Data Management and File Organization

Indexing

B-Tree Operations

B+Trees

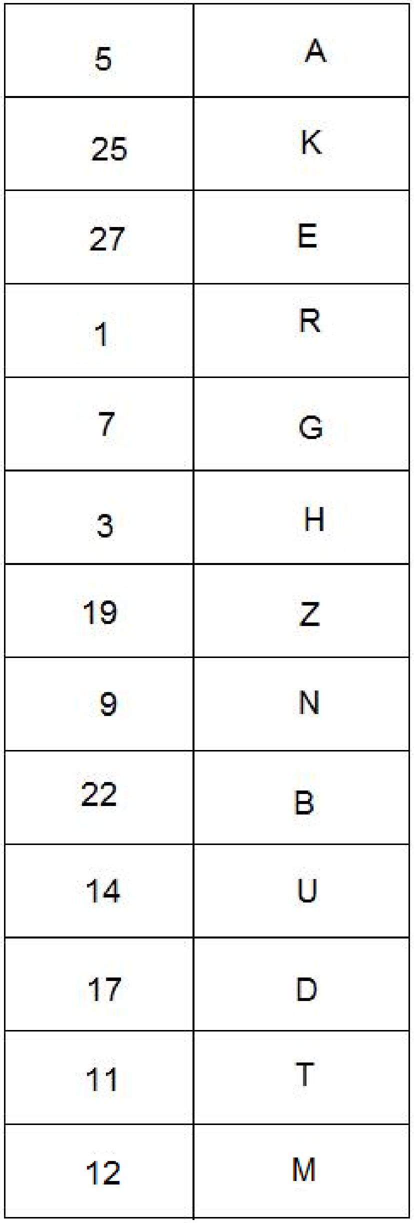
Topics

- Insertion in a B-Tree
- Deletion from a B-Tree
- B+Trees

Insertion in a B-Tree

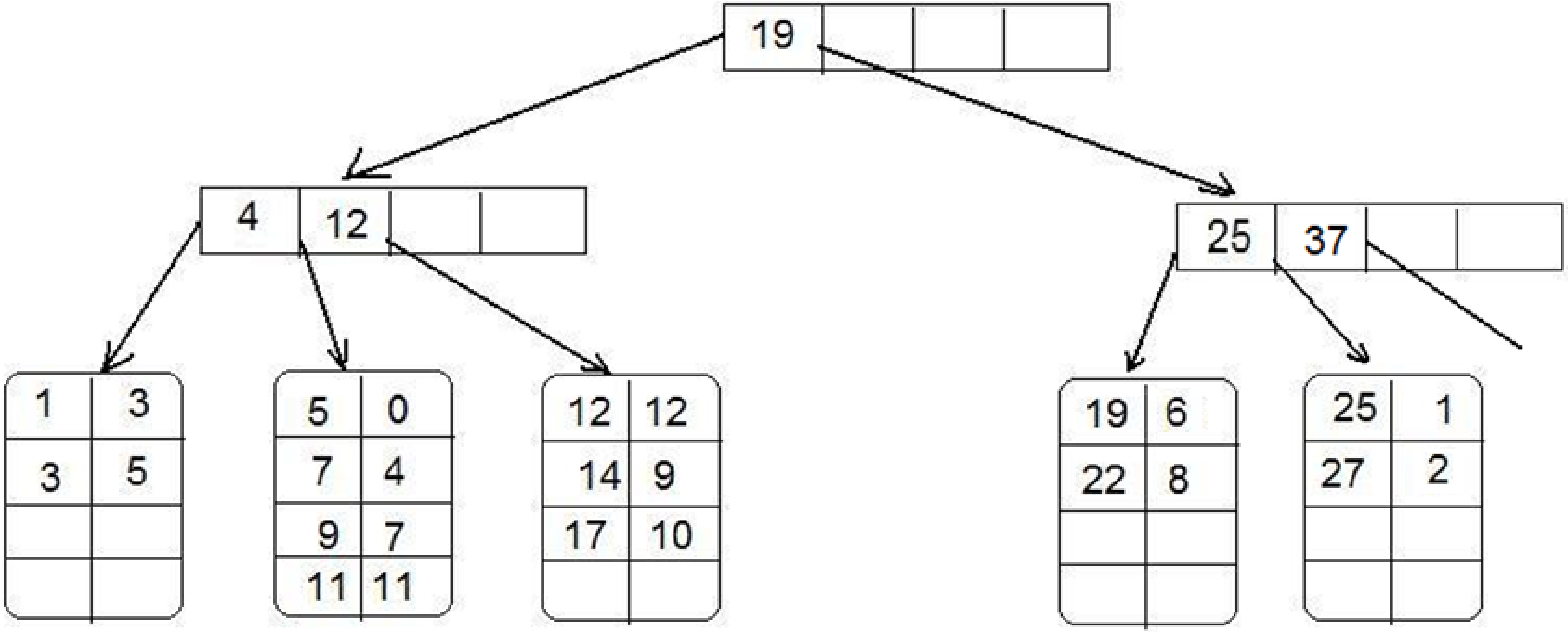
- Start searching the leaf node to insert the new record
- If the leaf node is full, split it into two nodes.
- Add the smallest key in the new leaf to the internal node.
- Update the tree if necessary

Sample Data File



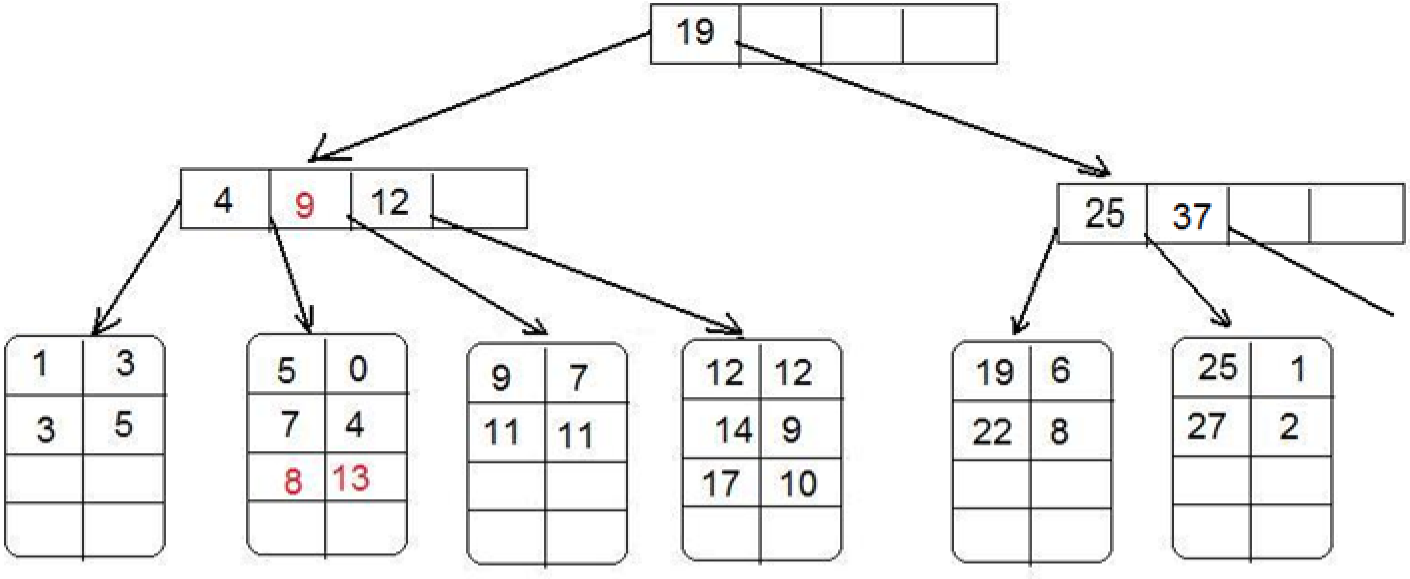
5	A
25	K
27	E
1	R
7	G
3	H
19	Z
9	N
22	B
14	U
17	D
11	T
12	M

B-Tree of the Sample Data File (N=2)

