Multiway Search Trees

Multiway search tree

- In a binary tree, each node contains a single key and may point to maximum two subtrees
- In a multiway search tree of order n, each node has **n or fewer subtrees** and contains one less key values than its subtrees.
- But, It is not compulsory that every node has exactly n-1 keys.
- In a multiway search tree, the size of node will be much larger than Binary search tree node but the total number of nodes will be less and also the height of the tree will be small.
- If a node has 4 subtrees, it shall contain three keys.
- Keys are arranged in ascending order.

Multiway search tree

- If s0, s1,… sm-1 are m subtrees of a node containing k0, k1 .. Km-2 keys in ascending order then all keys in s0 will be less than k0, all keys in s1 will be less than k1 and so on.
- Hence a node of the multiway search tree is like this:

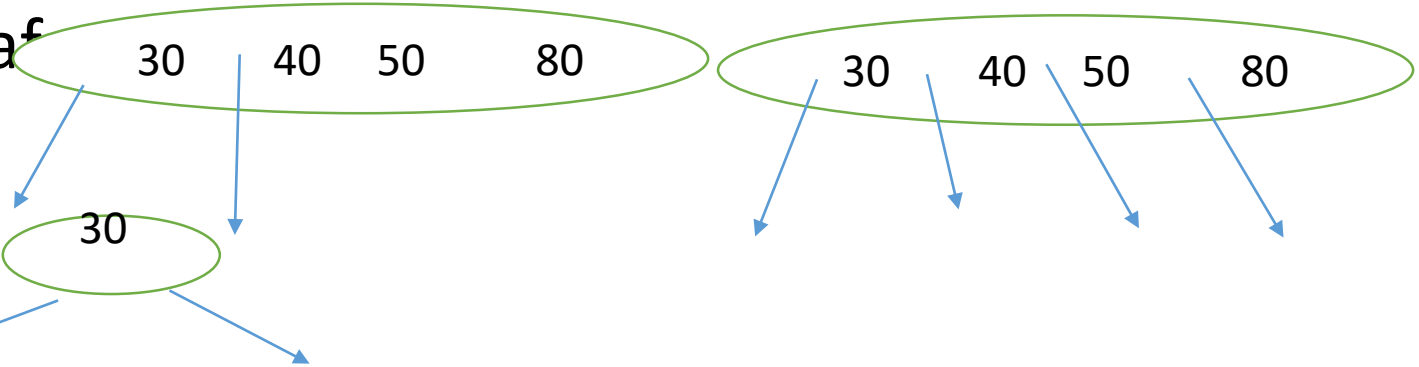
([] [10] [] [40] [] [65] [] [90] [])

- But we generally denote it as (10, 40, 65, 90)
- Multiway search trees can be built in many ways.
- We will define two types of multiway search trees: **Top down multiway search tree** and **Balanced multiway search tree**

Leaf and semi leaf

- A leaf is a node which has all the subtrees empty
- A semileaf is a node which has atleast one empty subtree
- A full node is a node which has maximum number of keys.
- Every leaf is a semi leaf.
- Leaf-----

