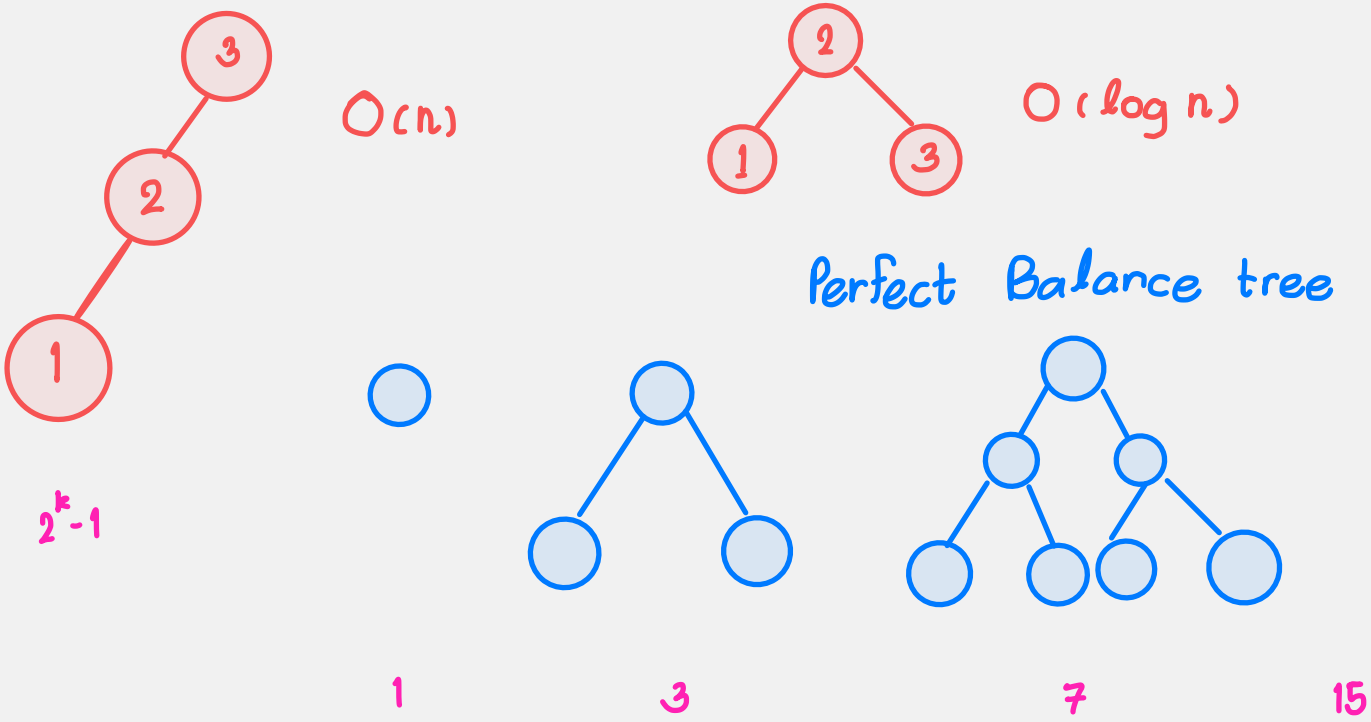
ทบทวน

Perfect balance trees : Binary search trees ที่มี balanced condition ต้องมี node 2^k-1



AVL Trees : a binary search tree, except that for every node in the tree, the height of the left and the right subtree can differ by at most 1. ความสูงของ left subtree กับ right subtree ต่างกันไม่เกิน 1

- single rotation
  3, 2, 1
  1, 2, 3

- double rotation
  3, 1, 2
  1, 3, 2

Code Insert

```
1 ถ้า tree == NULL
2     สร้าง node ใหม่ ใส่ค่า

3 else
4     ถ้า x น้อยกว่า tree->value  ให้ recursive ลงไปทางซ้าย
5         tree->left = insert (tree->left)
6     ถ้าความสูงของทรีทางซ้ายและทางขวาต่างกันเกิน 2
7         ถ้า x น้อยกว่า tree->left->value
8             single rotation  (อยู่นอกช่วง)
9         else
10            double rotation (อยู่ในช่วง)

11 update ความสูง
12 return tree
```

เมื่อ insert แล้ว return กลับมาเช็คความสูง

3.5 Splay Trees

- มี bianry search tree อยู่แล้ว แต่ไม่ balance
- O(N) worst-case time per operation for binary search trees.

