

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;

    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;

    if(argc != 2)
    {
        printf(stderr, "ERRORE: serve un parametro con il nome del file\n");
        exit(1);
    }
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        printf(stderr, "ERRORE: impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    while( fgets( riga, MAXRIGA, f ) != NULL )
```

## Trees and BSTs

### **BSTs: Extension 01**

Paolo Camurati and Stefano Quer

Dipartimento di Automatica e Informatica

Politecnico di Torino

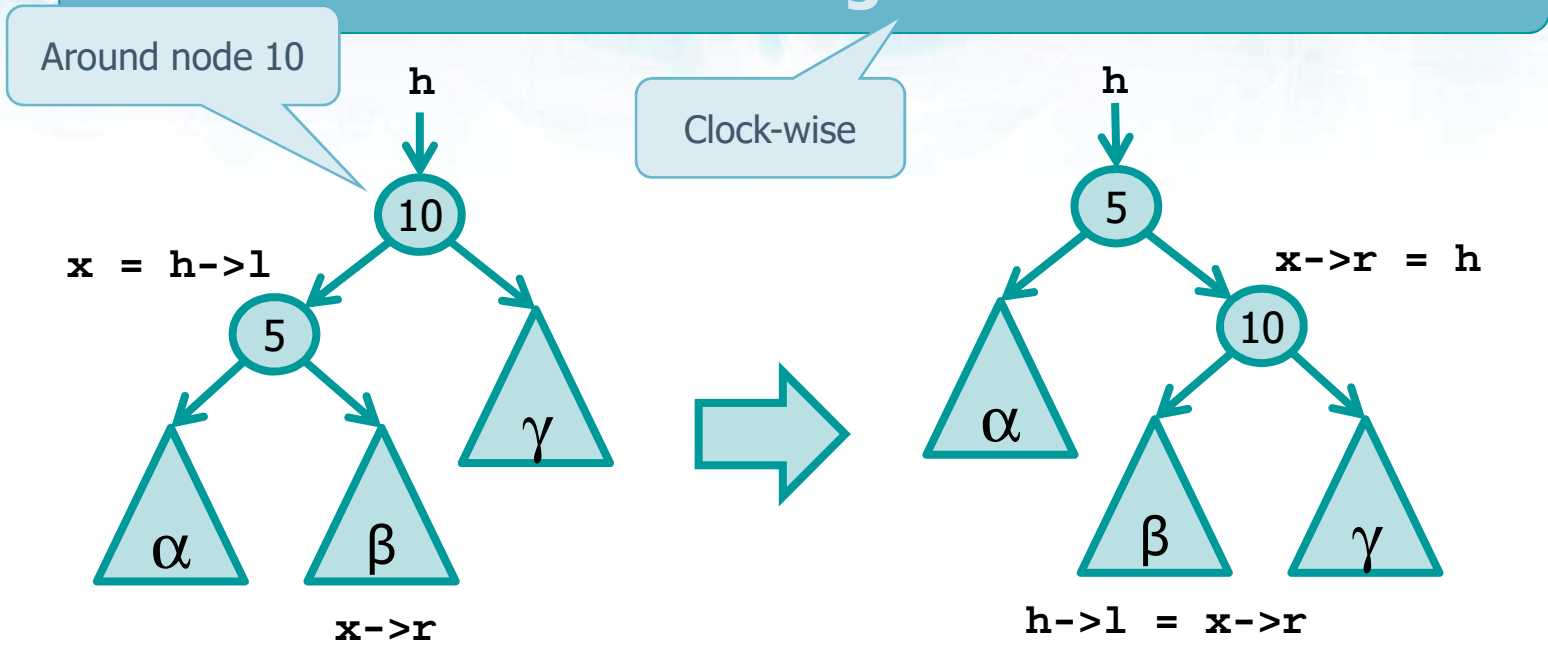
## Root insertion

- ❖ Given a BST, nodes can be inserted
  - On the leaves, using the strategy previously analyzed
  - On the root, using a "root insertion" procedure
