1. **Introduction about B-Trees:**

Linked lists have great advantages of flexibility over the contiguous representation of data structures, but they have one weak feature: They are sequential lists; that is, they are arranged so that it is necessary to move through them only one position at a time. Therefore, we can consider trees or graphs as a data structure,using the methods of pointers and linked lists for their implementation.Data structures organized as trees or graphs can prove valuable for a range of applications, especially for problems of information retrieval.

At some moments, we have been drawing trees or graphs to illustrate the behavior of algorithms look like this:



In data structures for programming, we already learned about some data structures such as: binary tree and 2-3 tree, etc. ; which are very useful for programming.





Therefore, we can come to the generalized form of these data structures, which is called: M-Way tree.



Extending more on the M-Way tree, we will get a more special type of data structure: M-Way search tree.



Going deeper on the M-way search tree we get a more specific type of data structure : B-Tree.



1. ***Definition of B-Tree:***

To get a specific definition for B-Tree, first, we should know about M-way trees.

* An **M-way(multi-way) tree** is a tree that has the following properties:
  + Each node in the tree can have at most **m** children.
  + Nodes in the tree have at most **(m-1)** key fields and pointers(references) to the children.



The above image is a 3-way tree, where each node has at most (3-1) = 2 keys and 3 children.

* An **M-way search tree** is a more constrained **M-way tree**, which has more property:
  + Each node in the tree can associate with m children and **m-1** key fields.
  + The keys in any node of the tree are arranged in a sorted order(ascending).
  + The keys in the first **K** children are less than the **K**th key of this node.
  + The keys in the last **(m-K)** children are higher than the **K**th key.



🔼Therefore, a B-tree is a special case of M-way search tree, and we got a new definition:





⏫**Note:**

* 2-3 trees and binary search trees that we learned before can be B-trees.
* If n ≥ 1, then for any n-key B-tree of height h and minimum degree t ≥ 2, h ≥ (n+1)/2

1. ***Application and Advantage for B-tree:***

* Advantage:
  + The need for B-tree arose with the rise in the need for lesser time in accessing the physical storage media like a hard disk. The secondary storage devices are slower with a larger capacity. There was a need for such types of data structures that minimize the disk accesses.
  + Other data structures such as a binary search tree, avl tree, red-black tree, etc can store only one key in one node. If you have to store a large number of keys, then the height of such trees becomes very large and the access time increases.
  + However, B-tree can store many keys in a single node and can have multiple child nodes. This decreases the height significantly allowing faster disk accesses.
* Application :
  + databases and file systems
  + to store blocks of data (secondary storage media)
  + multilevel indexing

**Introduction About Binary Trees.**

Linked lists have great advantages of flexibility over the contiguous representation of data structures, but they have one weak feature: They are sequential lists; that is, they are arranged so that it is necessary to move through them only one position at a time. Therefore, we can consider trees or graphs as a data structure,using the methods of pointers and linked lists for their implementation.Data structures organized as trees or graphs can prove valuable for a range of applications, especially for problems of information retrieval.

At some moments, we have been drawing trees or graphs to illustrate the behavior of algorithms look like this:

And we also know about binary search, which is very useful to search in an ordered array to reduce time complexity when searching, but an ordered array usually has a fixed size. As we know before, linked lists have flexibility for the number of elements, but the searching on the linked lists is a disadvantage. Therefore, we were thinking about how to combine both data structures for both their advantages.

Let's take an example:

We consider a set of 15 numbers: {1;2;3;4;5;6;7;8;9;10;11;12;13;14;15}.

If we apply the binary search for this set of numbers, we will get the order of traversing:

1. 8.
2. 4 and 12
3. 2 , 6, 10 and 14
4. 1 , 3 , 5, 7 , 9, 11, 13, 15

Then, we can draw a tree for this algorithm look like:



Therefore, we have a new definition of a new data structure call Binary Tree:

* *Definition 1*: A binary tree is either empty, or it consists of a node called the root together with two binary trees called the left subtree and the right subtree of the root

We can also have the same and shorter definition for binary tree:

* *Definition 2*: A tree in which each node has at most two children.

So, extend more on the definition, we have some types of binary tree:

1. **Full binary tree**: A binary tree of height h whose leaves are all at the level h and whose nodes all have two children, or a shorter definition is that: a binary with each node having 2 children or 0 children.
2. **Binary search tree**: A binary search tree is a binary tree that is either empty or in which every node has a key (within its data entry) and satisfies the following conditions:

1. The key of the root (if it exists) is greater than the key in any node in the left subtree of the root.

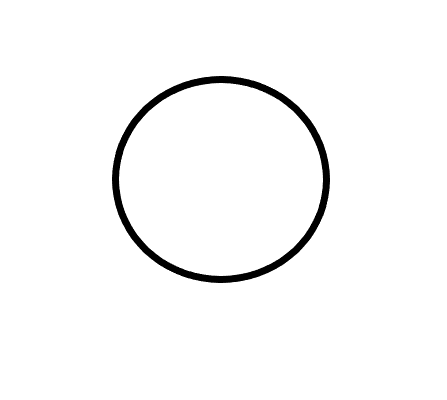
2. The key of the root (if it exists) is less than the key in any node in the right subtree of the root.

3. The left and right subtrees of the root are again binary search trees.

\*\* Note: No two entries (nodes) in a binary search tree may have equal keys.

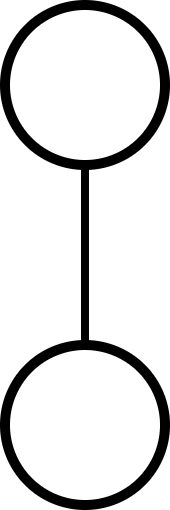
Let’s go deeper on binary tree first:

* First of all, consider some small binary trees:
  + 0 node: There is nothing at all ( of course!)
  + 1 node: There is only 1 binary tree with 1 node.

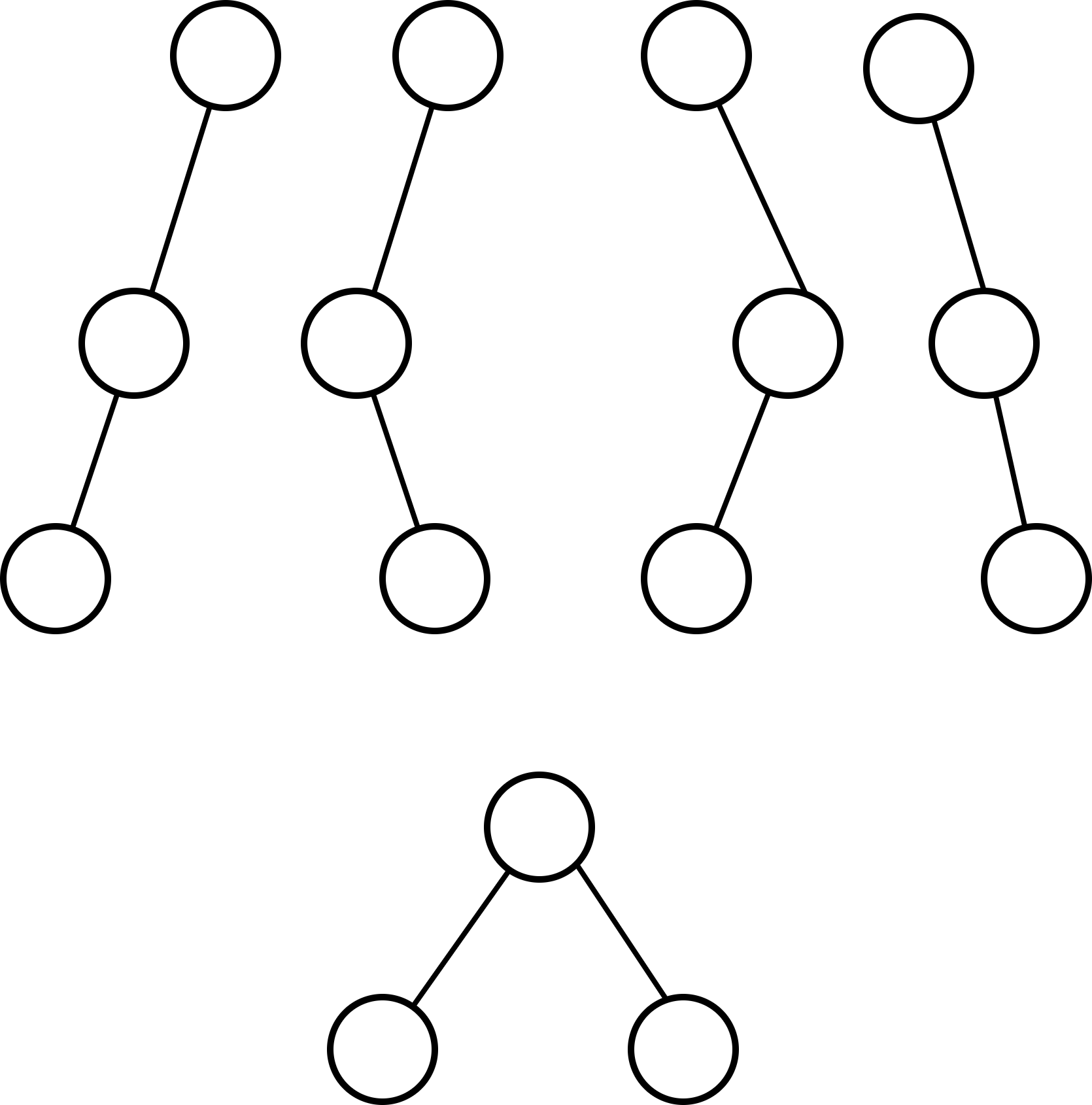


* + 2 node: 2 different tree:



\*\* Note: 2 binary trees above are different from each other.   
\*\* Note: We shall never draw any part of a binary tree to look like:  
Because there is no way to tell if the lower node is the left or the right child of its parent.

* + 3 nodes:



* More nodes on trees can construct more binary trees.