แก้ปัญหาทรีที่มีอยู่แล้วแต่ไม่ balance ด้วยเทคนิค splay trees

- กรณีมีการใช้งาน(accessed) ข้อมูลใน trees เช่น การค้นหาข้อมูล จะทำการค้นหาและนำข้อมูลที่ค้นหาได้แล้วนั้นกลับขึ้นมาเป็น root
- After a node is accessed, it is pushed to the root

มีทรีอยู่แล้ว
แต่ไม่ Balance

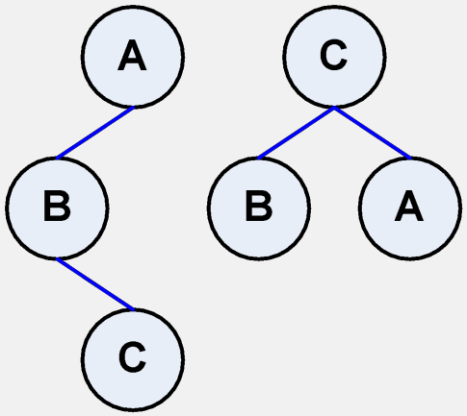
3.5 Splay Trees

- O(N) worst-case time per operation for binary search trees.
- After a node is accessed, it is pushed to the root

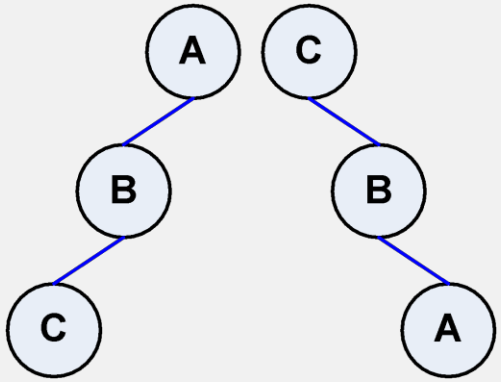
Rotation idea (วิธีการนำ node ที่ถูก accessed ขึ้นมาเป็น root)

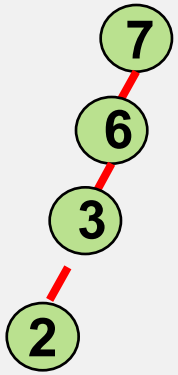
เรียกว่า <u>Transform</u>

1. zig-zag

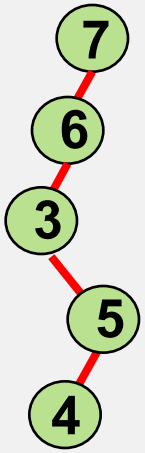


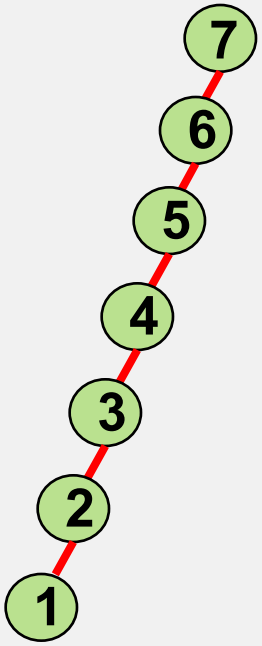
2. zig-zig





ถ้า zig-zag หรือ zig-zig แล้วโหนดไม่ถึง root จะต้องใช้ single rotation หมุนโหนดต่อ