- ❖ Motivation
  - By inserting new nodes on the BST root, more recent key remains closer to the root
  - For the **locality** principle more recent keys have a higher probability to be addressed again soon
  - Thus, it is faster to search more recent keys speeding-up the BST manipulation

## Root insertion

- ❖ Root insertions
  - Must maintain the BST property
  - Make use of local adjustments easy to implement and efficient to execute
- ❖ The core idea is to
  - Insert a new node onto a leaf
    - This can be done using the original standard insertion procedure
  - Move the node onto the tree root
    - This step maybe performed using **rotations**
    - There are two form of rotations: **Right** and **Left**

# Right Rotation of a BST



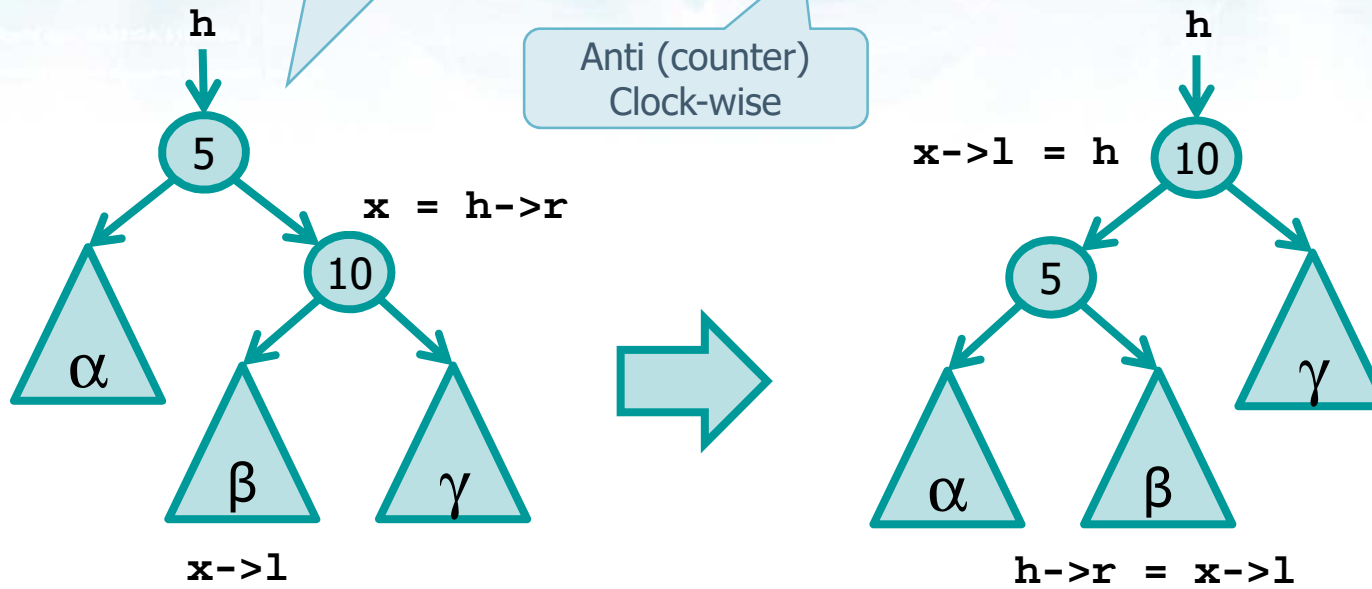
```