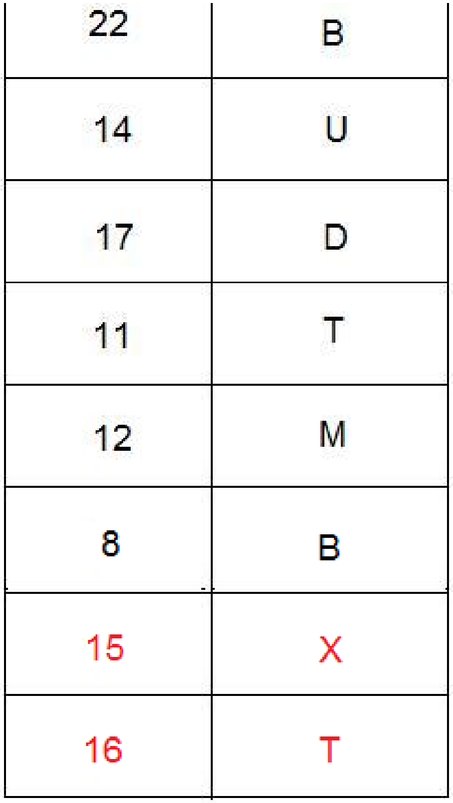


Insert <8, B>

5	A
25	K
27	E
1	R
7	G
3	H
19	Z
9	N
22	B
14	U
17	D
11	T
12	M
8	B

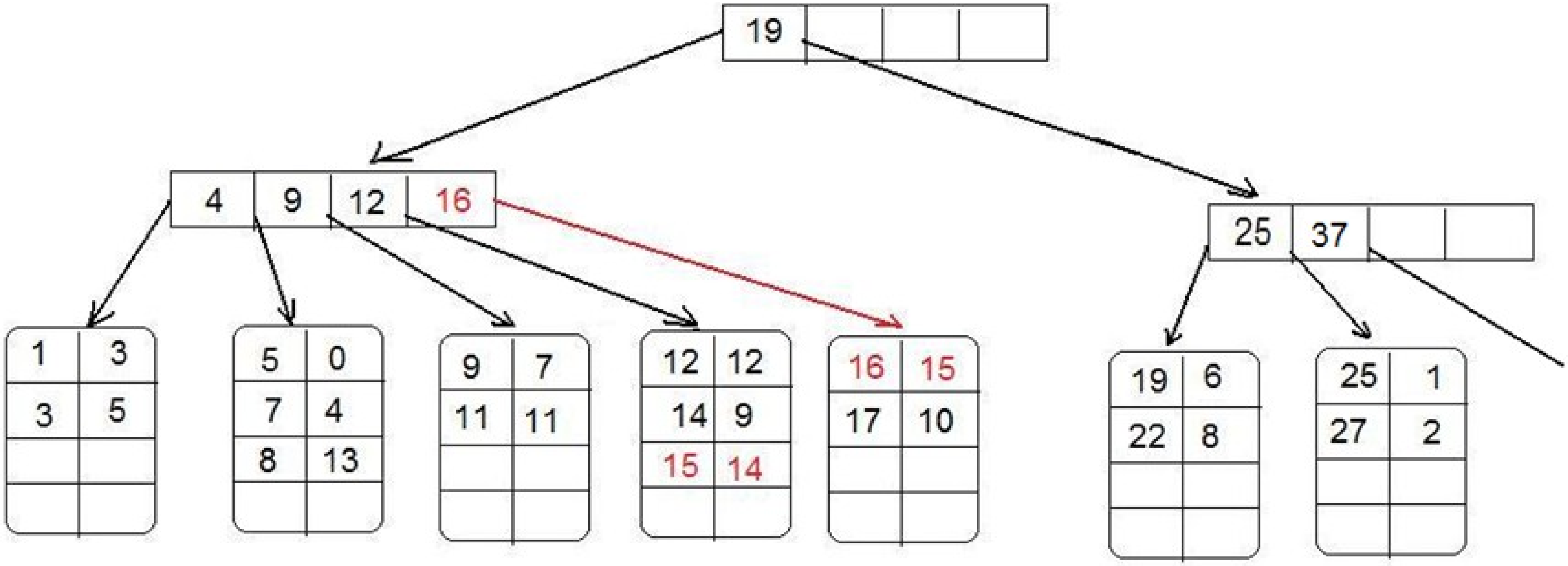


Insert <15, X> and <16, T>

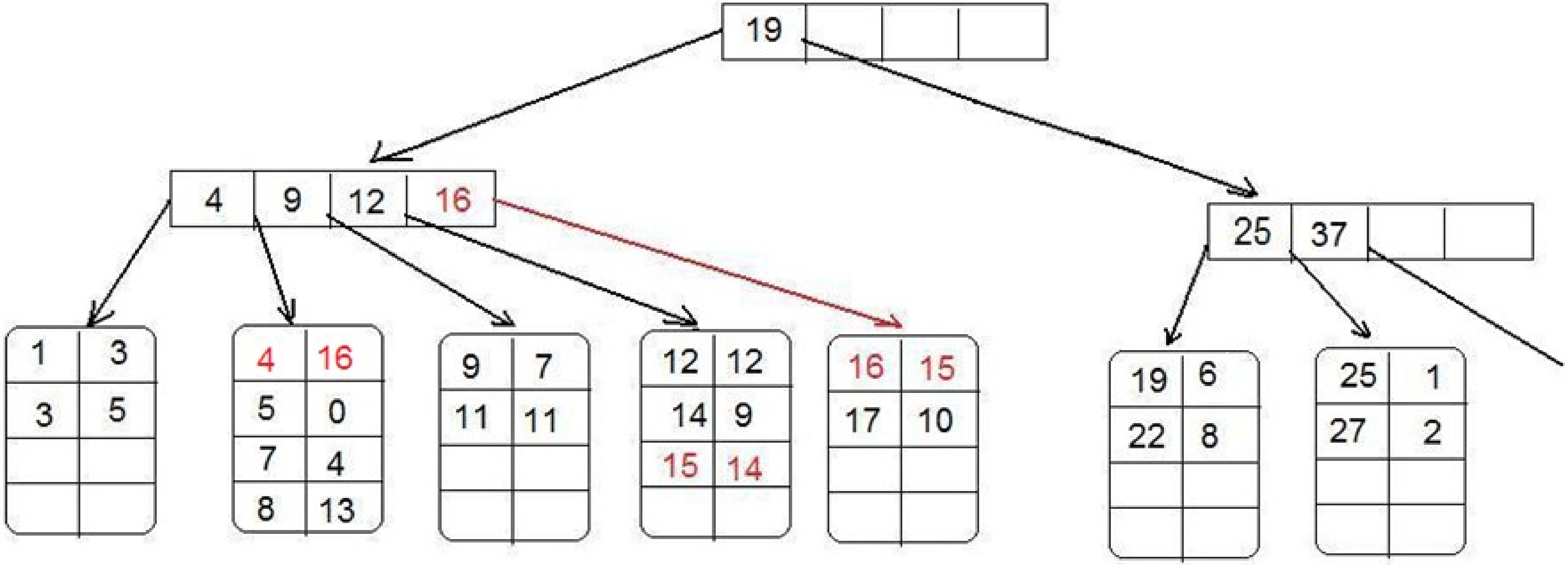


