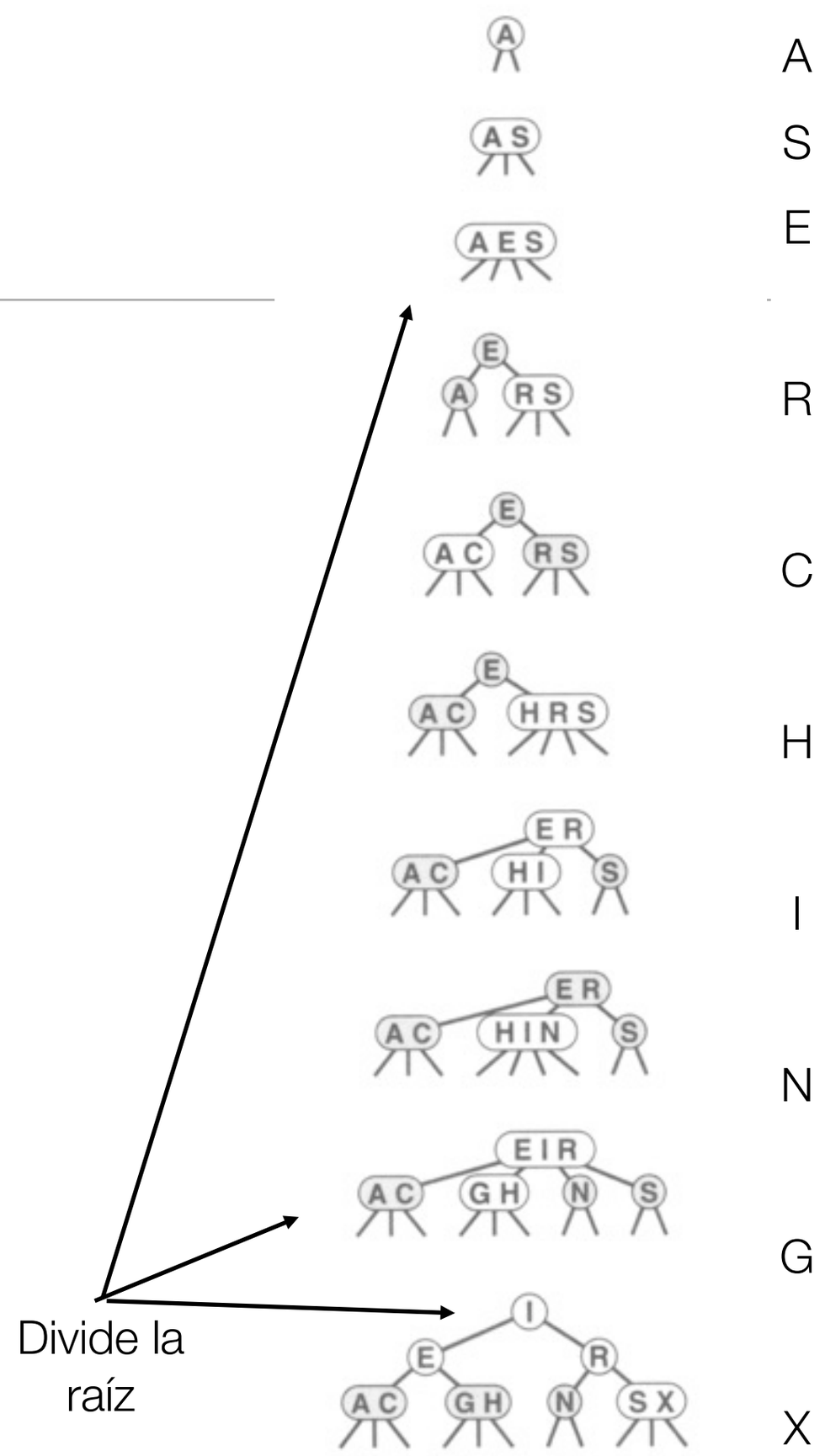
Árboles Top-down 2-3-4



Podemos partir cualquier 4-node que no es el hijo de otro 4-node en 2-node's, pasando la llave de en medio a su padre.

Árboles Top-down 2-3-4 (construccion)

En la construcción, dividimos cualquier 4-node que encontramos en el camino, asegurando así que hay espacio para para el nuevo Item hasta abajo.



