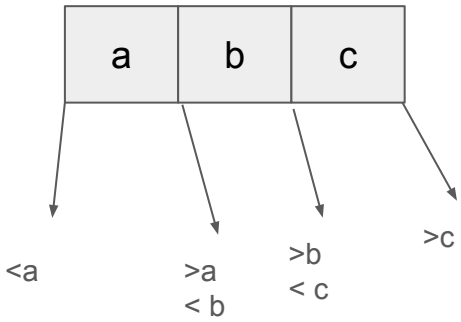
-most common indexing data structure used in relational databases systems
-optimized for disk-based indexing
-minimizing disk accesses for indexing

B+ Tree is an m-way with order M

- M -> maximum number of keys in each node
- M+1 -> max children of each node

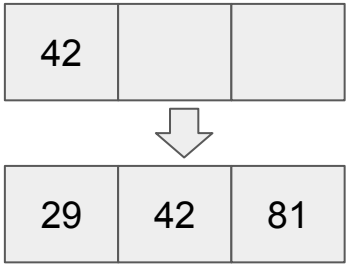
Node structure for m=3



B+ Tree Properties:

- All node (excpet root) must be ½ full min
- Root node doesn't have to be ½ full
- Insertions are always done at leaves
- Leaves are stored as a DLL
- Keys in nodes are kept sorted

Insert 42, 29, 81, …. Into M=3



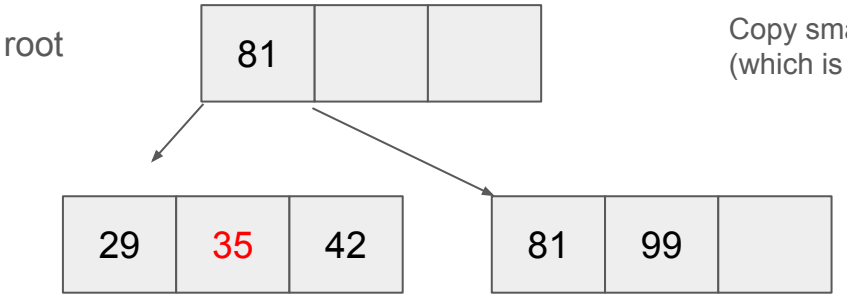
- Internal nodes
  - Store pointers to children and keys
  - Acts as the index
  - Just indexes relevant info -> pack keys so minimize depth
- Leaf nodes
  - Store keys and also data

Insert 99, 35, 2

Now two nodes and no way to get between them

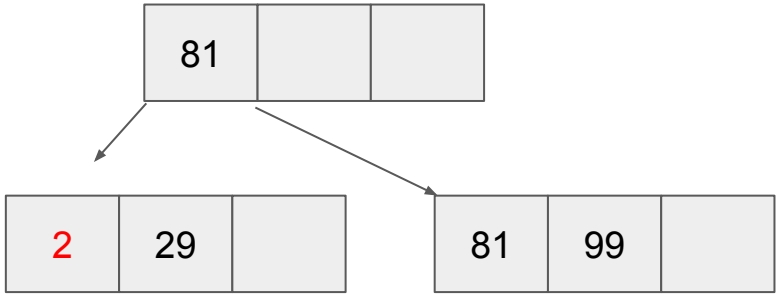
B+ Tree is extension of B tree, but in B tree internal nodes store keys and values

- B+ tree minimizes disk access (disk space indexing)
- B tree for in memory indexing



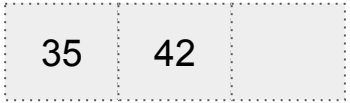
Copy smallest value of new node (which is 81)

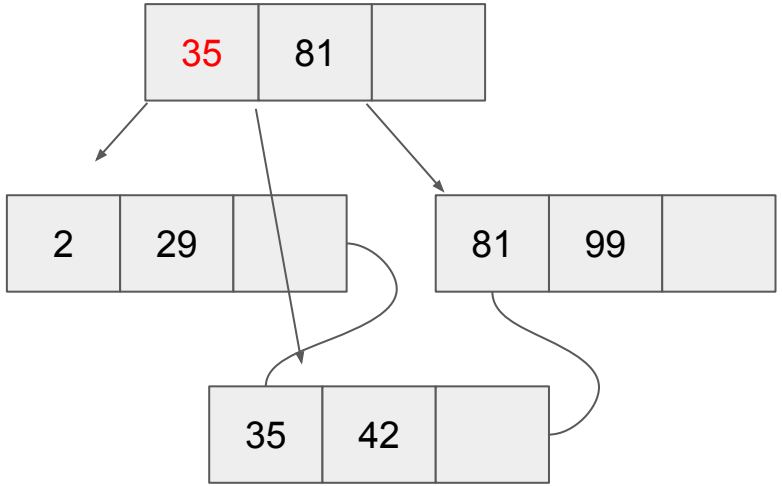
Insert 2



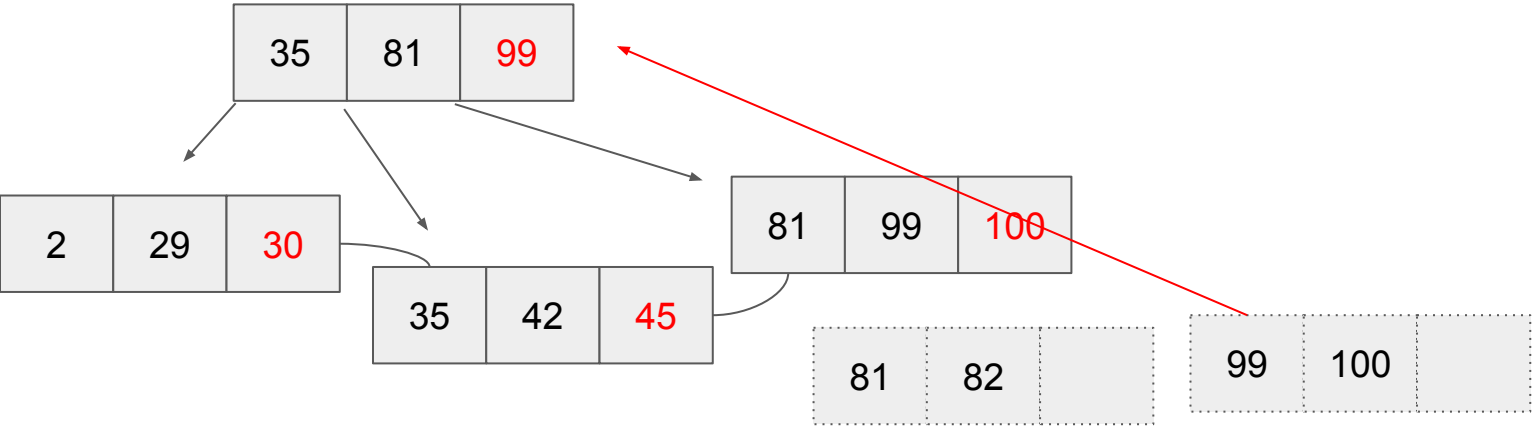
Operations to split nodes and insert is still faster than doing more disk accesses

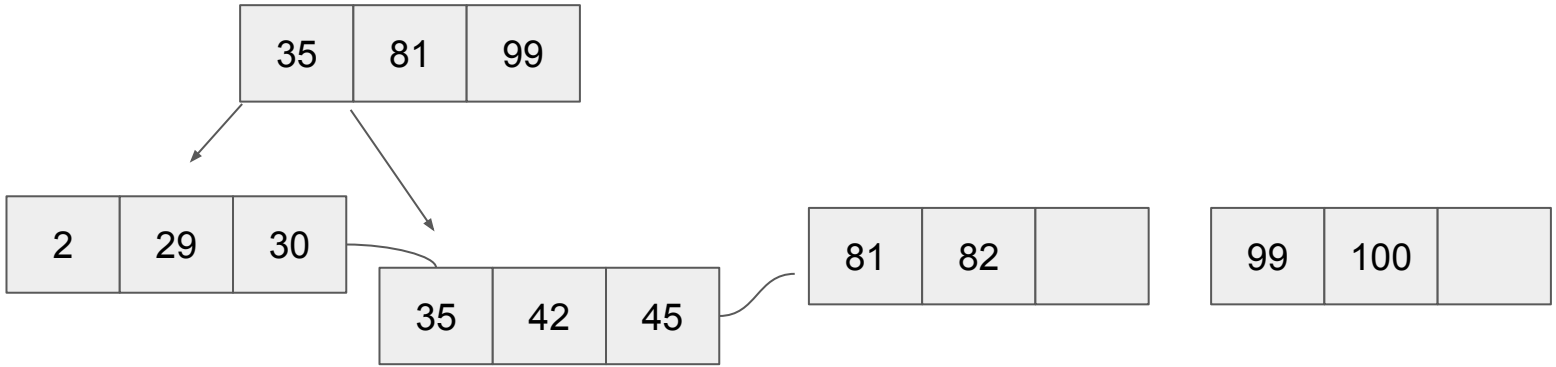
Split 2, 29, 35, 42 in half into diff nodes



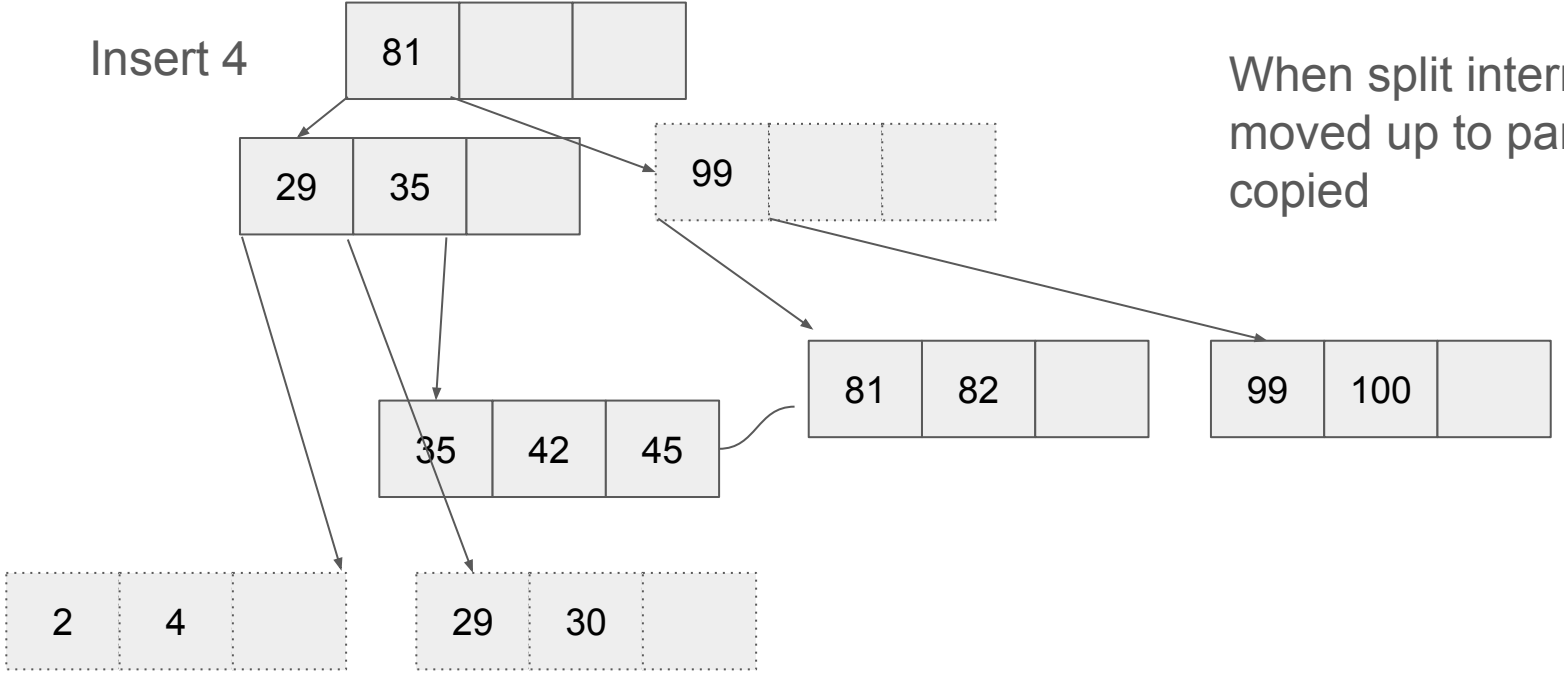


Insert: 100, 30, 45, 82





Insert 4



When split internal node, gets moved up to parent not copied