Binary search trees are **recursive data structures**

* Each node is a root of 0,1,2 sub-trees
* LHS values = **smaller** **than root**
* RHS values = **larger than root**

Operations on trees:

* Insert(Tree, item) / delete(Tree, item) / search(Tree, item)
* Housekeeping = New(); dispose(); show(); empty();

Nodes contain Items. We just show **Item.key.**

Shape of trees is determined by order of insertion (balanced vs non-balanced). We want trees to be as shallow as possible.

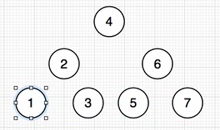
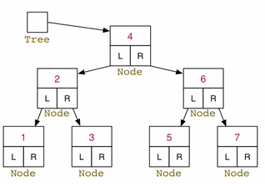
**LEVEL of node** = path length from ROOT 🡪 NODE

**DEPTH of tree** = max path length from ROOT 🡪 LEAF

* Depth of tree with N nodes:
  + Min depth = **floor(log2n)**
  + Max depth = **n-1**
* Height balanced tree: **depth(LHS subtree)** = **depth(RHS subtree)**
* Time complexity of tree = **O(depth)**

Insertion into BSTs:

* Example: 4 2 6 5 1 7 3

Representation of BSTs:

