2-3 Trees, 2-3-4 Trees

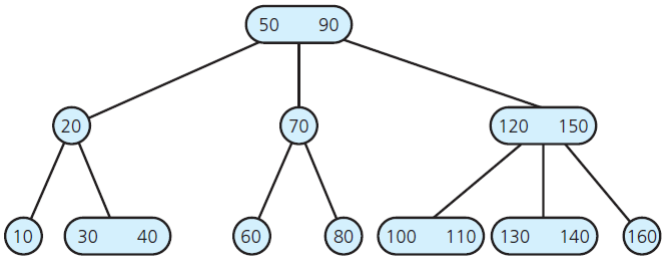
86

2-3 Trees, 2-3-4 Trees

- A 2-3 tree is not a binary tree. Neither is a 2-3-4 tree.
- A 2-3 tree, a 2-3-4 tree are never taller than a minimum-height binary tree.

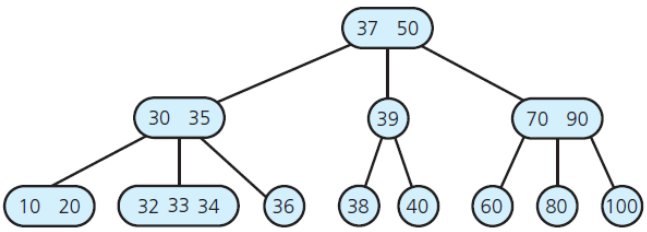
87

2-3 Trees



88

2-3-4 Trees



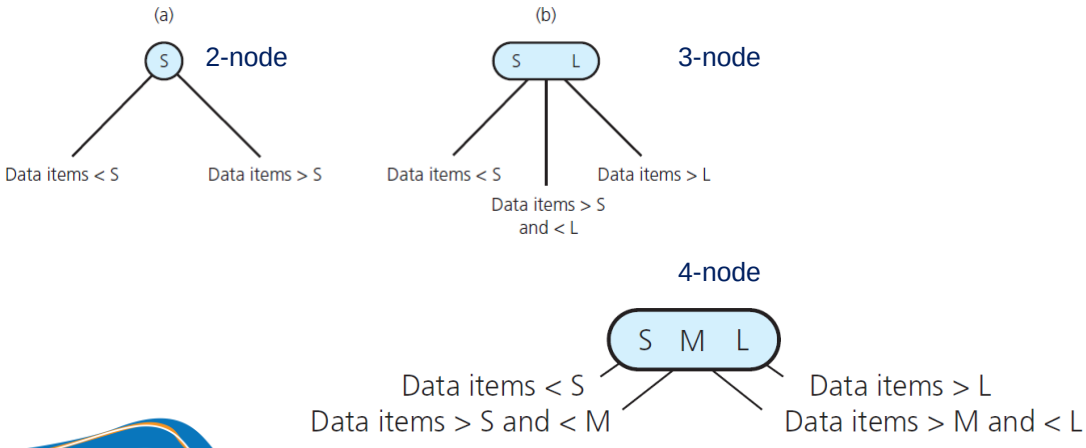
89

2-node, 3-node, 4-node

- A `2-node` (has two children) must contain single data item greater than left child's item(s) and less than right child's item(s).
- A `3-node` (has three children) must contain two data items, `S` and `L`, such that
  - S is greater than left child's item(s) and less than middle child's item(s);
  - L is greater than middle child's item(s) and less than right child's item(s).
- A `4-node` (has our children) must contain three data items `S`, `M`, and `L` that satisfy:
  - S is greater than left child's item(s) and less than middle-left child's item(s)
  - M is greater than middle-left child's item(s) and less than middle-right child's item(s);
  - L is greater than middle-right child's item(s) and less than right child's item(s).

90

2-node, 3-node, 4-node



91

2-3 Trees

- 2–3 trees were invented by John Hopcroft in 1970.
- 2-3 tree is a tree in which
  - Every internal node is either a 2-node or a 3-node.
  - Leaves have no children and may contain either one or two data items.

92

2-3-4 Trees

- 2-3 tree is a tree in which
  - Every internal node is a 2-node, a 3-node or a 4-node.
  - Leaves have no children and may contain either one, two or three data items.

93

Insertion Operation

101

Inserting Data into a 2-3 Tree

```
// Inserts a new item into a 2-3 tree whose items are distinct and differ from the
// new item.
insertItem(23Tree: TwoThreeTree, newItem: ItemType)

   Locate the leaf, leafNode, in which newItem belongs
   Add newItem to leafNode

   if (leafNode has three items)
      split(leafNode)

// Splits node n, which contains two items. Note: If n is
//  not a leaf, it has four children.
split(n: TwoThreeNode)

   if (n is the root)
      Create a new node p
   else
      Let p be the parent of n

   Replace node n with two nodes, n1 and n2, so that p is their parent
   Give n1 the item in n with the smallest value
```


102

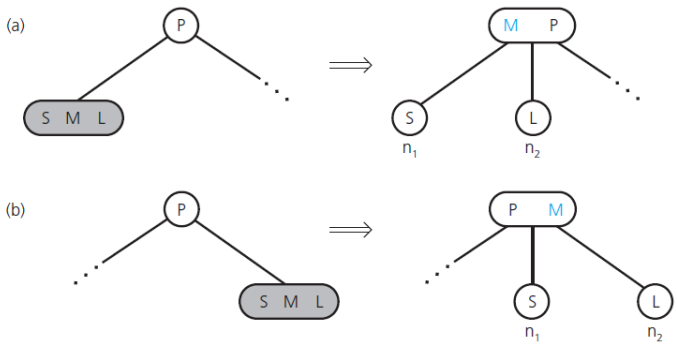
Inserting Data into a 2-3 Tree

```
split(n: TwoThreeNode)

    if (n is the root)
        Create a new node p
    else
        Let p be the parent of n

    Replace node n with two nodes, n1 and n2, so that p is their parent
    Give n1 the item in n with the smallest value
    Give n2 the item in n with the largest value

    if (n is not a leaf)
    {
        n1 becomes the parent of n's two leftmost children
        n2 becomes the parent of n's two rightmost children
    }
    Move the item in n that has the middle value up to p
    if (p now has three items)
        split(p)
```


103

Inserting Data into a 2-3 Tree

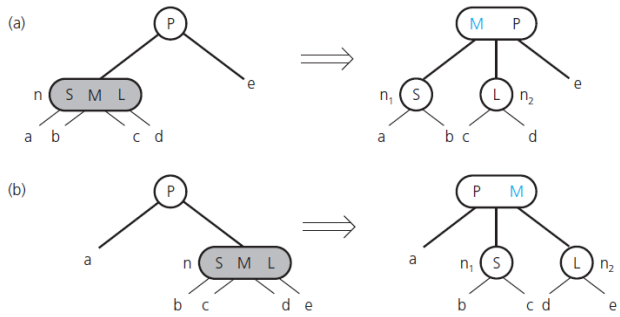
- Splitting a leaf in a 2-3 tree when the leaf is a
  - (a) left child; (b) right child



104

Inserting Data into a 2-3 Tree

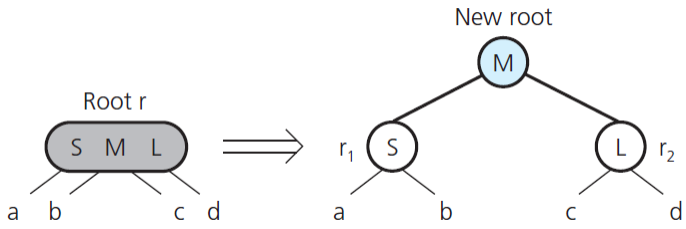
- Splitting an internal node in a 2-3 tree when the node is a (a) left child; (b) right child



