Data structures and algorithms

Tutorial 6

Amr Keleg

Faculty of Engineering, Ain Shams University

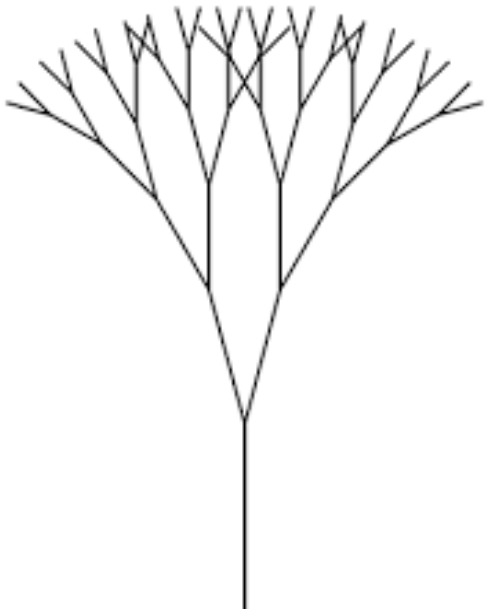
March 27, 2019

Contact: amr_mohamed@live.com

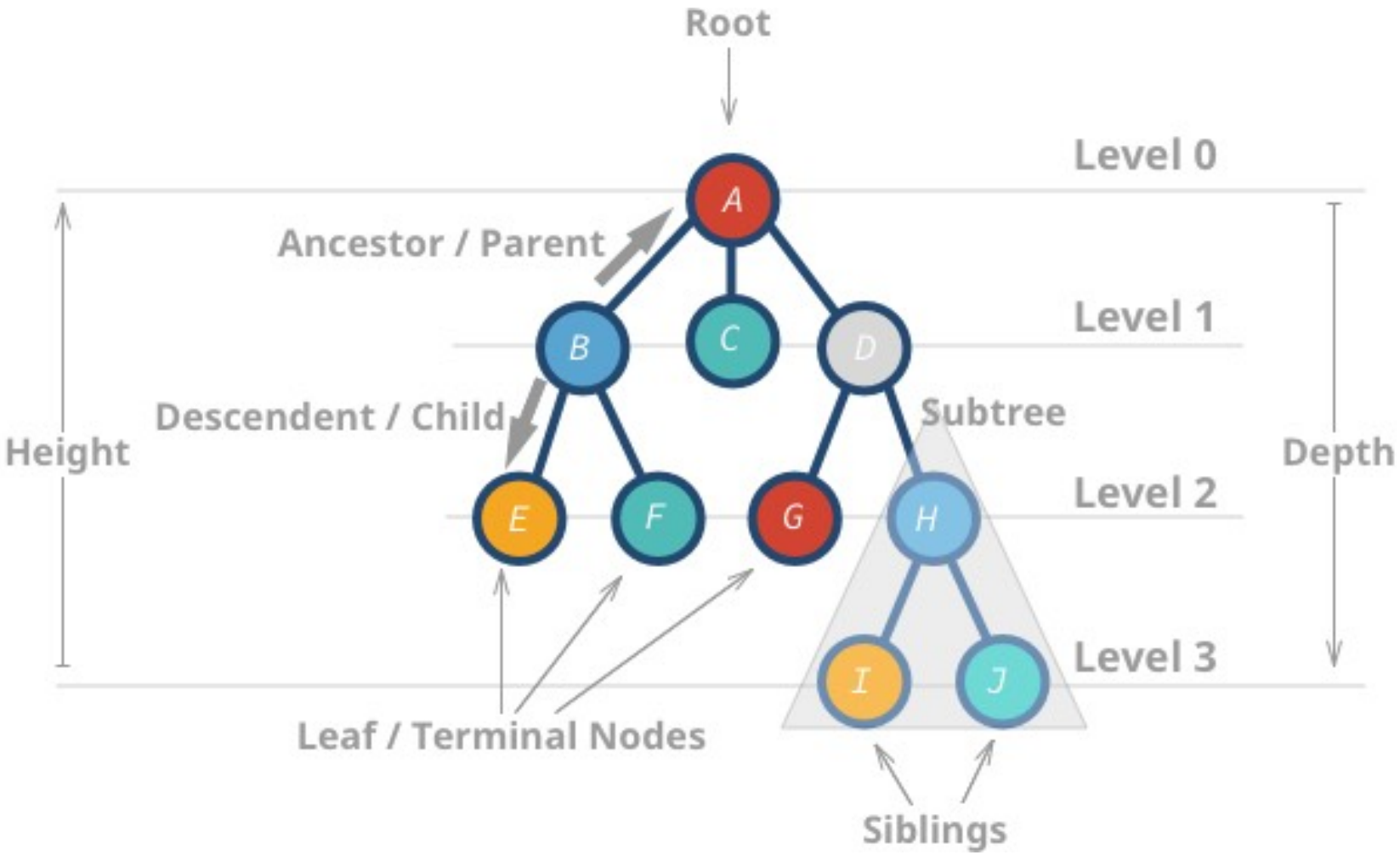
Outline

Outline

In real-life, a tree has roots and leaves.



In computer science, the tree is inverted.



- The important feature of a tree is that each node has one and only one parent except for one special node (WHICH IS?).
- So if we have a tree with n nodes, What is the number of edges in the tree?
- This fact implies an intersting feature of the tree, There is a unique path from the root to any node of the tree.

Outline

```
#define MAX_NO_OF_CHILDREN 100

class Node{
  /* datatype can be a primitive type (int- ..)
  or an object */
  datatype data;

  Node * children[MAX_NO_OF_CHILDREN];
  // OR a linked list of children
  list<Node *> children;
};

class Tree{
  Node * root;

  // And a set of methods to add/ delete/ print / ...
};
```

Outline

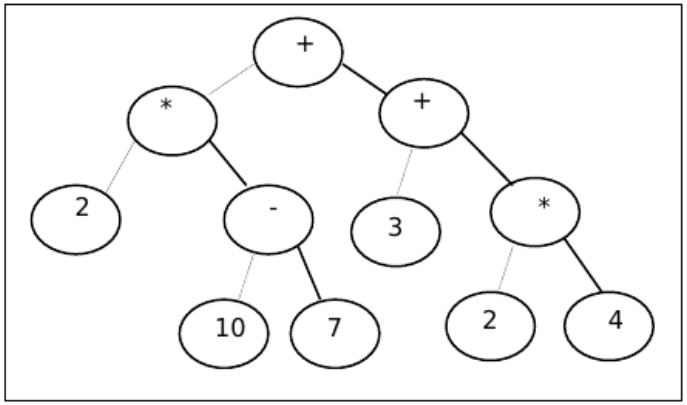
Outline

A binary tree is a tree where each node has either 0 or 2 children.

Outline

Q1. The following Binary tree holds the mathematical expression,

a) Traverse the tree using in-order, pre-order, post-order
b) Manually evaluate the numeric value of the expression held by the tree
c) Write down a recursive function to evaluate the whole tree or sub-tree of it.



```
#include<cstdlib> // for atoi
class Node{
  public:
    string value;
    Node * left;
    Node * right;

    bool is_numeric(){
      for (auto c: value){
        if (! (c>='0' && c<='9'))
          return 0;
      }
      return 1;
    }

    int to_numeric(){
      return atoi(value.c_str());
    }
};
```

```
int eval(int left_operand, int right_operand,
      string operator){
  if (operator == "+")
    return left_operand + right_operand;
  if (operator == "-")
    return left_operand - right_operand;

  // .....
}
```

```
int evaluate(Node * subtree_root){
  if(subtree_root->is_numeric())
    return subtree_root->to_numeric();

  // Evaluate the left and right subtrees
  int left_subtree = evaluate(subtree_root->left);
  int right_subtree = evaluate(subtree_root->right);

  return eval(left_subtree, right_subtree,
        subtree_root->value);
};
```

Outline

Write a method that counts the number of [leaf] nodes in a binary tree.

```
class Tree{
  public:
    int number_of_leaves();
};
```

Let's add a new private method to count the leaves in a subtree.

```
class Tree{
  private:
    int count_leaves_in_subtree(Node * subtree_root);
  public:
    int number_of_leaves();
};
```

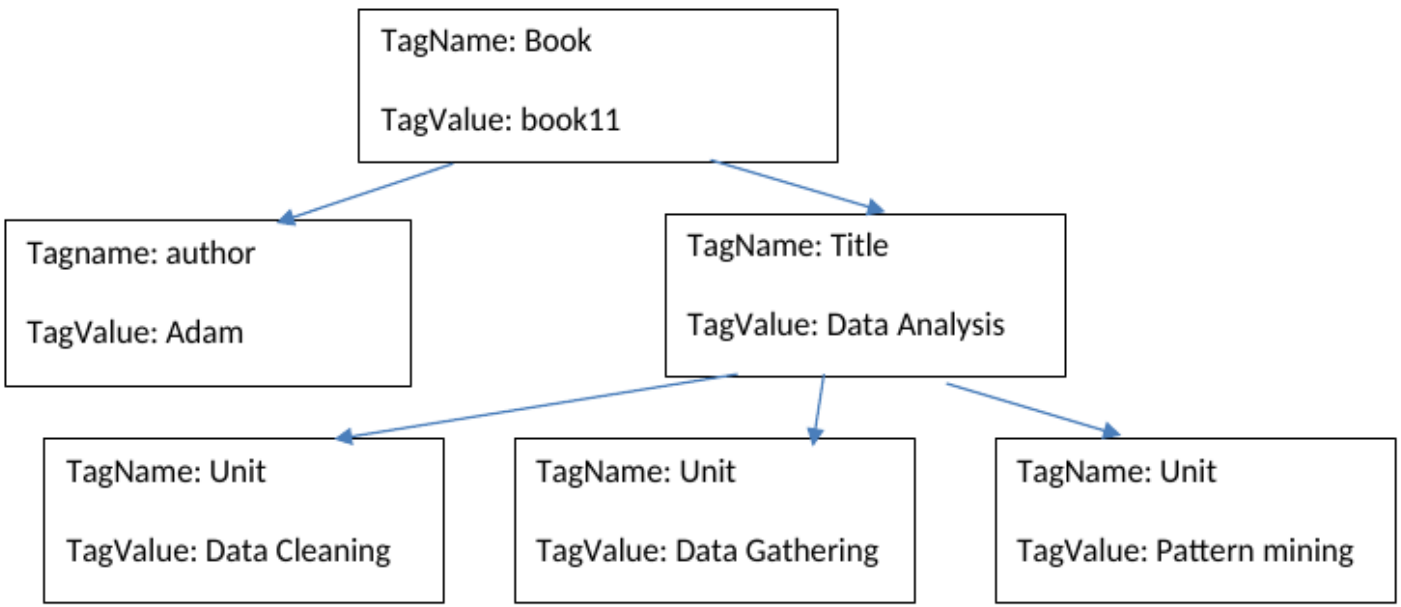
```
int Tree::count_leaves_in_subtree(Node * subtree_root){
  if (subtree_root->left == nullptr &&
    subtree_root->right == nullptr)
    return 1;

  int leaves_in_subtree = 0;
  if (subtree_root->left !=nullptr)
    leaves_in_subtree +=
      count_leaves_in_subtree(subtree_root->left);

  if (subtree_root->right !=nullptr)
    leaves_in_subtree +=
      count_leaves_in_subtree(subtree_root->right);

  return leaves_in_subtree;
}
```

Outline

Q6. Write down a program to translate a tree into XML text file. each tree node has two strings one is the tag name and the other is the tag value, each node has also a number of strings pointing to its children. Traversing the XML file tree will be translated into text a file as follows: <tag> value </tag>. When a node has children the XML appears as follows:

<parenttag> parentvalue <childtag> child 1 </childtag><childtag> child 2 </childtag> </parenttag>

The tree could be as following figure:



```
// Assume that the node class has the
// following attributes
class Node{
  public:
    string TagName;
    string TagValue;
    Node * children[100];
};

void print_XML(Node * node){
  cout<< "<" << node->TagName << ">";
  cout<< node->TagValue;

  for (int i=0; i< 100; i++){
    if (children[i] != nullptr)
      print_XML(children[i]);
  }

  cout<< "</" << node->TagName << ">";
}
```

Outline

Outline

- It's a binary tree.
- Each node has a key.
- The left child node should have a key that is smaller than its parent's key.
- The right child node should have a key that is larger than its parent's key.

Outline

Create a BST and add to it the following numbers 12, 5, 7, 9, 13, 100, 80, 16, 18, 55, 10, 11, Show the tree after deleting nodes 10, 13, 100. Do you think a different sequence of numbers would be used to have a more compressed tree (same number of elements with minimum depth)

Outline

Write down a function that returns the maximum and minimum values within a BST.

```
int tree_max(Node * root);
int tree_min(Node * root);
```

```
int tree_max(Node * root){
  if (root->right == nullptr)
    return root->value;

  return tree_max(root->right);
}
```

```
int tree_min(Node * root){
  if (root->left == nullptr)
    return root->value;

  return tree_min(root->left);
}
```

Outline

Using a BST and an unsorted array of integers, develop an algorithm to sort the array.

- Insert all the values into a BST using a for loop.
- Use in-order traversal to generate a sorted version of the array.
- NOTE: in-order traversal: Left subtree - Current node - Right Subtree

Outline

- Insert: Best O(log(n)) Worst O(n)
- Delete: Best O(log(n)) Worst O(n)
- Insert n elements to a BST: Best O(nlog(n)) Worst O(n*n)

Outline

- BST visualization: `https://visualgo.net/en/bst`
- Feedback form: `https://forms.gle/FUX95z7YVieDF7Wx6`
- CSE 2016 - Eng. Eslam Mounier's solutions: `https://drive.google.com/drive/folders/0BxD159ORSl13THZrVm1oQTFzUTA?usp=sharing`
- Q&A Google Doc.: `https://docs.google.com/document/d/174sf5H_9oVWHHAAM4eutQ7yrRA-uOTuj3aWwyokbUTo`
- Tree problems on Hackerrank: `https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=trees`