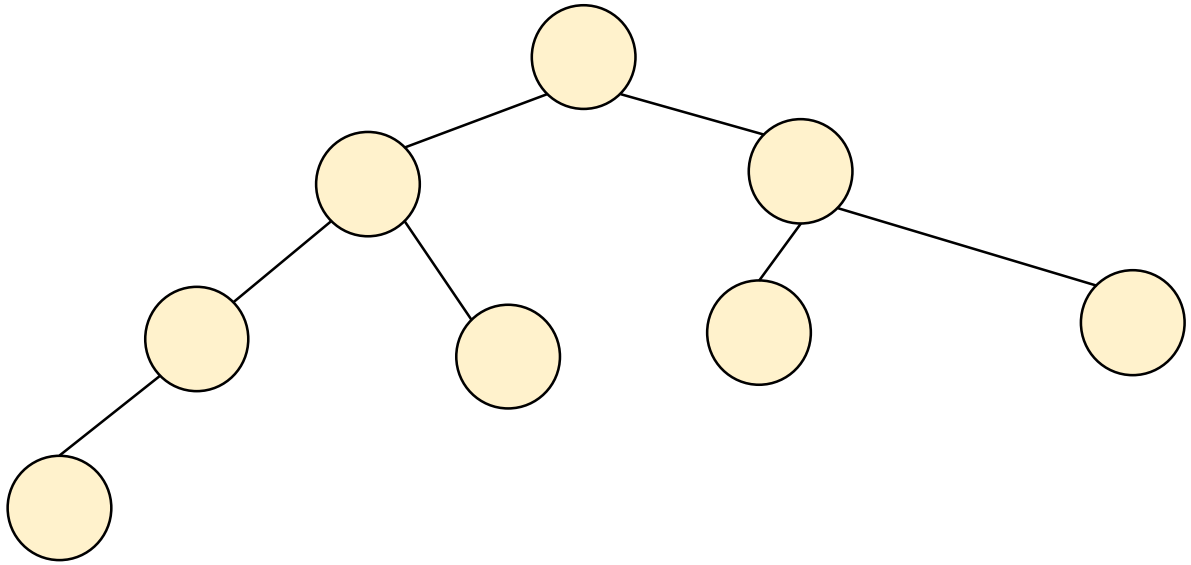
Leftist trees

Leftist trees

- They are binary trees.
- They can perform everything that a heap can do in the same asymptotic complexity.
- We can meld two leftist tree priority queues in O(log n) time.

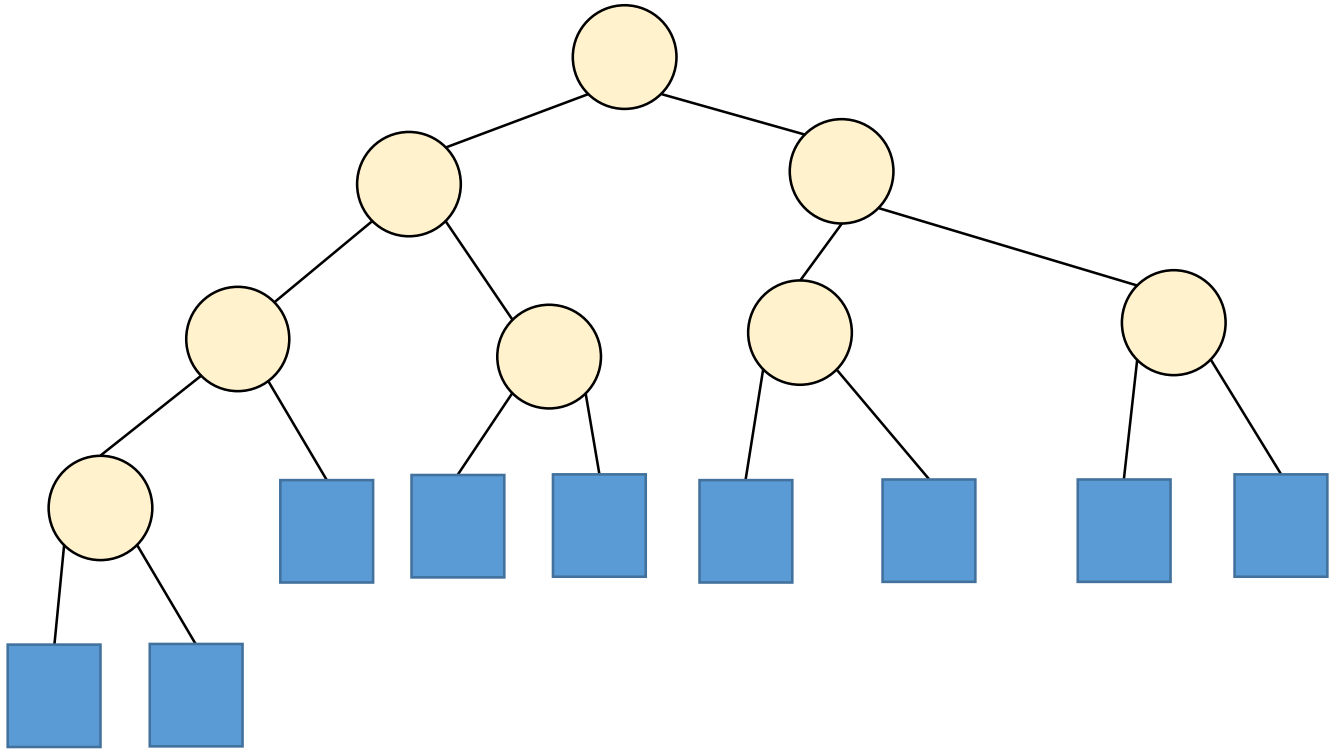
Extended Binary Trees

- We add an external node to each node of a binary tree that has an empty subtree.
- The resulting binary tree is an extended binary tree.



Extended Binary Trees

- We add an external node to each node of a binary tree that has an empty subtree.
- The resulting binary tree is an extended binary tree.

