CẤU TRÚC DỮ LIỆU CÂY

Nguyen Thien Luong

TechMaster
Học là có việc

TechMaster
Học là có việc

NỘI DUNG

1 CẤU TRÚC DỮ LIỆU CÂY

2 CÂY NHỊ PHÂN

3 CÂY NHỊ PHÂN TÌM KIẾM

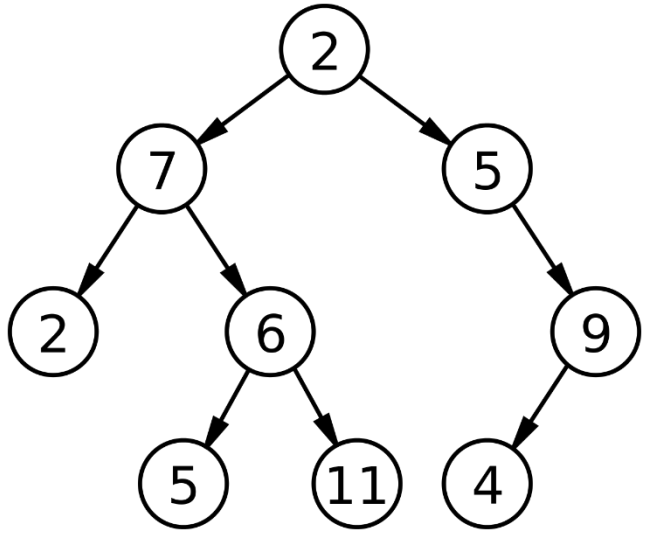
4 CÀI ĐẶT

CẤU TRÚC DỮ LIỆU CÂY LÀ GÌ?

TechMaster
Học là có việc

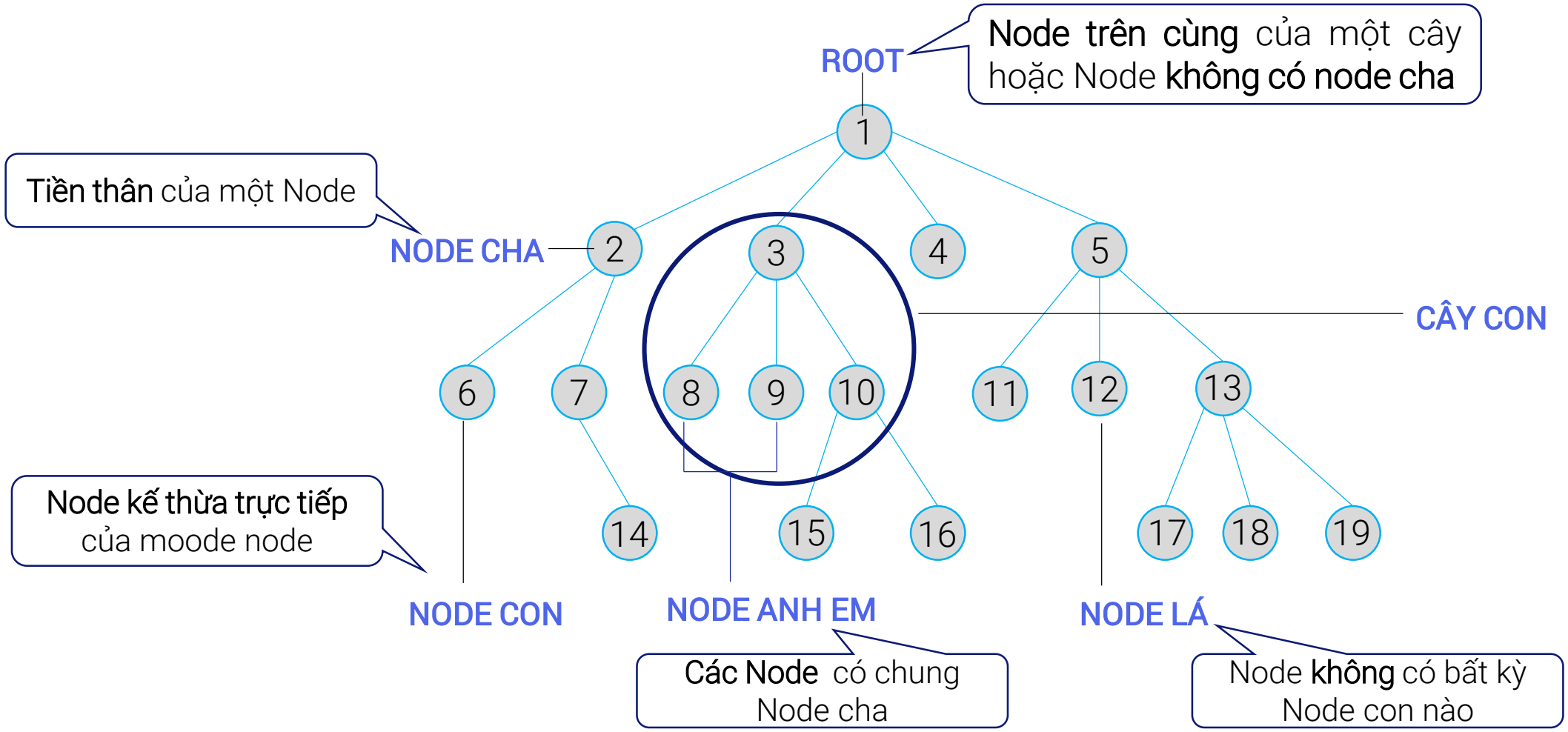
CÂY – TREE

- Cây là một **đồ thị liên thông** và **không có chu trình**
- Cấu trúc dữ liệu cây gồm một tập hợp các **node** được liên kết với nhau theo quan hệ cha-con

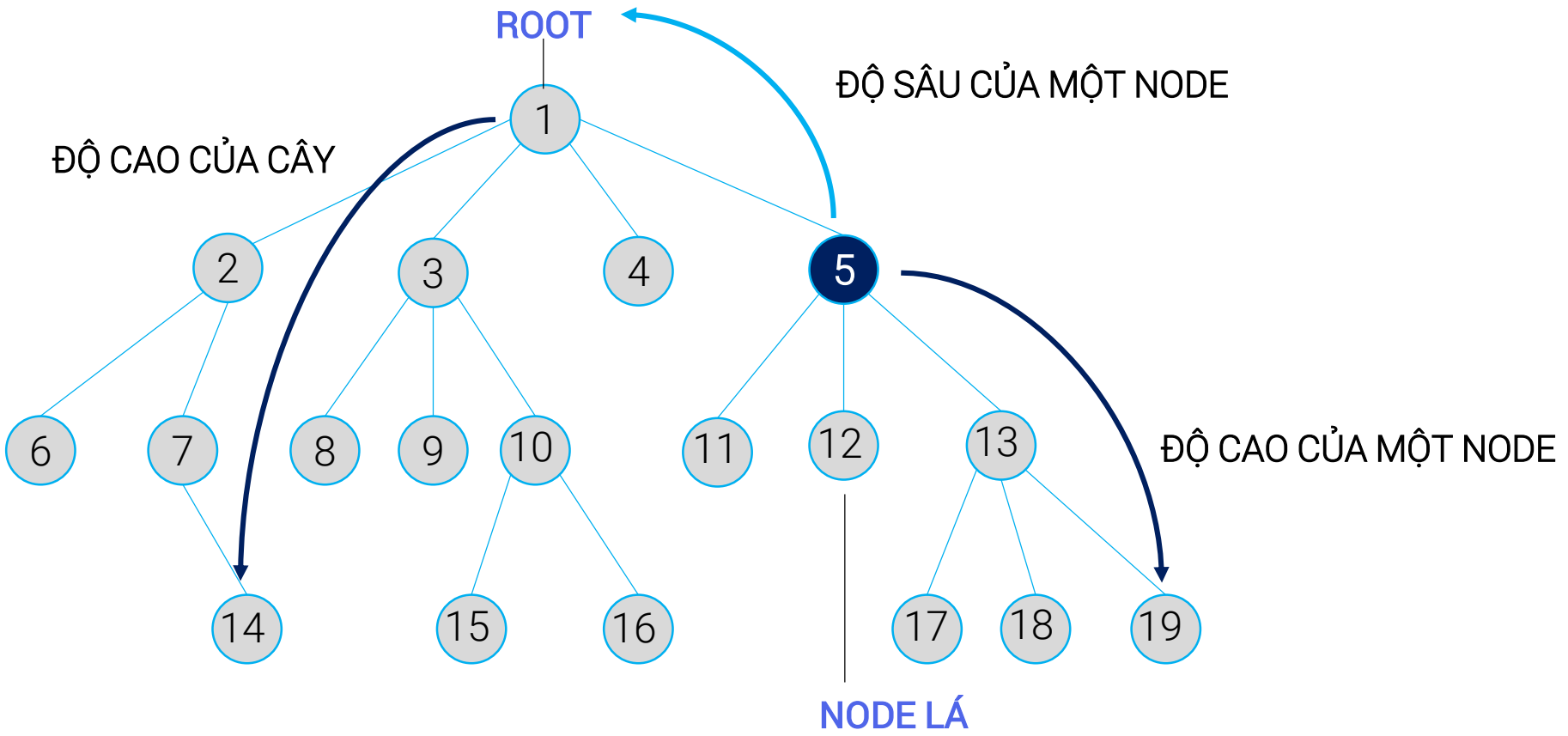


https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

CÂY – TREE


ROOT
Node trên cùng của một cây hoặc Node không có node cha
Tiền thân của một Node
NODE CHA
CÂY CON
Node kế thừa trực tiếp của moode node
NODE CON
NODE ANH EM
Các Node có chung Node cha
NODE LÁ
Node không có bất kỳ Node con nào
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

CÂY – TREE



https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

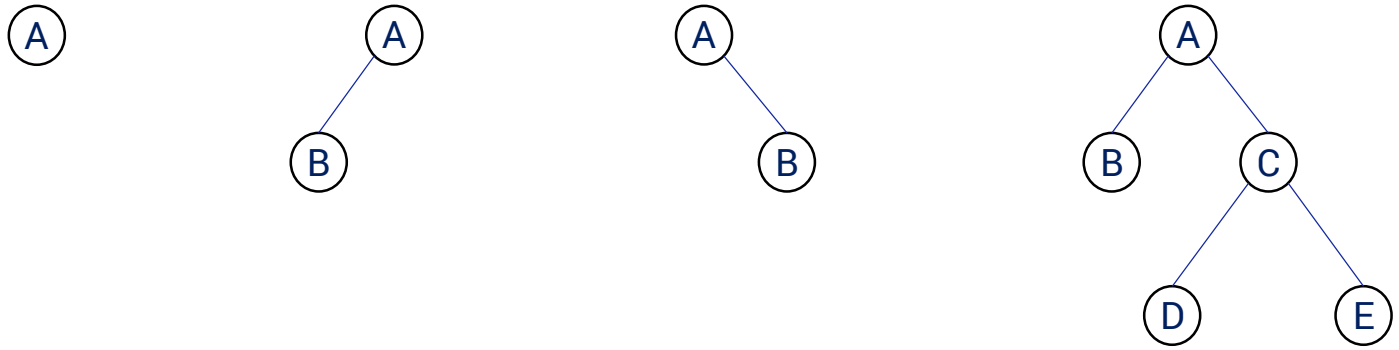
CÂY NHỊ PHÂN – BINARY TREE

Cây nhị phân là gì?

