LAB 1: CÂY NHỊ PHÂN (Binary Tree)

TS. Võ Phương Bình – Email: binhvp@dlu.edu.vn

Dalat University

Website: https://sites.google.com/view/vophuongbinh

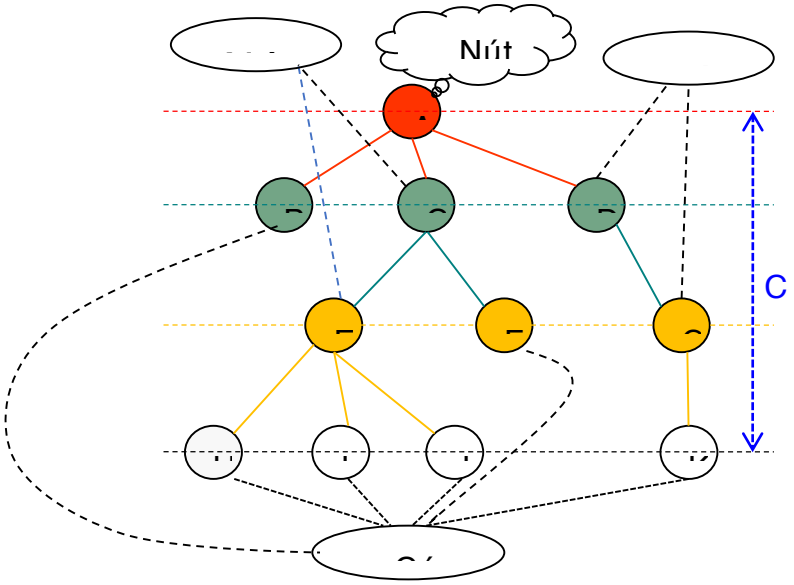
I. LÝ THUYẾT

1. Định nghĩa – thuật ngữ cơ bản

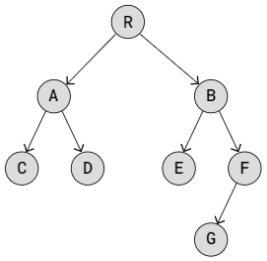
Cây là một tập hợp gồm:

- Gồm các phần tử (node), trong đó có một node đặc biệt gọi là node gốc.
- Các nút trong cây có quan hệ phân cấp (cha – con). Nút ở cấp thứ i sẽ chứa các liên kết để có thể đi tiếp đến các nút ở cấp i+1.
- Nút gốc: là nút đầu tiên của cây, không có cha.
- Nút cha i chứa các liên kết đến nút con cấp i+1.
- Nút lá là nút không có con.
- Nút anh em là các nút con của cùng một cha.
- Nút nhánh (nút trong): là nút khác gốc và khác lá

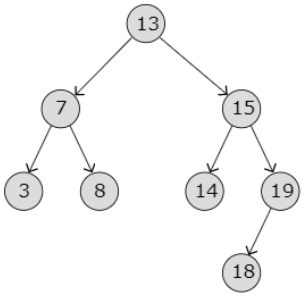
Ví dụ:



- Cây nhị phân: là cây mà mỗi phần tử chỉ có tối đa hai hai phần tử con.



- Cây nhị phân tìm kiếm: là cây nhị phân mà tại mỗi nút đang xét, dữ liệu của nó lớn hơn dữ liệu của các nút ở nhánh con bên trái và bé hơn dữ liệu của các nút ở nhánh con bên phải.



2. *Định nghĩa cấu trúc cây nhị phân*

- Cấu trúc một nút trong cây nhị phân: là một cấu trúc gồm trường data chứa dữ liệu và hai trường dữ liệu lưu trữ địa chỉ nút con trái và nút con phải.

Ví dụ C/C++:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct TreeNode {
    char data;
    struct TreeNode* left;
    struct TreeNode* right;
} TreeNode;

TreeNode* createNewNode(char data) {
    TreeNode* newNode = (TreeNode*)malloc(sizeof(TreeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

int main() {
    TreeNode* root = createNewNode('R');
    TreeNode* nodeA = createNewNode('A');
    TreeNode* nodeB = createNewNode('B');
    TreeNode* nodeC = createNewNode('C');
    TreeNode* nodeD = createNewNode('D');
    TreeNode* nodeE = createNewNode('E');
    TreeNode* nodeF = createNewNode('F');
    TreeNode* nodeG = createNewNode('G');
```

```c
    root->left = nodeA;
    root->right = nodeB;

    nodeA->left = nodeC;
    nodeA->right = nodeD;

    nodeB->left = nodeE;
    nodeB->right = nodeF;

    nodeF->left = nodeG;

    // Test
    printf("root->right->left->data: %c\n", root->right->left->data);

    free(nodeG);
    free(nodeF);
    free(nodeE);
    free(nodeB);
    free(nodeC);
    free(nodeD);
    free(nodeA);
    free(root);

    return 0;
}

//C
```

Ví dụ Python:

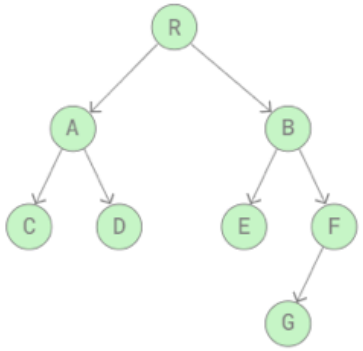
```python
class TreeNode:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

root = TreeNode('R')
nodeA = TreeNode('A')
nodeB = TreeNode('B')
nodeC = TreeNode('C')
nodeD = TreeNode('D')
nodeE = TreeNode('E')
nodeF = TreeNode('F')
nodeG = TreeNode('G')

root.left = nodeA
root.right = nodeB

nodeA.left = nodeC
nodeA.right = nodeD

nodeB.left = nodeE
nodeB.right = nodeF

nodeF.left = nodeG

# Test
print("root.right.left.data:", root.right.left.data)

#Python
```

3. Các thao tác cơ bản trên cây nhị phân

- **Duyệt cây:**

Pre-order Traversal (NLR) of Binary Trees

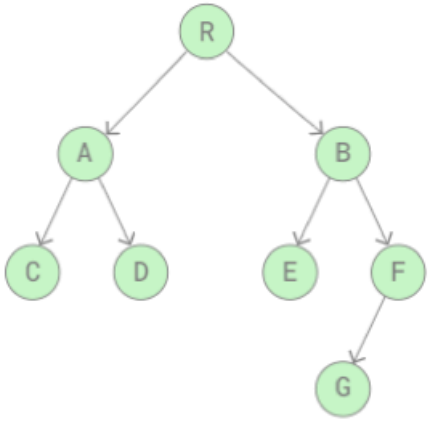
```
• def preOrderTraversal(node):
•     if node is None:
•         return
•     print(node.data, end=", ")
•     preOrderTraversal(node.left)
•     preOrderTraversal(node.right)
```




```
Result: R,A,C,D,B,E,F,G
```

In-order Traversal (LNR) of Binary Trees

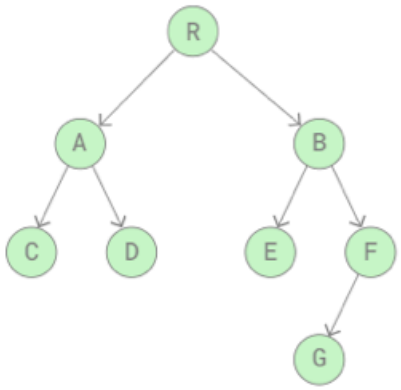
```
• def inOrderTraversal(node):
•     if node is None:
•         return
•     inOrderTraversal(node.left)
•     print(node.data, end=", ")
•     inOrderTraversal(node.right)
```




```
Result: C,A,D,R,E,B,G,F
```

Post-order Traversal (LRN) of Binary Trees

```
• def postOrderTraversal(node):
•     if node is None:
•         return
•     postOrderTraversal(node.left)
•     postOrderTraversal(node.right)
•     print(node.data, end=", ")
```




```
Result: C,D,A,E,G,F,B,R
```

II. YÊU CẦU THỰC HÀNH

Bài 1: Cài đặt cây nhị phân (Binary Tree) với đầy đủ các thao tác duyệt: thứ tự giữa LNR, thứ tự trước NLR, thứ tự sau LRN.

---HẾT---