# B Tree

# Structure of B tree node

```
class node;
Struct pair
{
node *next;
int key;
}
class node;
{
node *first;
node *parent;
pair data[max];
int noofkeys;
}
```

# B tree Insertion

Key is always inserted into leaf node.

```
Step 1) If root is NULL then
    Acquire memory for new node
    Insert a pair ( x, Null) in the root node.
    Root-> insertnode(pair)

Step 2)If root is not NULL then
    Find current node to which x should be added.
    If leaf node has space to accommodate x, it is inserted.
    if ( p->noofkeys < mkeys)
    p->insertnode(pair)
```

# B tree Insertion

```
void insertnode( pair mypair)
    {
        int l;
    for( i=noofkeys-1; i>=0 && data[i].key> mypair.key ;i--)
        {
            data[i+1]= data[i];
        }
        data[i+1]=pair;
        noofkeys++;
    }
```

# B tree Deletion

Case 1)

1)If the value to be deleted doesn’t occur in leaf, we replace it with the smallest value in its right subtree and then proceed to delete the value from leaf node.

Deletion may cause underflow.

Minimum key in a node must be m-1/2

So borrow a key from its left sibling.

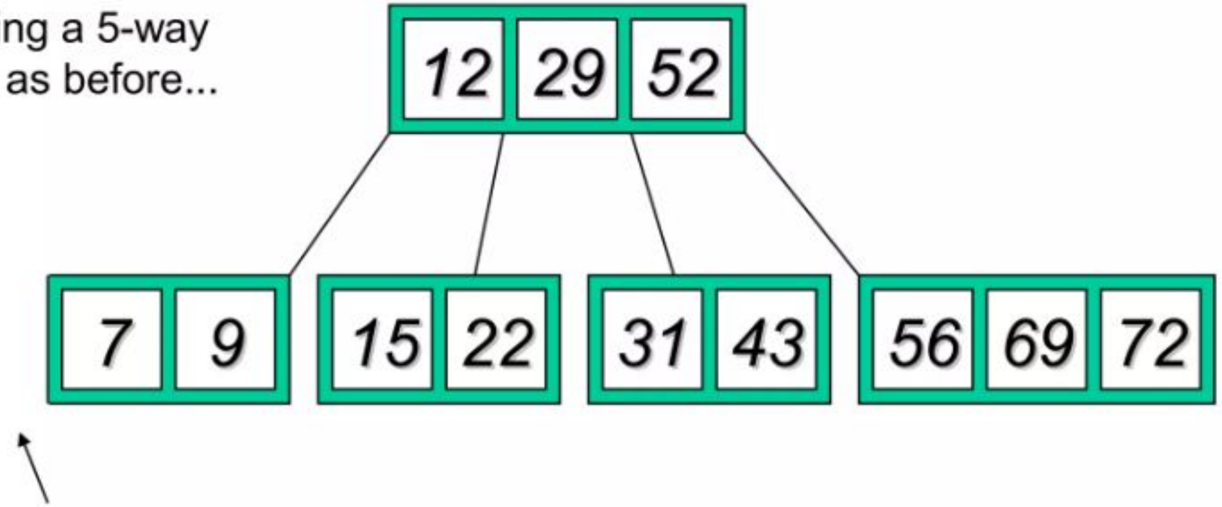
2)If left neighbour contains m-1/2 keys and right neighbour contains more than m-1/2 keys, it can borrow key from right neighbour.

# B tree Deletion

Case 2) if both left and right neighbour have same m-1/2 keys, then we have to join together the current node and its neighbour, either left or right, to form a combined node, we must include the value of parent node in combined node.

# Type #1: Simple leaf deletion

Assuming a 5-way B-Tree, as before...



Delete 2: Since there are enough keys in the node, just delete it

# Analysis Of B Tree

- For a B-Tree of order M

  1. Each internal node has up to M-1 keys to search

  2. Each internal node has between $\lceil M/2 \rceil$ and M children

  3. Depth of B-Tree storing N items is $O(\log_{\lceil M/2 \rceil} N)$

- Find: Run time is:

  1. O(log M) to binary search which branch to take at each node. But M is small compared to N.

  2. Total time to find an item is O(depth*log M) = O(log N)

# Thank You!!!!