2-3-4 trees

2-3-4 trees

- A search tree such that
  - Each non-leaf node has 2 or 3 or 4 children ≡ each node has 1 or 2 or 3 keys (in ascending order)
  - All leaf nodes are at the same level.

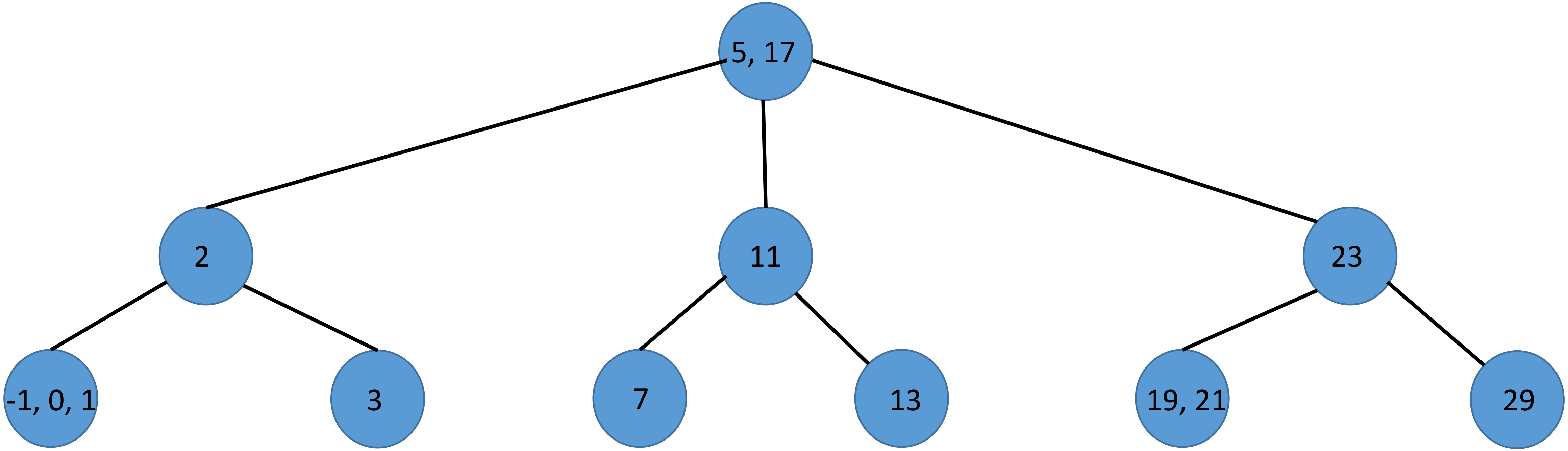
- A 2-3-4 tree on n nodes has the maximum height of $\log_2 n$ when all its nodes are 2-nodes and has the minimum height of $\log_4 n$ when all its nodes are 4-nodes. So, the height of a 2-3-4 tree on n nodes has height $O(\log_2 n)$.

- A 2-3-4 tree can be converted into a Red-Black tree by
  - Splitting a 3-node into two red-black nodes (one black, one red)
  - Splitting a 4-node into three red-black nodes (one black, two red)

Searching in a 2-3-4 tree

O(log n)

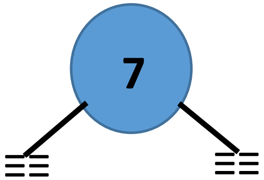
- Search(x, T)
    - Start the search at the root node. Take the
        - Appropriate branch
- **Note:** Search key x is to be compared with at most 3 keys at each node (on the search path). So, 3h is the maximum number of comparisons.