105

Inserting Data into a 2-3 Tree

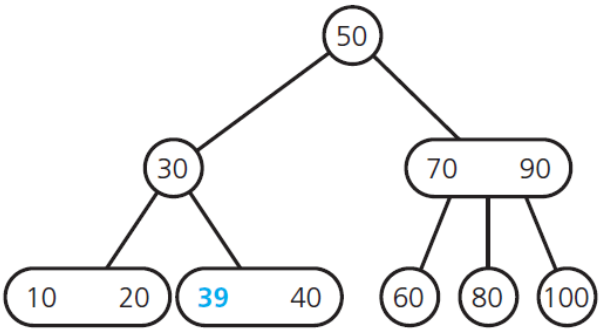
- Splitting the root of a 2-3 tree



106

4.0 Examples

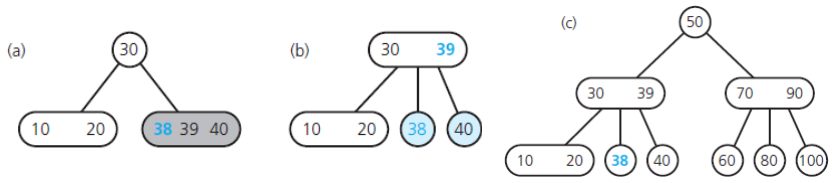
- After inserting 39 into the tree



107

4.0 Examples

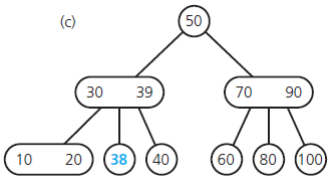
- The steps for inserting 38 into the tree
  - (a) The located node has no room;
  - (b) the node splits;
  - (c) the resulting tree



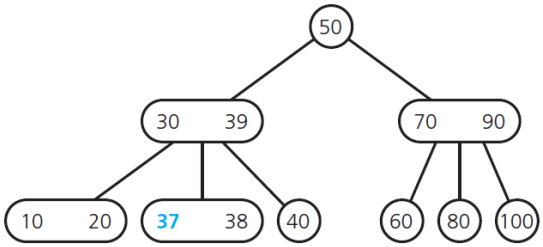
108

4.0 Examples

- After inserting 37 into the tree



Original tree

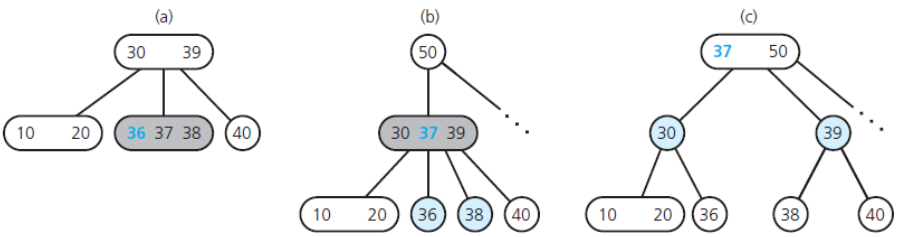


Resulting tree

109

4.0 Examples

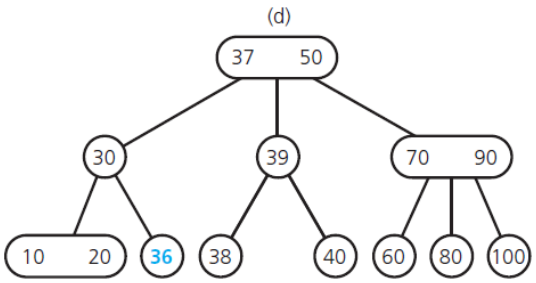
- The steps for inserting 36 into the tree



110

4.0 Examples

- the resulting tree



111

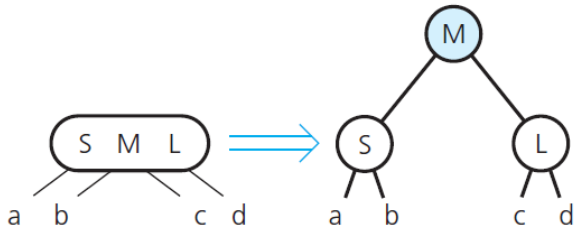
4.0 Inserting Data into a 2-3-4 Tree

- Insertion algorithm splits a node by moving one of its items up to its parent node
- Splits 4-nodes as soon as it encounters them on the way down the tree from the root to a leaf

112

Inserting Data into a 2-3-4 Tree

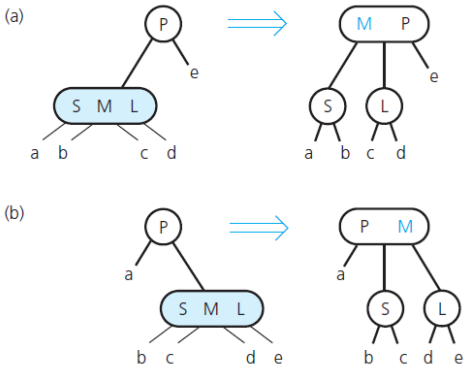
- Splitting a 4-node root during insertion into a 2-3-4 tree



113

Inserting Data into a 2-3-4 Tree

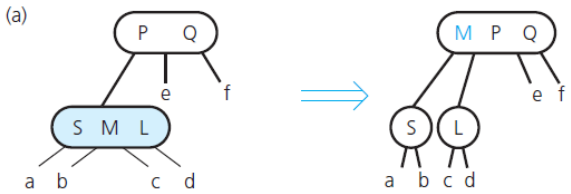
- Splitting a 4-node whose parent is a 2-node during insertion into a 2-3-4 tree, when the 4-node is a (a) left child; (b) right child



114

Inserting Data into a 2-3-4 Tree

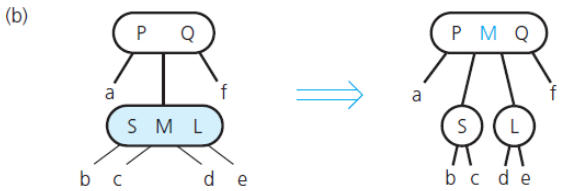
- Splitting a 4-node whose parent is a 3-node during insertion into a 2-3-4 tree, when the 4-node is a (a) left child



115

Inserting Data into a 2-3-4 Tree

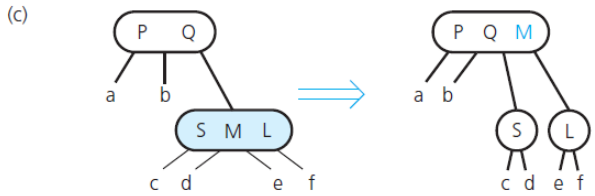
- Splitting a 4-node whose parent is a 3-node during insertion into a 2-3-4 tree, when the 4-node is a (b) middle child



116

Inserting Data into a 2-3-4 Tree

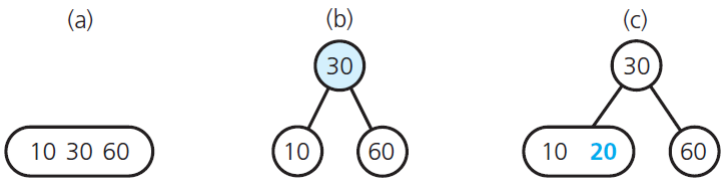
- Splitting a 4-node whose parent is a 3-node during insertion into a 2-3-4 tree, when the 4-node is a (c) right child



117

Examples

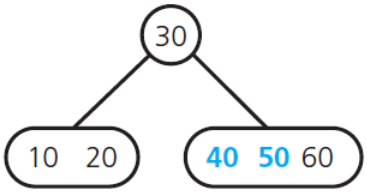
- Inserting 20 into a 2-3-4 tree
  - (a) the original tree;
  - (b) after splitting the node;
  - (c) after inserting 20



118

4.0 Examples

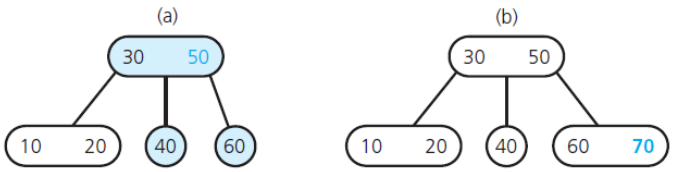
- After inserting 50 and 40 into the tree



119

4.0 Examples

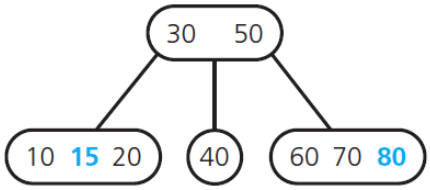
- The steps for inserting 70 into the tree
  (a) after splitting the 4-node;
  (b) after inserting 70



120

4.0 Examples

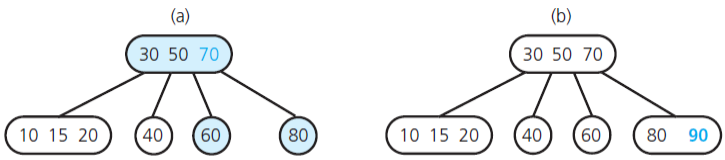
- After inserting 80 and 15 into the tree



121

4.0 Examples

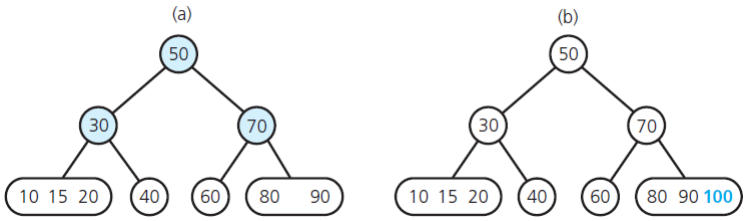
- The steps for inserting 90 into the tree



122

Examples

- The steps for inserting 100 into the tree



123

Removal Operation

125

4.0 Removing Data from a 2-3 Tree

```
// Removes the given data item from a 2-3 tree. Returns true if successful
// or false if no such item exists.
removeItem(23Tree: TwoThreeTree, dataItem: ItemType): boolean

   Attempt to locate dataItem
   if (dataItem is found)
   {
      if (dataItem is not in a leaf)
         Swap dataItem with its inorder successor, which will be in a leaf leafNode

      // The removal always begins at a leaf
      Remove dataItem from leaf leafNode
      if (leafNode now has no items)
         fixTree(leafNode)
      return true
   }
   else
      return false
```


126

4.0 Removing Data from a 2-3 Tree

```
   else
      return false

// Completes the removal when node n is empty by either deleting the root,
// redistributing values, or merging nodes. Note: If n is internal, it has one child.
fixTree(n: TwoThreeNode)

   if (n is the root)
      Delete the root
   else
   {
      Let p be the parent of n
      if (some sibling of n has two items)
      {
         Distribute items appropriately among n, the sibling, and p
```


127

Removing Data from a 2-3 Tree

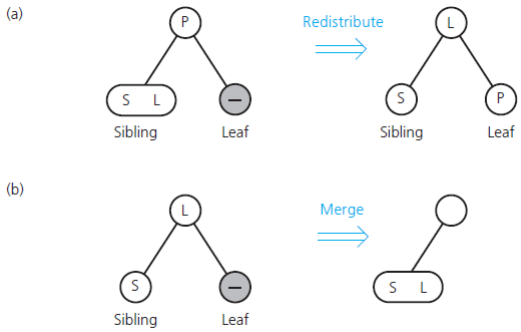
```
        if (n is internal)
            Move the appropriate child from sibling to n
    }
    else // Merge the node
    {
        Choose an adjacent sibling s of n
        Bring the appropriate item down from p into s
        if (n is internal)
            Move n's child to s
        Remove node n
        if (p is now empty)
            fixTree(p)
    }
}
```


128

Removing Data from a 2-3 Tree

(a) Redistributing values;

(b) merging a leaf:

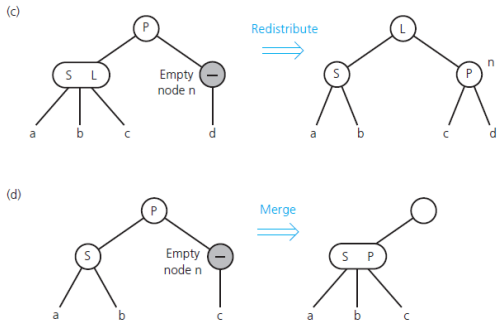


129

Removing Data from a 2-3 Tree

(c) redistributing values and children

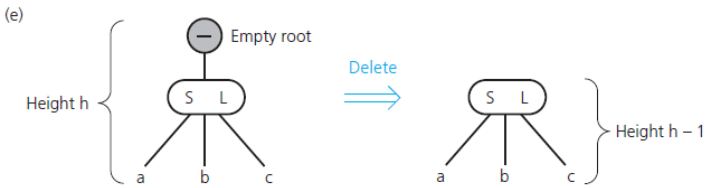
(d) merging internal nodes



130

Removing Data from a 2-3 Tree

(e) deleting the root

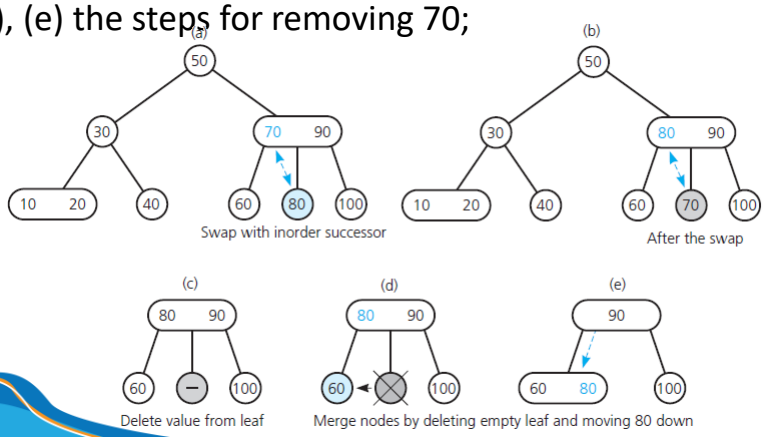


131

4.0 Examples

(a) A 2-3 tree;

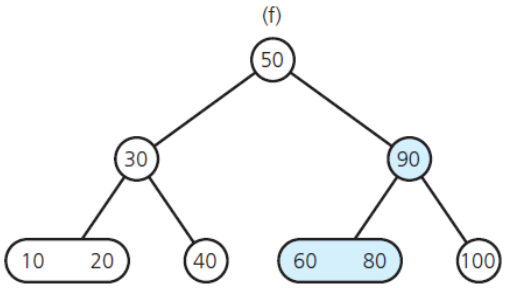
(b), (c), (d), (e) the steps for removing 70;



132

4.0 Examples

- the resulting tree

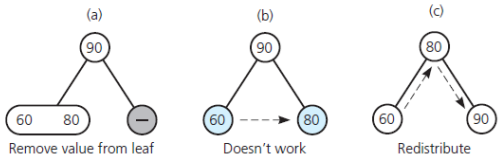


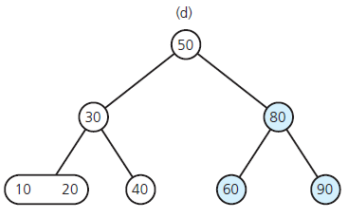
133

4.0 Examples

(a), (b), (c) The steps for removing 100 from the tree;

(d) the resulting tree

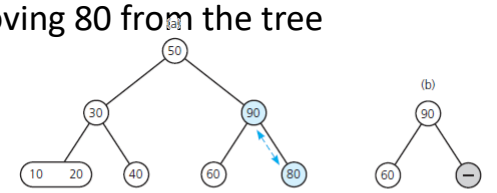


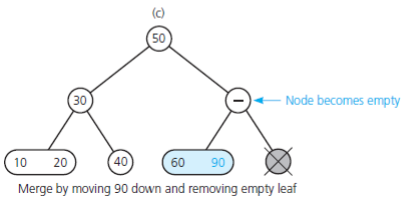


134

4.0 Examples

The steps for removing 80 from the tree

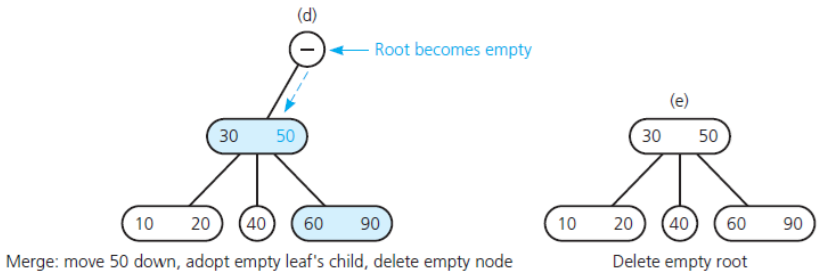




135

4.0 Examples

The steps for removing 80 from the tree



136

4.0 Removing Data from a 2-3-4 Tree

- Removal algorithm has same beginning as removal algorithm for a 2-3 tree
- Locate the node `n` that contains the item `I` you want to remove.
- Find `I` 's in-order successor and swap it with `I` so that the removal will always be at a leaf.
- If leaf is either a 3-node or a 4-node, remove `I` .

140

Questions and Answers