

NỘI DUNG

CÂY VÀ CÂY NHỊ PHÂN



Định Nghĩa Cây

- Cây là một tập hợp T các phần tử (gọi là nút của cây), trong đó có một nút đặc biệt gọi là nút gốc, các nút còn lại được chia thành những tập rời nhau T_1, T_2, \dots, T_n theo quan hệ phân cấp, trong đó T_i cũng là 1 cây. Mỗi nút ở cấp i sẽ quản lý một số nút ở cấp $i+1$. Quan hệ này người ta gọi là quan hệ cha – con.

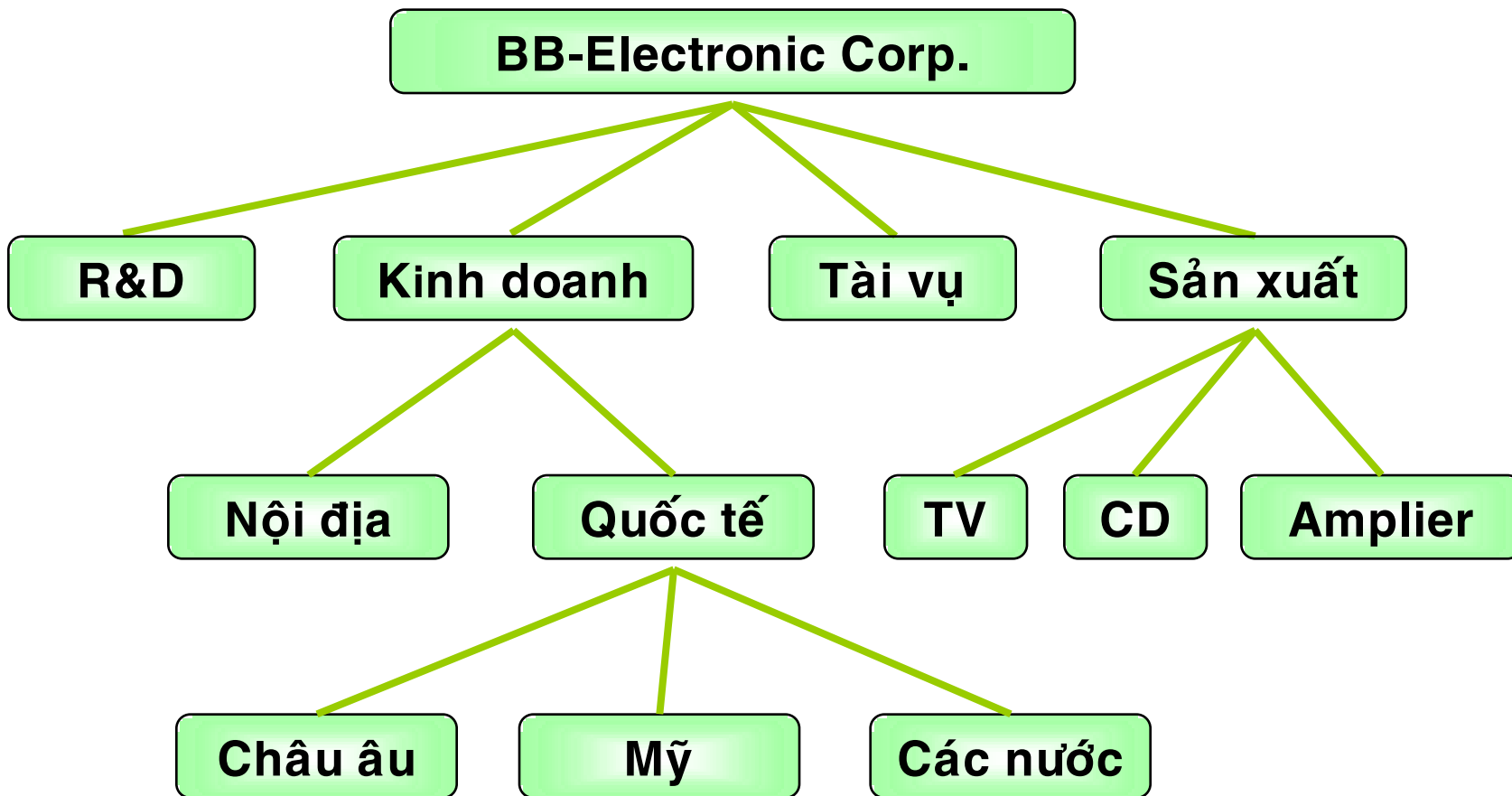


Một Số Khái Niệm

- Bậc của một nút: là số cây con của nút đó .
- Bậc của một cây: là bậc lớn nhất của các nút trong cây
- Nút gốc: là nút không có nút cha.
- Nút lá: là nút có bậc bằng 0 .
- Mức của một nút:
 - Mức (gốc (T)) = 0.
 - Gọi $T_1, T_2, T_3, \dots, T_n$ là các cây con của T_0 :
 $\text{Mức}(T_1) = \text{Mức}(T_2) = \dots = \text{Mức}(T_n) = \text{Mức}(T_0) + 1.$
- Độ dài đường đi từ gốc đến nút x: là số nhánh cần đi qua kể từ gốc đến x.

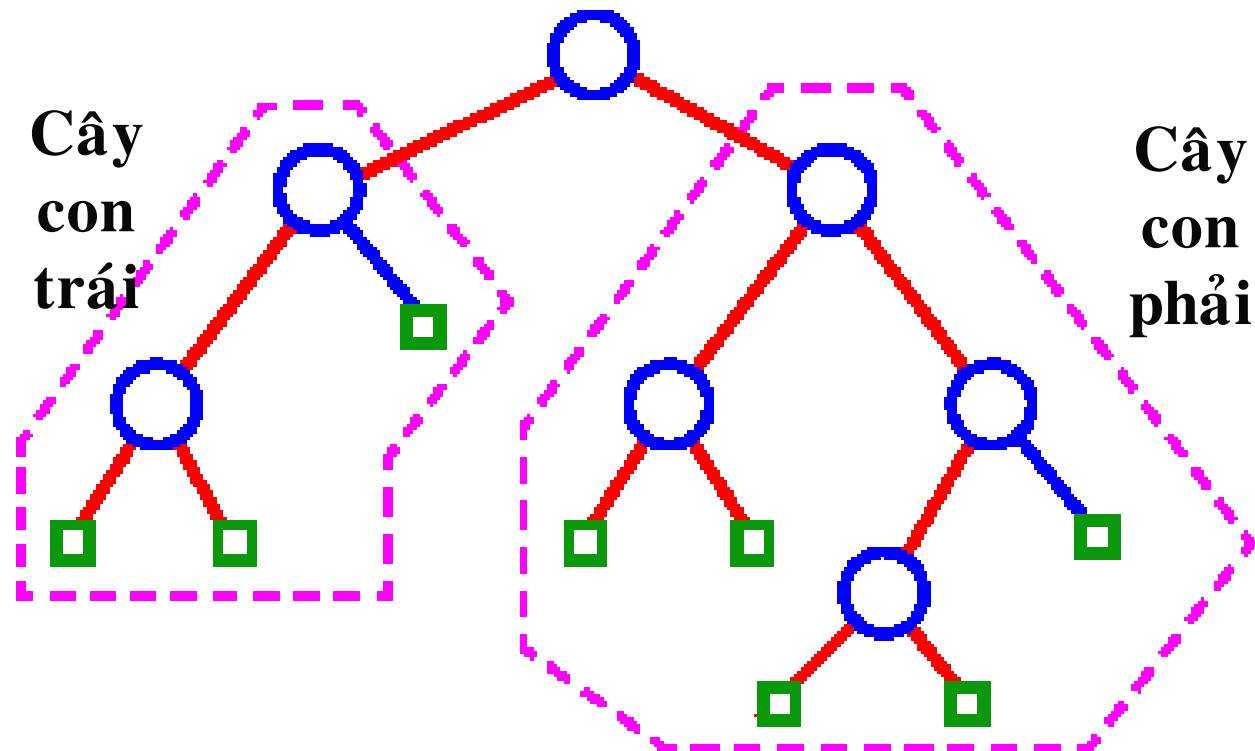


Ví Dụ 1 Tổ Chức Dạng Cây



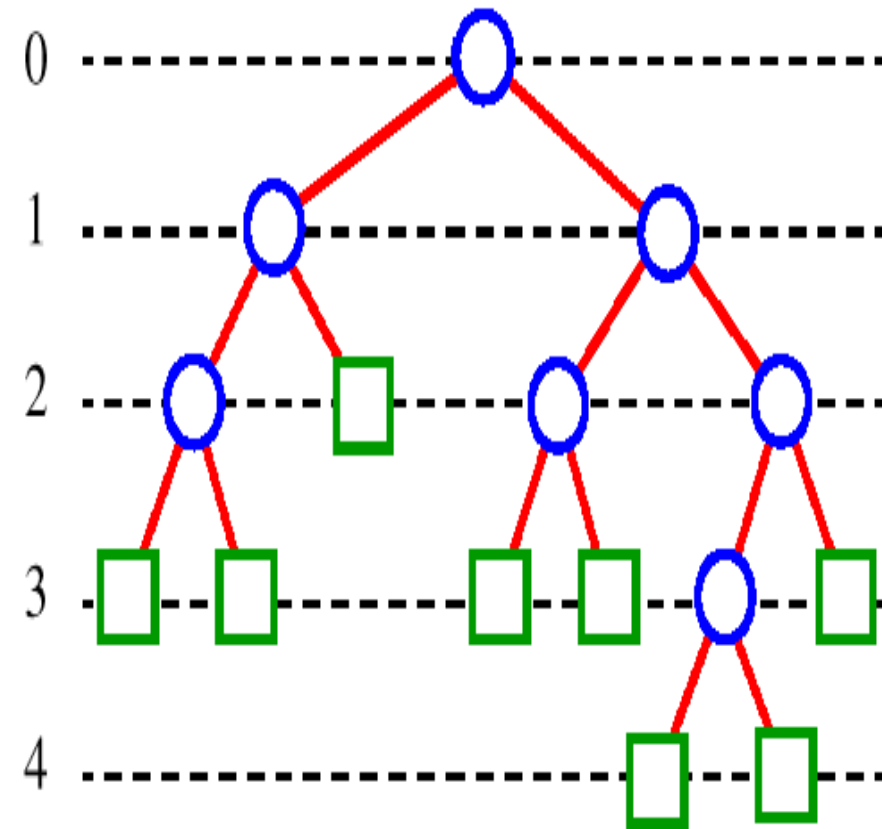
Cây Nhị Phân

- **Mỗi nút có tối đa 2 cây con**



Một Số Tính Chất Của Cây Nhị Phân

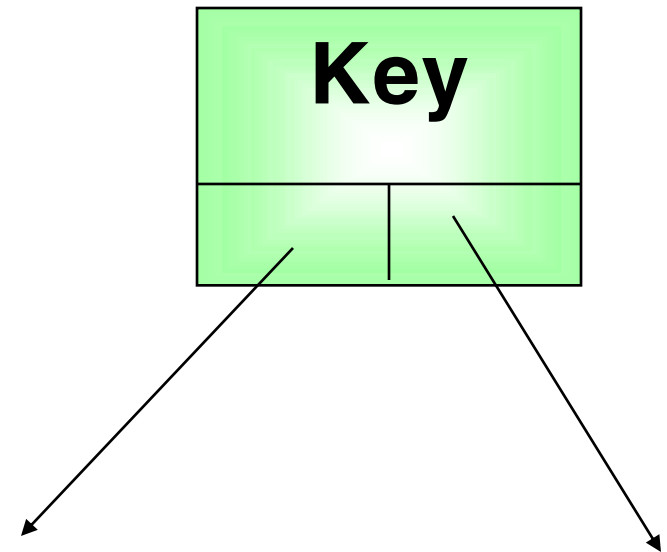
- Số nút nằm ở mức $i \leq 2^i$.
- Số nút lá $\leq 2^h - 1$, với h là chiều cao của cây.
- Chiều cao của cây $h \geq \log_2(N)$
 - N = số nút trong cây
- Số nút trong cây $\leq 2^h - 1$.



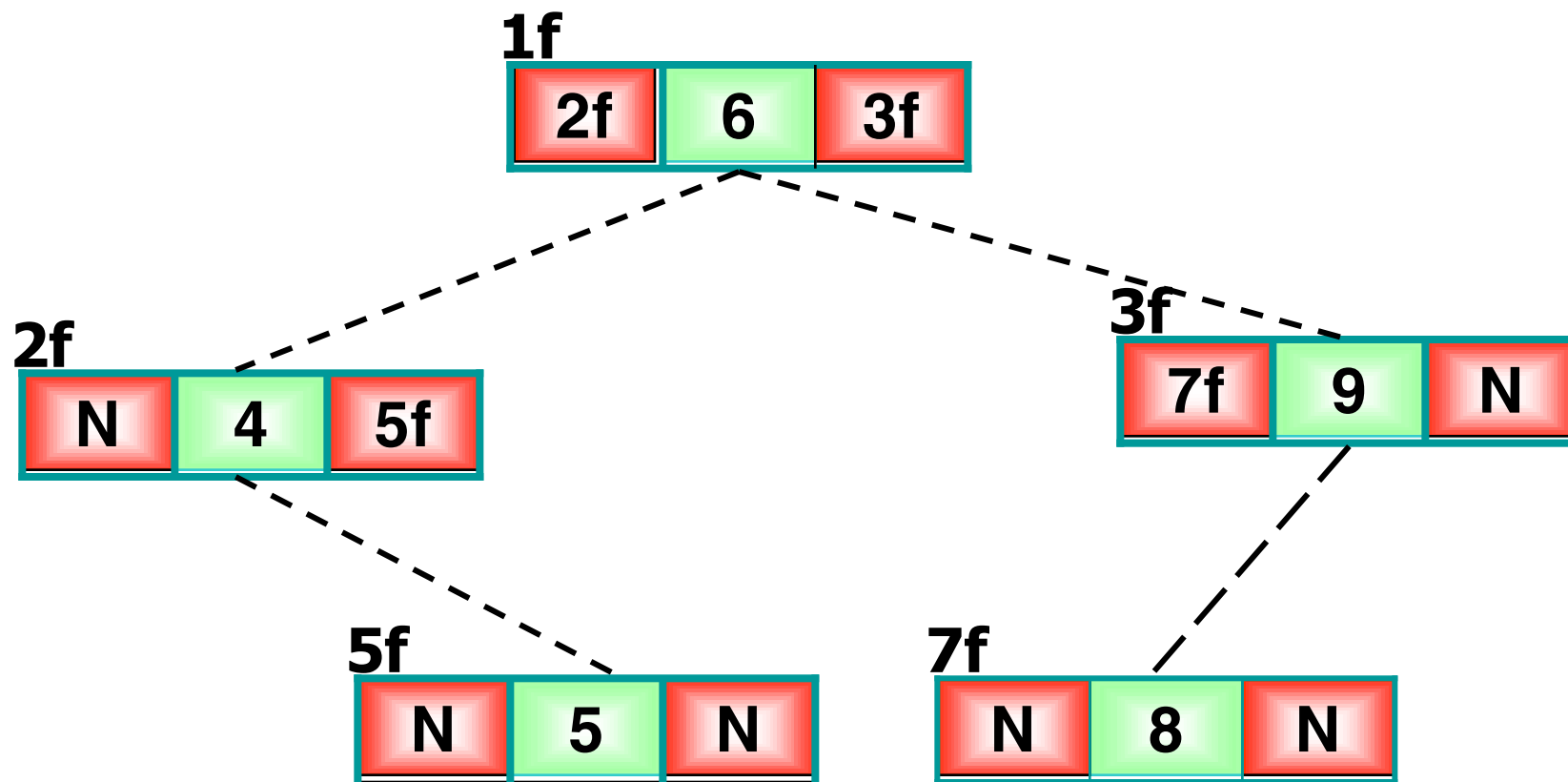
Cấu Trúc Dữ Liệu Của Cây Nhị Phân

```
typedef struct tagTNode
{
    Data    Key;
    struct tagTNode *pLeft;
    struct tagTNode *pRight;
}TNode;

typedef TNode *TREE;
```



Ví Dụ Cây Được Tổ Chức Trong Bộ Nhớ

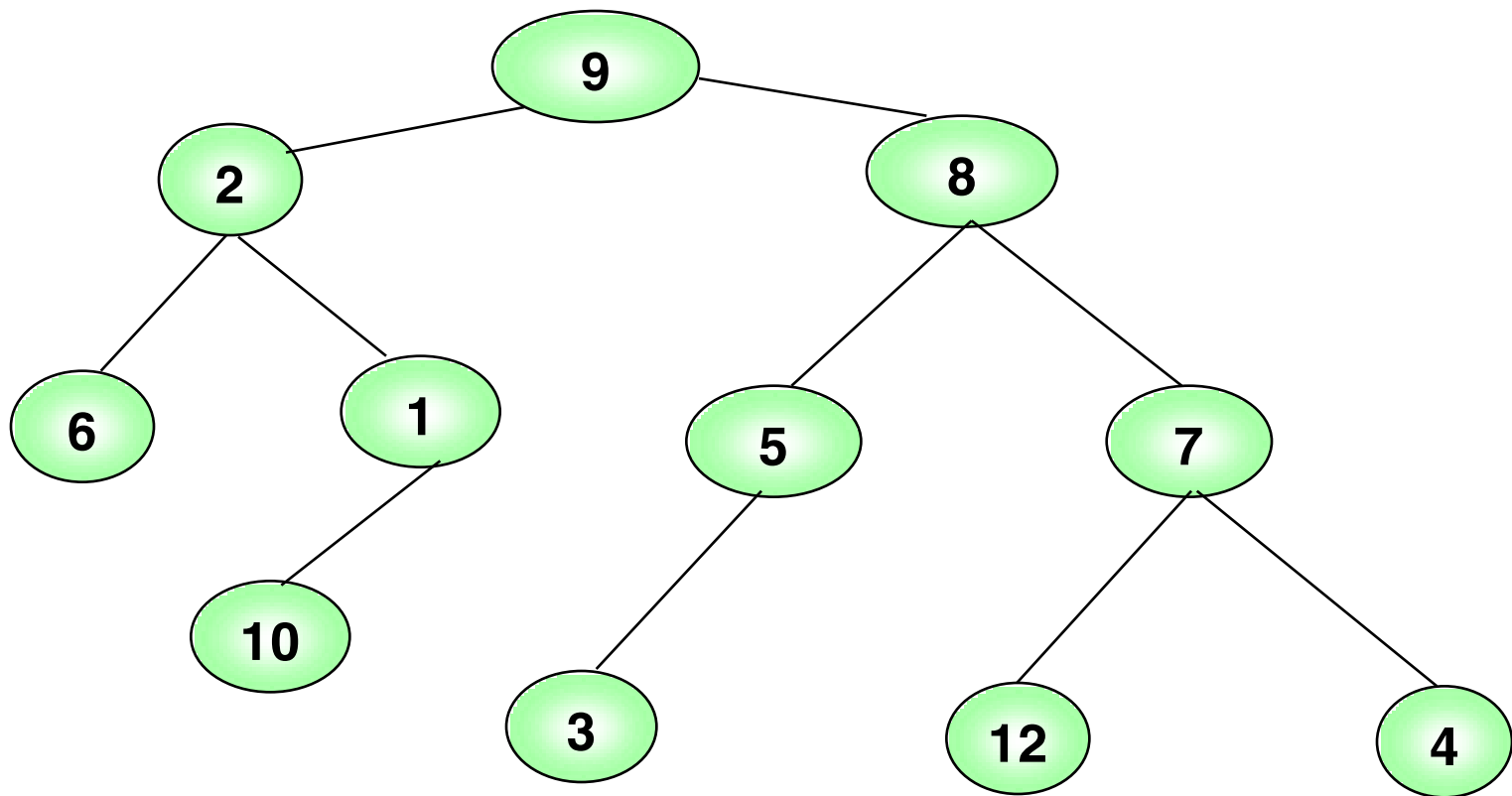


Duyệt Cây Nhị Phân

- Có 3 trình tự thăm gốc :
 - Duyệt trước
 - Duyệt giữa
 - Duyệt sau
- Độ phức tạp $O(\log_2(h))$
Trong đó h là chiều cao cây



Ví Dụ Kết Quả Của Phép Duyệt Cây



- NLR: 9, 2, 6, 1, 10, 8, 5, 3, 7, 12, 4.
- LNR: 6, 2, 10, 1, 9, 3, 5, 8, 12, 7, 4.
- Kết quả của phép duyệt : LRN, NRL,LRN, LNR?



Duyệt Trước

```
void NLR(TREE Root)
{
    if (Root != NULL)
    {
        <Xử lý Root>; //Xử lý tương ứng theo nhu cầu
        NLR(Root->pLeft);
        NLR(Root->pRight);
    }
}
```



Duyệt Giữa

```
void LNR(TREE Root)
{
    if (Root != NULL)
    {
        LNR(Root->pLeft);
        <Xử lý Root>; // Xử lý tương ứng theo nhu
                        cầu
        LNR(Root->pRight);
    }
}
```



Duyệt Sau

```
void    LRN (TREE Root)
{
    if (Root != NULL)
    {
        LRN (Root->pLeft) ;
        LRN (Root->pRight) ;
        <Xử lý Root>; // Xử lý tương ứng theo nhu
                        cầu
    }
}
```



Biểu Diễn Cây Tổng Quát Bằng Cây Nhị Phân