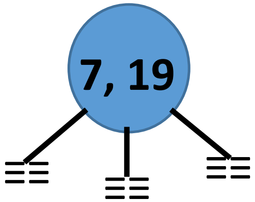
Inserting in a 2-3-4 tree

Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

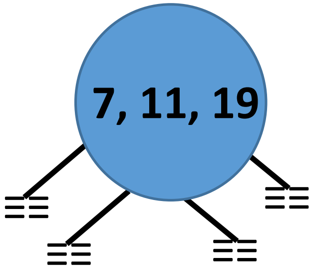
Insert 7



Insert 19



Insert 11

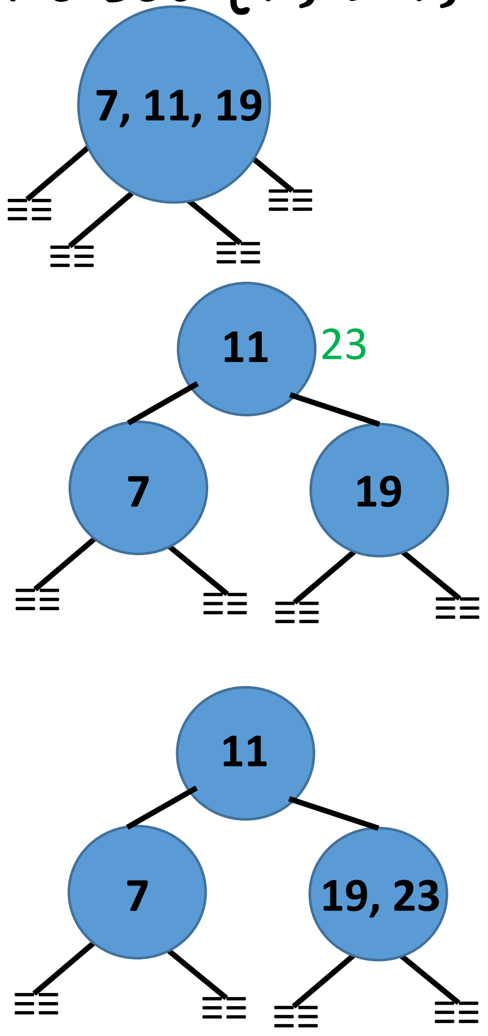


Insert key is always placed in a leaf node identified by the search path.

Inserting in a 2-3-4 tree

Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

Insert 23



Bottom-Up approach:

- Split when you must – if inserting key k into a node x would exceed the number of permitted keys, then split x before inserting k.
- The split process begins at a leaf node.
- May cause an upward propagation of splits along the search path.

Top-Down approach:

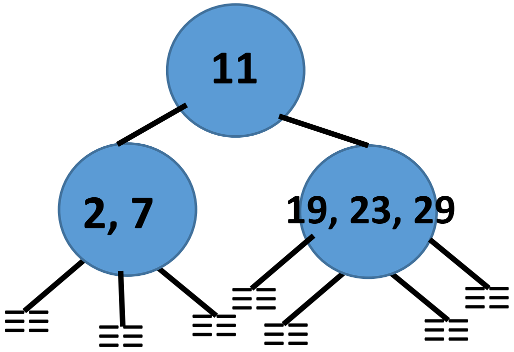
- Split when you can – never enter a full node if it is on the search path; even if the insertion key k is not to be inserted into the node.
- The split process begins at any node on the search path.
- Does not cause any propagation of splits.

Top-down approach offers better multi-threading (i.e., improves concurrency), so it is preferred.

