CSE221

Binary Trees

Fall 2021

Young-ri Choi

Acknowledgment: The content of this file is based on the slides of the textbook as well as the slides provided in former lectures at UNIST.

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Outline

- Binary tree traversal
- Counting binary trees
- Threaded binary trees

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

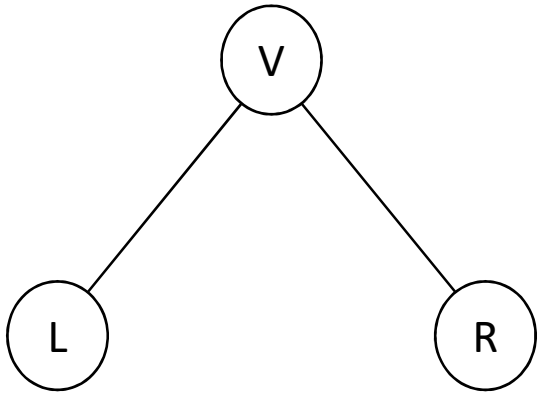
Binary Tree Traversal

- Each node is visited only once
- When a node is visited, some operation (e.g., printout) is performed on it
- After traversal, all nodes in the tree are visited in a linear order and only once

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Binary Tree Traversal

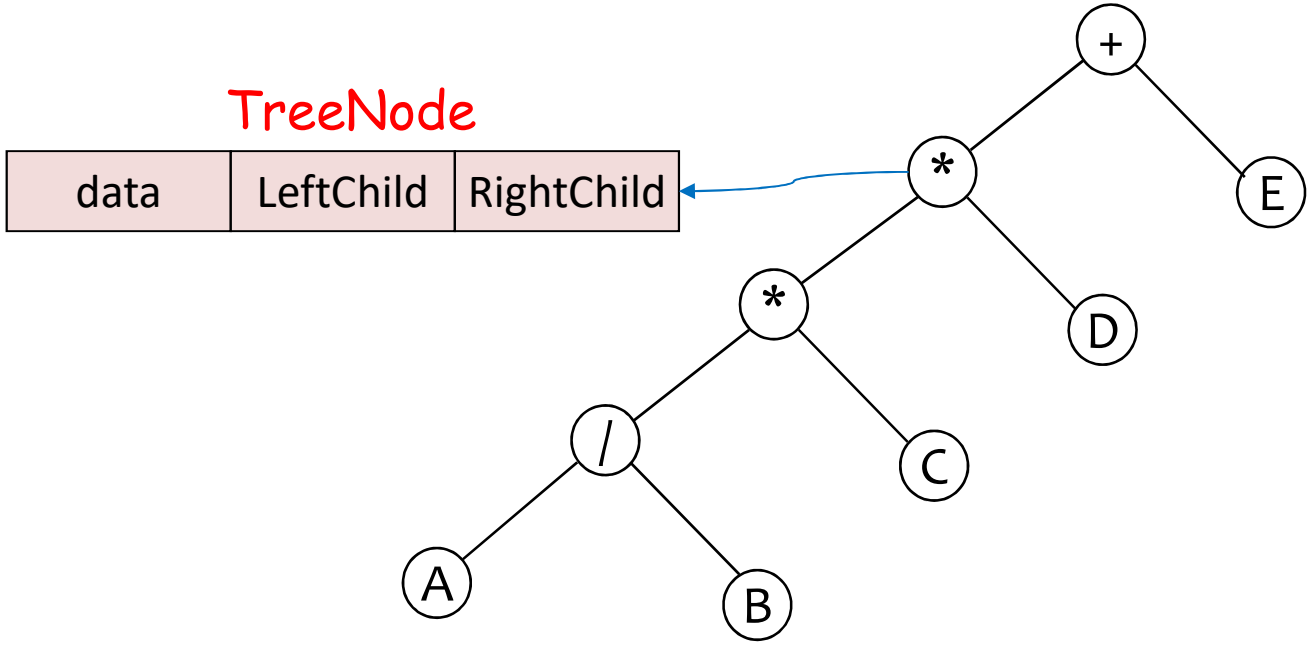
- On a node
  - L : moving to left child node
  - V : visiting current node
  - R : moving to right child node
- Available traversal order
  - LVR, LRV, VLR, VRL, RVL, RLV
- Traverse left before right
  - LVR : inorder
  - VLR : preorder
  - LRV : postorder



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Binary Tree Traversal

- Binary tree with arithmetic expression
  - Internal nodes: operators
  - Leaf nodes: operands



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

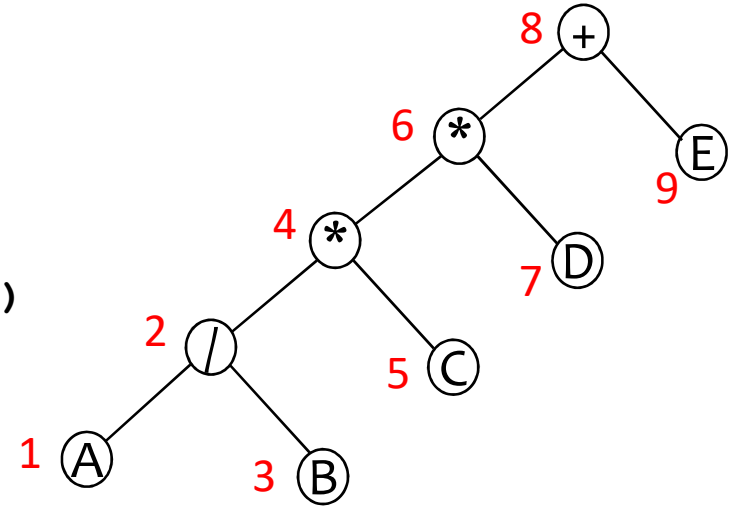
Inorder Traversal

- LVR

```
void Tree::inorder()
{
    inorder(root);
}

void Tree::inorder(TreeNode *CurrentNode)
{
    if (CurrentNode) {
        inorder(CurrentNode->LeftChild);
        cout << CurrentNode->data;
        inorder(CurrentNode->RightChild);
    }
}
```

A/B*C*D+E



Output : A/B*C*D+E

(infix notation)

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

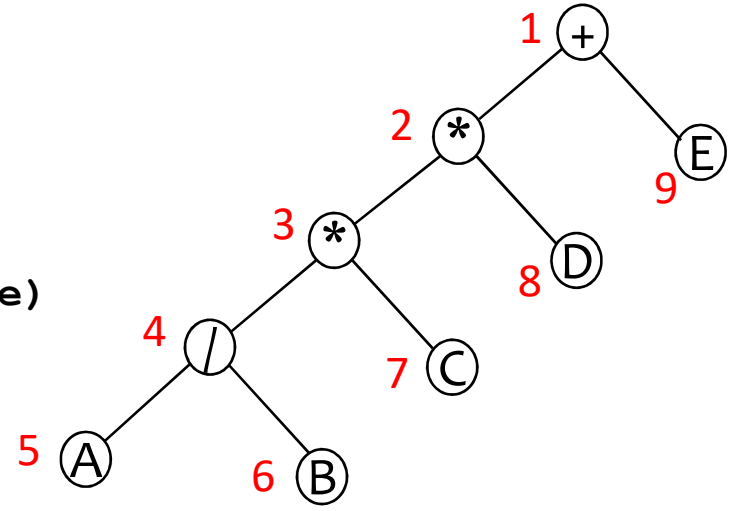
Preorder Traversal

- VLR

```
void Tree::preorder()
{
   preorder(root);
}

void Tree::preorder(TreeNode *CurrentNode)
{
   if (CurrentNode) {
      cout << CurrentNode->data;
      preorder(CurrentNode->LeftChild);
      preorder(CurrentNode->RightChild);
   }
}
```

A/B*C*D+E



Output : +**/ABCDE

(prefix notation)

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

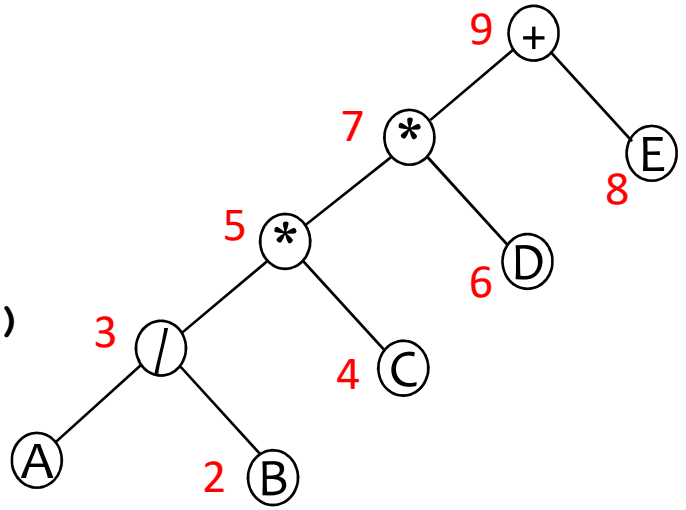
Postorder Traversal

- LRV

```
void Tree::postorder()
{
   postorder(root);
}

void Tree::postorder(TreeNode *CurrentNode)
{
    if (CurrentNode){
       postorder(CurrentNode->LeftChild);
       postorder(CurrentNode->RightChild);
       cout << CurrentNode->data;
    }
}
```

A/B*C*D+E



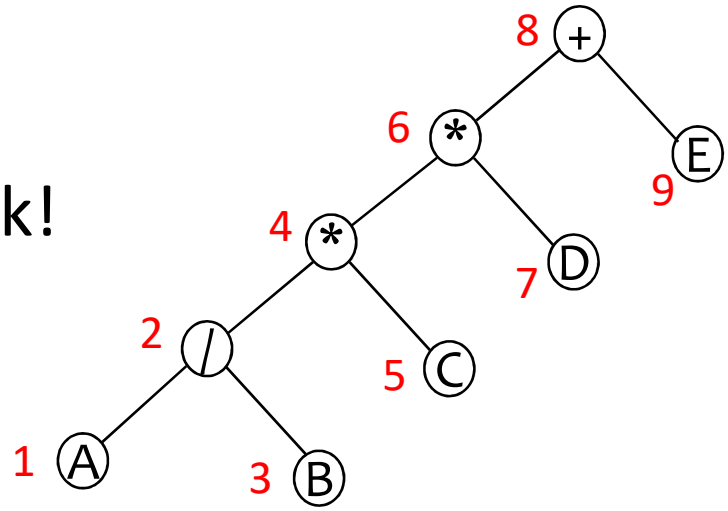
Output : AB/C*D*E+

(postfix notation)

UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Non-recursive Inorder Traversal

- Use a stack
  - Recursion implicitly uses a stack!




```
void Tree::NonrecInorder()
{
   Stack<TreeNode *> s;
   TreeNode *CurrentNode = root;
   while(1){
         // move down left child
         while(CurrentNode) {
             s.Push(CurrentNode);
             CurrentNode = CurrentNode->LeftChild;
         }

         if (s.IsEmpty()) return;

         CurrentNode = s.Top();
         s.Pop();
         Visit(CurrentNode);
         CurrentNode = CurrentNode->RightChild;
   }
}
```

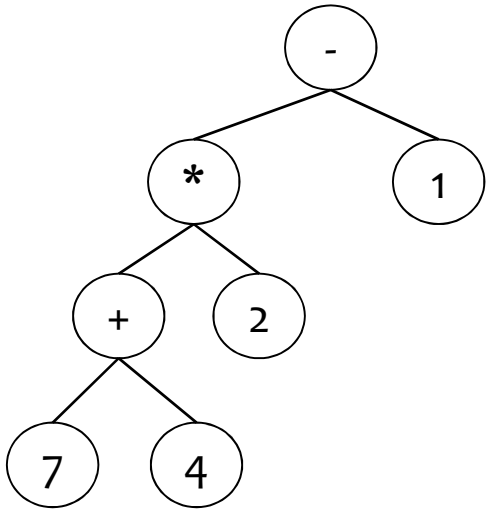
UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Building Expression Tree

- Overall procedure

(7 + 4) * 2 - 1

7 4 + 2 * 1 -

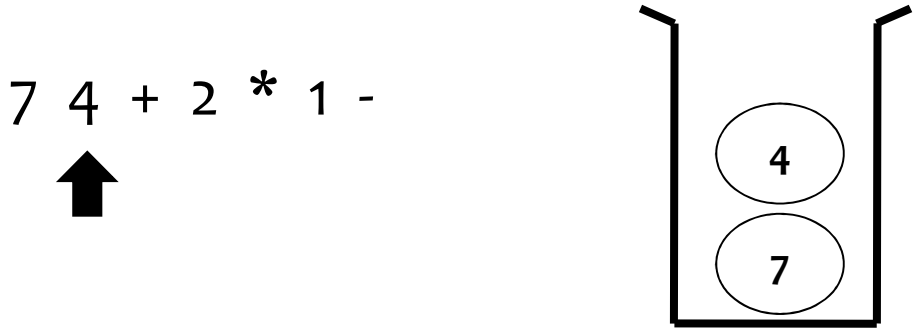


- Infix notation → postfix notation
  - Use a stack (see lecture note 4)
- Postfix notation → expression tree
  - Build a tree incrementally using a stack

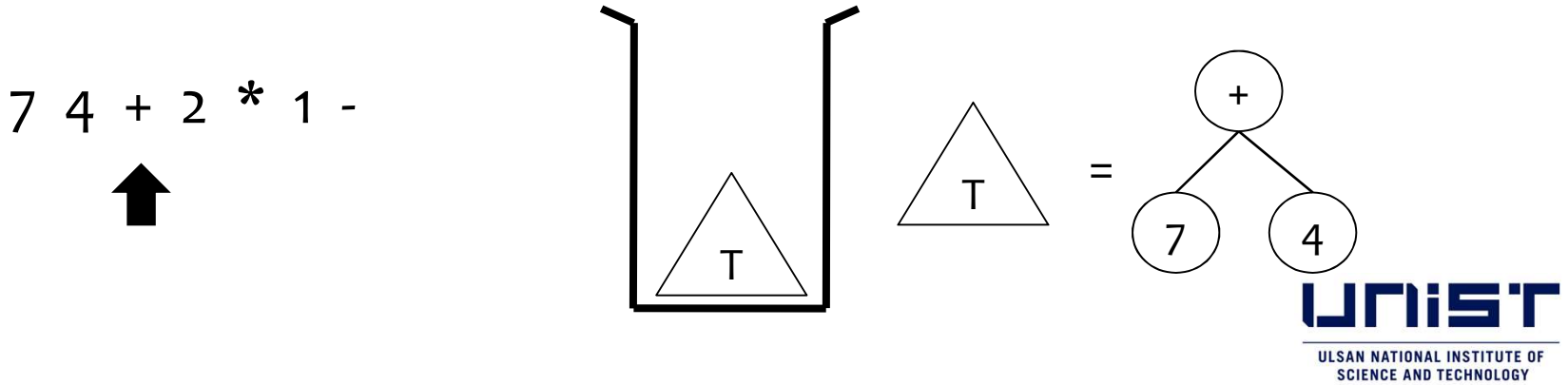
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Building Expression Tree

- Push nodes of operands until finding an operator



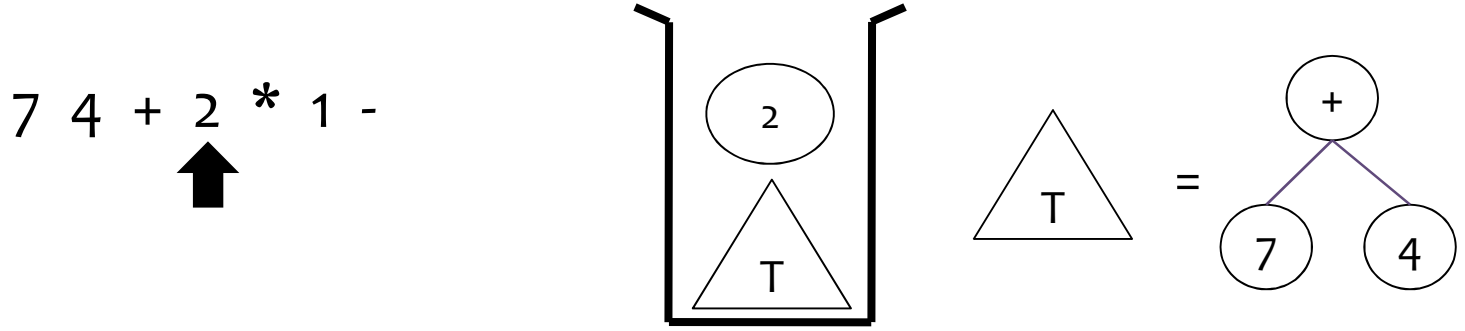
- Pop two nodes and push a partial expression tree



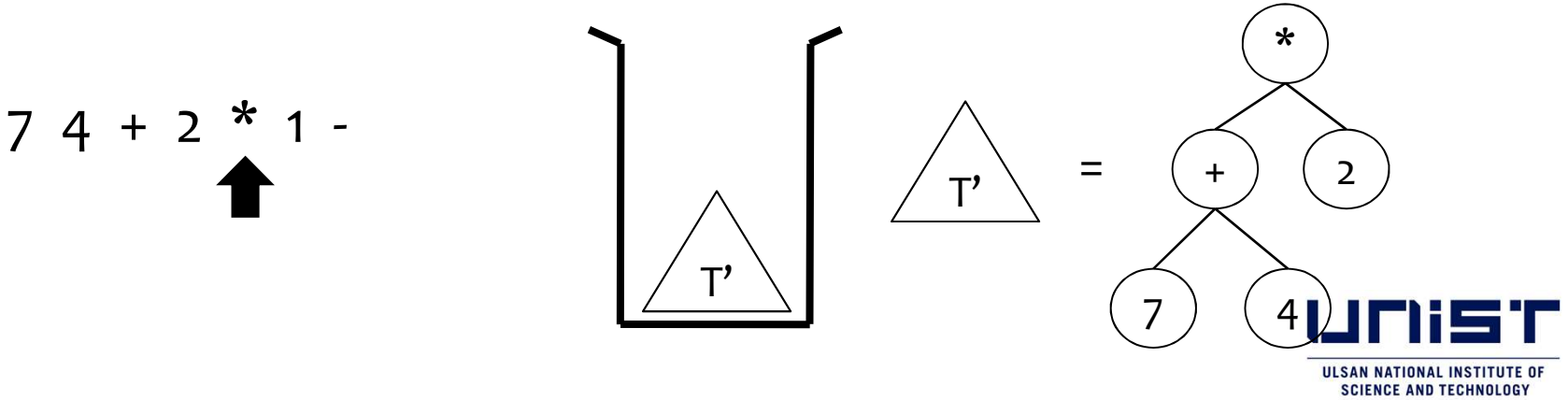
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Building Expression Tree

- Push nodes of operands until finding an operator



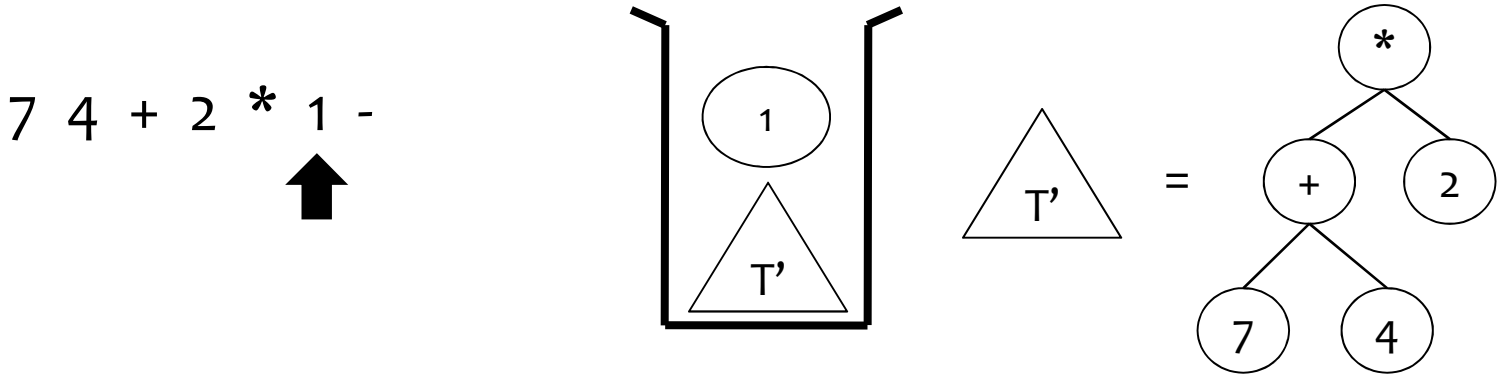
- Pop two nodes and push a partial expression tree



