Árboles Avanzados

Mauricio Avilés

Contenido

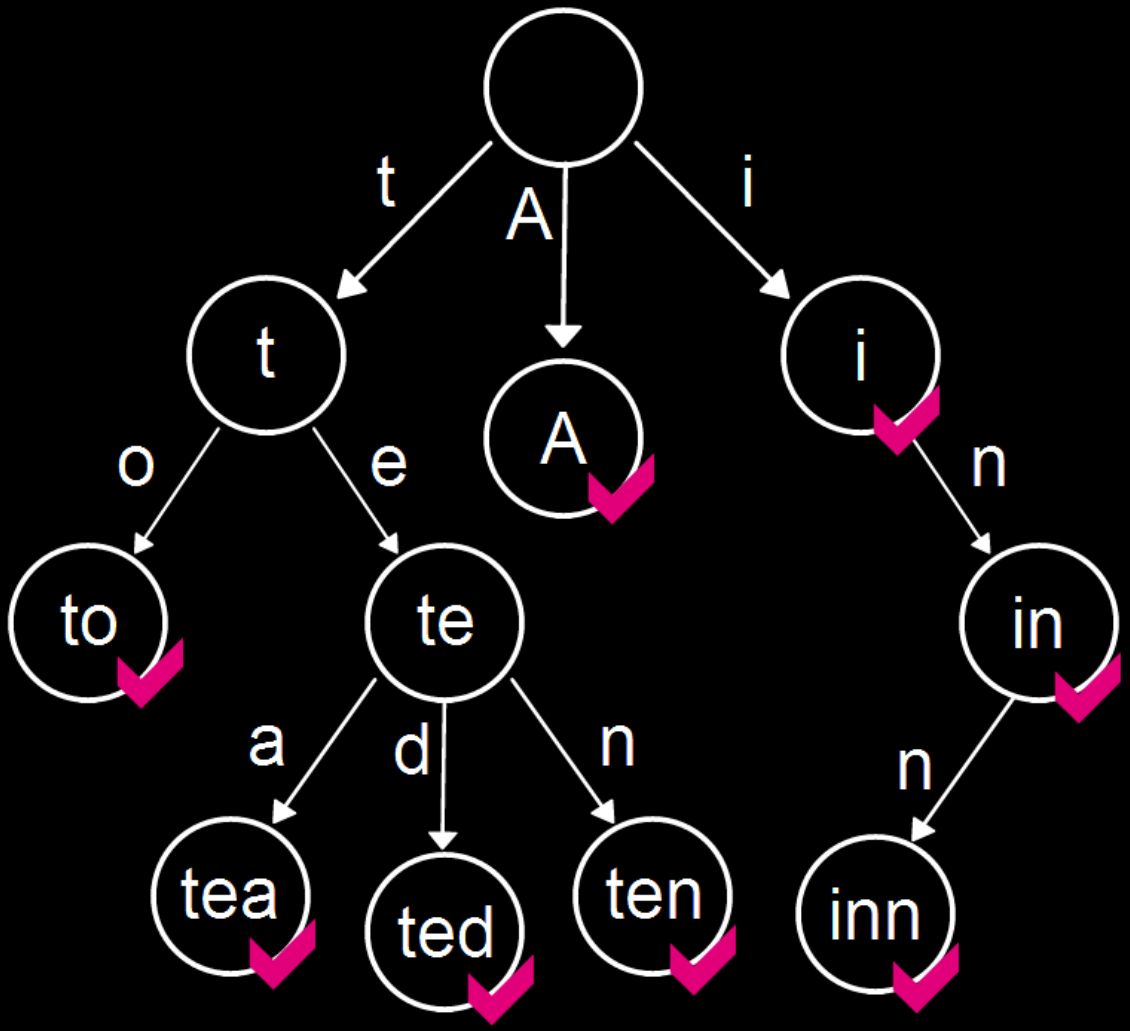
- Tries
- Árboles AVL
- Árboles Splay

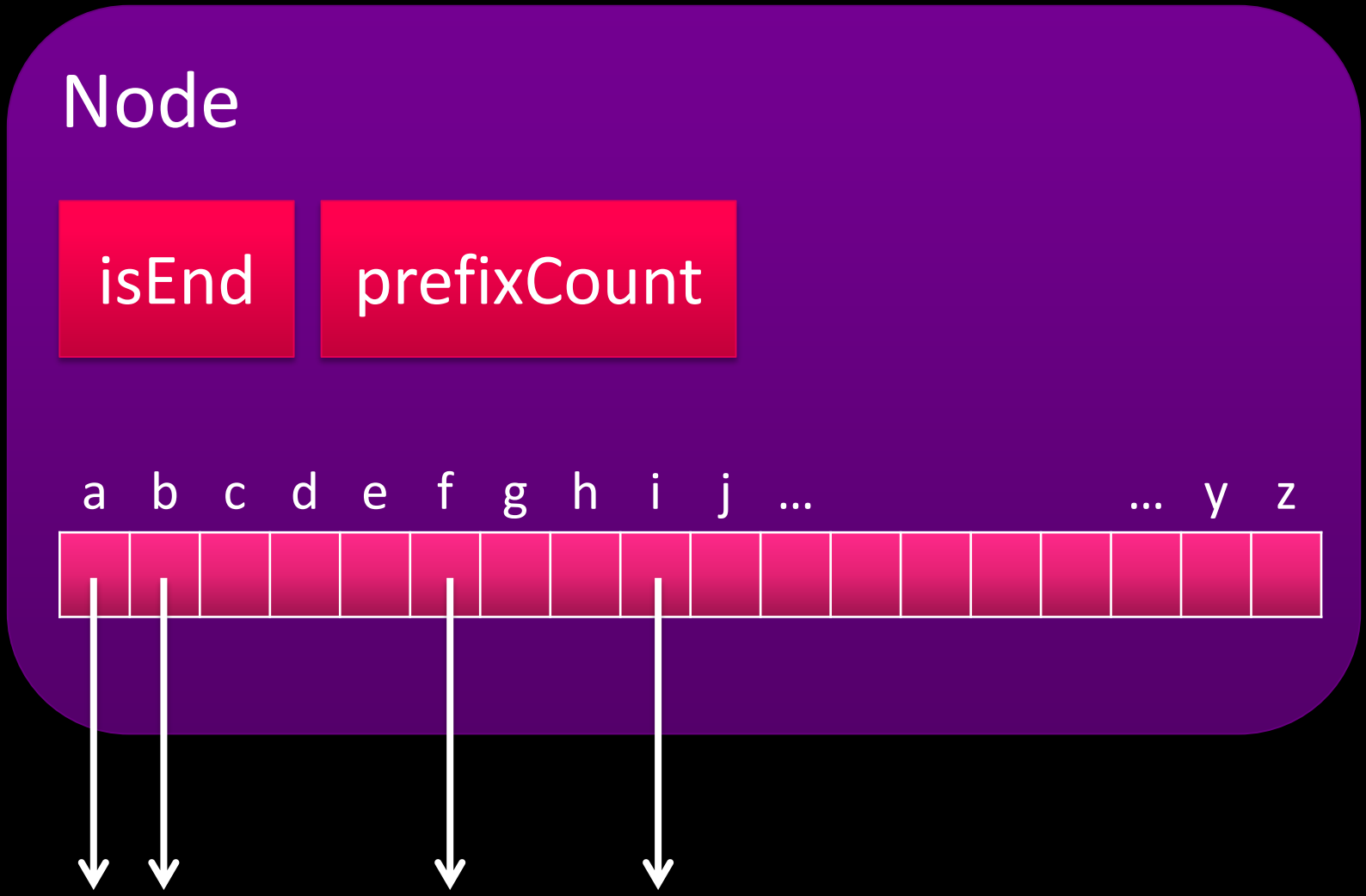
Trie

- También llamado árbol de prefijos
- Estructura de árbol para almacenar un conjunto dinámico o un arreglo asociativo donde las llaves son strings
- El nombre proviene de la palabra ‘reTRIEval’ (recuperación), y fue creado por Edward Fredkin
- Es útil para realizar búsquedas eficientes en repositorios de datos muy voluminosos

- Ningún nodo almacena la llave asociada, si no que la posición del nodo define la llave asociada
- Todos los descendientes de un nodo tienen un prefijo común que es el string asociado a ese nodo
- No hay valores asociados a todos los nodos
- Sólo las hojas y algunos nodos internos van a tener un valor asociado

“A”, “to”, “tea”, “ted”, “ten”, “i”, “in”, “inn”




Node
isEnd
prefixCount
a b c d e f g h i j …
… y z

```
class Node {
private:
  bool isEnd;
  int prefixCount;
  Node* child[ALPHABET_SIZE];

public:
  Node() { prefixCount = 0; isEnd = false; }
  ~Node() {}
  bool getIsEnd() { return isEnd; }
  void setIsEnd(bool i) { isEnd = i; }
  int getPrefixCount() { return prefixCount; }
  void incPrefixCount() { prefixCount++; }
  void decPrefixCount() { prefixCount--; }
  Node* findChild(char c) { … }
  void appendChild(char c) { … }
}
```

```
class Trie {
private:
  Node* root;

public:
  Trie() { root = new Node(); }
  ~Trie() { … }
  void insert(string s) { … }
  bool search(string s) { … }
  void delete(string s) { … }
  int wordsWithPrefix(string prefix) { … }
}
```

```
void insert(string s) {
  Node* curr = root;
  curr->incPrefixCount();
  for (int i=0; i < s.length(); i++) {
    if (curr->findChild(s[i]) == NULL)
      curr->appendChild(s[i]);
    curr = curr->findChild(s[i]);
    curr->incPrefixCount();
  }
  curr->setIsEnd(true);
}
```

Ejercicios

- Dibuje los tries generados por las siguientes listas de palabras.
  - si, sabe, sabia, don, dona, diez, dia di, mil, mi, mas
  - cas, casa, cesar, cede, cesarea, casi, de, dar, dio, dior
- Escriba el pseudocódigo y código en C++ para las funciones findChild y appendChild de la clase Node.
- Escriba el pseudocódigo para la función insert de la clase Trie.
- Tomando como referencia el código del insert, escriba el código C++ para la función search de la clase Trie.

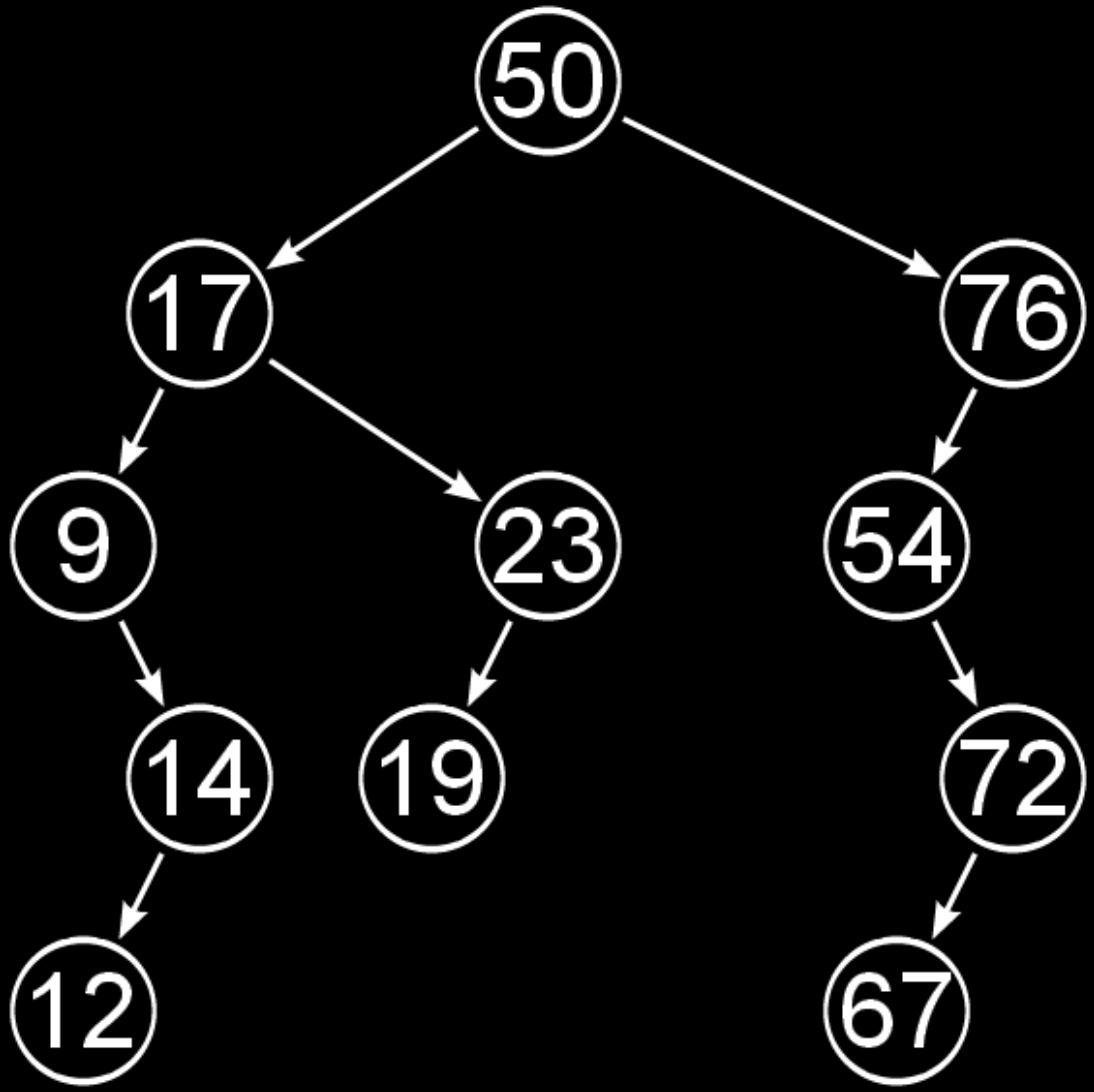
Repaso

- En grupos de máximo 3 con al menos 1 integrante ausente de la clase pasada
- Responder:
  - ¿Qué es un Trie?
  - ¿Cómo se realizan inserciones en un Trie?
  - Dibujar Trie: reloj, red, res, relax, agua, agil, agudo, libro, libre, lindo
  - Explique con sus propias palabras la implementación de un Trie

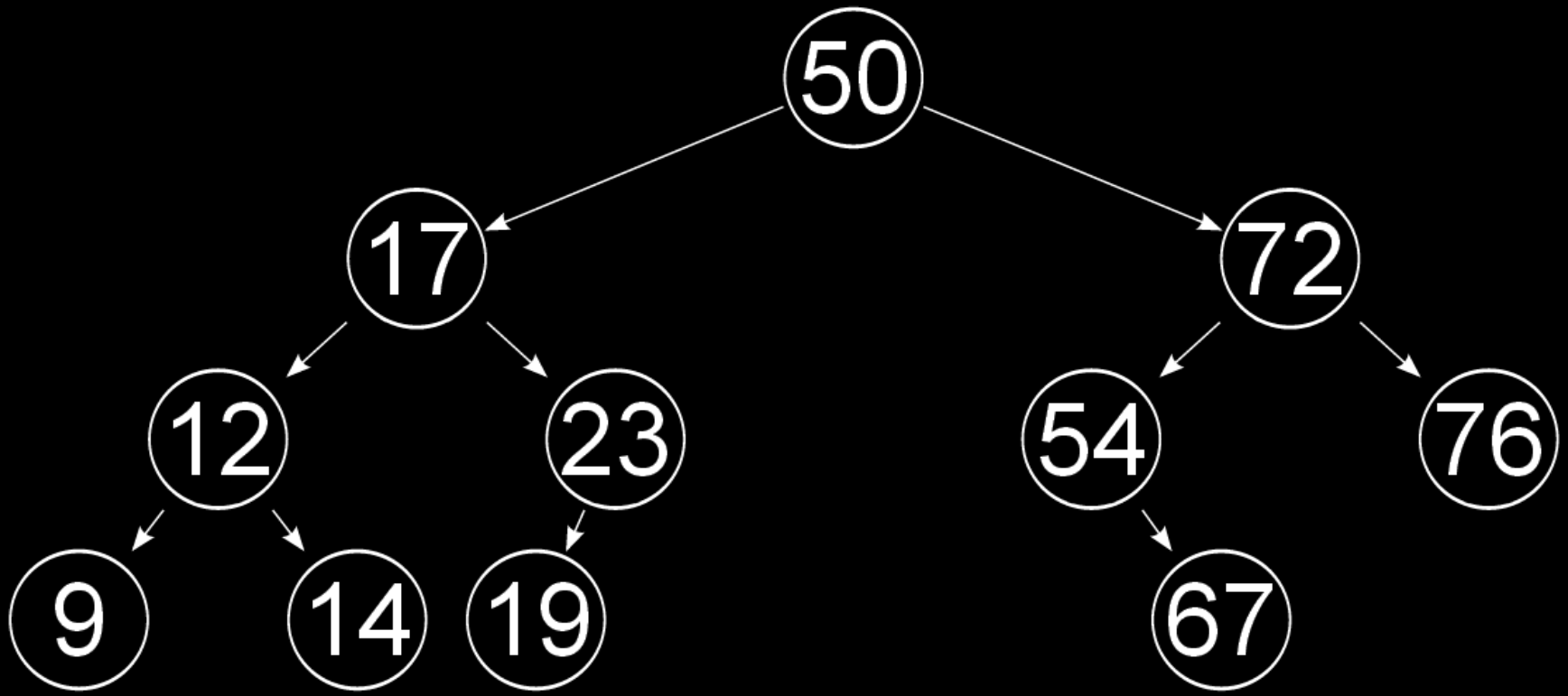
Árboles balanceados

- Un árbol de búsqueda binario obtiene su forma según el orden de las inserciones
- Un árbol desbalanceado da como resultado operaciones de inserción y búsqueda muy costosas
- Al estar desbalanceado tiene una altura excesiva al ser comparada con la cantidad de nodos
- Un heap se mantiene balanceado al mantener la propiedad de árbol completo
- Hacer lo mismo con un BST requiere modificaciones excesivas

Árbol desbalanceado



Mismo árbol balanceado



Árboles AVL

- Primer estructura de datos autobalanceada en ser inventada
- AVL = Adelson-Velskii-Landis (sus inventores)
- Utiliza rutinas de inserción y borrado modificadas para realizar el balanceado
- Para cualquier nodo la altura de sus hijos izquierdo y derecho difieren máximo en uno
- Mientras se mantenga esta propiedad el acceso a los datos se mantiene eficiente

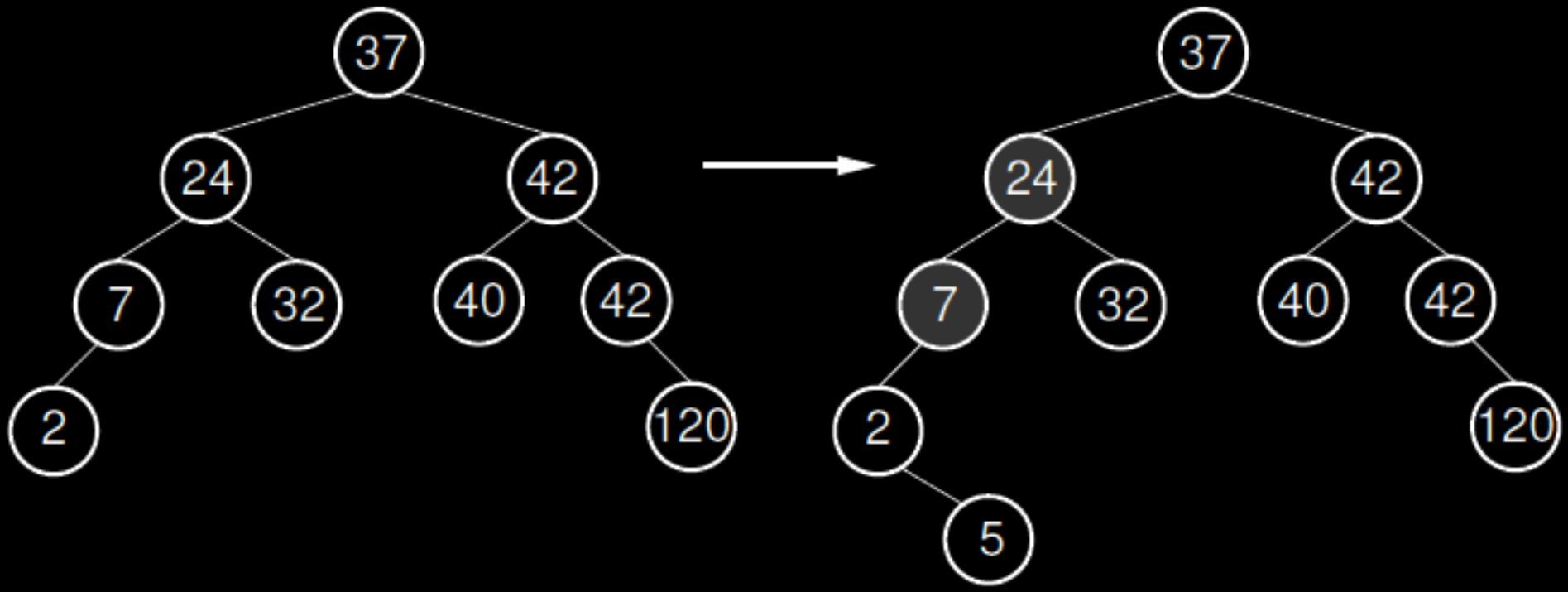
Búsqueda

- La búsqueda de un nodo en un árbol AVL es igual que la búsqueda en un BST
- Diferente a lo que sucede en otros árboles autobalanceados, el AVL no modifica el árbol cuando se busca un nodo específico

Inserción

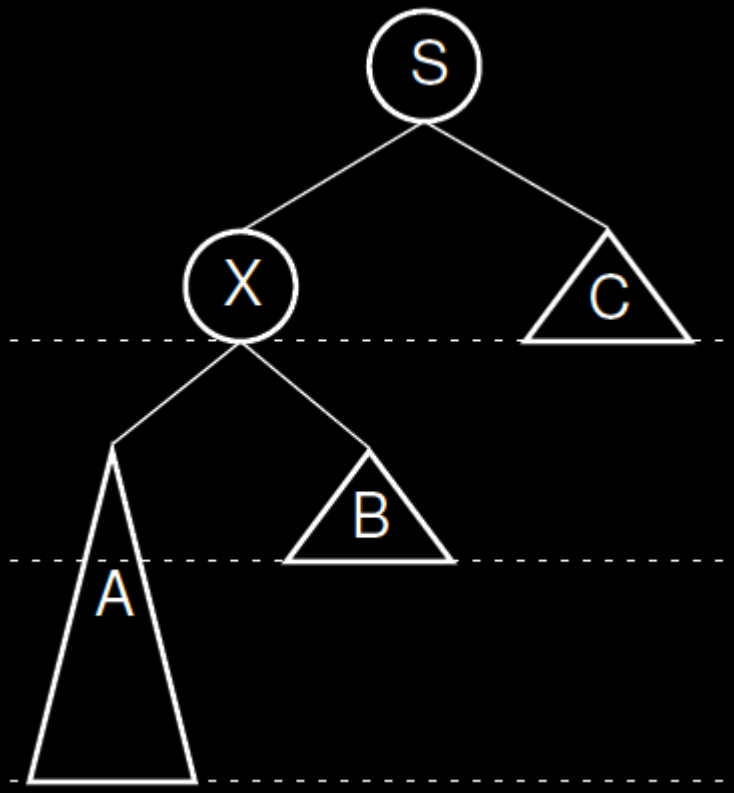
- La inserción en un AVL consta de dos partes
  - Insertar un nodo de la misma forma que en un BST
  - Revisar recursivamente desde el nodo más profundo hasta la raíz si se cumple la condición de balanceo y realizar las rotaciones necesarias

