PRESENTATION

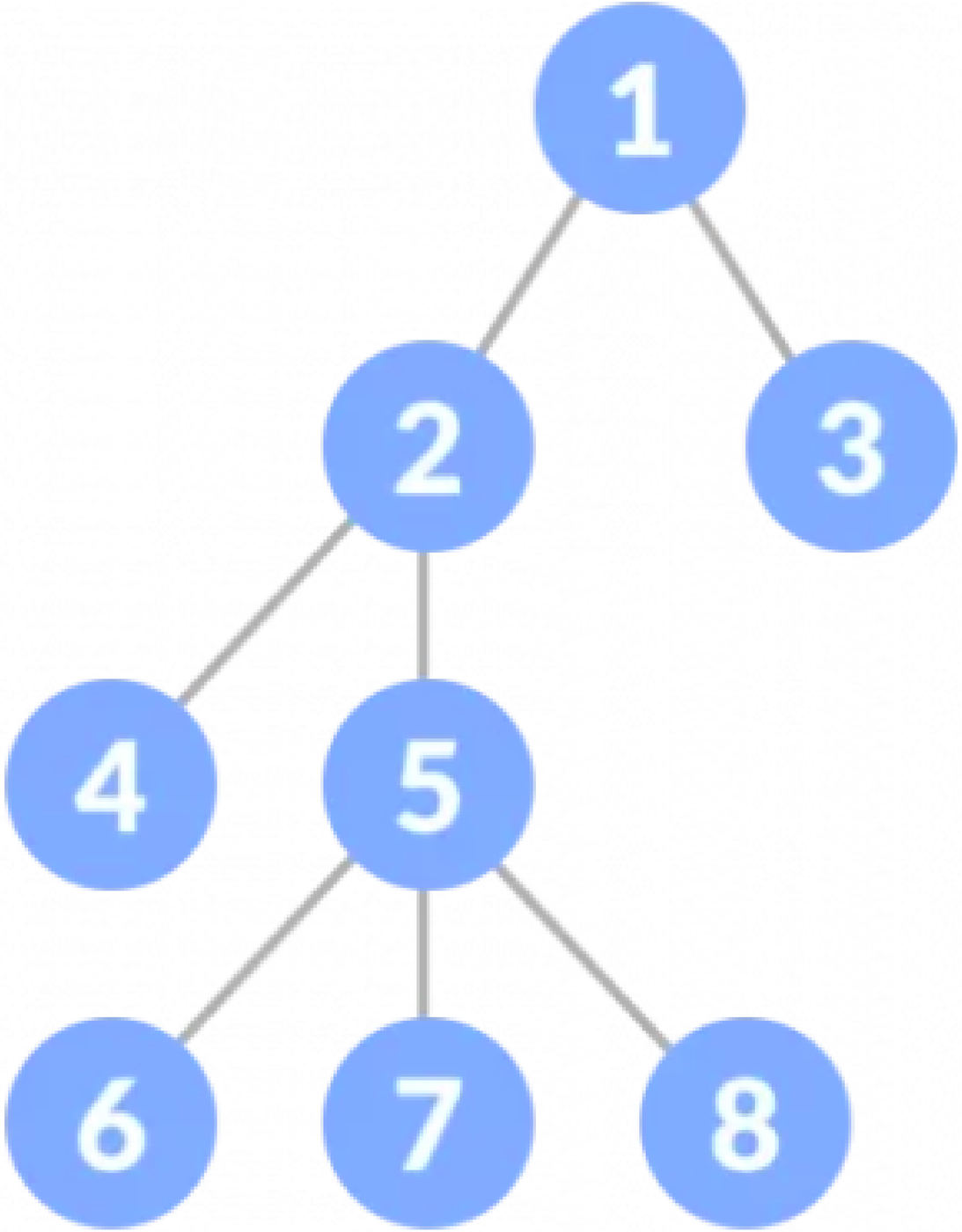
TREE BINARY TREE

THÀNH VIÊN NHÓM

LƯU BÌNH
MSSV: 23520156

NGUYỄN HỮU DUY
MSSV: 23520374

TRƯƠNG NGỌC SANG
MSSV: 23521348



I. CẤU TRÚC CÂY

- Định nghĩa, khái niệm
- Tính chất
- Ví dụ

ĐỊNH NGHĨA VÀ MỘT SỐ KHÁI NIỆM

CẤU TRÚC CÂY

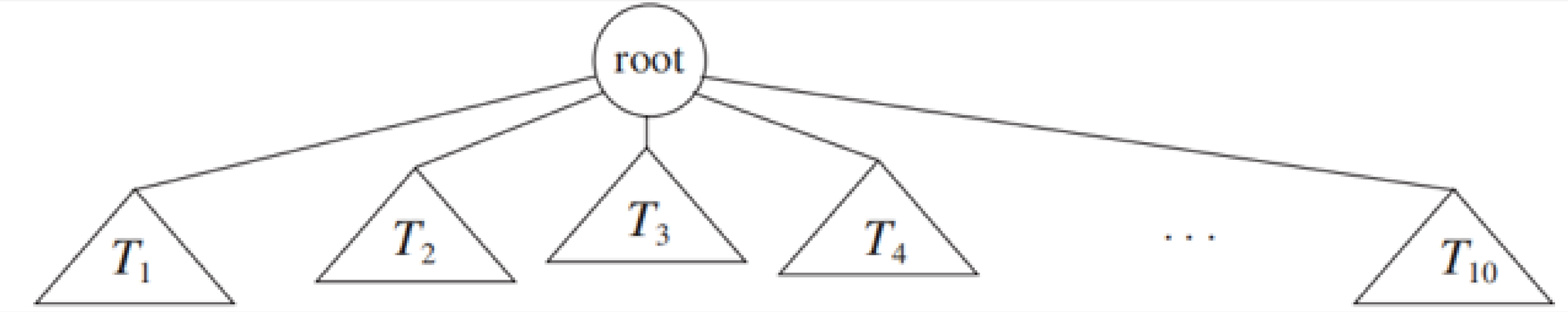
ĐỊNH NGHĨA

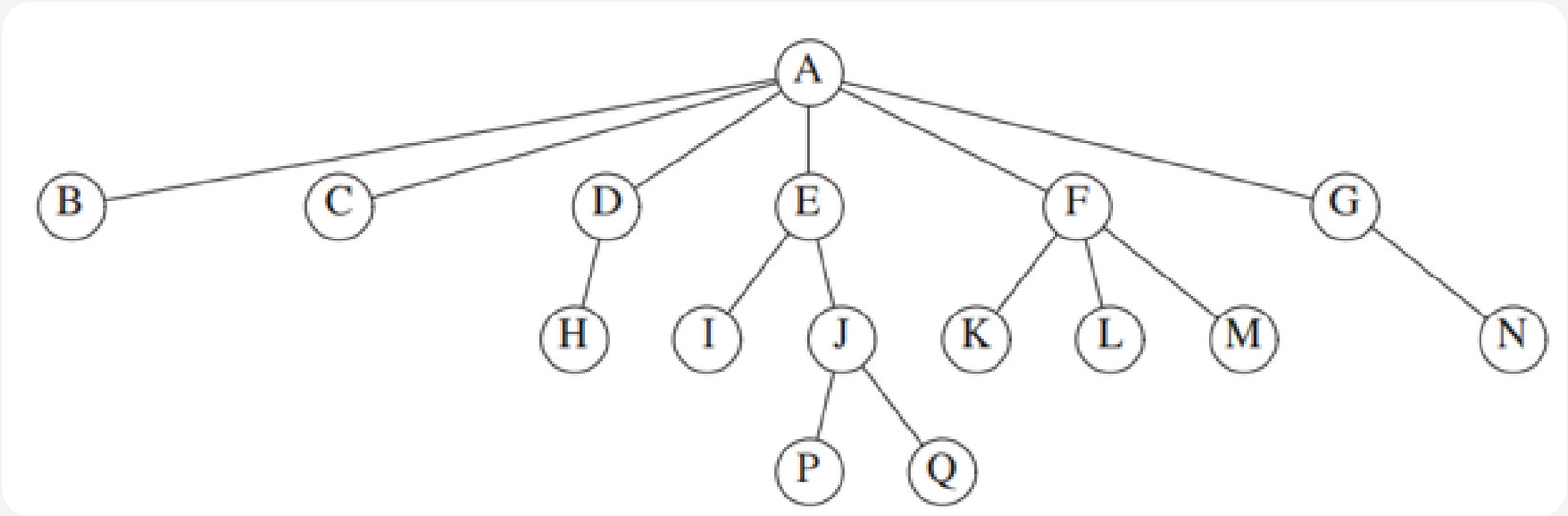
Cây là một tập hợp T các phần tử (gọi là nút hay node):