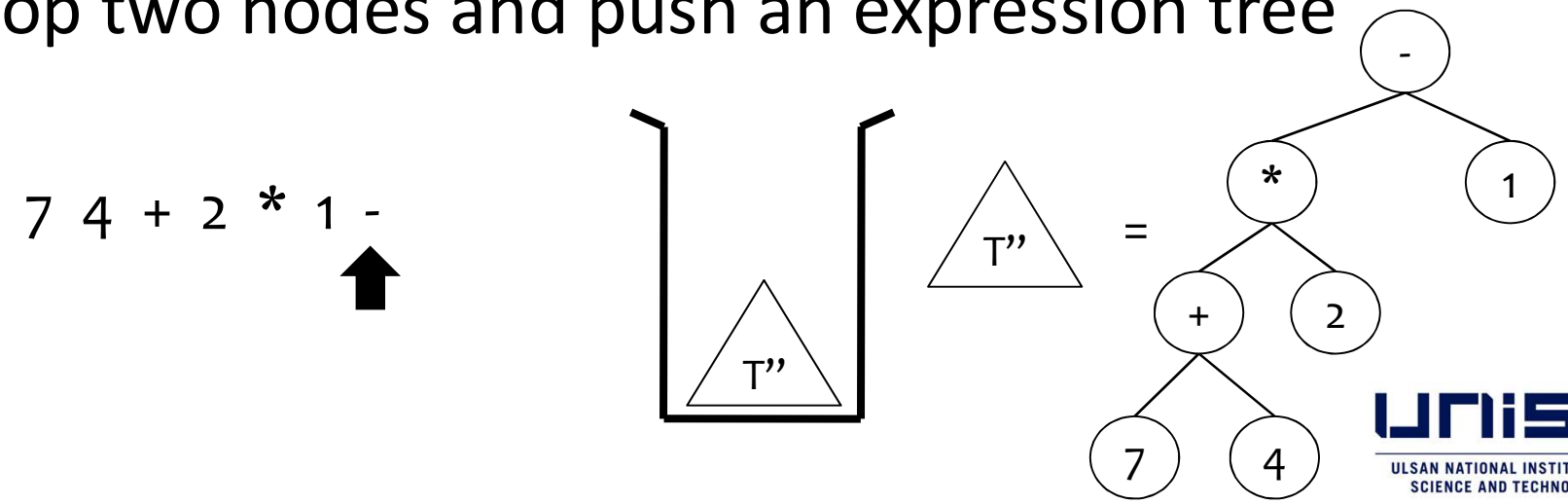
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Building Expression Tree

- Push nodes of operands until finding an operator



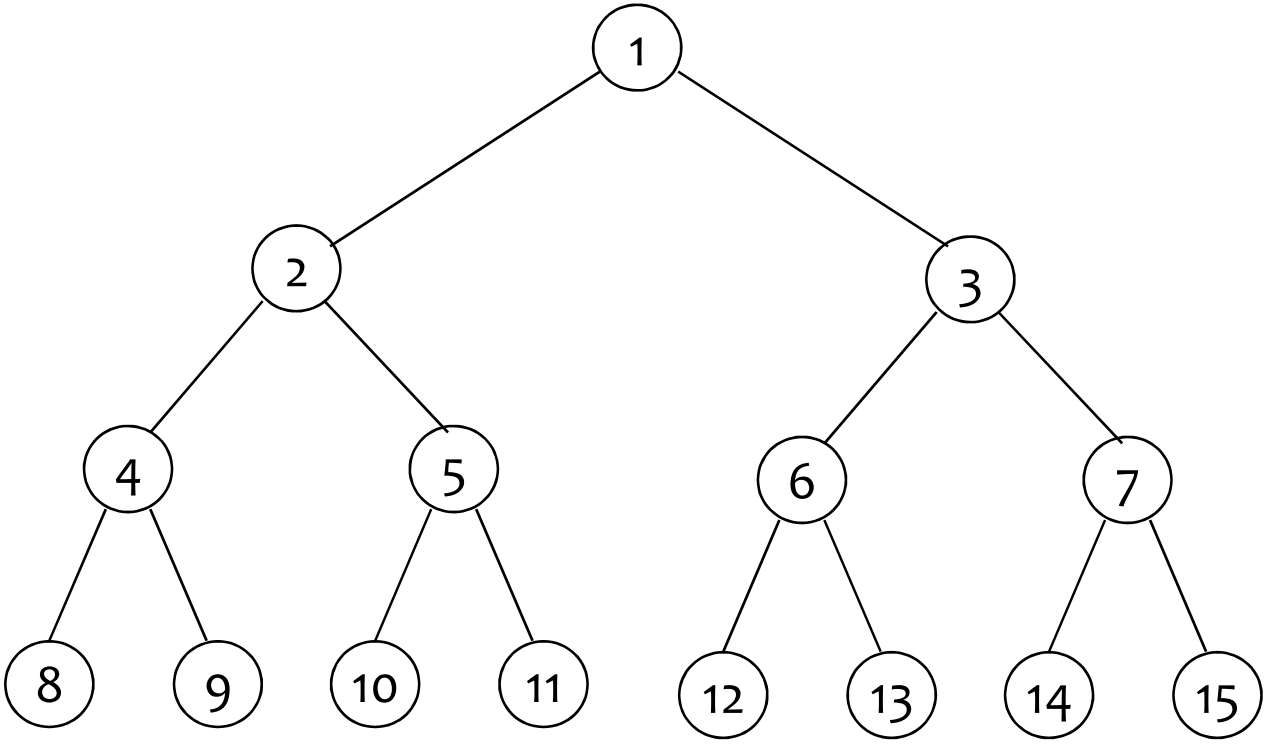
- Pop two nodes and push an expression tree



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

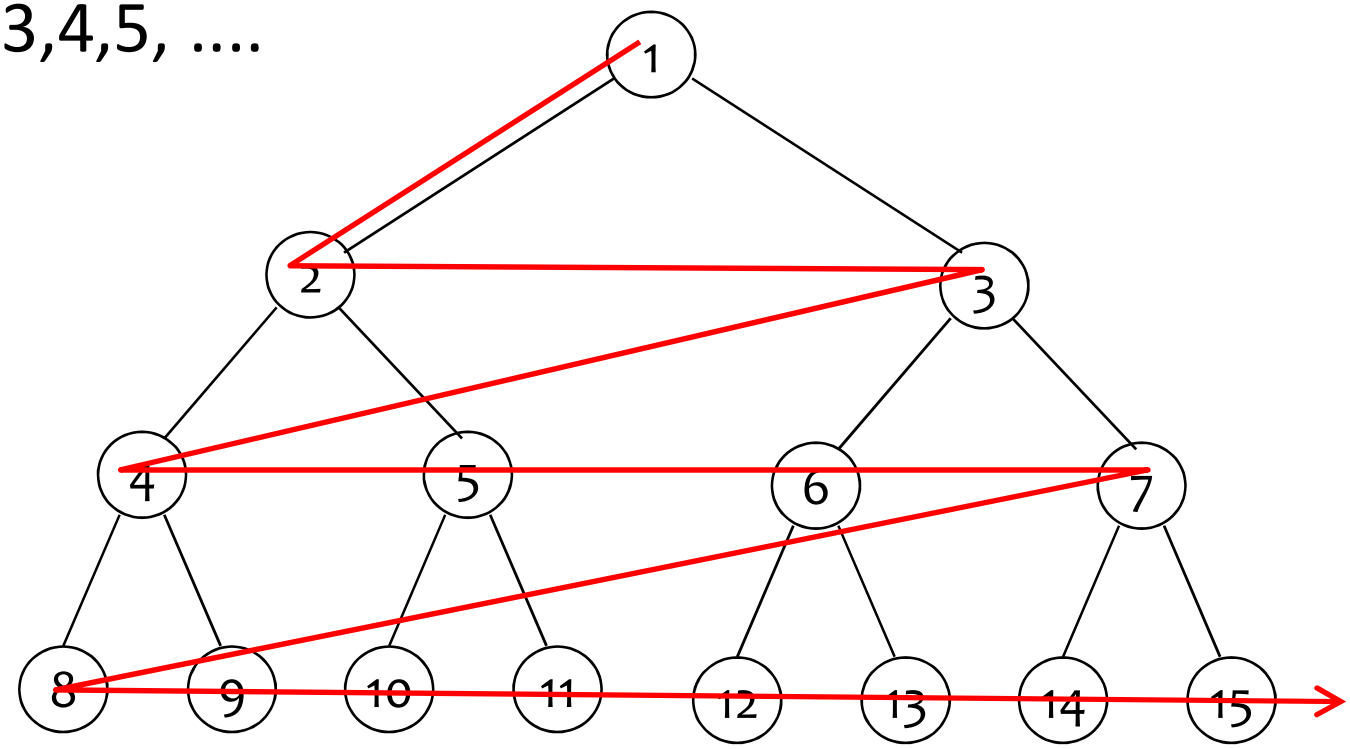
- Traverse index order



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

- Traverse index order
  - 1,2,3,4,5, ...



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

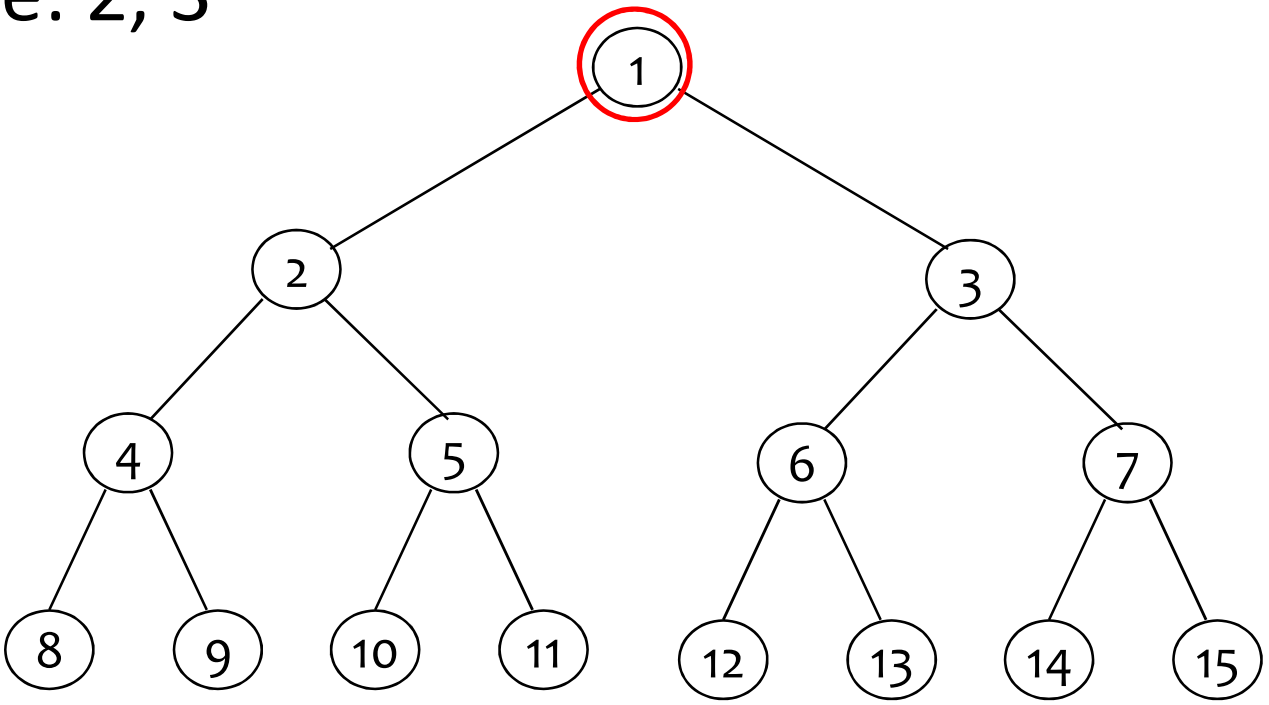
- Using a queue
  - Visit current node, push two children (left, right)

```
void Tree::LevelOrder()
{
    Queue<TreeNode*> q;
    TreeNode *CurrentNode = root;
    while(CurrentNode) {
        Visit(currentNode);
        if (CurrentNode->LeftChild)
            q.push(CurrentNode->LeftChild);
        if (CurrentNode->RightChild)
            q.push(CurrentNode->RightChild);
        if(q.IsEmpty()) return;
        CurrentNode = q.Front();
        q.Pop();
    }
}
```

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

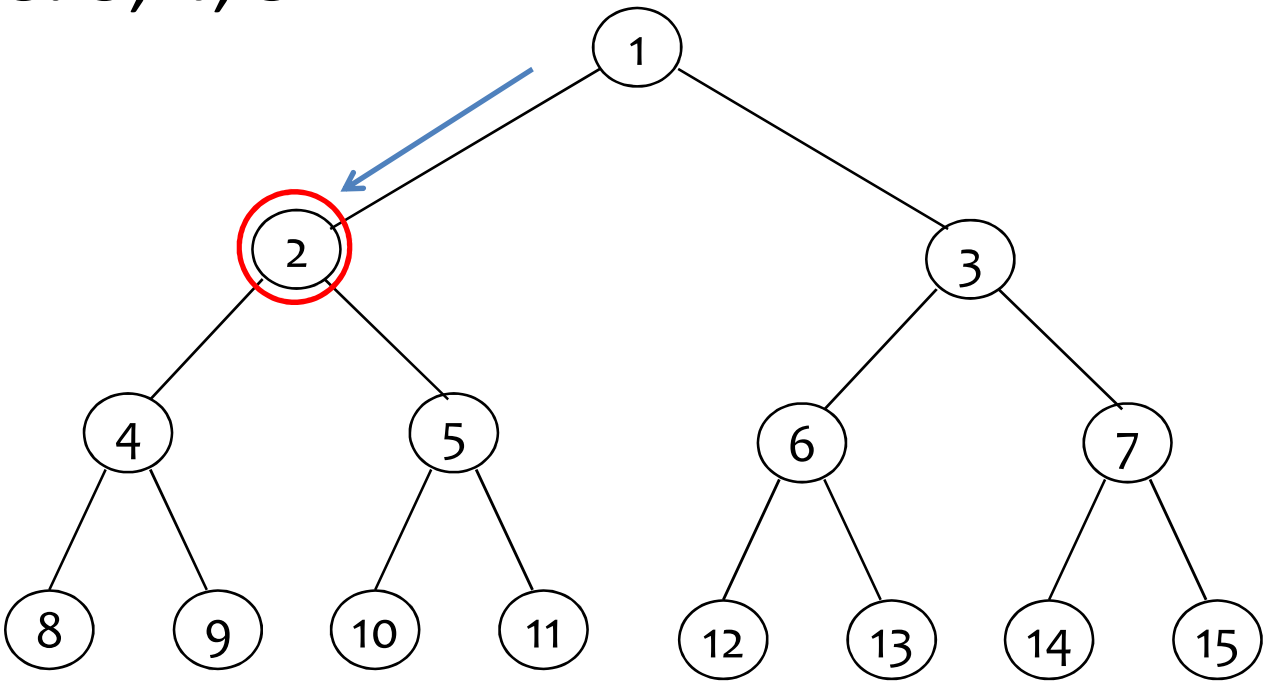
- Visit: 1
- Queue: 2, 3



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

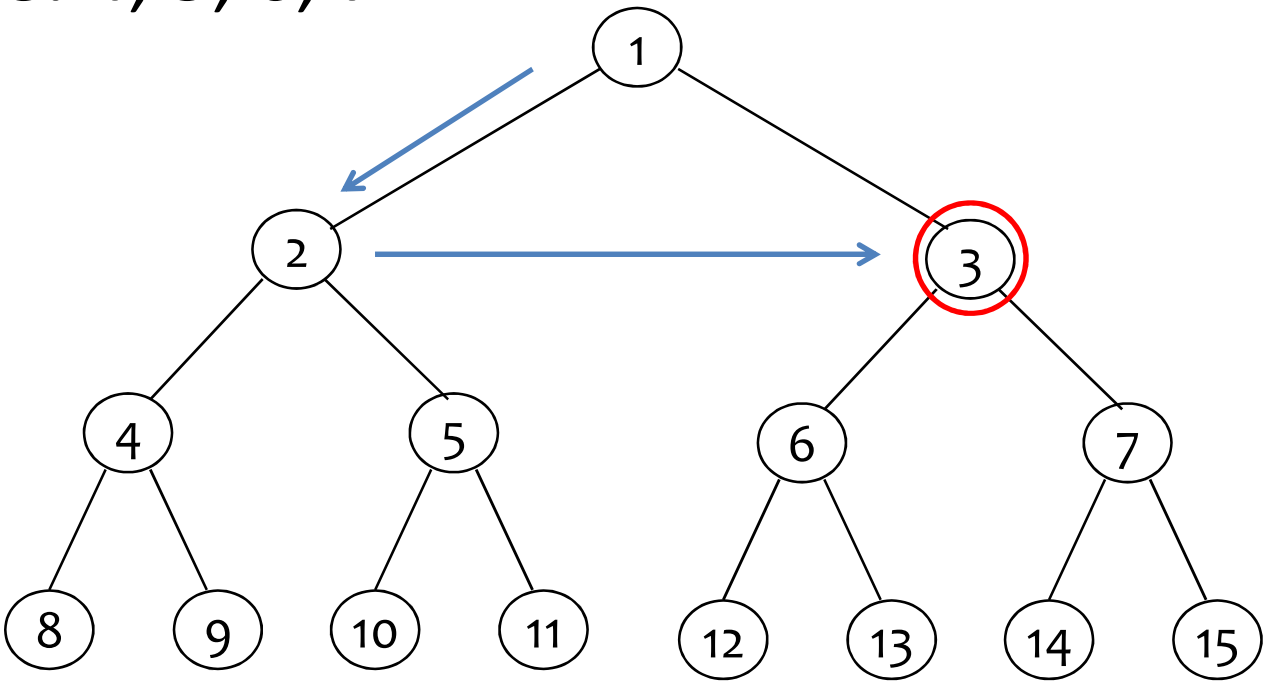
- Visit: 1, 2 (pop)
- Queue: 3, 4, 5



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

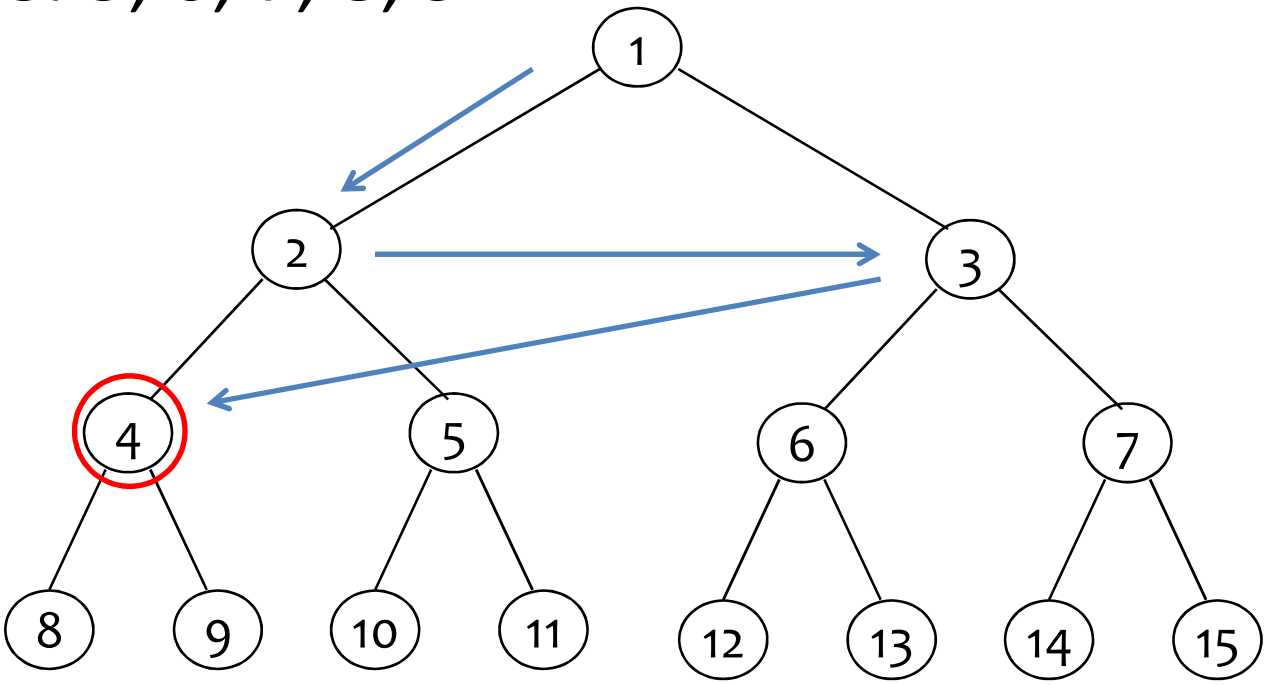
- Visit: 1, 2, 3 (pop)
- Queue: 4, 5, 6, 7



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Level-order Traversal

- Visit: 1, 2, 3, 4 (pop)
- Queue: 5, 6, 7, 8, 9



UNIST
ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Testing Equality

```
// Driver
bool operator==(const Tree& s, Tree& t)
{
   return equal(s.root, t.root);
}

// Workhorse
bool equal(TreeNode *a, TreeNode *b)
{
   if ((!a) && (!b)) return true;  // both a and b are null
   if (a && b
        &&  (a->data == b->data)  // data is equal
        &&  equal(a->LeftChild, b->LeftChild)  // left subtrees equal
        &&  equal(a->RightChild, b->RightChild)) // right subtrees equal
      return true;
   return false;
}
```

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

- Propositional calculus
  - Formulas defined by boolean variables x_1, x_2, ..., x_n and operators $\wedge(\text{and}),\ \vee(\text{or}),\ \neg(\text{not})$
- Satisfiability problem
  - Finding out values for the variables that result in the formula being *true*
  - *e.g.,* $x_1 \vee (x_2 \wedge \neg x_3)$ is true if

    $x_1 = false$

    $x_2 = true$

    $x_3 = false$

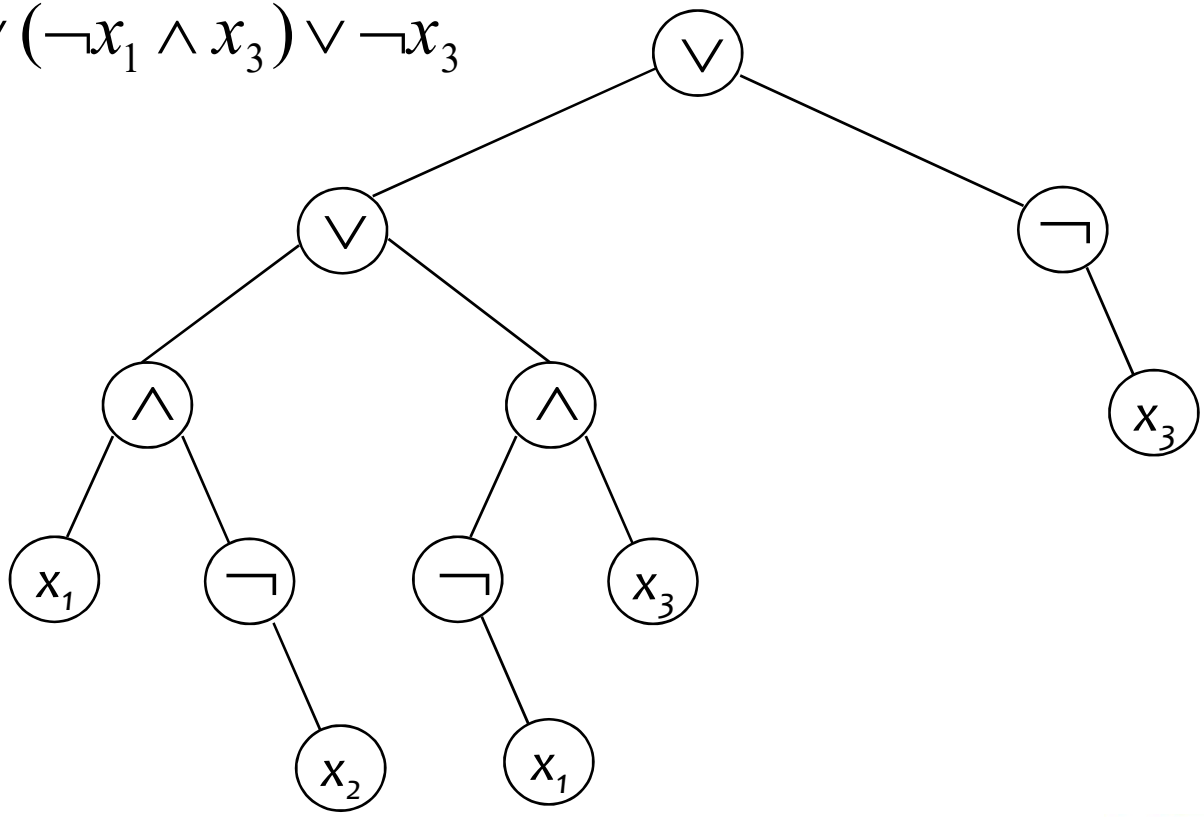
Complexity?

