

Ayudantía ABB y AVL

MATERIAL DE APOYO



1. Cheatsheet C (notion resumen)
2. Ejercicios de práctica C
(útil - quizá - para T1 ver el de ABB)
3. Cápsulas de semestres pasados

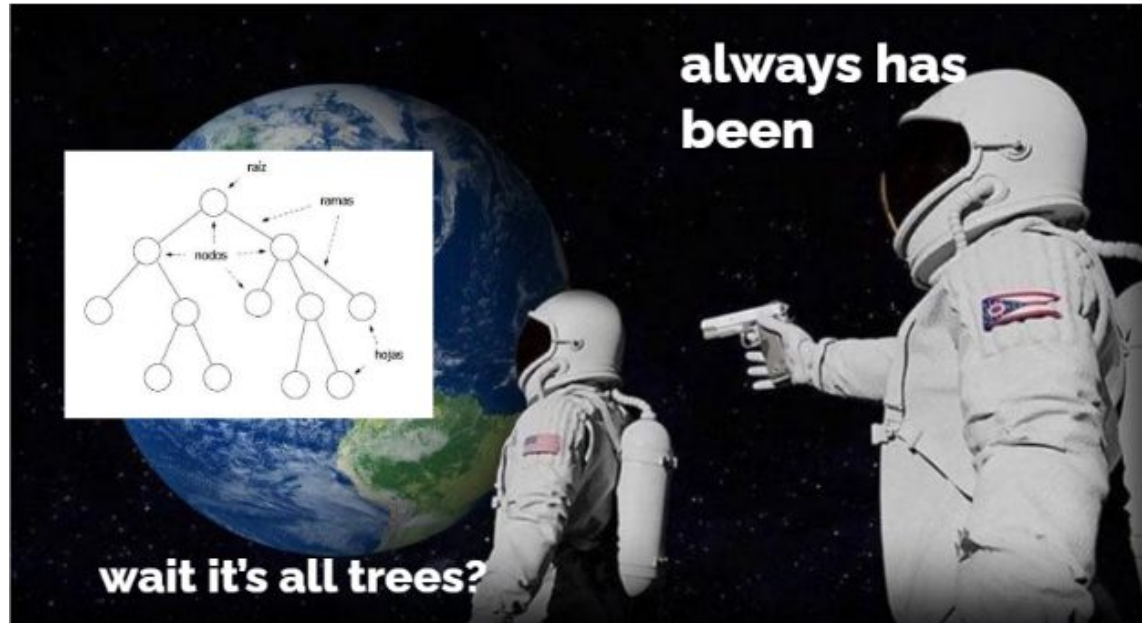


Sonrisa coqueta

Dónde encuentro esto?

Links en ReadMe carpeta "Ayudantías" del repo

Menti:



Árbol binario de búsqueda (ABB)

- En inglés **BST**
- Cada nodo tiene asociados **dos ABB** , **mediante punteros**
- Sea x el nodo, entonces:
 - $x.key = llave$
 - $x.value = valor$
 - $x.p$ es el padre del nodo, como máximo puede tener 1. En caso de no tener padre es la raíz
 - **tiene a lo más dos hijos**
 - $x.left$ es el nodo izquierdo el cual cumple que **$x.left < x$**
 - $x.right$ es el nodo derecho el cual cumple que **$x.right > x$**
 - en caso de **$x.right == null$ y $x.left == null$** , entonces x es hoja

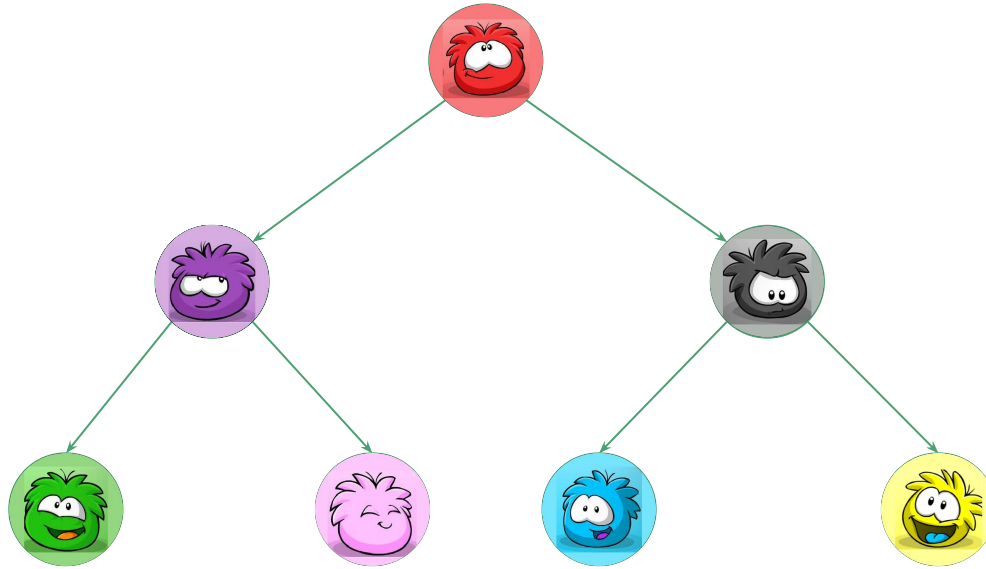
¿Como se ve un nodo en C?

```
struct node {  
    int key;  
    struct node *left, *right;  
};
```



Árbol binario de búsqueda (ABB)

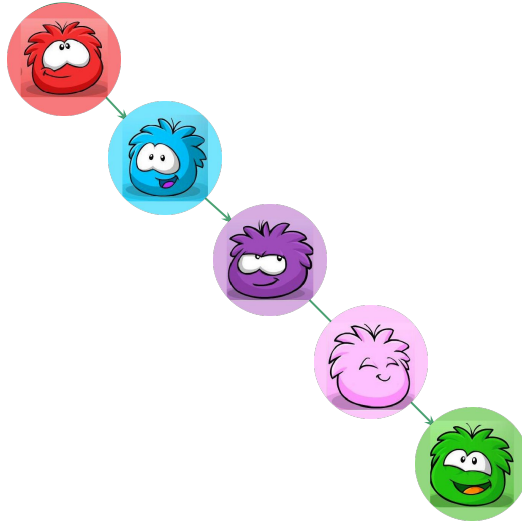
- Altura mínima de la rama más larga es $O(\log(n))$ -> MEJOR CASO