- Examples of SemiLeaf



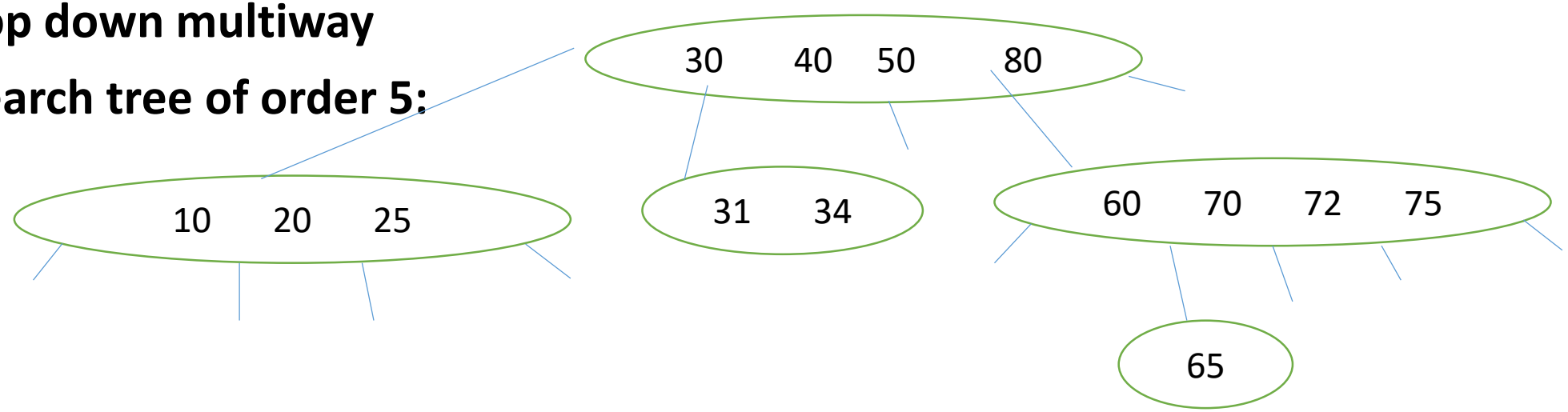
- Not semi leaf:

Top Down Multiway search tree

- A tree is called top down multiway search tree if it satisfies the condition that every non full node is a leaf.
- This means we create new subtrees of a node only when the node is full, This type of a tree can become biased

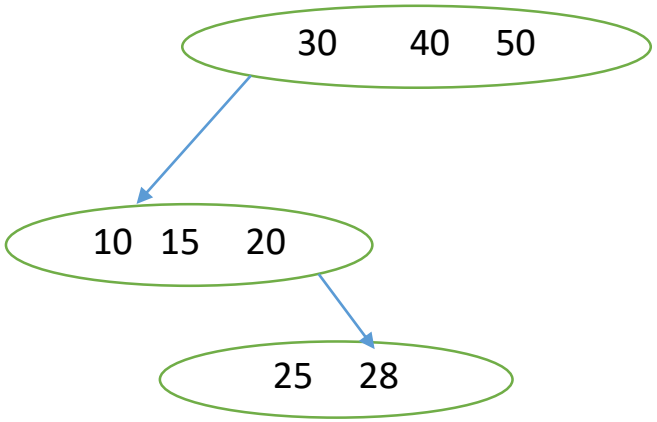
Top down multiway

search tree of order 5:



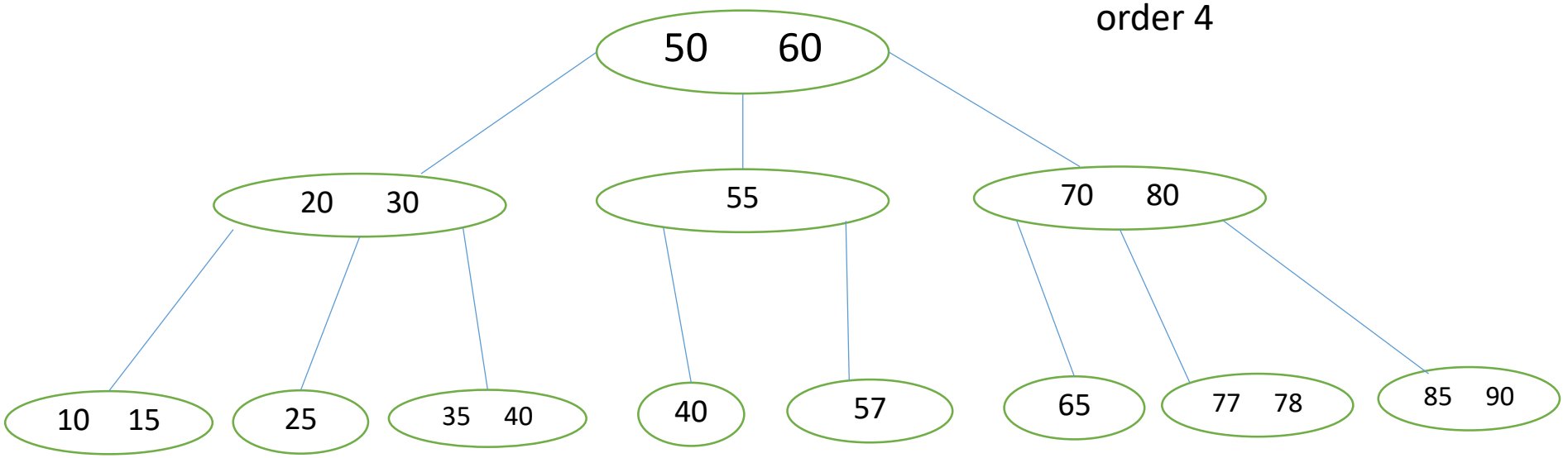
Balanced Multiway search tree

- A Multiway search tree is called **balanced multiway search tree** if all its semi leaves are at the same level.
- All semi leaves are leaves



Top down tree of order 4

Balanced tree of order 4



Pros and Cons of two types of trees

- ***Top down***:

1. Except leaf nodes all other nodes will be full
2. But height of the tree may be more(depending upon data) and tree may become quite imbalanced
3. Many subtrees may remain empty

Balanced:

1. Height of the tree will not be more and the tree will be balanced
2. Most of the nodes may remain partially full only.

Insertion in a top down multiway search tree

- Same as in Binary search tree
- Start searching from root node and find appropriate place for the new key
- If node is not full, insert the key in that node
- Otherwise create a new node and insert the key into that node and attach the node with the existing tree.

Structure of a multiway search tree node

A node of MST looks like this : ([] [10] [] [40] [] [65] [] [90] [])

But actually while implementing it consists of : (([][ ] [ ])

1. A count to keep track of number of data items
2. Array of data items to be stored
3. Array of subtrees

If size is n and data items are of integer type then:

Count of existing subtrees: numtrees

Array of integers – key[n-1]

Array of subtrees--- son[n]

Commonly used structure of a multiway search tree node

1. A count to keep track of number of data items
2. Array of data items to be stored ***
3. Array of subtrees

Most of the time the array of data elements is not a simple array of integers, it will be an array of some structures which will contain the key value and the address to the actual record

struct student { int rollno; char name [50; char branch[20]; int year;}

struct node_data{ int rollno; struct student *s;}

1. A count to keep track of number of data items
2. Array of node_data items
3. Array of subtrees

Structure of a multiway search tree node

- Struct multiwaytree{ int numtree;
- int key[n-1];
- struct multiwaytree * son[n];
- }

Operations on MST:

Insertion: Requires search and then space

Deletion: Requires search and then free space

Searching: Two steps: 1. find the node which may contain the key and then 2. search in the key

Searching a multiway search tree

```
p = tree
 if(p== null){ position =-1;
                    return -1;}
i = nodesearch(p, key)
If (i<numtree(p)-1 && key = k(p,i))
{ position = i;
return(p);
}
return(search(son(p,i)))
```

Deletion in Multiway search tree

1. Just mark the key as deleted without deleting
2. Actually delete the key maintaining the properties of top down multiway search tree
3. The simplest way to delete a record from MST is to retain the key in the tree and mark it as deleted record.

In this case key remains in the node and works as a guide post to subtrees but does not represent a record in the file

Deletion in Multiway search tree

This can be done in the following ways:

- Setting pointer to record to null

- Allocating an extra field ‘flag’ indicating deleted or not deleted record

Disadvantage: Space occupied by the key is wasted (specially when large number of records are deleted), Marking requires extra space

Can we utilize the space by deleted records

1. If leaf node contains the deleted node, at the time of inserting a new key, this can be replaced by a neighbouring key

2. In case of non leaf node, if a record with the same key value is inserted, the deleted record can be replaced

2. Actually deleting the record

- If key to be deleted has an empty right and left subtree, simply delete the key and compact the node and ***if it is the only key then free the node***

- If the key to be deleted has non empty left and right subtree, find its successor key(which must have an empty left subtree) and then successor key will take its place and compact the node containing successor key.

Cont..

- But this procedure may result in a tree that does not satisfy the requirements for a top down Multiway search tree

- To make it top down Multiway search tree, we may have to move the largest or smallest key of its non empty subtree

Delete 50..

- Example..