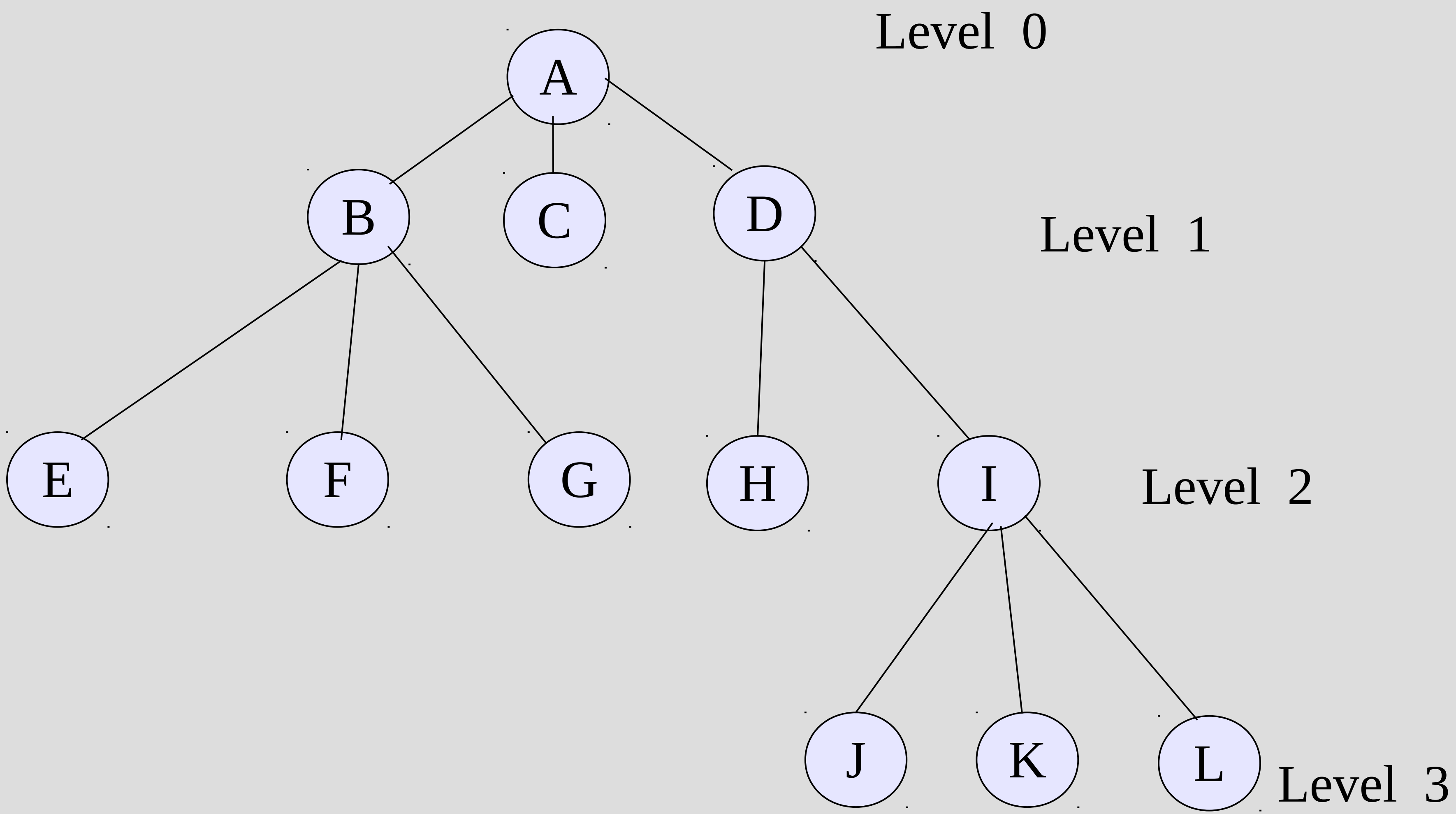
TREES

- A tree is a non linear data structure
- It is a finite set of one or more data items such that

  There is a special data item called the root of the tree

  And its remaining data items are partitioned into number of mutually exclusive subsets, each of which is itself a tree. And they are called subtrees

TREES



TREES

- ROOT

  It is a specially designed data item in a tree. It is the first in the hierarchical arrangement of data items. In the above tree, A is the root item
- NODE

  Each data item in a tree is called a node. It specifies the data information and links to other data items
- Degree of a node

  It is the number of subtrees for a node in a given tree

The degree of node A is 3

The degree of node D is 2

TREES

- <u>Degree of a tree</u>

  It is the maximum degree of nodes in a given tree. So, the degree of the above tree is 3
- <u>Terminal nodes</u>

  A node with degree zero is called a terminal node or a leaf
- <u>Non terminal nodes</u>

  Any node whose degree is not zero is called non terminal node
- <u>Siblings</u>

  The children nodes of a given parent node are called siblings. H & I are siblings of node D

TREES

- Path

  It is a sequence of consecutive edges from the source node to the destination node. In the above tree, the path between A & J is given by (AD), (DI) and (IJ)

- Depth

  It is the maximum level of any node in a given tree. This equals the length of the longest path from the root to any leaf