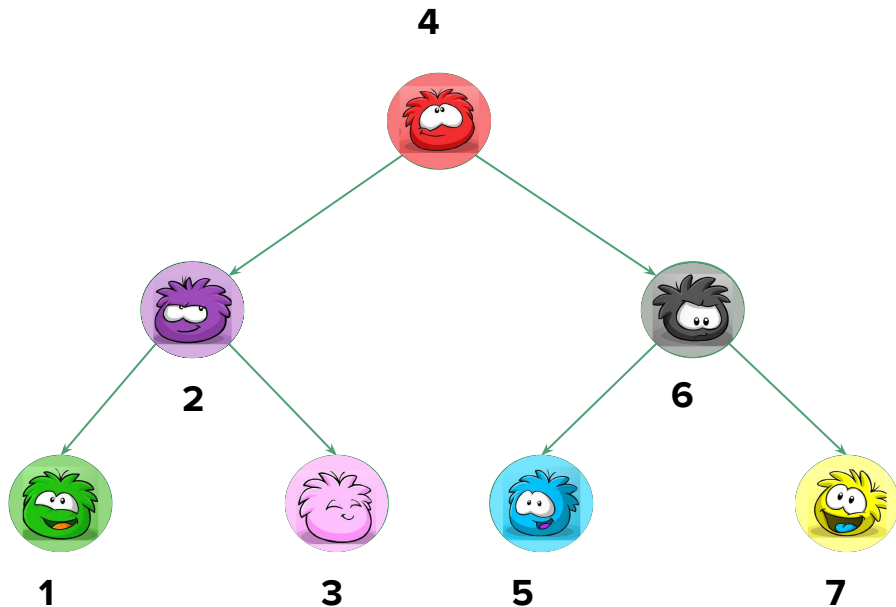
- Cuando está perfectamente balanceado.
- Dado que la diferencia de altura entre el subárbol izquierdo y el subárbol derecho de cada nodo es como máximo 1. con n el número de nodos.

Árbol binario de búsqueda (ABB)

- Altura máxima de $O(n)$, cuando no está balanceado



¿Cómo recorrer árboles?

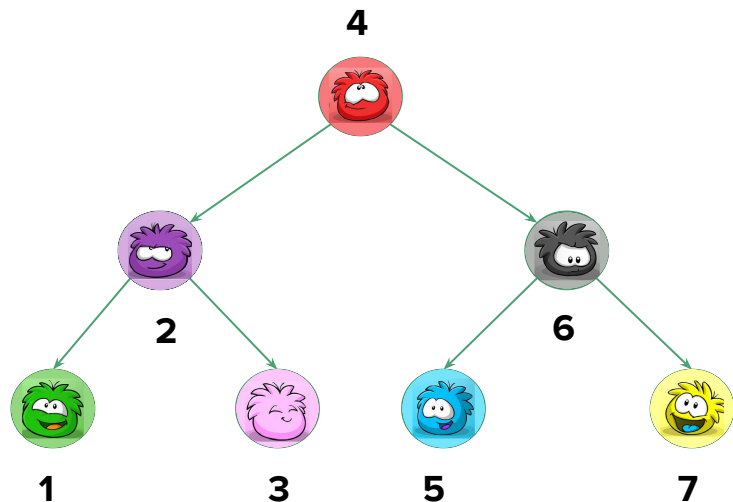


```
void printfunction (Node* node)
{
    if (node == NULL)
        return;
    printfunction(node->left);
    printf('%d', node->value);
    printfunction(node->right);
}
```

```
void printfunction (Node* node)
{
    if (node == NULL)
        return;
    printf('%d', node->value);
    printfunction(node->left);
    printfunction(node->right);
}
```

```
void printfunction (Node* node)
{
    if (node == NULL)
        return;
    printfunction(node->left);
    printfunction(node->right);
    printf('%d', node->value);
}
```


Cómo recorrer árboles?



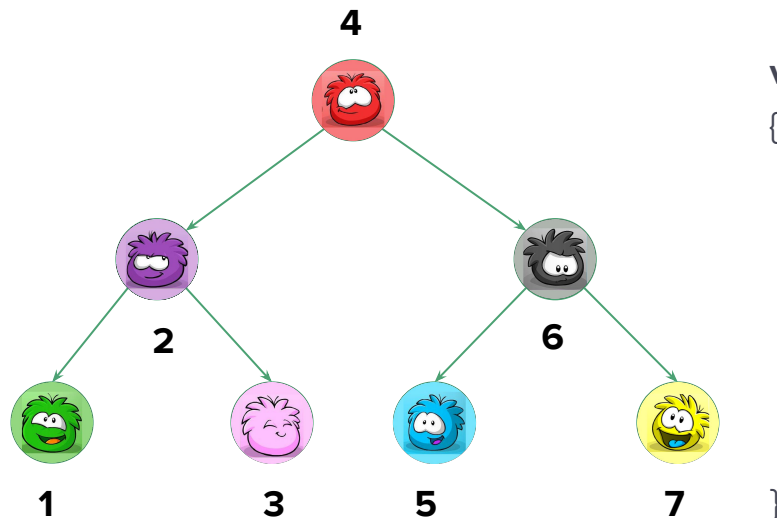
InOrder

```
void printfuncton (Node* node)
{
    if (node == NULL)
        return;
    printfuncton(node->left);
    printf('%d', node->value);
    printfuncton(node->right);
}
```

1234567

Cómo recorrer árboles?