22	B
14	U
17	D
11	T
12	M
8	B
15	X
16	T

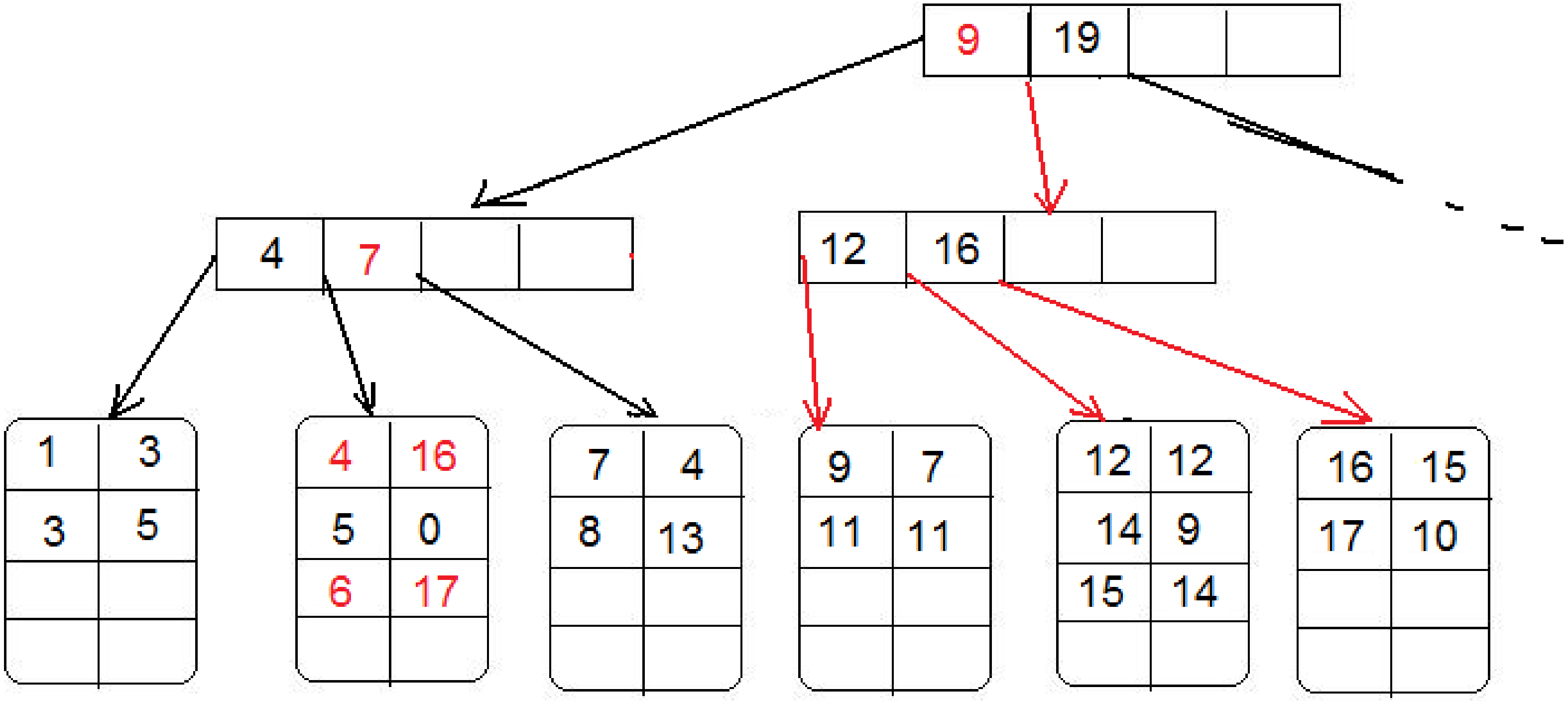
Insert <15, X> and <16, T>



Insert <4, Q>


19
4 | 9 | 12 | 16
25 | 37
1 3
3 5
4 16
5 0
7 4
8 13
9 7
11 11
12 12
14 9
15 14
16 15
17 10
19 6
22 8
25 1
27 2

