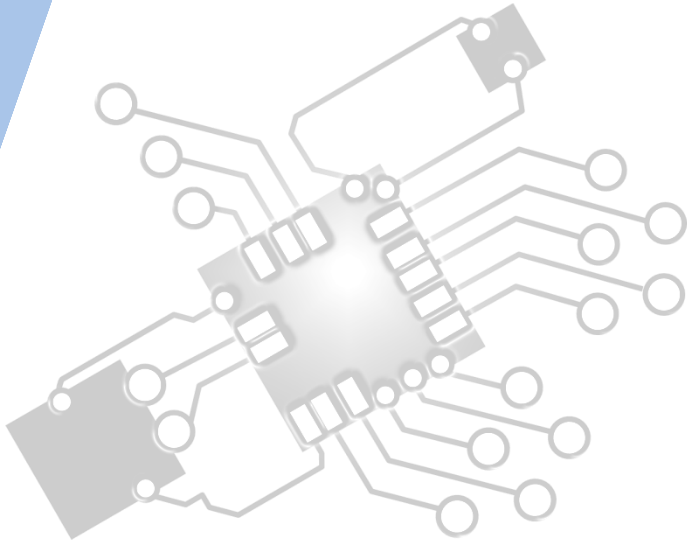
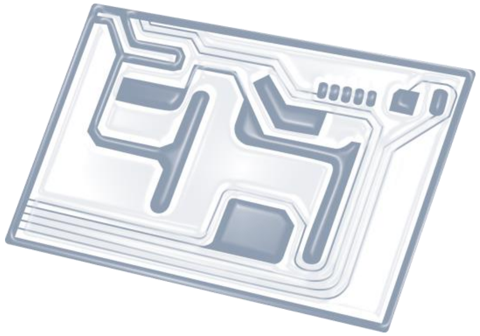
Abstract Data Structures

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL only 5 Overview

Thinking recursively

5.1.1 Identify a situation that requires the use of recursive thinking
5.1.2 Identify recursive thinking in a specified problem solution
5.1.3 Trace a recursive algorithm to express a solution to a problem

Abstract data structures

5.1.4 Describe the characteristics of a two-dimensional array
5.1.5 Construct algorithms using two-dimensional arrays
5.1.6 Describe the characteristics and applications of a stack
5.1.7 Construct algorithms using the access methods of a stack
5.1.8 Describe the characteristics and applications of a queue
5.1.9 Construct algorithms using the access methods of a queue
5.1.10 Explain the use of arrays as static stacks and queues

Linked lists

5.1.11 Describe the features and characteristics of a dynamic data structure
5.1.12 Describe how linked lists operate logically
5.1.13 Sketch linked lists (single, double and circular)

Trees

5.1.14 Describe how trees operate logically (both binary and non-binary)
5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf
5.1.16 State the result of inorder, postorder and preorder tree traversal
5.1.17 Sketch binary trees

Applications

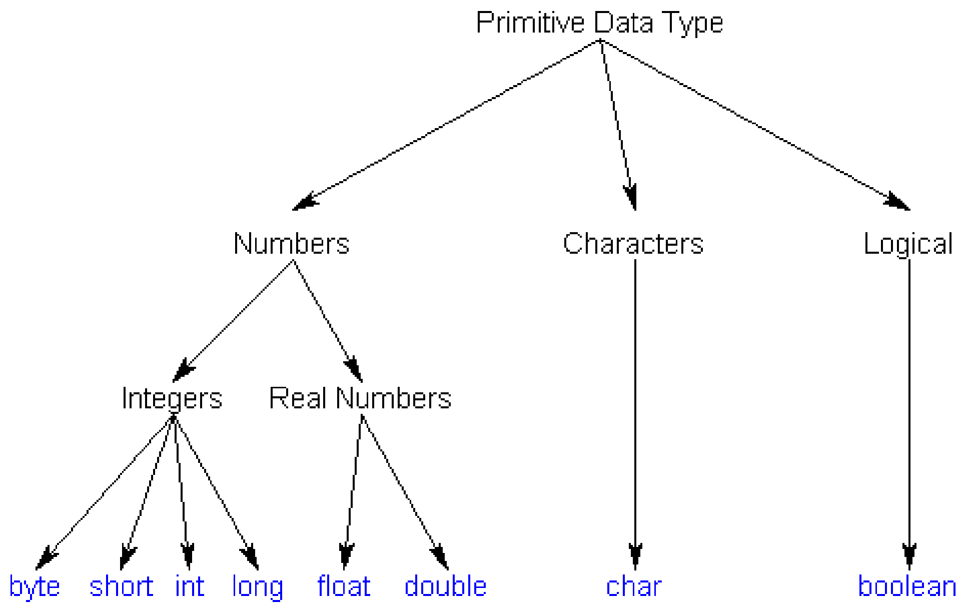
5.1.18 Define the term dynamic data structure
5.1.19 Compare the use of static and dynamic data structures
5.1.20 Suggest a suitable structure for a given situation

Topic 5.1.15

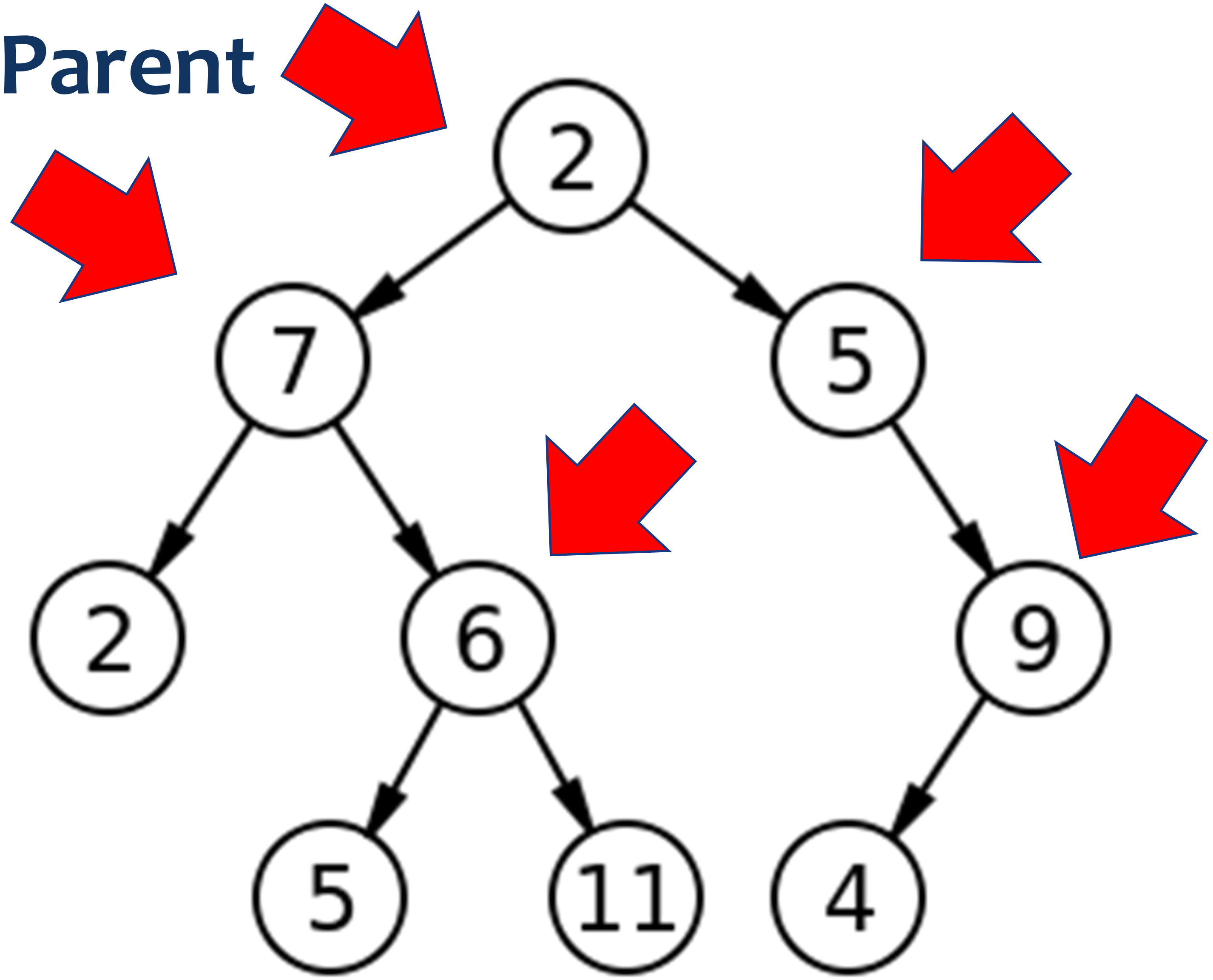
Define the terms: **parent**, **left-child**, **right-child**, **subtree** and **leaf**