Một dạng cây đặc biệt mà tại tất cả các node chỉ có thể có tối đa hai cây con

Một cây với mỗi node có tối đa hai node con:

- Node con trái
- Node con phải



https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

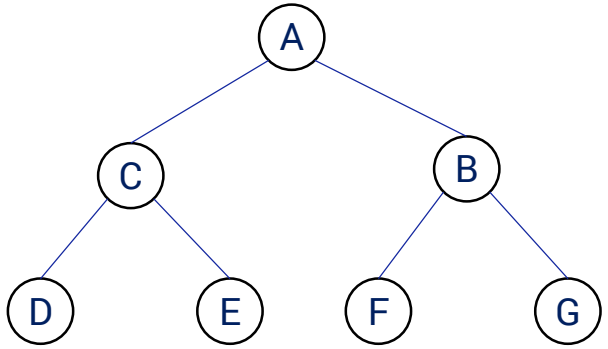
CÂY NHỊ PHÂN – BINARY TREE

Một số loại cây nhị phân

Cây nhị phân đầy đủ

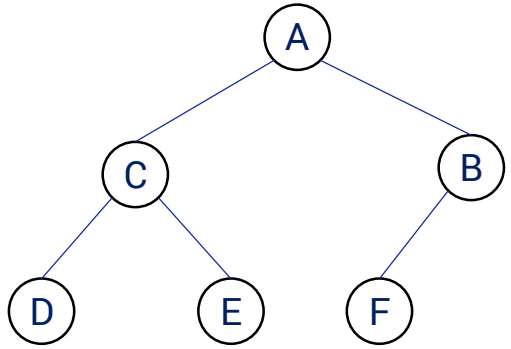
Mọi node có 0 hoặc 2 node con

Tất cả các nodes ngoại trừ các nodes lá (leaf nodes) đều có hai node con



Cây nhị phân hoàn chỉnh

Tất cả các level (cấp) đều được lấp đầy hoàn toàn bởi các node, có thể ngoại trừ level cuối cùng và có tất cả các nodes càng nằm ở bên trái càng tốt.

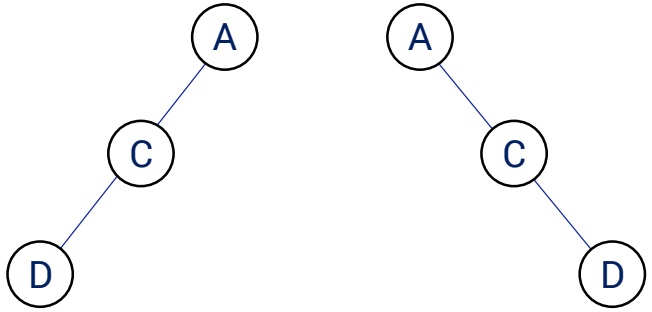


Cây nhị phân lệch

Tất cả các node chỉ có:

Cây con trái (Cây lệch trái)

Cây con phải (Cây lệch phải)



CÂY NHỊ PHÂN – BINARY TREE

DUYỆT CÂY

Thăm tất cả các node của cây duy nhất một lần

Duyệt thứ tự trước - VLR

Node gốc được thăm trước các node con của nó

Duyệt thứ tự giữa - LVR

Node gốc dược duyệt sau cây con trái của nó và trước cây con phải của nó

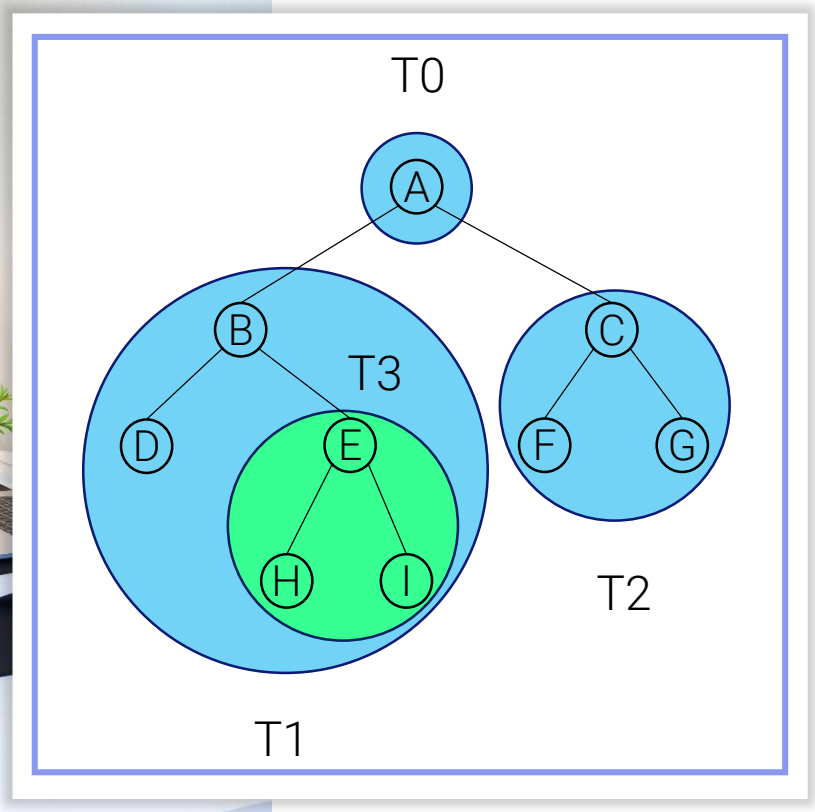
Duyệt thứ tự sau - LRV

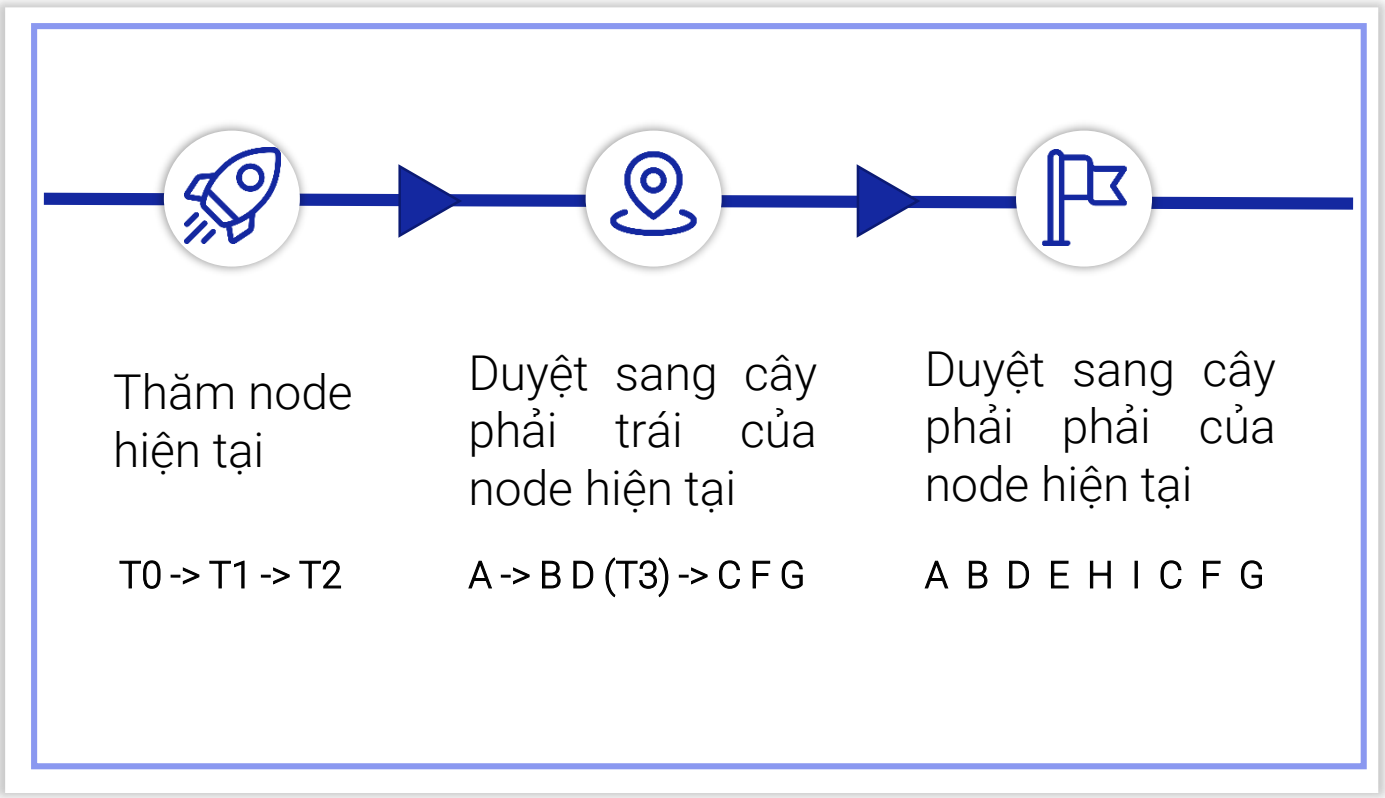
Node gốc được duyệt sau các node con cháu của nó

https://vi.wikipedia.org/wiki/Duyệt_cây

CÂY NHỊ PHÂN – BINARY TREE

Duyệt thứ tự trước

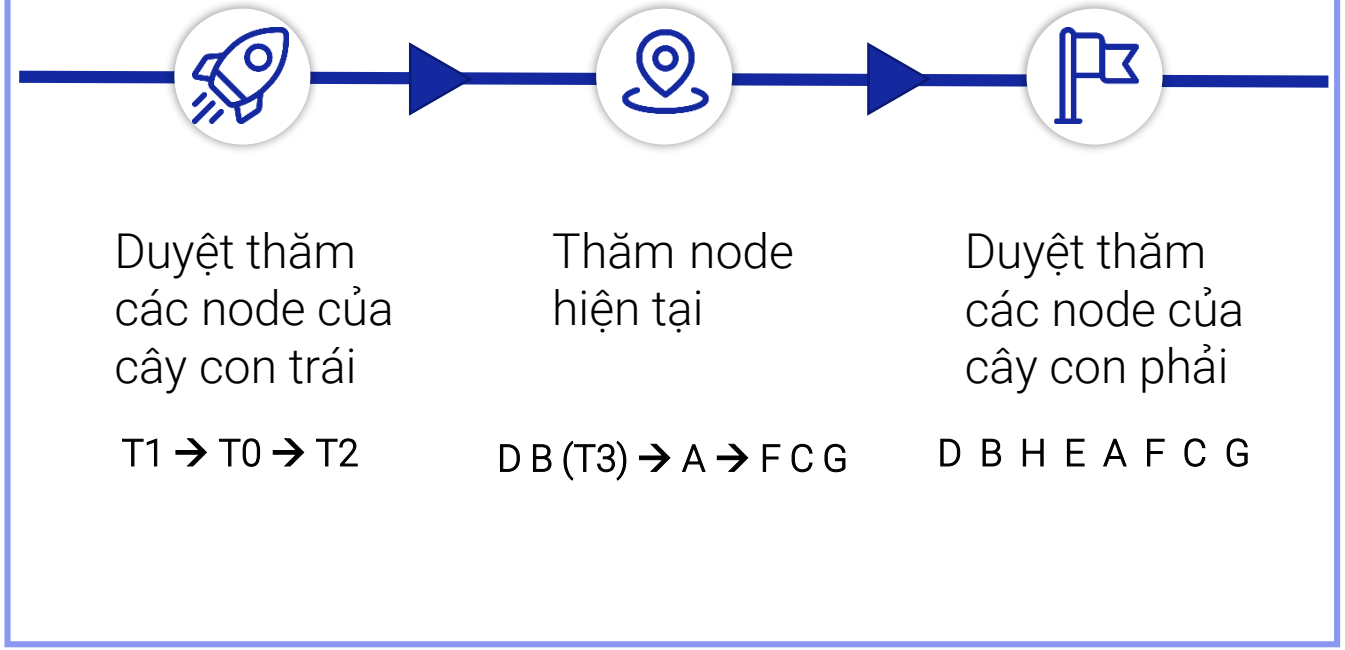


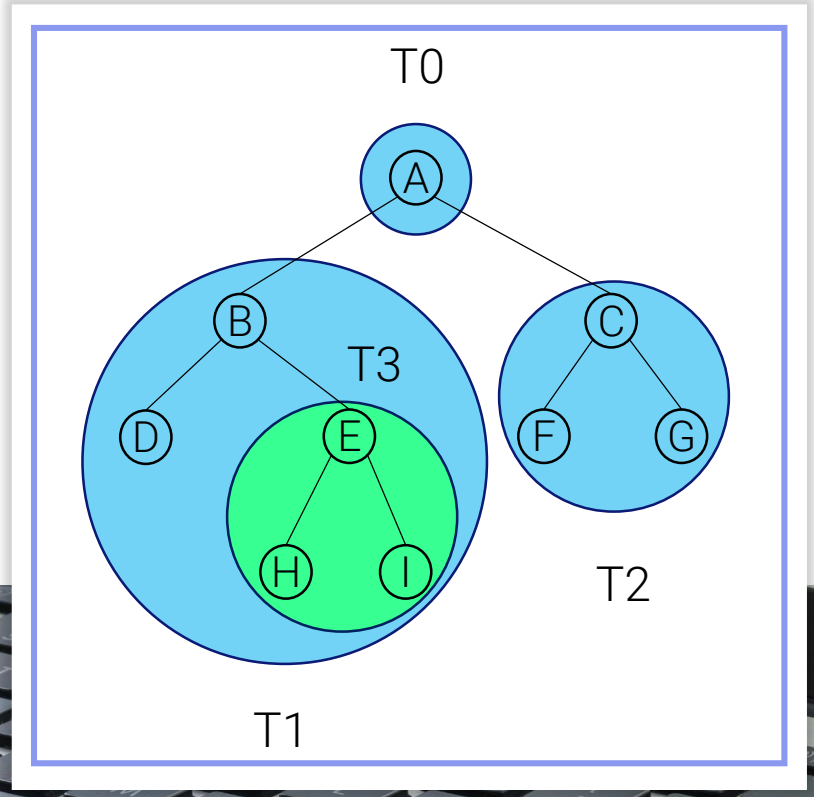


https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

CÂY NHỊ PHÂN – BINARY TREE

Duyệt thứ tự giữa

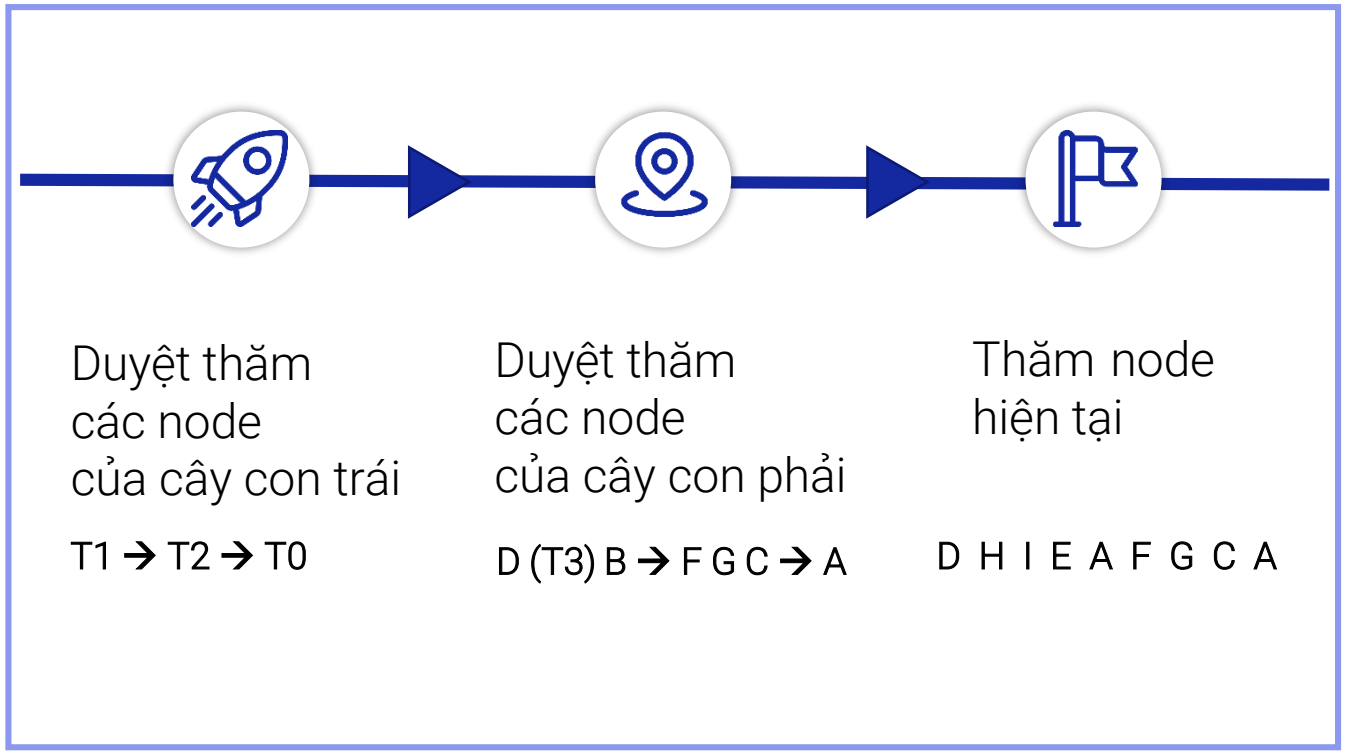


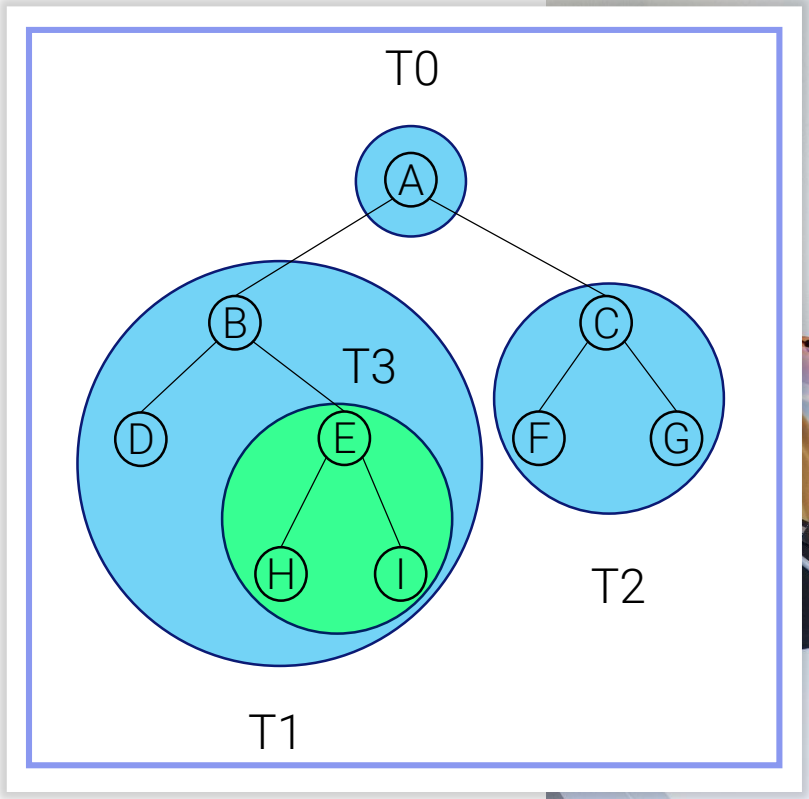


https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

CÂY NHỊ PHÂN – BINARY TREE

Duyệt thứ tự sau





https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

CÂY NHỊ PHÂN – BINARY TREE

DUYỆT CÂY

Mã giả cho các phương pháp duyệt cây

```
preorder_traverse(TREE T)
    if T is not null
        visit(T)
        preoder_traverse(T.left)
        preoder_traverse(T.right)
```

```
inorder_traverse(TREE T)
    if T is not null
        inoder_traverse(T.left)
        visit(T)
        inoder_traverse(T.right)
```

```
posorder_traverse(TREE T)
    if T is not null
        posoder_traverse(T.left)
        posoder_traverse(T.right)
        visit(T)
```

https://vi.wikipedia.org/wiki/Duyệt_cây

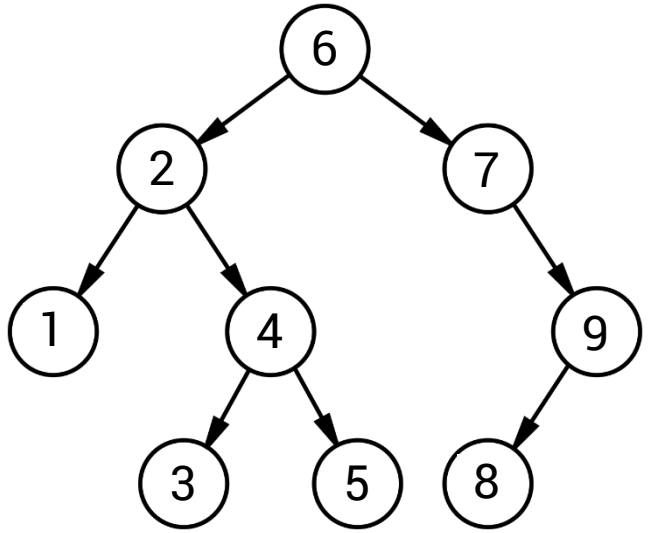

TechMaster
Học là có việc
CÂY NHỊ PHÂN TÌM KIẾM

CÂY NHỊ PHÂN
TÌM KIẾM LÀ GÌ?

CÂY NHỊ PHÂN TÌM KIẾM

TechMaster
Học là có việc

- **Cây con trái** chứa những node có giá trị **nhỏ hơn** node hiện tại
- **Cây con phải** chứa những node có giá trị **lớn hơn** node hiện tại
- **Cây con trái** và **cây con phải** đều là **cây nhị phân tìm kiếm**



https://vi.wikipedia.org/wiki/Cây_(cấu_trúc_dữ_liệu)