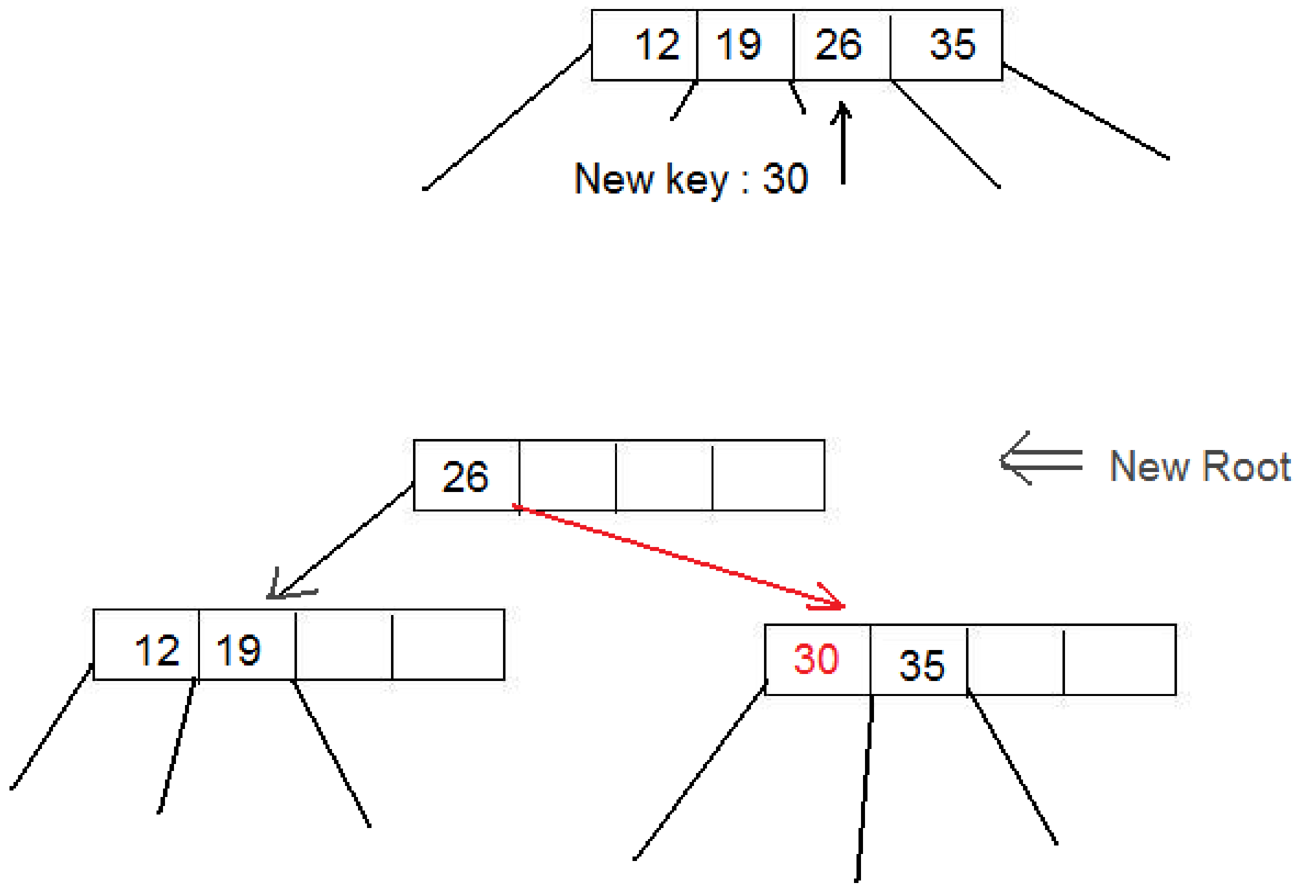
Insert <6, J>


9 | 19
4 | 7
12 | 16
1 3
3 5
4 16
5 0
6 17
7 4
8 13
9 7
11 11
12 12
14 9
15 14
16 15
17 10

Insertion when the root node is full

- If the root node is full and a new key is to be added to it,
  - split the root node into two nodes
  - Put last N keys in new node
  - Create a new node and put the middle key in it
  - Make the new node the new root. (old root and the new node split from it will be its children)

Split the Root



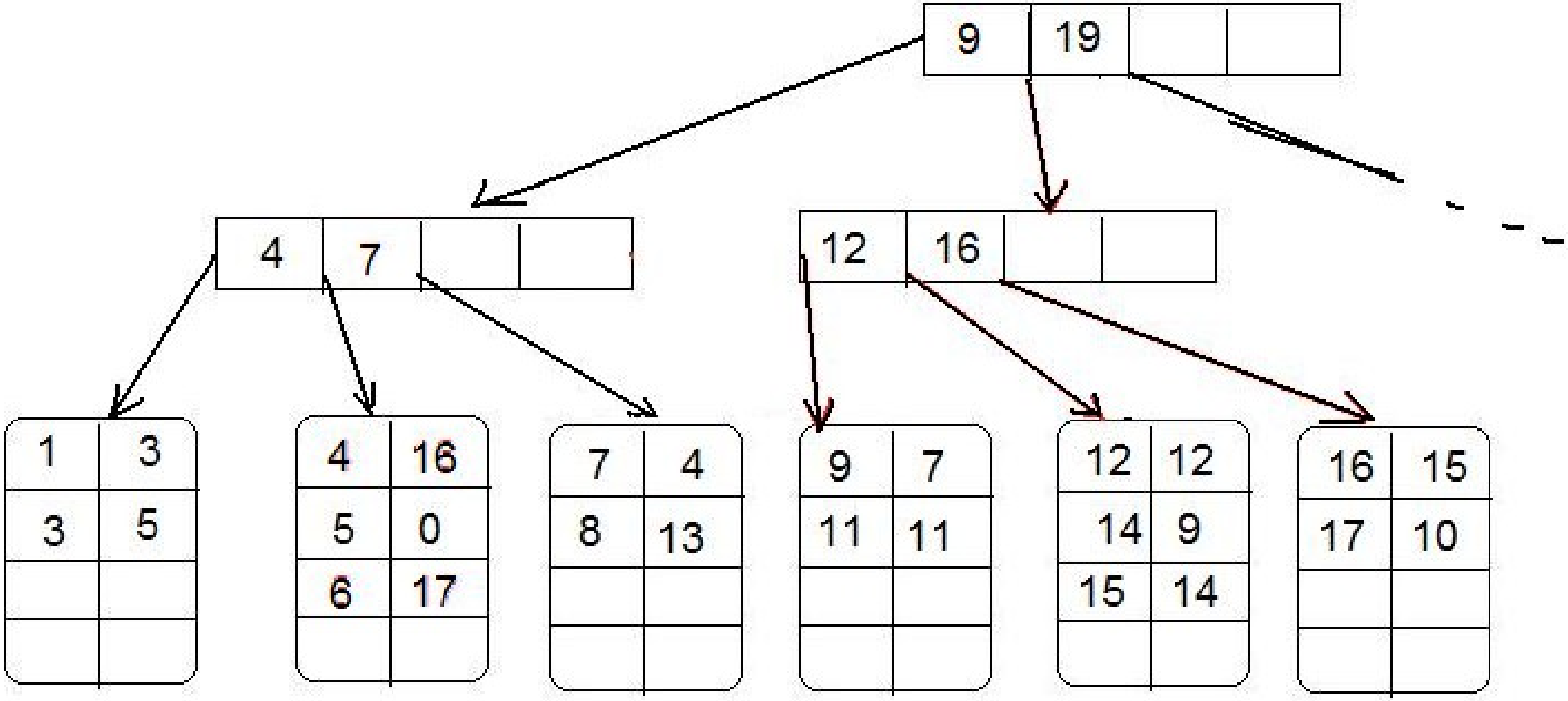
Deletion from a B-Tree

- When two leaf nodes are merged, a key is removed from the internal node.

