Amirkabir University of Technology
(Tehran Polytechnic)

Department of
Computer Engineering

Data Structure & Algorithms

Augmenting Data Structures

Augmentation Process

Augmentation is a process of extending a data structure in order to support additional functionality. It consists of four steps:

1. Choose an underlying data structure.

2. Determine the additional information to be maintained in the underlying data structure.

3. Verify that the additional information can be maintained for the basic modifying operations on the underlying data structure.

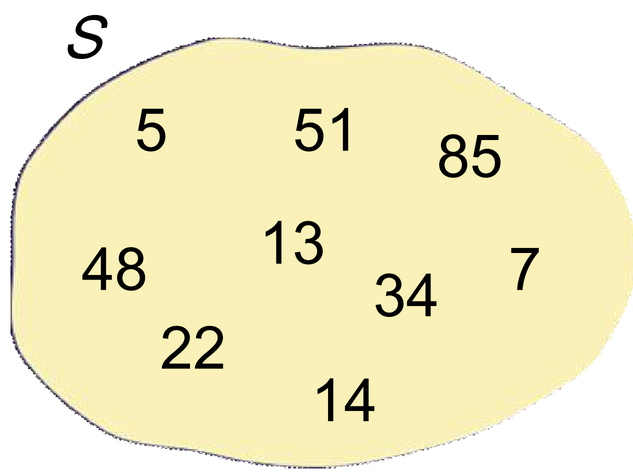
4. Develop new operations.

Examples for Augmenting DS

- Dynamic order statistics: Augmenting binary search trees by size information
- Interval trees
- Priority search trees

Dynamic Order Statistics

Problem: *Given a set S of numbers that changes under insertions and deletions, construct a data structure to store S that can be updated in O*(log *n*) *time and that can report the k-th order statistic for any k in O*(log *n*) *time.*

