Data Structures 2

DATAMATIKER-B

2020-12-1 LYNGBY

Agenda

Missing topic: rethrow exceptions

Recap

Binary Trees

Exercises

Missing topic: rethrow exception

https://www.tutorialspoint.com/how-to-rethrow-an-exception-in-java

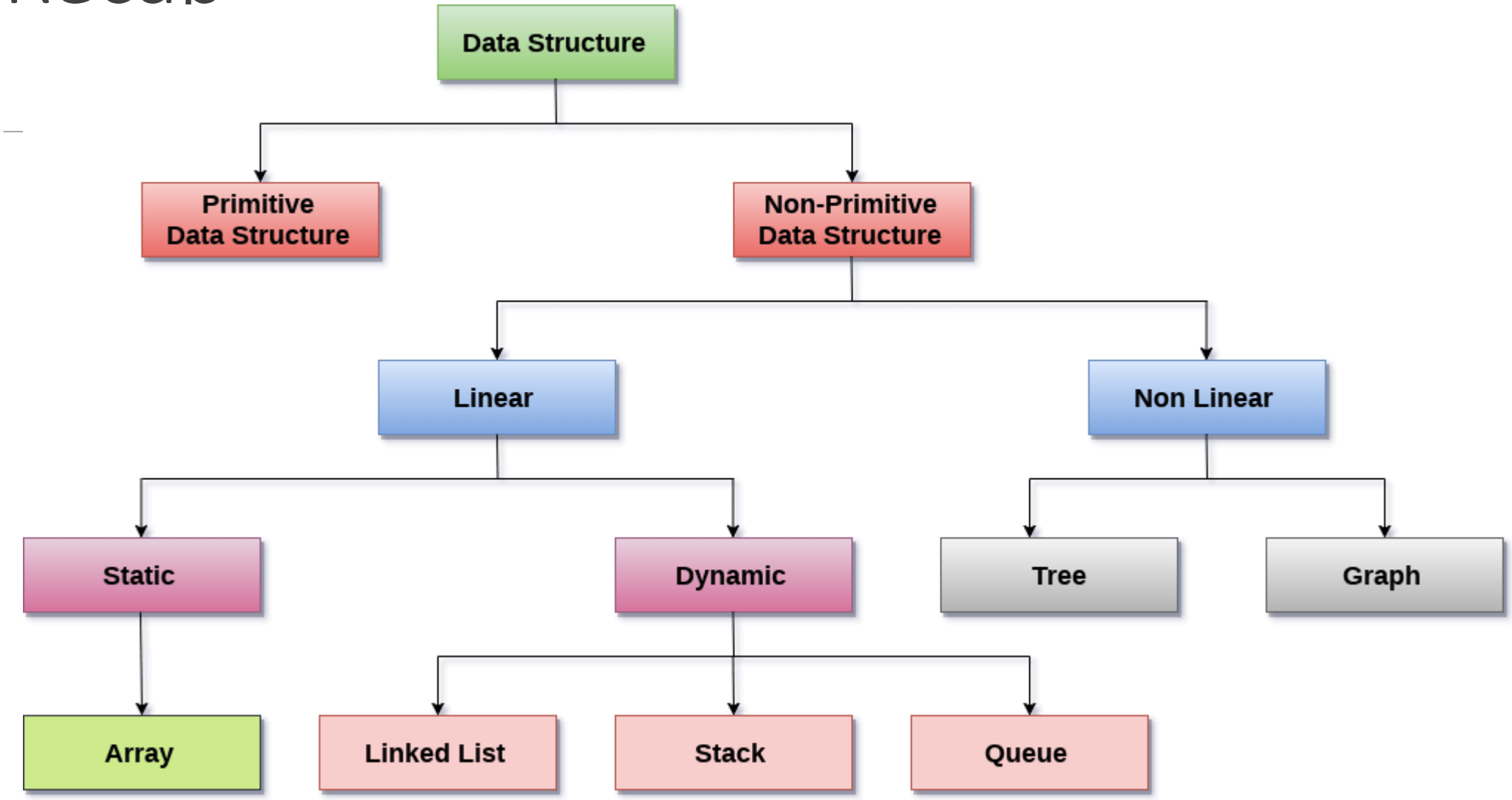
```java
public class RethrowException {
   public static void test1() throws Exception {
      System.out.println("The Exception in test1() method");
      throw new Exception("thrown from test1() method");
   }
   public static void test2() throws Throwable {
      try {
         test1();
      } catch(Exception e) {
         System.out.println("Inside test2() method");
         throw e;
      }
   }
   public static void main(String[] args) throws Throwable {
      try {
         test2();
      } catch(Exception e) {
         System.out.println("Caught in main");
      }
   }
}
```