Inserting in a 2-3-4 tree

Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

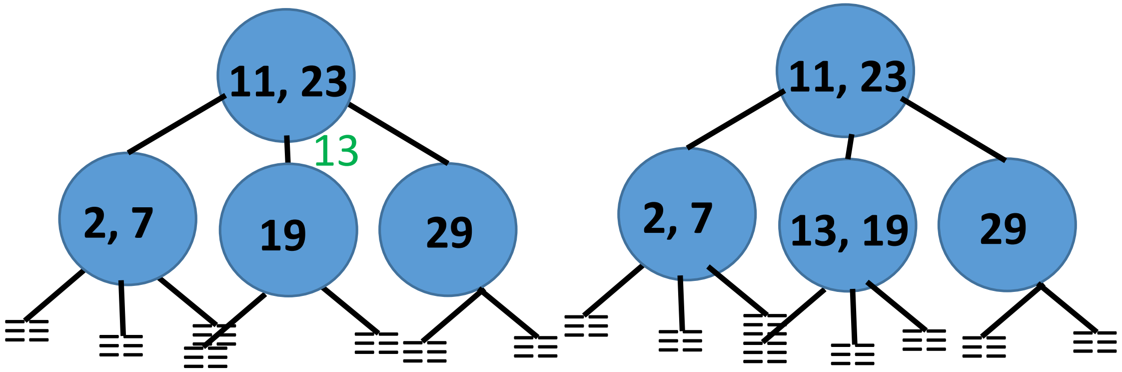
Insert 29, 2



Insert 13

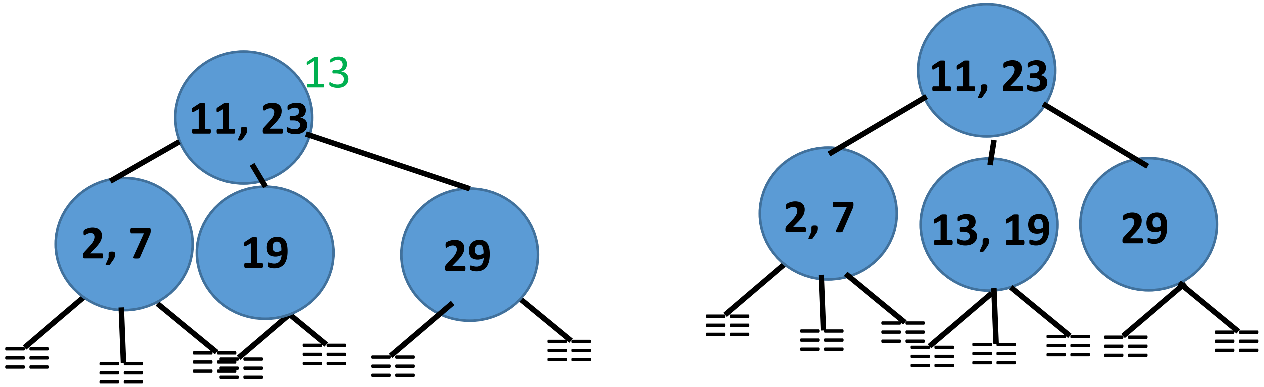
Bottom-Up approach:

- Identify [19, 23, 29] as the leaf node for insertion (of 13 to the left of 19)
- Since the location is a full node, split it before insertion.



Top-Down approach:

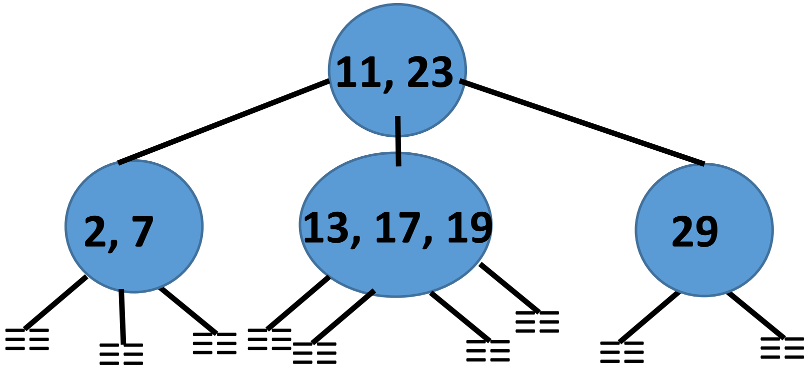
- [19, 23, 29] is a full node on the search path, so split it before continuing with the search.



Inserting in a 2-3-4 tree

Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

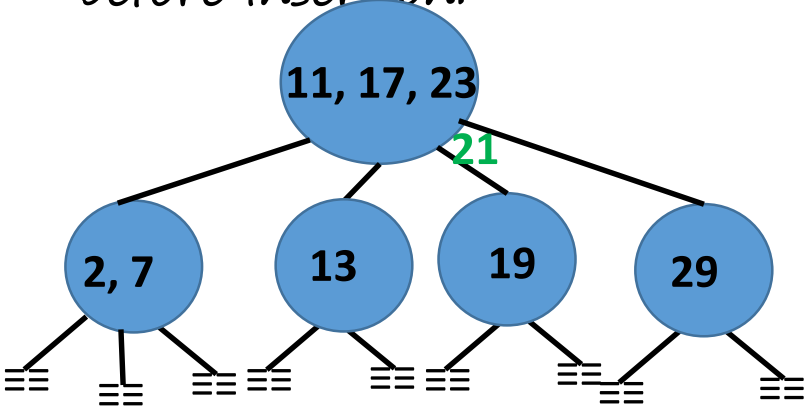
Insert 17

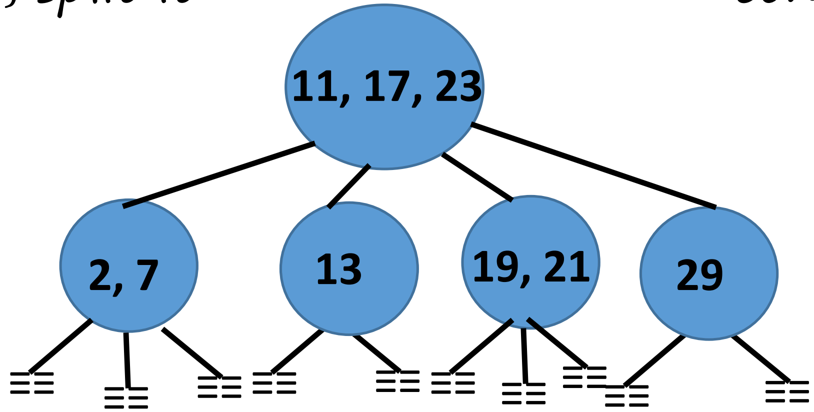


Insert 21

Bottom-Up approach:

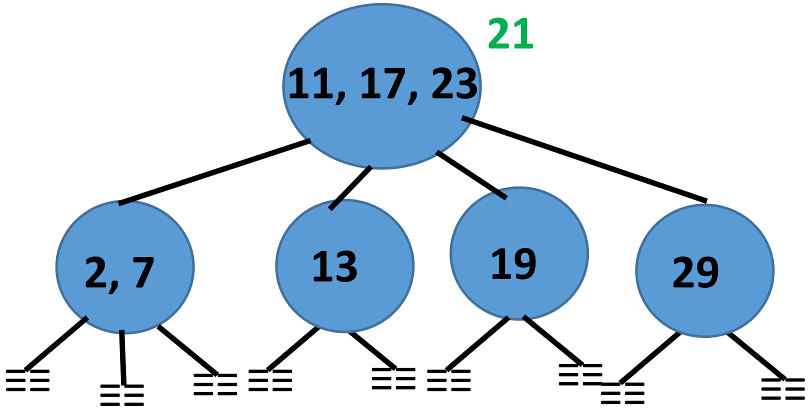
- Identify [13, 17, 19] as the leaf node for insertion (of 21 to the right of 19)
- Since the location is a full node, split it before insertion.





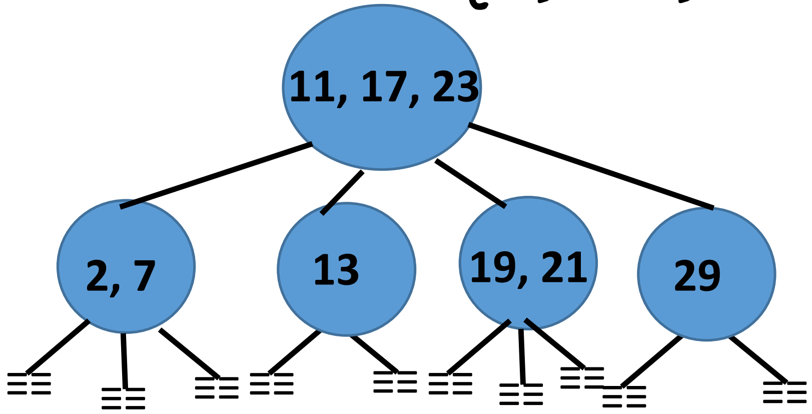
Top-Down approach:

- [13, 17, 19] is a full node on the search path, so split it before continuing with the search.



Inserting in a 2-3-4 tree

Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}



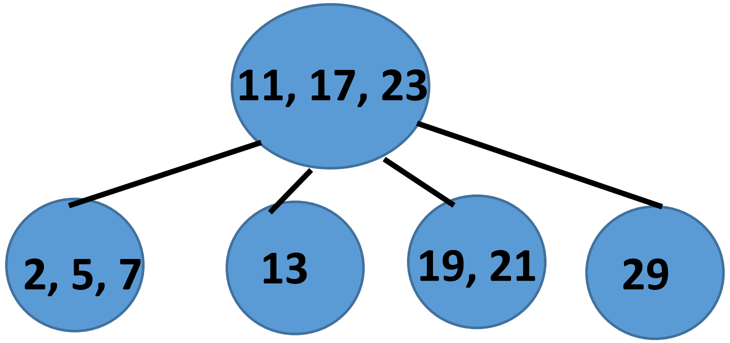
The two approaches produce different 2-3-4 trees.

The set of leaf nodes are the same.

Insert 5

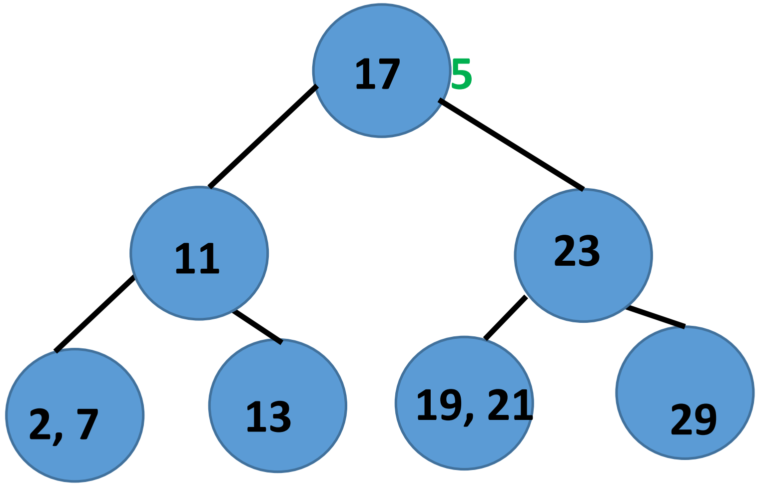
Bottom-Up approach:

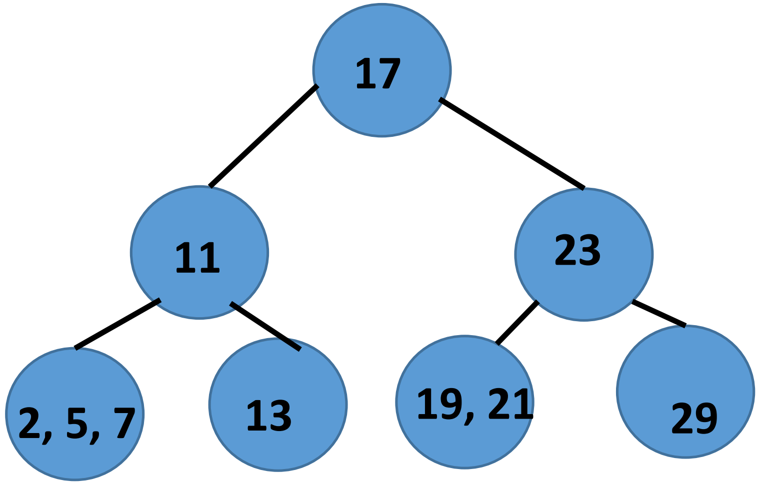
- No split required.



Top-Down approach:

- [11, 17, 23] is a full node on the search path, so split it before continuing with the search.

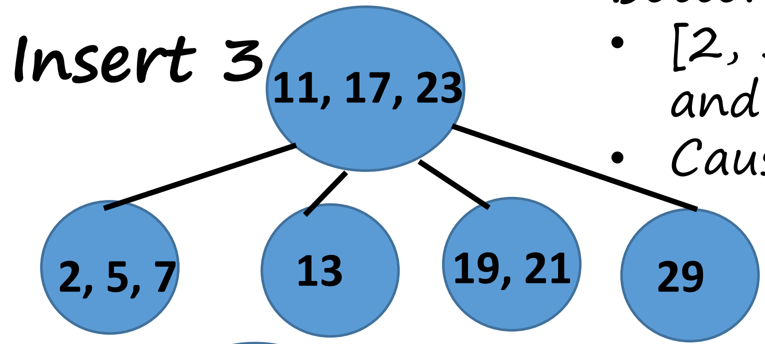




Inserting in a 2-3-4 tree

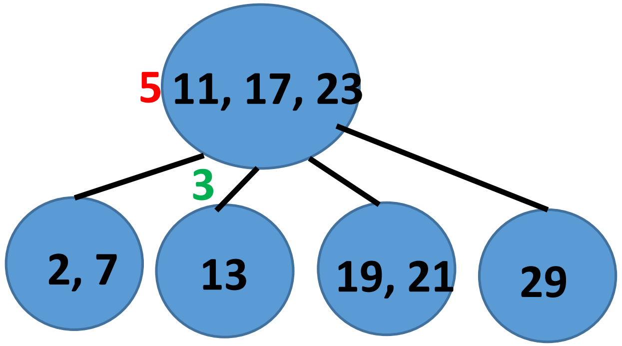
Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

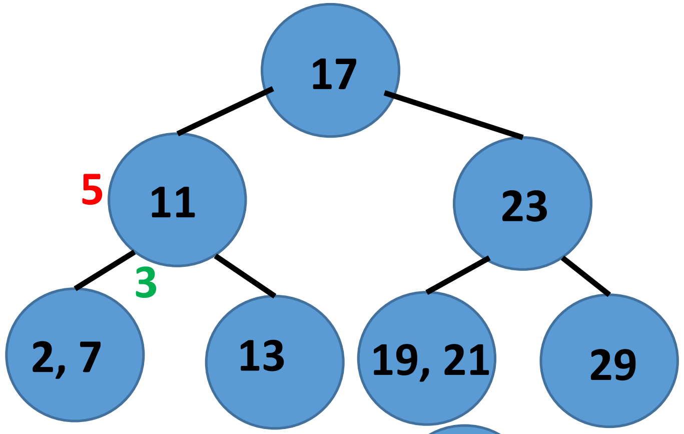
Insert 3

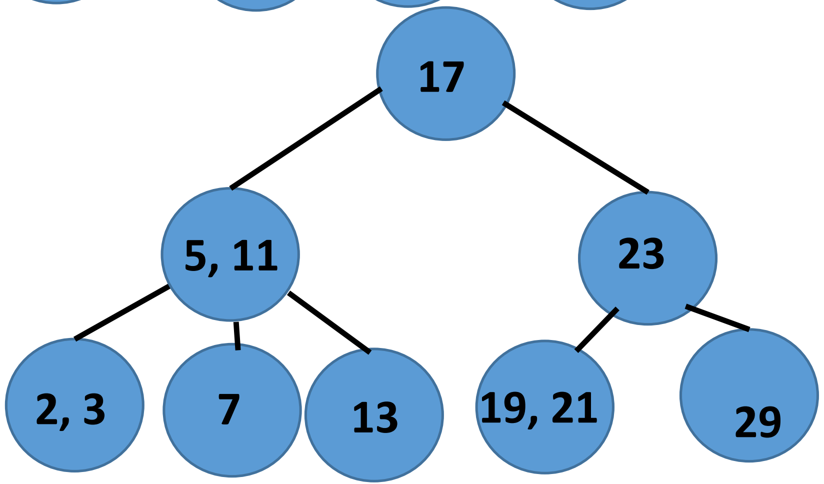


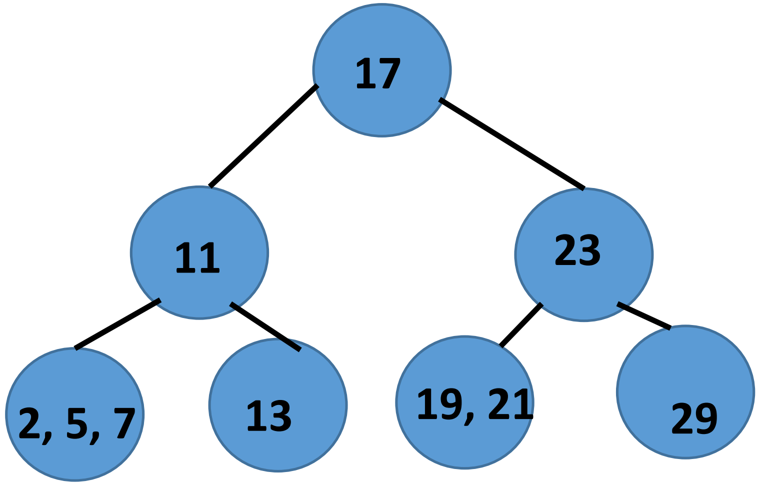
Bottom-Up approach:

- [2, 5, 7] is the location for insertion and is a full node, so split it.
- Cause propagation of split.