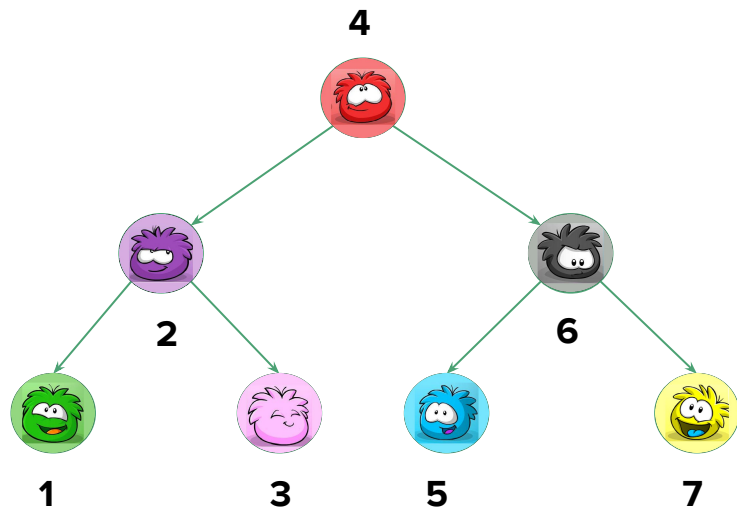
PreOrder



```
void printfunction (Node* node)
{
    if (node == NULL)
        return;
    printf('%d', node->value);
    printfunction(node->left);
    printfunction(node->right);
}
```

4213657

Cómo recorrer árboles?



PostOrder

```
void printfuncton (Node* node)
{
    if (node == NULL)
        return;
    printfuncton(node->left);
    printfuncton(node->right);
    printf('%d', node->value);
}
```

1325764

Qué es balance

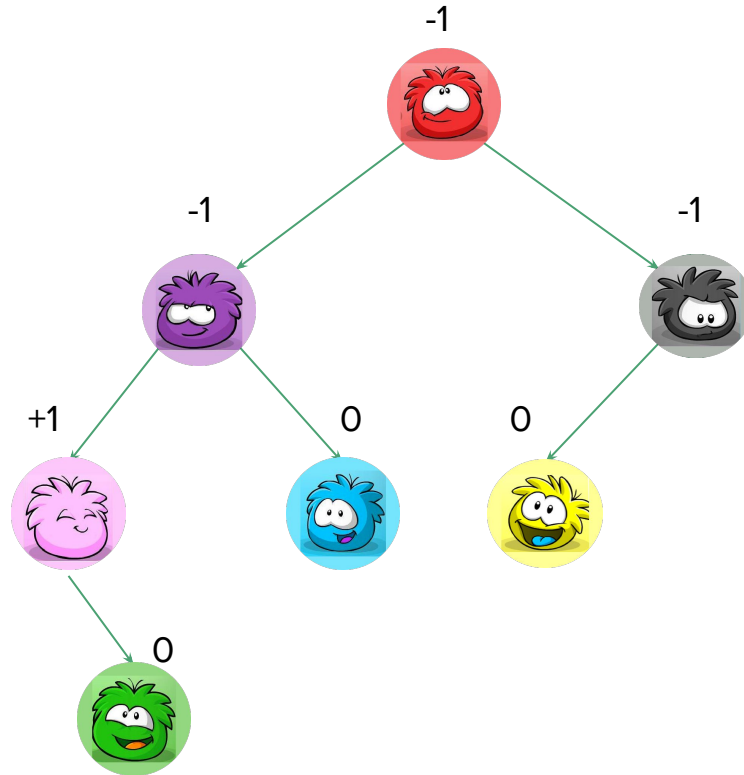
- Sean n nodos, que la altura sea $O(\log(n))$
- Sea fácil de mantener
- La definición debe ser de carácter recursivo
- y para que sea fácil de mantener se realizan luego de cada operaciones, para que el proceso de balanceo sea $<O(\log(n))$



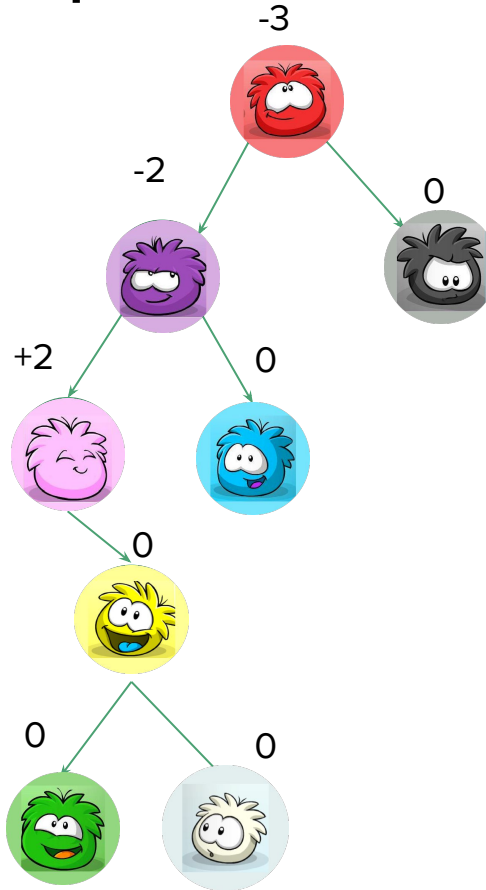
AVL

- Es un tipo de ABB
- Se encuentra balanceado, entonces altura $O(\log(n))$
- La diferencia máxima de sus hijos es de 1
- Cada hijo es AVL (definición recursiva)

¿Que significa que difiera a lo más de 1?