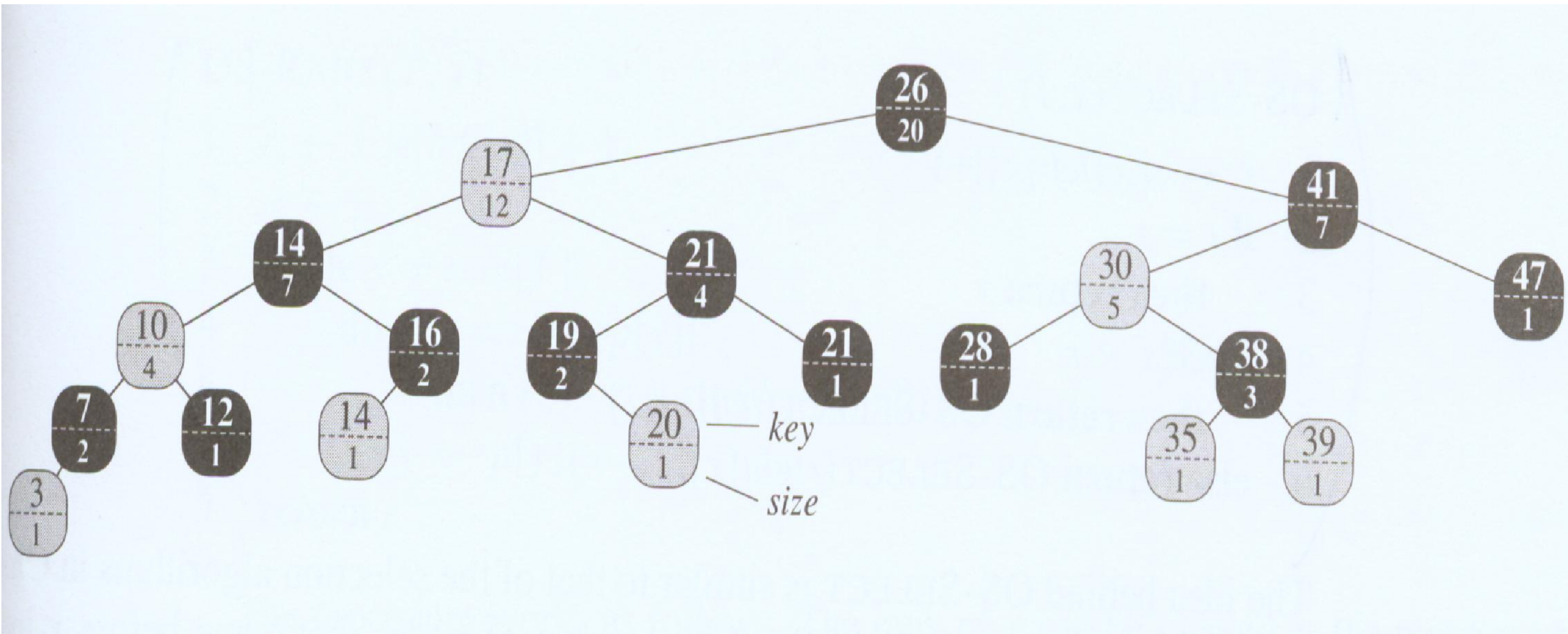


Order Statistics Tree

Beside the usual red-black tree fields $key[x]$, $color[x]$, $p[x]$, $left[x]$, and $right[x]$ in a node x, we have another field $size[x]$. This field contains the number of (internal) nodes in the subtree rooted at x (including x itself), that is the size of the subtree. If we define the sentinel's size to be 0, that is, we set $size[nil[T]]$ to be 0, then we have the identity:

$$size[x] = size[left[x]] + size[right[x]] + 1$$

An order-statistic tree



Retrieving an element with a given rank

Note that: rank(x) = size[left[x]] + 1

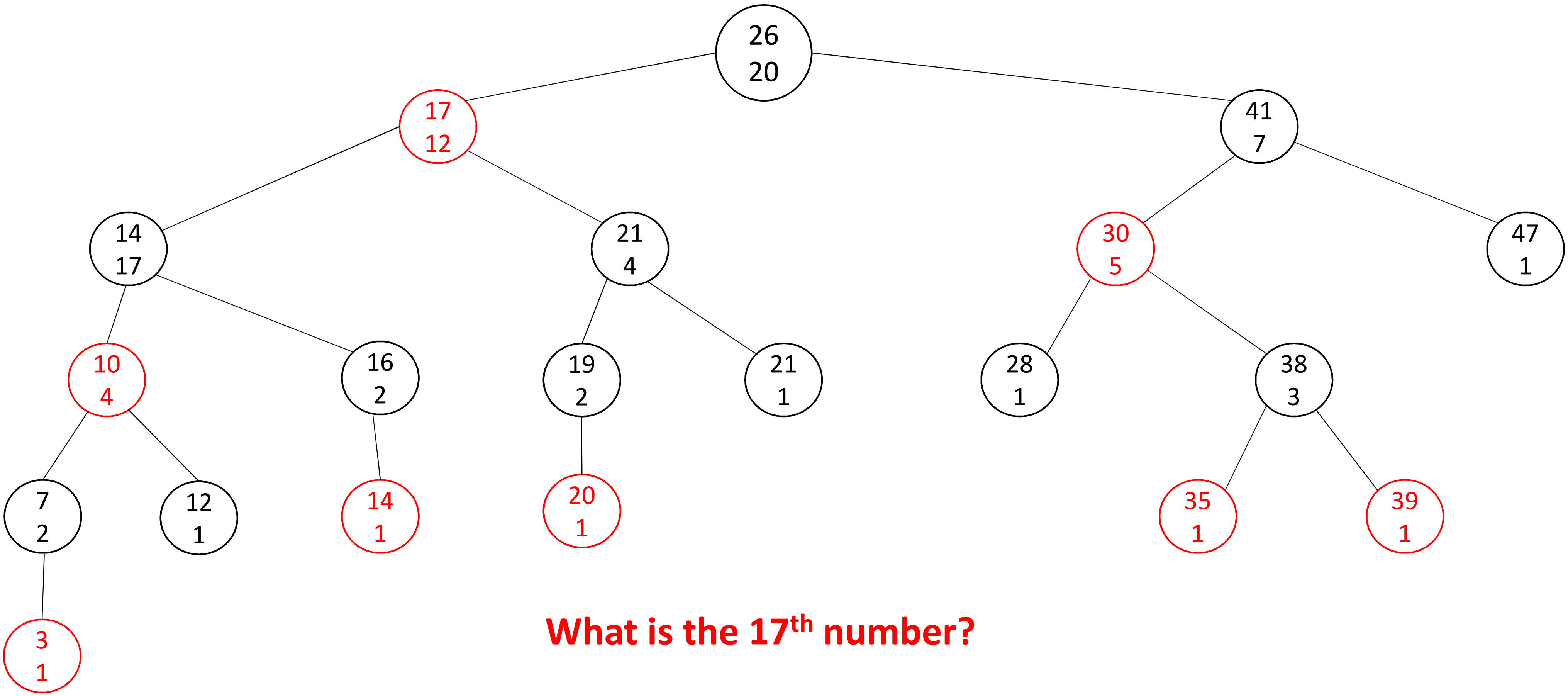
OS-SELECT(x, i)

```
r ← size[left[x]] + 1
    if i == r
        then return x
    else if i < r
        then return OS-SELECT(left[x], i)
    else return OS-SELECT(right[x], i – r)
```