TREES

- Binary Trees

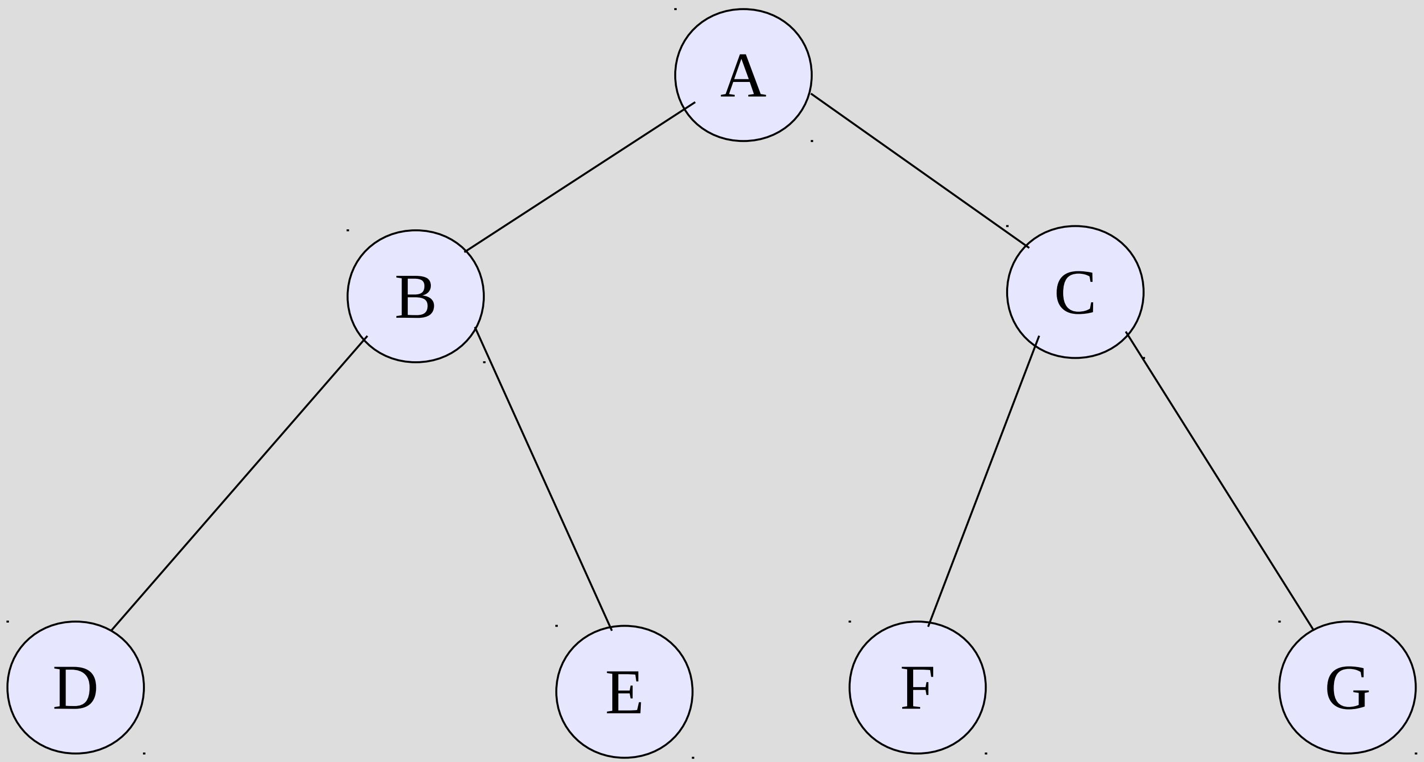
  A binary tree is a finite set of elements that is either empty or partitioned into three disjoint subsets.

  The first subset contains a single element called the root of the tree.

  The other two subsets are themselves binary trees called the left and right subtrees of the original tree. A left or right subtree can be empty

- In a binary tree the maximum degree of any node is at most two

BINARY TREES



BINARY TREES

- If A is the root of a binary tree and B is the root of its left or right subtree, then A is said to be the father of B, and B is said to be the left or right son of A

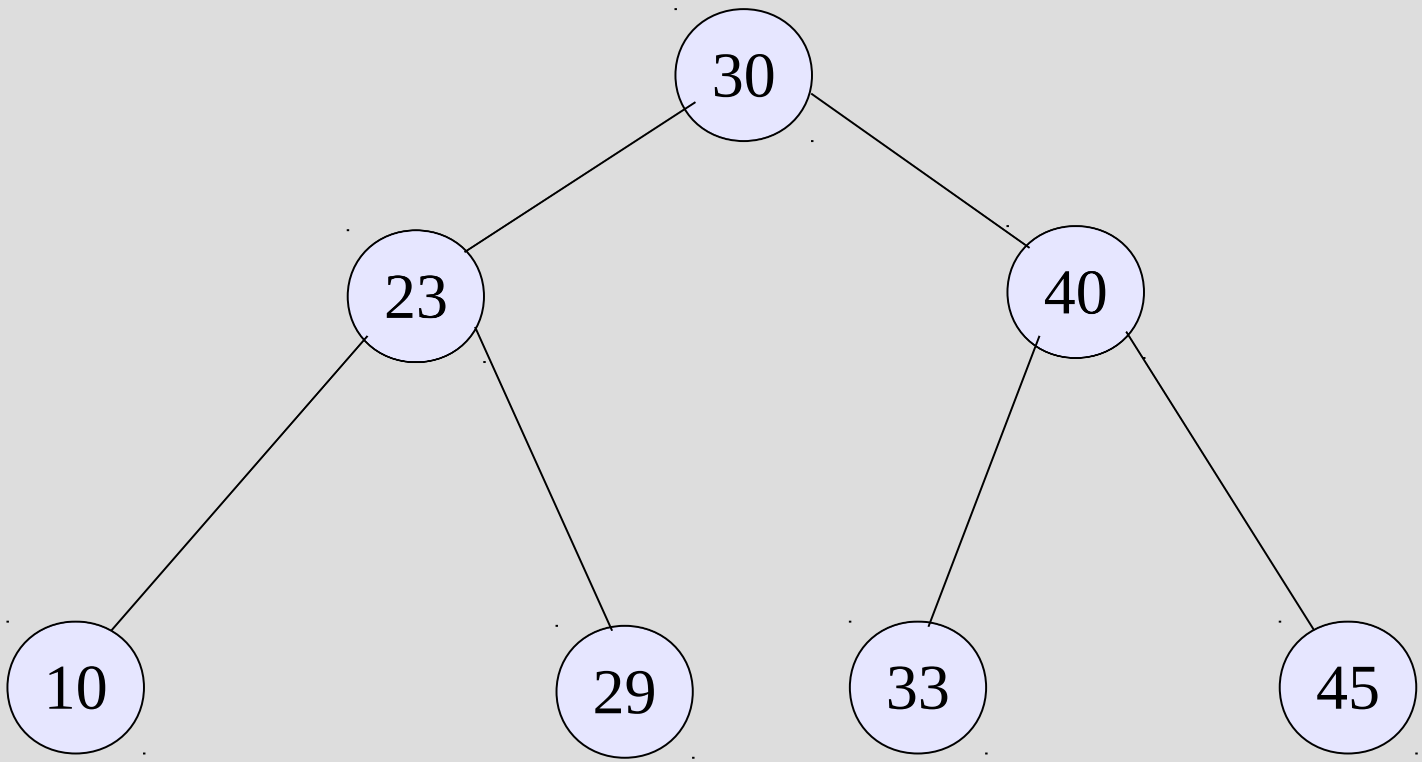
- Node n1 is an ancestor of node n2 if n1 is either the father of n2 or the father of some ancestor of n2