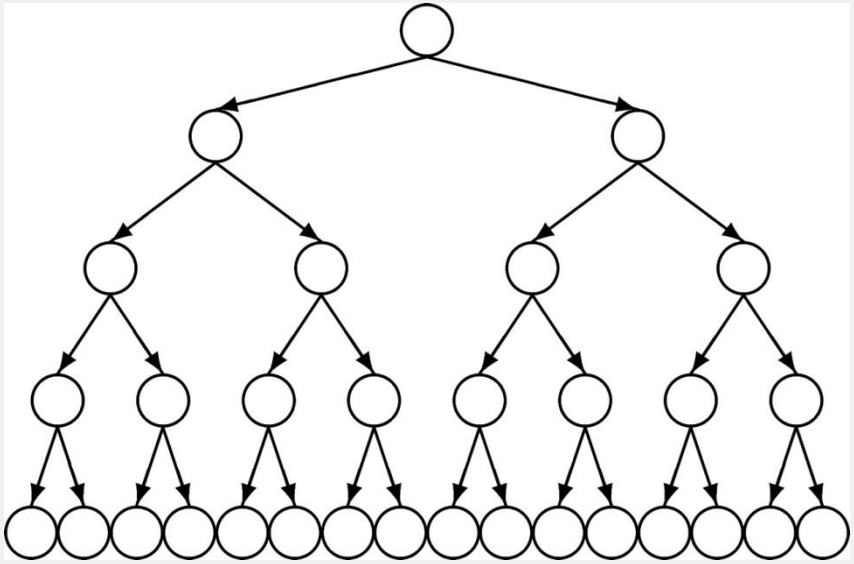

7
6
3
5
4

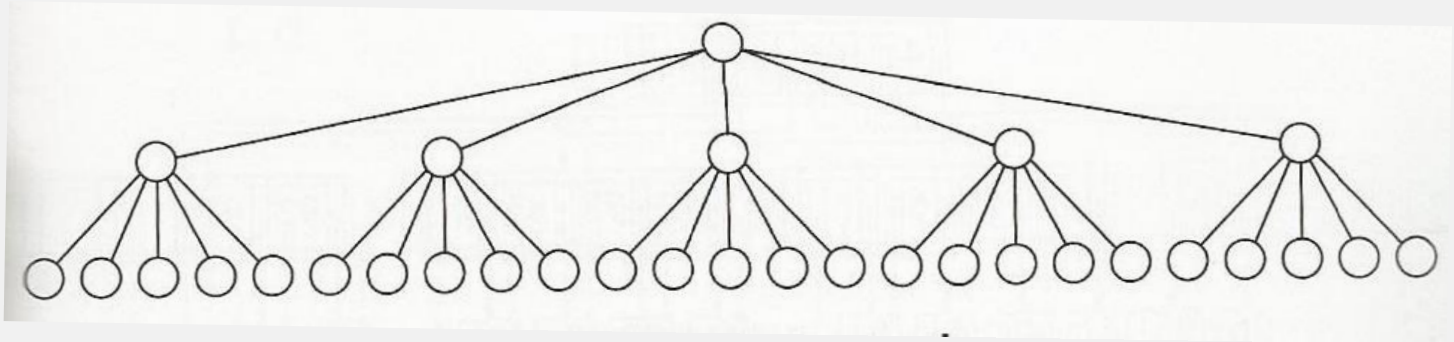

7
6
5
4
3
2
1

3.6 B-Trees

Complete binary

Tree 31 nodes

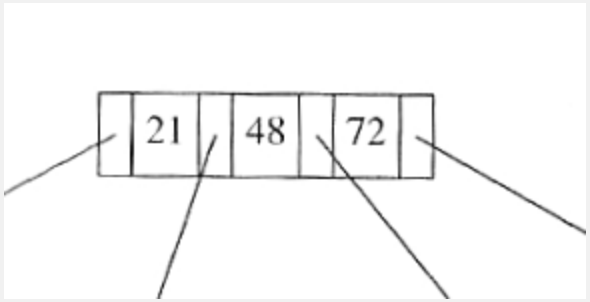


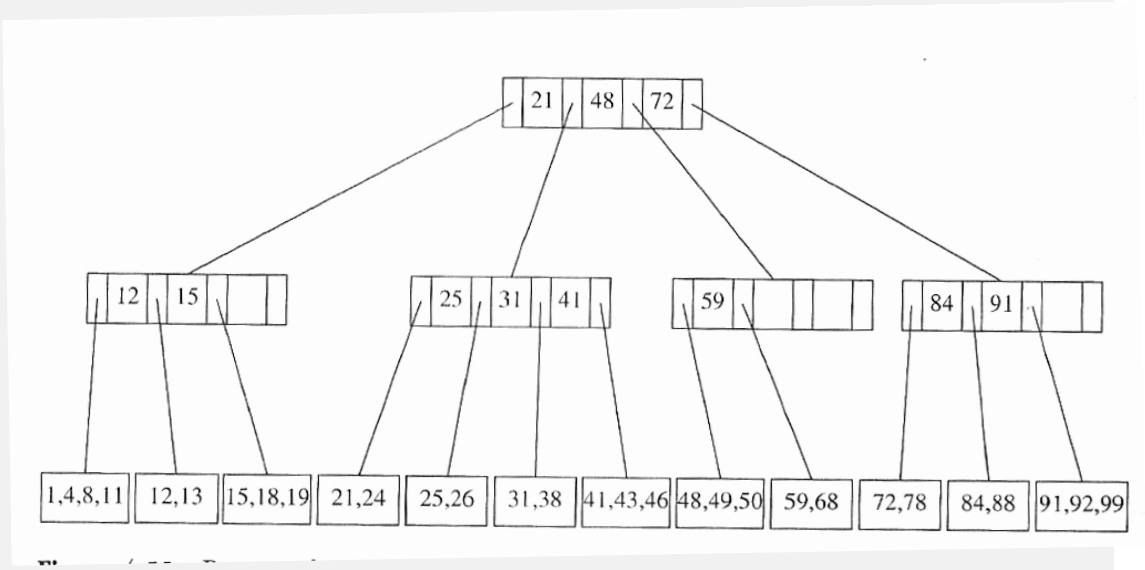


- สำหรับข้อมูลขนาดใหญ่
- ส่วนใหญ่ใช้ในระบบฐานข้อมูลและระบบไฟล์