Time complexity :O(lg n)

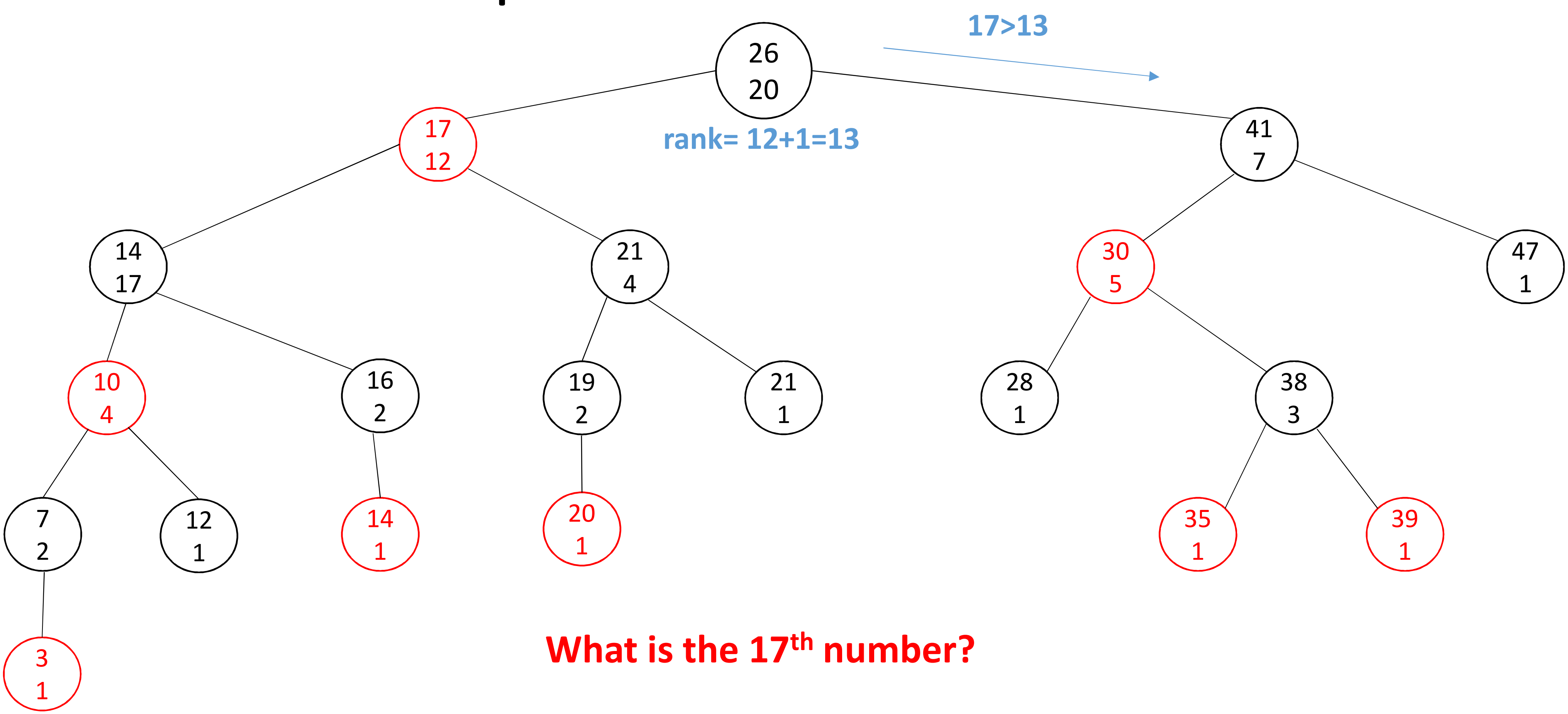
Because each level of tree has a constant cost, so overall cost will be equal to heightOfTree * ConstantTime → O(lgn).