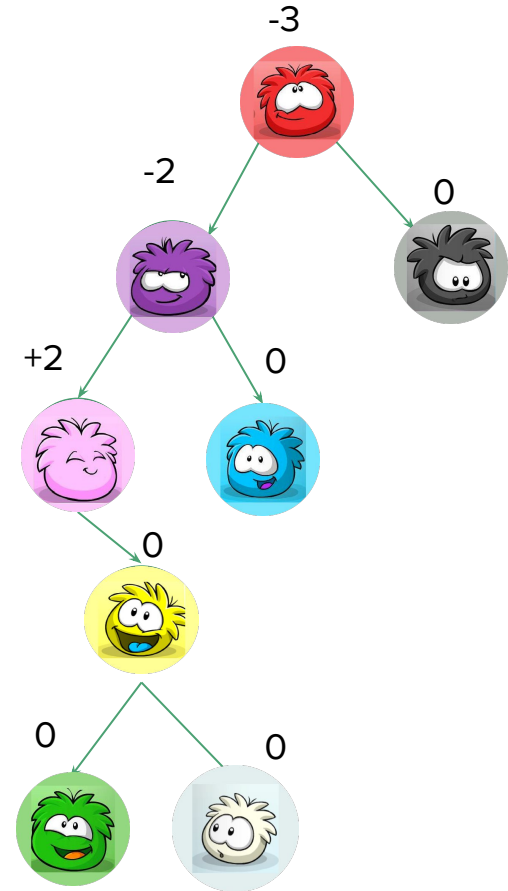


¿Que significa que difiera a lo más de 1?

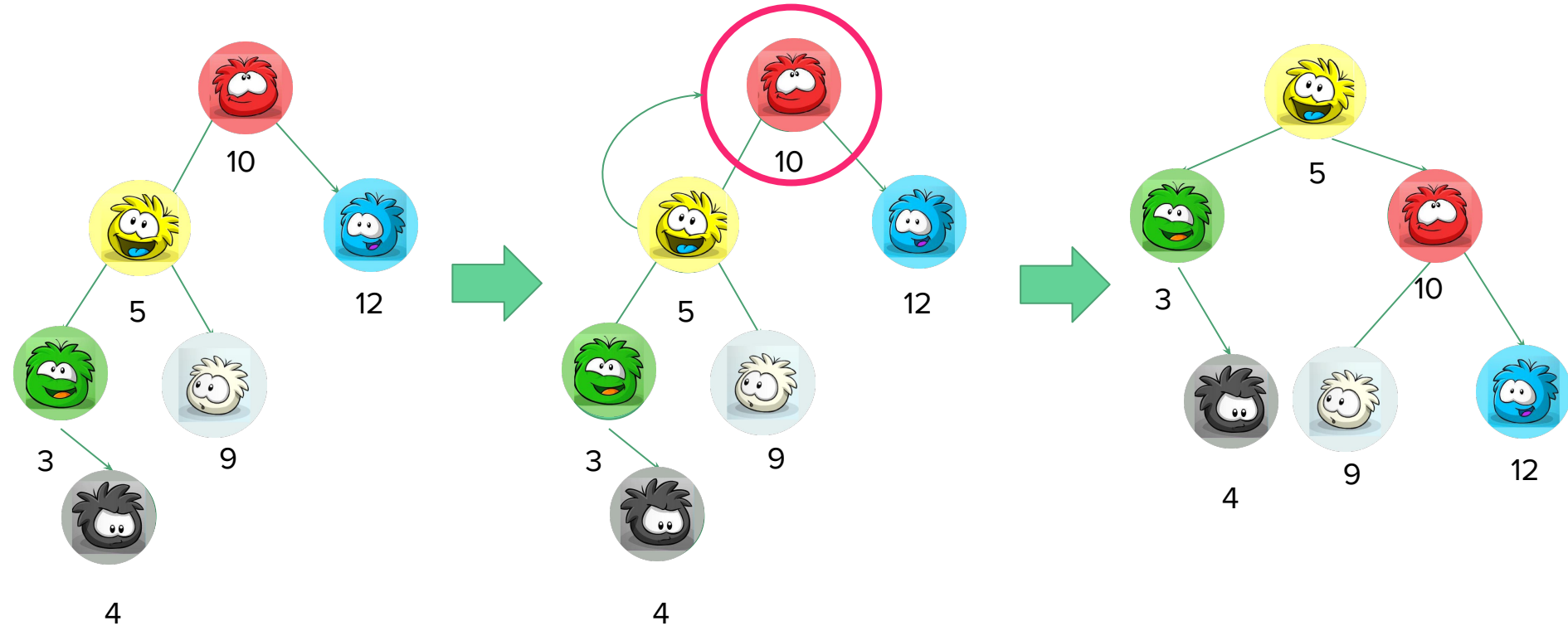


¿Cómo lo arreglamos?

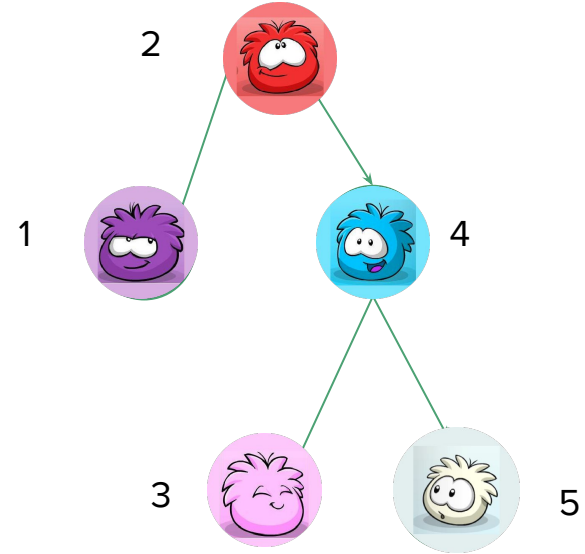
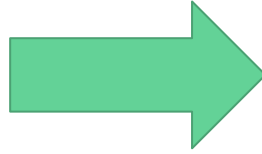
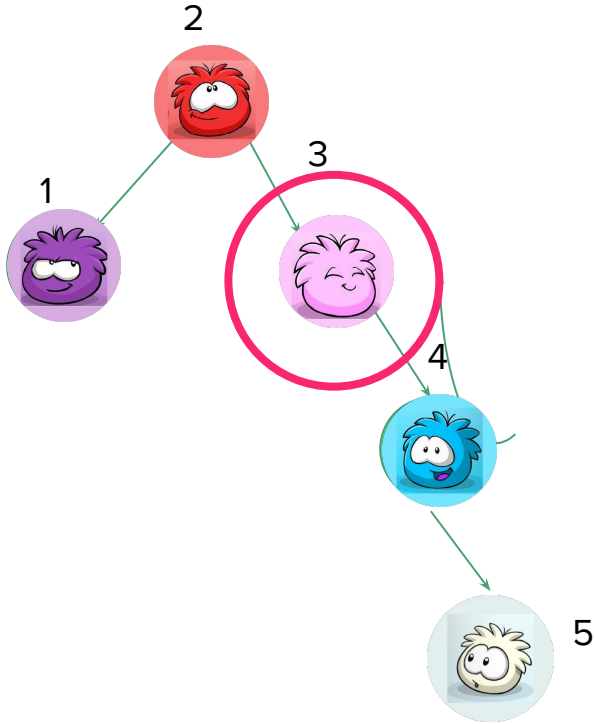
1. Identificar primer nodo desbalanceado de forma ascendente
2. Identificar rotación
3. Rotar
4. Evaluar balance



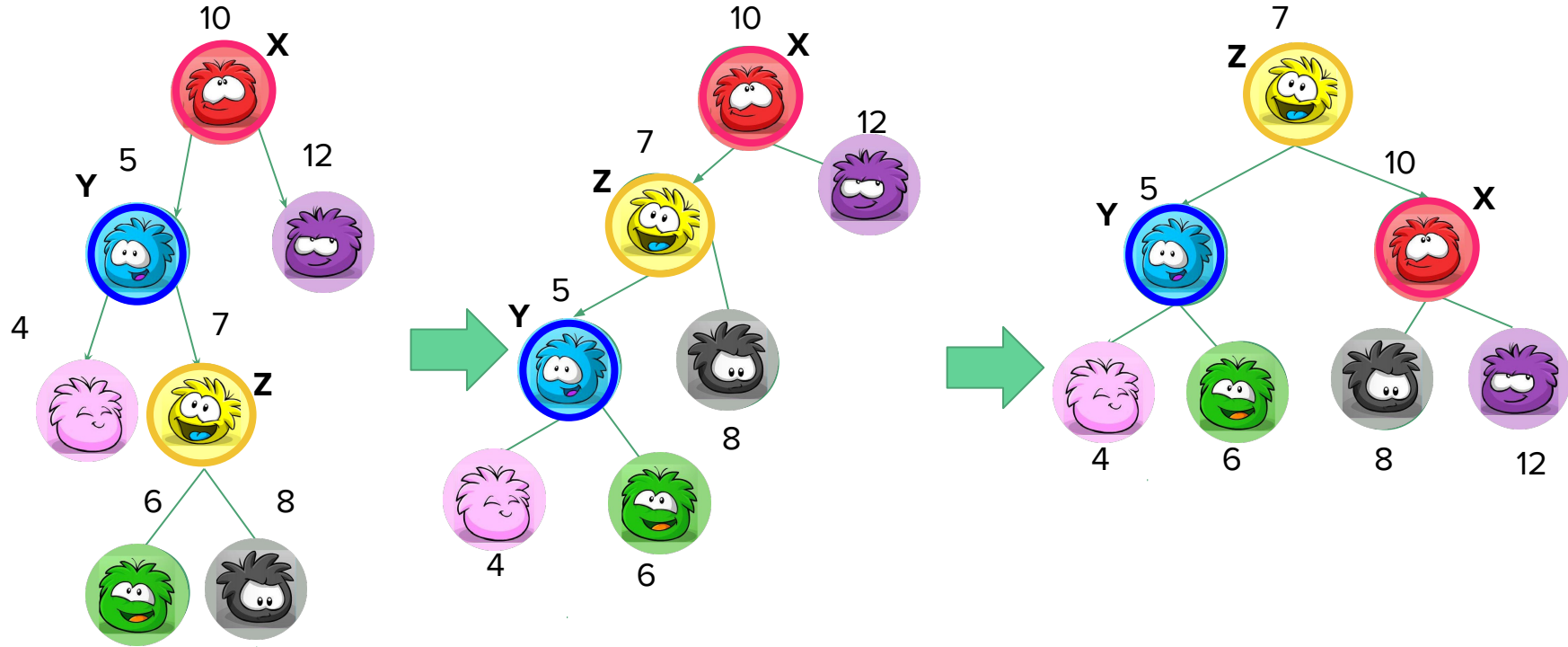
Rotación Simple



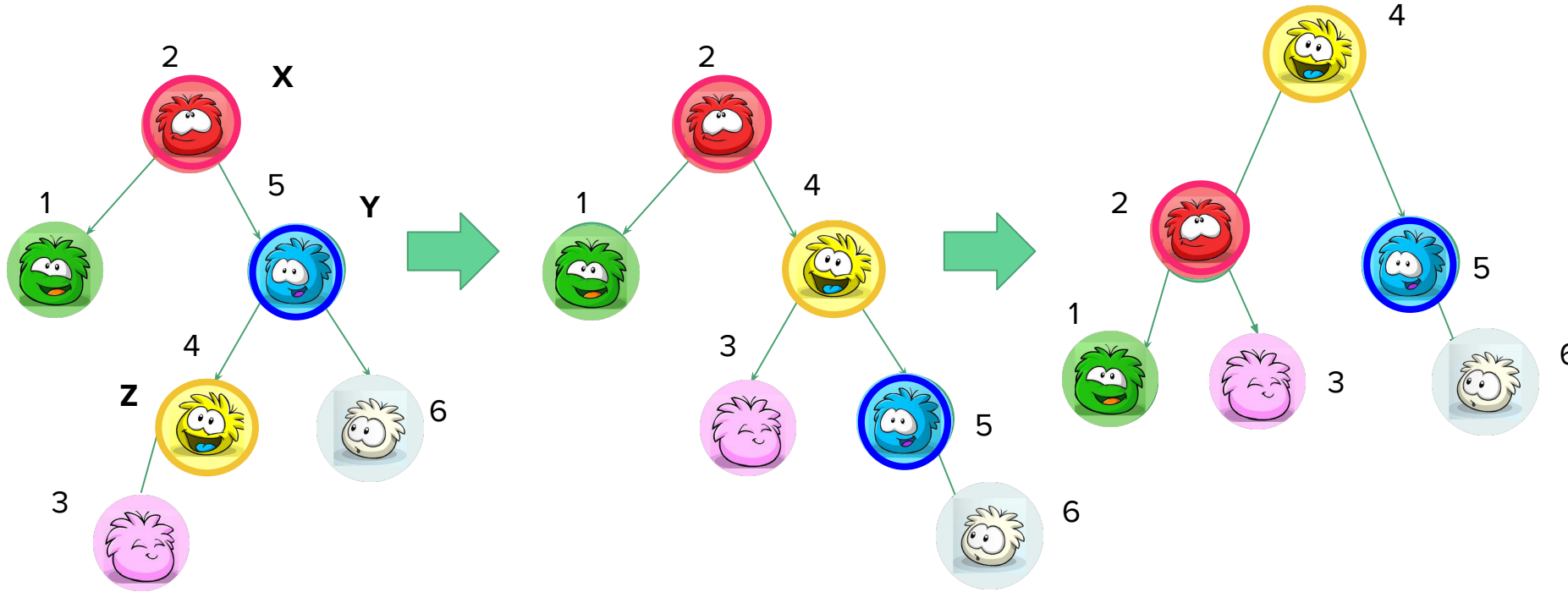
Rotación Simple



Rotación: Doble



Rotación Doble



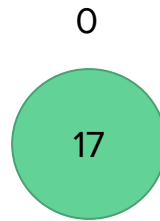
Ejercicio rotación AVL

Considera que tienes un AVL vacío y una lista desordenada:

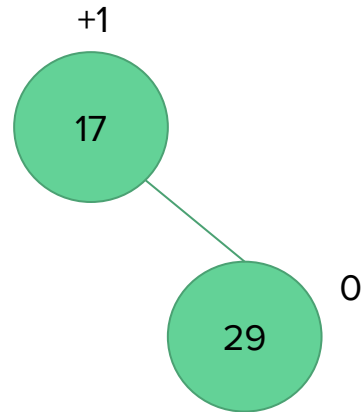
17, 29, 53, 61, 73, 37

Inserta los nodos realizando las rotaciones diferentes

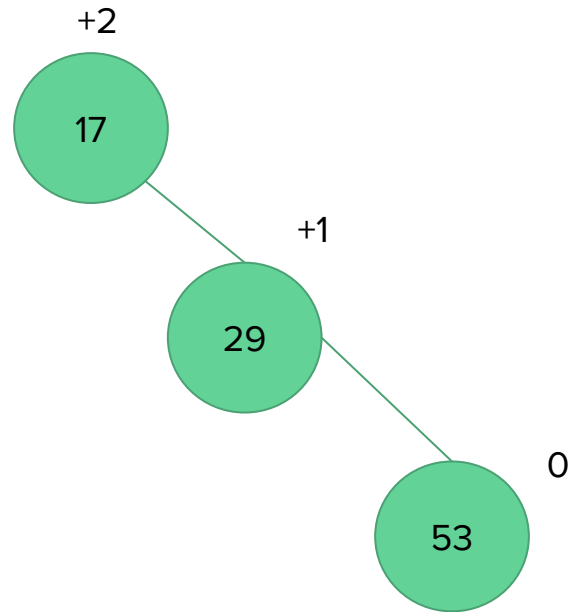
Insertar 17



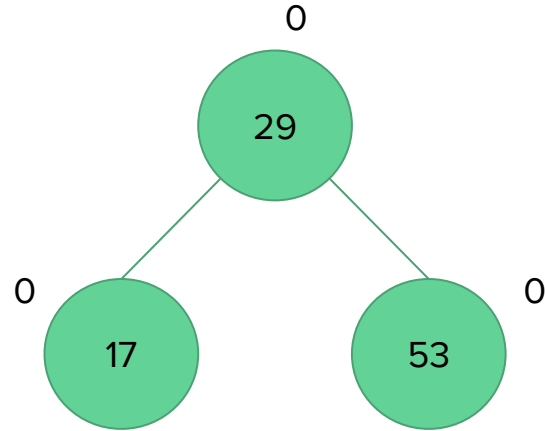