link rotR (link h) {
    link x = h->l;
    h->l = x->r;
    x->r = h;
    return x;
}
    
```

# Left Rotation of a BST

Around node 5

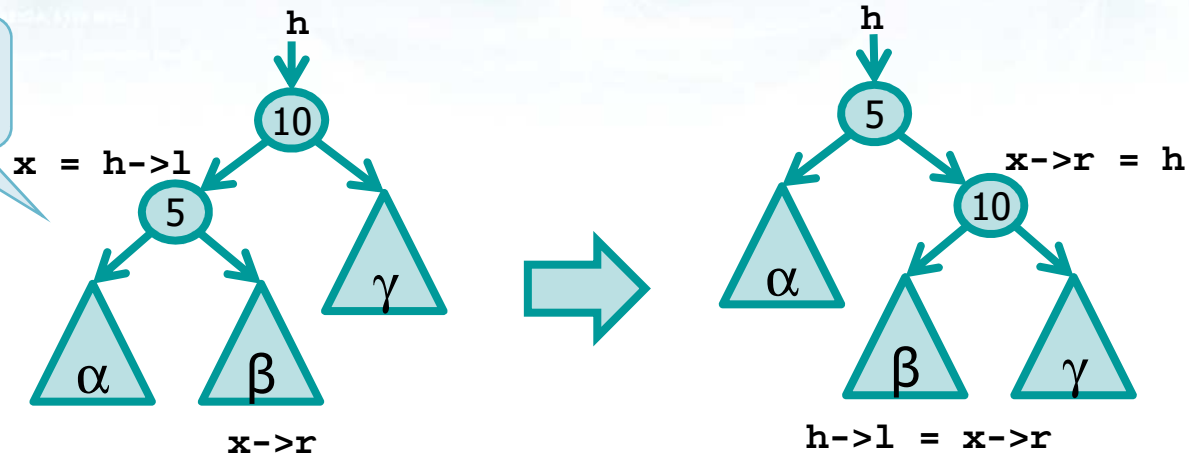
Anti (counter) Clock-wise



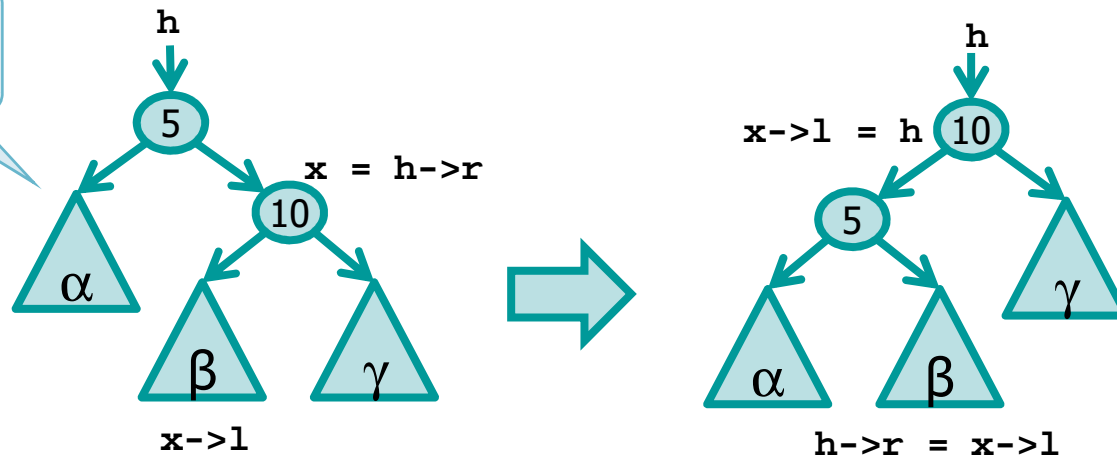
```
link rotL(link h) {
    link x = h->r;
    h->r = x->l;
    x->l = h;
    return x;
}
```

# Right vs Left Rotation of a BST

Right Rotation



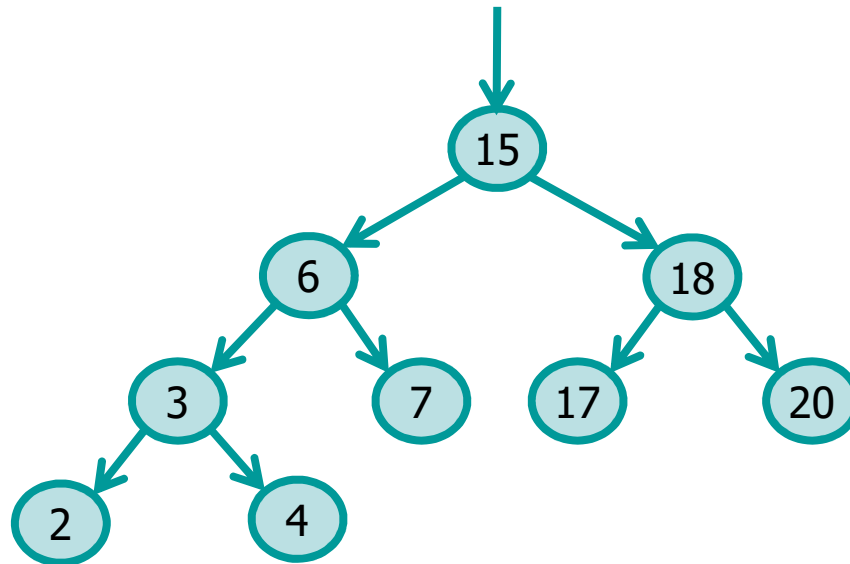
Left Rotation



# Examples

- ❖ Given the following tree perform
  - A right rotation around node 6
  - A left rotation around node 18

Around node ... its  
the node **on top**



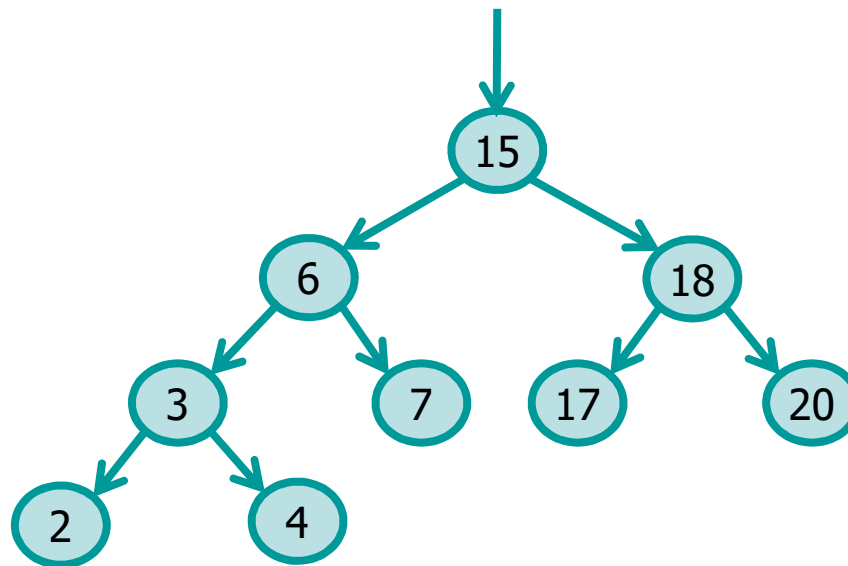
## Root insertion

- ❖ To insert a new node on the root of a BST it is possible to use the following recursive procedure
  - Insert the new node into the appropriate sub-tree following the leaf insertion procedure
  - Rotate the node moving it upward, to force it onto the tree root



## Examples

- ❖ Given the following tree perform the insertion **on the root** of the following keys
  - key = 16
  - Key = 5



## Implementation

```
link insert_root_r (link root, Item x, link z) {  
    if (root == z)  
        return NEW (x, z, z);  
  
    if (item_less(x, root->item)) {  
        root->l = insert_root_r (root->l, x, z);  
        root = rotR (root);  
    } else {  
        root->r = insert_root_r (root->r, x, z);  
        root = rotL (root);  
    }  
  
    return root;  
}
```

Recur left  
Rotate right

Recur right  
Rotate left

## Exercise

- ❖ Given an initially empty BST insert the following keys on the root
  - 20 12 9 30 36 31 33

## Exercise

- ❖ Given an initially empty BST perform the following insertions (+) and extractions (−)
  - +10 +11 +21 +9 +18 +37 +25 + 7 +5 +39  
+30 +6 +32 +17 +15 +13 +2 +24 −21 −18  
−11 −10
  - Insert on leaves before (exercise 1)
  - Insert on root after (exercise 2)

## Exercise

- ❖ Given an initially empty BST perform the following insertions (+) and extractions (−)
  - +A +S +E +R +C +H +I +N +G +X +M  
+P +L −A −R −I −G
  - Insert on leaves before (exercise 1)
  - Insert on root after (exercise 2)