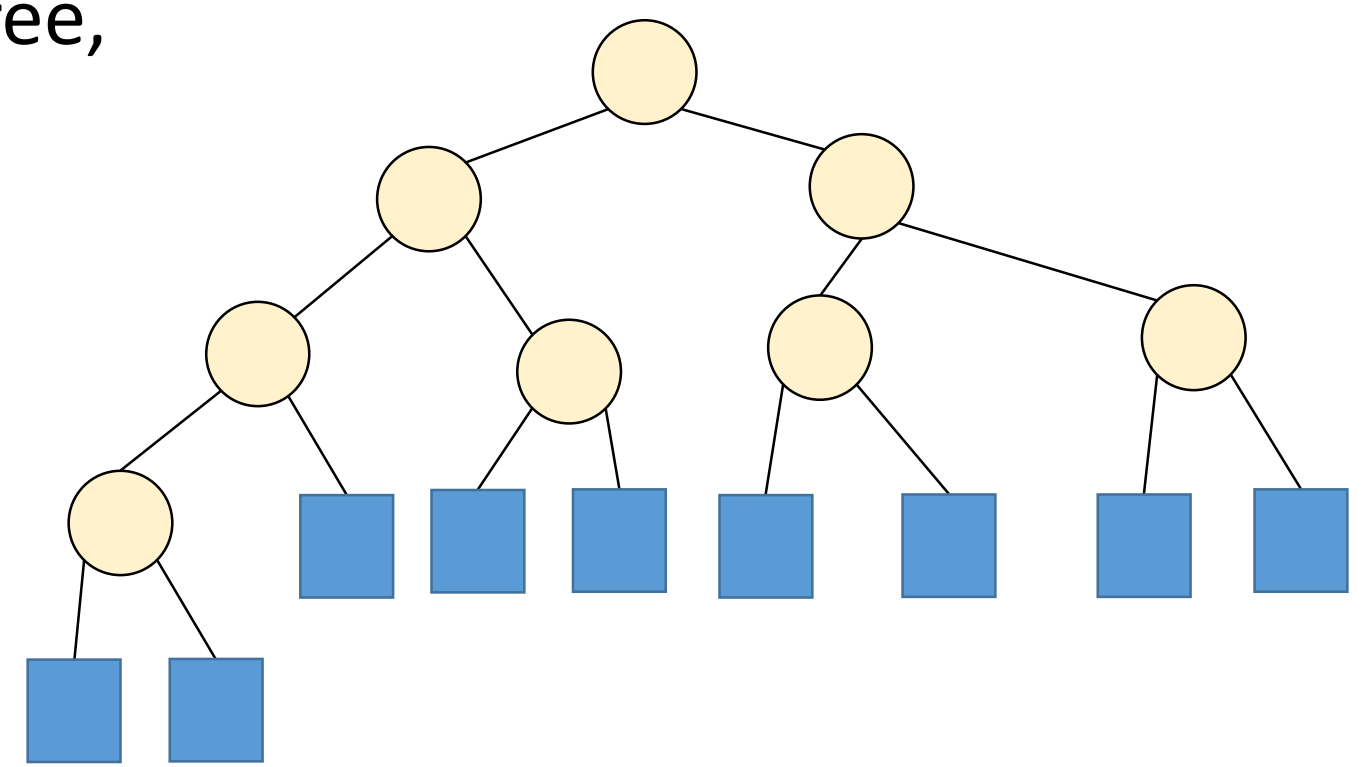


Extended Binary Trees

- We add an external node to each node of a binary tree that has an empty subtree.
- The resulting binary tree is an extended binary tree.
- Given an n-node binary tree,

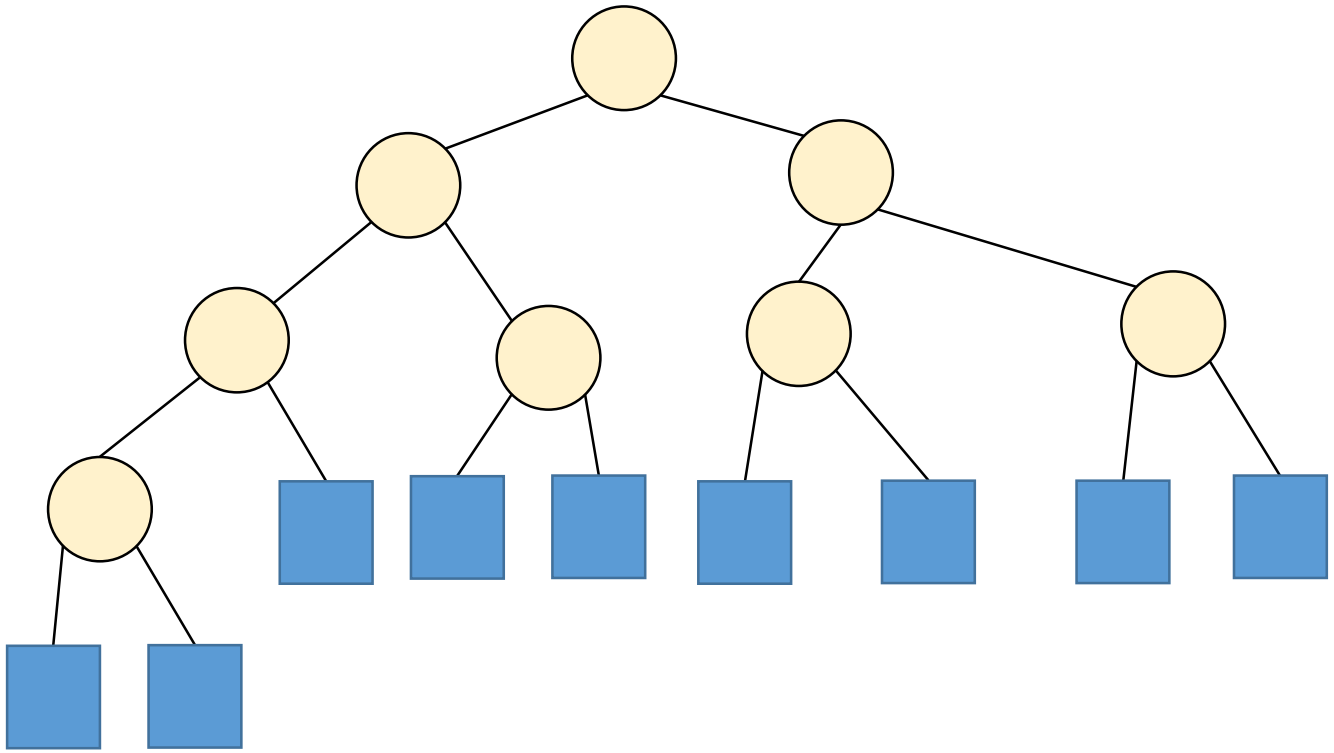
  the extended binary tree

  has n+1 nodes.



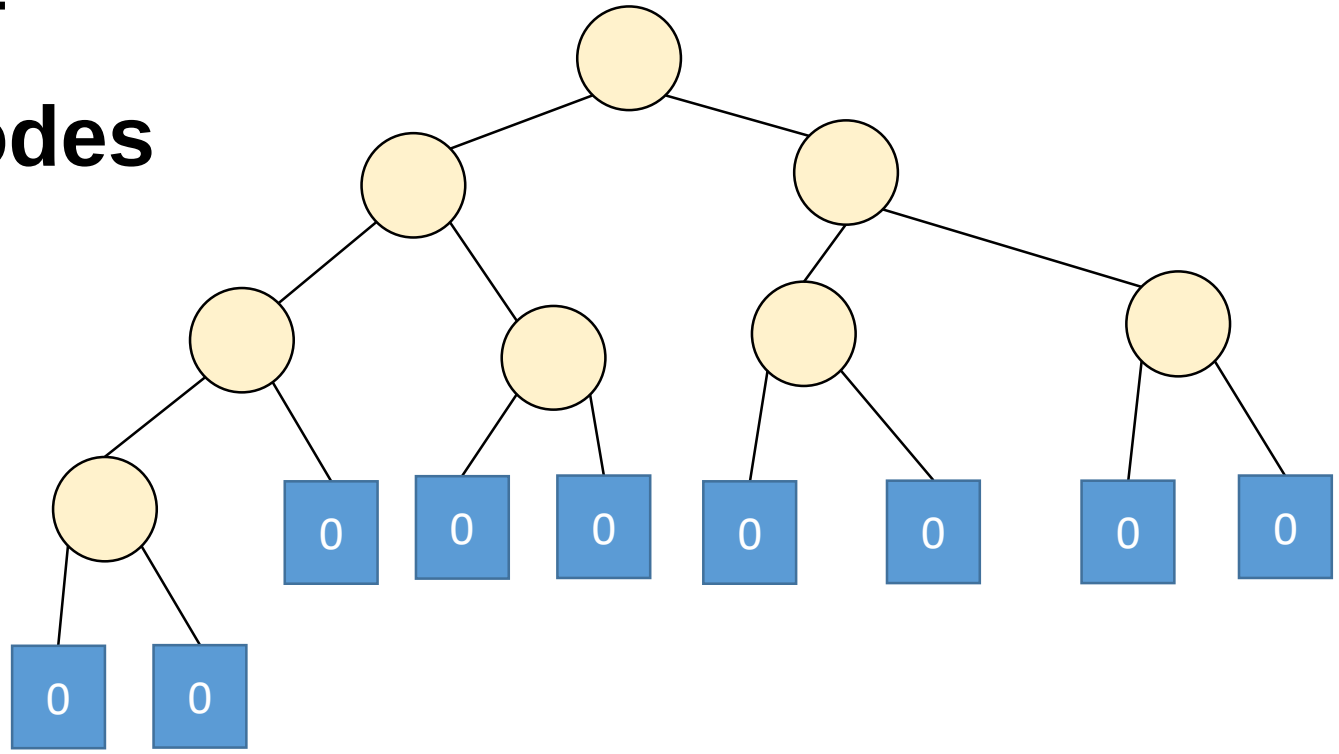
The Function s()