CÂY NHỊ PHÂN TÌM KIẾM – BST

CÁC THAO TÁC

TÌM KIẾM

Tìm kiếm một node ở trên cây

THÊM

Thêm một node mới vào cây

XÓA

Xóa một node của cây

https://vi.wikipedia.org/wiki/Duyệt_cây

CÂY NHỊ PHÂN TÌM KIẾM – BST

Thao tác tìm kiếm

TechMaster
Học là có việc

1

Cây nhị phân: T
Giá trị tìm kiếm: x

2

So sánh x với giá trị key tại node gốc

3

Nếu giá trị x < key
Thực hiện tìm kiếm trên cây con trái

4

Nếu giá trị x > key
Thực hiện tìm kiếm trên cây con phải

5

Nếu giá trị x == key
Tìm thấy x trên cây T
Kết thúc tìm kiếm

6

Thực hiện đệ quy thao tác tìm kiếm trên các cây con

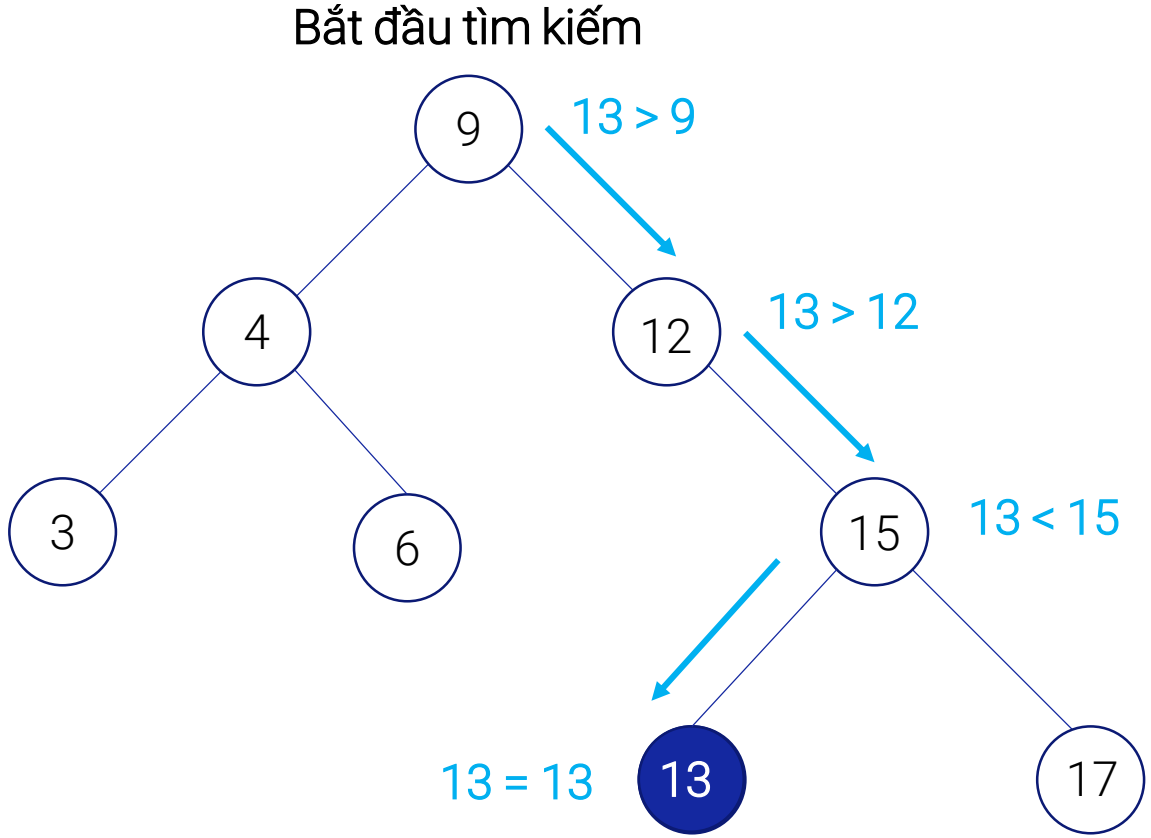
7

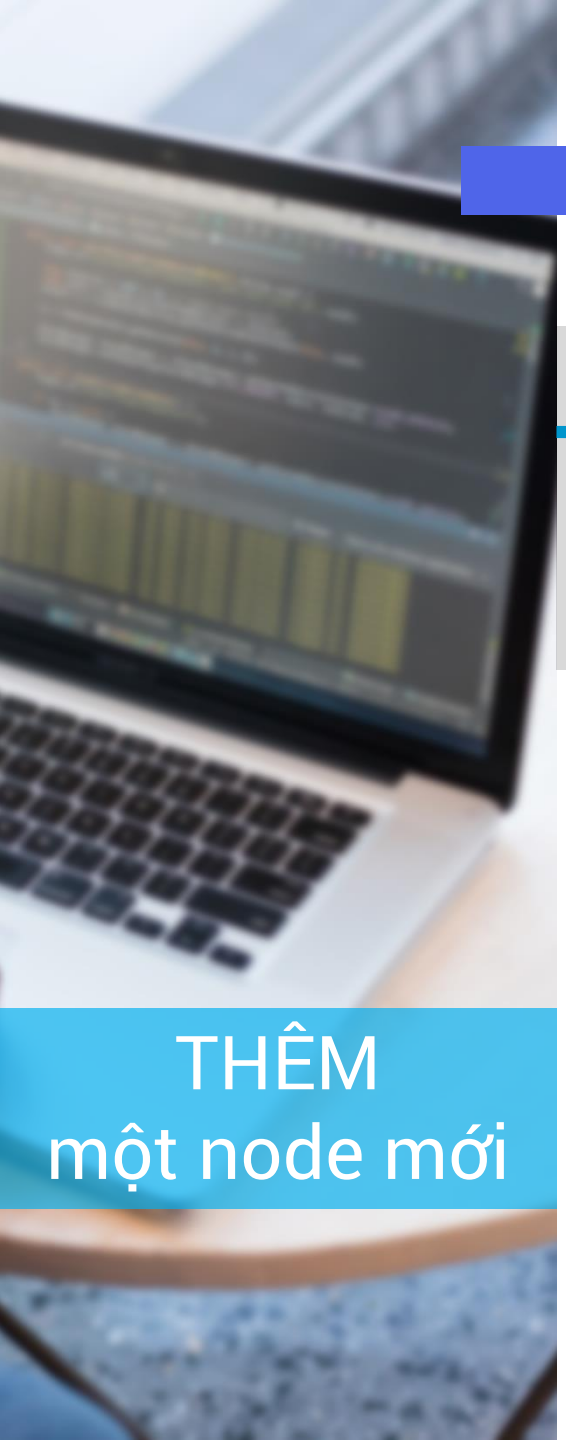
Không tìm thấy giá trị x nếu duyệt qua toàn bộ cây T