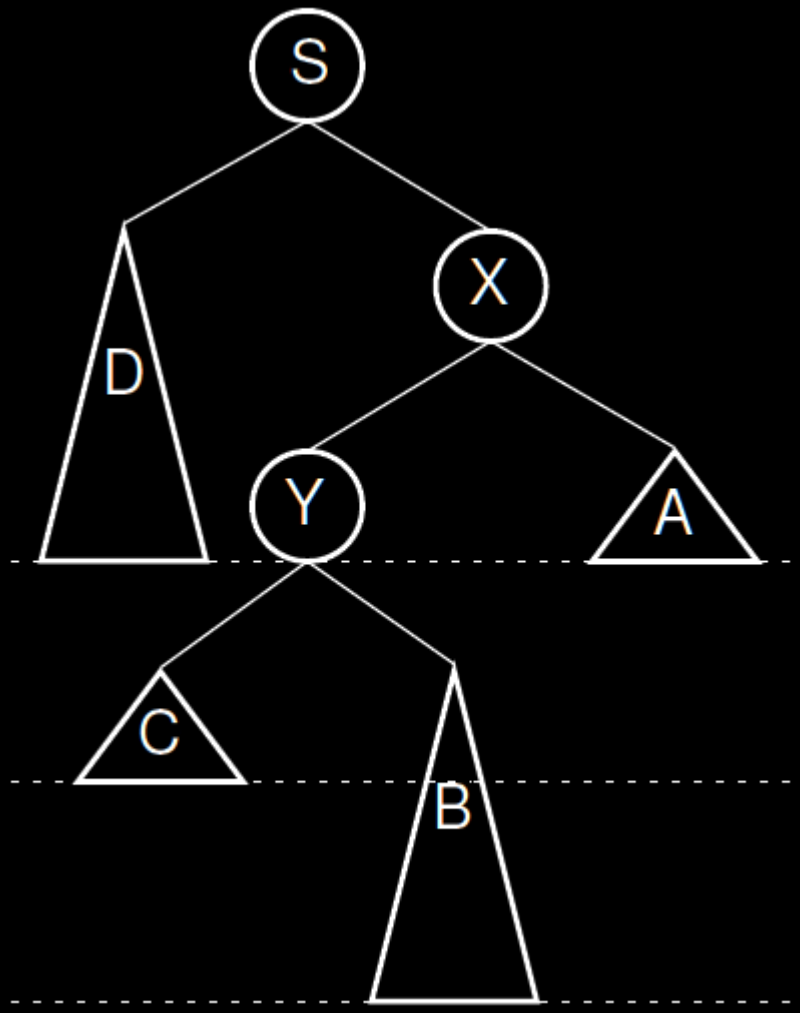
Inserción que desbalancea un árbol



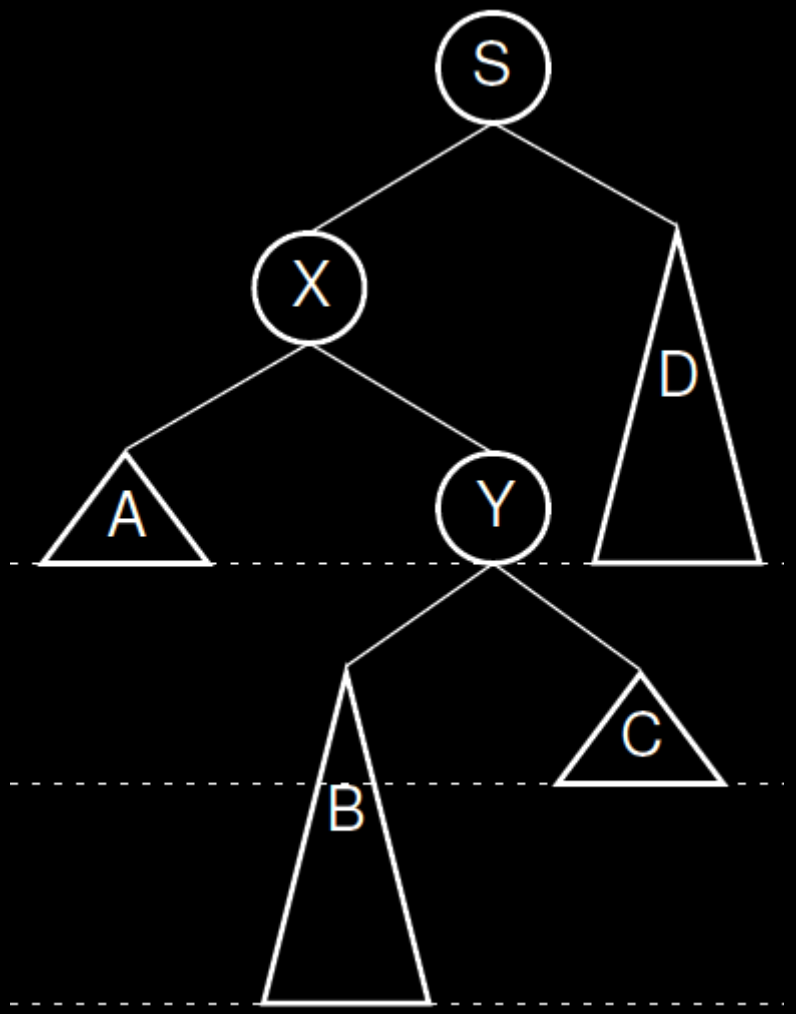
- Si tenemos un AVL, después de una inserción se puede asegurar que el nuevo árbol está desbalanceado por una diferencia de 2 como máximo
- Para el nodo más profundo que se encuentra desbalanceado se pueden dar 4 casos donde se encuentra el nodo extra:
  1. Hijo izquierdo del hijo izquierdo
  2. Hijo derecho del hijo izquierdo
  3. Hijo izquierdo del hijo derecho
  4. Hijo derecho del hijo derecho