- A node x has a value s(x).
- Define s(x) as the length of a shortest path from x to an external node in the subtree rooted at x.



The Function s()

- A node x has a value s(x).
- Define s(x) as the length of a shortest path from x to an external node in the subtree rooted at x.
- **The s value of all external nodes is zero.**

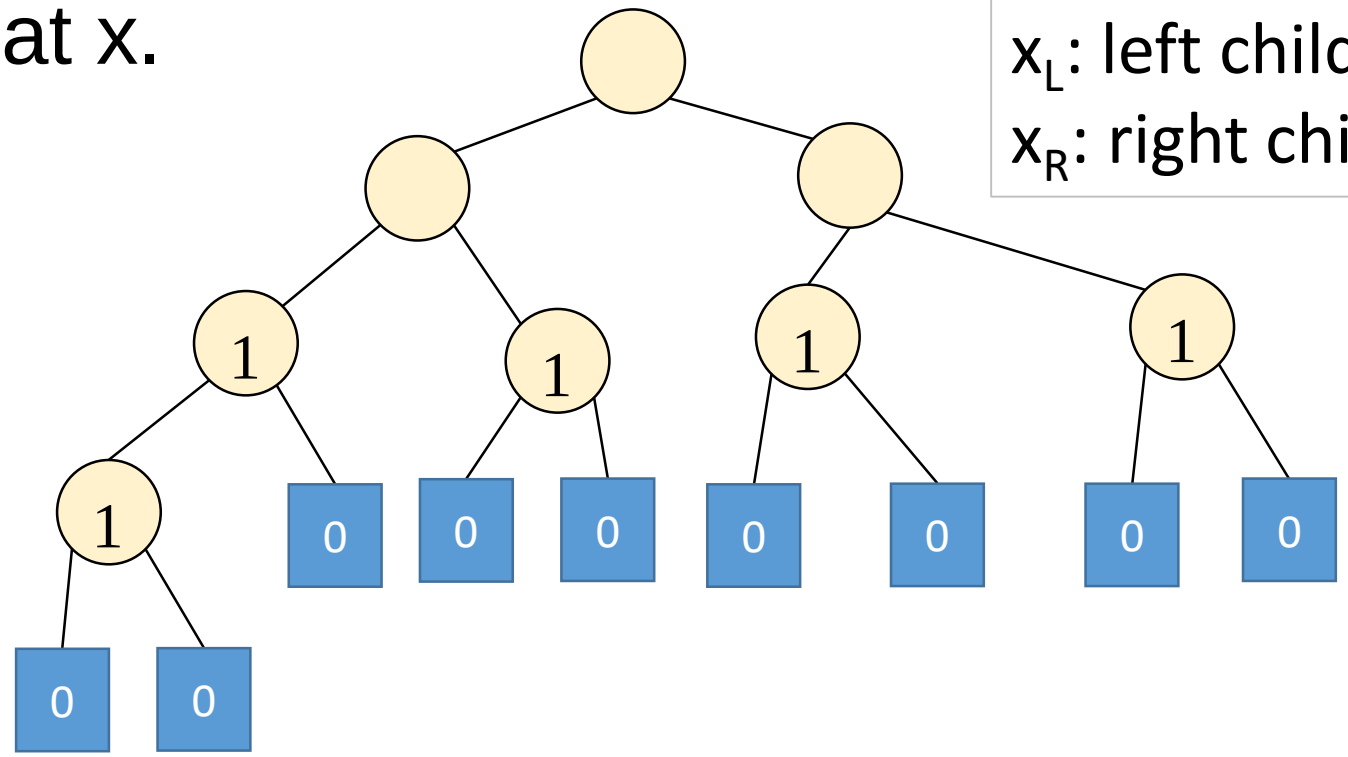


The Function s()

- A node x has a value s(x).
- Define s(x) as the length of a shortest path from x to an external node in the subtree rooted at x.

s(x)
= 1+min ($s(x_L)$, $s(x_R)$)

x_L: left child
x_R: right child

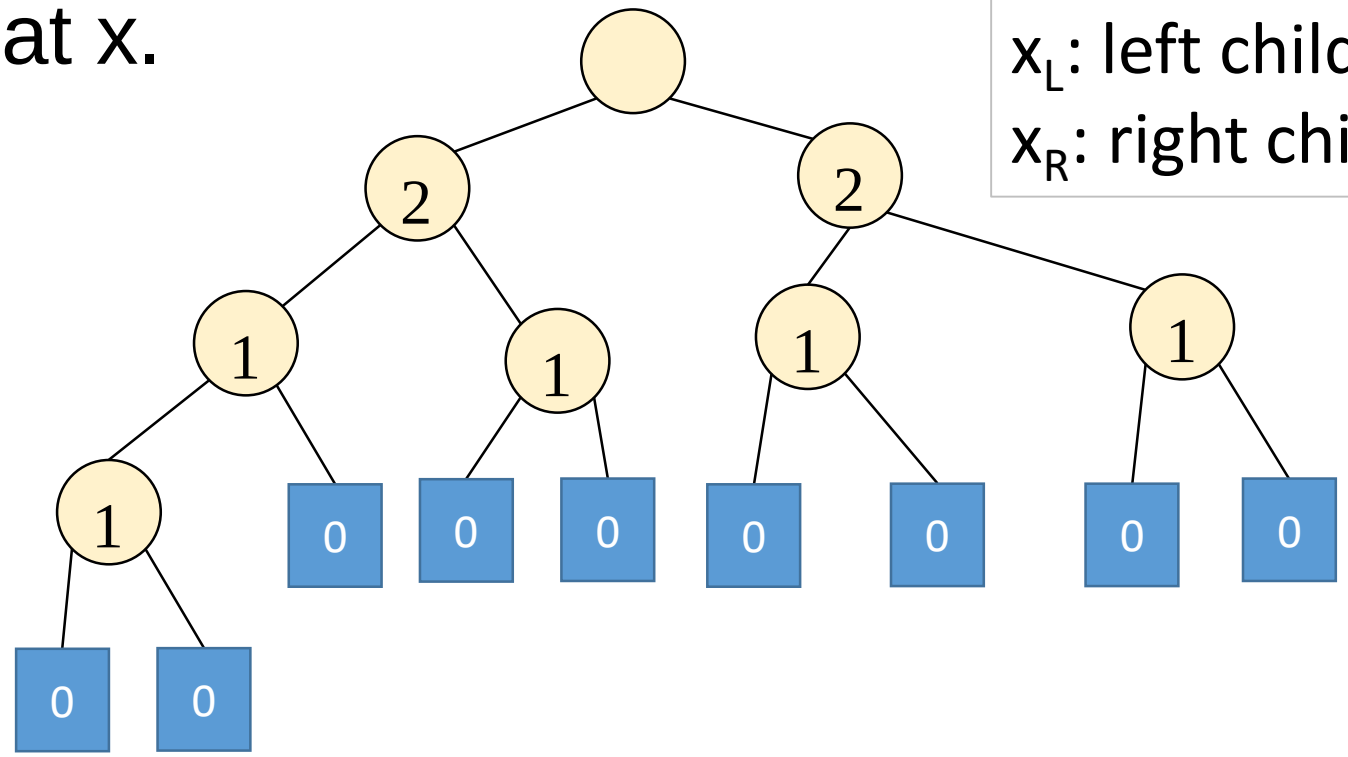


The Function s()

- A node x has a value s(x).
- Define s(x) as the length of a shortest path from x to an external node in the subtree rooted at x.

s(x)
= 1+min ($s(x_L)$, $s(x_R)$)

x_L: left child
x_R: right child

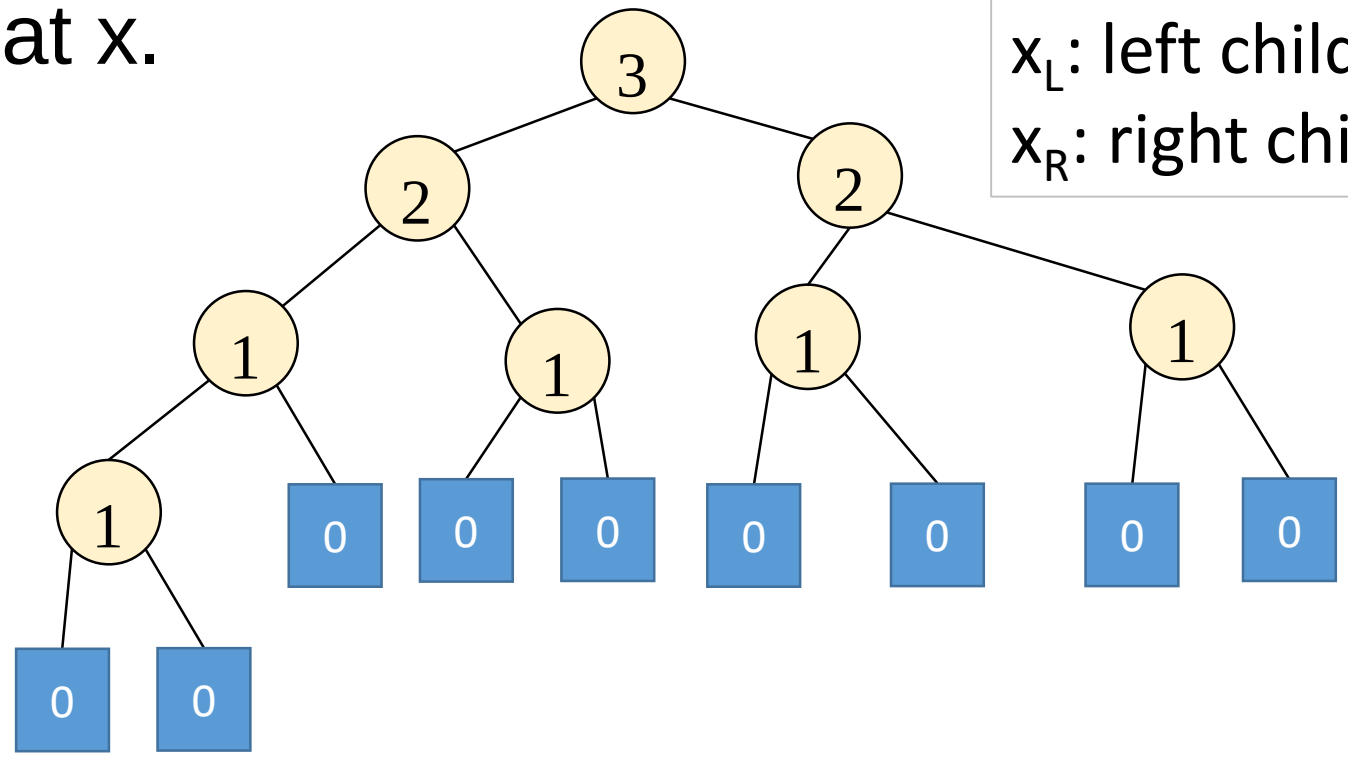


The Function s()

- A node x has a value s(x).
- Define s(x) as the length of a shortest path from x to an external node in the subtree rooted at x.

s(x)
= 1+min ($s(x_L)$, $s(x_R)$)