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

- Propositional formula in a binary tree

$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$$



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

- Node structure

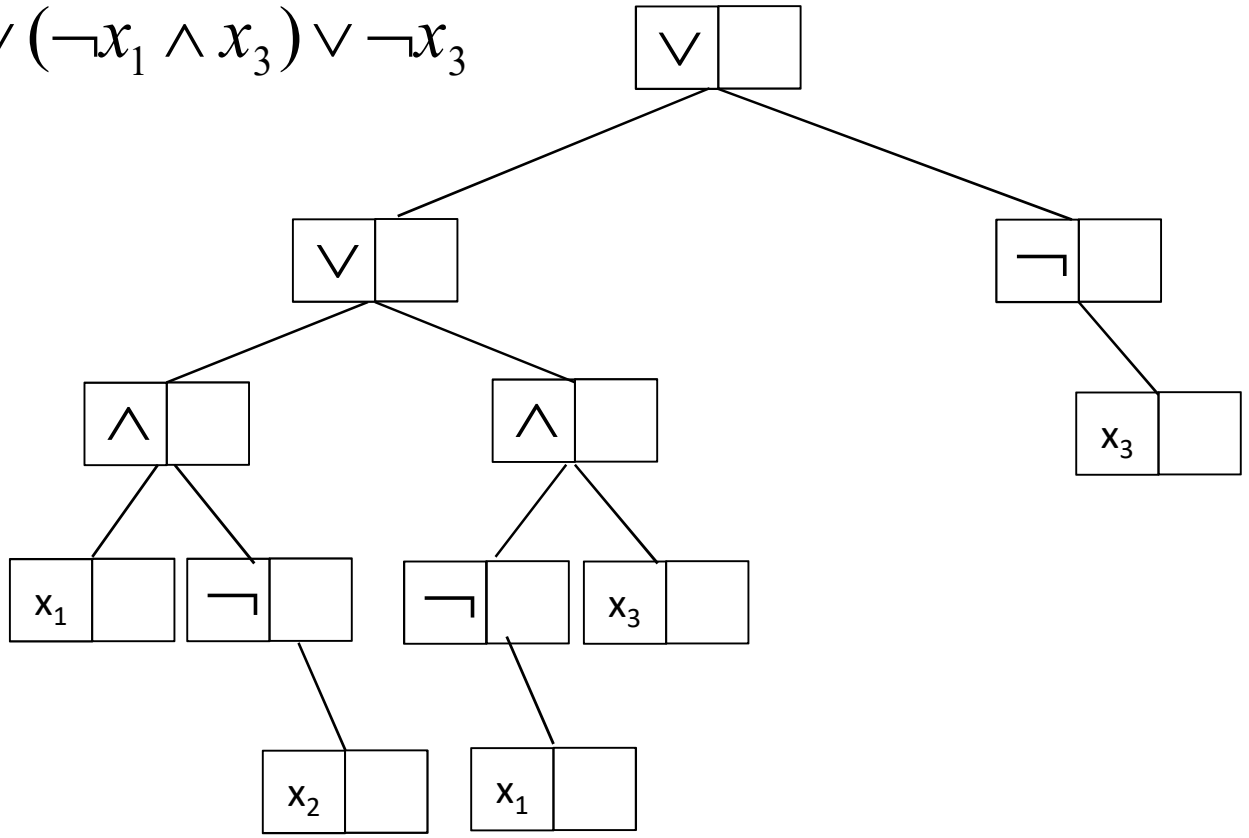
Leftchild	first	second	Rightchild

  - first: given values for formular
    - Operators($\wedge$, $\vee$, $\neg$), True/False values
  - second
    - True/False values after evaluation

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

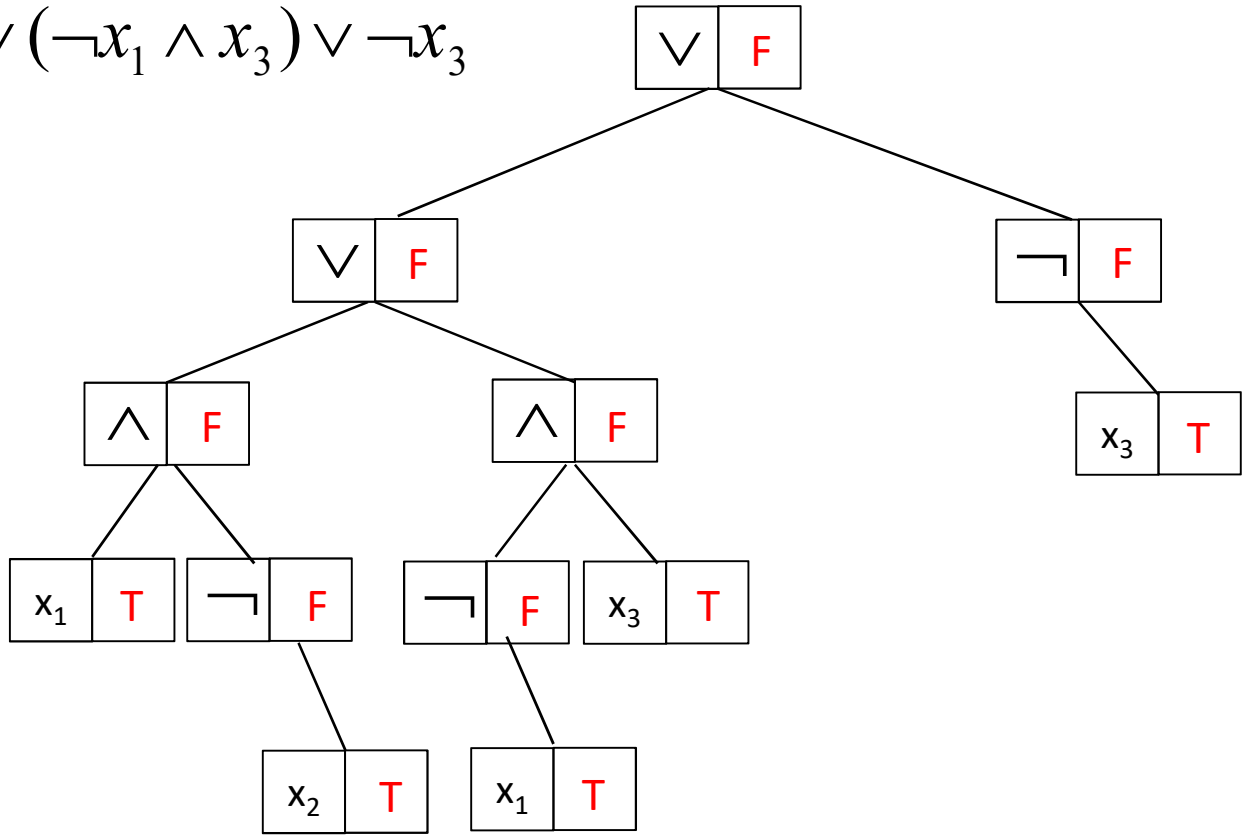
$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$$



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Example

$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \neg x_3$$



ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

- Algorithm

```
for each of 2^n possible truth value combinations for the n variables
{
  replace data.first by the values in the current combination;
  evaluate the formula by traversing the tree in postorder;
  if (root.data.second)
  {
    cout << combination;
    return;
  }
}
cout << "no satisfiable combination";
```

why?

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

The Satisfiability Problem

- Evaluate propositional formula

```
void SatTree::PostOrderEval() {  // Driver
          PostOrderEval(root);
}

void SatTree::PostOrderEval(SatNode *s) // Workhorse
{
   if (s) {
      PostOrderEval(s->LeftChild);
      PostOrderEval(s->RightChild);  // postorder – left & right subtrees are evaluated
      switch(s->data.first) {
         case Not: s->data.second = !s->RightChild->data.second; break;
         case And: s->data.second =
              s->RightChild->data.second && s->LeftChild->data.second;
              break;
         case Or: s->data.second =
              s->RightChild->data.second || s->LeftChild->data.second;
              break;
         case True: s->data.second = TRUE; break;
         case False: s->data.second = FALSE;
      }
   }
}
```

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY