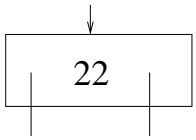
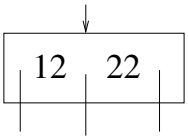
**CSC221 Spring 2012**
**Data Structures & Algorithms I**
**Inserting in a 2-3 Tree**

We begin with an empty tree, represented by a NULL pointer.

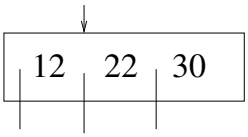
The first data item inserted is **22**. The B-tree is illustrated below:




The second data item inserted is **12**. There is enough space in the node for the new data item.

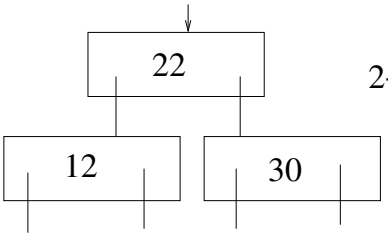



When the third data item, **30**, is inserted, the single node overflows (too many data items in one node). Overflow is resolved by splitting the node and passing the middle element up to the parent node. When the root of the tree splits, the tree grows in height, as illustrated below:



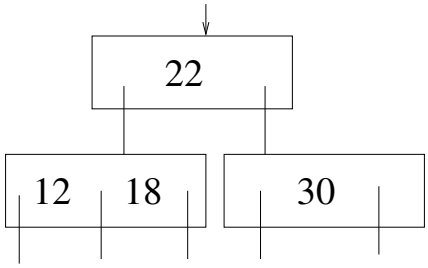

Temporary overflow state.

Split the node and promote the middle data item.

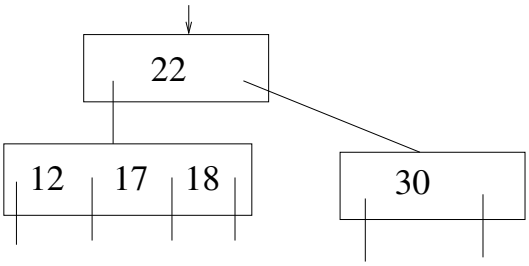



2–3 tree properties restored

The fourth data item, **18** fits in an available node. Notice that the insert is always done on a leaf node.

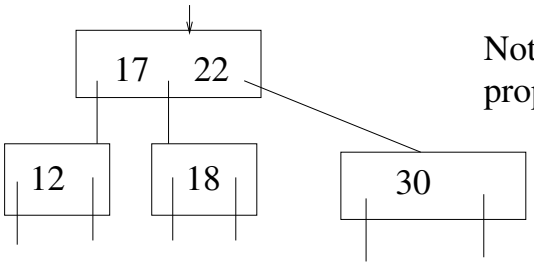



The fifth data item, **17** causes a leaf node to overflow and split:



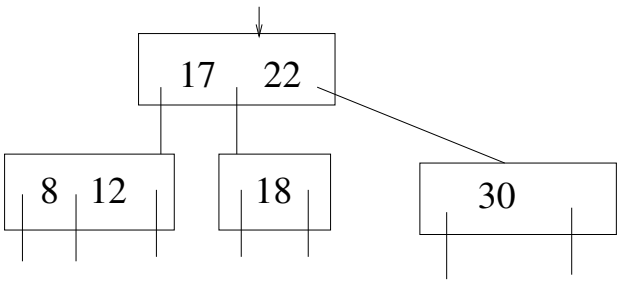

Temporary overflow

Split over−filled node and pass the middle element to the parent node.

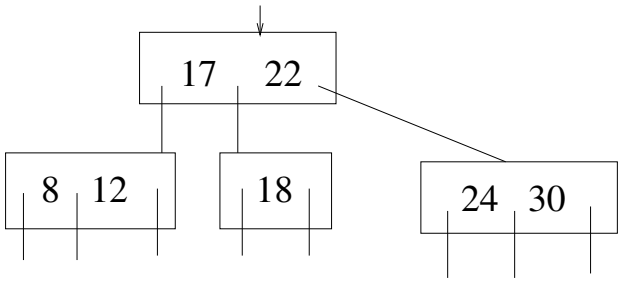



Notice that the extended search property is preserved.