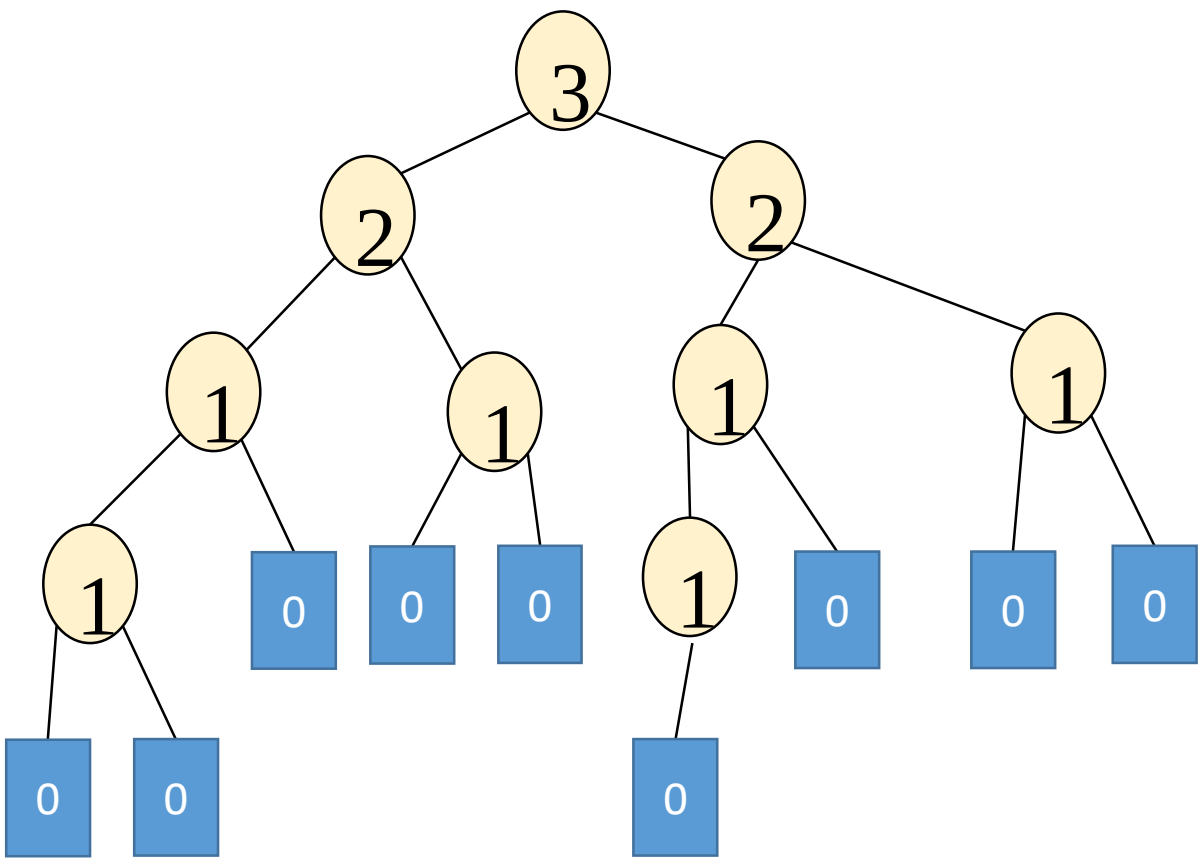
x_L: left child
x_R: right child

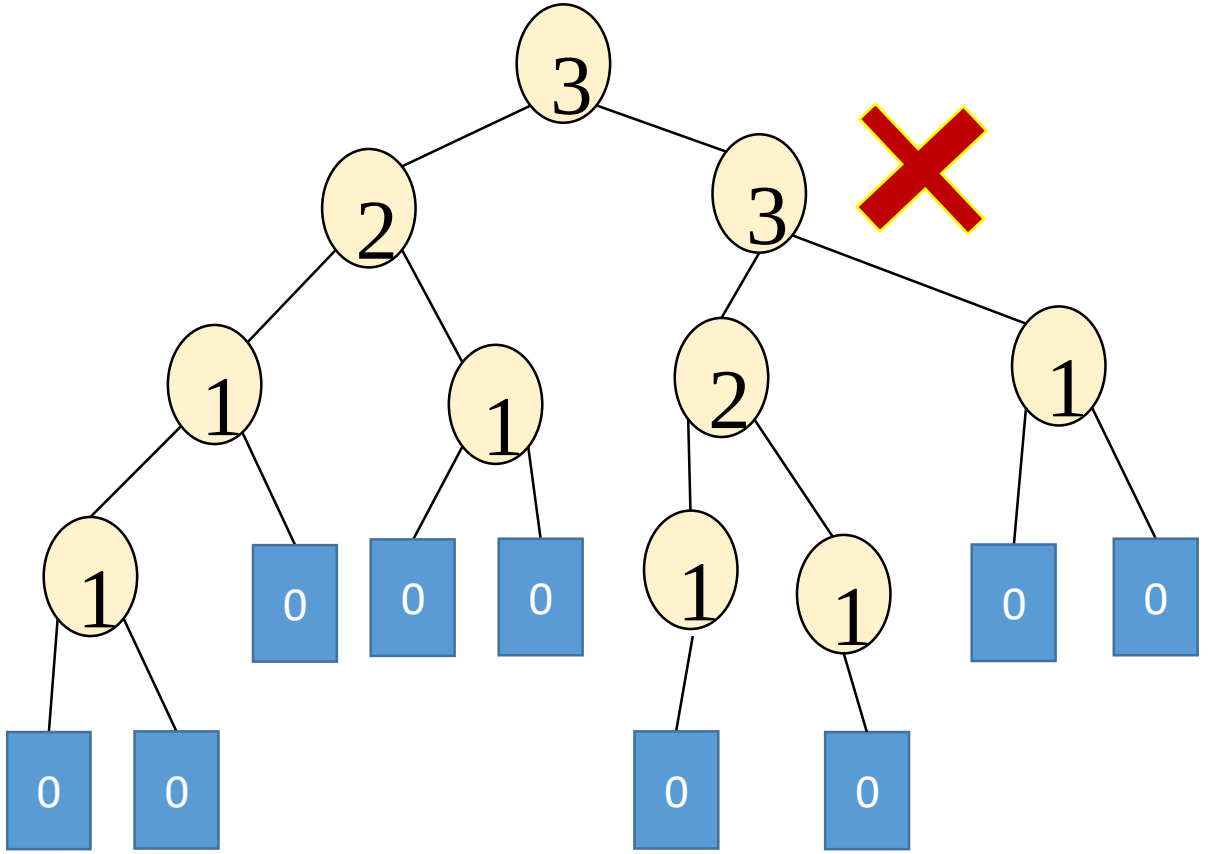


Height Biased Leftist Trees

A binary tree is a leftist tree iff for every internal node x,

s(leftChild(x)) >= s(rightChild(x))

