**AVL Trees :**

* (The name comes from the two inventors of this method: G.M. Adel’son-Velskii and Y.M. Landis. )
* PERFORMANCE OF BINARY SEARCH TREES
* AVL trees are height-balanced binary search trees
* Balance factor of a node

height(left subtree) - height(right subtree)

* An AVL tree has balance factor calculated at every node
* For every node, heights of left and right subtree can differ by no more than 1
* Store current heights in each node



* If after an Insertion or deletion the tree goes unbalanced, perform rotations to balance tree, in such a way that in-order traversal of tree should not change.

LeftRotation(p)

q = right( p);

hold = left( q );

left ( q ) = p;

right ( p ) = hold;

RightRotation(p)

q = left( p);

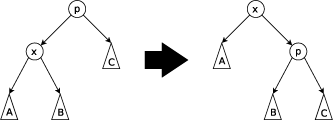
hold = right( q );

right ( q ) = p;

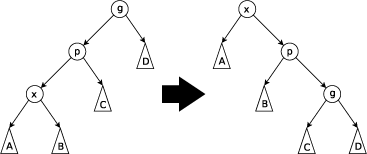
left ( p ) = hold;

**Splay Trees :**

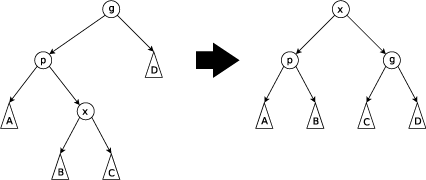
* A splay tree is a self-balancing binary search tree with the additional property that recently accessed elements are quick to access again.
* All normal operations on a binary search tree are combined with one basic operation, called splaying.
* Splaying the tree for a certain element rearranges the tree so that the element is placed at the root of the tree.
* One way to do this is to first perform a standard binary tree search for the element in question, and then use tree rotations in a specific fashion to bring the element to the top.
* Let ‘x’ be the node to be splayed, ‘p’ be its parent and ‘g’ be its grandparent.
* The three types of splay steps are:
* **Zig Step**: This step is done when p is the root. The tree is rotated on the edge between x and p.



* **Zig-zig Step**: This step is done when p is not the root and x and p are either both right children or are both left children.



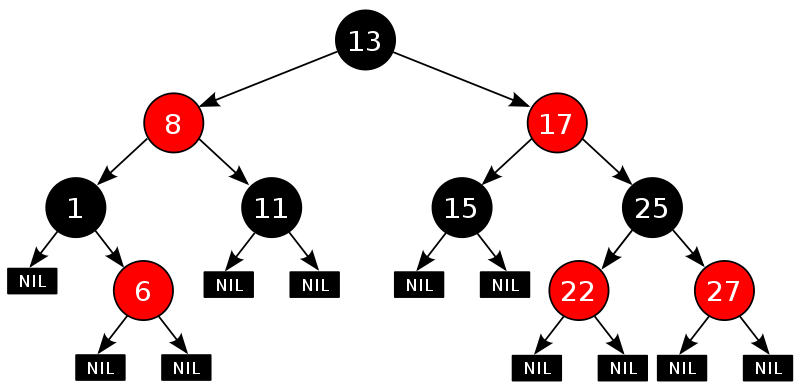
* **Zig-zag Step**: This step is done when p is not the root and x is a right child and p is a left child or vice versa.



**Red – Black Trees**

* Are binary search trees
* inserts and removes intelligently, to ensure the tree is reasonably balanced.
* used in computer science to organize pieces of comparable data, such as text fragments or numbers.
* the leaf nodes are not relevant and do not contain data
* A red-black tree is a binary search tree where each node has a color attribute, the value of which is either red or black.
* In addition to the ordinary requirements imposed on binary search trees, the following additional requirements apply to red-black trees:

1. A node is either red or black.
2. The root is black.
3. All leaves are black.
4. Both children of every red node are black.
5. Every simple path from a given node to any of its descendant leaves contains the same number of black nodes.

