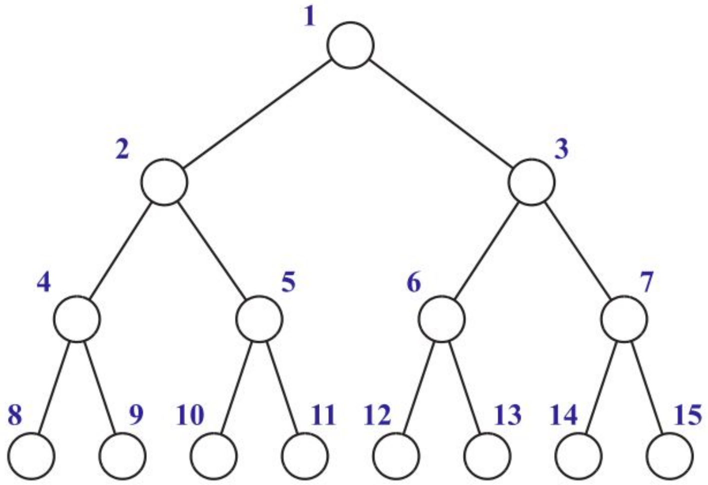
Data Structures & Programming

Binary Trees
Vector Implementation & Traversals

Golnar Sheikhshab

Vector Implementation of Binary Trees

Based on **level numbering**



- If v is the root of T, then $f(v) = 1$
- If v is the left child of node u, then $f(v) = 2f(u)$
- If v is the right child of node u, then $f(v) = 2f(u) + 1$

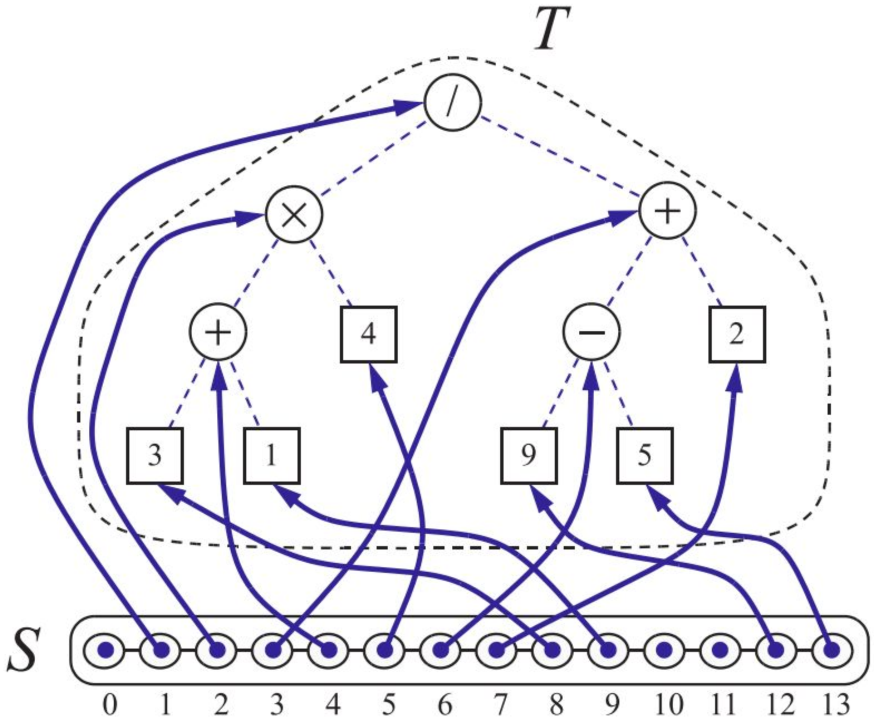


Figure 7.17: Representation of a binary tree *T* by means of a vector *S*.

Complexity analysis

Operation	*Time*
left, right, parent, isExternal, isRoot	$O(1)$
size, empty	$O(1)$
root	$O(1)$
expandExternal, removeAboveExternal	$O(1)$
positions	$O(n)$

Table 7.3: Running times for a binary tree T implemented with a vector S. We denote the number of nodes of T with n, and N denotes the size of S. The space usage is $O(N)$, which is $O(2^n)$ in the worst case.

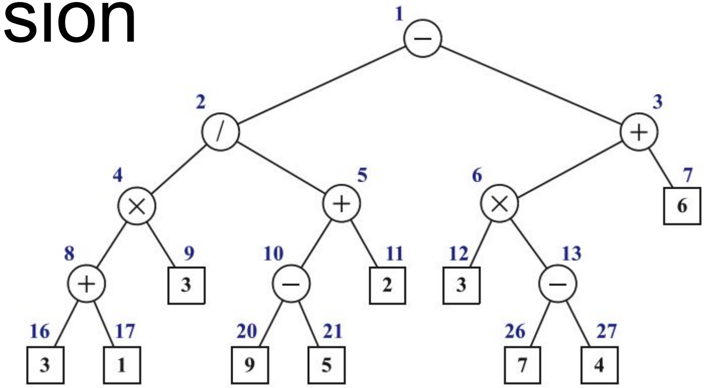
Traversals of a binary tree

Algorithm binaryPreorder(T, p):
  perform the "visit" action for node p
  if p is an internal node **then**
    binaryPreorder(T, p.left())
    binaryPreorder(T, p.right())

Algorithm binaryPostorder(T, p):
  if p is an internal node **then**
    binaryPostorder(T, p.left())
    binaryPostorder(T, p.right())
  perform the "visit" action for the node p

Algorithm inorder(T, p):
  if p is an internal node **then**
    inorder(T, p.left())
  perform the "visit" action for node p
  if p is an internal node **then**
    inorder(T, p.right())

Evaluating an Arithmetic Expression



Algorithm evaluateExpression(T, p):

  if p is an internal node **then**

    $x \leftarrow$ evaluateExpression(T, p.left())

    $y \leftarrow$ evaluateExpression(T, p.right())

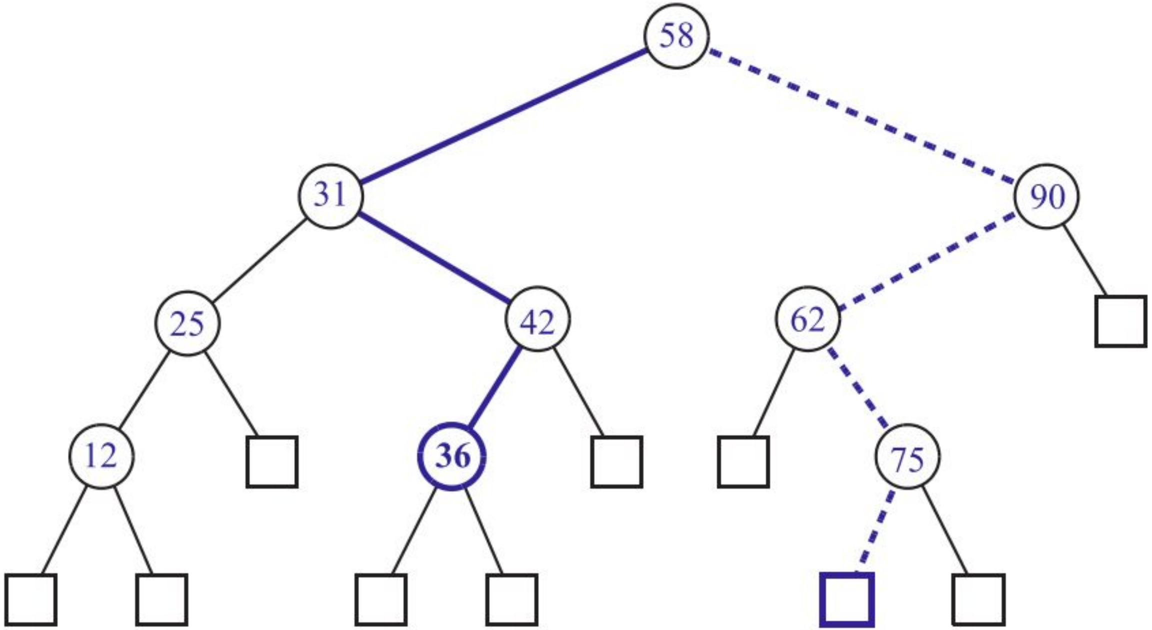
    Let $\circ$ be the operator associated with p

    return $x \circ y$

  else

    return the value stored at p

Binary Search Tree



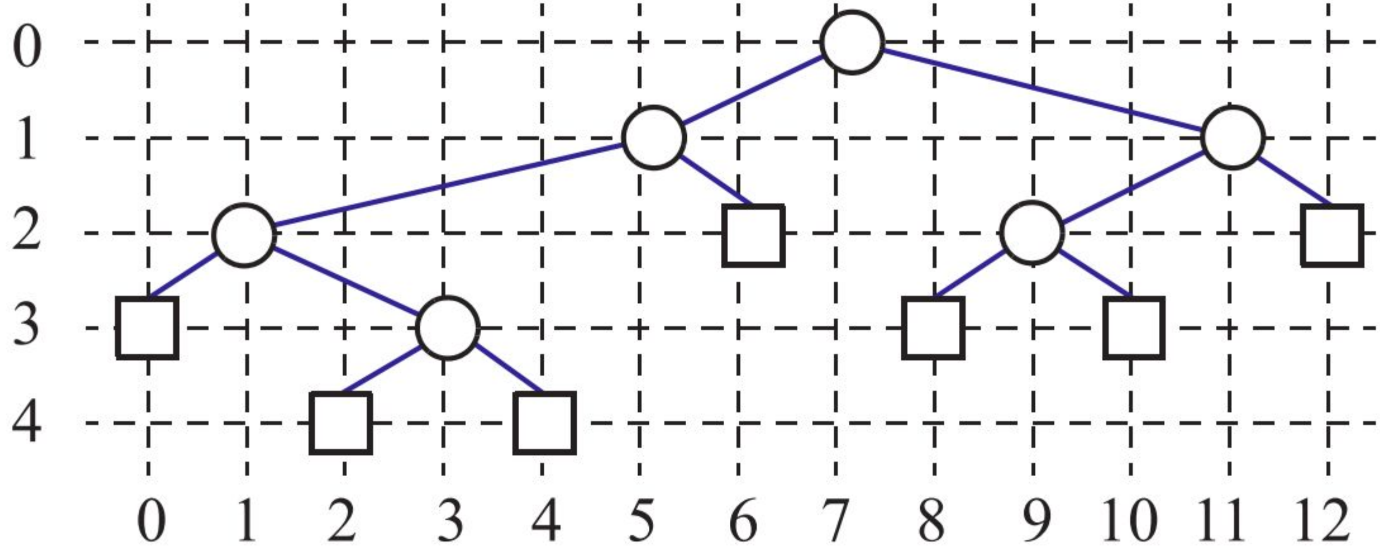
Binary Search Tree

Do we have a binary search tree when we do binary search?

Time analysis of search in a binary search tree?

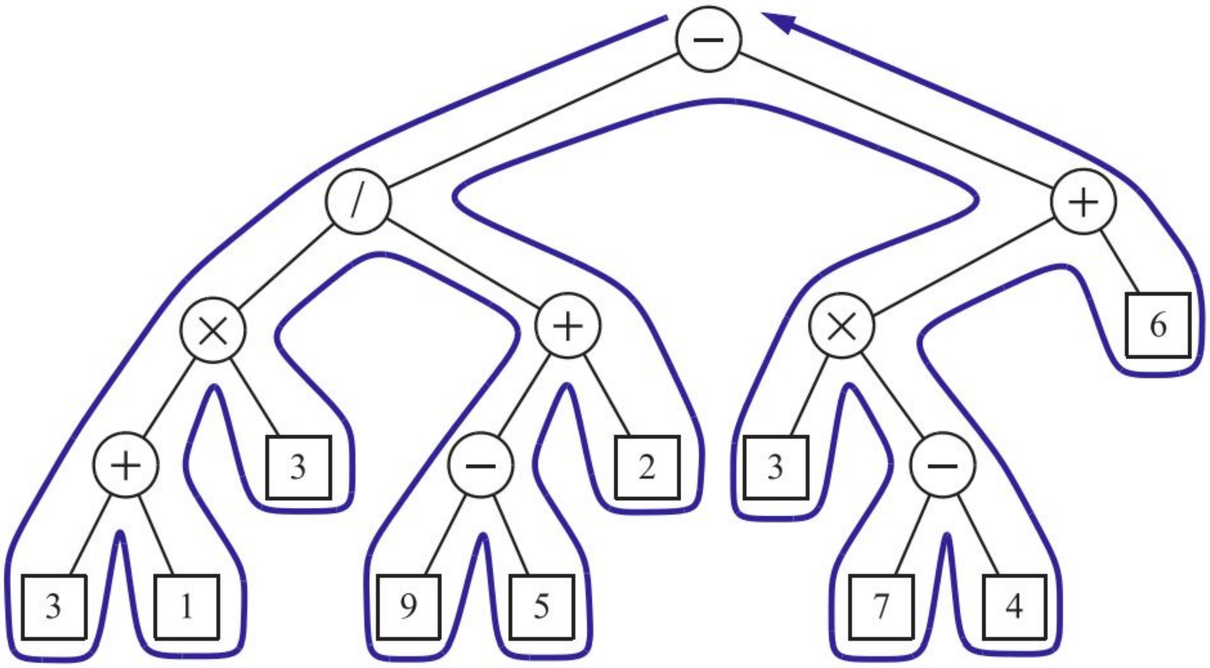
How to sort using a binary search tree?

Using Inorder Traversal for Tree Drawing



- $x(p)$ is the number of nodes visited before p in the inorder traversal of T.
- $y(p)$ is the depth of p in T.

The Euler Tour Traversal of a Binary Tree



The Euler Tour Traversal of a Binary Tree (2)

Algorithm eulerTour(T, p):
  perform the action for visiting node p on the left
  if p is an internal node **then**
    recursively tour the left subtree of p by calling eulerTour(T, p.left())
  perform the action for visiting node p from below
  if p is an internal node **then**
    recursively tour the right subtree of p by calling eulerTour(T, p.right())
  perform the action for visiting node p on the right

The Euler Tour Traversal of a Binary Tree (3)

Algorithm templateEulerTour(T, p):
  $r \leftarrow$ initResult()
  if p.isExternal() **then**
    r.finalResult $\leftarrow$ visitExternal(T, p, r)
  else
    visitLeft(T, p, r)
    r.leftResult $\leftarrow$ templateEulerTour(T, p.left())
    visitBelow(T, p, r)
    r.rightResult $\leftarrow$ templateEulerTour(T, p.right())
    visitRight(T, p, r)
  return returnResult(r)

Reading Material

Sections 7.3.5 and 7.3.6 of the textbook

Optional: sections 7.3.6 (The Template Function Pattern) mainly to see an instance of using class inheritance