# Java Basic

lecture #10. Trees, Binary Tree

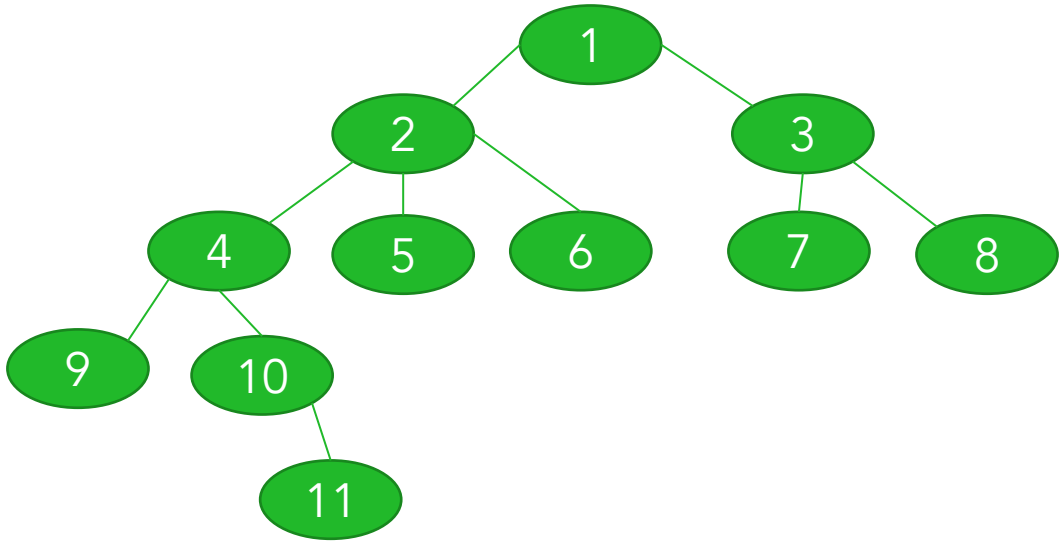
Mentor:Il'yas Miftakhov

# lecture #10. Trees, Binary Tree

- Introduction to Tree Data Structure
  - What is a Tree data structure
- Why Tree is considered a non-linear data structure
- Basic Terminologies In Tree Data Structure
- Properties of a Tree
- Basic Operation Of Tree
- Types of Tree data structures
- Applications of Tree data structure
- AVL
- Red&Black

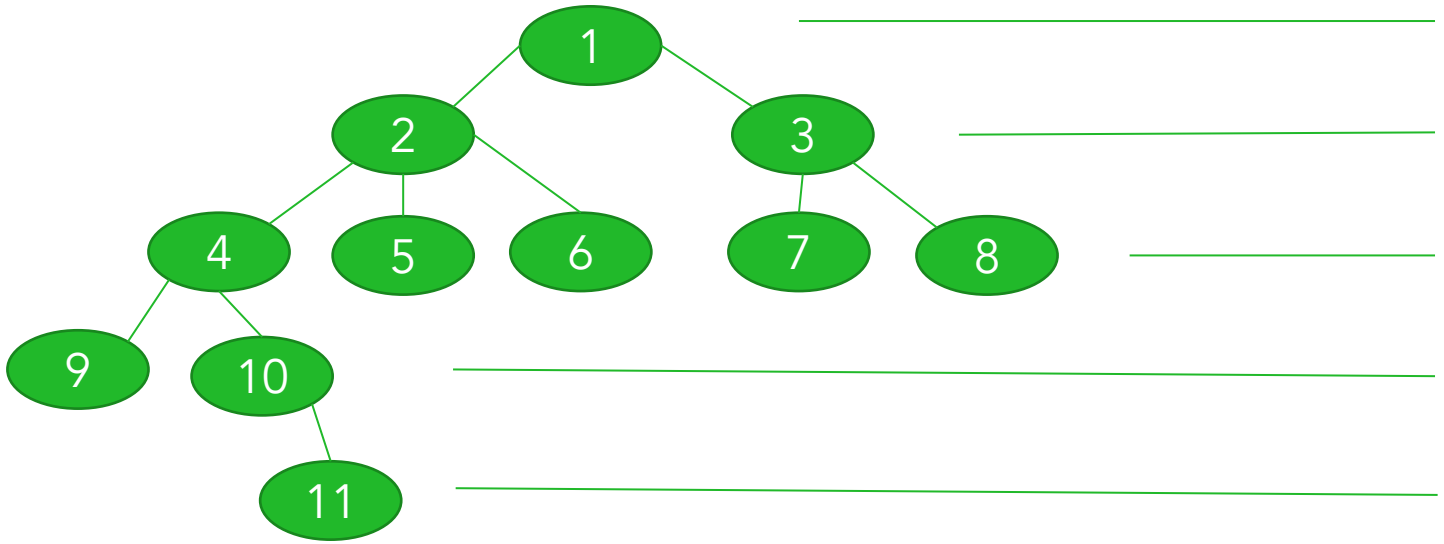
# Introduction to Tree Data Structure

Дерево является нелинейной и представляет собой иерархическую структуру данных, состоящую из набора узлов, так что каждый узел дерева хранит значение и список ссылок на другие узлы («потомки»).



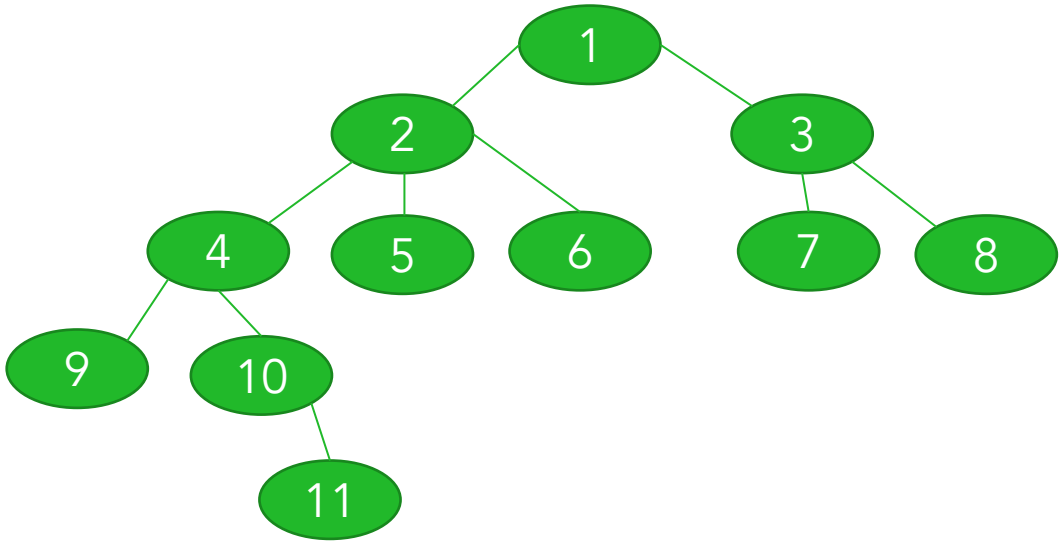
# Why Tree is considered a non-linear data structure

- Данные в дереве не хранятся последовательно, т. е. не линейно.
- Вместо этого они расположены на нескольких уровнях, или можно сказать, что это иерархическая структура.
- По этой причине дерево считается нелинейной структурой данных.



# Basic Terminologies In Tree Data Structure

Root Node: Корневой узел 1
Parent Node: Родительский узел 2 -> 4,5,6
Child Node: Дочерний узел 4,5,6 -> 2
Leaf Node: Лист 5,6,7,8,9,11
Ancestor of a Node: Предок узла 1,2 -> 5
Descendant: Потомок 9,10,11 -> 4
Sibling: Родные братья 4,5,6
Level of a node: Уровень узла
Internal node: Внутренний узел 2
Subtree: Поддерево 4 – 9 – 10 - 11



## Properties of a Tree

- **Количество ребер**: ребро можно определить как соединение между двумя узлами
- **Глубина узла**: Глубина узла определяется как длина пути от корня до этого узла.
- **Высота узла**: высота узла может быть определена как длина самого длинного пути от узла до конечного узла дерева.
- **Высота дерева**: Высота дерева — это длина самого длинного пути от корня дерева до конечного узла дерева.
- **Степень узла**: общее количество поддеревьев, прикрепленных к этому узлу, называется степенью узла.
  - Степень листового узла должна быть 0 .
  - Степень дерева — это максимальная степень узла среди всех узлов дерева.

# Basic Operation Of Tree

Create – создать дерево в структуре данных.

Insert – вставляет данные в дерев.

Search – ищет определенные данные в дереве, чтобы проверить их наличие или нет.

Delete – удаляет узел из дерева.

In order Traversal – выполнить Обход дерева по порядку.

Определить сложность базовых операций в обычном дереве:

1. поиск - ?
2. вставка - ?
3. удаление - ?

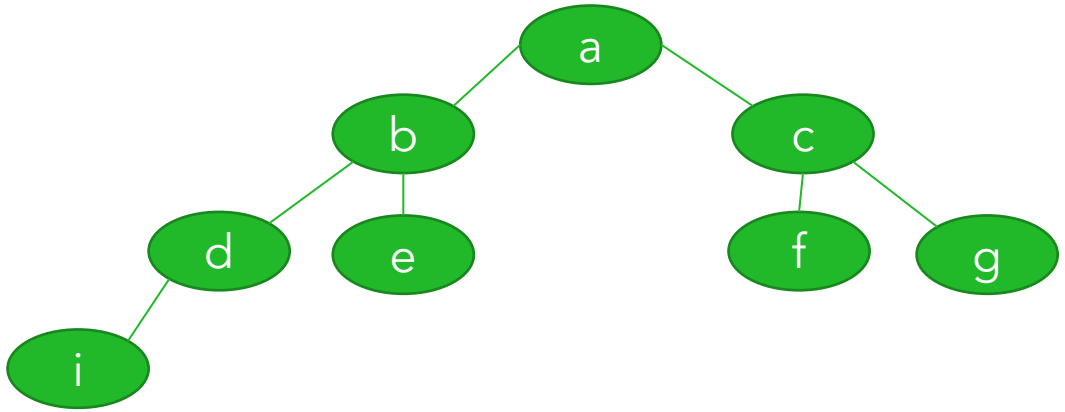
# Types of Tree data structures

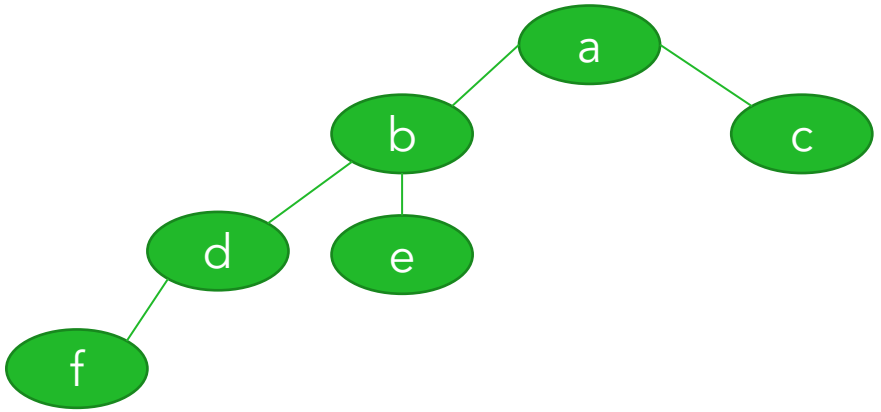
1. General tree
Общая древовидная структура данных не имеет ограничений на количество узлов. Это означает, что родительский узел может иметь любое количество дочерних узлов.

2. Binary tree
У узла бинарного дерева может быть максимум два дочерних узла.

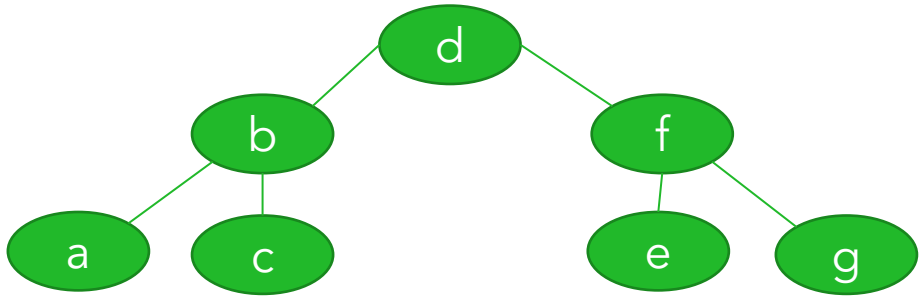
3. Balanced tree
Если высота левого поддерева и правого поддерева равна или отличается не более чем на 1, дерево называется сбалансированным.

4. Binary search tree (BST)
Бинарные деревья поиска используются для различных алгоритмов поиска и сортировки. Это нелинейная структура данных, которая показывает, что значение левого узла меньше, чем у его родителя, а значение правого узла больше, чем у его родителя.


a
b
c
d
e
f
g
i


a
b
c
d
e
f


f
d
b
a


d
b
f
a
c
e
g

# BST = AVL

Самобалансирующееся дерево-это двоичное дерево поиска, которое балансирует высоту после вставки и удаления в соответствии с правилами балансировки:

1. В дереве AVL коэффициент баланса узла может быть только одним из значений 1, 0 или -1.
2. Сохранить дерево AVL сбалансированным после любого изменения в его узлах
3. Ключ уникален в AVL дереве — нет двух узлов, имеющих один и тот же ключ

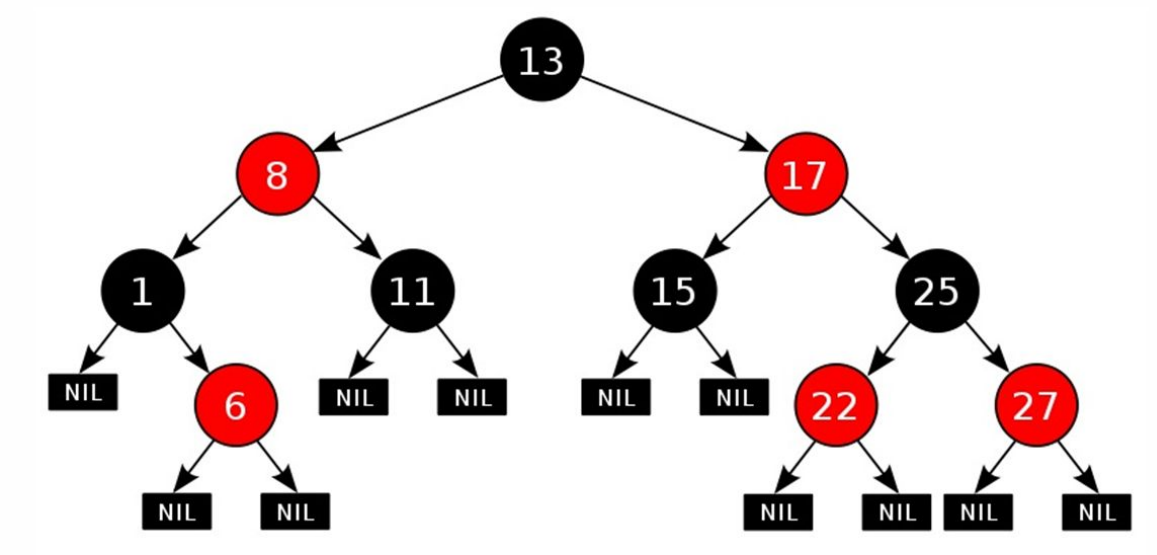
Коэффициент баланса (k) = высота (слева (k)) - высота (справа (k))

- Если коэффициент баланса любого узла равен 1, это означает, что левое поддерево на один уровень выше правого поддерева.
- Если коэффициент баланса любого узла равен 0, это означает, что левое поддерево и правое поддерево имеют одинаковую высоту.
- Если коэффициент баланса любого узла равен -1, это означает, что левое поддерево на один уровень ниже правого поддерева.

# This is Red-Black tree

Правила, которым следует каждое красно-черное дерево:

- Каждый узел имеет красный или черный цвет.
- Корень дерева всегда черный.
- Нет двух соседних красных узлов (красный узел не может иметь красного родителя или красного дочернего элемента).
- Каждый путь от узла (включая корень) к любому из его потомков NULL узлов имеет одинаковое количество черных узлов.
- Все листовые узлы являются черными узлами.



# SOLID

S

Принцип единственной ответственности (single responsibility principle)

Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.

O

Принцип открытости/закрытости (open-closed principle)

«программные сущности … должны быть открыты для расширения, но закрыты для модификации».

L

Принцип подстановки Лисков (Liskov substitution principle)

«функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом». См. также контрактное программирование.

I

Принцип разделения интерфейса (interface segregation principle)

«много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения»

D

Принцип инверсии зависимостей (dependency inversion principle)

«Зависимость на Абстракциях. Нет зависимости на что-то конкретное»