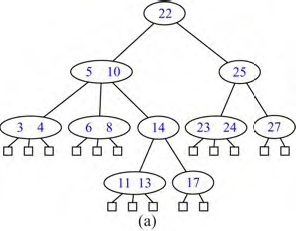


* These constraints enforce a critical property of red-black trees: that the longest path from the root to any leaf is no more than twice as long as the shortest path from the root to any other leaf in that tree.
* The result is that the tree is roughly balanced.

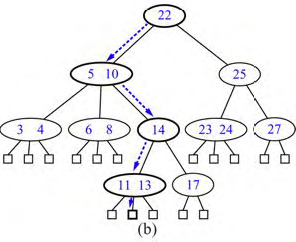
**Multi-Way Search Trees**

* Each internal node of T has at least two children

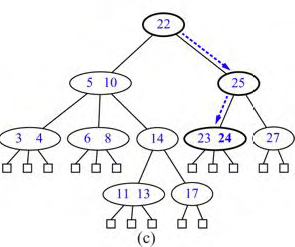
(a) A multi-way search tree T;



(b) search path in T for key 12 (unsuccessful search);



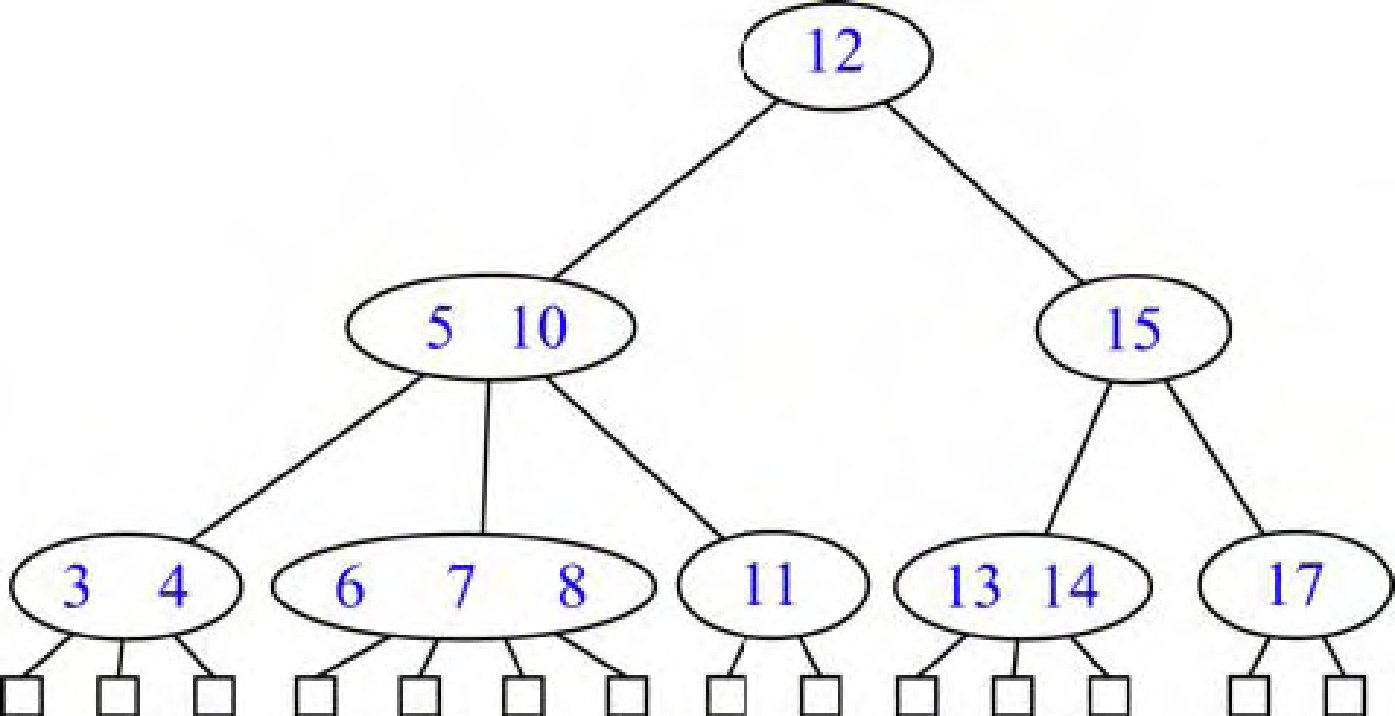
(c) search path in T for key 24 (successful search).



**2-3-4 Trees**

* Is a balanced multi-way tree
* Achieves this goal by maintaining following properties
  + ***Size Property:*** Every internal node has at most four children.
  + ***Depth Property:*** All the external nodes have the same depth.

***e.g.***

* 

**External Searching**

Consider the problem of implementing a dictionary for a large collection of items that do not fit in main memory.

* Since one of the main uses of large dictionaries is in databases, we refer to the secondary-memory blocks as disk blocks.
* Likewise, we refer to the transfer of a block between secondary memory and primary memory as a disk transfer.
* Recalling the great time difference that exists between main memory accesses and disk accesses, the main goal of maintaining a dictionary in external memory is to minimize the number of disk transfers needed to perform a query or update.
* In fact, the difference in speed between disk and internal memory is so great that we should be willing to perform a considerable number of internal memory accesses if they allow us to avoid a few disk transfers.

**An (a,b) Tree**

Definition : An (*a,b*) ***tree***, where *a* and *b* are integers, such that 2 ≤ *a* ≤ (*b* + 1)/2, is a multiway search tree *T* with the following additional restrictions:

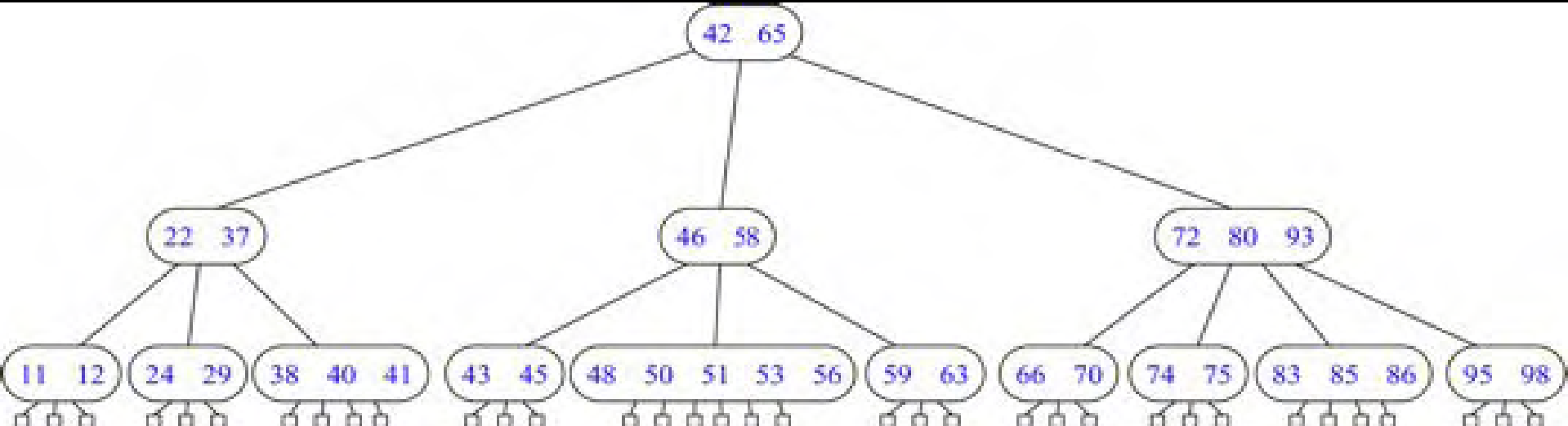
***Size Property:*** Each internal node has at least *a* children, unless it is the root, and has at most *b* children.

***Depth Property:*** All the external nodes have the same depth.

**B-Trees**

A version of the (*a*, *b*) tree data structure, which is the best known method for maintaining a dictionary in external memory, is called the "B-tree."

A ***B-tree of order*** *d* is an (*a*, *b*) tree with *a* = *d*/2 and *b* = *d*.



An important property of B-trees is that we can choose d so that the d children references and the d − 1 keys stored at a node can all fit into a single disk block, implying that d is proportional to B.