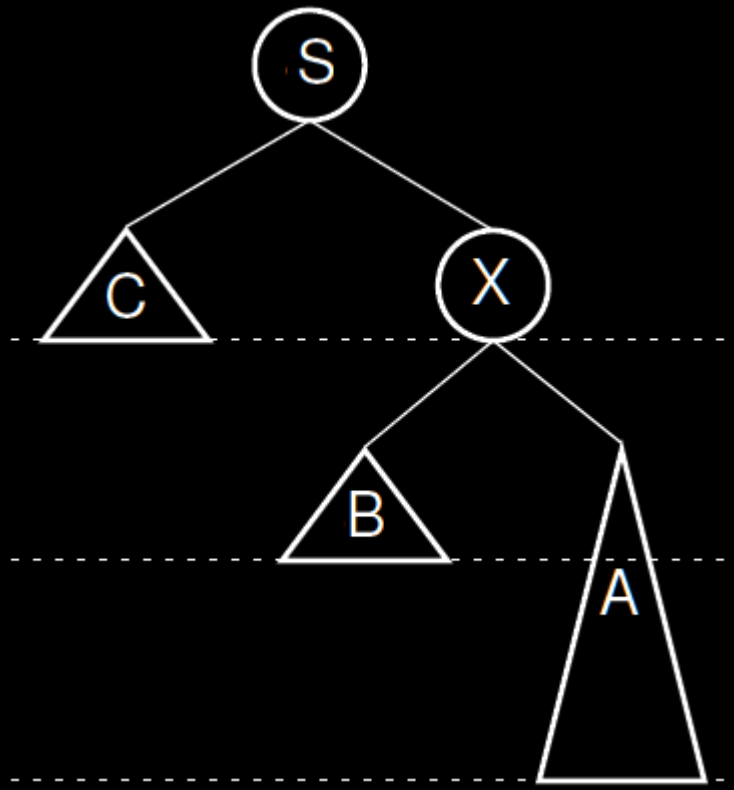
Casos 1 y 2



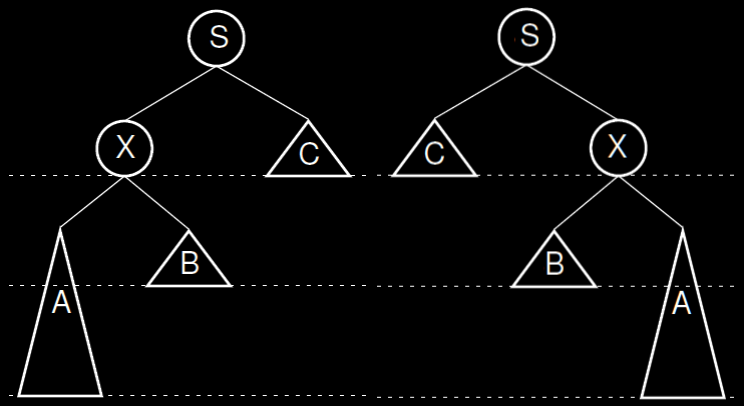


Casos 3 y 4

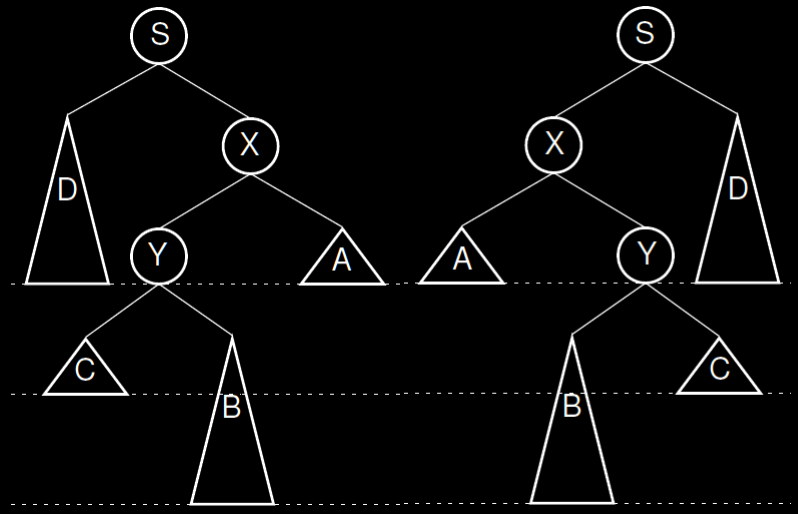




- Los casos 1 y 4 son simétricos

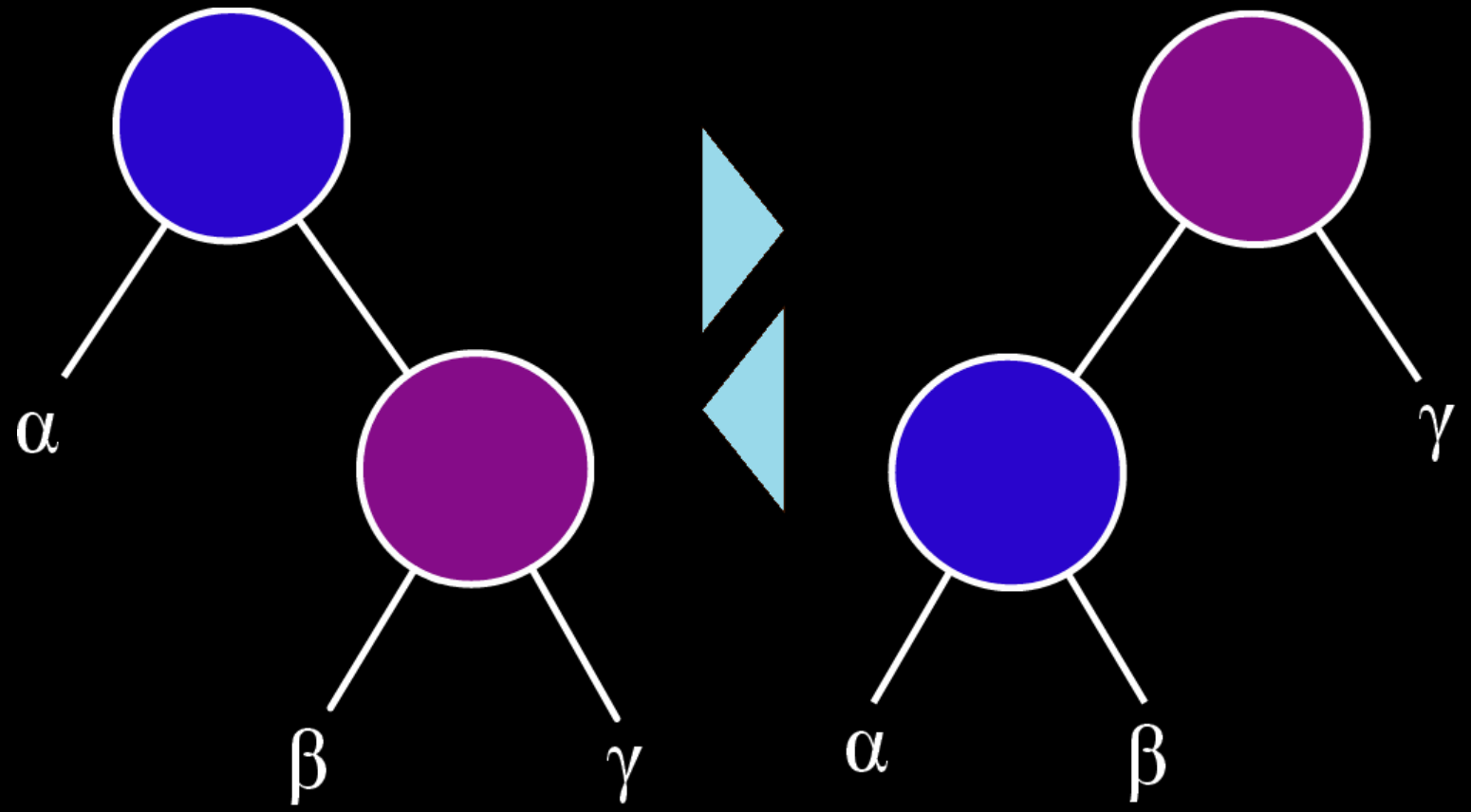


- Los casos 2 y 3 son simétricos



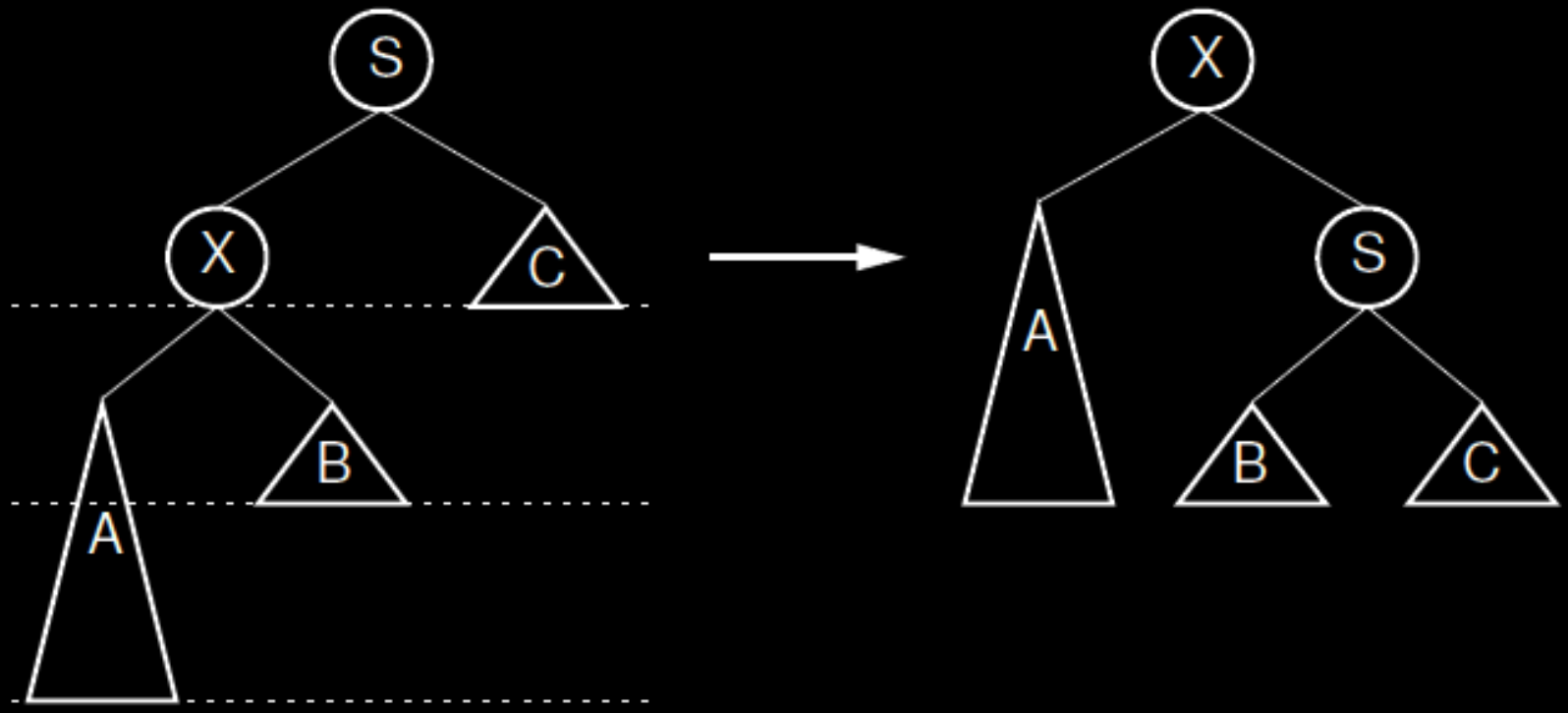
- El problema de balancear los árboles se soluciona con una serie de operaciones denominadas rotaciones
- Los casos 1 y 4 se solucionan con rotaciones simples
- Los casos 2 y 3 se solucionan con rotaciones dobles

Rotación



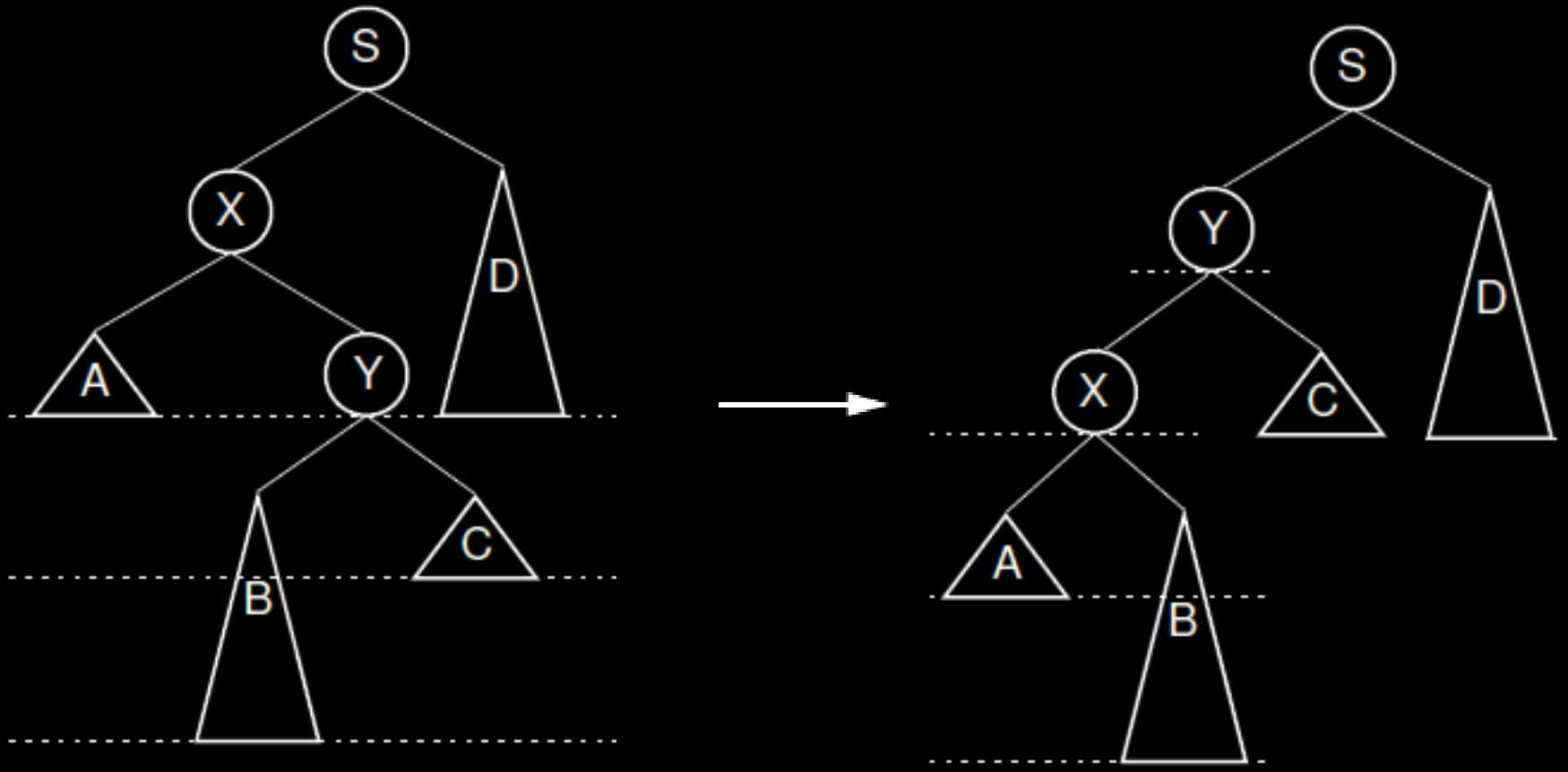
Nótese que $\alpha < \beta < \gamma$ se sigue cumpliendo después de la rotación