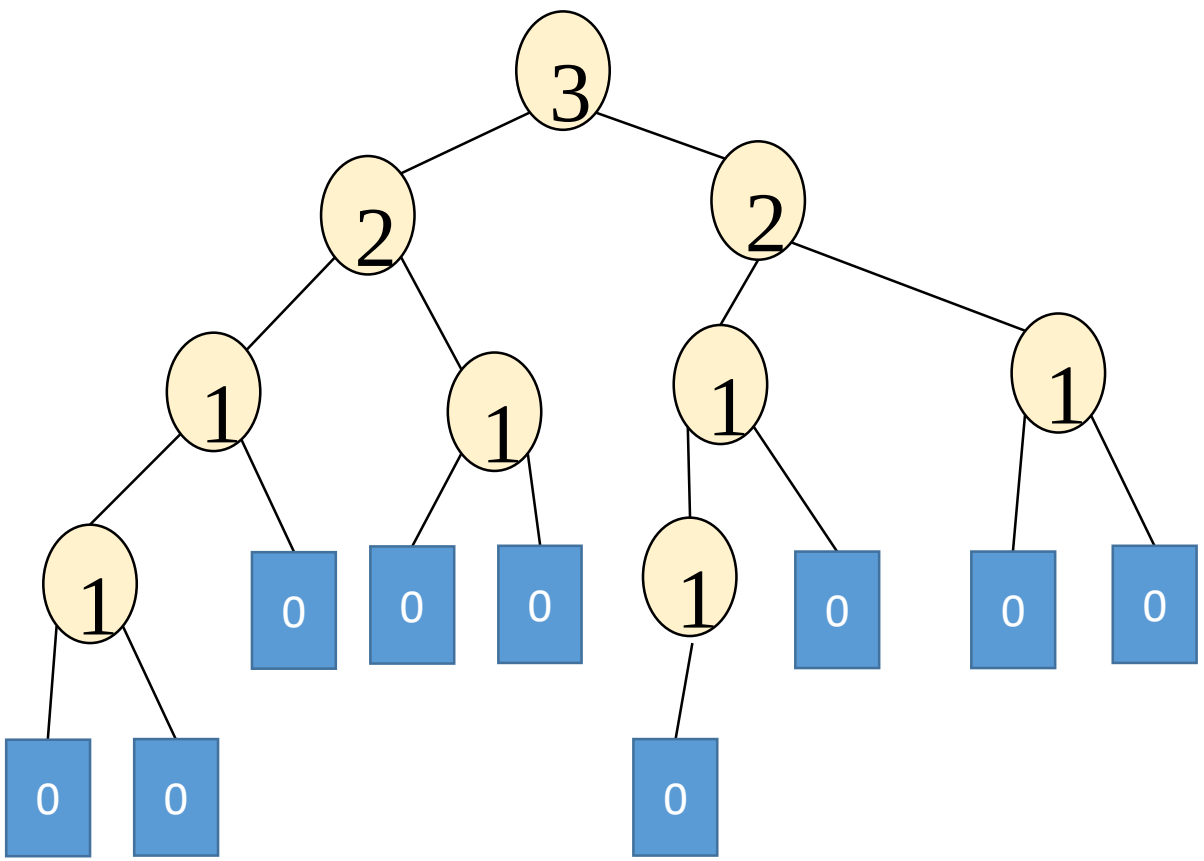


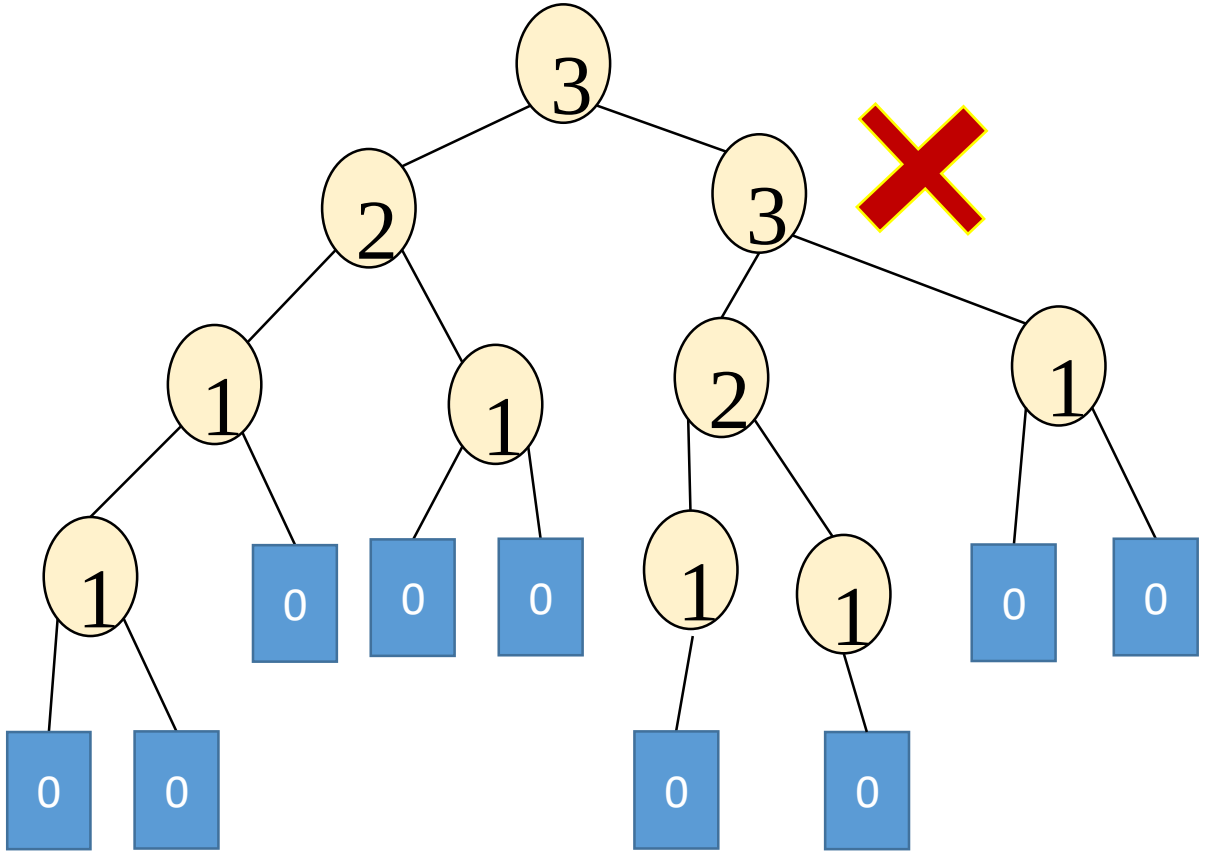


Height Biased Leftist Trees

A binary tree is a leftist tree iff for every internal node x,

s(leftChild(x)) >= s(rightChild(x))

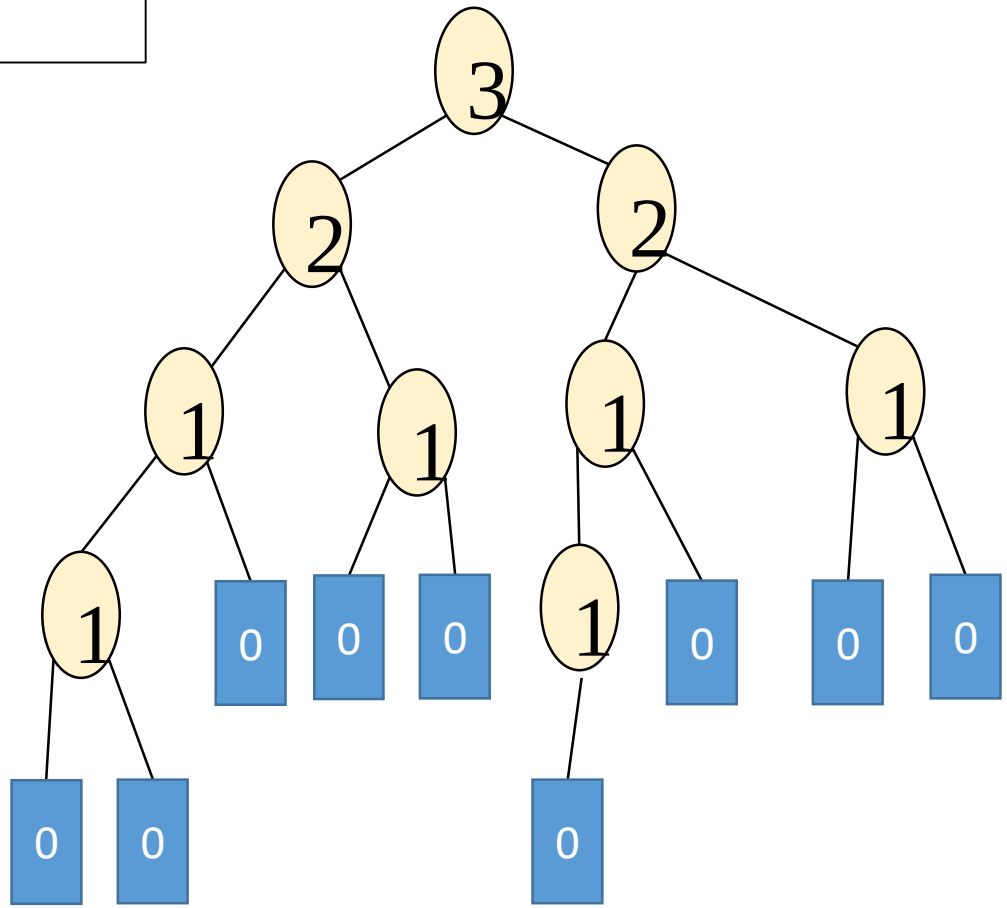




Leftist Trees--Property 1

A binary tree is a leftist tree iff for every internal node x,

s(leftChild(x)) >= s(rightChild(x))