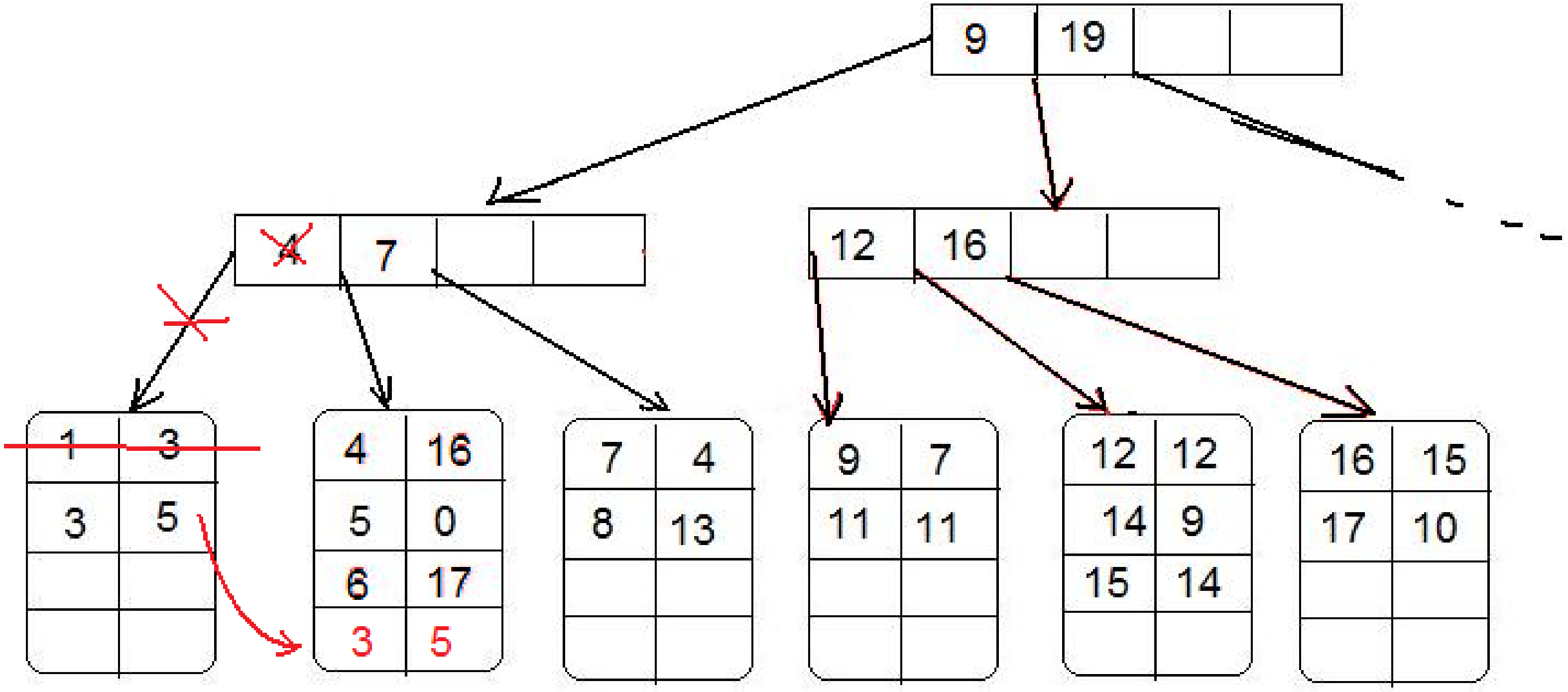
- If after removing a key, the internal node has less than N keys, it is merged with its neighboring internal node. (Except the root)

- When only one leaf node is left in the tree, the root is removed.

