Lecture 3: Balanced Search Trees

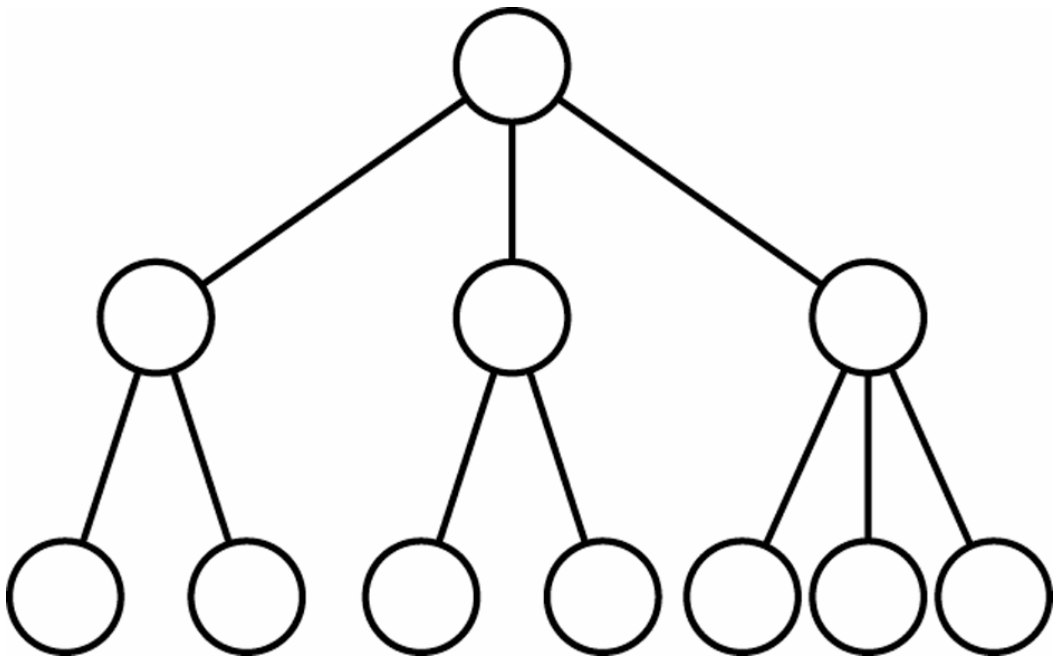
Chúng ta có các loại cây cân bằng sau:

- AVL Trees
- Red-Black Trees
- 2-3 Trees
- 2-3-4 Trees
- B Trees
- B+ Trees

2-3 Trees

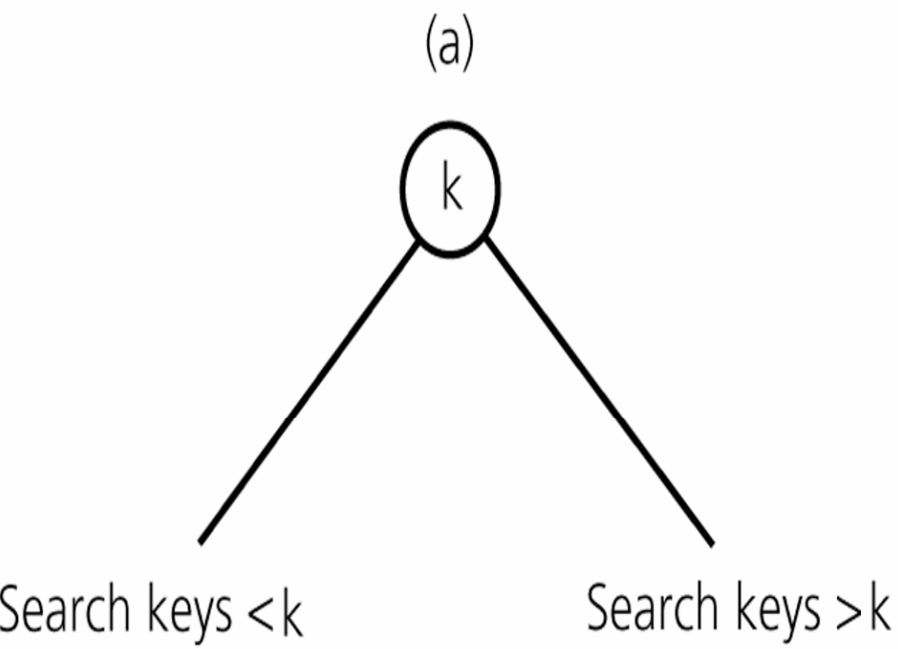
Tính chất

- Mỗi nút trong có từ 2 đến 3 nút con
- Tất cả các nút lá ở cùng một mức

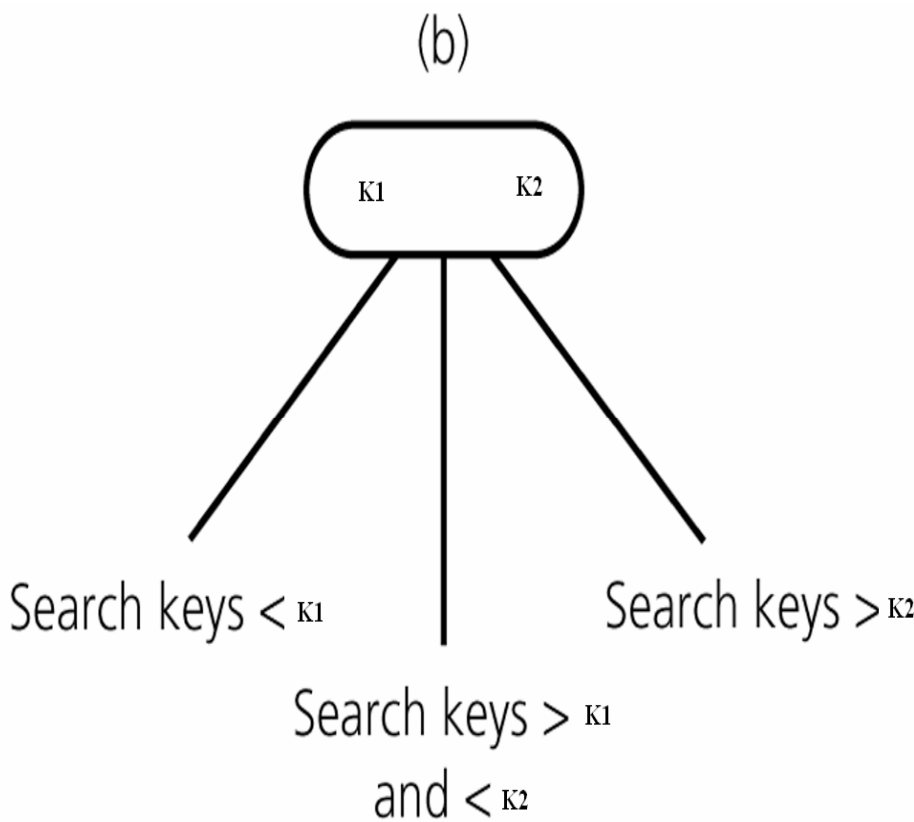


2-3 Trees

2-node

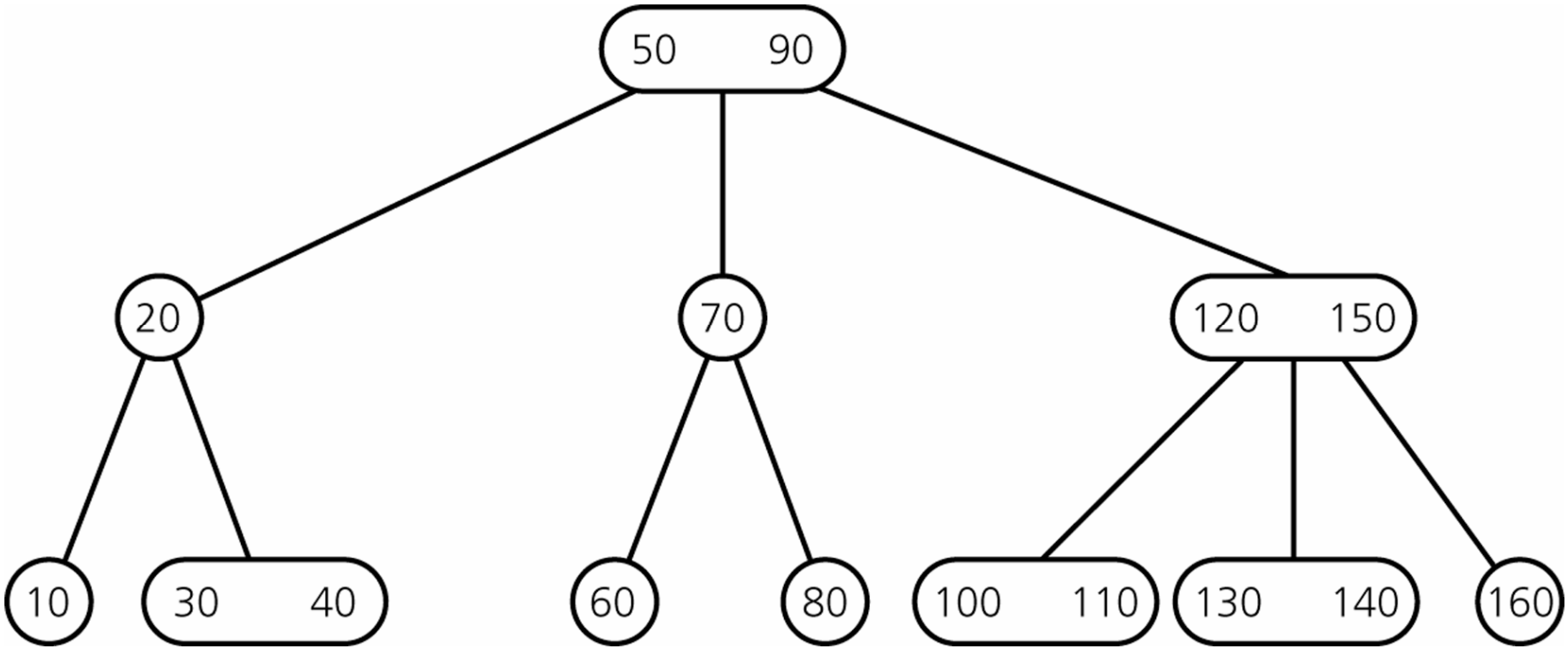


3-node



- Mỗi nút có thể có từ 1 đến 2 giá trị khóa

Example of 2-3 Node



Traversing a 2-3 Node

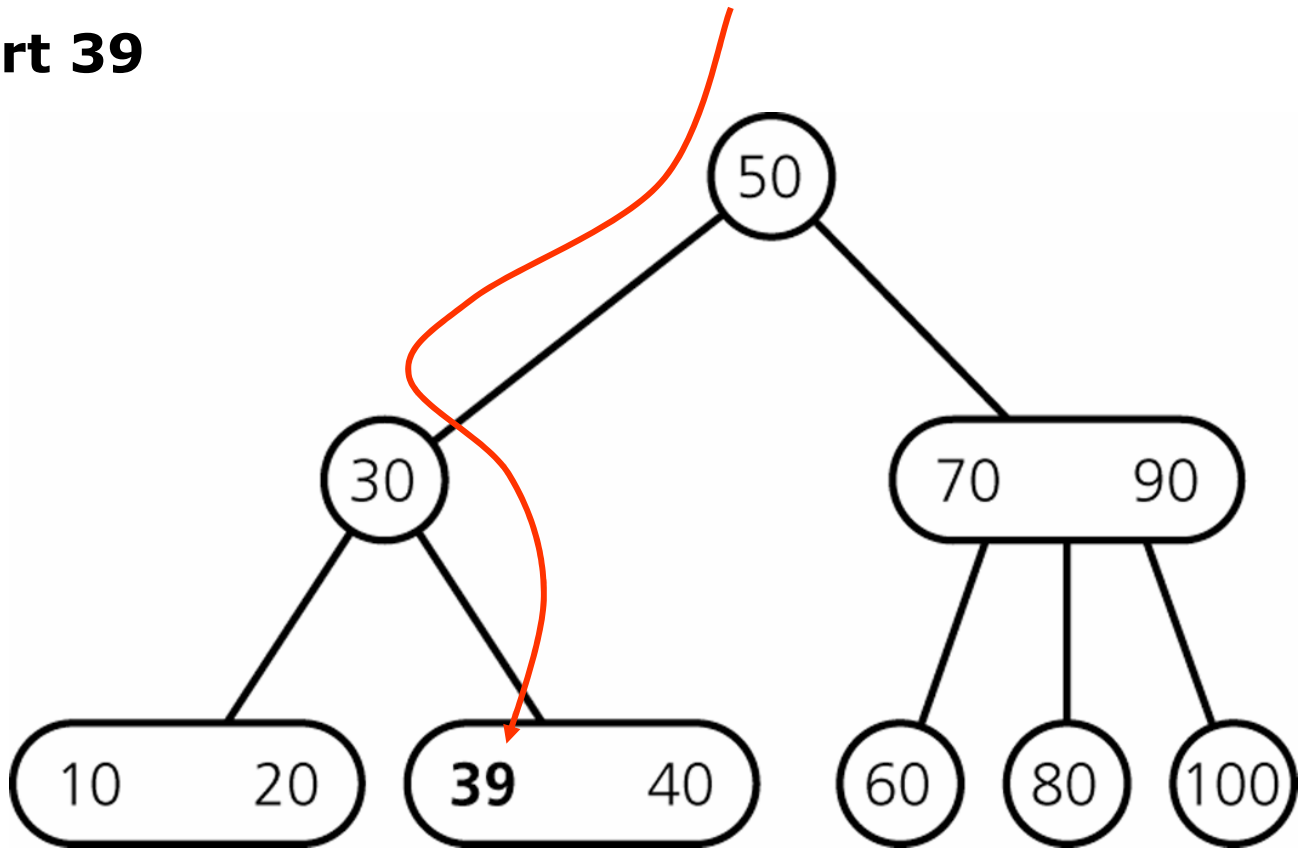
```
inorder(in ttTree: TwoThreeTree)
    if(ttTree's root node r is a leaf)
        visit the data item(s)
    else if(r has two data items)
    {
        inorder(left subtree of ttTree's root)
        visit the first data item
        inorder(middle subtree of ttTree's root)
        visit the second data item
        inorder(right subtree of ttTree's root)
    }
    else
    {
        inorder(left subtree of ttTree's root)
        visit the data item
        inorder(right subtree of ttTree's root)
    }
```

Searching a 2-3 Node

```
retrieveItem(in ttTree: TwoThreeTree,
             in searchKey:KeyType,
             out treeItem:TreeItemType):boolean
    if(searchKey is in ttTree's root node r)
    {
        treeItem = the data portion of r
        return true
    }
    else if(r is a leaf)
        return false
    else
    {
        return retrieveItem(  appropriate subtree,
                              searchKey, treeItem)
    }
```

Inserting Items

Insert 39



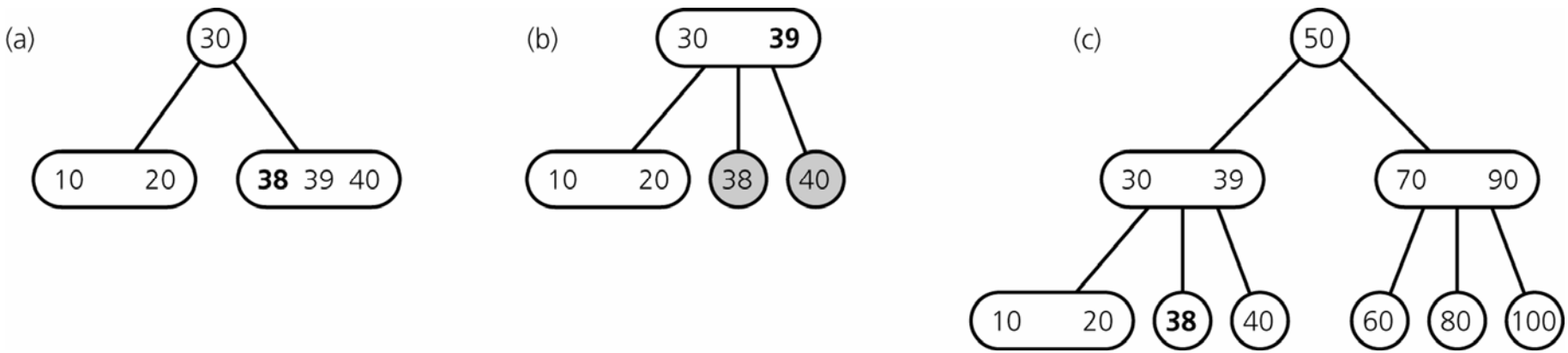
Inserting Items

Insert 38

insert in leaf

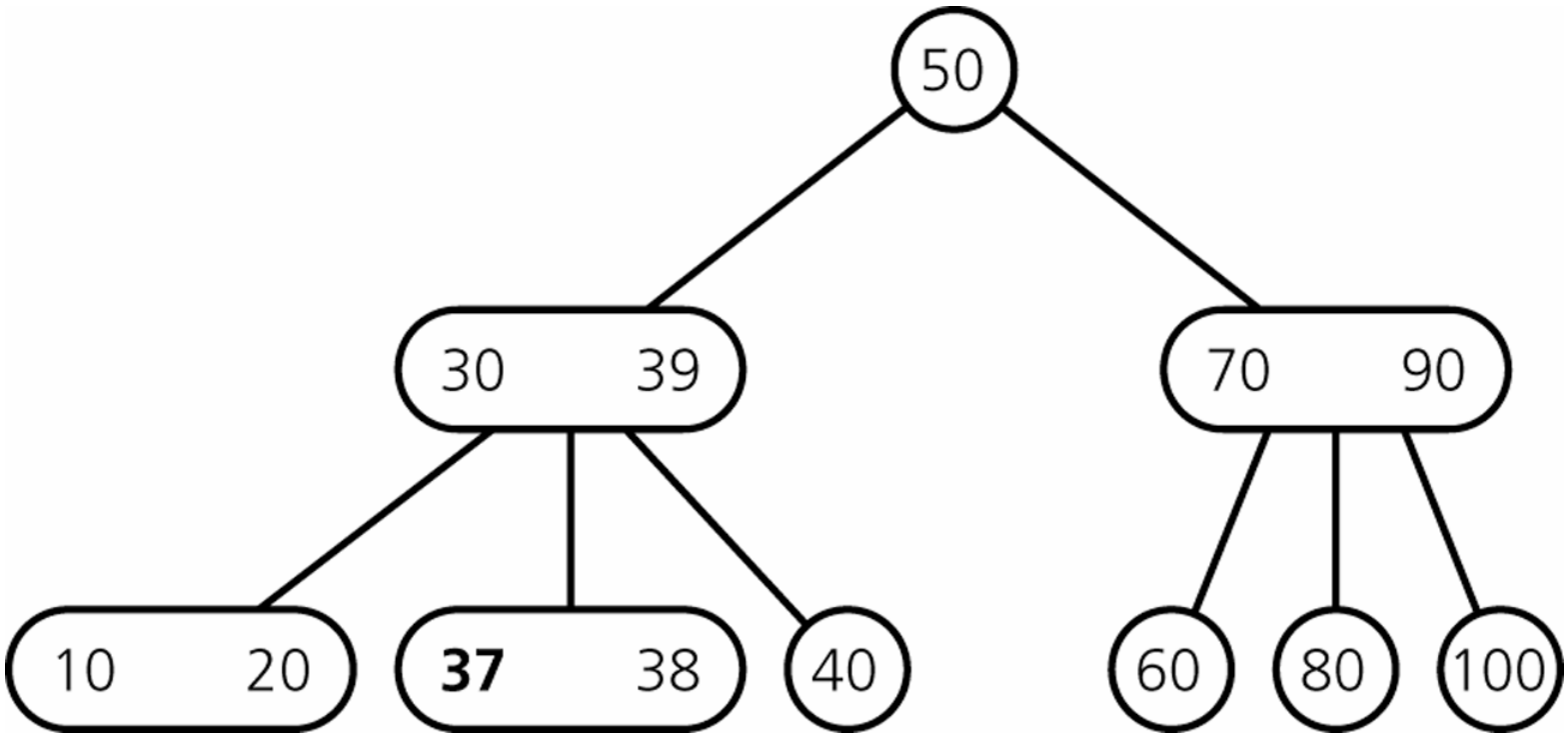
divide leaf
and move middle
value up to parent

result



Inserting Items

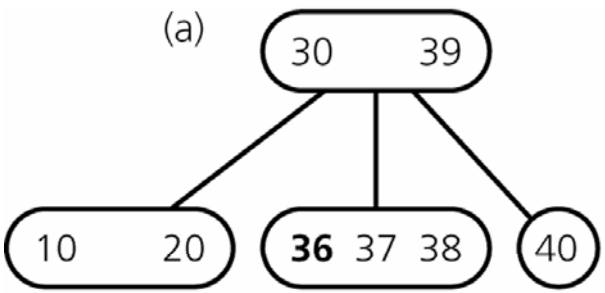
Insert 37



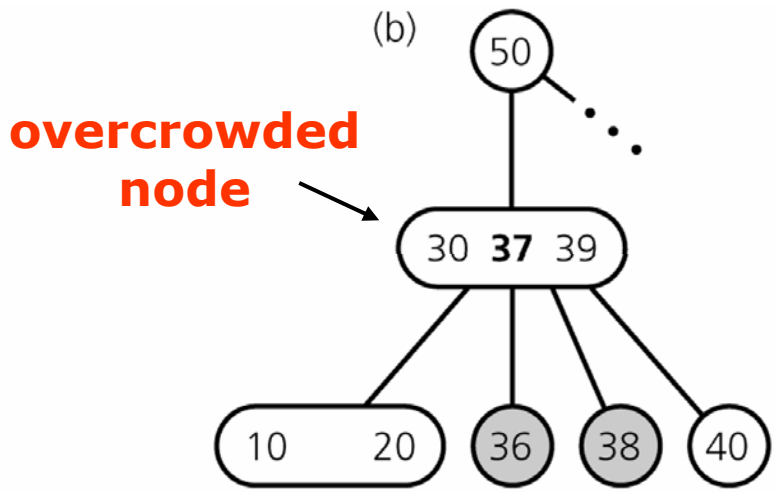
Inserting Items

Insert 36

insert in leaf



divide leaf
and move middle
value up to parent

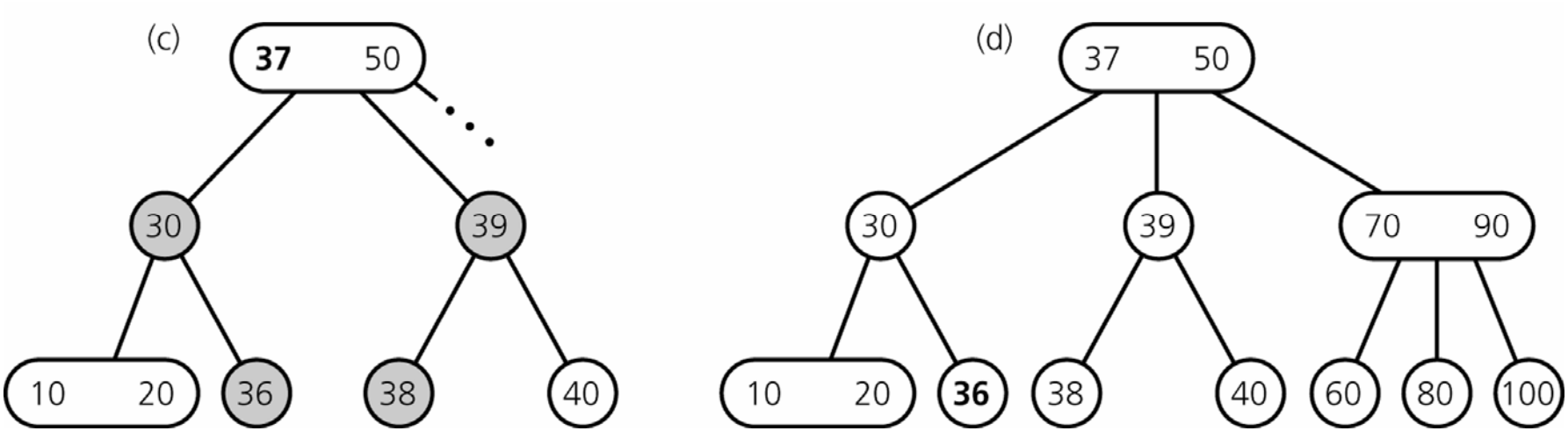


Inserting Items

... still inserting 36

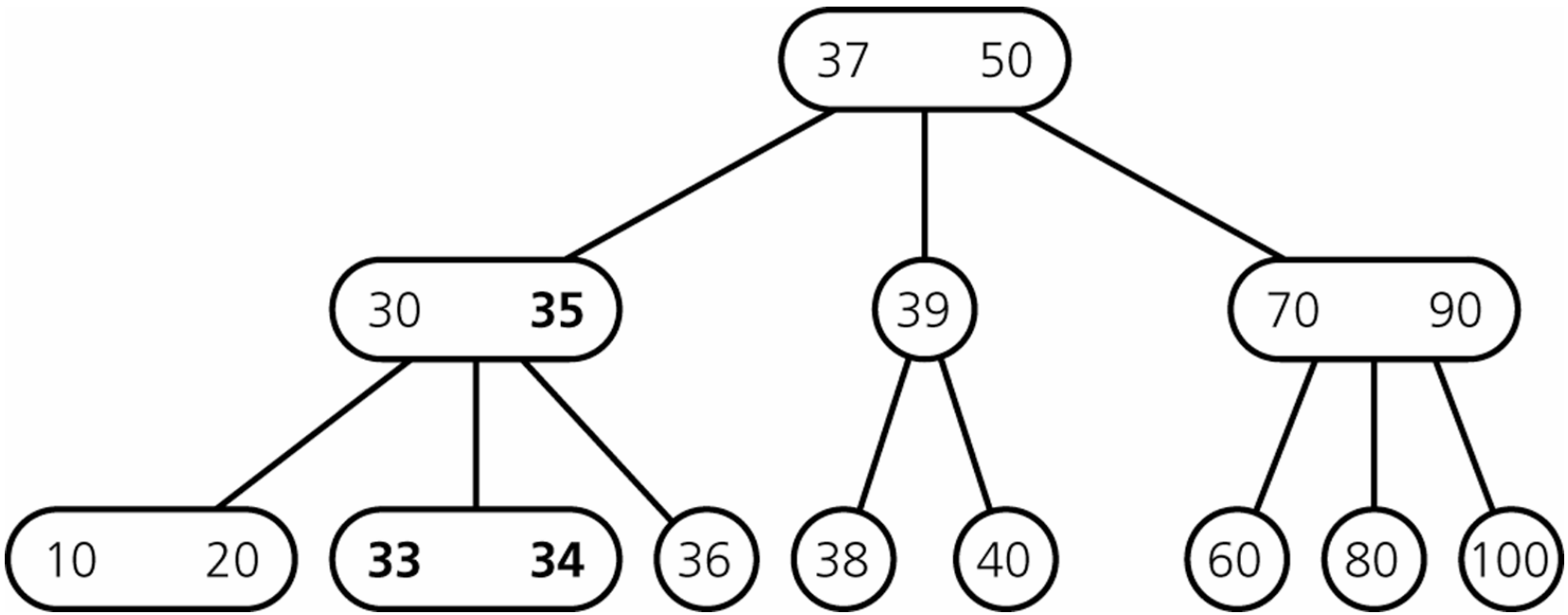
divide overcrowded node,
move middle value up to parent,
attach children to smallest and largest

result

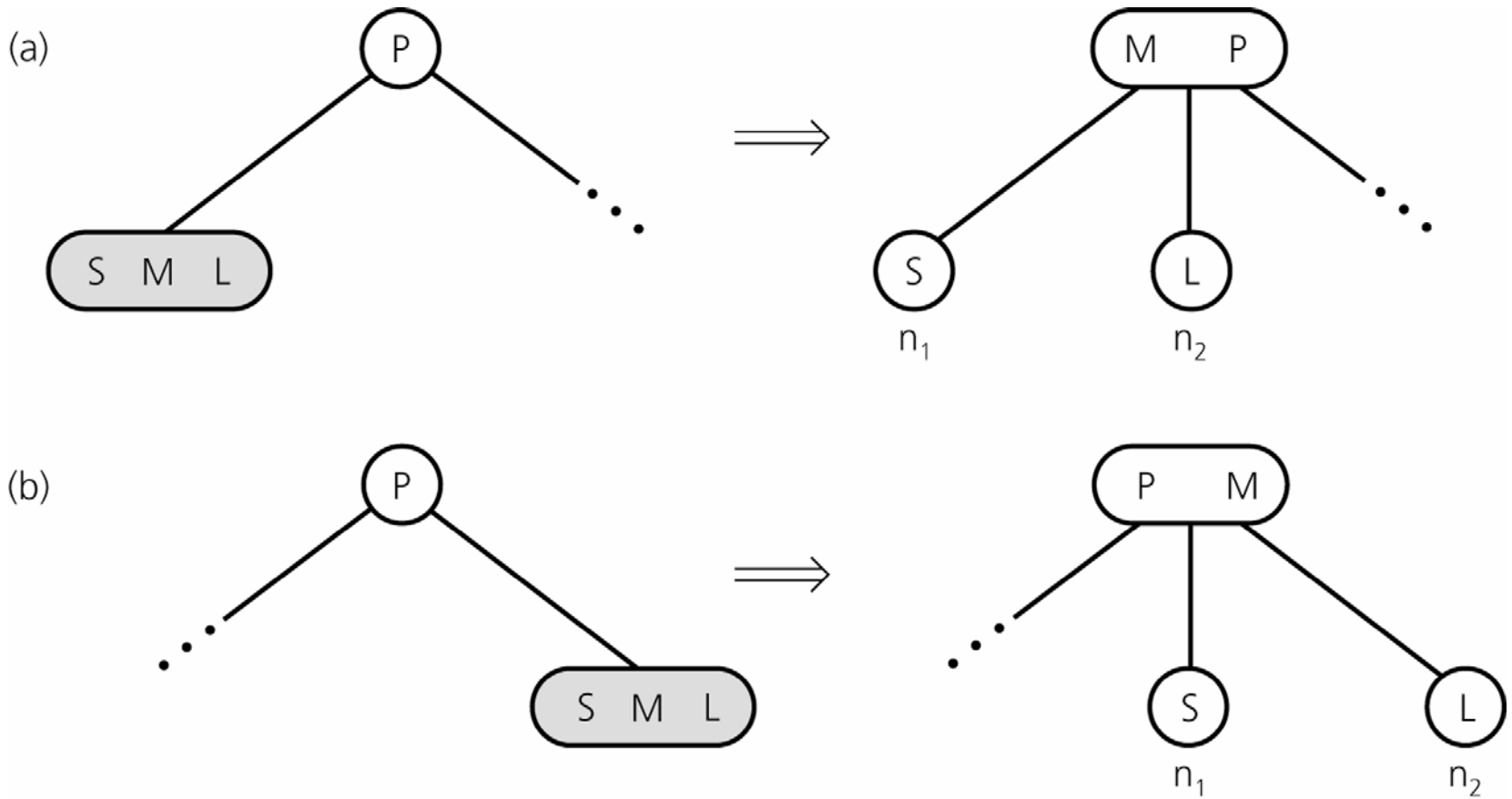


Inserting Items

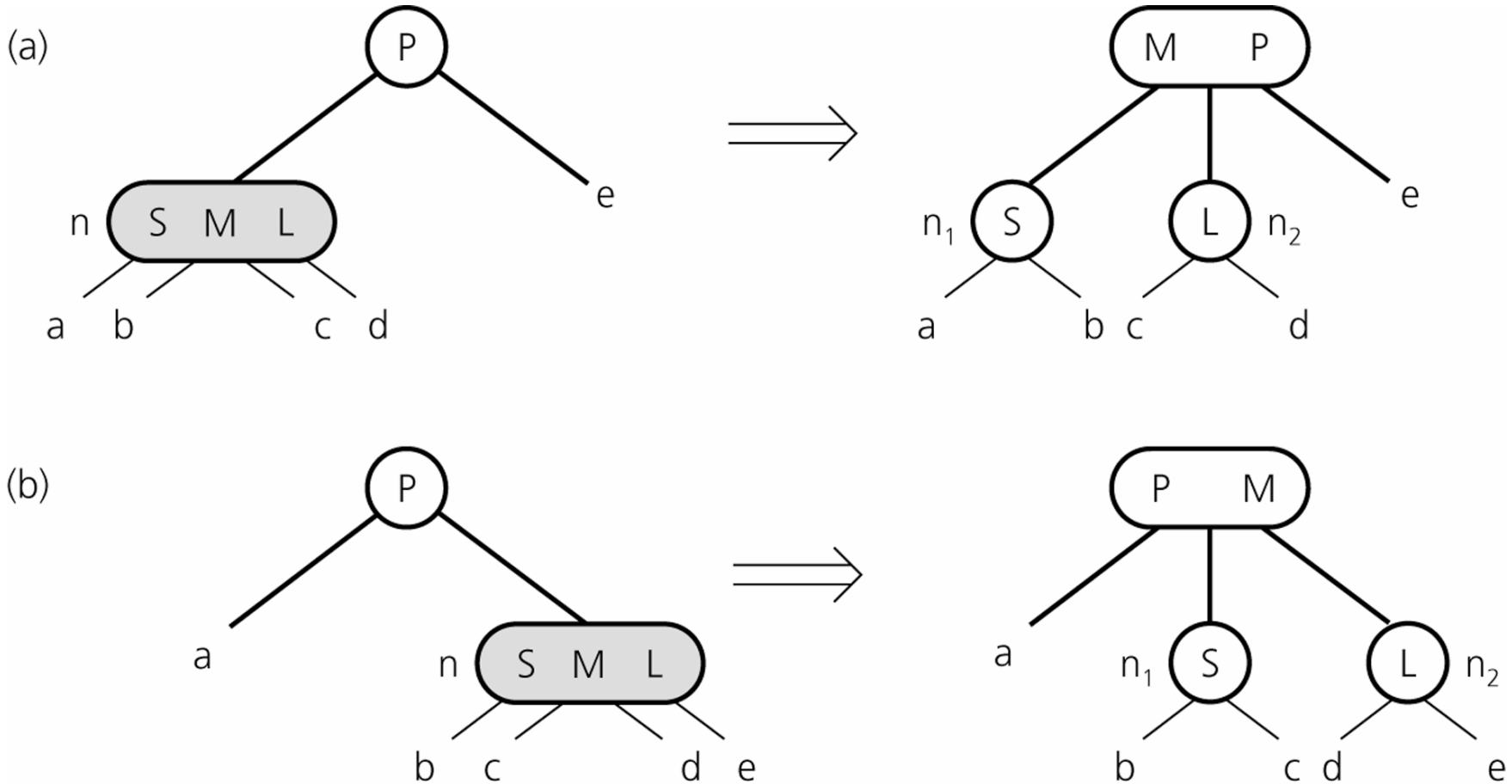
After Insertion of 35, 34, 33



Các trường hợp của phép thêm khóa

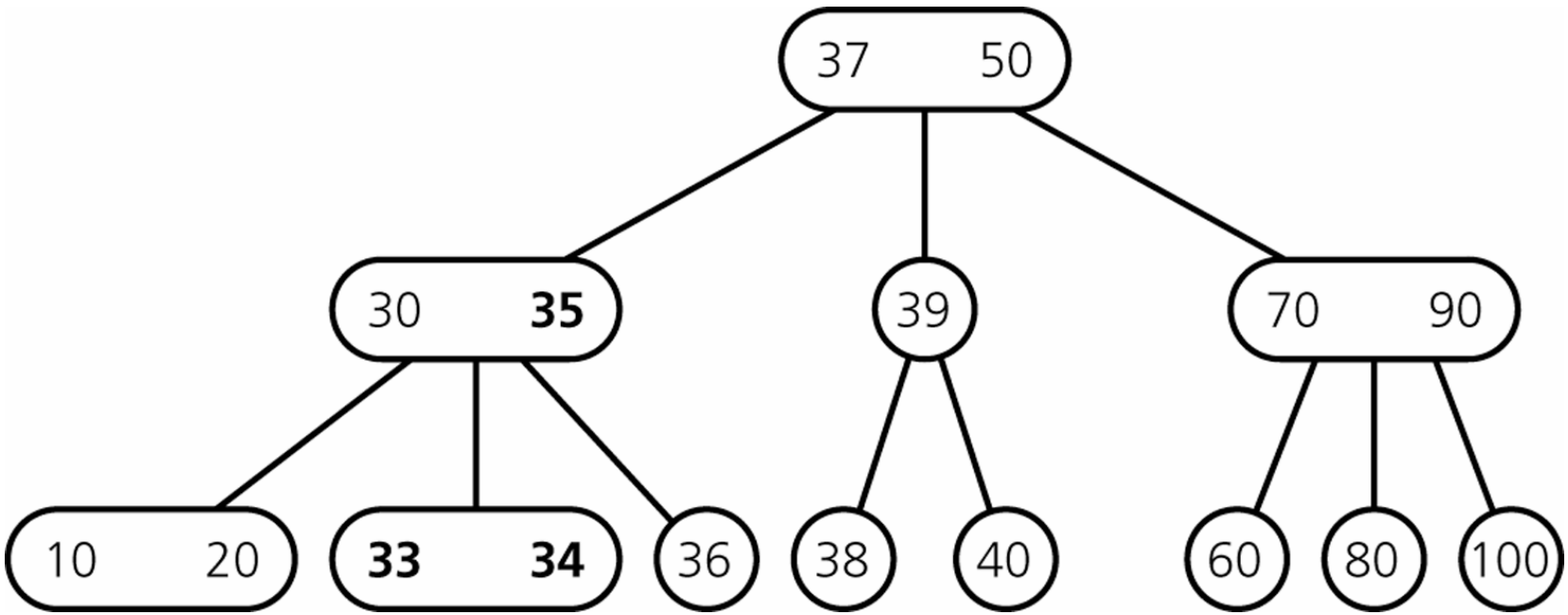


Các trường hợp của phép thêm khóa



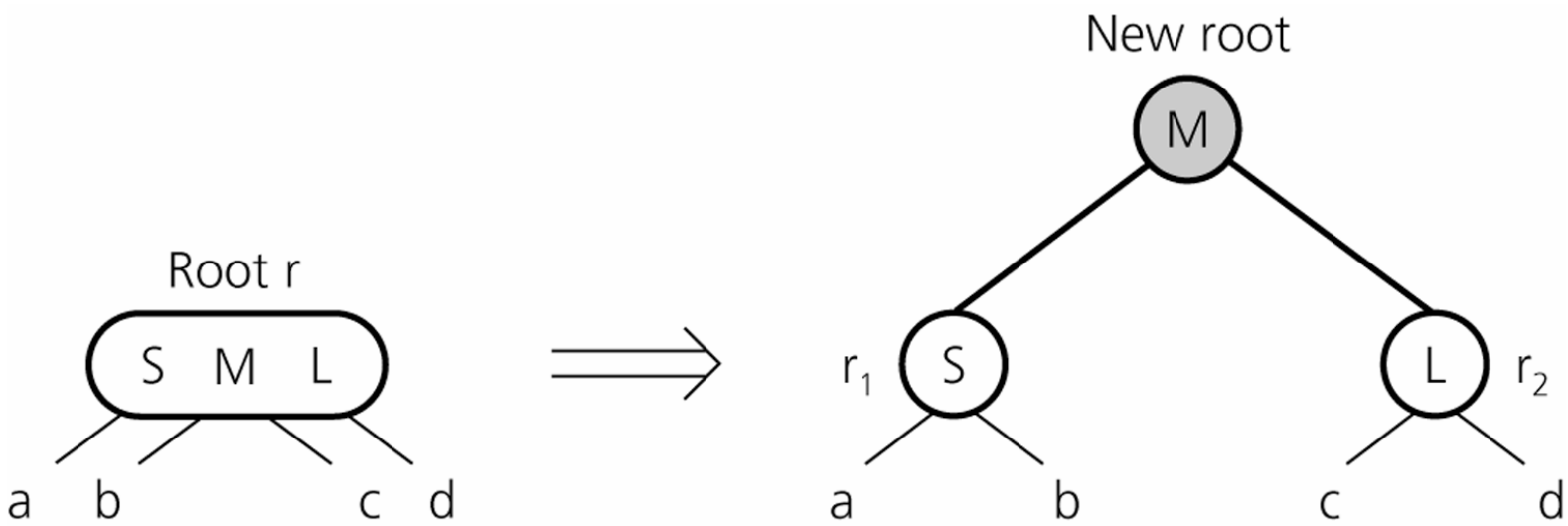
Inserting Items

How do we insert 32?



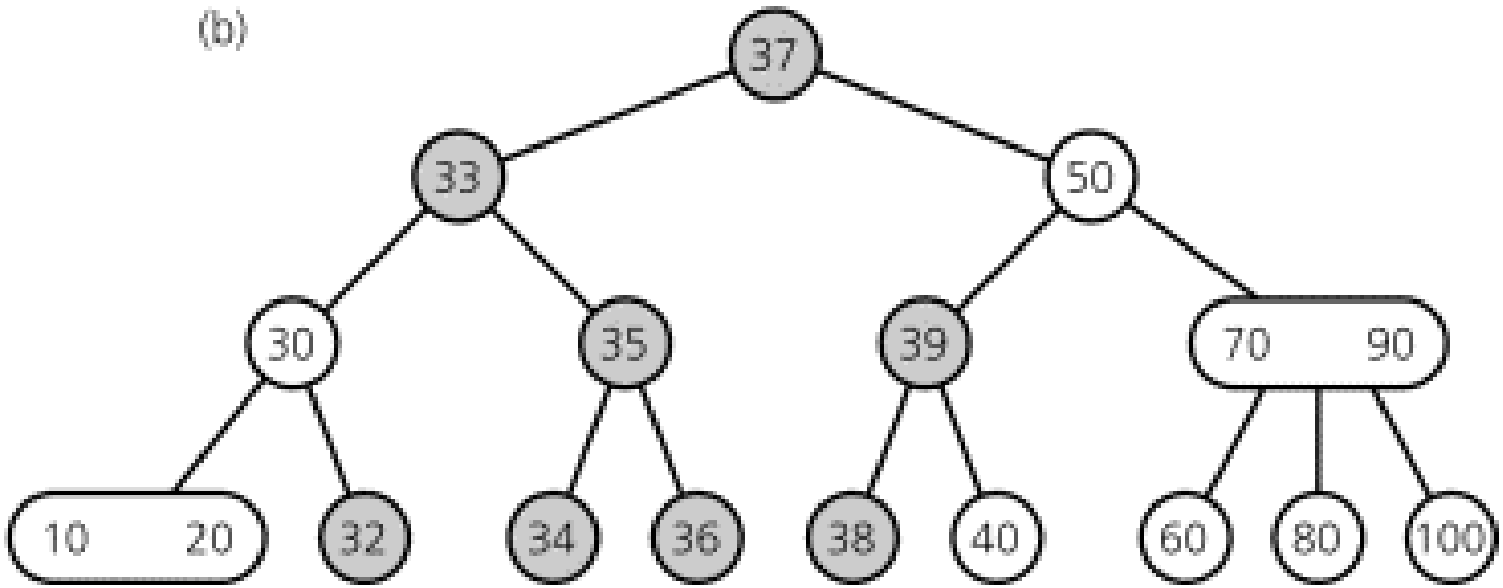
Inserting Items

→ **creating a new root if necessary**
→ **tree grows at the root**



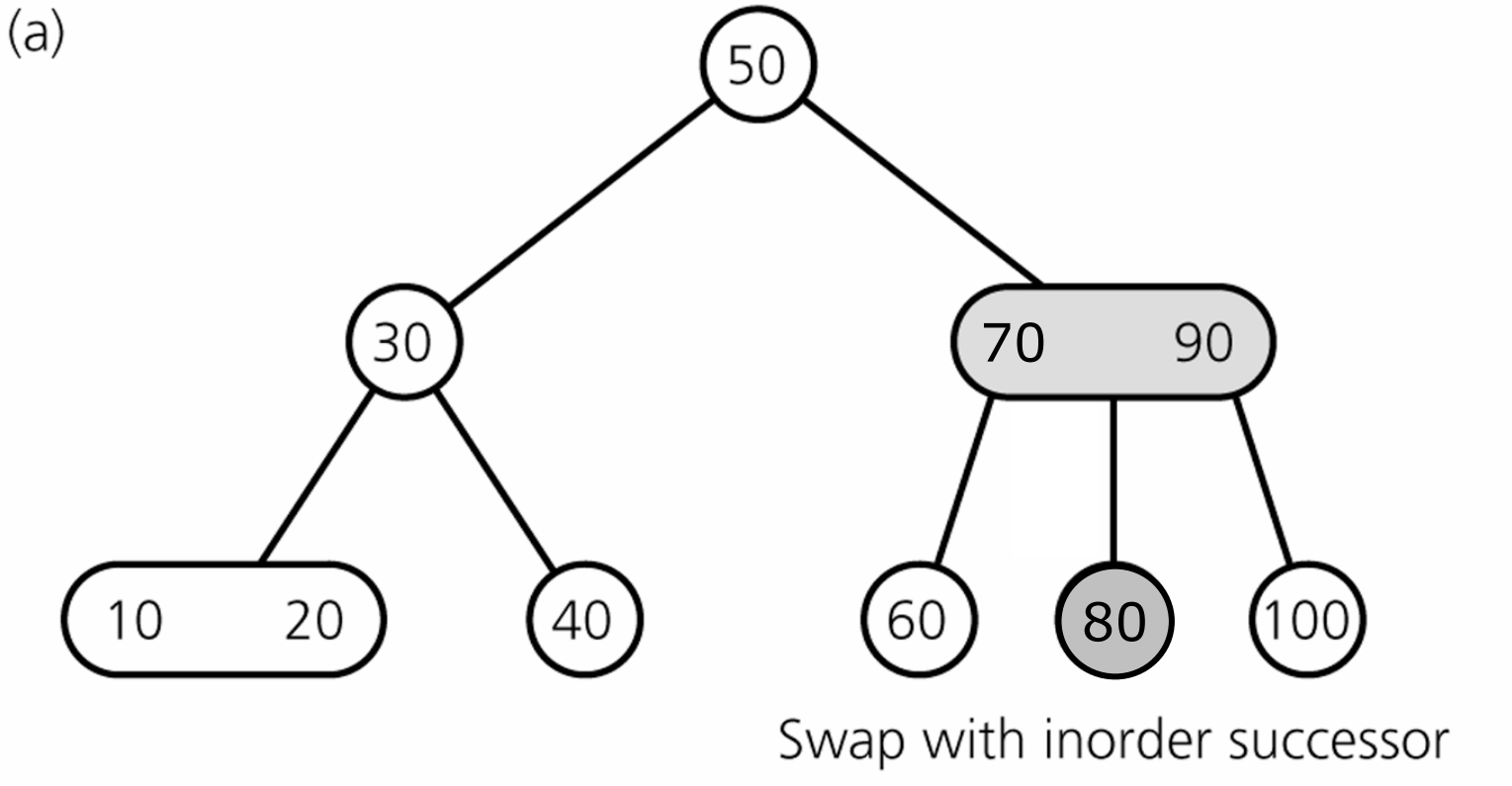
Inserting Items

Final Result



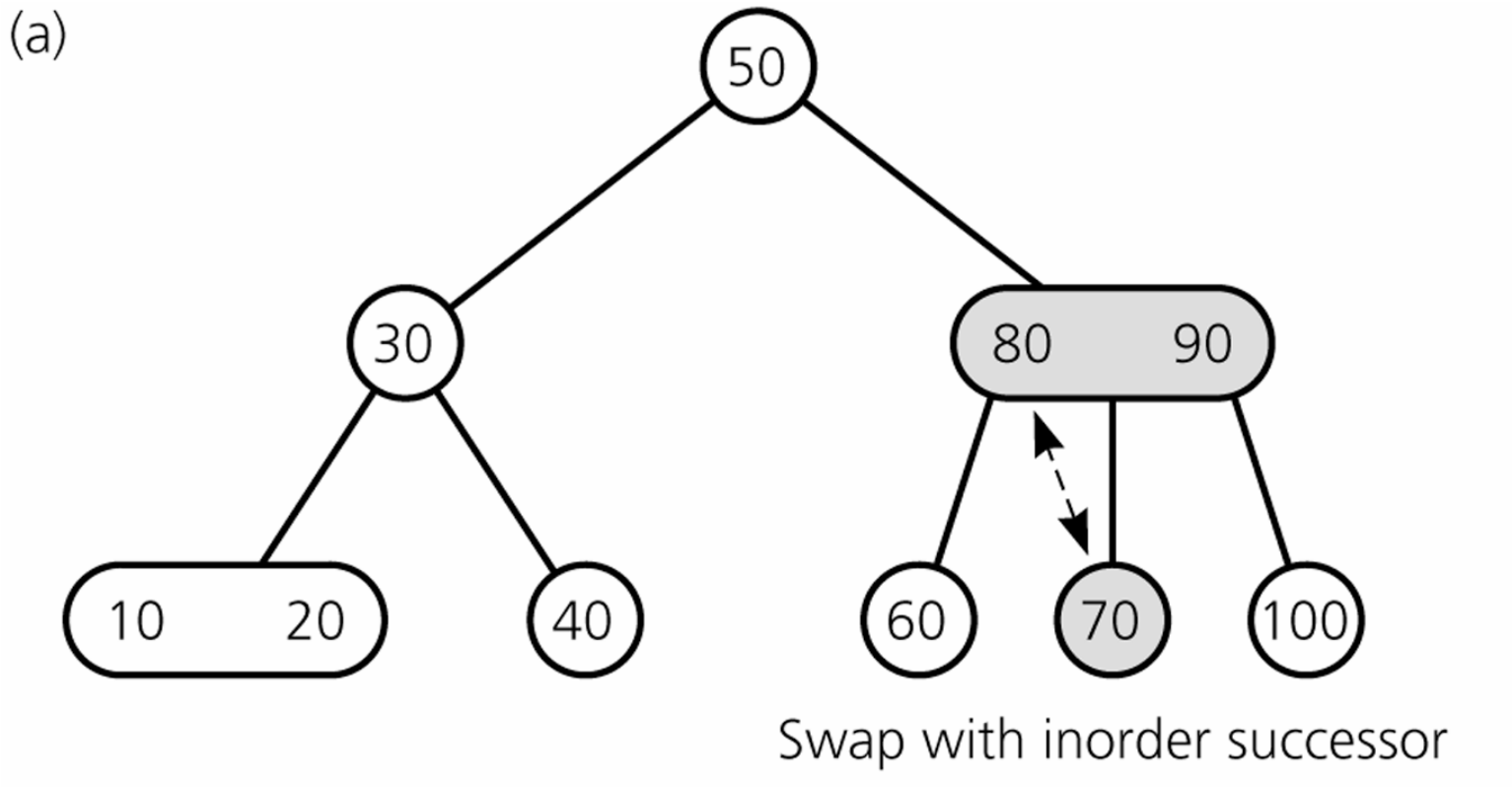
Deleting Items

Delete 70



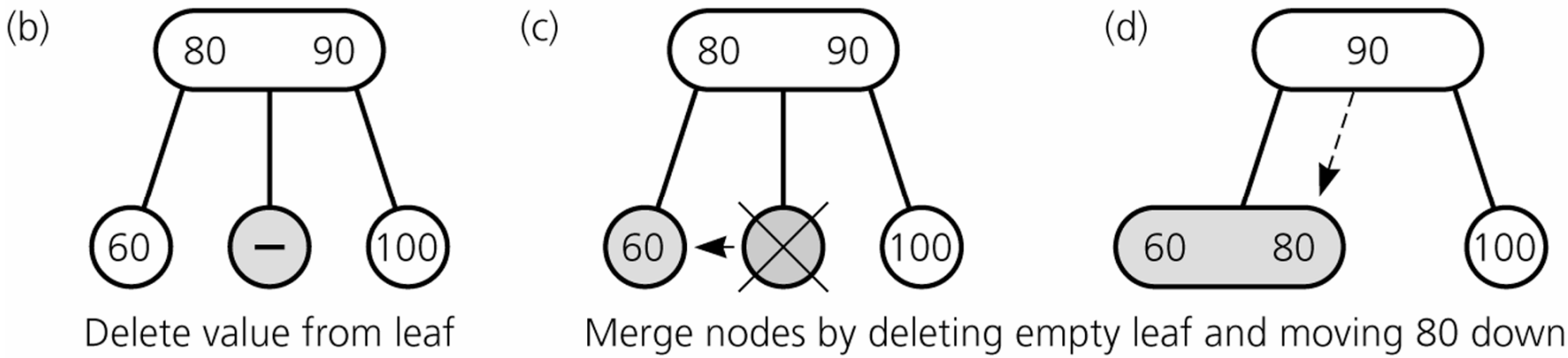
Deleting Items

Deleting 70: swap 70 with inorder successor (80)



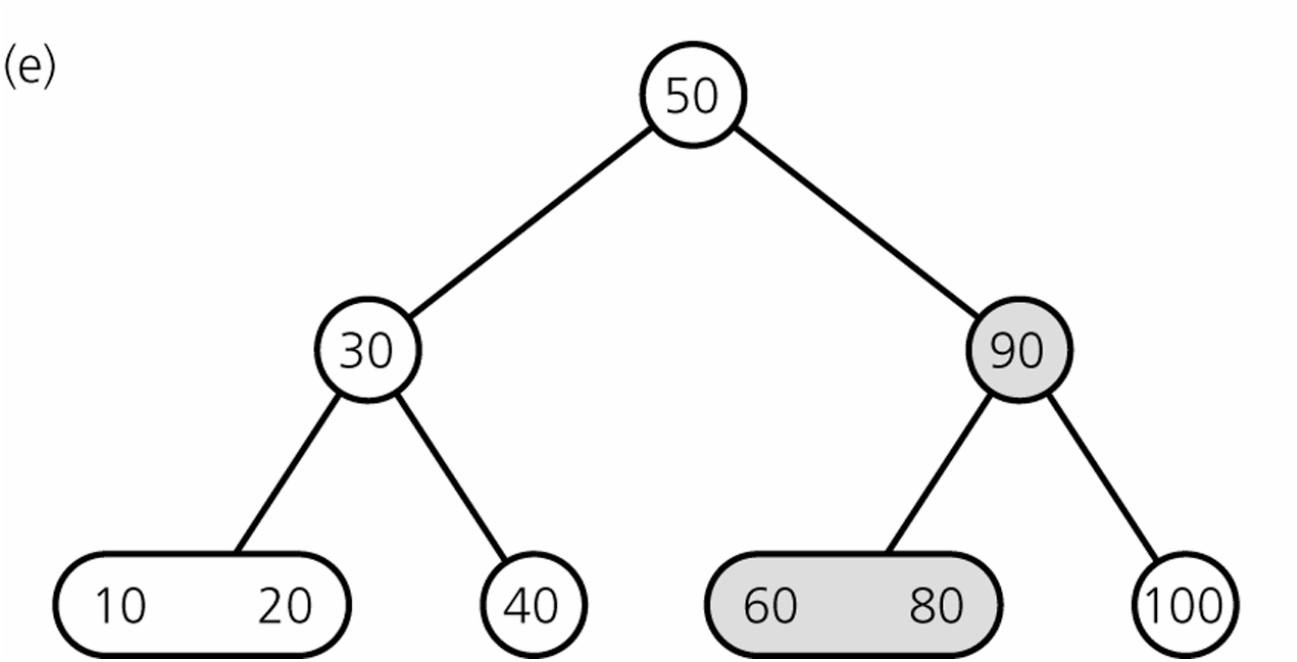
Deleting Items

Deleting 70: … get rid of 70



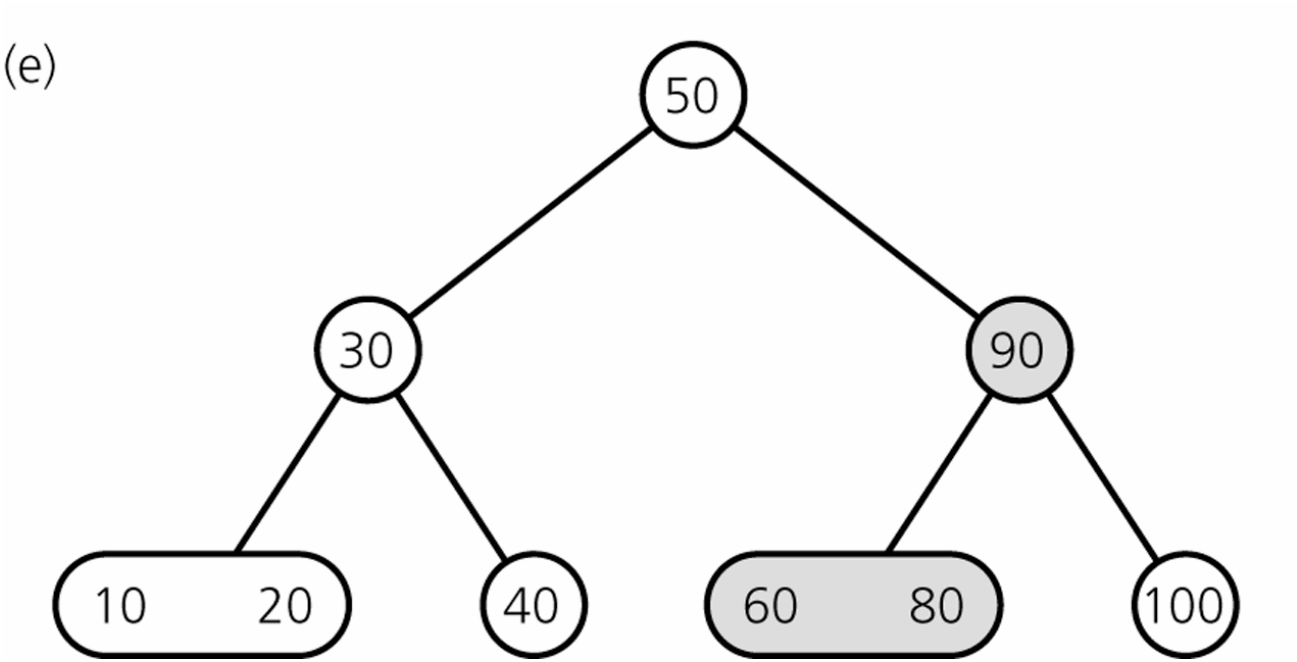
Deleting Items

Result



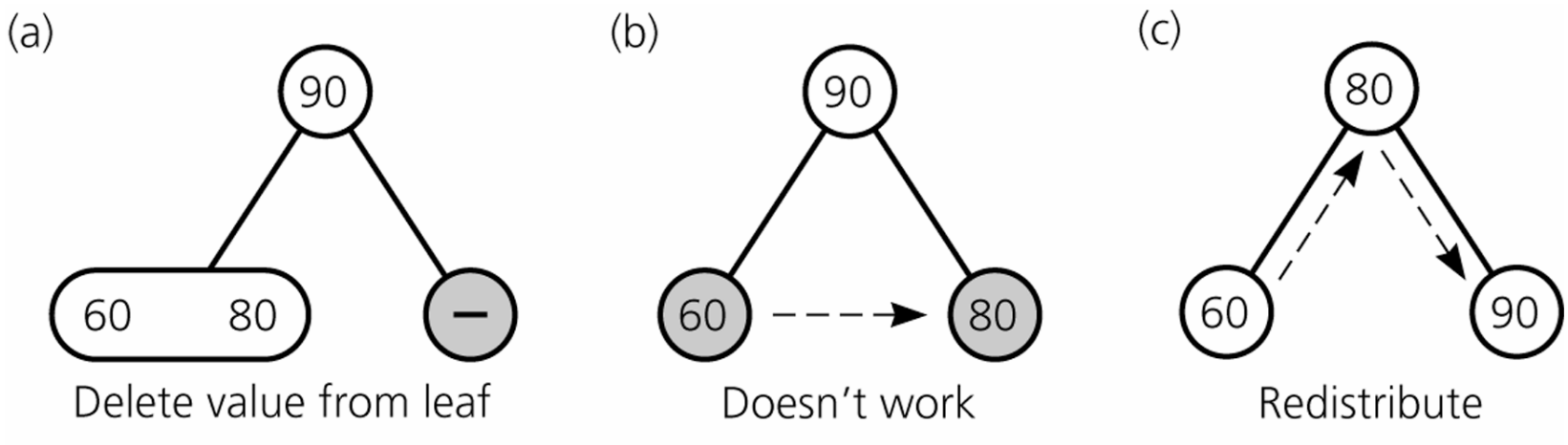
Deleting Items

Delete 100



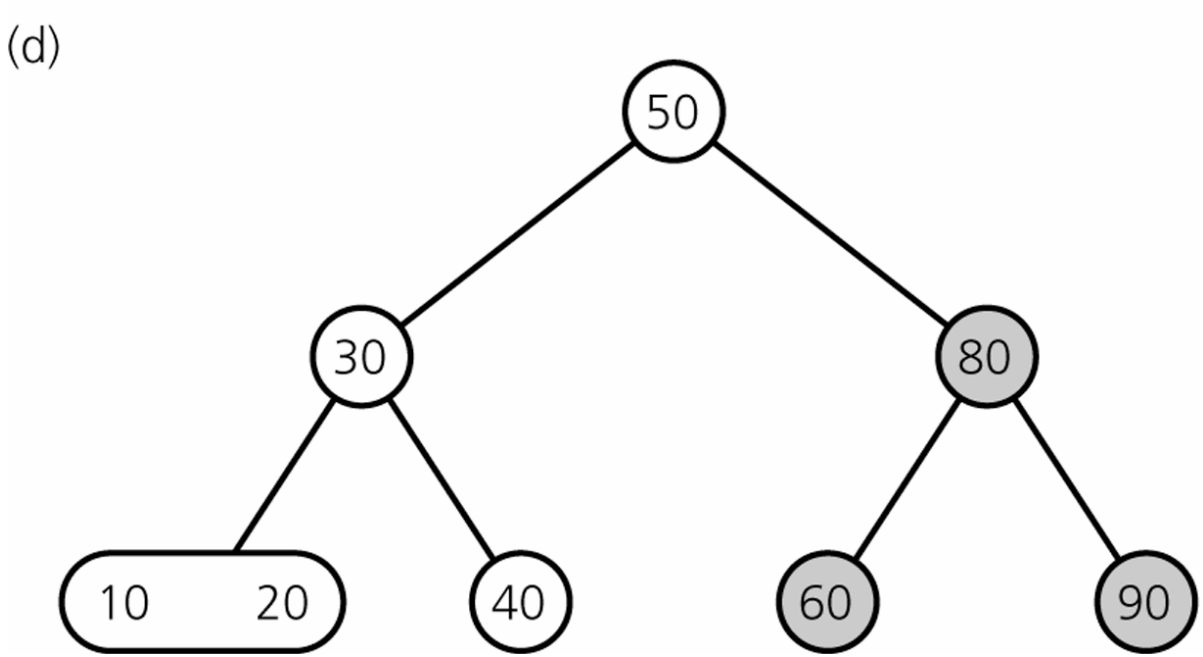
Deleting Items

Deleting 100



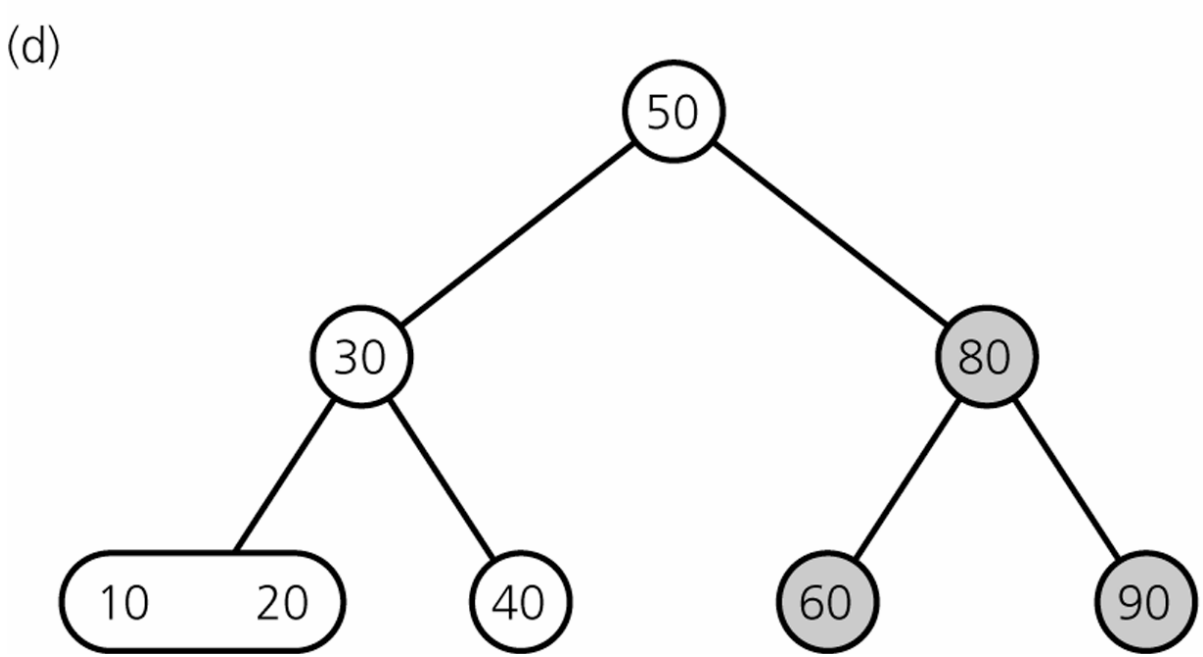
Deleting Items

Result



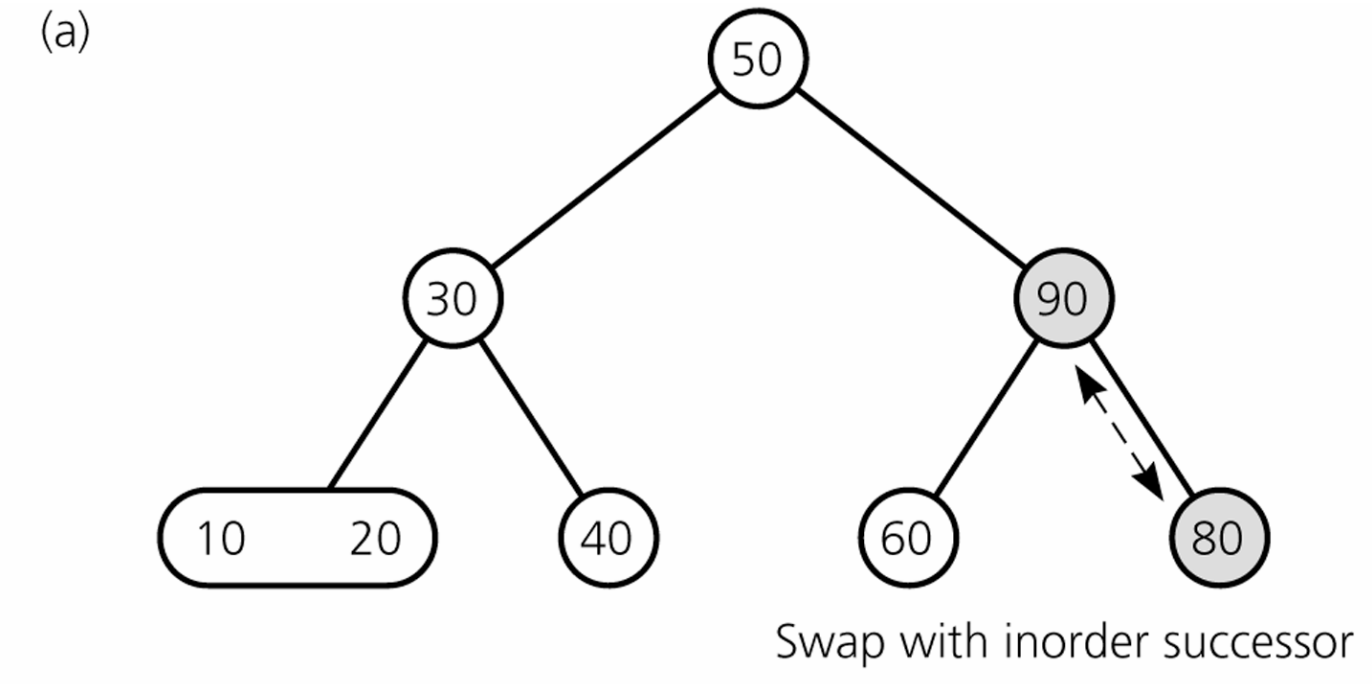
Deleting Items

Delete 80



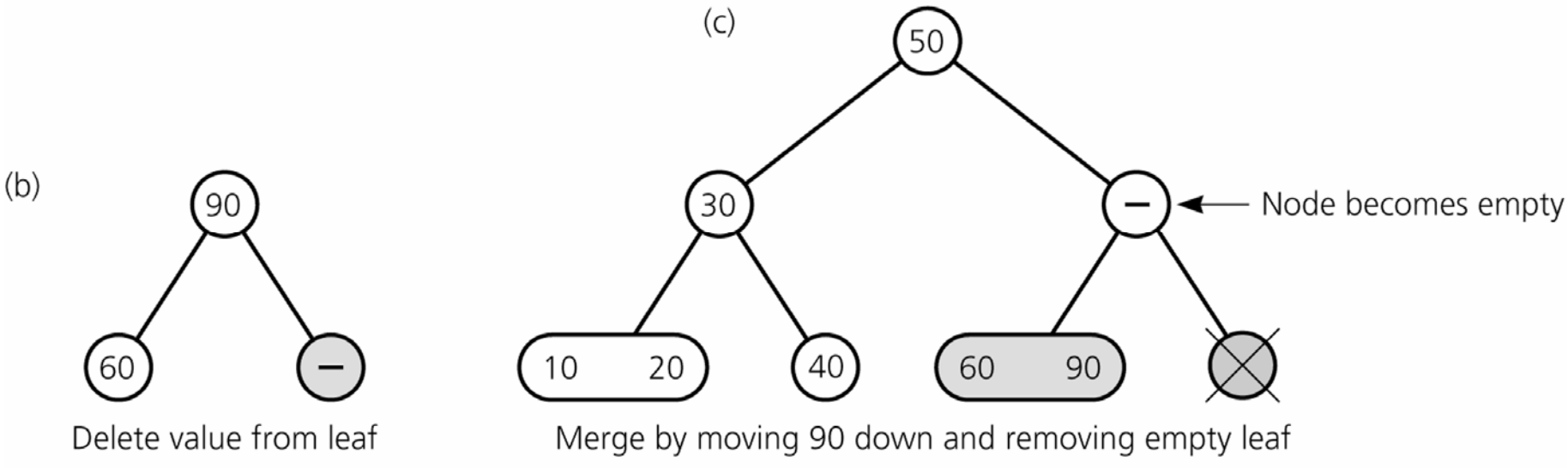
Deleting Items

Deleting 80 ...



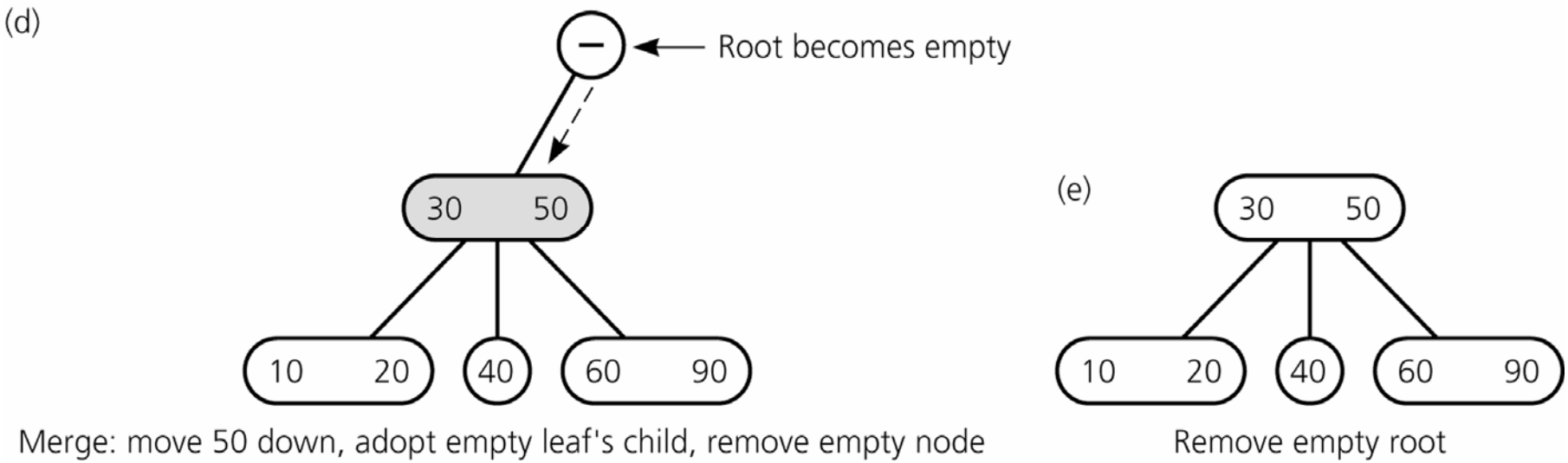
Deleting Items

Deleting 80 ...



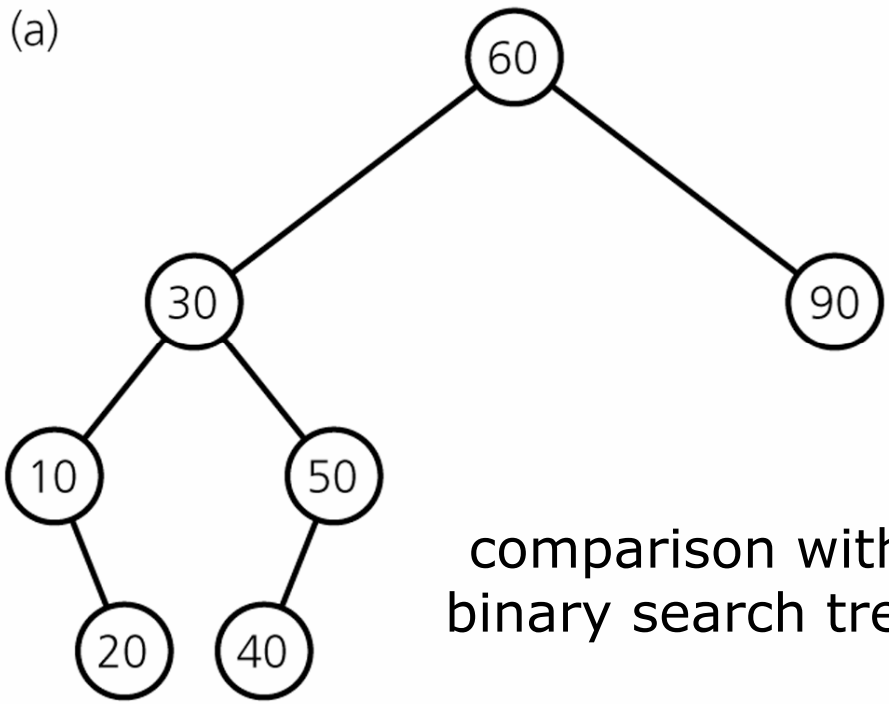
Deleting Items

Deleting 80 ...

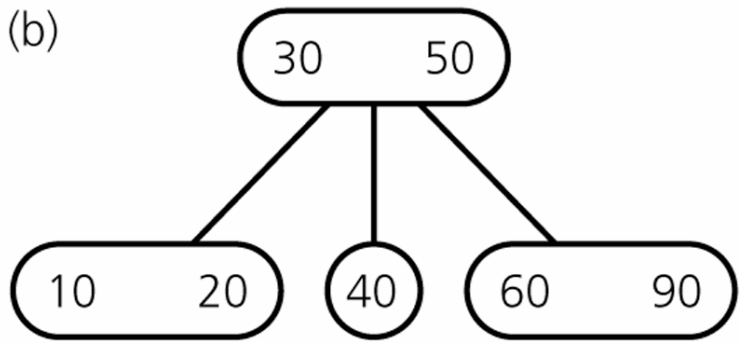


Deleting Items

Final Result



comparison with
binary search tree



Deletion Algorithm (1)

Deleting item *I*:

1. Locate node *P*, which contains item *I*
2. If node *P* is not a leaf → swap *I* with inorder successor

→ deletion always begins at a leaf

3. If leaf node *P* contains another item, just delete item *I*
   else
   try to redistribute nodes from siblings (see next slide)
   if not possible, merge node (see next slide)

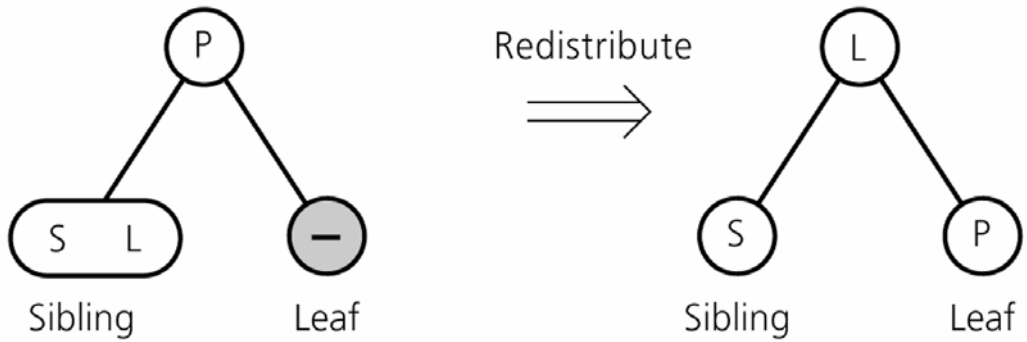
Deletion Algorithm (2)

Redistribution

(a)

A sibling has 2 items:

→ redistribute item between siblings and parent



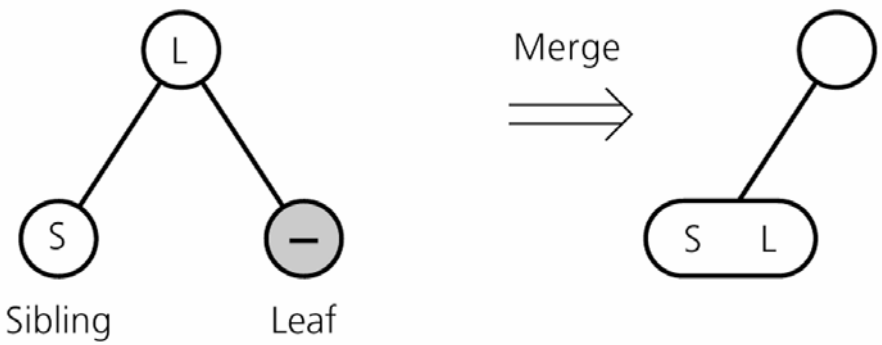
Merging

(b)

No sibling has 2 items:

→ merge node

→ move item from parent to sibling

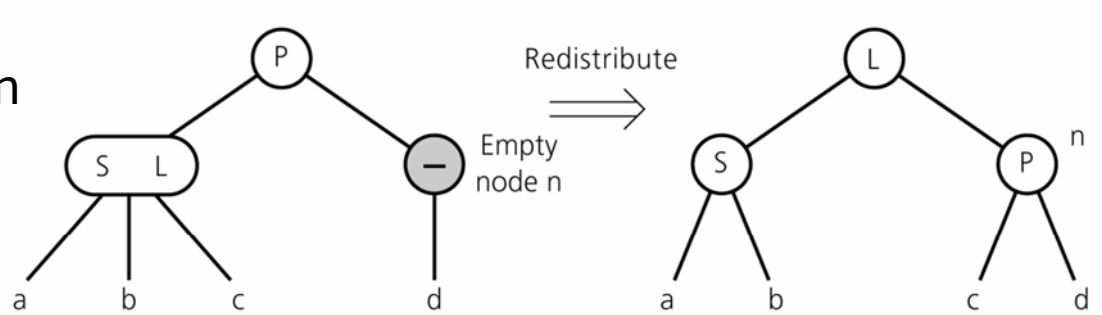


Deletion Algorithm (3)

Redistribution

Internal node *P* has no item

→ redistribute



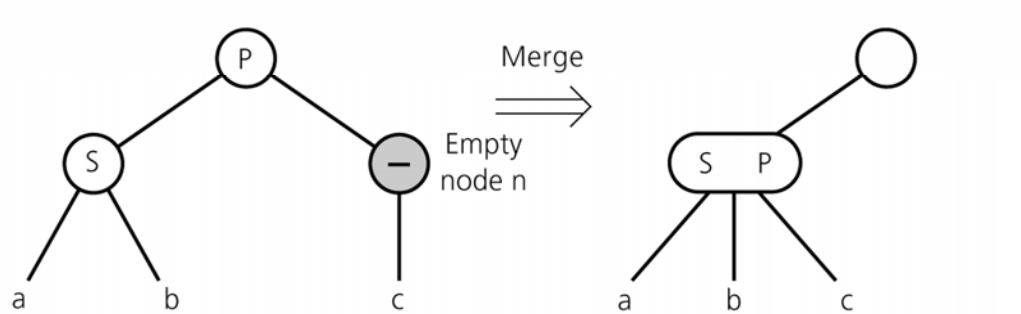
Merging

Redistribution not possible:

→ merge node

→ move item from parent to sibling

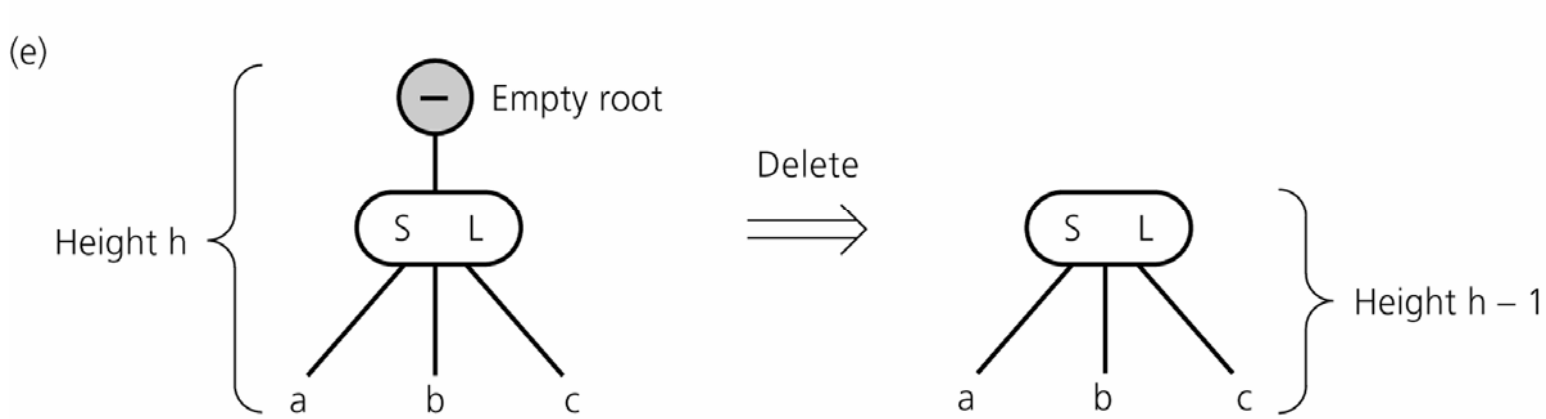
→ adopt child of *P*



Nếu nút cha của nút P lại không có khóa nào cả, thì ta lại tiếp tục phân bố lại khóa hoặc trộn

Deletion Algorithm (4)

If merging process reaches the root and root is without item
→ delete root



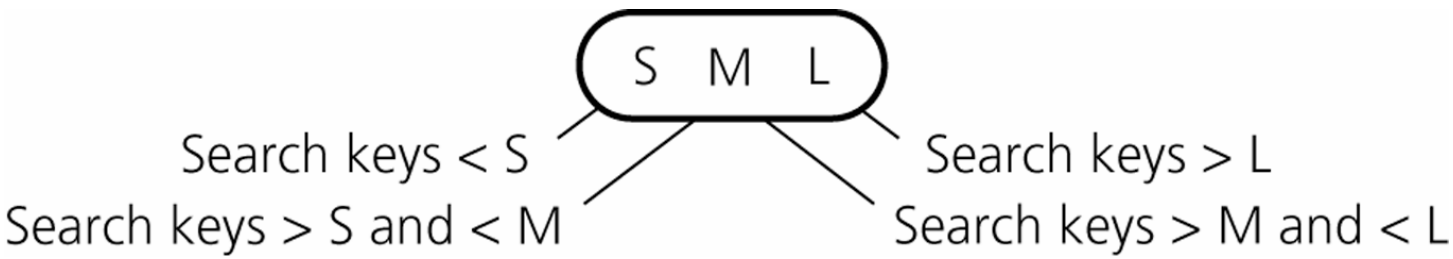
Operations of 2-3 Trees

all operations have time complexity of log n

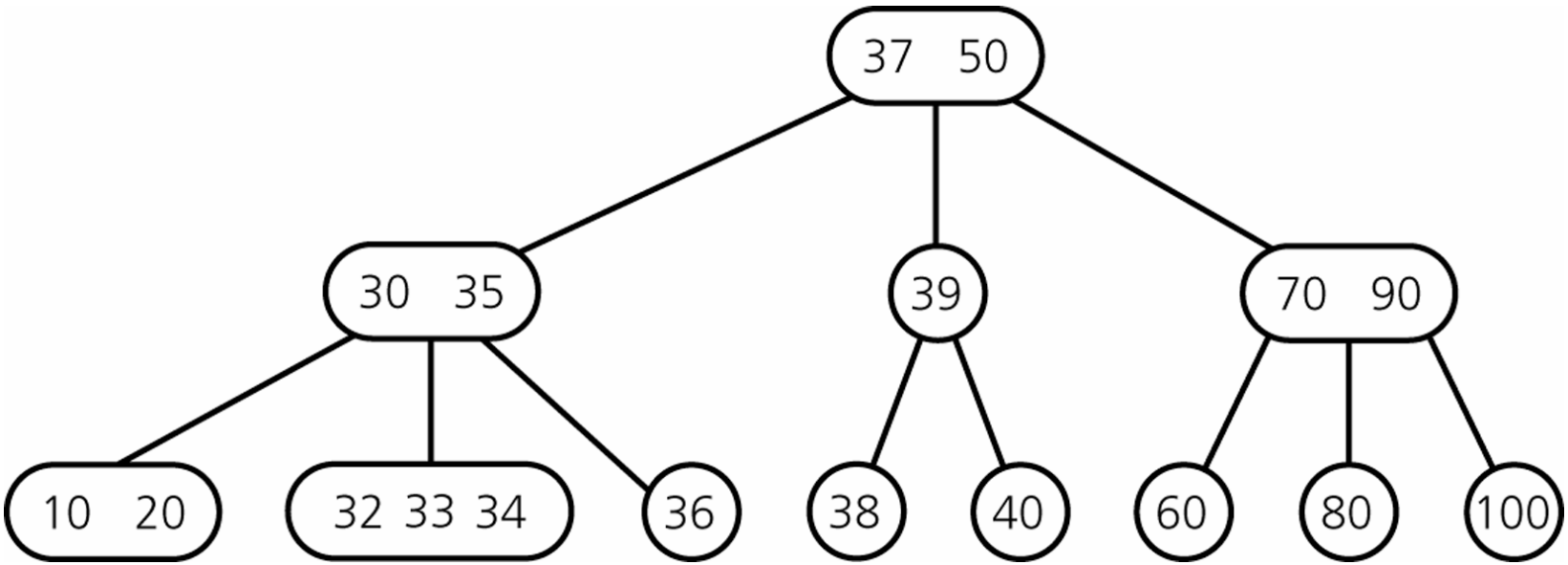
2-3-4 Trees

- Tương tự cây 2-3
- Mỗi nút có tối đa 3 khóa và 4 nút con

4 - node
Có 4 nút con



2-3-4 Tree Example



2-3-4 Tree: Insertion

Insertion procedure:

- Thêm khóa vào nút lá

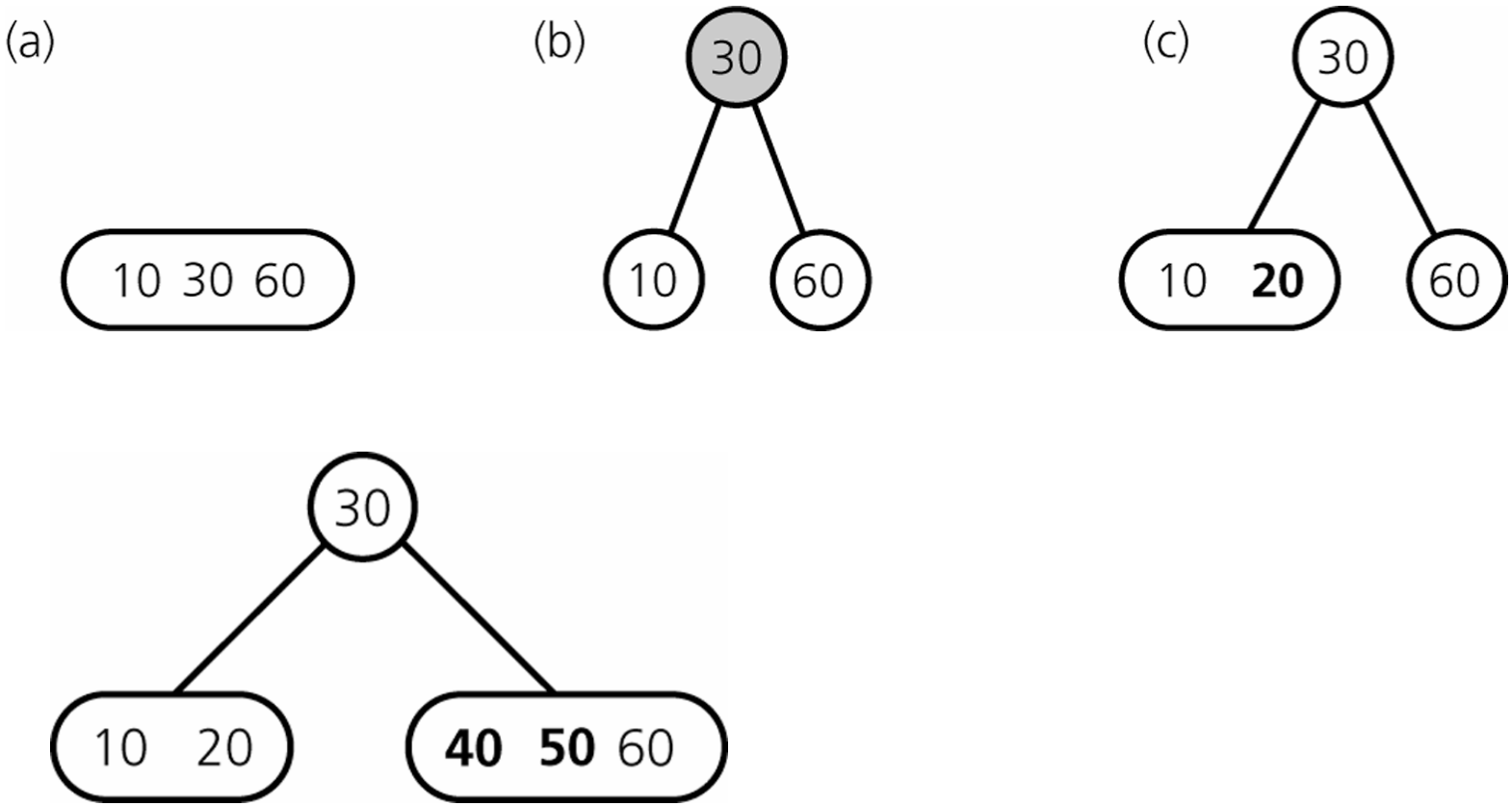
Phương pháp:

- Trên path từ gốc xuống nút lá có thể chèn khóa vào nếu gặp nút đầy (nút 4) thì tách nút

→ insertion can be done in one pass

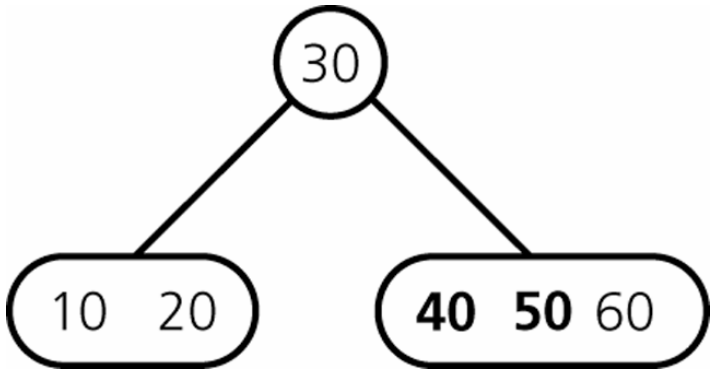
2-3-4 Tree: Insertion

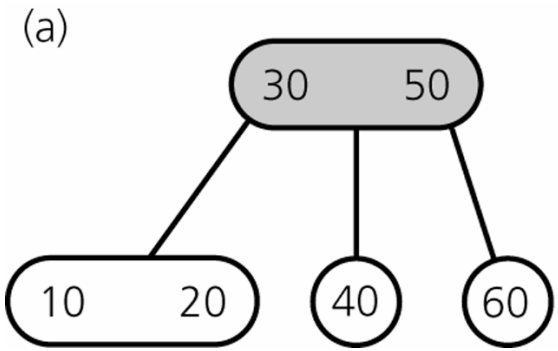
Inserting 60, 30, 10, 20, 50, 40, 70, 80, 15, 90, 100

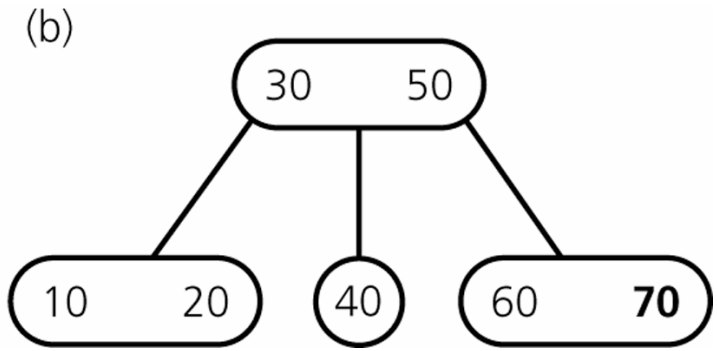


2-3-4 Tree: Insertion

Inserting 70 ...



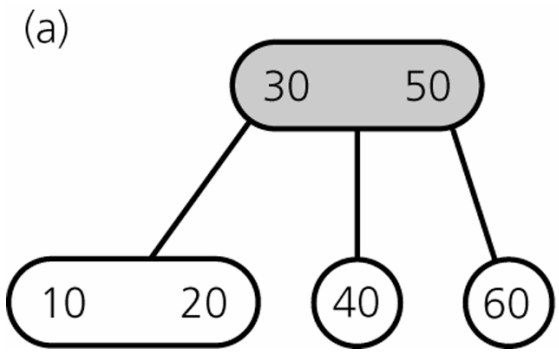


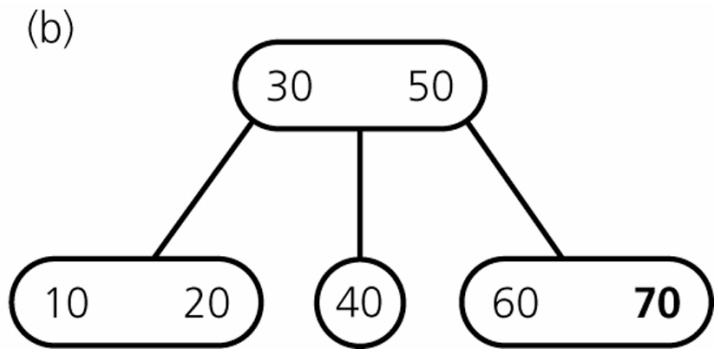


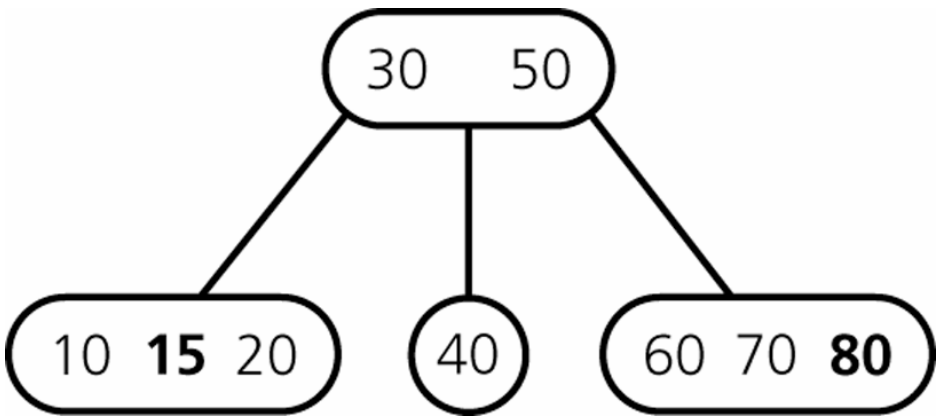
... 80, 15 ...

2-3-4 Tree: Insertion

Inserting 80, 15 …



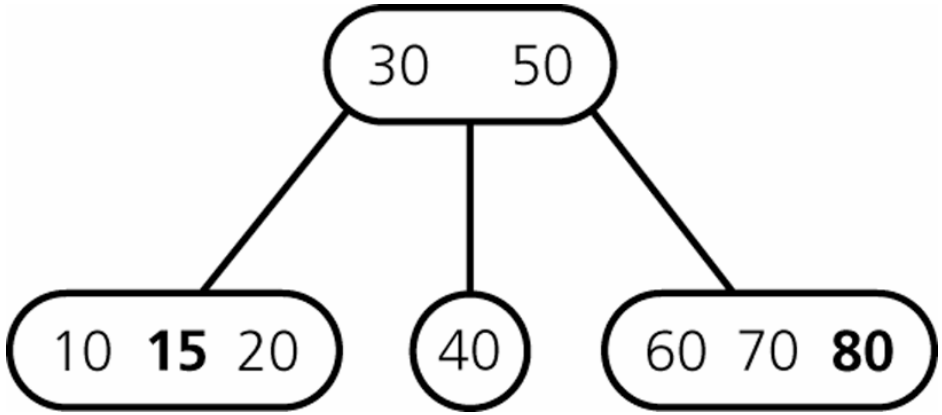




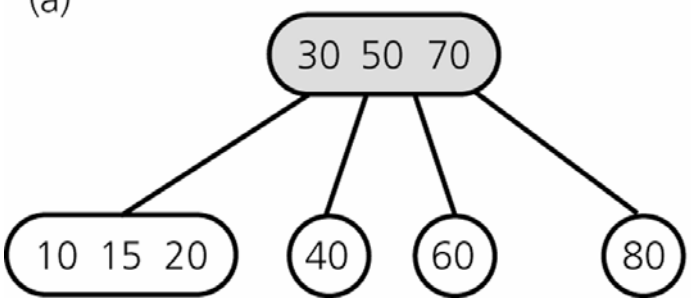
… 90 …

2-3-4 Tree: Insertion

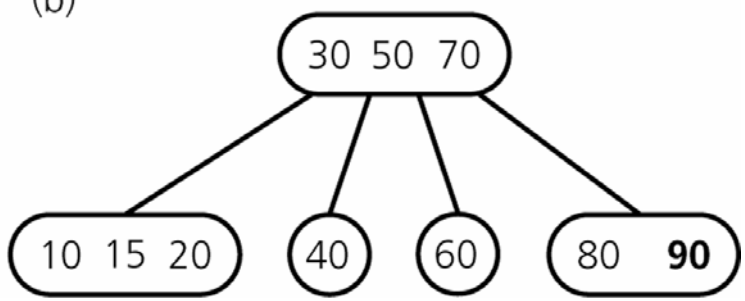
Inserting 90 …



(a)



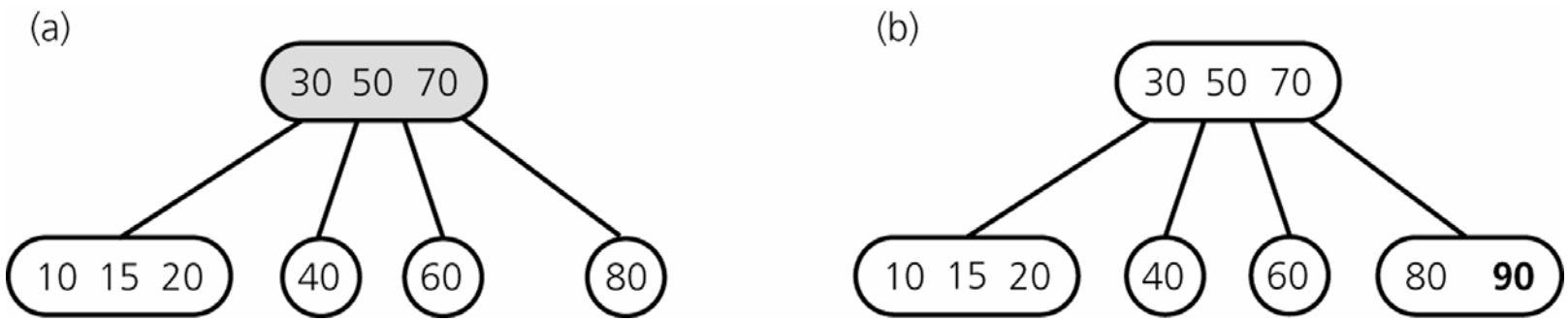
(b)

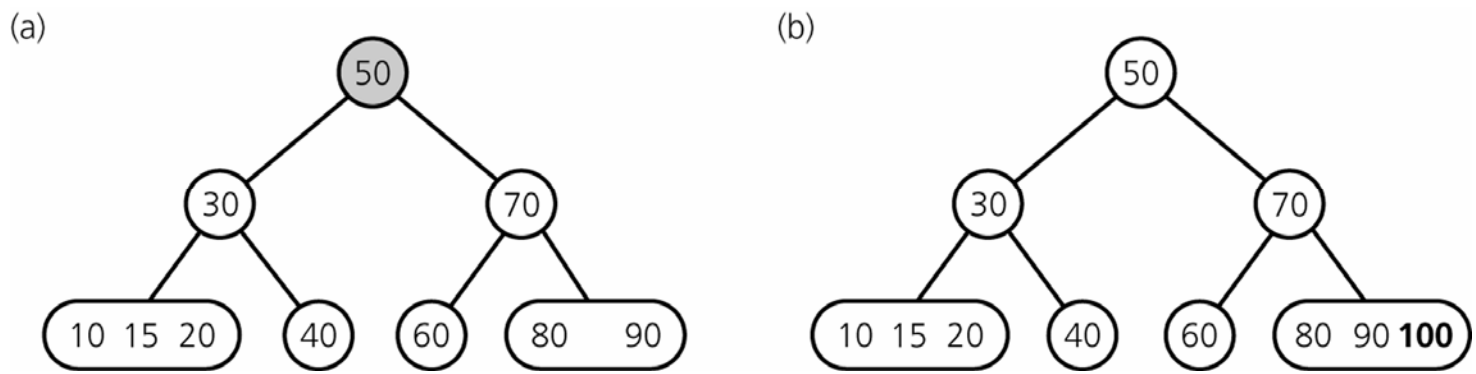


… 100 …

2-3-4 Tree: Insertion

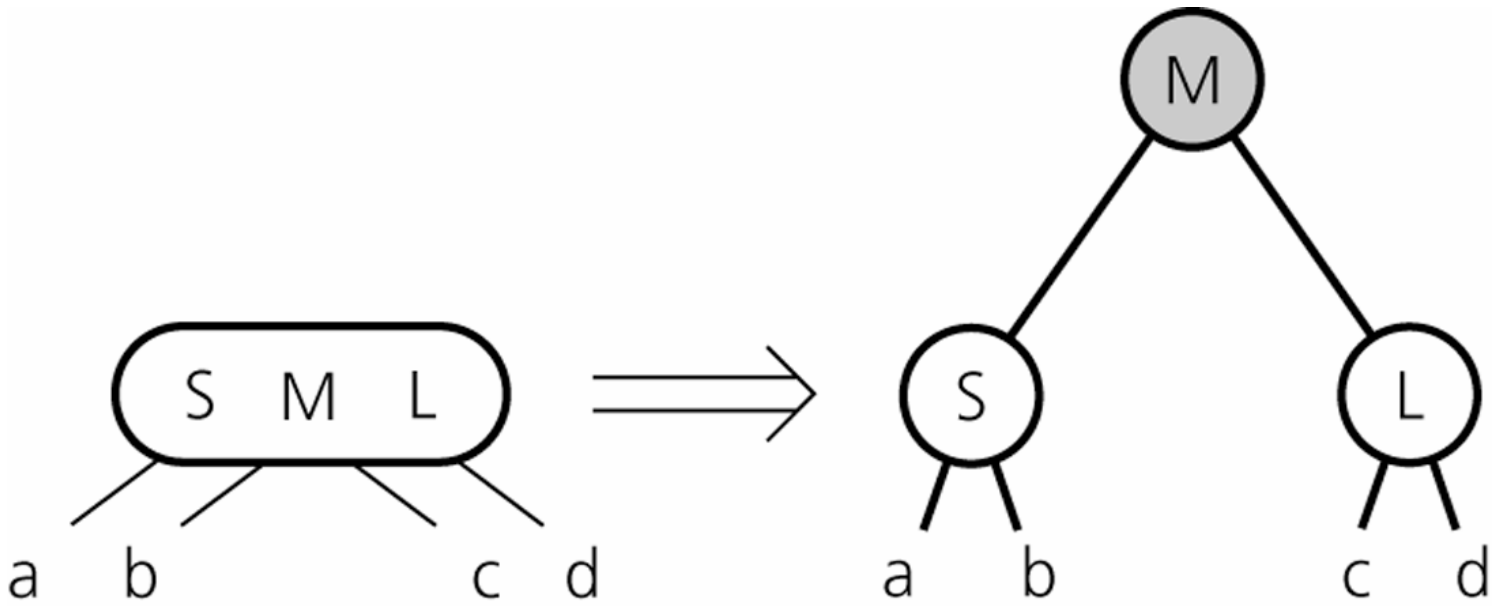
Inserting 100 …





2-3-4 Tree: Insertion Procedure

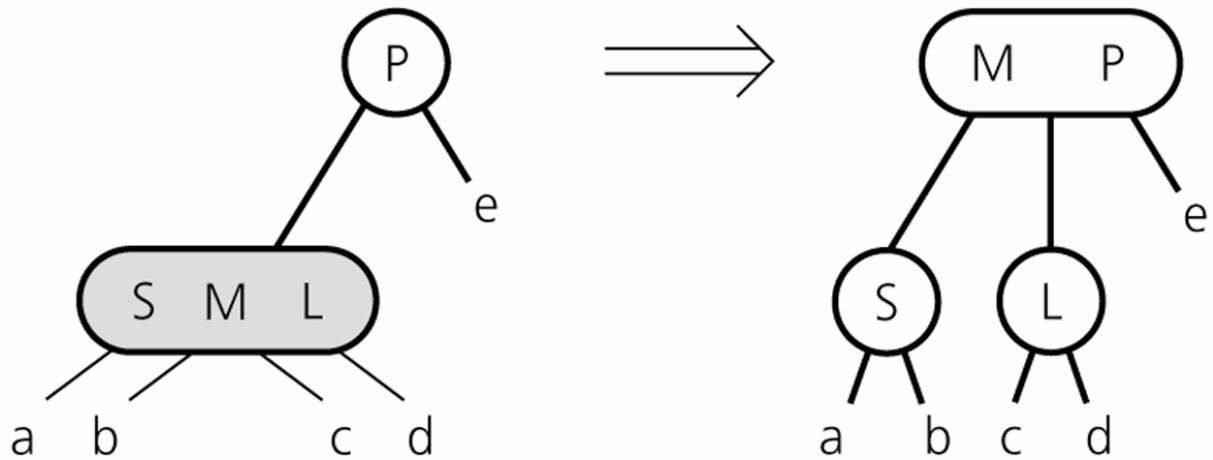
Tách nút trong quá trình thêm
(đẩy khóa giữa lên nút cha)



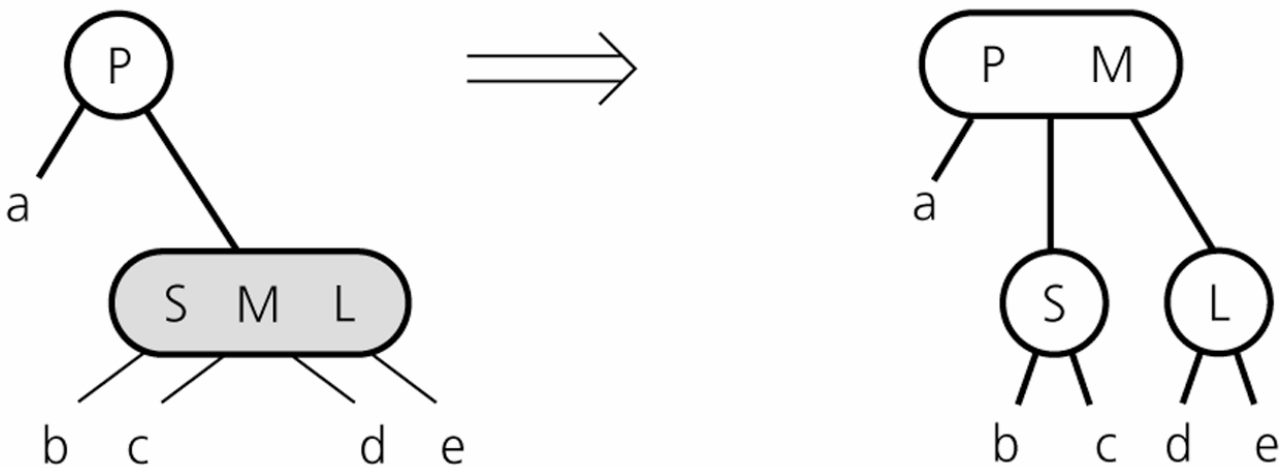
2-3-4 Tree: Insertion Procedure

Tách nút khi nút cha của nó có 2 nút con

(a)

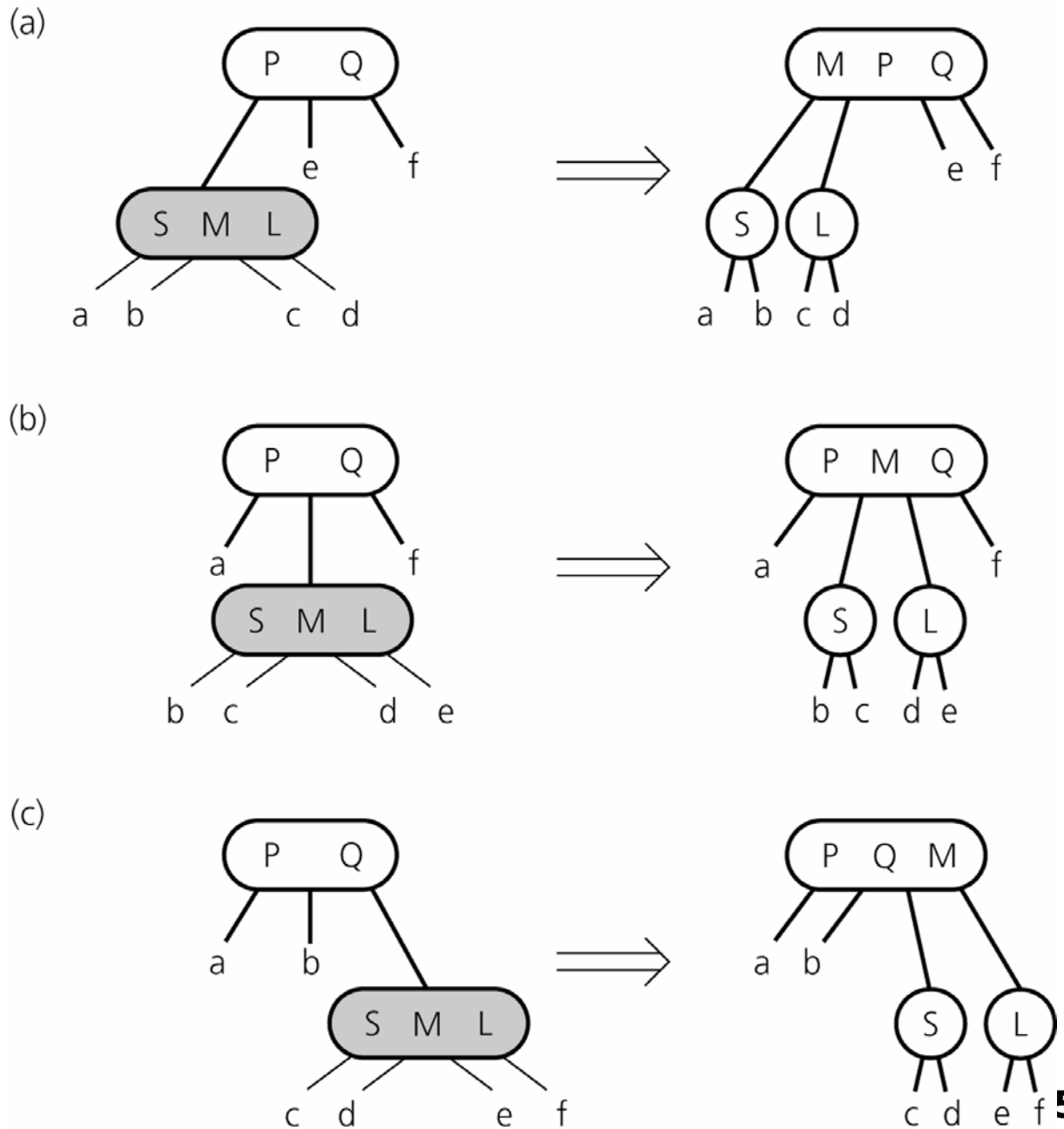


(b)



2-3-4 Tree: Insertion Procedure

Tách nút khi nút cha của nó có 3 nút con



2-3-4 Tree: Deletion

Deletion procedure:

- Chọn khóa thay thế ở nút lá bằng cách đẩy dần các khóa ở mức dưới lên.

Phương pháp

- Trên path từ gốc xuống nút lá chứa khóa thay thế nếu gặp nút đơn chứa 1 khóa (trừ nút gốc) thì tăng khóa cho nút này.

→ deletion can be done in one pass

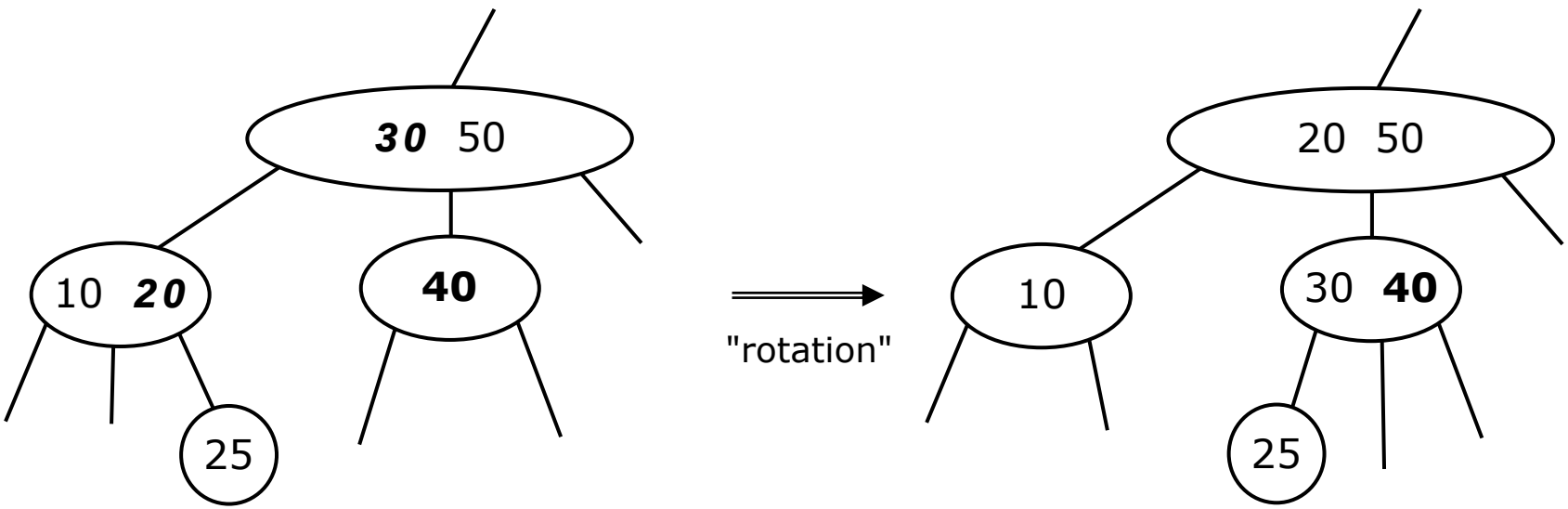
2-3-4 Tree: Deletion

Case 1: Nút anh em có từ 2 đến 3 khóa

→Lấy một khóa từ nút anh em đẩy lên nút cha

→Lấy một khóa từ nút cha xuống

→ Ví dụ cần xóa **40**

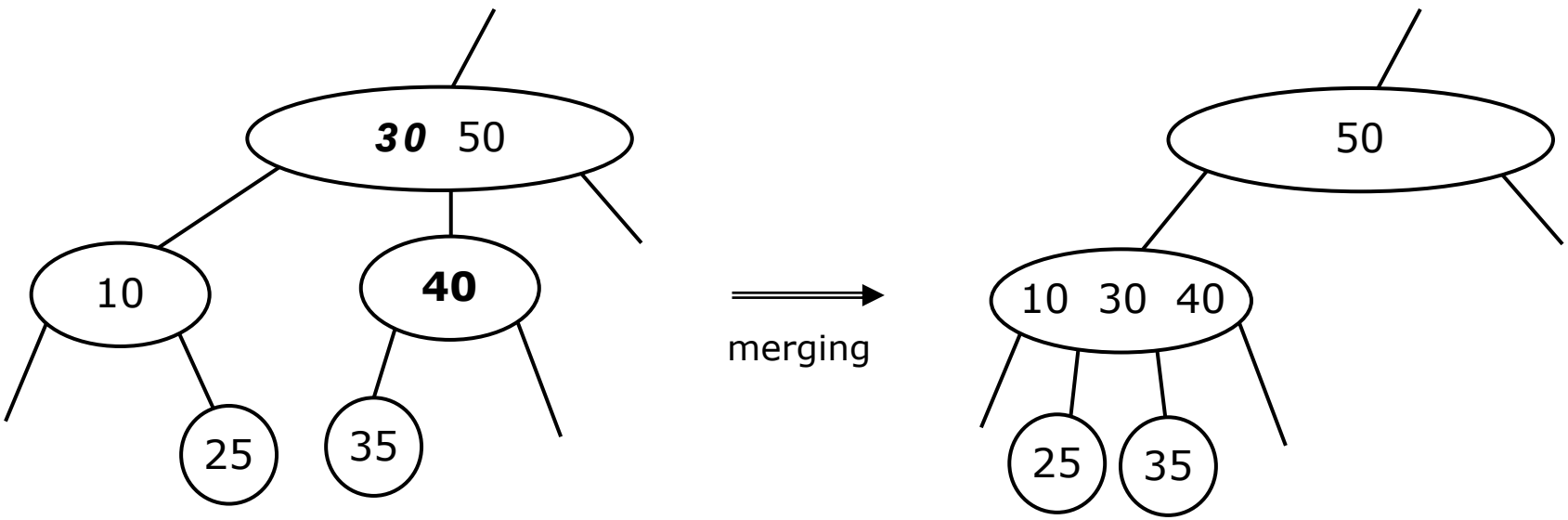


2-3-4 Tree: Deletion

Case 2: Nút anh em chỉ có 1 khóa

→Lấy một khóa từ nút cha xuống ghép nó với nút và với nút anh em

→ Ví dụ cần xóa **40**



2-3-4 Tree: Deletion Practice

Delete 32, 35, 40, 38, 39, 37, 60