Insertar 29



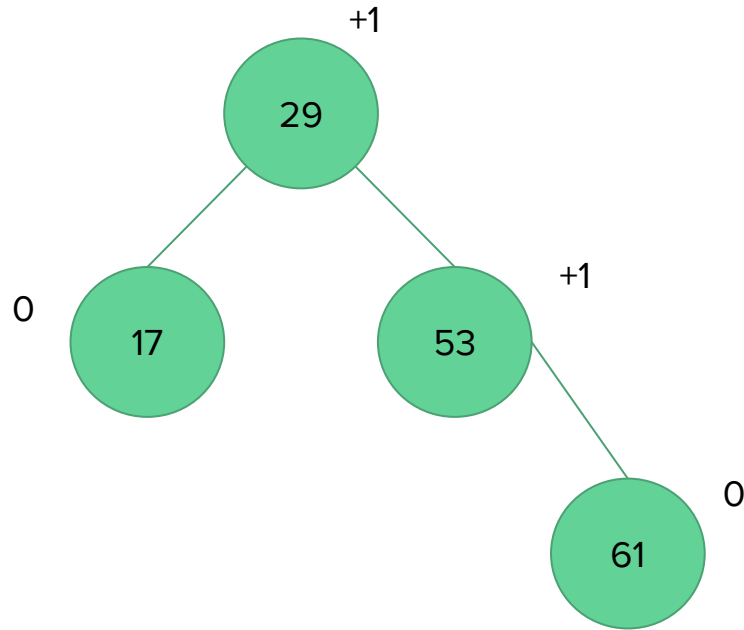
Insertar 53



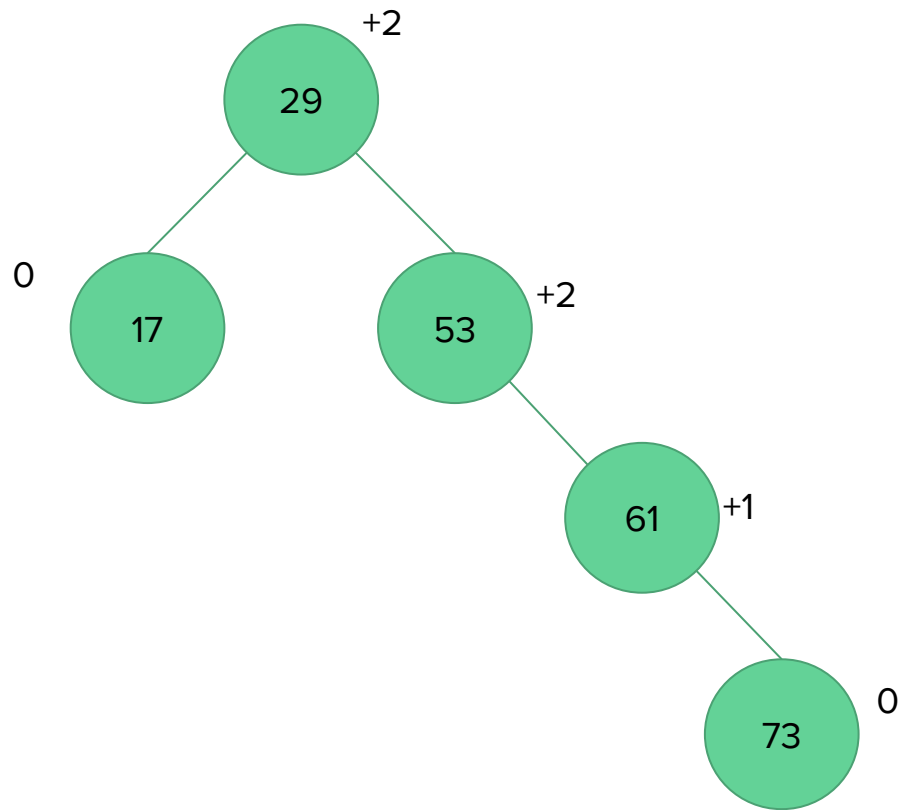
Rotación Simple



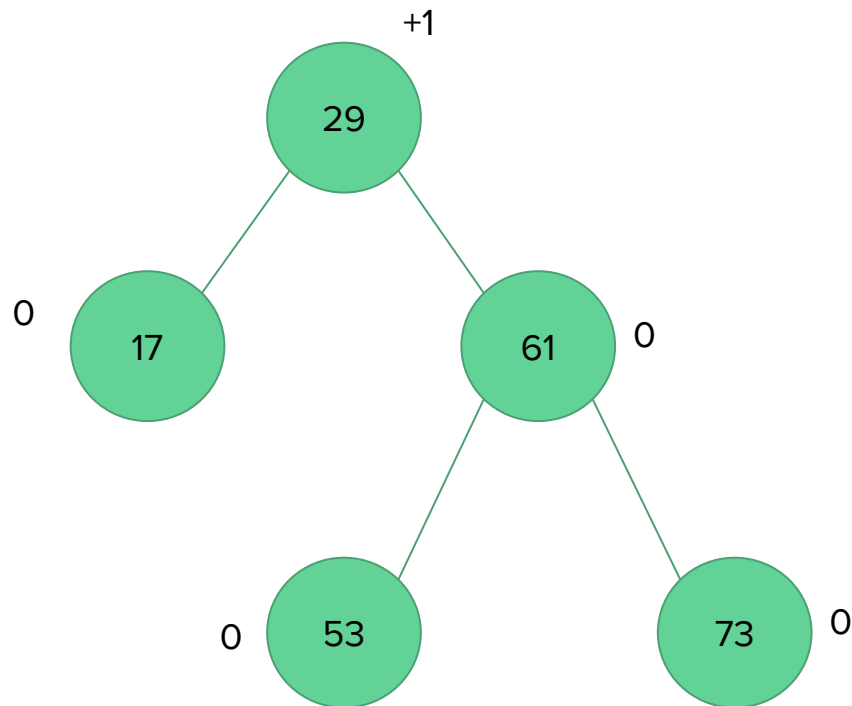
Insertar 61



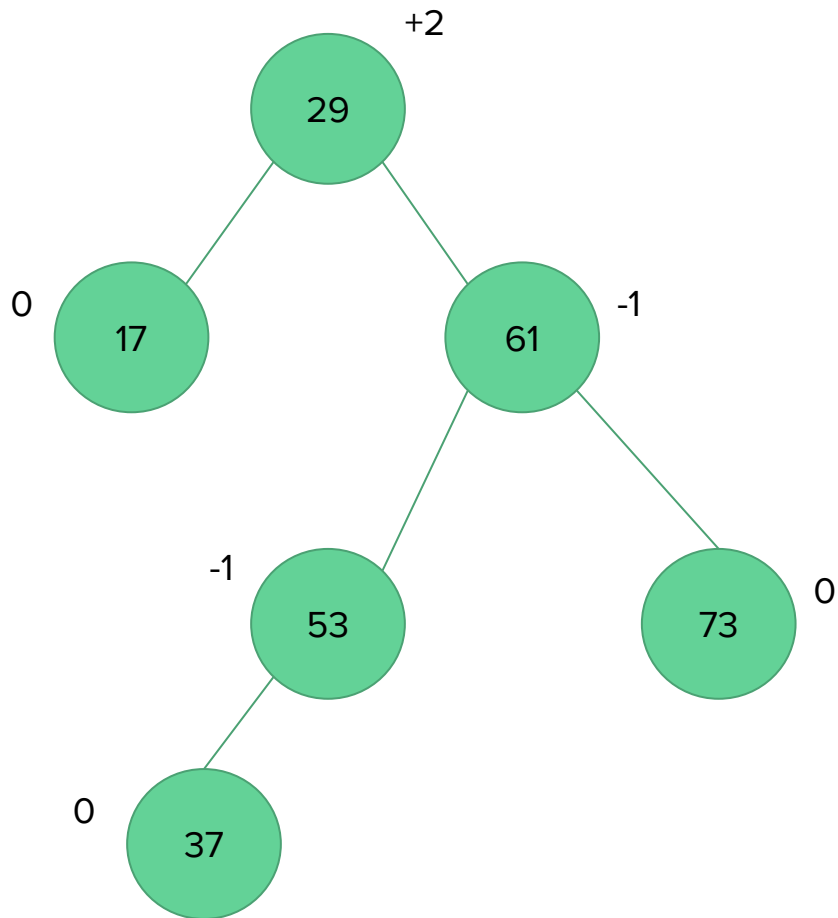