Rotación simple, casos 1 y 4



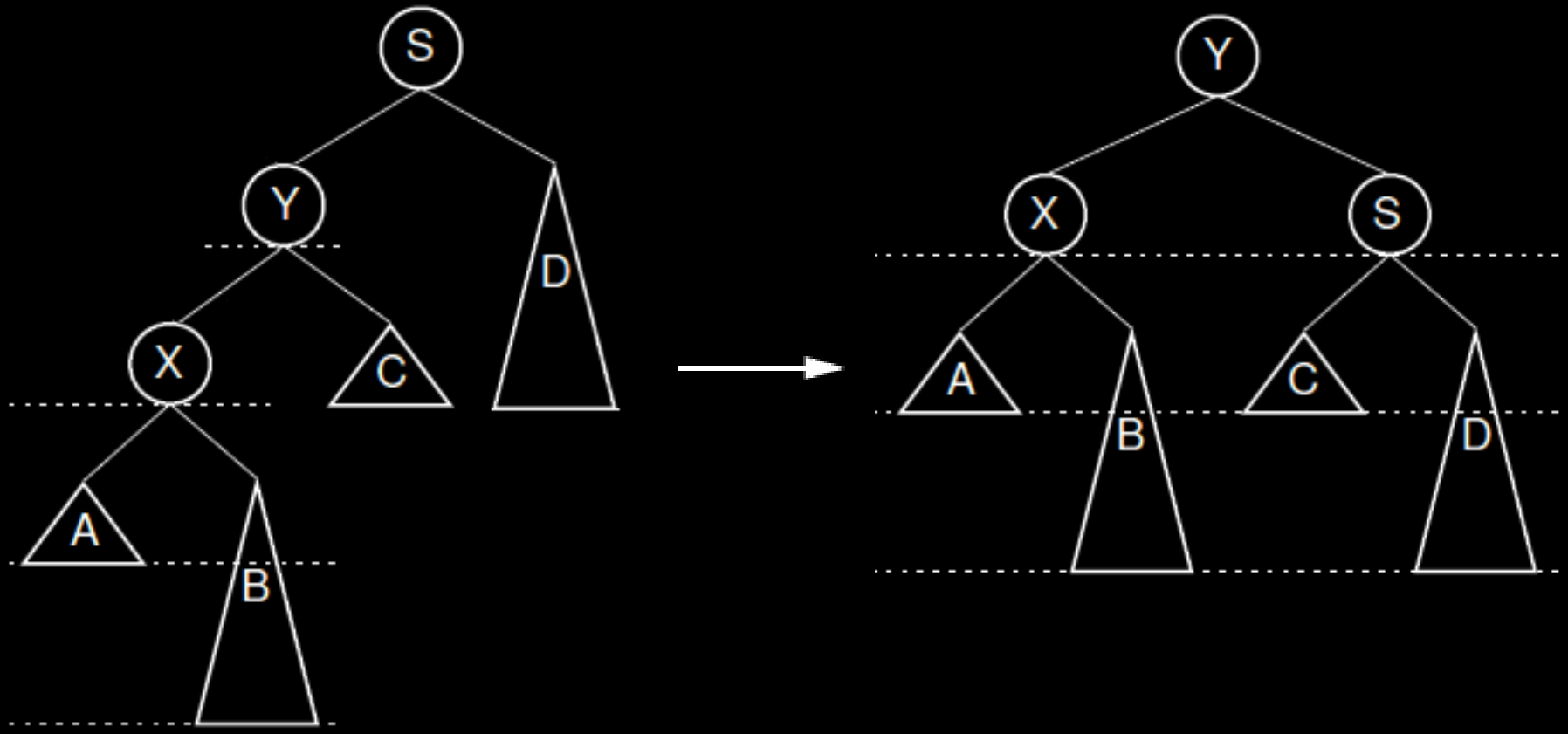
- X pasa a ser la raíz
- S pasa a ser hijo derecho de X
- B cambia de padre y pasa a ser hijo izquierdo de S

Rotación doble, casos 2 y 3



- Rotación simple hacia la izquierda en X
- B pasa a ser hijo derecho de X

Rotación doble, casos 2 y 3



- Rotación simple en S hacia la derecha

[90, 45, 99, 9, 93, 2, 39, 65, 34, 25, 66, 28, 31, 27, 41, 15]

Borrado

- El borrado es igual al borrado de un BST
- Casos:
  - Nodo sin hijos
  - Nodo con un hijo
  - Nodo con dos hijos
- Se realiza el borrado y luego se verifica el balanceado desde el padre del nodo borrado hasta la raíz

Ejercicios

- Realizar las siguientes inserciones y borrados en un árbol AVL
  - Ins.: [27, 67, 31, 76, 65, 74, 38, 7, 30, 80] Bor.: [31, 65, 30]
  - Ins.: [5, 45, 52, 83, 88, 29, 42, 19, 86, 12] Bor.: [29, 42, 19]
  - Ins.: [0, 85, 34, 48, 42, 66, 98, 14, 35, 90] Bor.: [66, 85, 14]
- Analice las siguientes inserciones. ¿Qué sucede en los AVL cuando existen nodos repetidos?
  - [56, 46, 25, 71, 21, 9, 43, 25, 42, 11, 71, 64, 84, 66, 35]
  - Sugiera una posible solución al problema encontrado

Árboles Splay

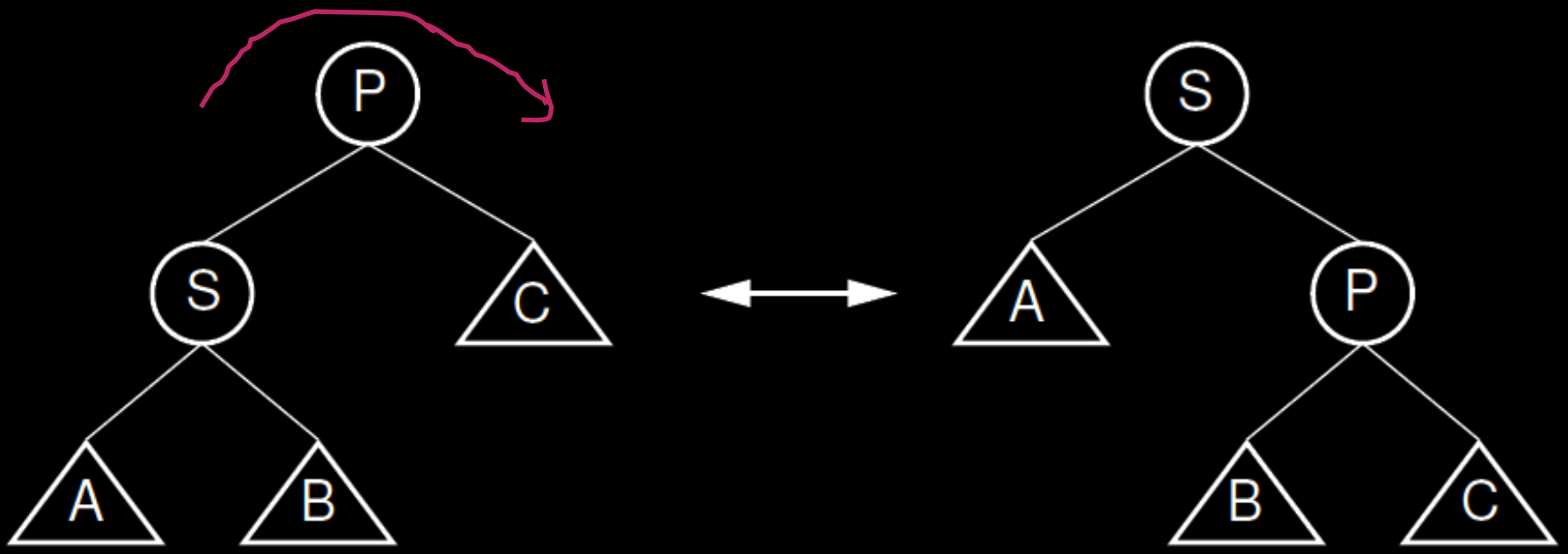
- Se basa en un BST
- Reimplementa la inserción, borrado y búsqueda
- No garantiza eficiencia para estas operaciones individuales
- Garantiza eficiencia para un conjunto de ellas
- Inventados por Daniel Dominic Sleator y Robert Endre Tarjan en 1985

- No es balanceado por altura
- Se basa en la idea de que lo que importa es el costo de una serie de operaciones, no si el árbol en sí está balanceado o no
- Cuando un nodo es accedido el árbol ejecuta una operación de *splay*, que significa esparcir o separar
- El *splaying* mueve al nodo accedido a la raíz del árbol
- Esto mantiene a los nodos utilizados recientemente cerca de la raíz y a los valores parecidos

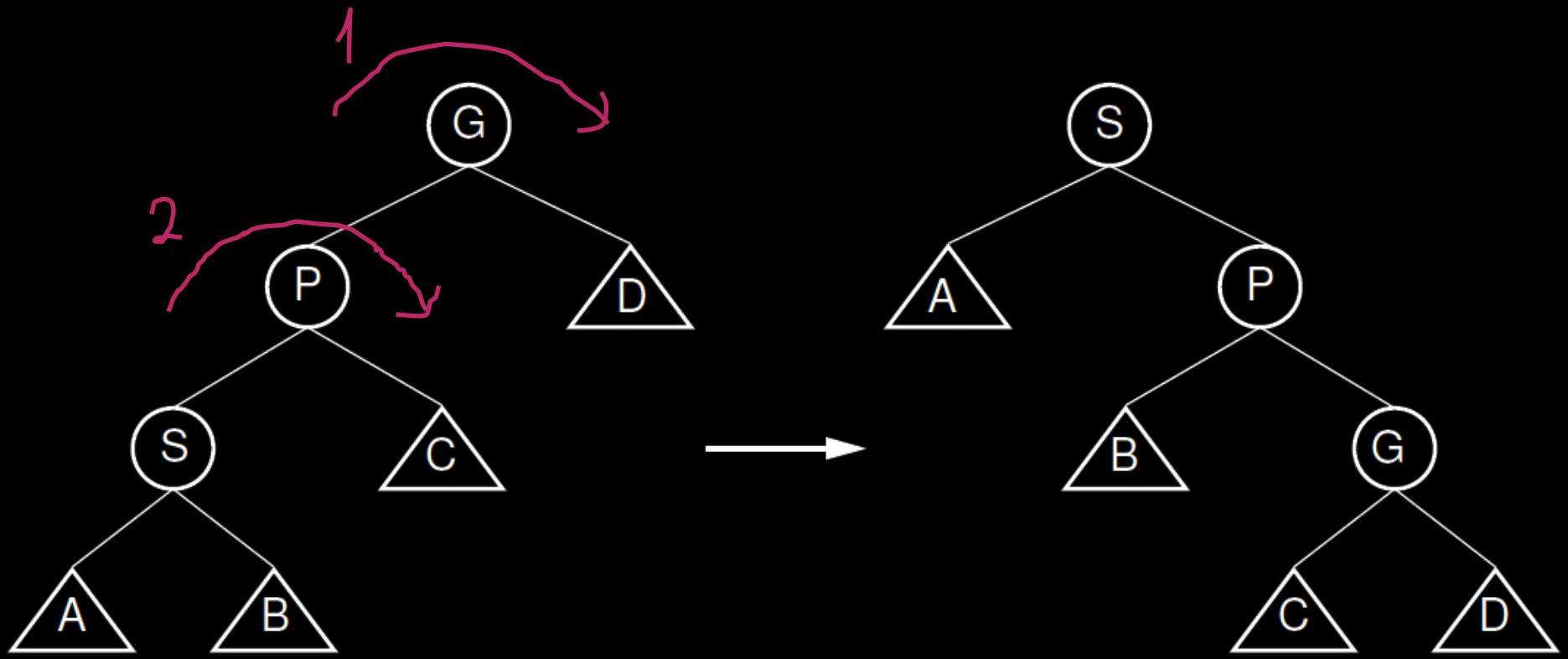
Operación *splay*

- La operación se implementa con una serie de rotaciones
- Casos, S es el nodo en cuestión
  - El padre (P) de S es la raíz
    - Rotación en P (zig)
  - S debe subir al nivel de su abuelo (G)
    - Rotación en G y en P misma dirección (zig-zig)
    - Rotación en P y en G en diferentes direcciones (zig-zag)
- Las rotaciones deben repetirse hasta que el nodo quede en la raíz

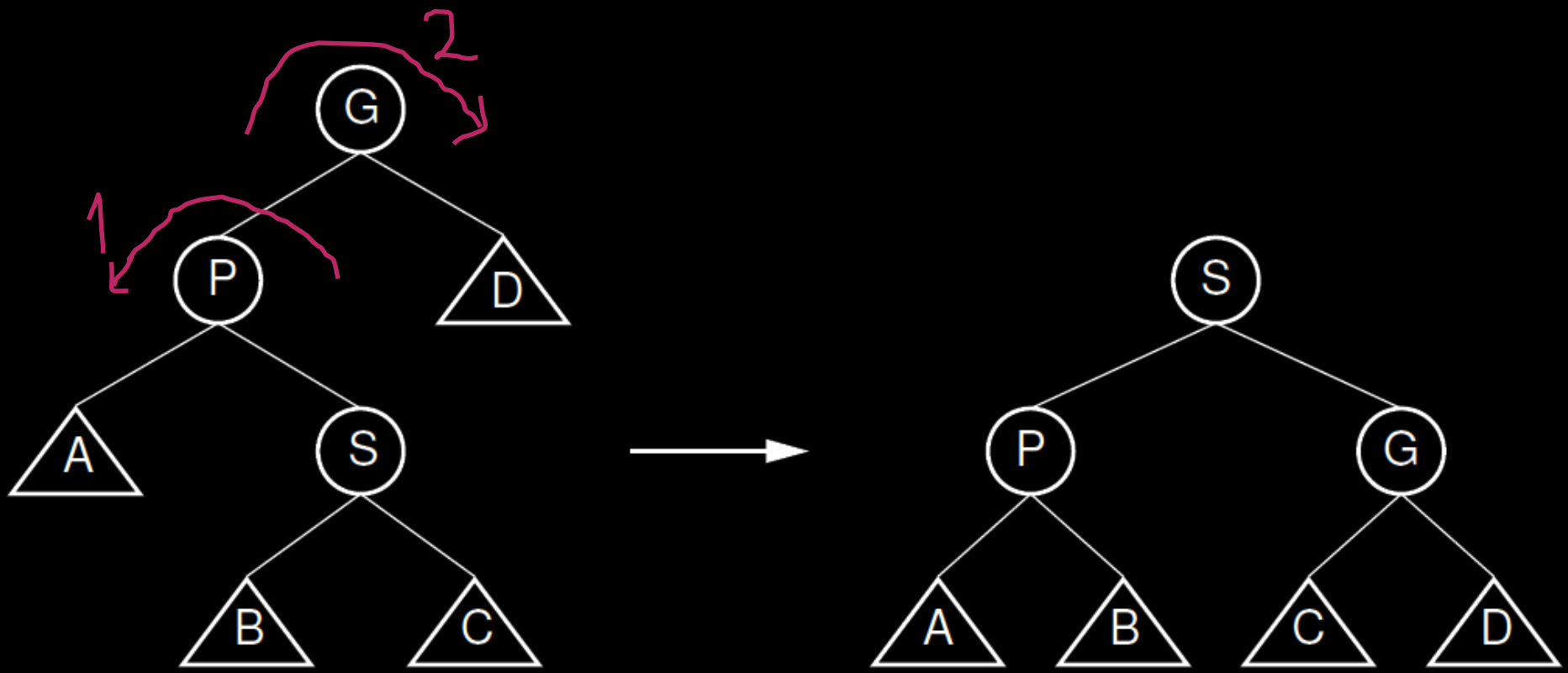
zig



zig-zig



zig-zag



Inserción

- Se inserta un nodo nuevo en el árbol de la misma forma que un BST
- Luego se ejecuta el *splay* de ese nodo

Borrado

- Método 1:
  - Hacer *splay* del nodo a borrar
  - Borrar igual que un BST
- Método 2:
  - Borrar igual que un BST
  - Hacer *splay* del padre del nodo borrado

Búsqueda

- Se busca el nodo igual que en un BST
- Si se encuentra, se hace *splay* del nodo
- Si no se encuentra, se hace *splay* del último nodo encontrado

[50, 75, 45, 47, 90, 80, 60, 23, 51, 89, 69, 3]

Ejercicios

- Realizar las siguientes inserciones y borrados en un árbol *splay*
  - Ins.: [35, 99, 15, 0, 69, 45, 18, 91, 72, 12, 65, 14, 14, 62, 73]
    Bor.: [99, 45, 73]
  - Ins.: [36, 15, 74, 2, 85, 2, 53, 94, 92, 68, 32, 98, 41, 40, 28]
    Bor.: [94, 2, 28]
  - Ins.: [85, 89, 99, 42, 96, 0, 28, 5, 90, 87, 47, 40, 91, 77, 86]
    Bor.: [28, 40, 89]

Lecturas

- Shaffer – Capítulo 13
- Tamassia – Capítulo 10

Árboles Avanzados

Mauricio Avilés