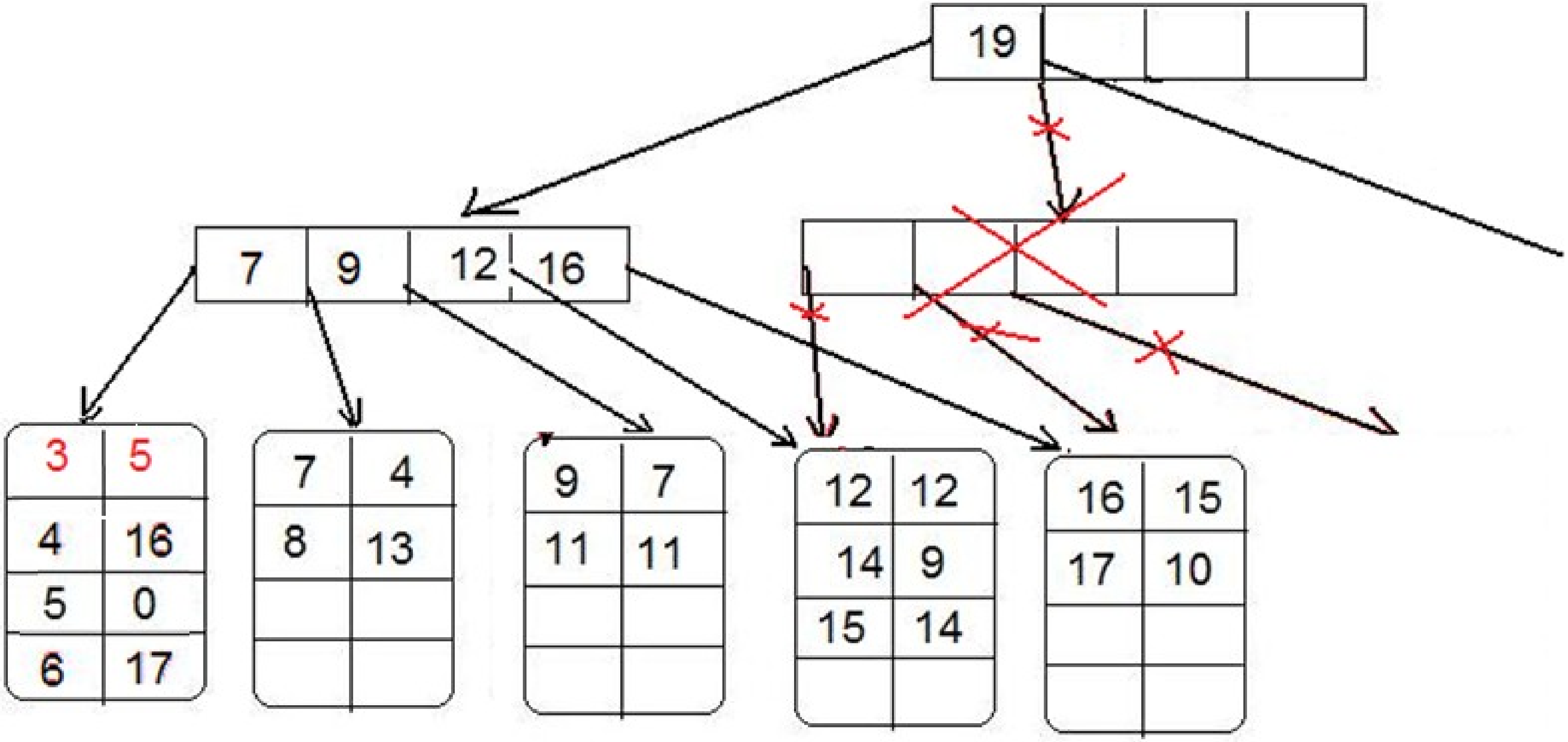
Example: Delete 1



Example, Delete 1



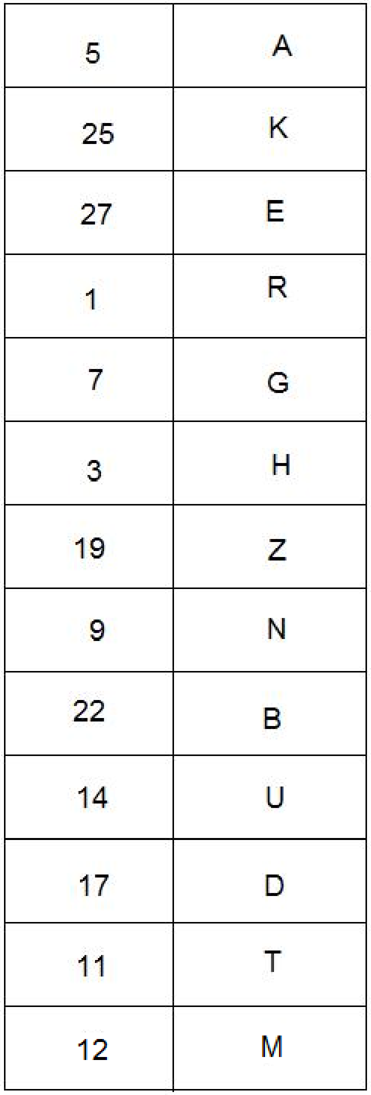
Example, Delete 1



B+Trees

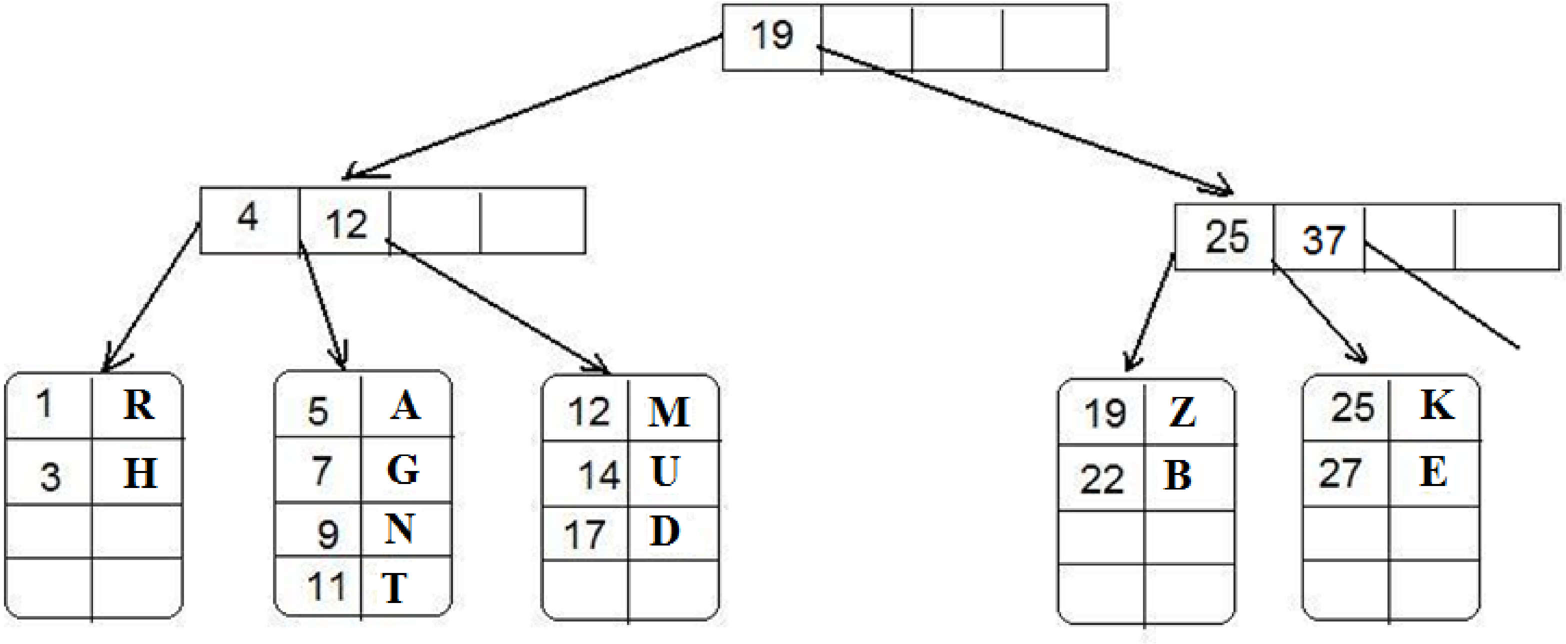
- B-Trees are used to find the location of a record in a data file
- The index and data files are two separate files
- B+Tree combines the data and index files in a single tree
- Leaf nodes are used to store data records

Sample Data



5	A
25	K
27	E
1	R
7	G
3	H
19	Z
9	N
22	B
14	U
17	D
11	T
12	M

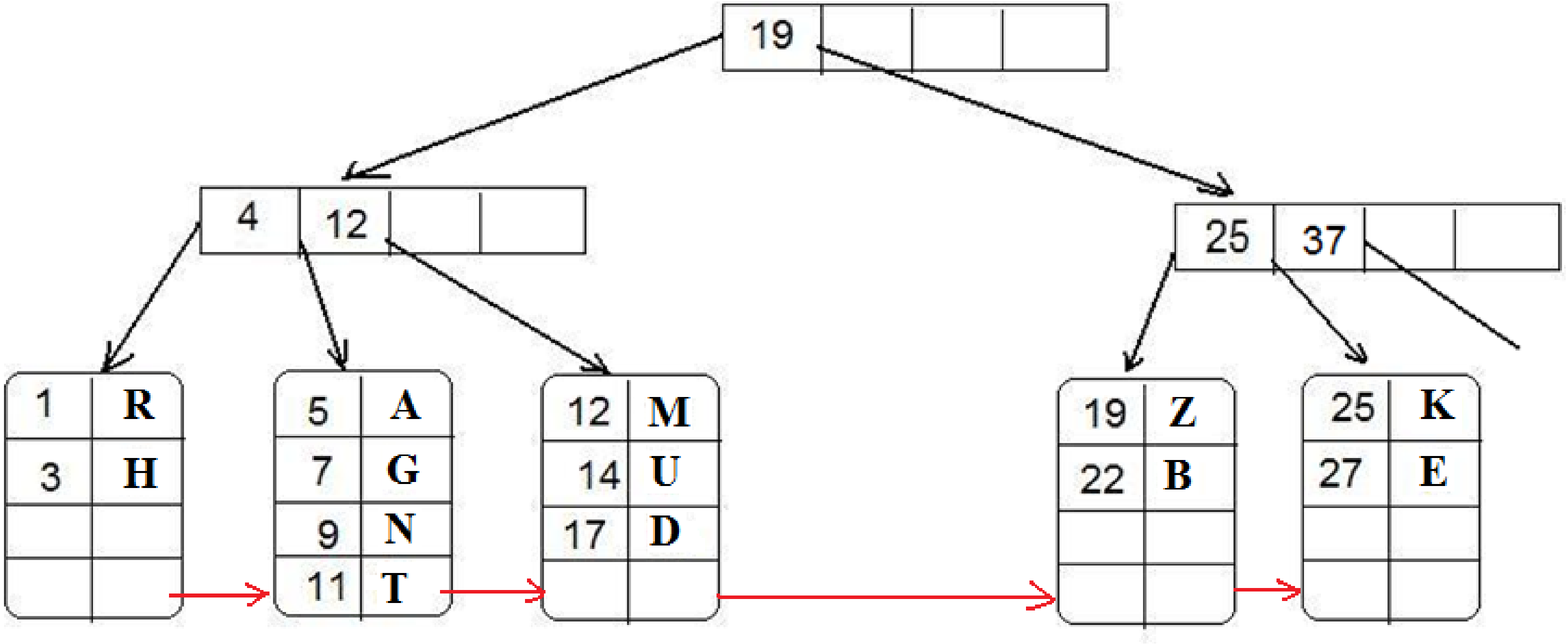
Sample B+Tree (N=2)



Exhaustive Reading in Index Files

- Exhaustive reading from a B-Tree needs starting from the root each time
- In a B+Tree leaf nodes are connected by pointers
- Exhaustive reading a B+Tree is as fast as exhaustive reading of a sorted file without overflow area

Exhaustive Reading



Improving Access Speed

- Motivation: The number of file access in an indexed file is as many as the tree height (3 or 4 for example)
- Hashing method provides a quick access to the records (1 or 2 file access)

Questions?

Quiz

- The following data has been given in a pile file. Create a B+Tree index for the data.
- Assume N=2 (4 keys, 5 pointers in each internal node)

Emp.ID	Name
118	Hasan
223	Mehmet
195	Emre
104	Hatice
102	Zeynep
113	Fatma
167	Tolga
142	Onur
136	Arda