- Node n2 is a left descendant of node n1 if n2 is either the left son of n1 or a descendant of the left son of n1

BINARY SEARCH TREE

- A binary search tree is a binary tree where all the elements in the left subtree of a node n are less than the contents of n, and all the elements in the right subtree of n are greater than or equal to the contents of n

BINARY SEARCH TREES



HEIGHT BALANCED BINARY TREES

Let us consider the following data

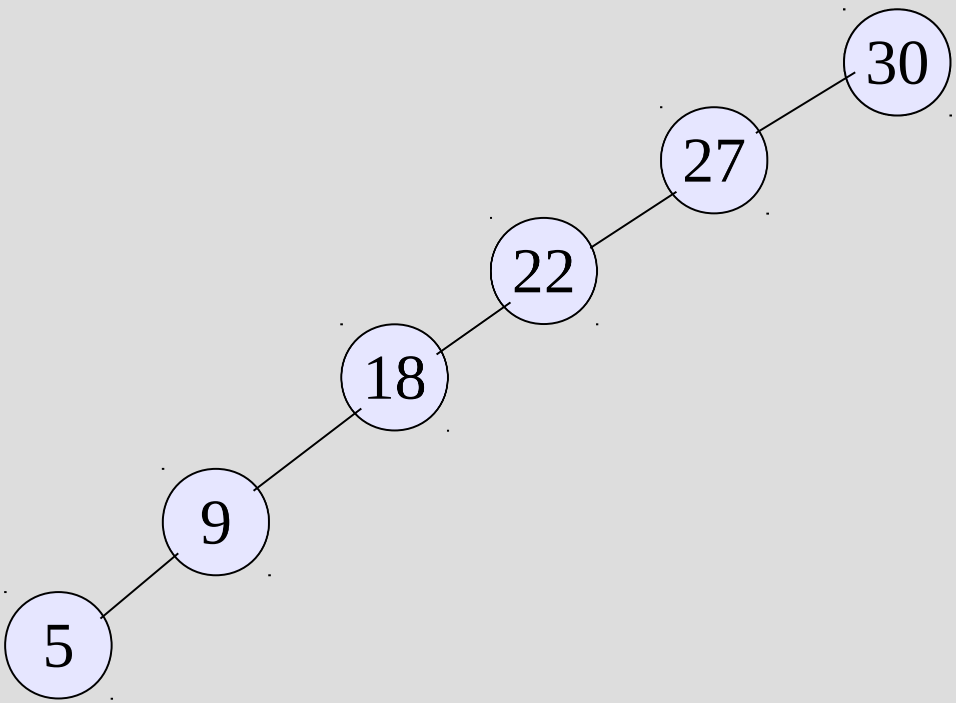
30 27 22 18 9 5

Build a binary search tree

HEIGHT BALANCED BINARY TREES

Let us consider the following data

30 27 22 18 9 5



Left skewed binary tree

HEIGHT BALANCED BINARY TREES

- The balance factor of a binary tree

  Balance factor= Height of the left subtree(h_L) – Height of the right subtree(h_R)

- A binary search tree is said to be height balanced binary search tree if all its nodes have a balance factor of 1, 0, or -1

$|\text{Balance factor}| = |h_L - h_R| <= 1$, for every node in the tree

HEIGHT BALANCED BINARY TREES

To balance an unbalance tree due to insertion of a node the steps are

- Insert node into a binary search tree
- Compute the balance factors
- Decide the pivot node

  The node whose absolute value of balance factor is switched from 1 to 2, mark it as a special node called pivot node. There may be more than one node whose balance factor is switched from 1 to 2 but the nearest node to newly inserted node will be the pivot node
- Balance the unbalance tree using AVL rotation

HEIGHT BALANCED BINARY TREES

AVL Rotations

- In 1962 two Russian mathematicians G M Adelson-Velski and E M Lendis

- So AVL rotation

- 4 cases of rotations

AVL ROTATIONS

CASE I

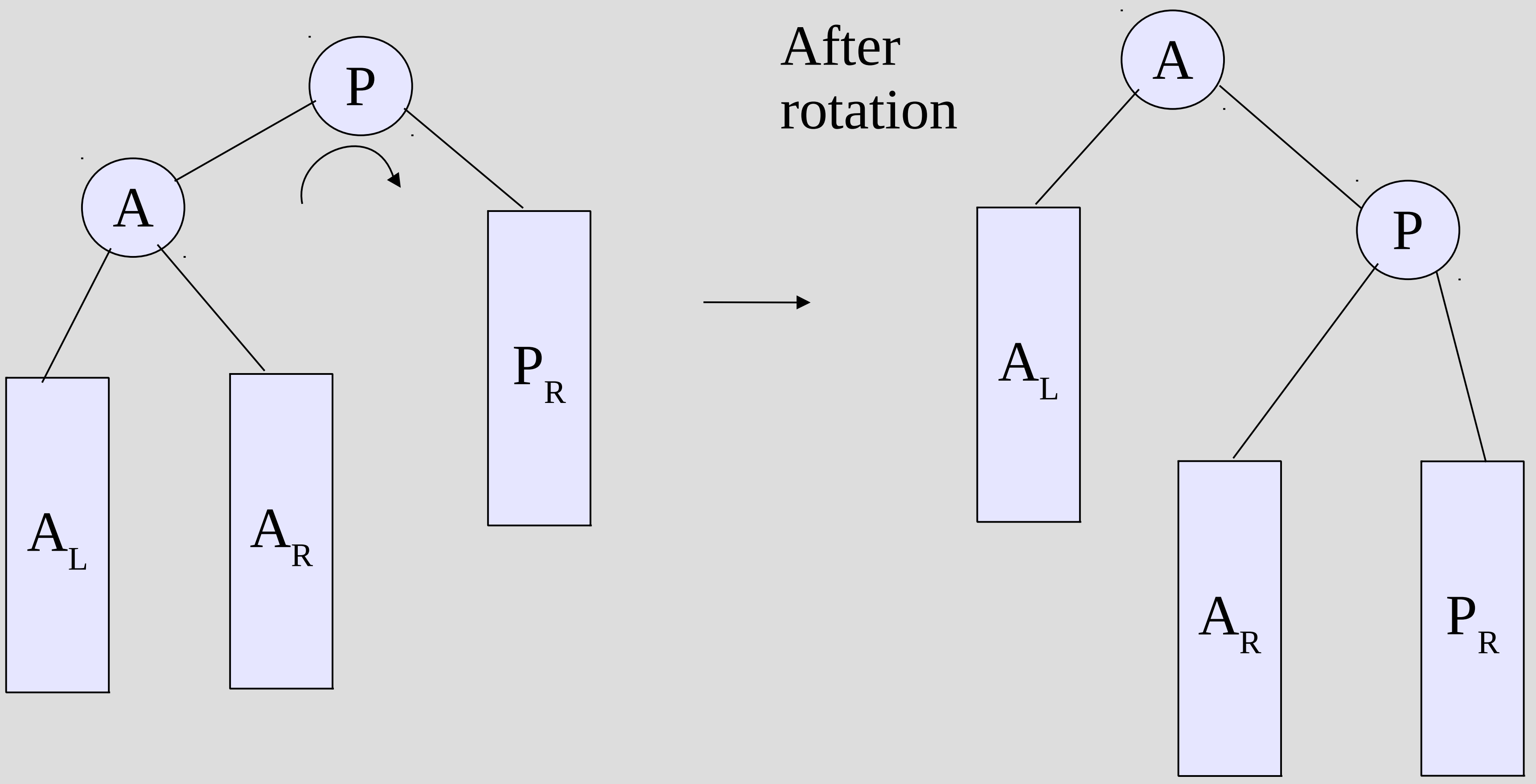
Unbalance occurred due to the insertion in the left subtree of the left child of the pivot node

The steps are

- Right subtree(A_R) of the left child (A) of the pivot node (P) becomes the left subtree of P
- P becomes the right child of A
- Left subtree (A_L) of A remains the same

This is left-to-left insertions(LL Rotation)

AVL ROTATIONS



AVL ROTATIONS

CASE II

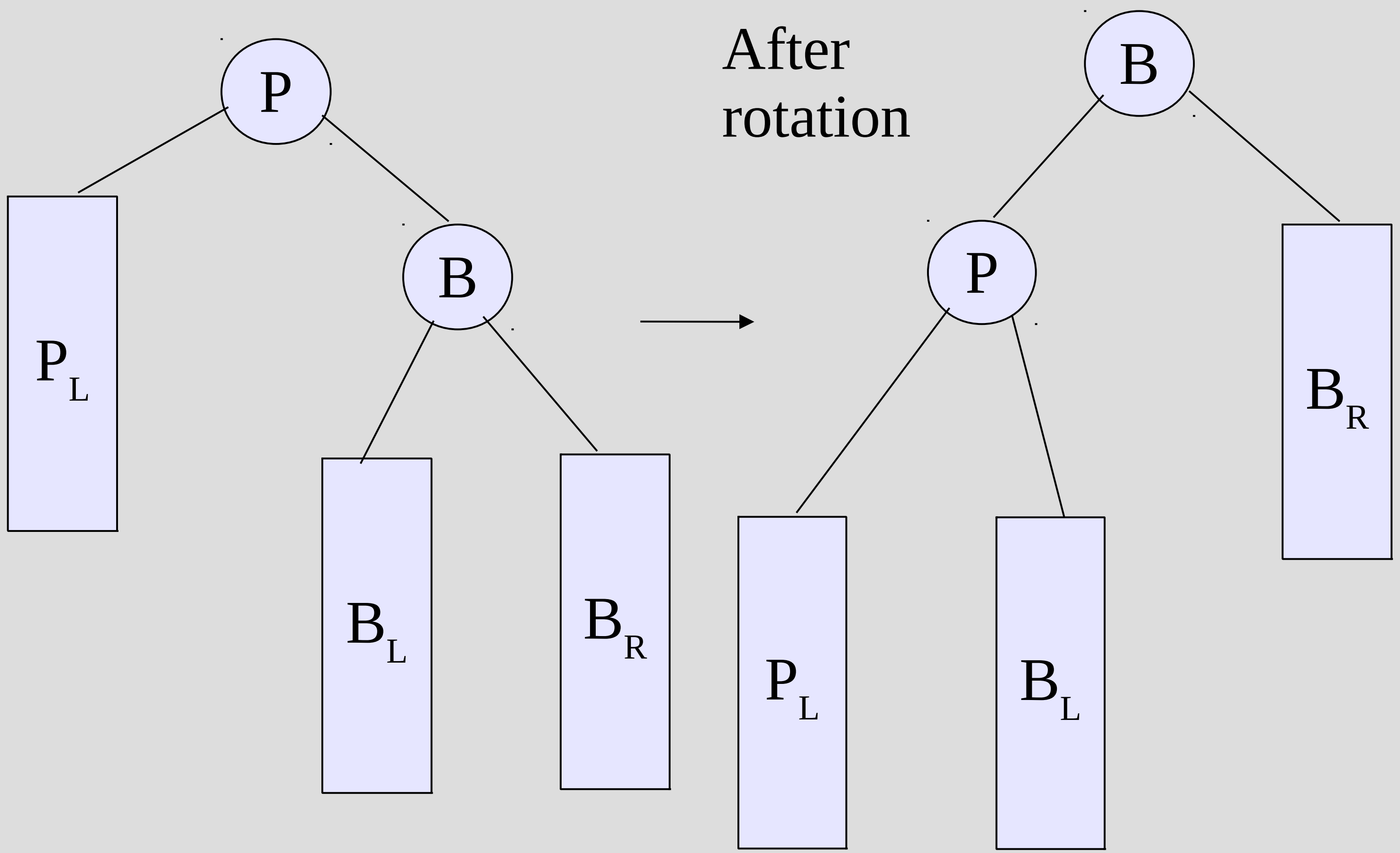
Unbalance occurred due to the insertion in the right subtree of the right child of the pivot node

The steps are

- Left subtree(B_L) of the right child (B) of the pivot node (P) becomes the right subtree of P
- P becomes the left child of B
- Right subtree (B_R) of B remains the same

This is Right-to-Right insertions(RR Rotation)

AVL ROTATIONS



AVL ROTATIONS

CASE III

Unbalance occurred due to the insertion in the right subtree of the left child of the pivot node

This involves two rotations

Rotation I

- Left subtree(B_L) of the right child (B) of the left child of the pivot node (P) becomes the right subtree of the left child (A)
- Left child (A) of the pivot node (P) becomes the left child of B

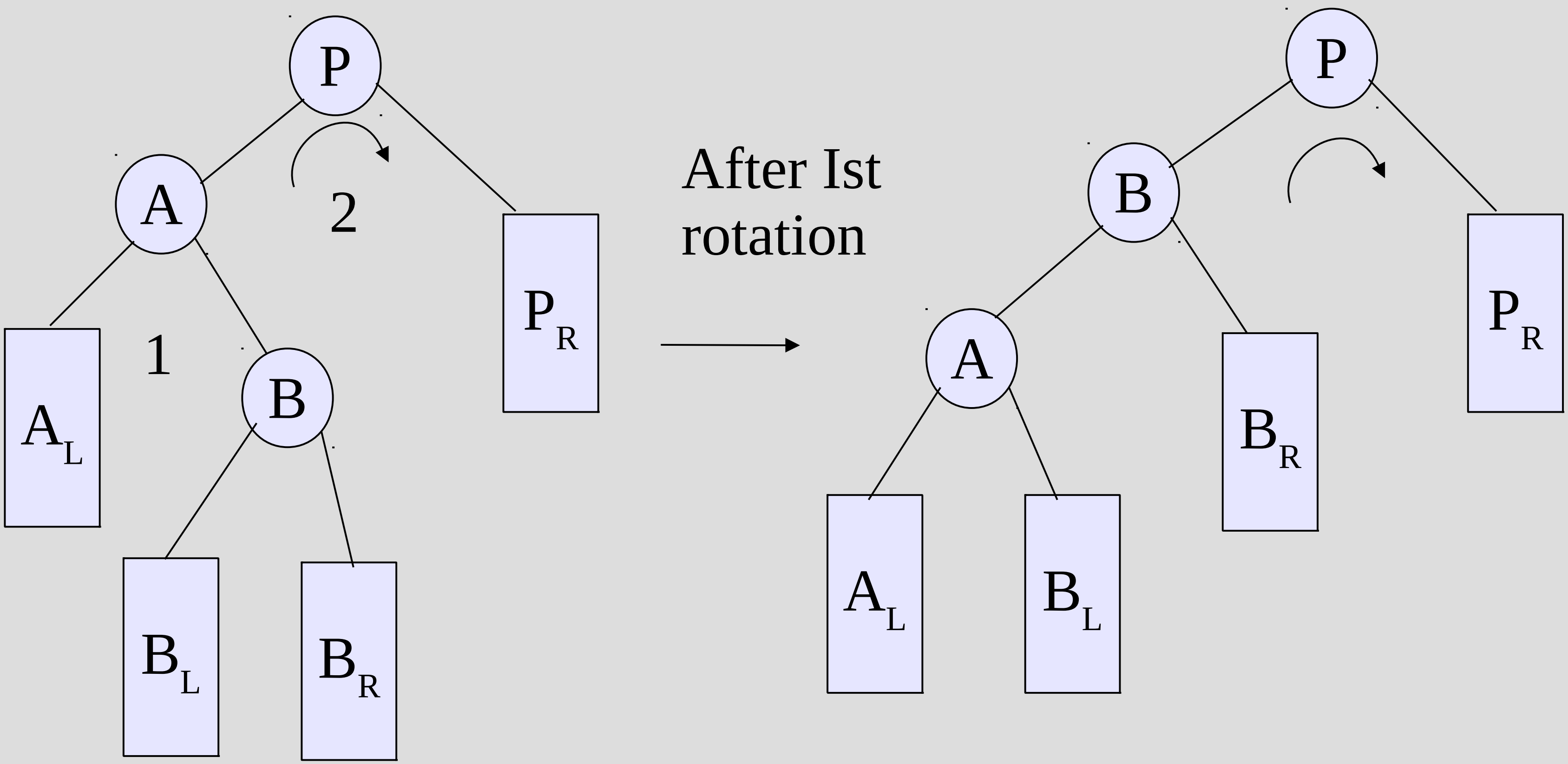
AVL ROTATIONS

Rotation 2

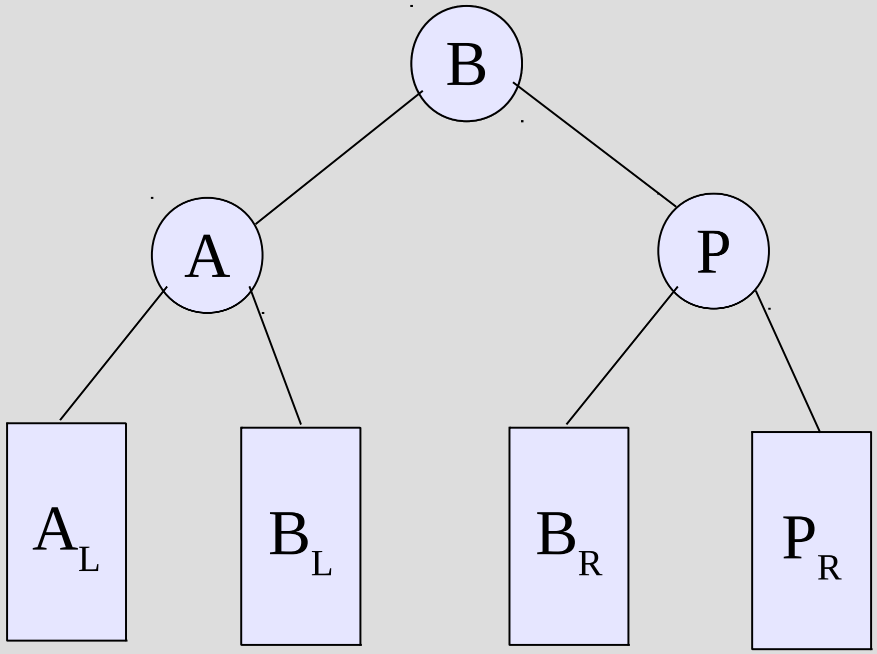
- Right subtree(B_R) of the right child (B) of the left child A of the pivot node (P) becomes the left subtree of P
- P becomes the right child of B

This is LR Rotation

AVL ROTATIONS



AVL ROTATIONS



After 2nd rotation

AVL ROTATIONS

CASE IV

Unbalance occurred due to the insertion in the left subtree of the right child of the pivot node

This involves two rotations

Rotation I

- Right subtree(B_R) of the left child (B) of the right child (A) of the pivot node (P) becomes the left subtree of(A)
- Right child (A) of the pivot node (P) becomes the right child of B

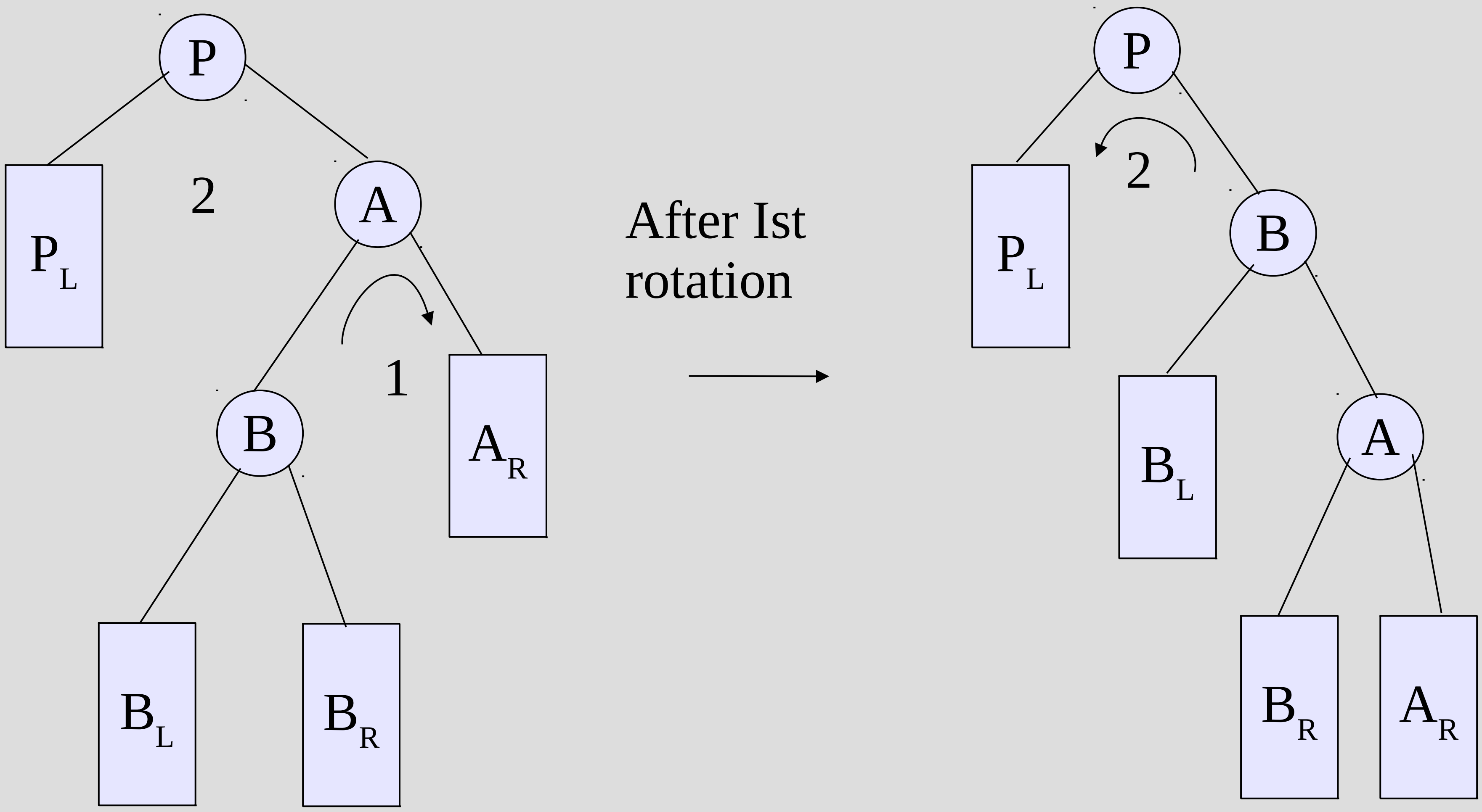
AVL ROTATIONS

Rotation 2

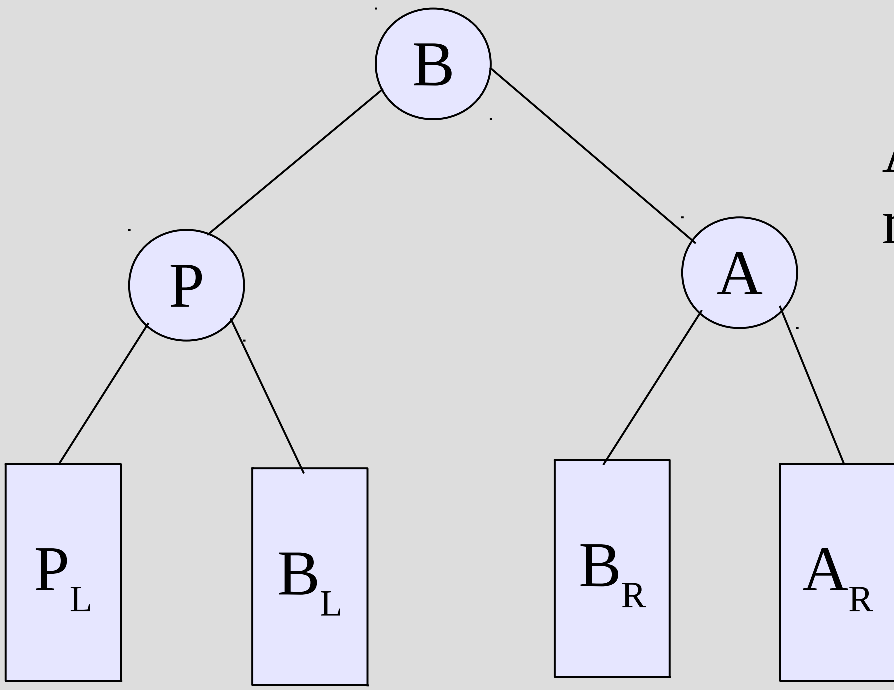
- Left subtree(B_L) of the right child (B) of the right child A of the pivot node (P) becomes the right subtree of P
- P becomes the left child of B

This is RL rotation

AVL ROTATIONS



AVL ROTATIONS



After 2nd rotation

B-Trees

m – way search tree

- An m-way search tree T is a tree in which all nodes are of degree <=m
- Each node in the tree contains the following attributes

P_0	K_1	P_1	K_2	P_2.......................	K_n	P_n

where $1<=n<=m$,

K_i $(1<=i<=n)$ are key values in the node

P_i $(0<=i<=n)$ are pointers to the subtrees of T

B-Trees

- $K_i < K_{i+1}$, $1 <= i <= n$
- All the key values in the subtree pointed by P_i are less than the key values K_{i+1}, $0 <= i <= n$
- All the key values in the subtree pointed by Pn is greater than Kn
- All the subtrees pointed by P_i ($0 <= i <= n$) are also m-way search trees

B-Trees

A B-Tree T of order m is an m-way search tree that is either empty, or it satisfies the following properties

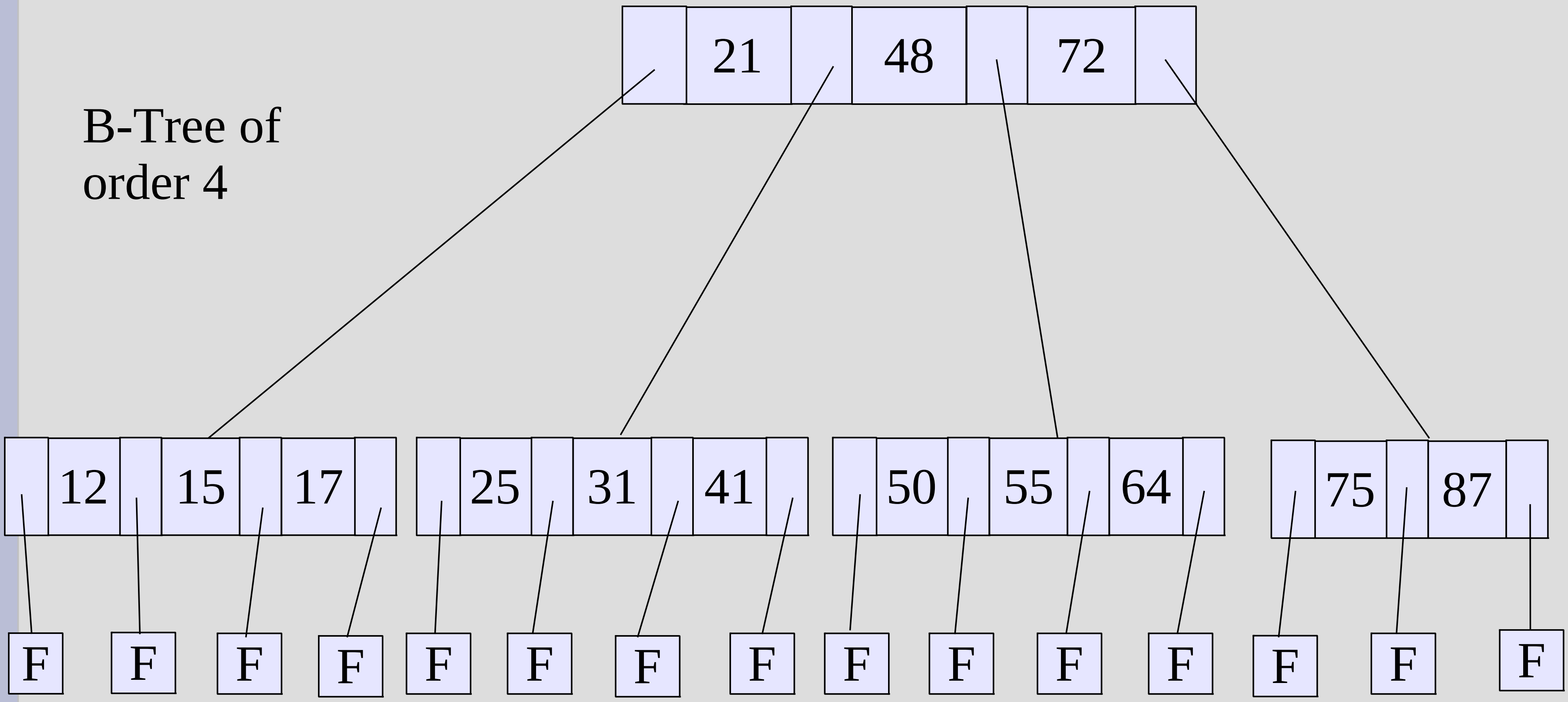
- The root node is either a leaf or has at least two children
- All nodes other than the root node have at least $\lceil m/2 \rceil$ child
- All failure nodes are at the same level

B-Trees

- A failure node represents a node which can be reached during a search only if the value, say X, being searched for is not in the tree
- These empty subtrees are replaced by hypothetical nodes called failure nodes

B-Trees

B-Tree of order 4

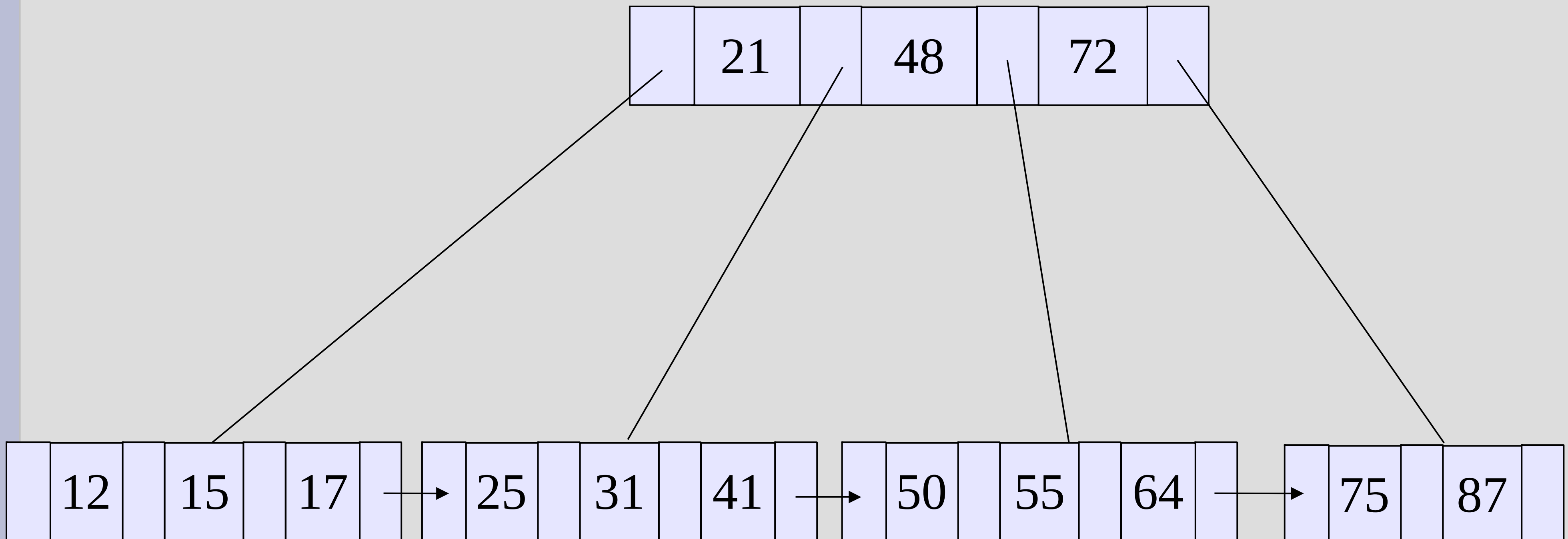


B^+-Trees

- One of the major drawbacks of the B-Tree is the difficulty of traversing the keys sequentially.

In B^+-trees the leafs are linked together to provide a sequential path for traversing the keys in the tree.

B+-Trees



Thank You