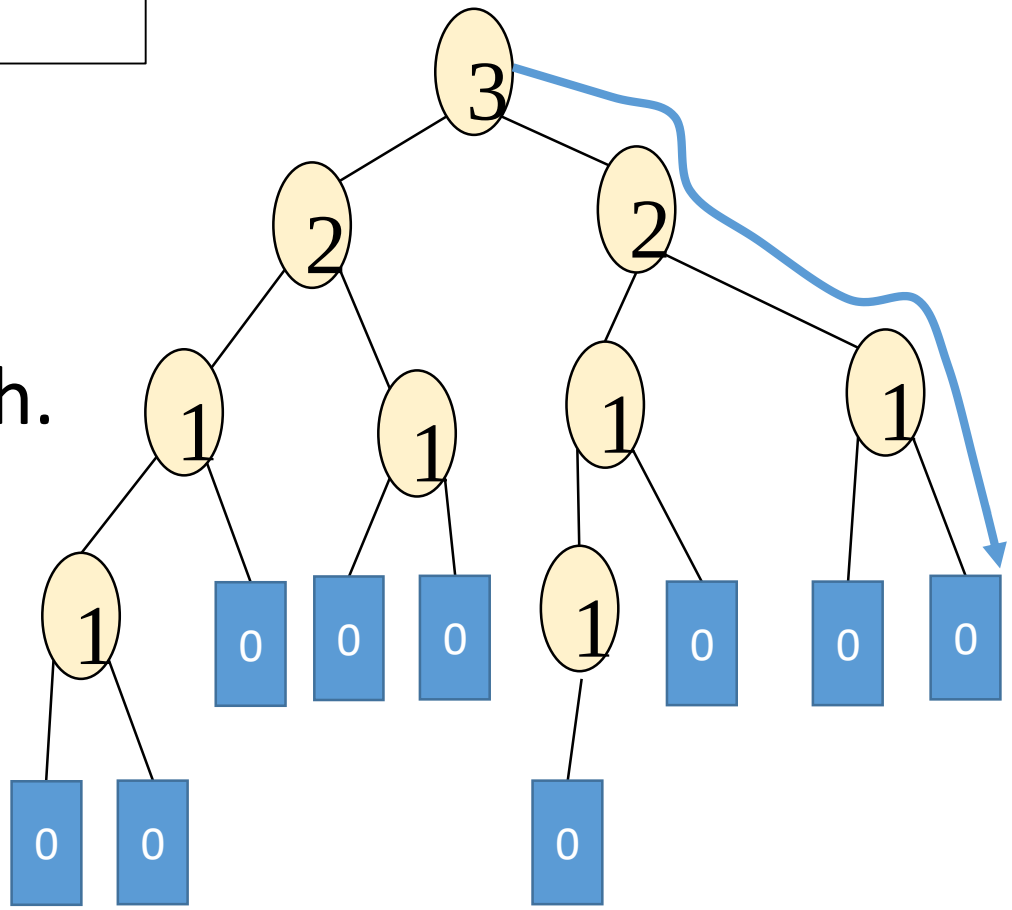


Leftist Trees--Property 1

A binary tree is a leftist tree iff for every internal node x,

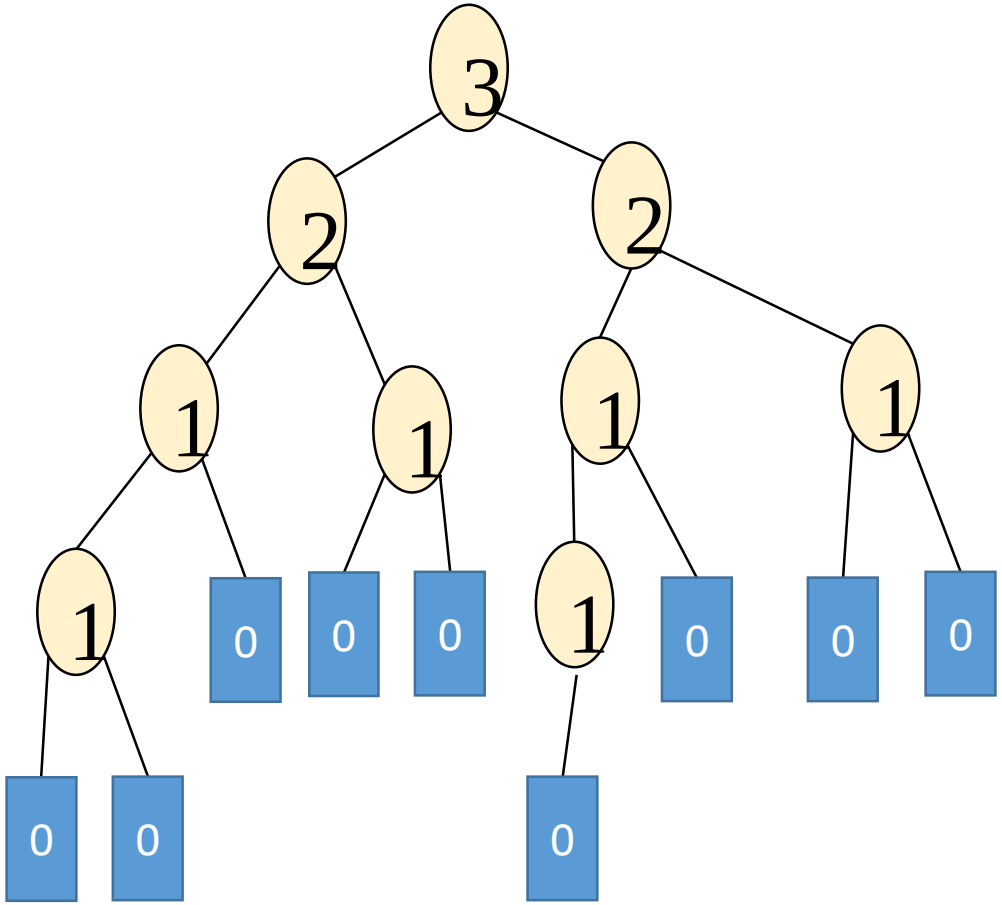
s(leftChild(x)) >= s(rightChild(x))

- The rightmost path is a shortest path from the root to external node.
- We call this path the shortest root path.
- The length of this path is s(root).