Selection Example



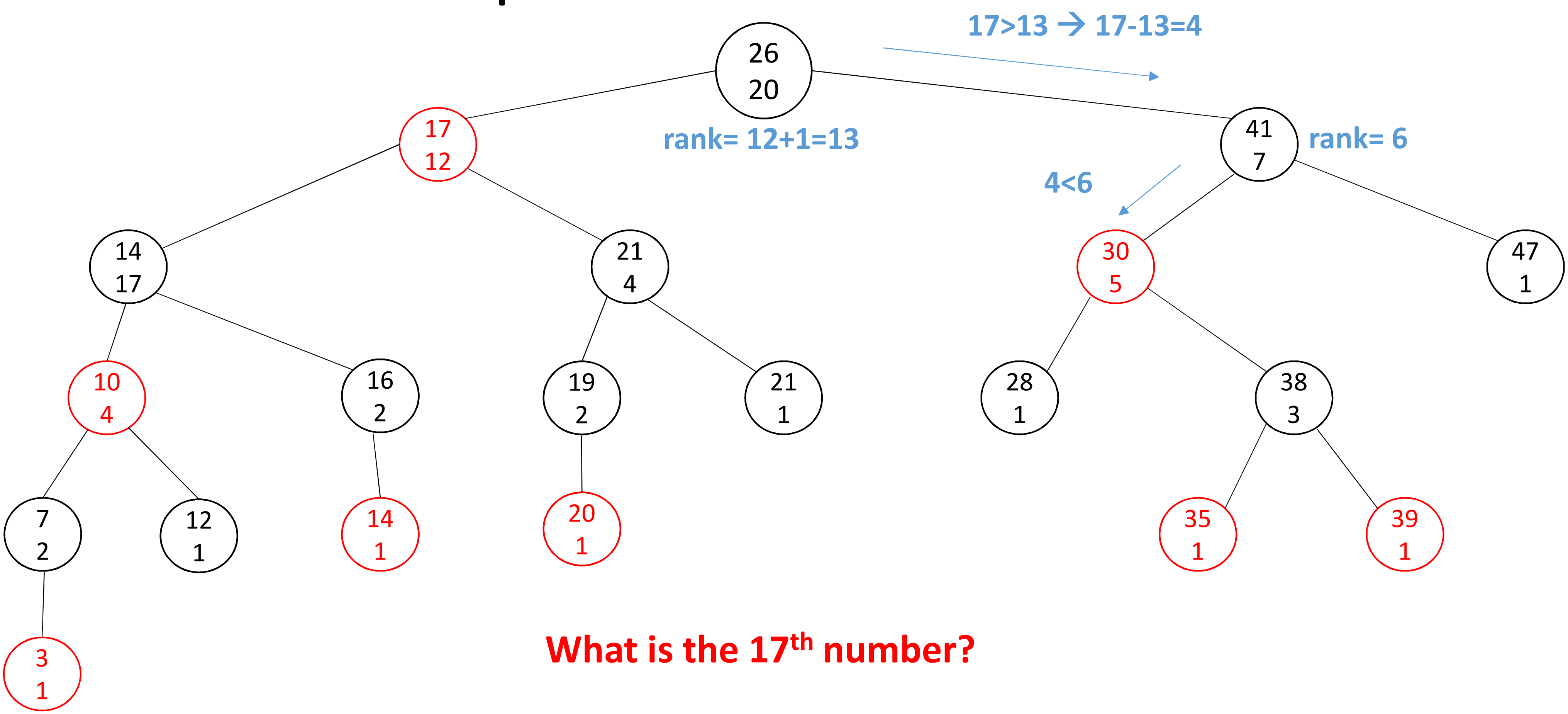
What is the 17th number?

Selection Example



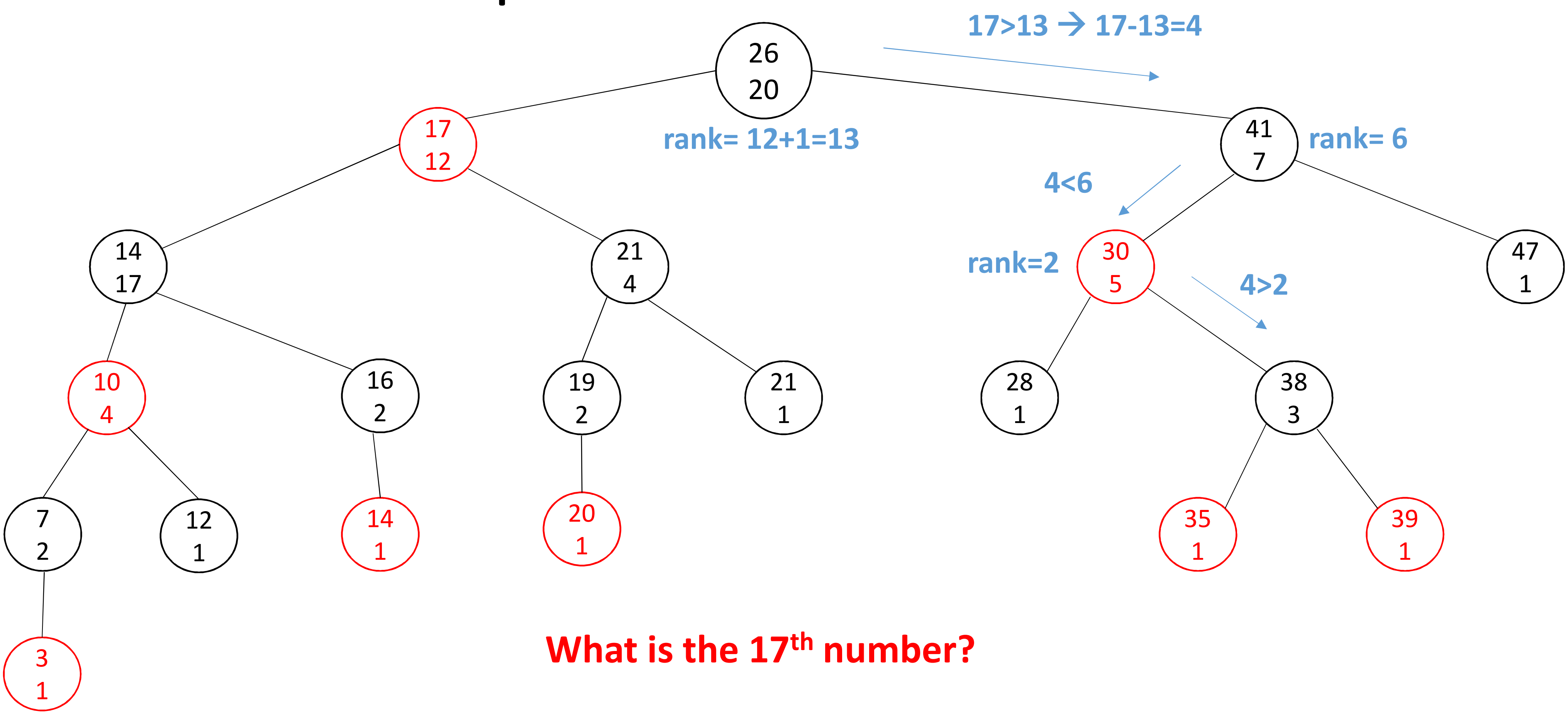
What is the 17^{th} number?