Insertar 73



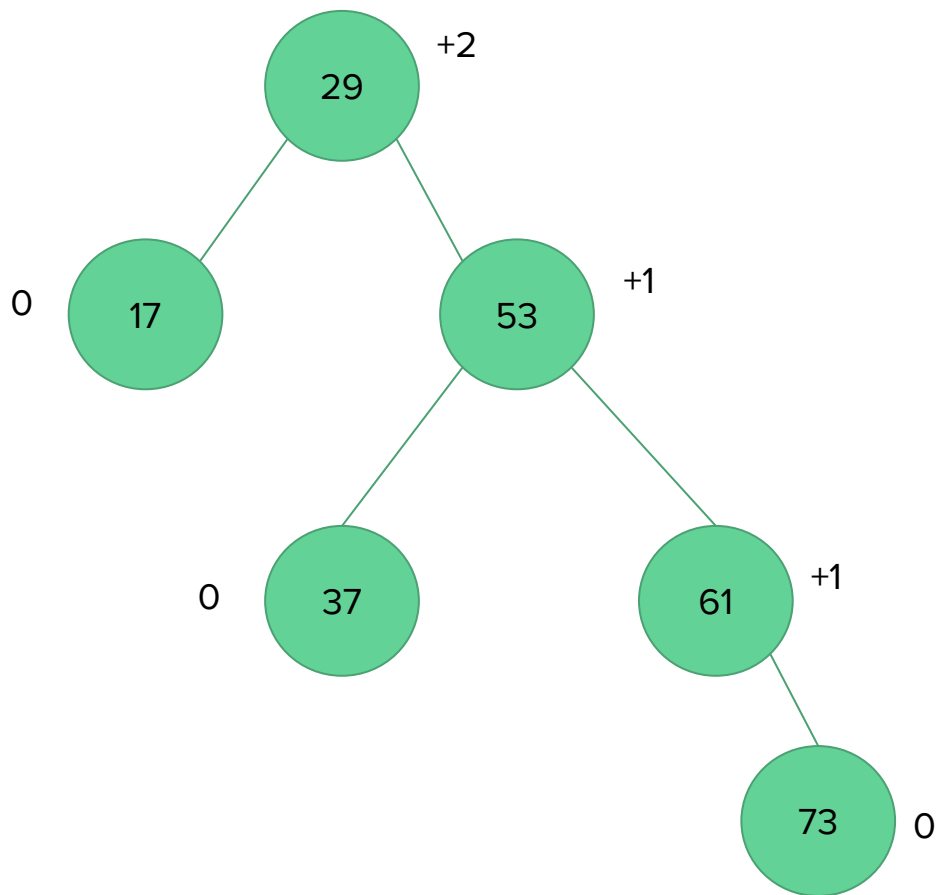
Rotación Simple



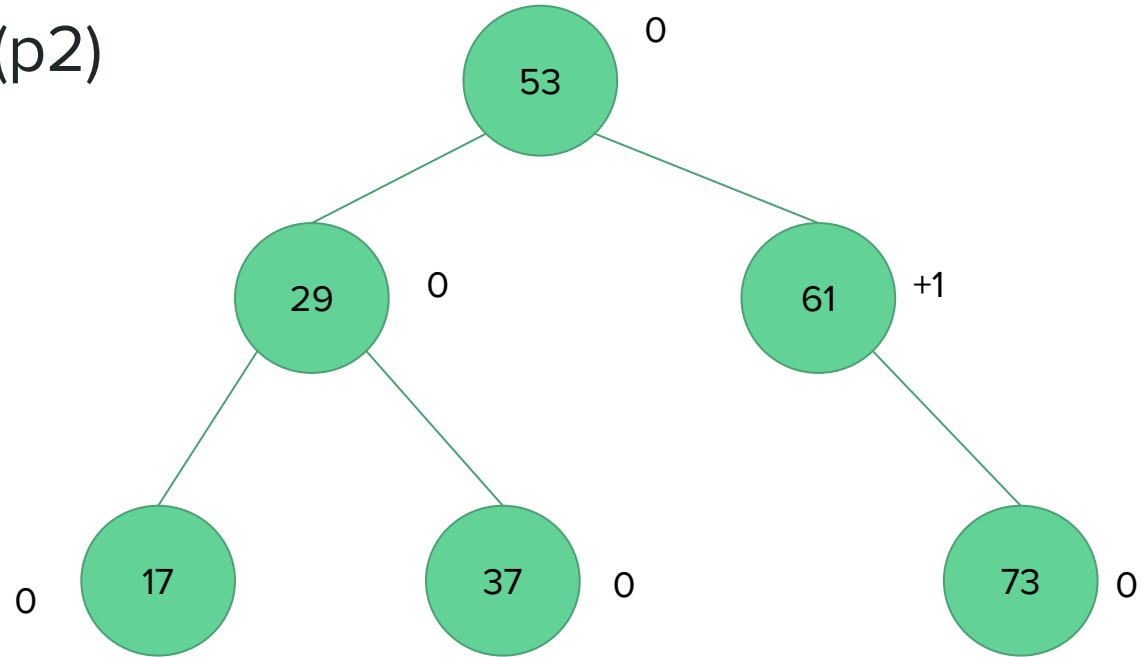
Insertar 37



Rotación doble (p1)



Rotación doble (p2)



¡Muchas gracias!