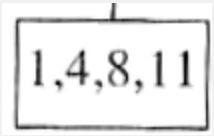
ตย. Order 4

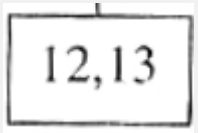
Nonleaf node





Leaf node

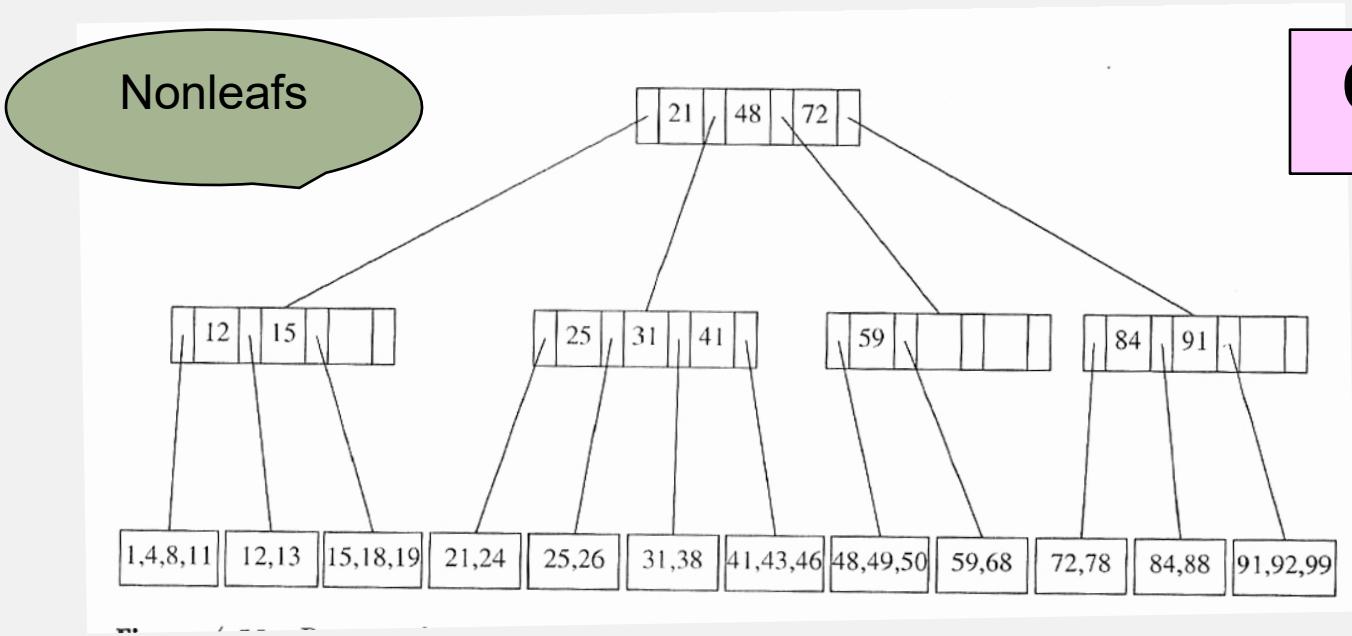




3.6 B-Trees

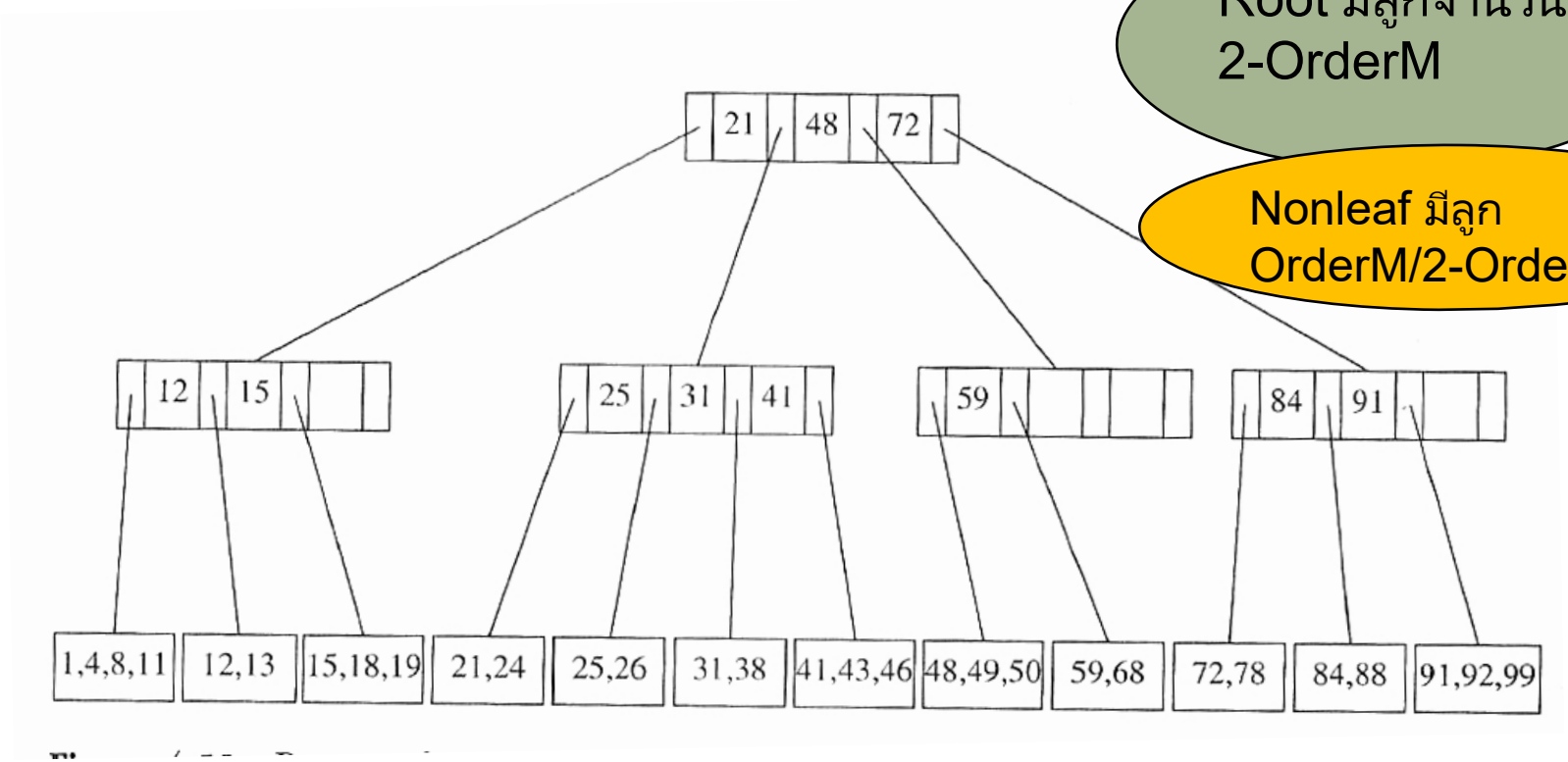
A B-tree of ***order m*** is a tree with the following structural properties:

1) The data items are stores at leaves.
2) The nonleafs nodes store up to M-1 keys to guide the searching; key i represents the smallest key in subtrees i+1

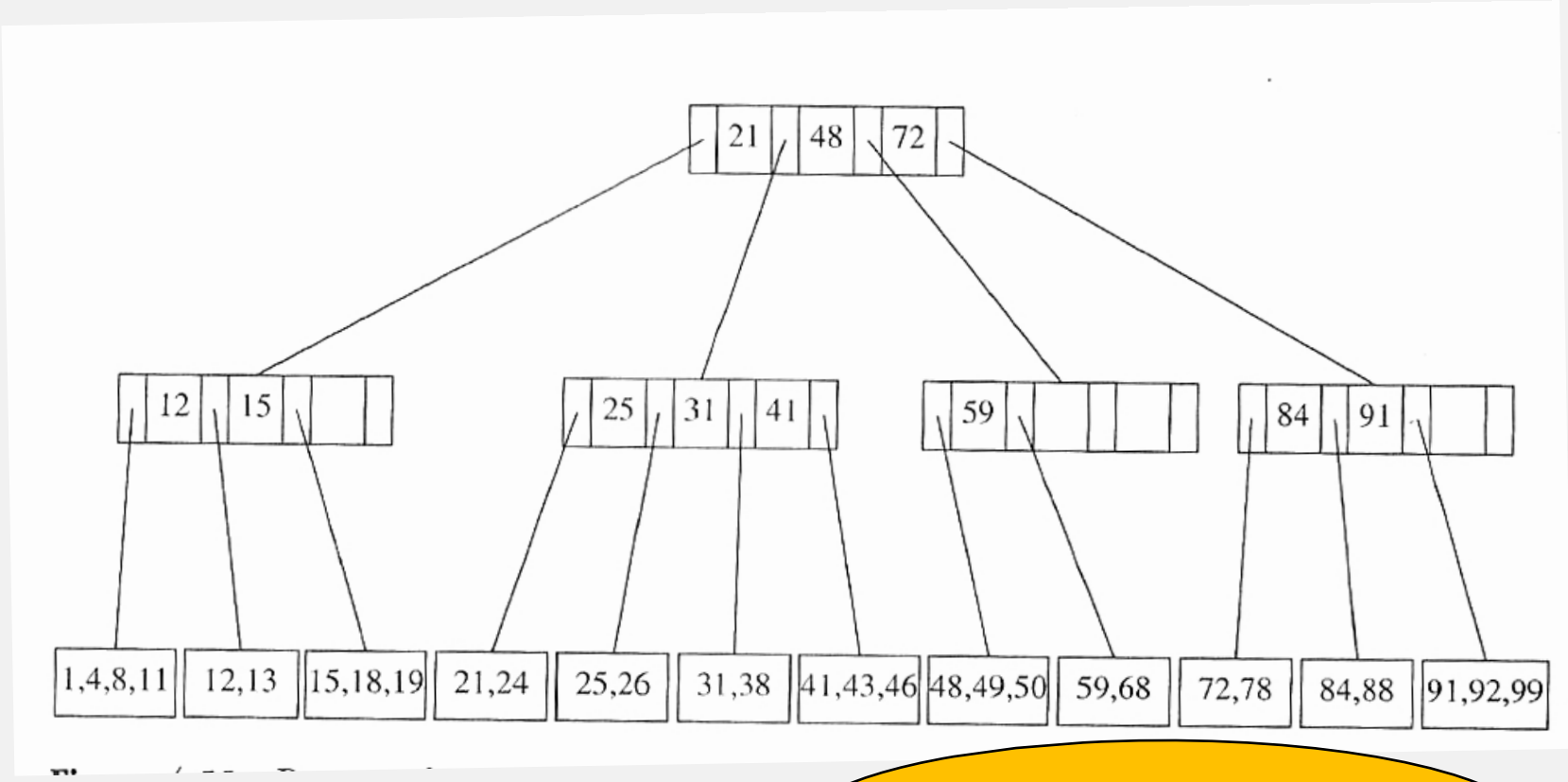


3) The root is either a leaf or has between two and M children.

4) All nonleaf nodes(Except the root) have between ⌈M/2⌉ and M children



5) All leaves are at the same depth and have between ⌈L/2⌉ and L children.



leaf มีลูกจำนวน L/2-OrderM

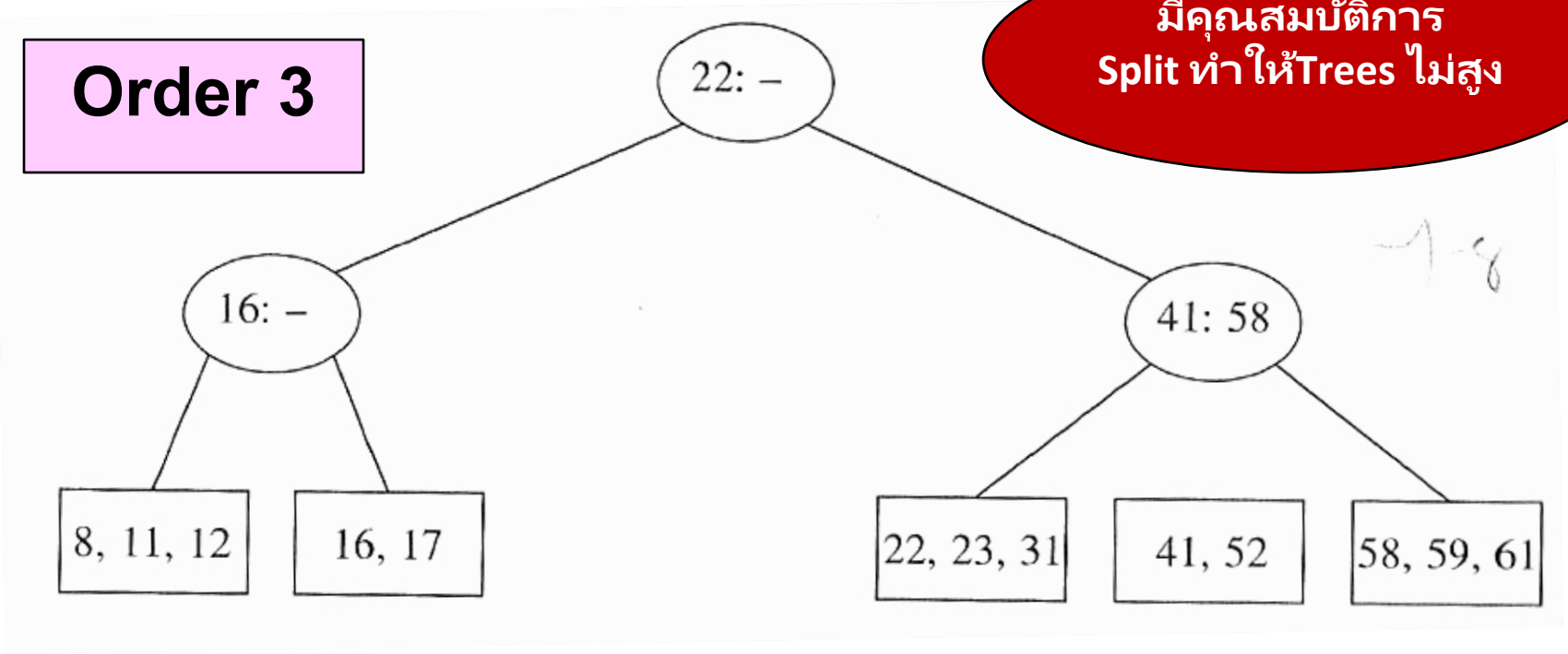
Order 3

มีคุณสมบัติการ
Split ทำให้Trees ไม่สูง

Insert 22,16,41,58,61,59

Order 3

มีคุณสมบัติการ
Split ทำให้Trees ไม่สูง



Insert 18, 1, 19, 28

Order 5

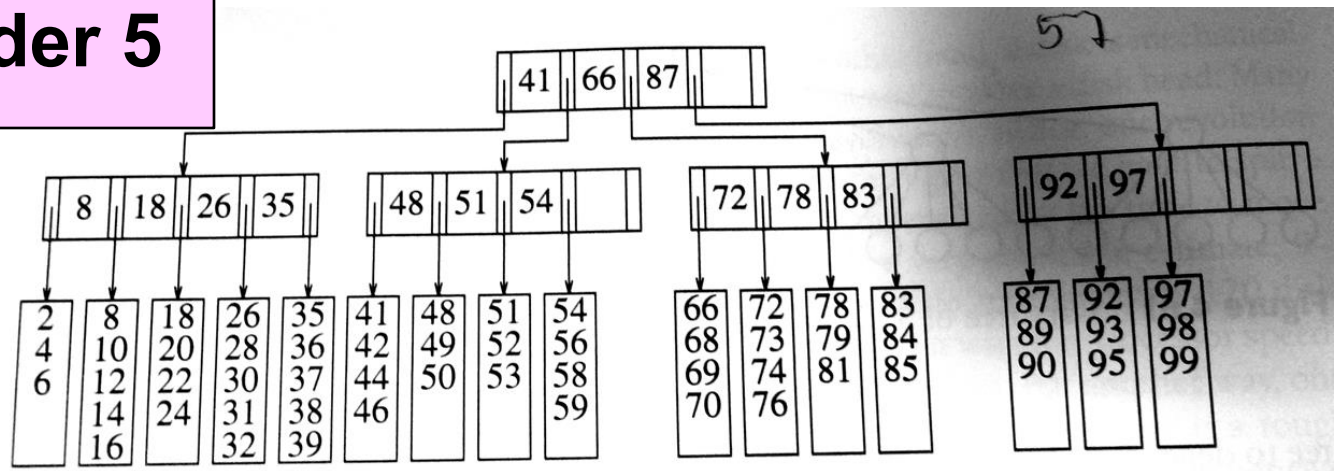
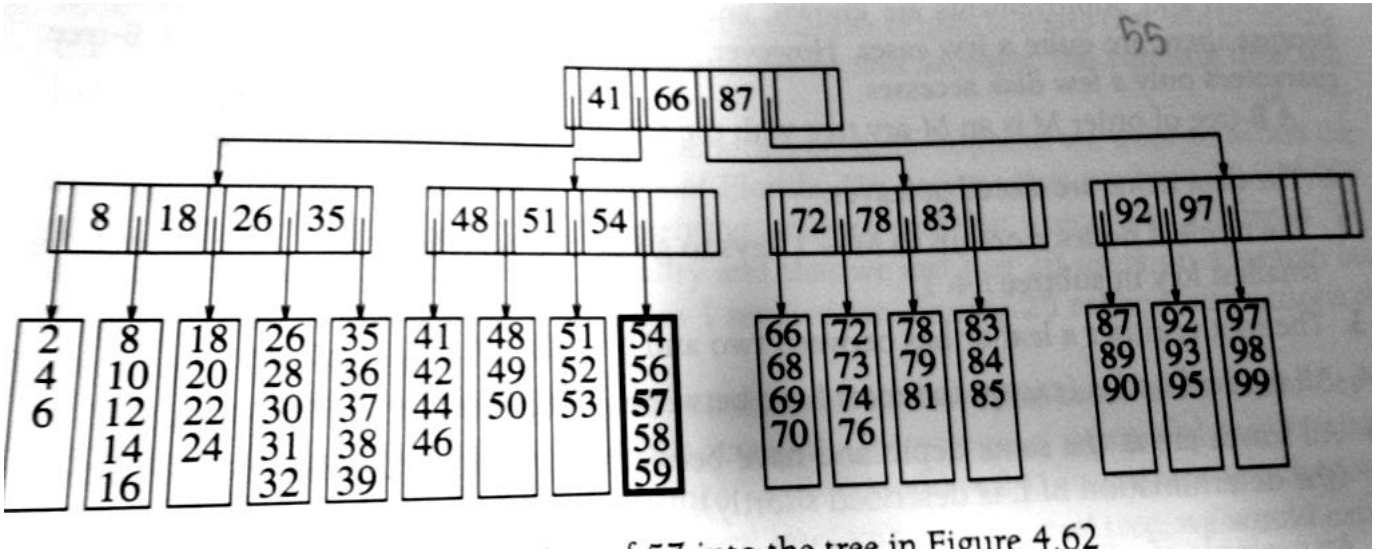
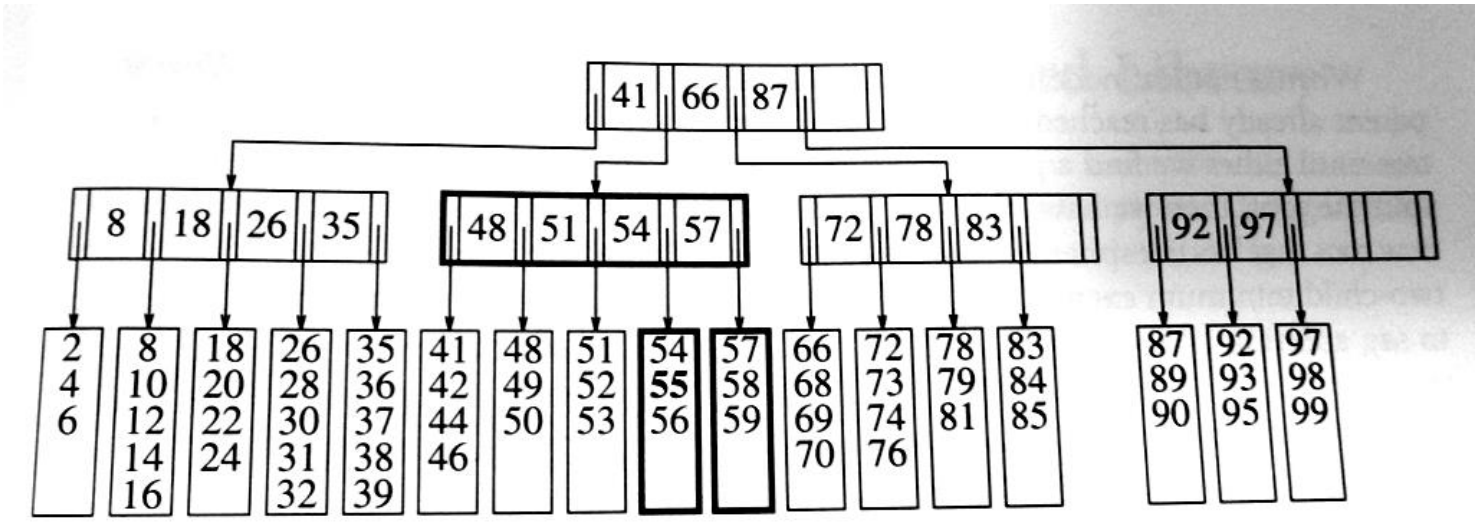
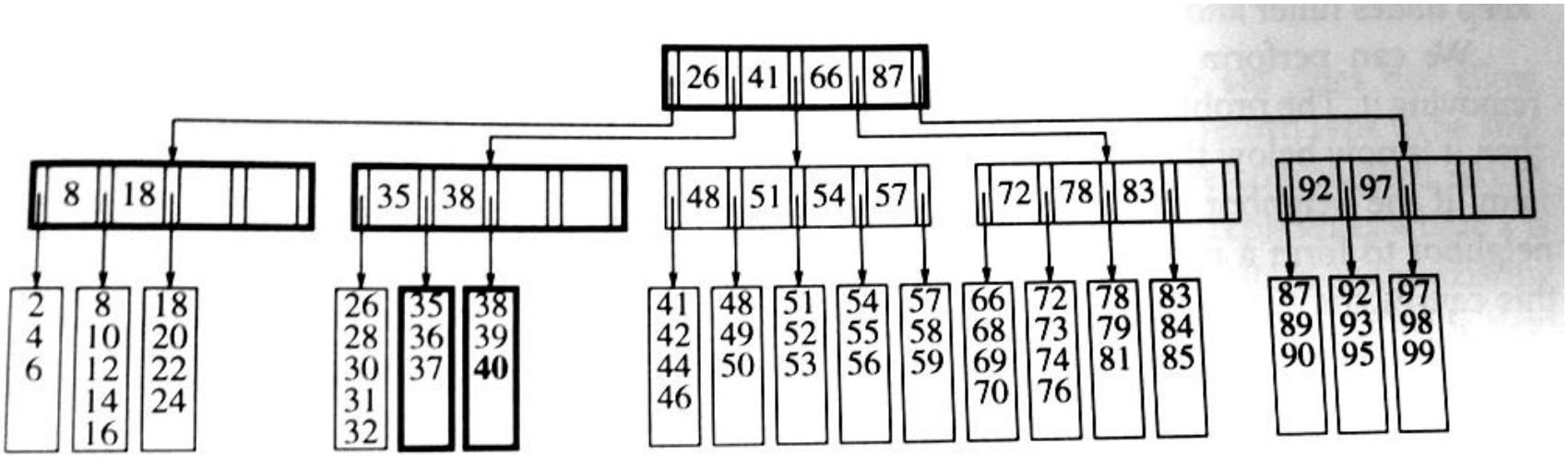


Figure 4.62 B-tree of order 5




41 66 87
8 18 26 35
48 51 54 57
72 78 83
92 97
2 4 6
8 10 12 14 16
18 20 22 24
26 28 30 31 32
35 36 37 38 39
41 42 44 46
48 49 50
51 52 53
54 55 56
57 58 59
66 68 69 70
72 73 74 76
78 79 81
83 84 85
87 89 90
92 93 95
97 98 99


26 41 66 87
8 18
35 38
48 51 54 57
72 78 83
92 97
2 4 6
8 10 12 14 16
18 20 22 24
26 28 30 31 32
35 36 37
38 39 40
41 42 44 46
48 49 50
51 52 53
54 55 56
57 58 59
66 68 69 70
72 73 74 76
78 79 81
83 84 85
87 89 90
92 93 95
97 98 99

การบ้าน AVL Trees

1. จงเขียนโปรแกรมสร้าง AVL Trees ตาม menu ดังนี้

==========
MENU
==========
1) Insert
2) Check height
3) Print Inorder
Please choose >

กรณีเลือกข้อ 1
Enter : 3
Success!
จากนั้นกลับไปยัง menu
ให้ทดลอง insert ข้อมูลต่อไปนี้ 3,2,1,7,5,4

กรณีเลือกข้อ 2
จะทำการแสดงว่าข้อมูลที่ enter มีความสูงเท่าใด
Check height : 2
Height = 2
หรือ
Check height : 5
Height = 1
จากนั้นกลับไป menu

ถ้าเลือกข้อ 3
Print inorder : 1 2 3 4 5 7