Selection Example



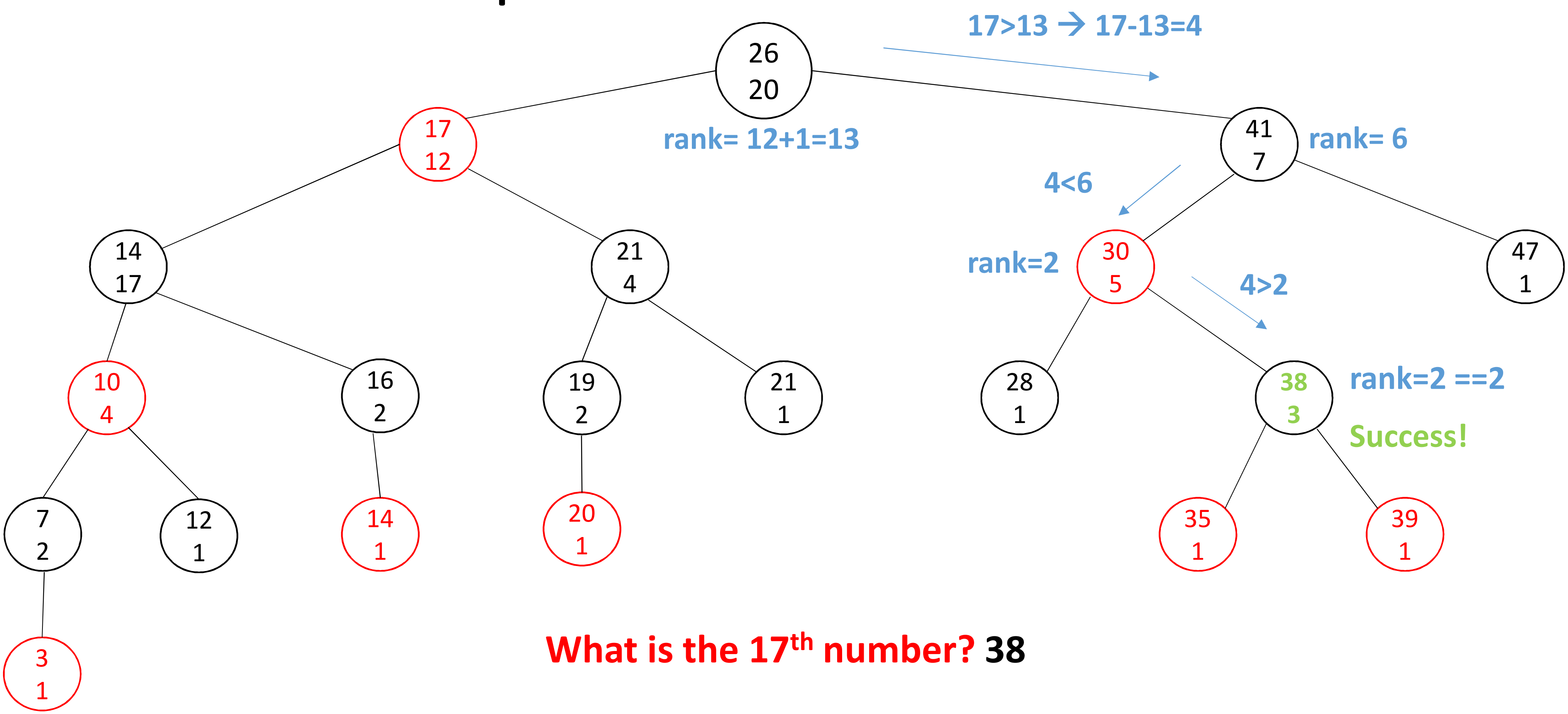
What is the 17th number?

Selection Example



What is the 17th number?

Selection Example



What is the 17th number? 38

Determining the rank of an element

OS-RANK(T, x)

```
r ← size[left[x]] + 1
y ← x
while y ≠ root[T]
    do if y == right[p[y]]
            then r ← r + size[left[p[y]]] + 1
        y ← p[y]
return r
```

The running time of OS-RANK is at worst proportional to the height of the tree: O(lg n)

Maintaining subtree sizes

To use OS_Select & OS_Rank, the size property should be correct. So while insertion and deletion we should update it.

Maintaining subtree sizes in insertion

Each insertion has 2 phases as we mentioned in previous lectures:

1. Adding the new element:

In this phase, we should increment the sizes in added element to root by 1 which takes O(lg n) time.

1. Fixup:

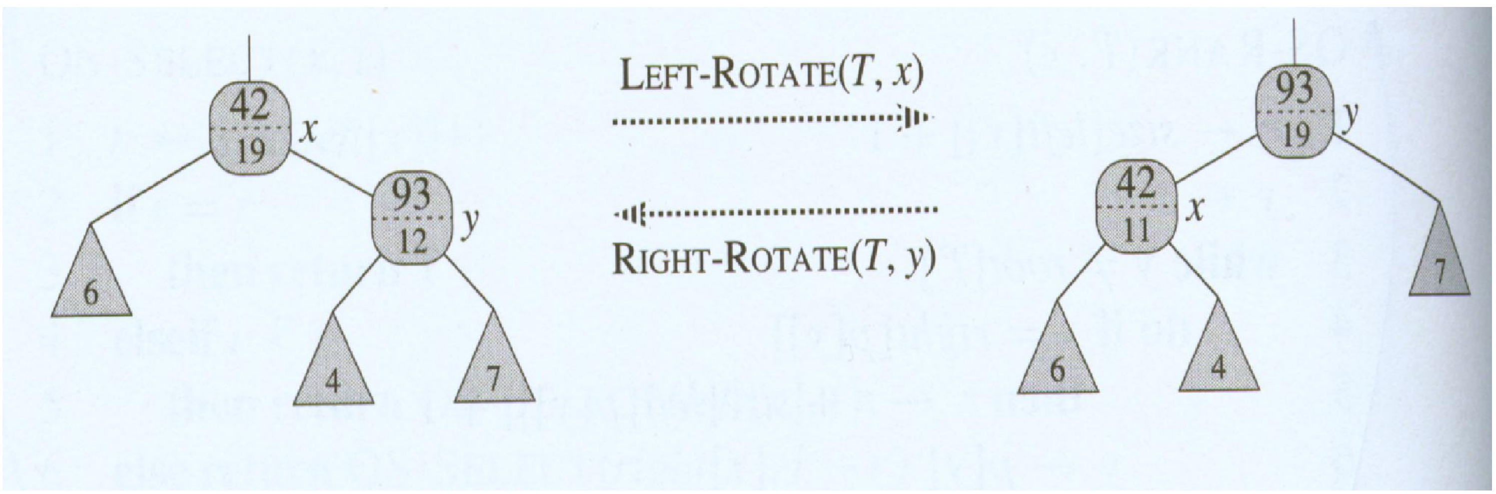
Rotation is the only operation which affects size property, so we change it as explained in the next slide.

Updating subtree sizes during rotations

Referring to the code for LEFT-ROTATE(T, x) we add the following lines:

12 $size[y] \leftarrow size[x]$

13 $size[x] \leftarrow size[left[x]] + size[right[x]] + 1$



Maintaining subtree sizes in deletion

Each deletion has 2 phases as we mentioned in previous lectures:

1. Deleting the desired element:

In this phase, we should decrement the sizes from y to root by 1which takes

O(lg n) time.

1. Fixup:

Rotation is the only operation which affects size property, so we change it as explained in the previous slide which also takes O(lg n) time.