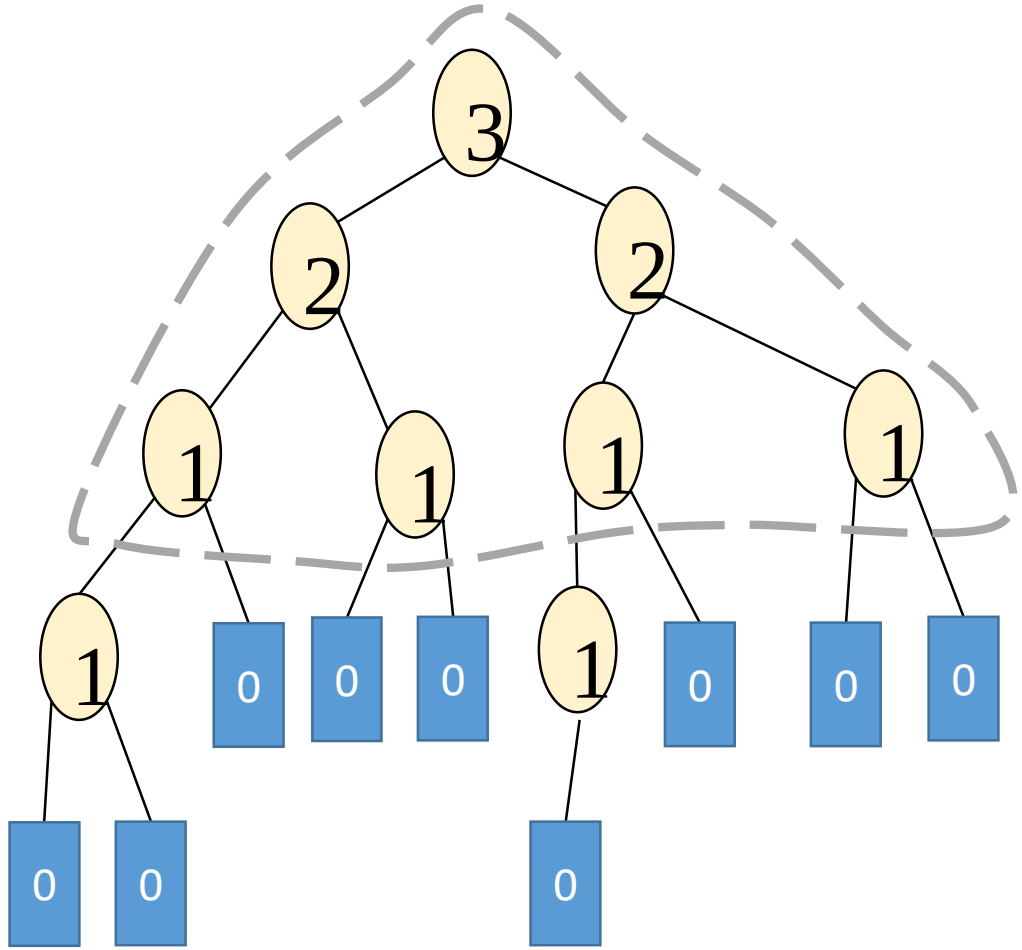
Leftist Trees--Property 2

- The number of internal nodes is at least $2^{s(root)} - 1$
- Levels 1 through s(root) have no external nodes.
- $2^{s(root)} - 1 <= n$
- We have s(root) <= log(n+1)



Leftist Trees--Property 2

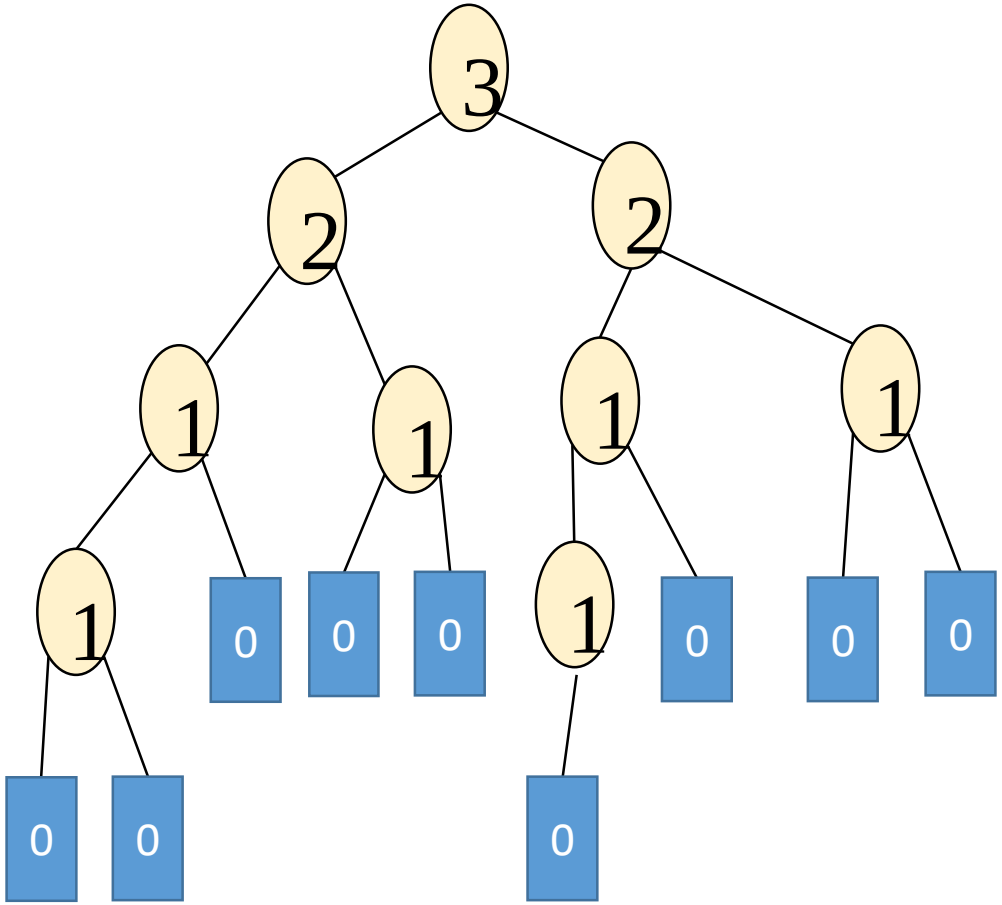
- The number of internal nodes is at least $2^{s(root)} - 1$
- Levels 1 through s(root) have no external nodes.
- $2^{s(root)} - 1 <= n$
- We have s(root) <= log(n+1)



Remark: A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

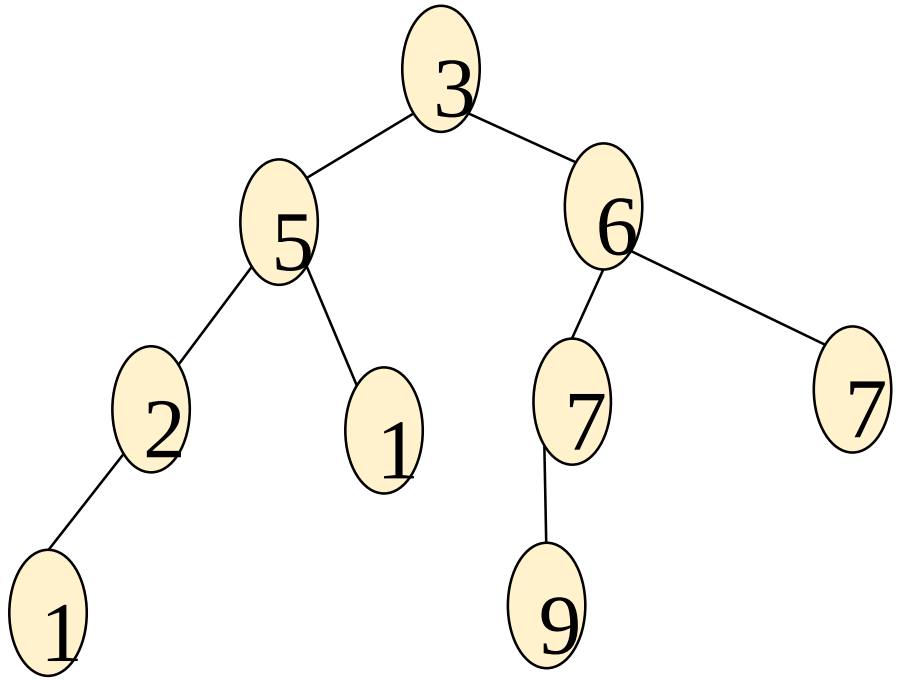
Leftist Trees--Property 3

- The number of internal nodes is at least $2^{s(root)} - 1$
- Levels 1 through s(root) have no external nodes.
- $2^{s(root)} - 1 <= n$
- We have $s(root) <= \log_2(n+1)$
- Thus, $s(root) = O(\log_2 n)$



Leftist Trees As Priority Queues

- Min leftist tree: a min tree.
  - Used as a min priority queue.
- Max leftist tree: a max tree.
  - Used as a max priority queue.



The external nodes are not drawn.
The numbers are elements in the nodes.

Supplemental Materials

Merging height biased leftist trees

After the trees are merged, the resulting leftist tree must maintain the properties of leftist trees.

Meld: operation