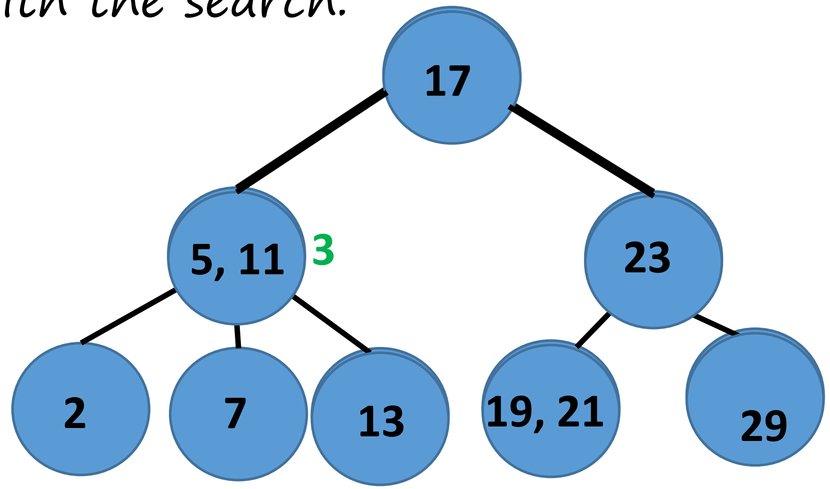






Top-Down approach:

- [2, 5, 7] is a full node on the search path, so split it before continuing with the search.

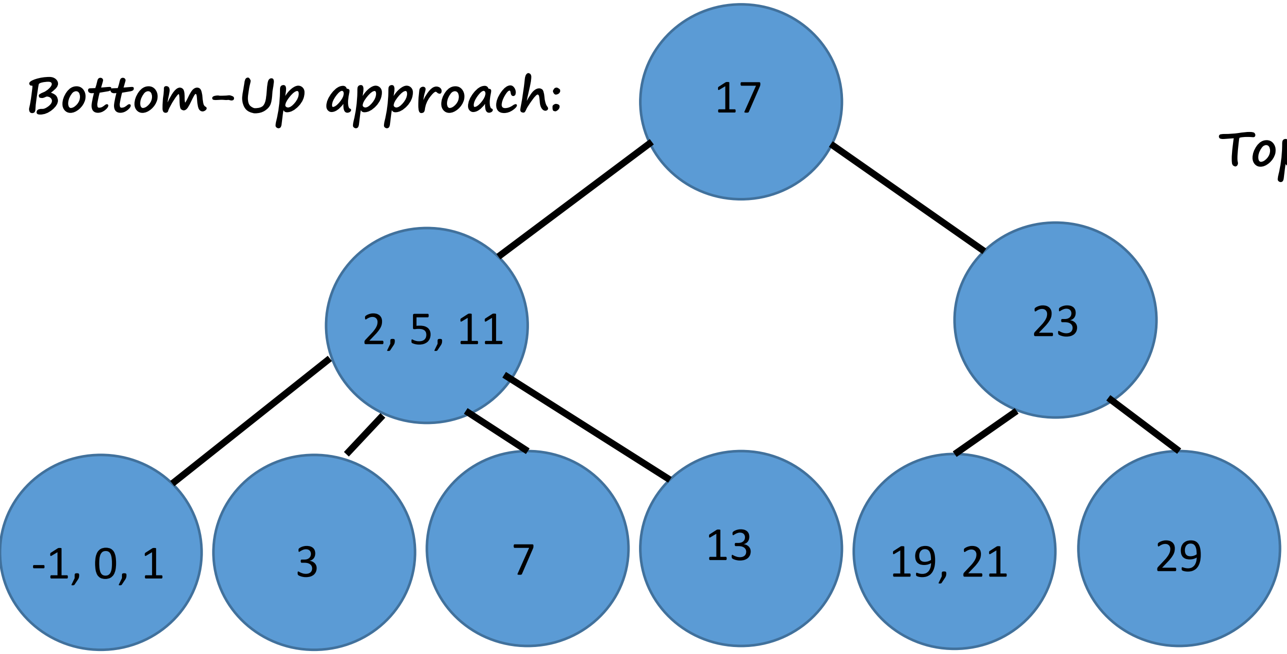


Inserting in a 2-3-4 tree

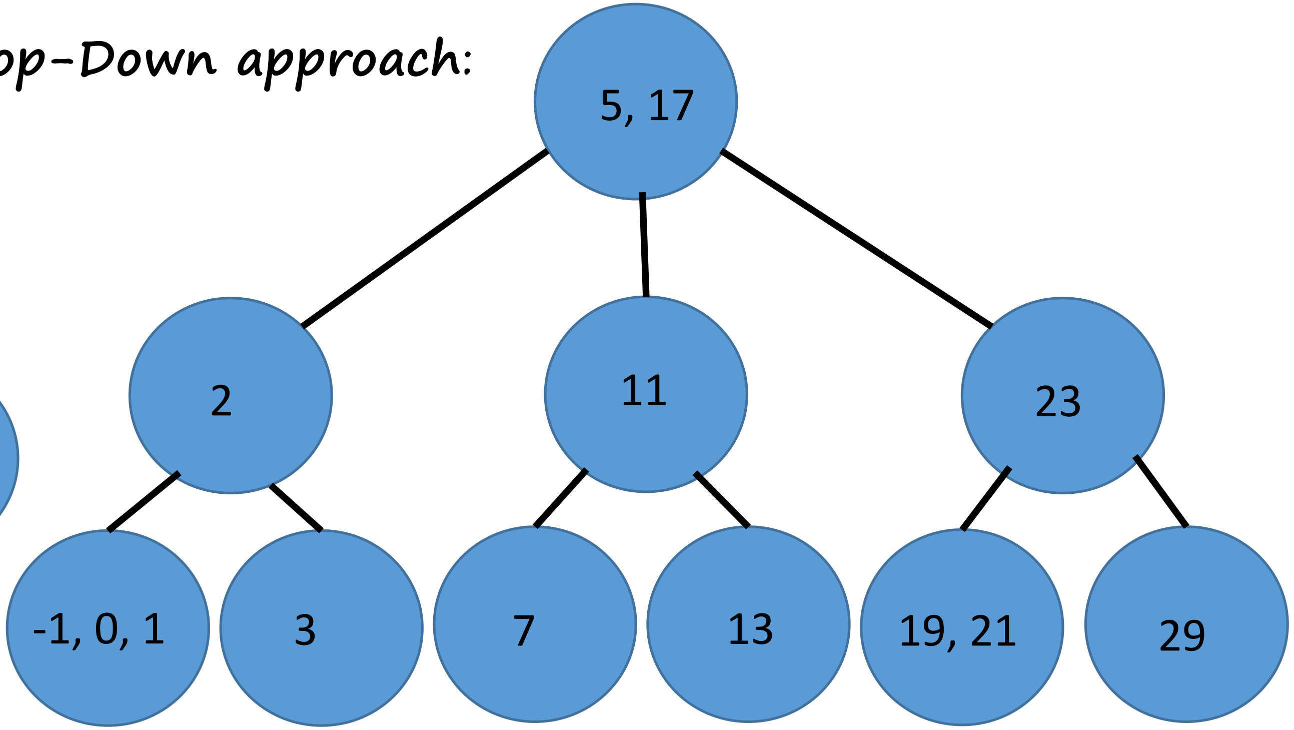
Construct a 2-3-4 tree for the set {7, 19, 11, 23, 29, 2, 13, 17, 21, 5, 3, 1, -1, 0}

Insert 1, -1, 0

Bottom-Up approach:



Top-Down approach:



The two approaches produce different 2-3-4 trees.
The set of leaf nodes are the same.

Proof of correctness of top-down insertion

- When a split is done at node x, its parent is not already a full node. So, every node except node x remain unaltered; i.e., remain as 2-3-4 nodes.
- The node x is split into two 2-nodes (and its children get distributed between these two 2-nodes).
- The splitting at node x increases the
  - Keys at Parent[x] by one.
  - Children at the Parent[x] by two but removes node x from its children; resultant increase is one.
- The number of levels do not increase (except when split happens at the root); so the leaf nodes are all at the same level.
- Irrespective of an insertion requiring a split or not, the resultant is a 2-3-4 tree.

Run-time for split = $O(1)$, # of splits ≤ 1, Total run-time = $O(h) = O(\log n)$

Exercise

Deletion in a 2-3-4 tree such that the resultant is a 2-3-4 tree.