Abstract Data Structures (ADTs)

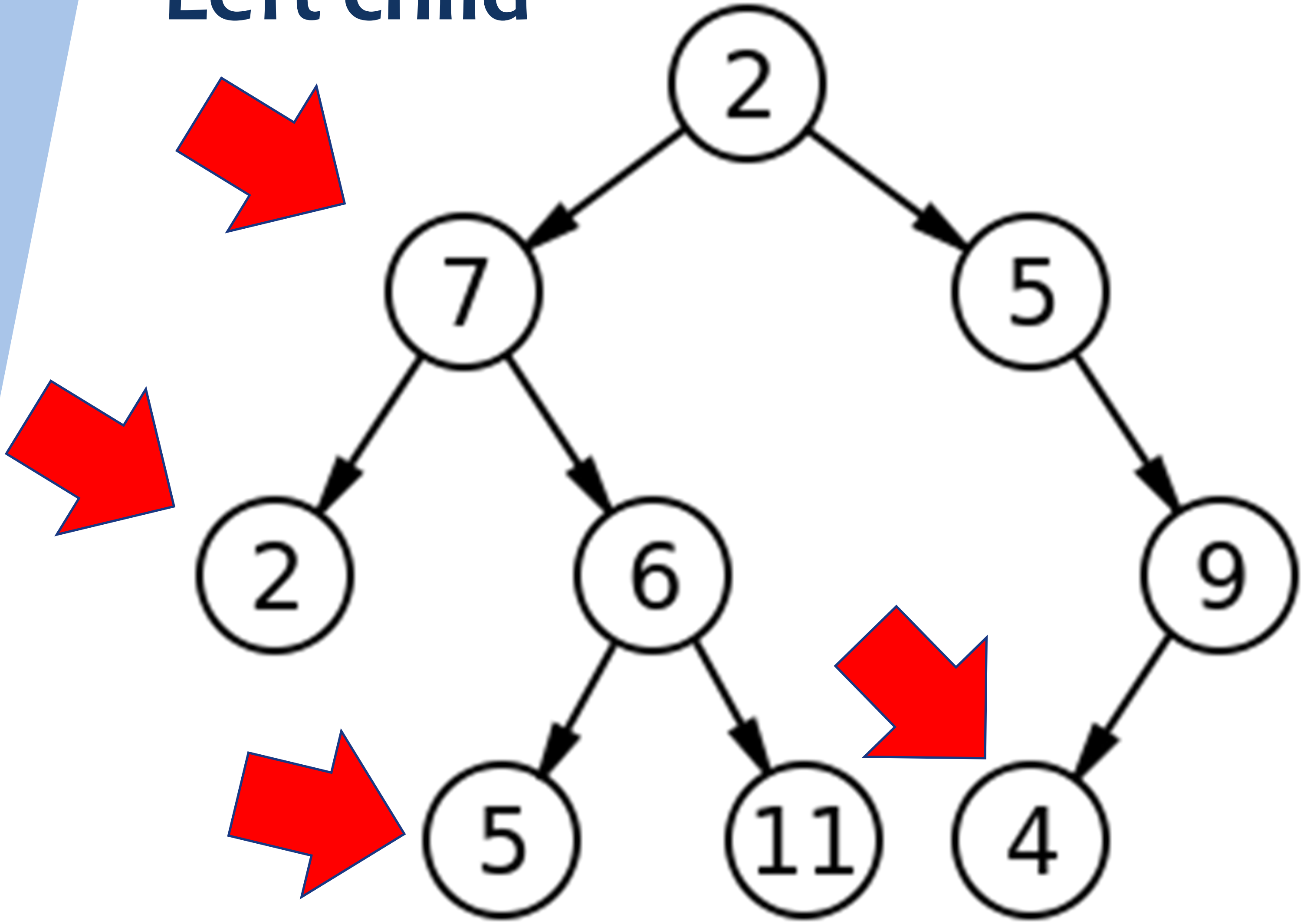
- 2D array
- Stack
- Queue
- Linked List
- **(Binary) Tree**
- Recursion



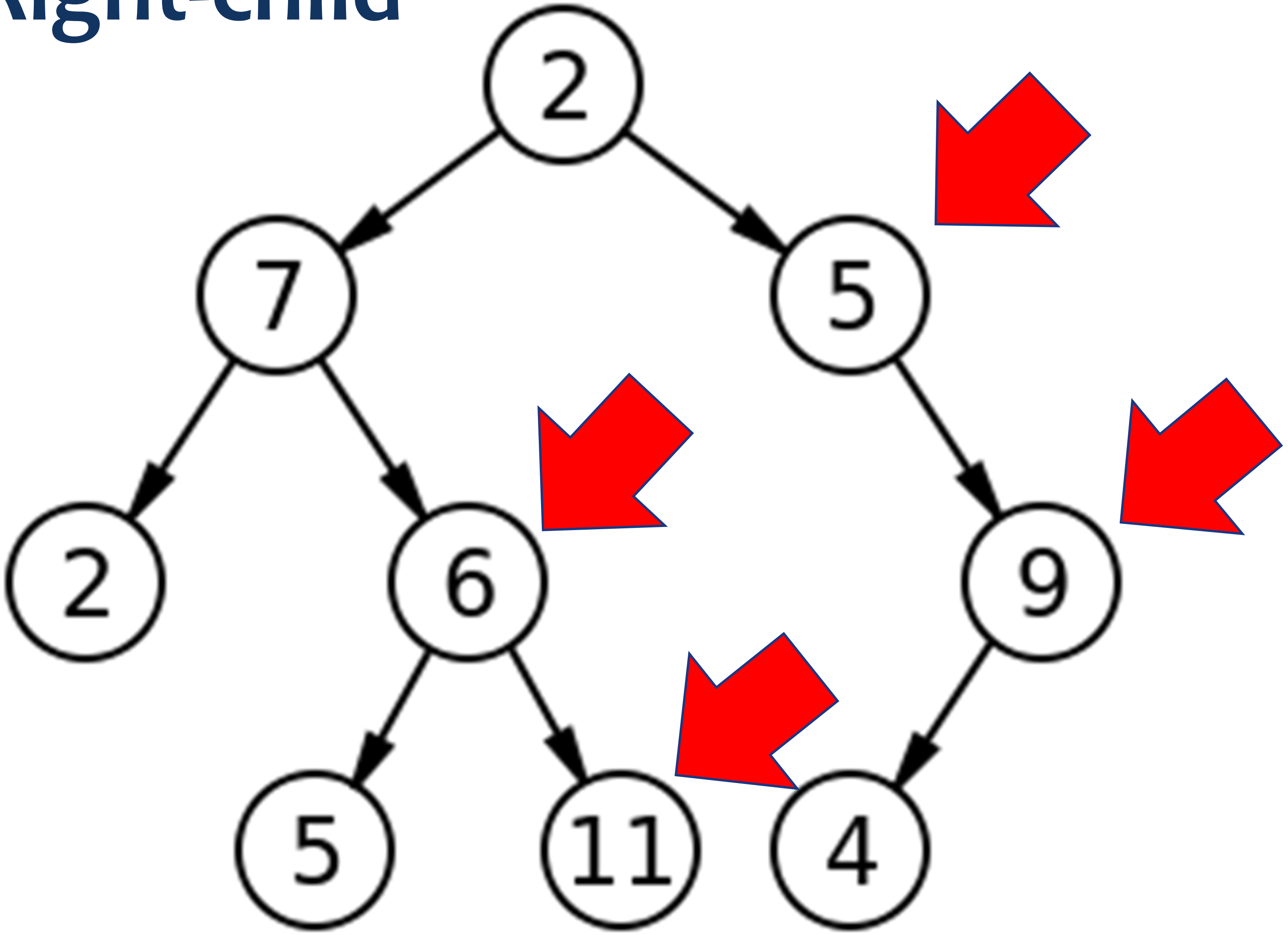
Parent



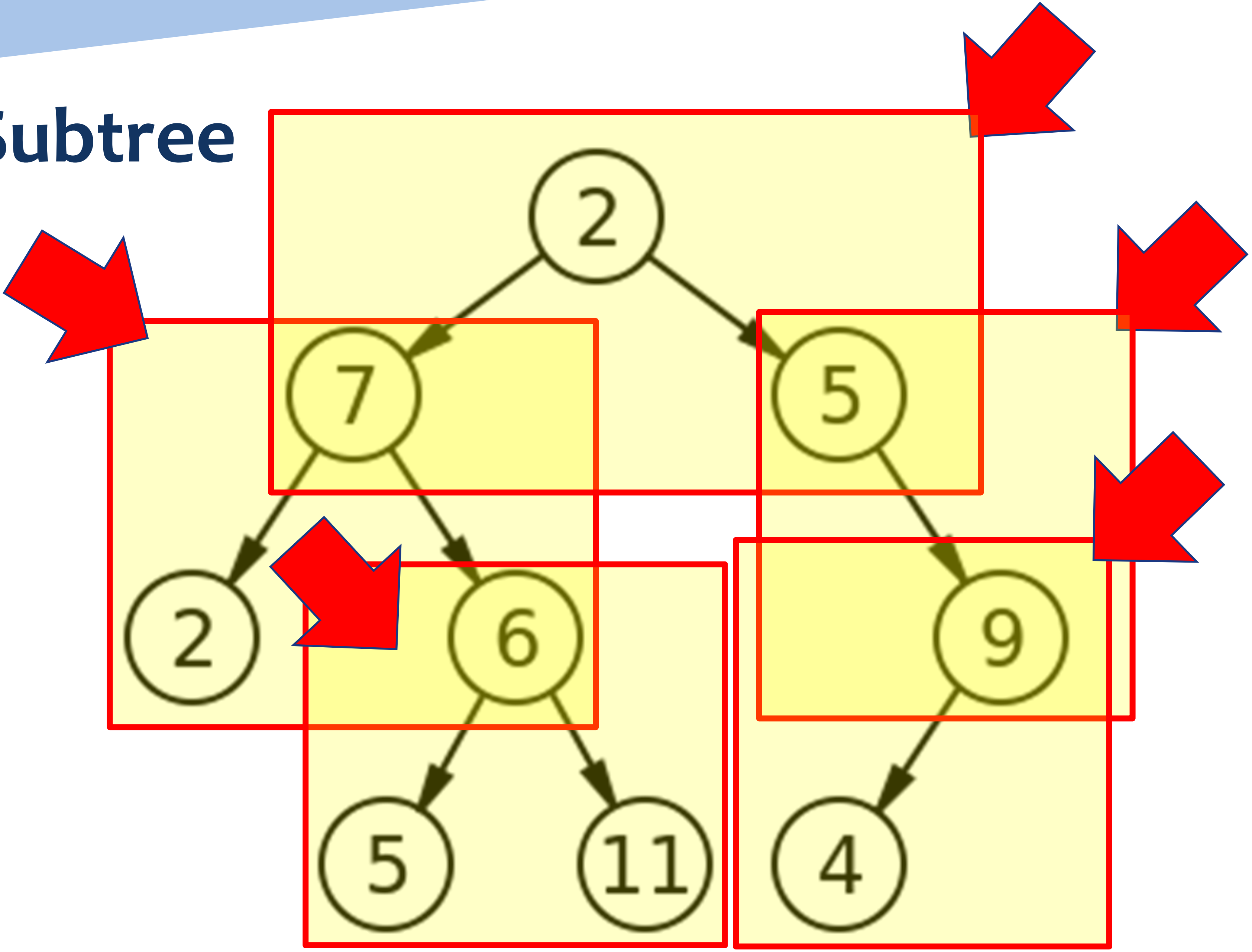
Left child



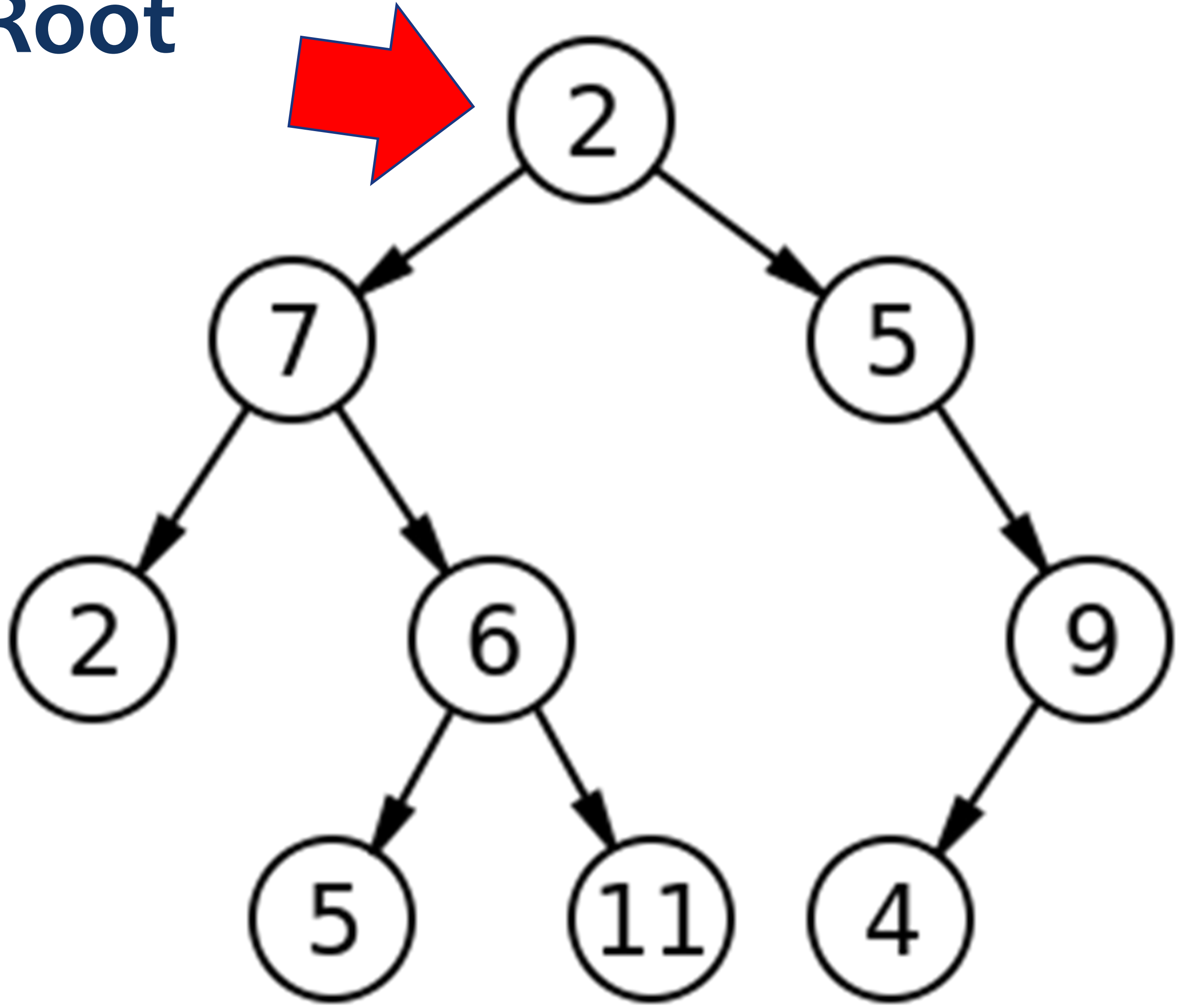
Right-child



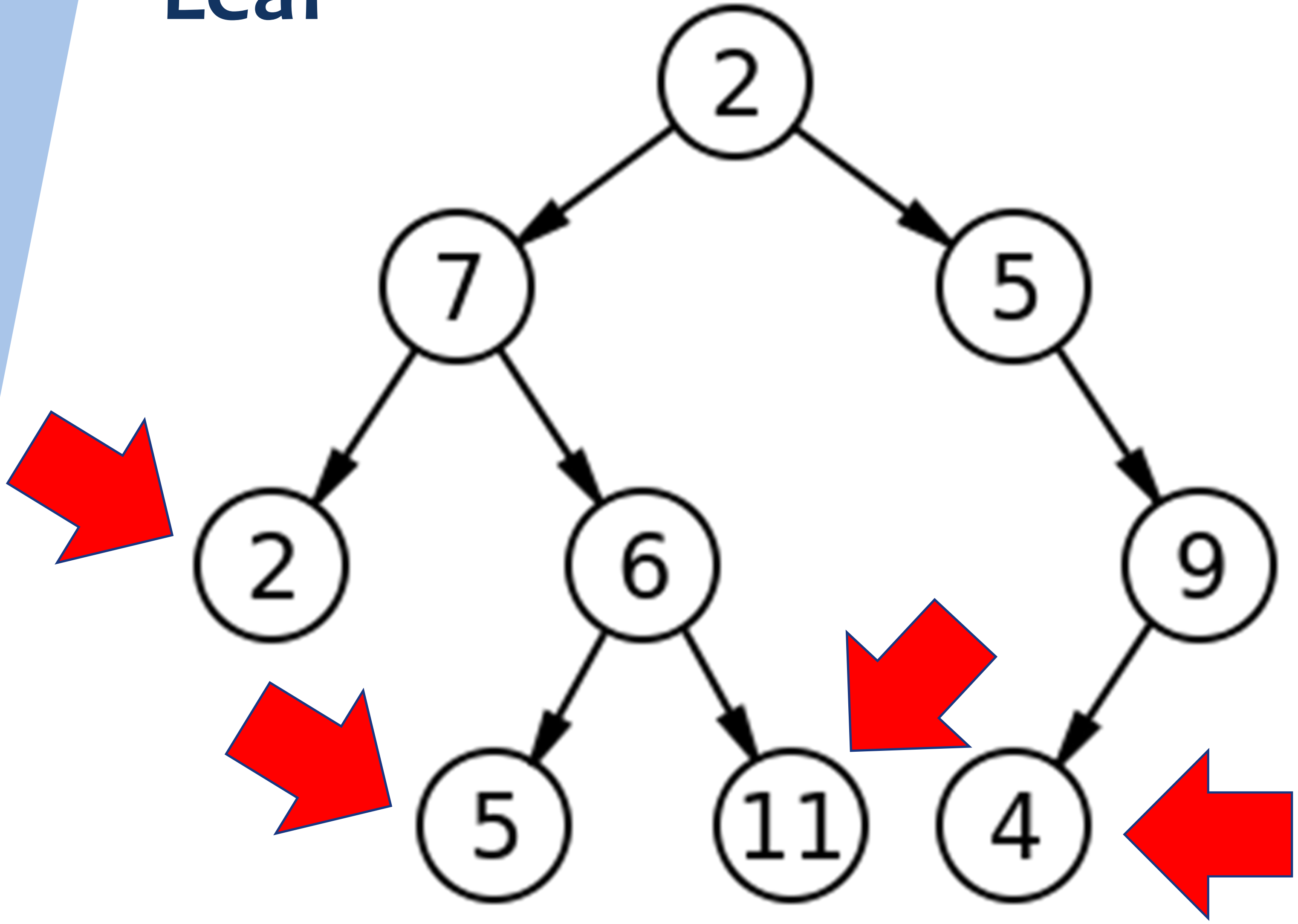
Subtree



Root



Leaf



Definitions

- **Parent**: A node in a tree that has children (left, right or both)
- **Left-child**: A node on the left hand side below a parent node
- **Right-child**: A node on the right hand side below a parent node
- **Root**: The top node in a tree
- **Subtree**: A parent with children with-in another parent-child relationship
- **Leaf**: A node with no children