CÂY NHỊ PHÂN TÌM KIẾM – BST

Thao tác tìm kiếm

- Tìm kiếm 13





THÊM
một node mới

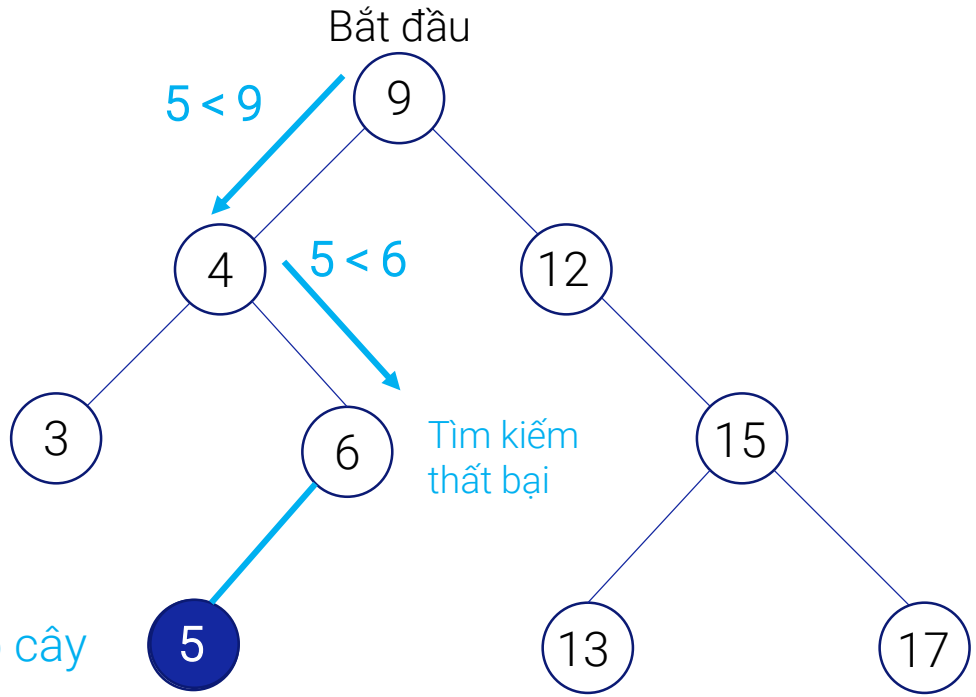
CÂY NHỊ PHÂN TÌM KIẾM – BST

1 Cây nhị phân: **T**
Giá trị tìm kiếm: **x**

2 Thực hiện thao tác **tìm kiếm**

3 Thực hiện **thêm node mới** tại vị trí **tìm kiếm thất bại**

- Thêm giá trị **5** vào cây

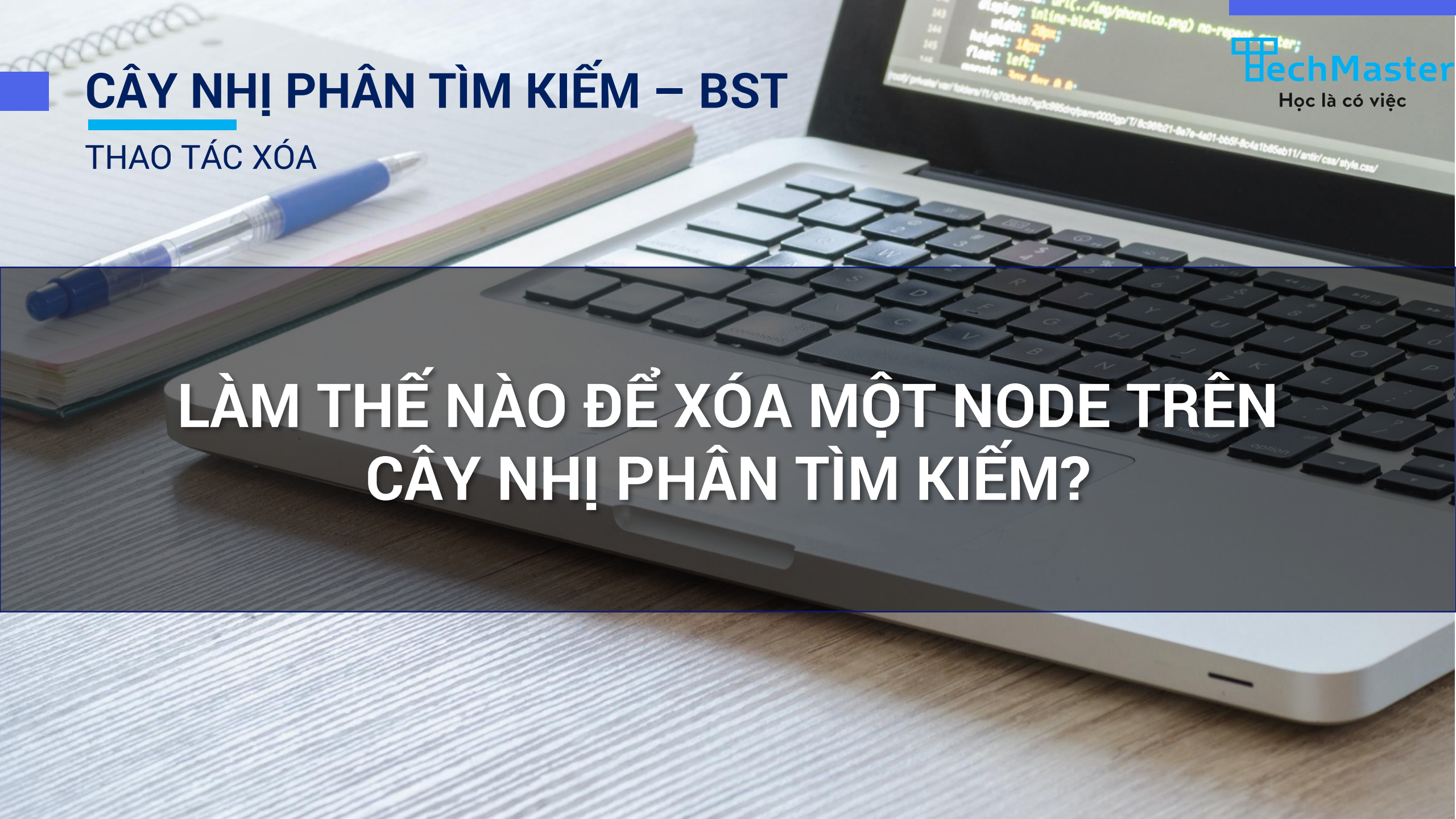


Thêm giá trị **5** vào cây

CÂY NHỊ PHÂN TÌM KIẾM – BST

THAO TÁC XÓA

TechMaster
Học là có việc



LÀM THẾ NÀO ĐỂ XÓA MỘT NODE TRÊN CÂY NHỊ PHÂN TÌM KIẾM?


TechMaster
Học là có việc
CÀI ĐẶT
CÂY NHỊ PHÂN TÌM KIẾM

CÂY NHỊ PHÂN TÌM KIẾM – BST

CÀI ĐẶT

ĐỊNH NGHĨA

```
static class NodeTree {
    int value;
    NodeTree left, right;

    public NodeTree(int value) {
        this.value = value;
    }
}
```

TÌM KIẾM

```
private static boolean searchVal(int val) {
    temp = root;
    while (temp != null) {
        if (temp.value == val) {
            return true;
        }
        if (temp.value > val) {
            temp = temp.left;
        } else {
            temp = temp.right;
        }
    }

    return false;
}
```

CÂY NHỊ PHÂN TÌM KIẾM – BST

TechMaster
Học là có việc

THÊM NODE

```
private static void addNode(int val) {
    if (root == null) {
        root = new NodeTree(val);
        size++;
    } else {
        temp = root;
        while (temp.value != val) {
            if (temp.value > val) {
                if (temp.left == null) {
                    temp.left = new NodeTree(val);
                    size++;
                    break;
                }
                temp = temp.left;
            } else {

                if (temp.right == null) {
                    temp.right = new NodeTree(val);
                    size++;
                    break;
                }
                temp = temp.right;
            }
        }
    }
}
```

XÓA NODE

```
private static NodeTree delete(NodeTree n, int val) {
    if (n == null) {
        return null;
    }

    if (n.value > val) {
        n.left = delete(n.left, val);
    } else if (n.value < val) {
        n.right = delete(n.right, val);
    } else {
        isDeleted = true;
        if (n.left == null) {
            return n.right;
        }
        if (n.right == null) {
            return n.left;
        }
        NodeTree maxL = n.left;
        while (maxL.right != null) {
            maxL = maxL.right;
        }
        n.value = maxL.value;
        n.left = delete(n.left, n.value);
    }

    return n;
}
```

HỎI ĐÁP

THANK YOU!