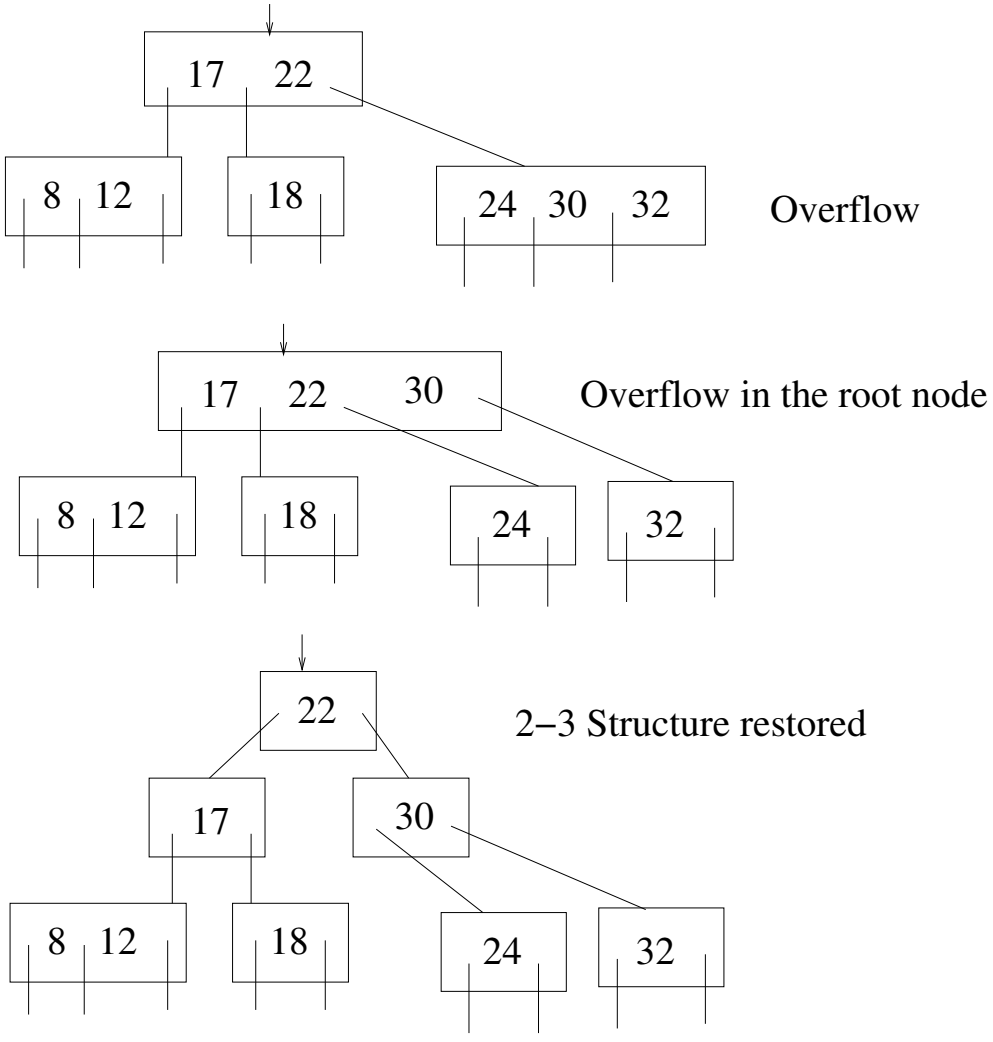
The sixth data item, **8** fits in the leftmost leaf node.



The seventh data item, **24** fits in the rightmost leaf node.




The eighth data item, **32** causes the rightmost leaf node to overflow. The middle data item, **30** is passed up. Subsequently, the root node overflows. The root splits, and the middle data element, **22** is moved to the newly created root node.



The ninth data item, **19** fits in the middle-left leaf node. Our final tree is: