Data Structures & Algorithms

Tree

B Tree

B Tree

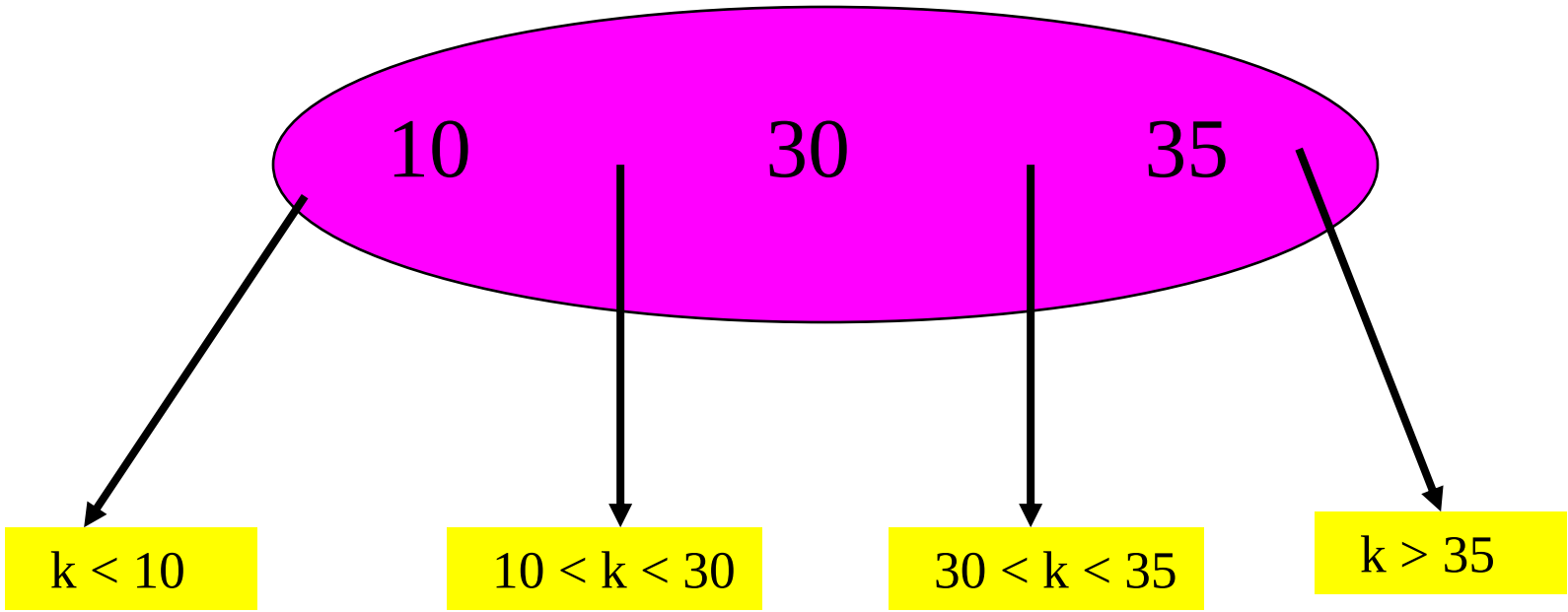
Invented by Rudolf **B**ayer (1970)

A B-tree **of order** M is an M-way search tree with three properties :

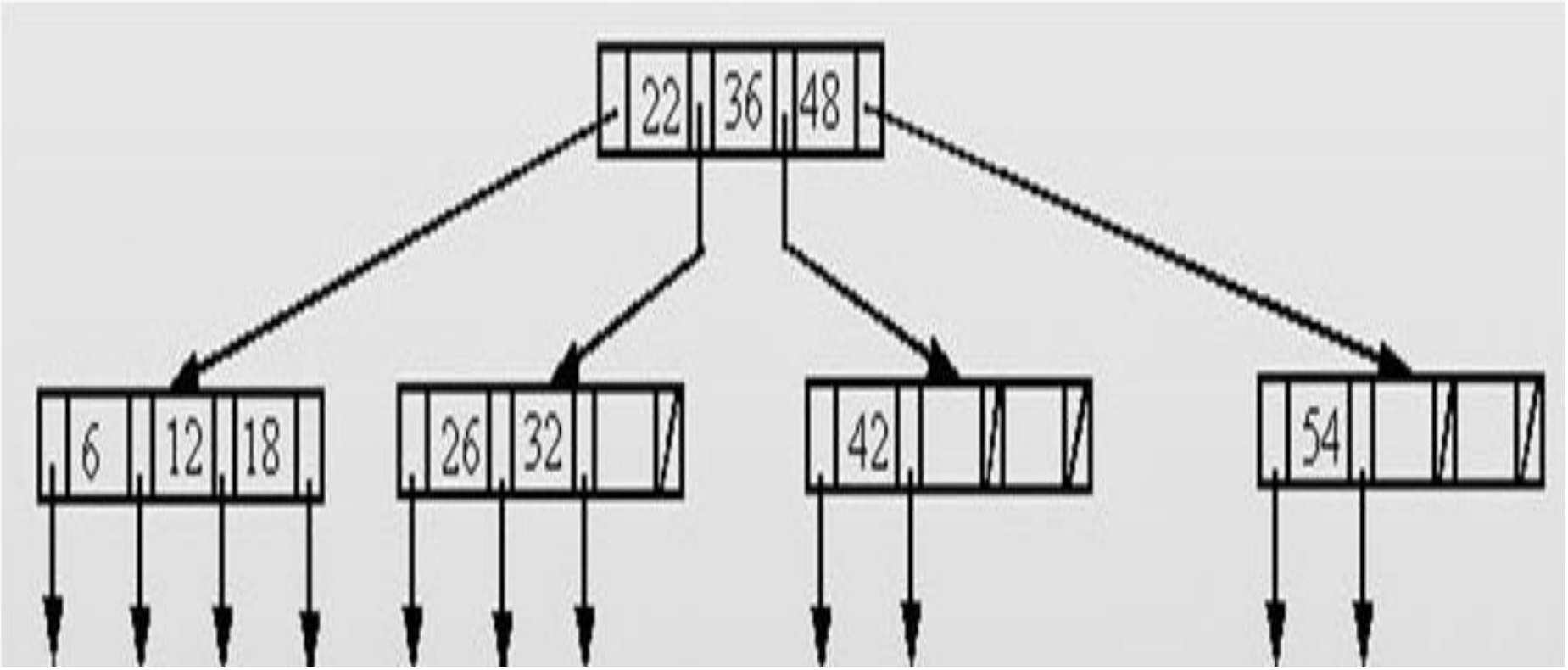
1. It is perfectly balanced: every leaf node is at the same depth/level
2. Every internal node other than the root, is at least half-full, i.e. M/2-1 ≤ #keys ≤ M-1
3. Every internal node with k keys has k+1 non-null children

For simplicity we consider M even and we use t=M/2:
We redefine 2. as
Every internal node other than the root is at least half-full, i.e. t-1≤ #keys ≤2t-1, t≤ #children ≤2t

4-Way Search Tree



Example of a B-tree of order 4

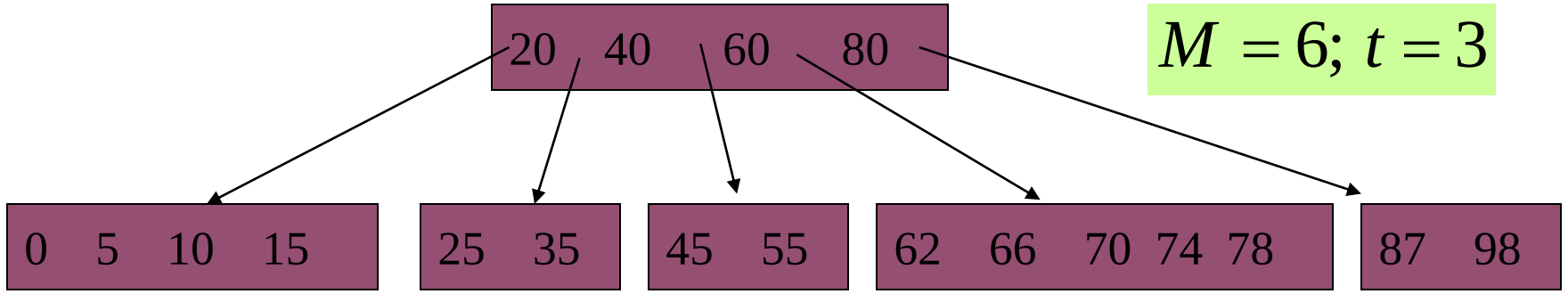


B Tree Insertion

To insert X

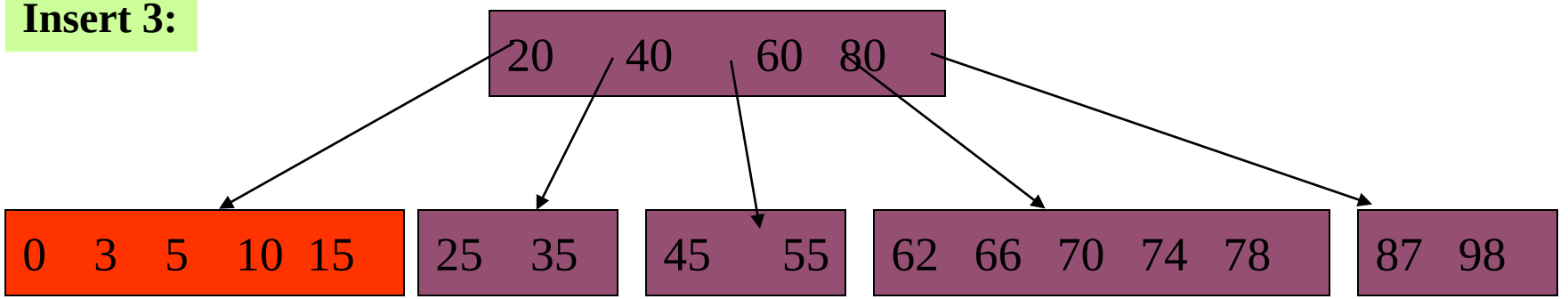
1. As in M-way tree find the leaf node to which X should be added
2. Add X to this node in the appropriate place among the values already there (there are no subtrees to worry about)
3. Number of values in the node after adding the key:
   - Fewer than 2t-1: done
   - Equal to 2t: overflowed
4. Split overflowed node, into two, promoting the middle key to the node's parent

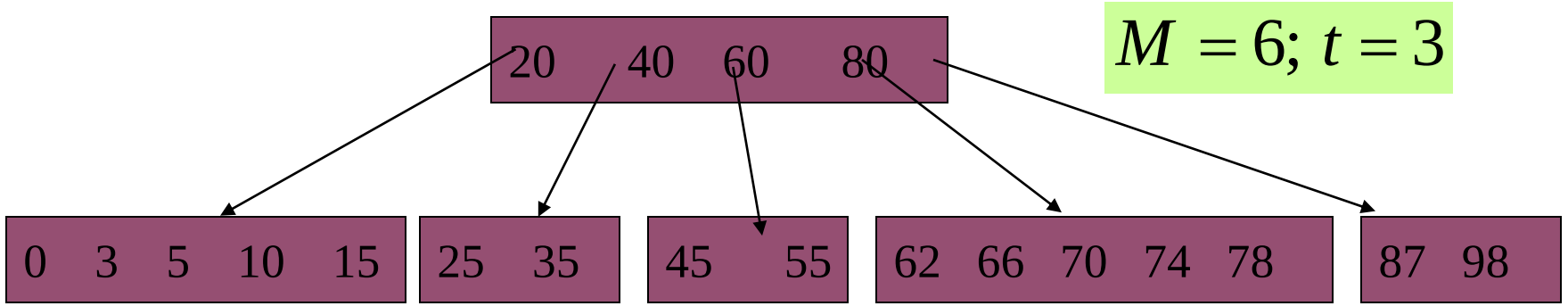
Insert example



$M = 6;\ t = 3$

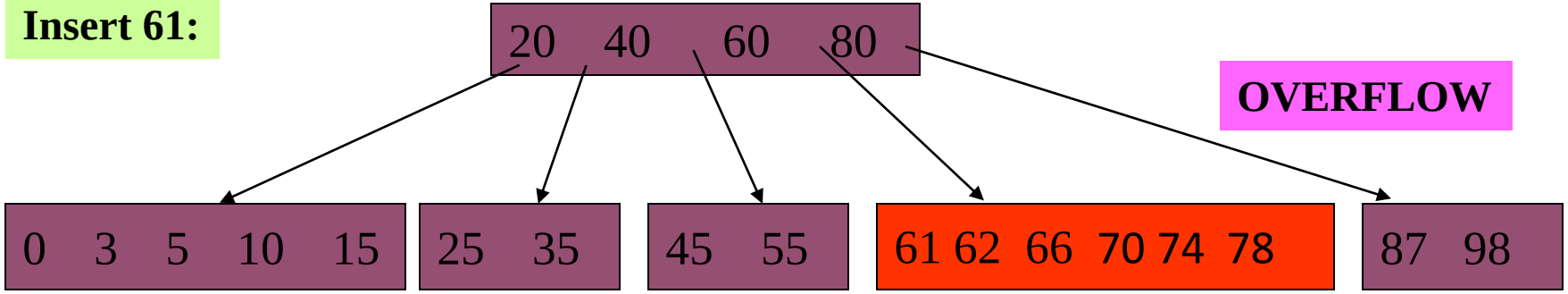
Insert 3:



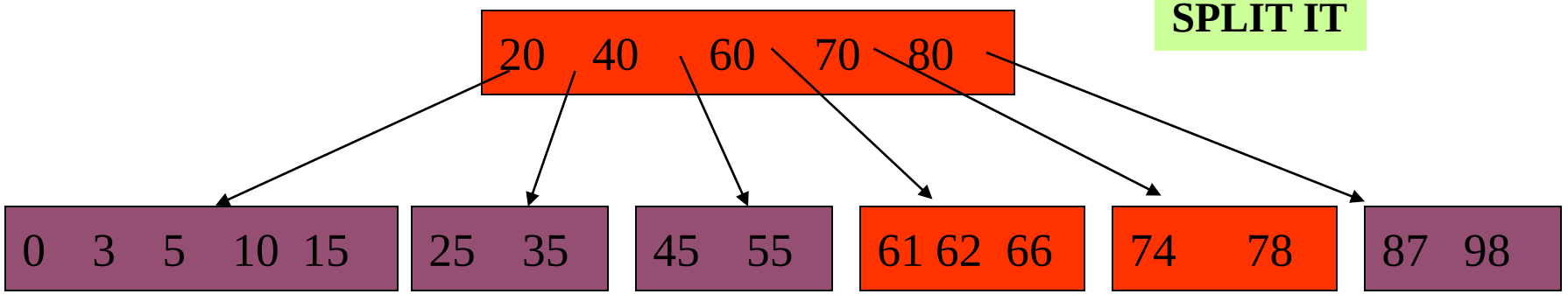


$M = 6;\ t = 3$

Insert 61:



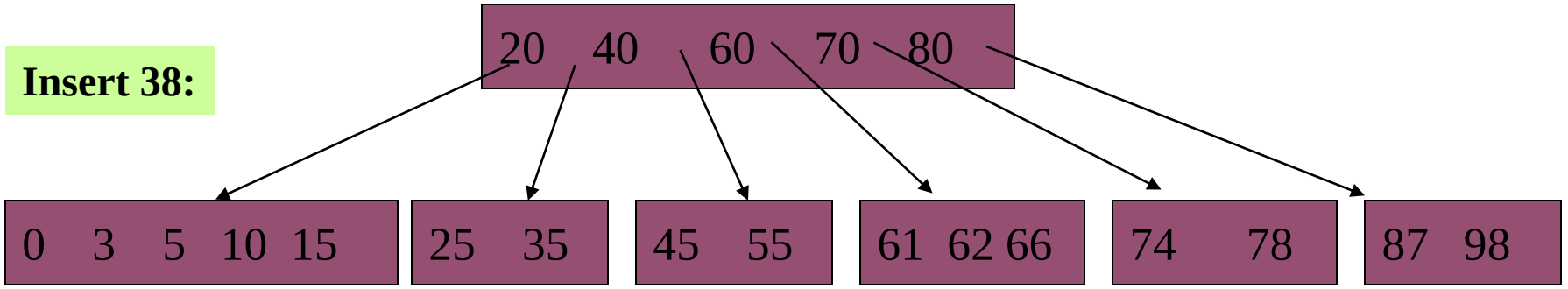
OVERFLOW

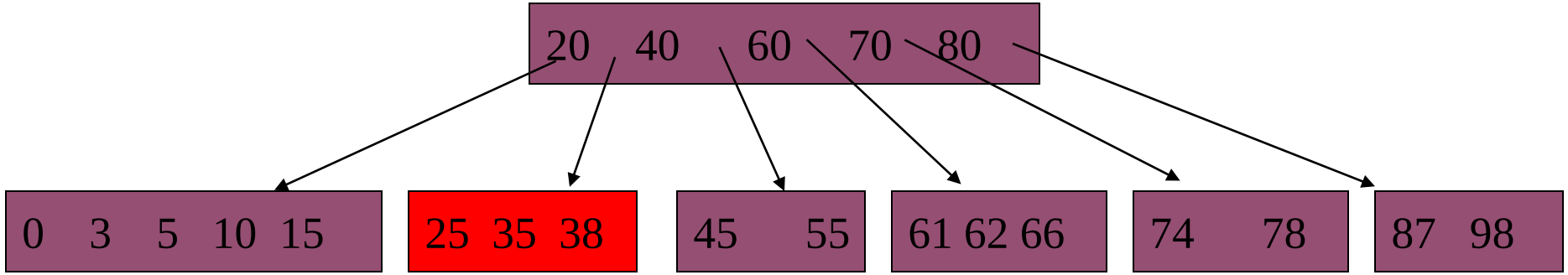


SPLIT IT

$M = 6;\ t = 3$

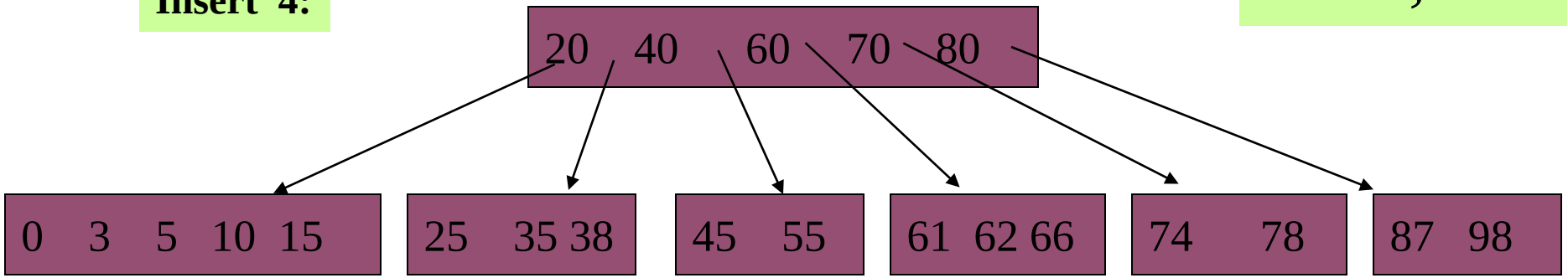
Insert 38:



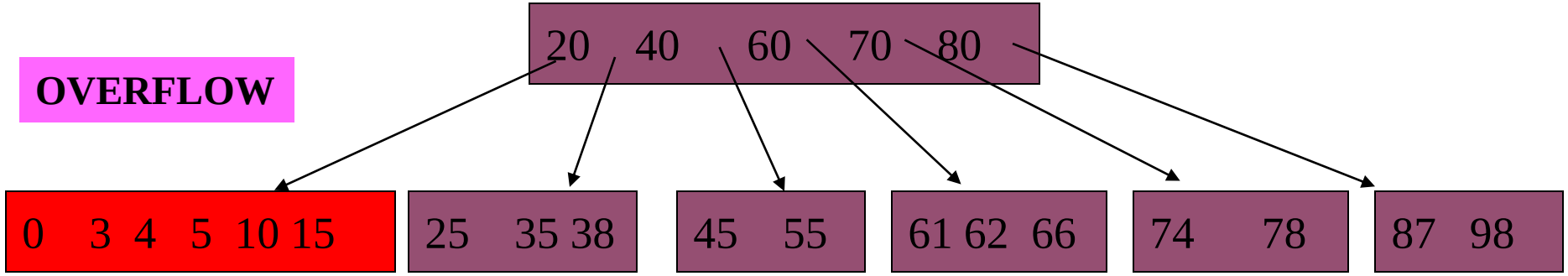


Insert 4:

$M = 6;\ t = 3$



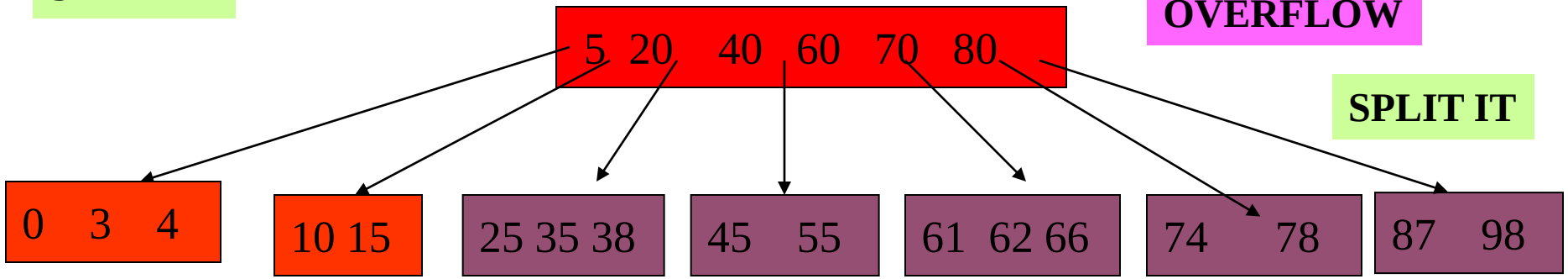
OVERFLOW



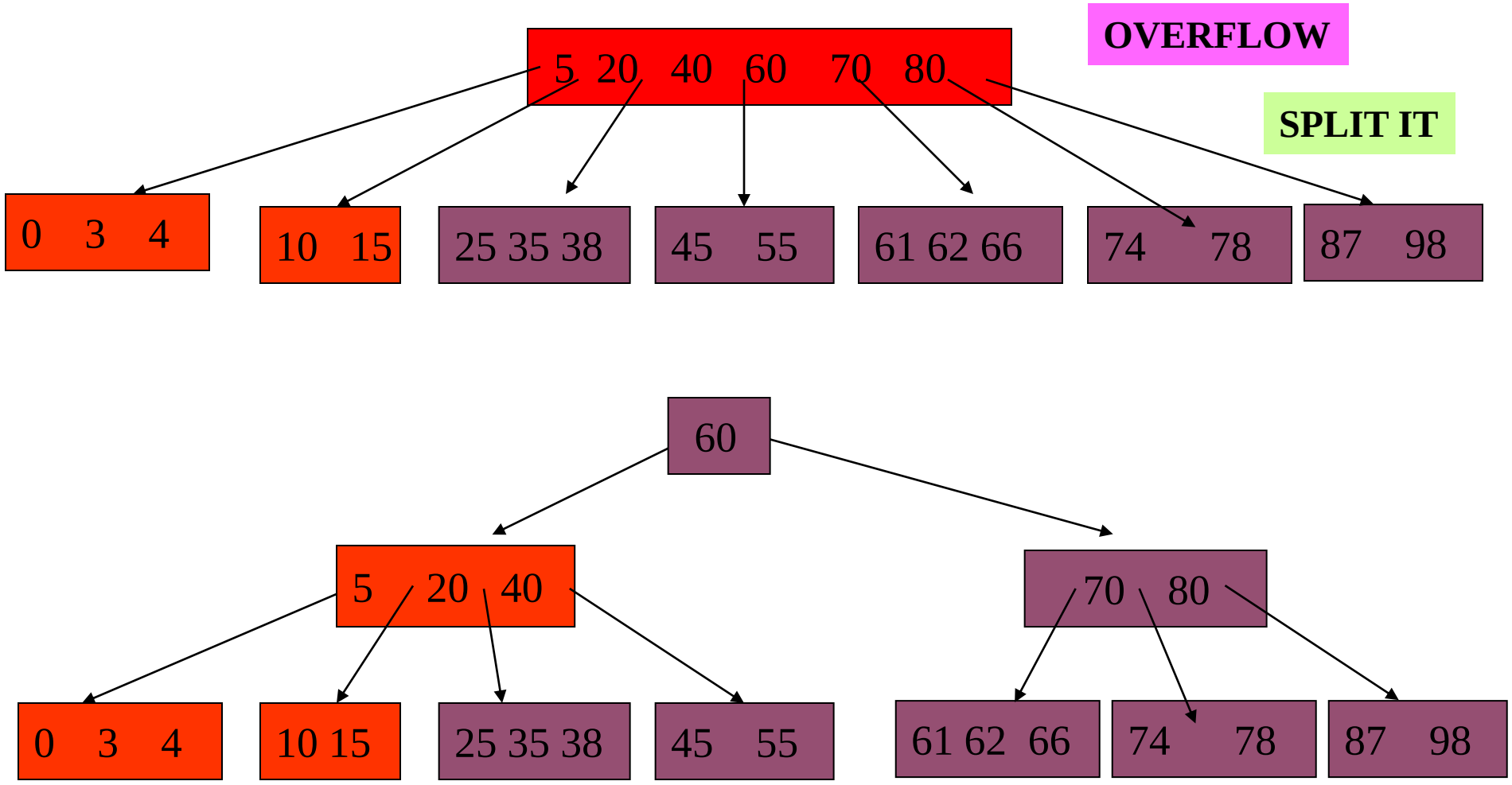
SPLIT IT

OVERFLOW

SPLIT IT



$M = 6; t = 3$


OVERFLOW
SPLIT IT
5 20 40 60 70 80
0 3 4
10 15
25 35 38
45 55
61 62 66
74 78
87 98
60
5 20 40
70 80
0 3 4
10 15
25 35 38
45 55
61 62 66
74 78
87 98

Time complexity

O(logn) – for Insertion, Deletion and Searching

Application:
B-tree and it’s variants used in databases

Other height balanced trees:
Red Black Tree, Splay Tree, 2–3 tree, 2–3–4 tree

Variants of B Tree:
B+ tree, B* tree