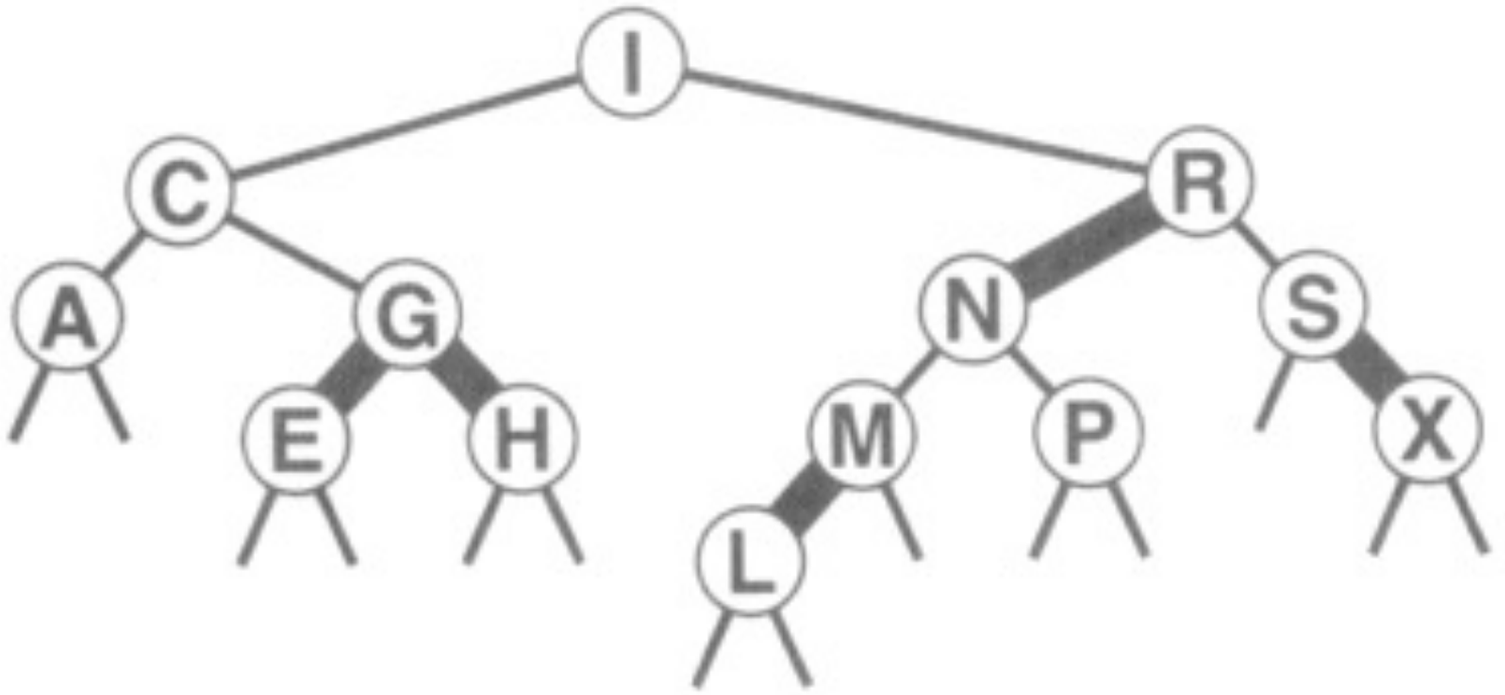
Árboles rojo-negros

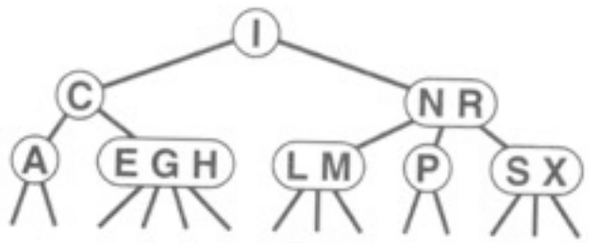
- Estos están basados en los árboles Top down 2-3-4.

El color del nodo es el color del link que llega a él desde su padre.

De tal forma que las conexiones internas son Rojas (anchas) y las conexiones externas son negras (delgadas).

Árboles rojo-negros





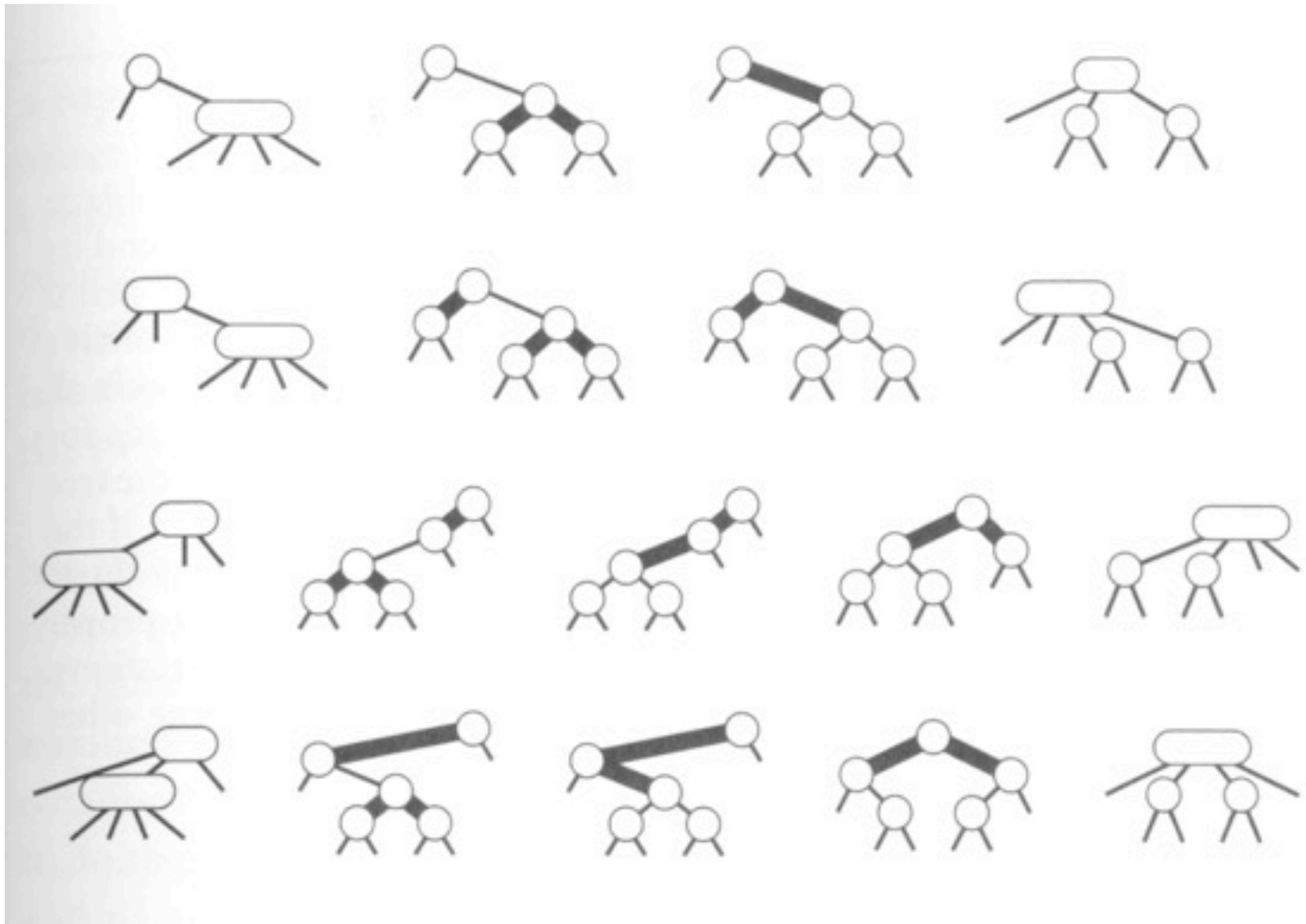
Arbol rojo-negro, (i) mismo que el anterior.

(ii)La búsqueda es la misma que en un BST.

Cualquier camino de raiz-hoja tiene 2 conexiones negras.

Árboles rojo-negros

- Divsión de los 4-node en un árbol rojo negro.



(Si un nodo tiene 2 hijos rojos es un 4-node)

Las operaciones comprenden el cambiar de color los nodos(ligas) y hacer 1 o 2 rotaciones.