Output

```
The Exception in test1() method
Inside test2() method
Caught in main
```

Recap

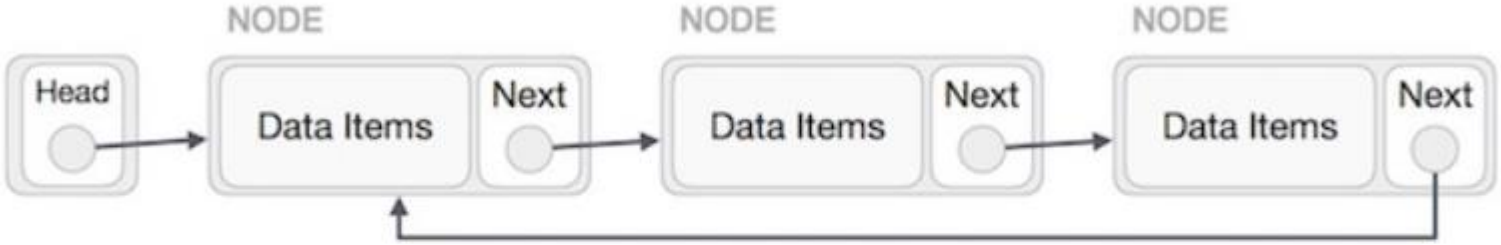
Recap


Data Structure
Primitive Data Structure
Non-Primitive Data Structure
Linear
Non Linear
Static
Dynamic
Tree
Graph
Array
Linked List
Stack
Queue

Recap

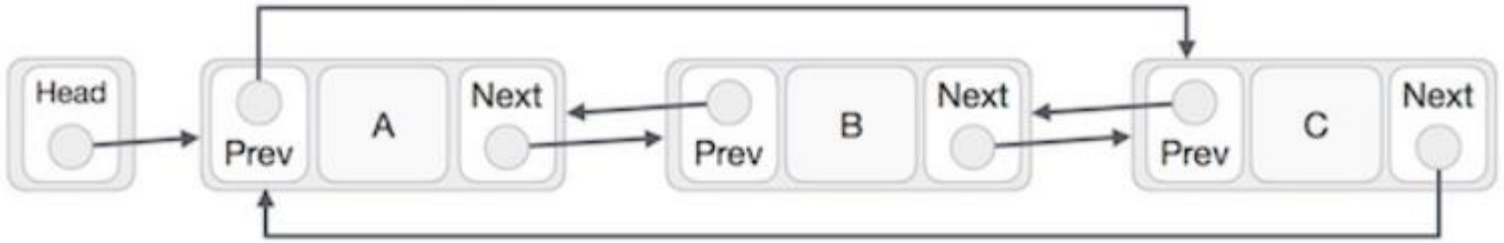
Singly Linked List as Circular

In singly linked list, the next pointer of the last node points to the first node.



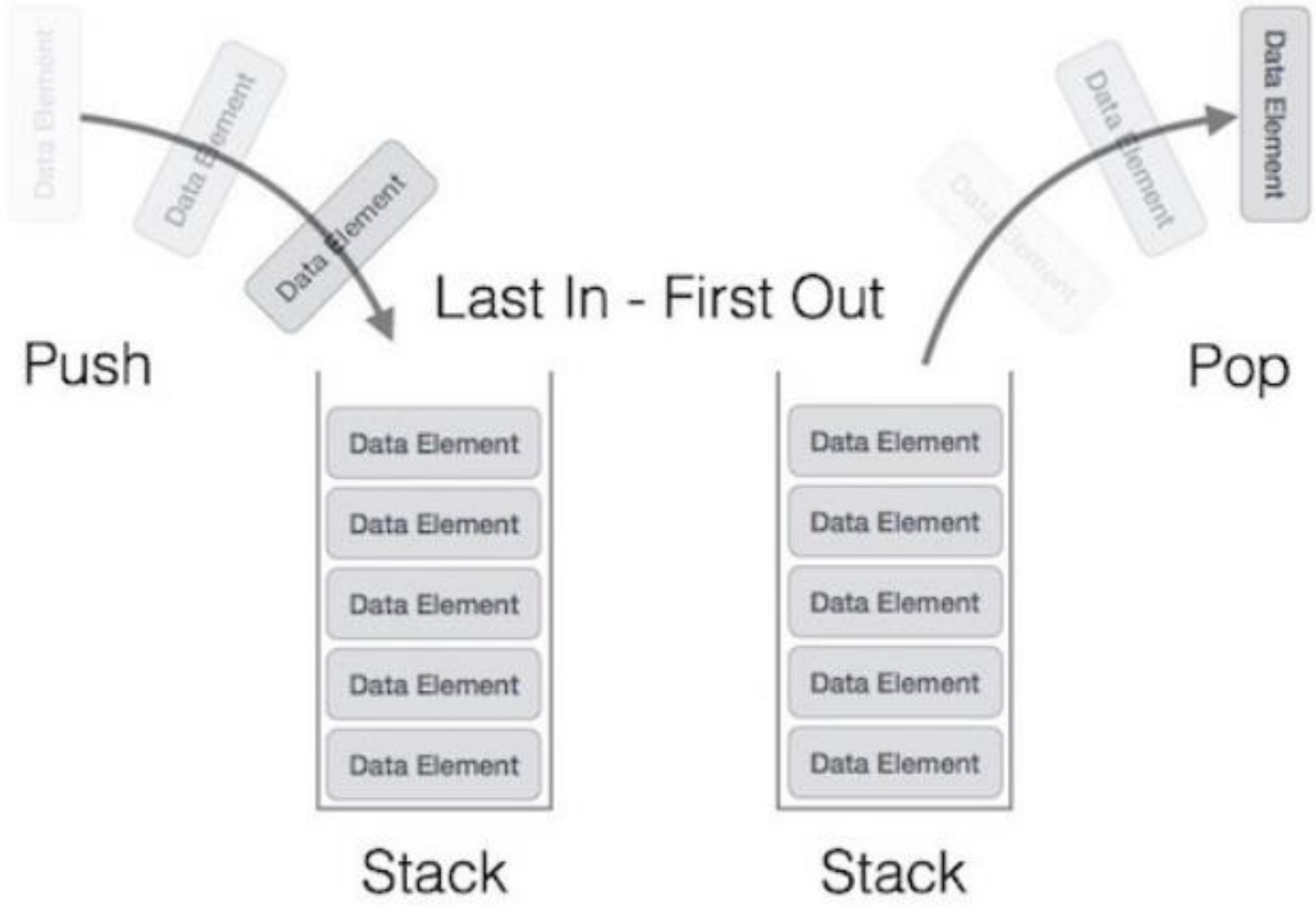
Doubly Linked List as Circular

In doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making the circular in both directions.



Stack Representation

The following diagram depicts a stack and its operations −

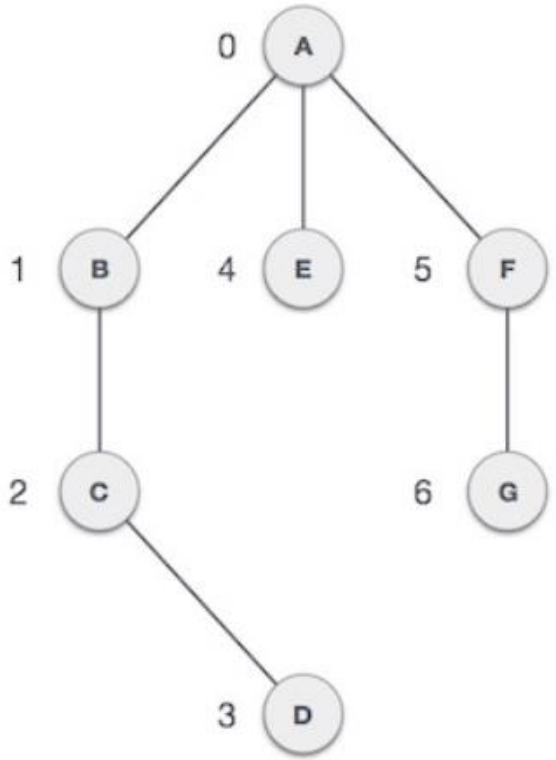


Queues

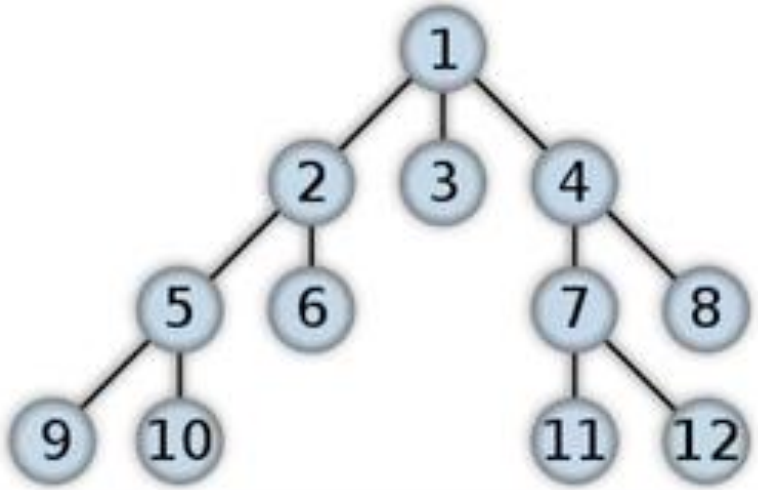

LAST IN
LAST OUT
ONE WAY
FIRST IN
FIRST OUT


In
Data
Data
Data
Data
Data
Data
Out
Last In Last Out
First In First Out
Queue

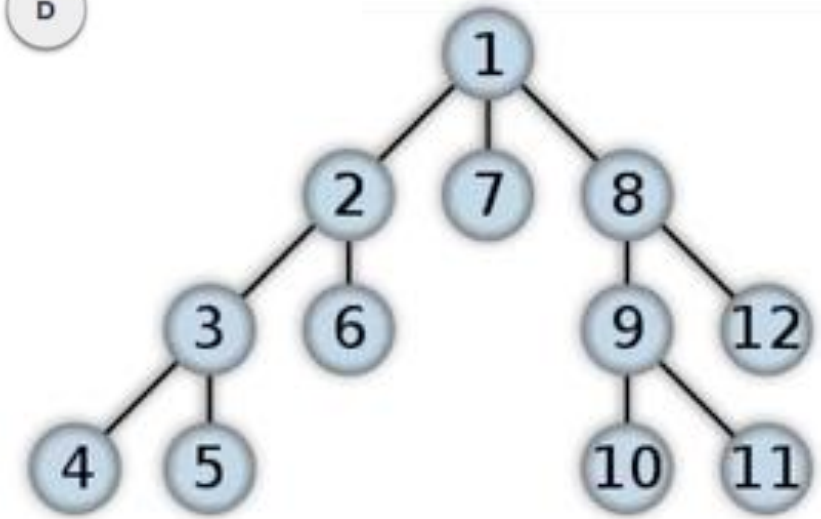
Recap



BFS



DFS



Video tid!

Google job interview

https://youtu.be/XKu_SEDAykw

én ting er opgaven han løser, der måske kunne have relevans for de emner vi sidder med disse dage. En anden ting der er virkelig interessant og relevant er hans evne til at tænke højt. Prøv at læg mærke til det.

Opgaven der stilles giver endvidere et eksempel på en lettere abstrakt problemformulering, som forhåbentlig giver et lille indblik i, hvornår og hvorfor man overhovedet har disse abstrakte data strukturer.

Ses om 30 minutter!

ze jobz interviews

NC:

Nummerplade registrering ved en bro. Hvilke kolonner i en tabel? Hvordan finder man data i en tabel?

Binary Trees

Important Terms

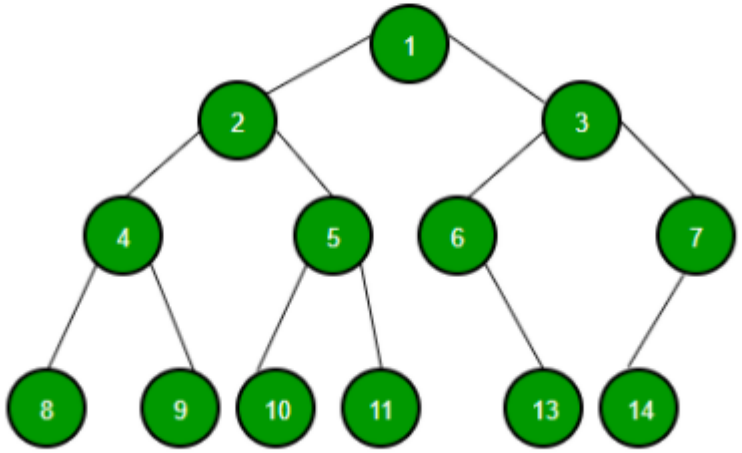
Following are the important terms with respect to tree.

- **Path** − Path refers to the sequence of nodes along the edges of a tree.
- **Root** − The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
- **Parent** − Any node except the root node has one edge upward to a node called parent.
- **Child** − The node below a given node connected by its edge downward is called its child node.
- **Leaf** − The node which does not have any child node is called the leaf node.
- **Subtree** − Subtree represents the descendants of a node.
- **Visiting** − Visiting refers to checking the value of a node when control is on the node.
- **Traversing** − Traversing means passing through nodes in a specific order.
- **Levels** − Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.
- **keys** − Key represents a value of a node based on which a search operation is to be carried out for a node.

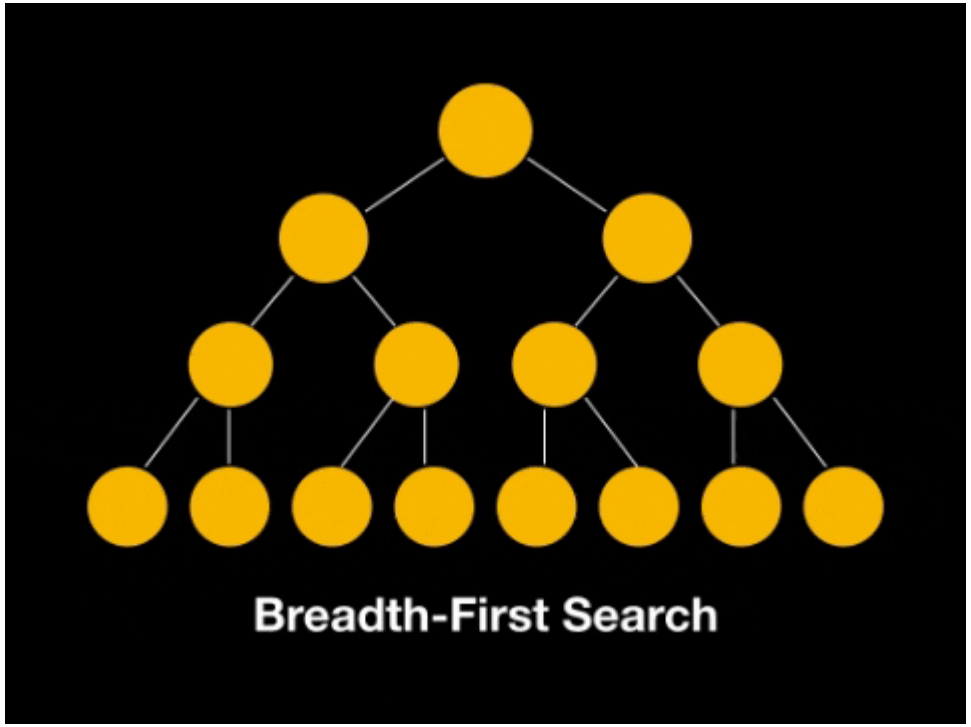
Binary Trees

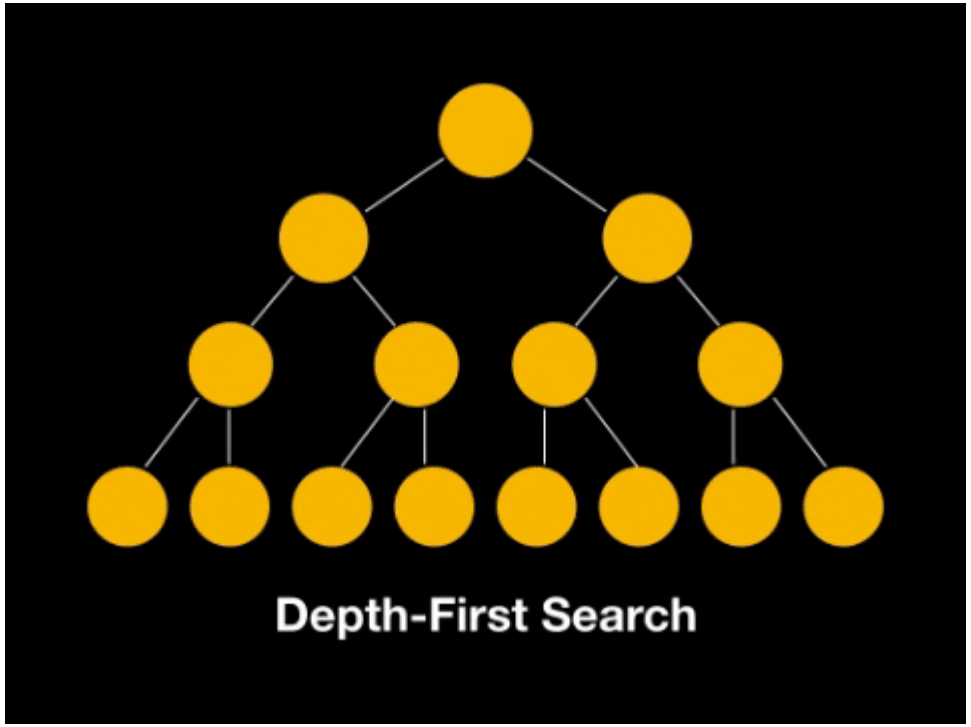
Binary Tree Data Structure

A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

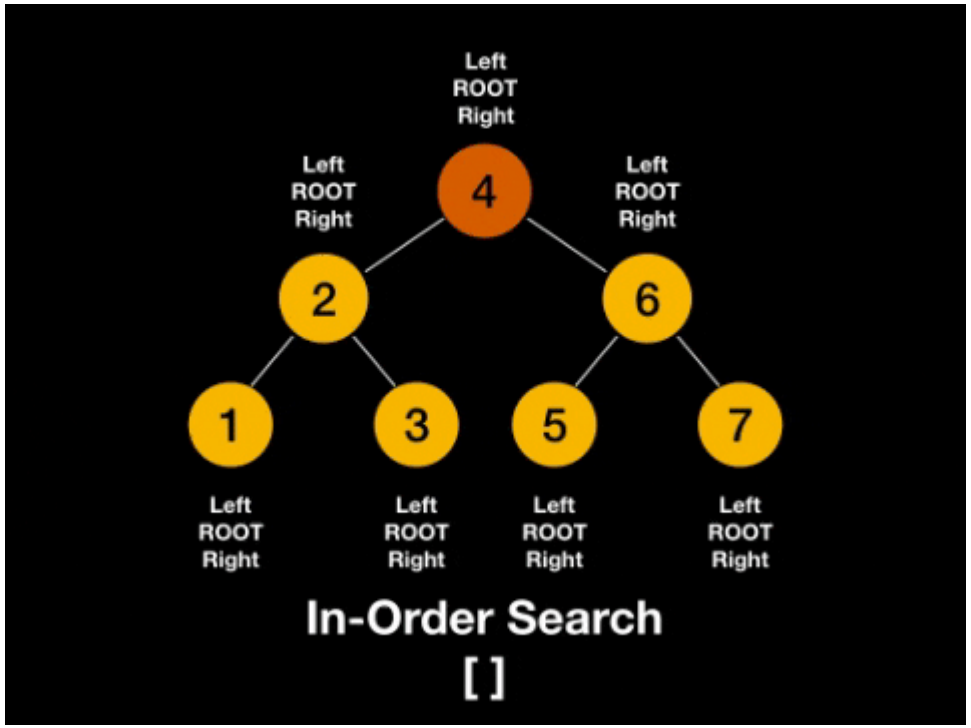


Traversal – BFS or DFS?





Traversal



Note:
In-Order = { left, ROOT, right };
Pre-Order = { ROOT, left, right };
Post-Order = { left, right, ROOT };

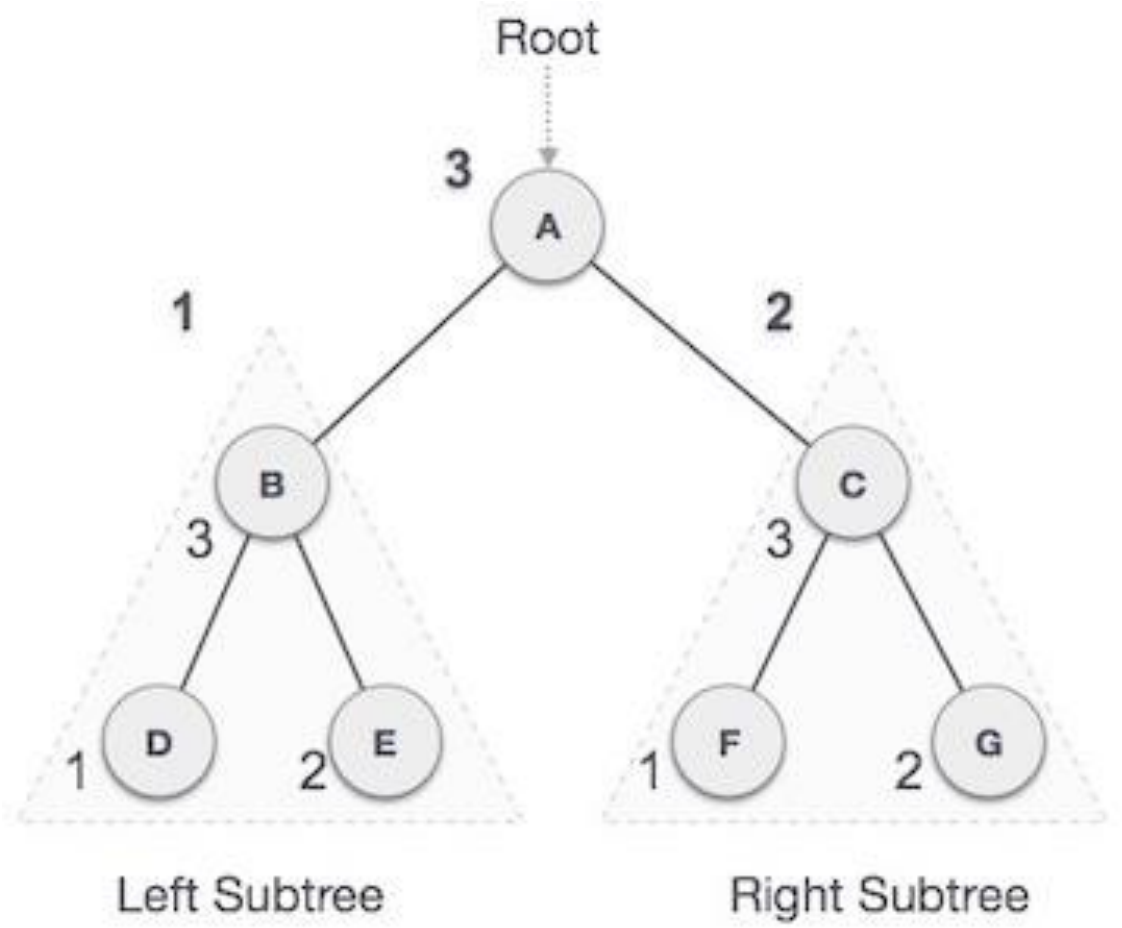
Traversal – steps example

Steps to In-Order Traversal:

1. Traverse the **left subtree** by recursively calling `inOrder` function
2. Process the **root value** by pushing it into `nodes`
3. Traverse the **right subtree** by recursively calling `inOrder` function

Pause

Recursion



Fælleskode

Vi laver et Binary Tree sammen

Exercises

Exercises

Ud fra det binary tree vi har lavet sammen, skal I nu:

Lav en funktion der printer hvor mange nodes der er i vores træ.

Lav en funktion der printer summen af alle values i vores træ.

Printe values af alle nodes (DFS)

Printe values af alle nodes (BFS)

Exercises

Endnu engang har I 3 muligheder, når I er nået hertil:

1. Fortsæt implementering af den data struktur I arbejdede på i går.

2. Start på Torsdag exercises (den er ganske stor!)

Afrunding

I morgen er der online undervisning kl 13, hvor den står på Sortering.