➢ Tập hợp T có thể rỗng

➢ (Định nghĩa đệ quy) Nếu cây không rỗng, có một nút đặc biệt gọi là nút gốc (root), các nút còn lại được chia thành tập các cây con (subtree) T1, T2, ...,Tn. Các cây con này liên kết trực tiếp với node gốc thông qua 1 cạnh (edge).

HÌNH MINH HỌA





MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

–Kích thước cây (size): Là số nút có trên cây

–Nút nhánh (node-branch node)
- Là nút có ít nhất một con
- Tên gọi khác: nút trong, nút không tận cùng

–Nút lá (leaf node or terminal node)
- Là nút không có con nào
- Tên gọi khác: nút ngoài, nút tận cùng

MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

-Bậc của một nút (degree of node)

- Là số cây con của node đó

-Bậc của một cây (degree of tree)

- Là bậc lớn nhất của các node trong cây

-Mức của một nút:

- Mức (gốc (T)) = 0.
- Gọi T1, T2, T3, ... , Tn là các cây con của T0 :

Mức (T1) = Mức (T2) = . . . = Mức (Tn) = Mức (T0) + 1.

MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

-Cha và con (parent and child)

- Mỗi node trừ node gốc đều có duy nhất 1 node cha
- Một node có thể có số lượng node con tùy ý
- Node A là node cha của node B khi node A ở mức i và node B ở mức i+1. Đồng thời có một cạnh nối giữa node A và B (ta còn gọi B là con của A).

MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

-Họ hàng (siblings)

- Node có cùng cha

-Đường đi (Path)

- Đường đi từ node n1 tới nk được định nghĩa là: Một tập các node n1, n2, ..., nk sao cho ni là cha của ni+i (1 ≤ i < k)

-Chiều dài đường đi (path length)

- Số lượng cạnh trên đường đi

MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

-Độ sâu của node (Depth/Level of node)

• Độ dài đường đi (path length) duy nhất từ node gốc tới node đó

• Độ sâu của cây (The depth of a tree) bằng với chiều sâu của node lá sâu nhất

-Chiều cao của node (Height of node)

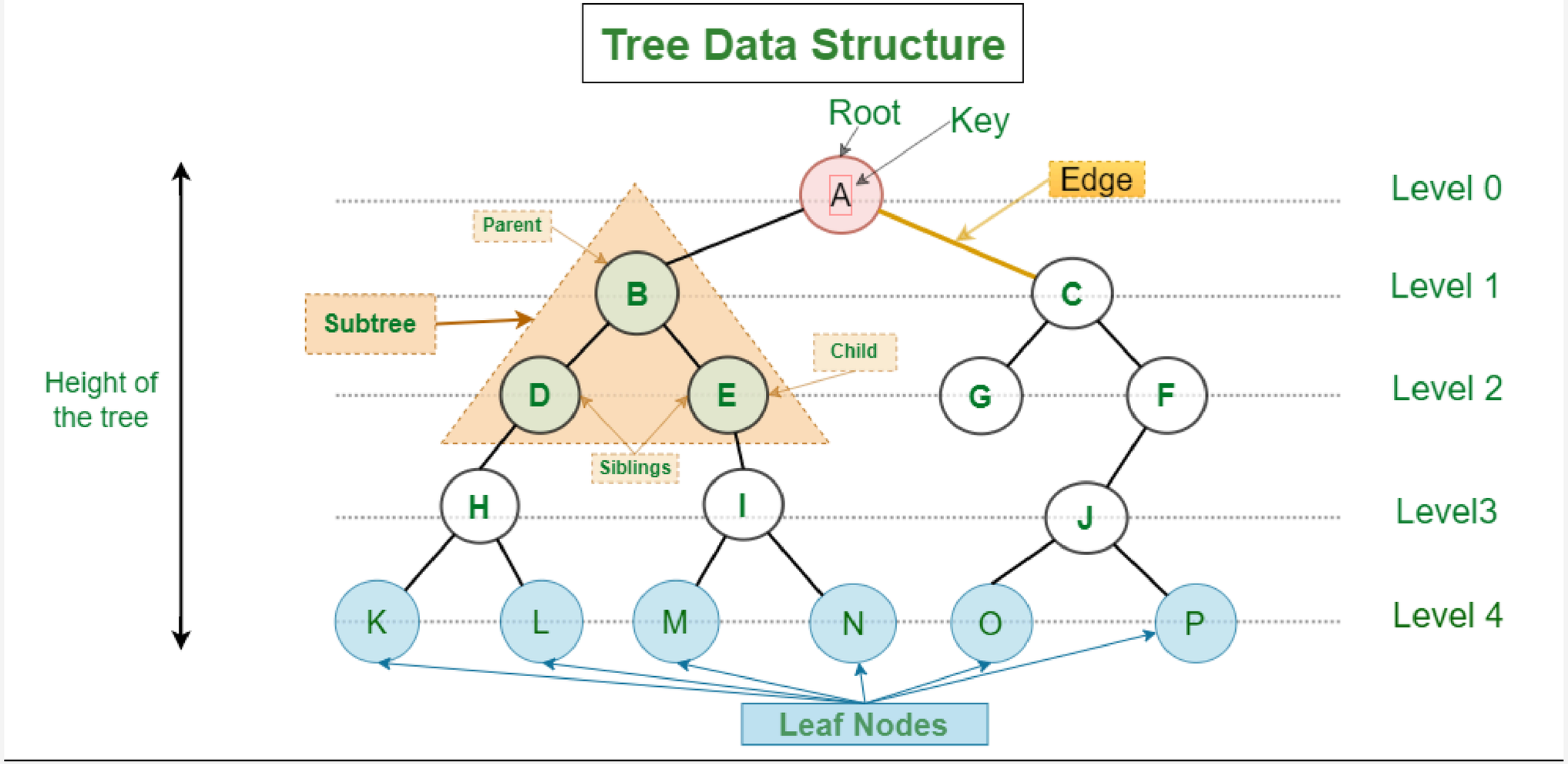
• Chiều cao của một node bất kỳ trong cây là chiều dài dài nhất của đường đi từ node đó tới một node lá

• Các node lá có chiều cao bằng 0

MỘT SỐ KHÁI NIỆM CƠ BẢN CỦA CẤU TRÚC CÂY

- Chiều cao của cây (Height of tree)
• Bằng chiều cao của node gốc

-Tổ tiên và hậu duệ (Ancestor and descendant)
• Nếu có một đường nối từ nút A đến nút B và mức của nút A < mức của nút B thì ta nói A là cha ông (tiền bối) của B và B gọi là con cháu (hậu duệ) của A.



CÁC KHÁI NIỆM CƠ BẢN

CÁC TÍNH CHẤT CƠ BẢN CỦA CÂY

CÁC TÍNH CHẤT CƠ BẢN CỦA CÂY

Số nút của cây bằng số nhánh cộng 1

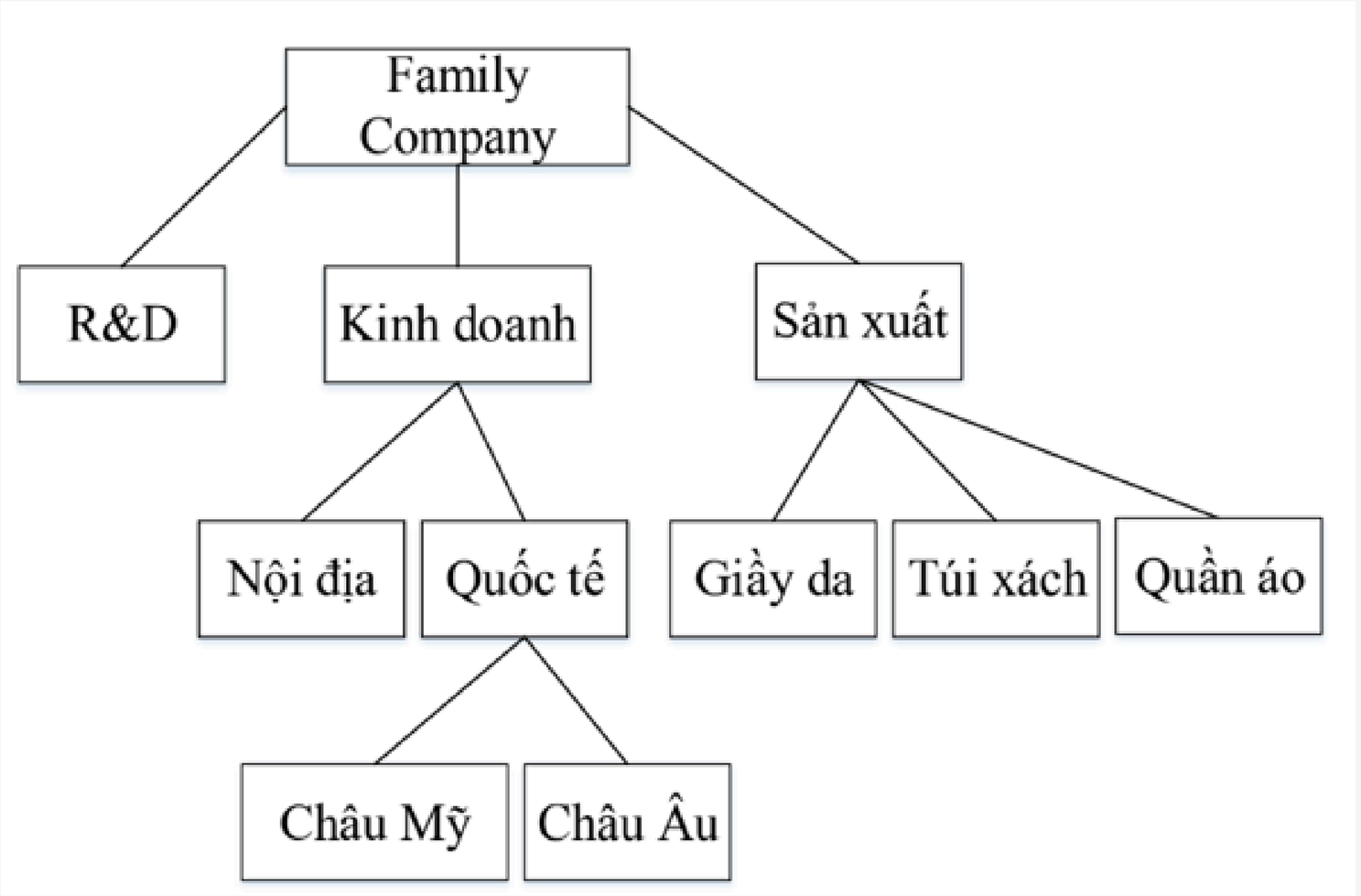
Cây có tính chất đệ quy. Tính chất này sẽ được sử dụng trong nhiều thao tác sau này.

Cây là một cấu trúc dữ liệu động, tức là kích thước của nó (số nút của cây) có thể thay đổi.

Cấu trúc cây không còn cấu trúc tuyến tính nữa mà là cấu trúc phân cấp.

Chỉ tồn tại duy nhất một đường đi từ gốc đến một nút khác.

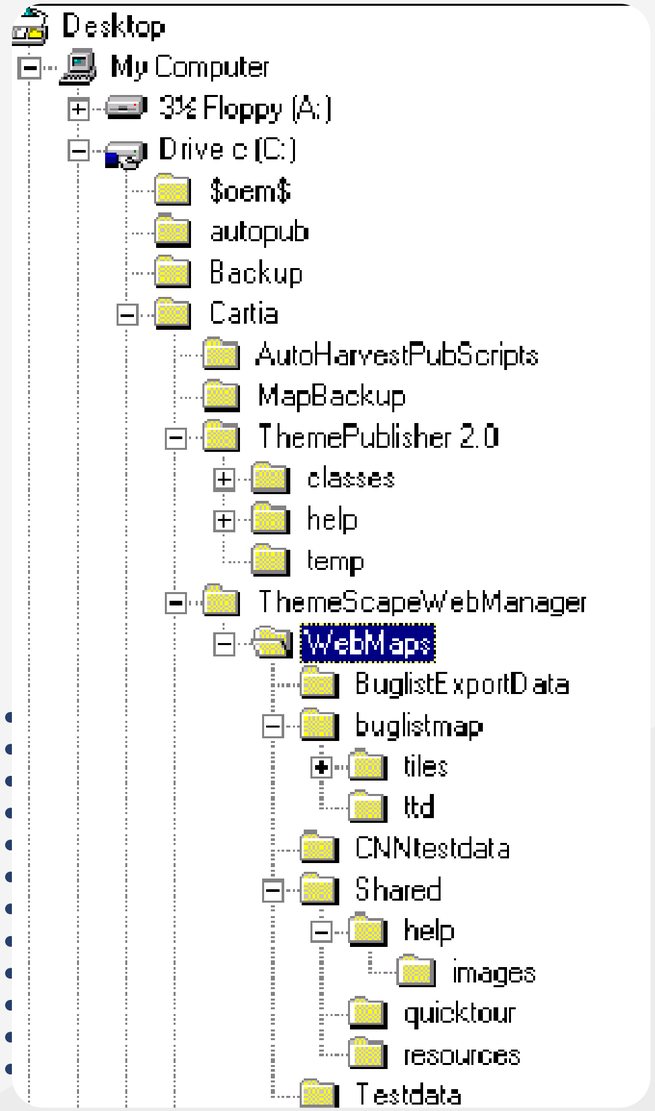
VÍ DỤ VỀ CÁC ĐỐI TƯỢNG CÓ CẤU TRÚC CÂY



FAMILY COMPANY

VÍ DỤ

CÂY THƯ MỤC TRONG Ổ CỨNG MÁY TÍNH



BIỂU THỨC TOÁN HỌC

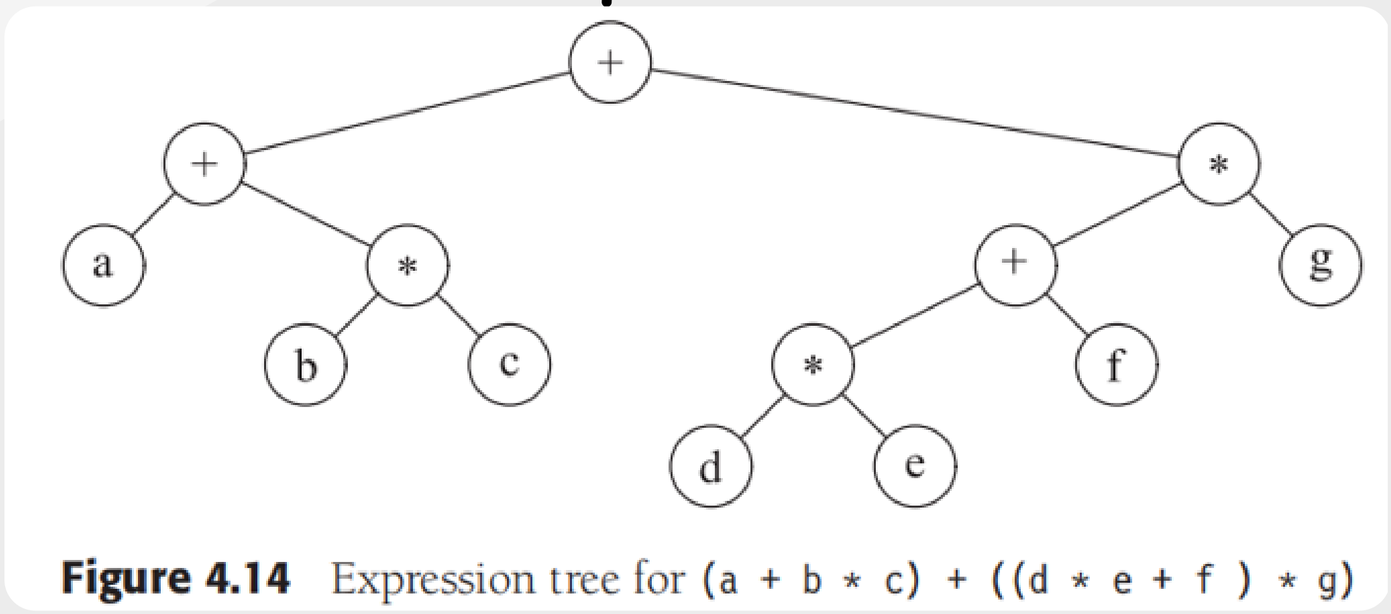
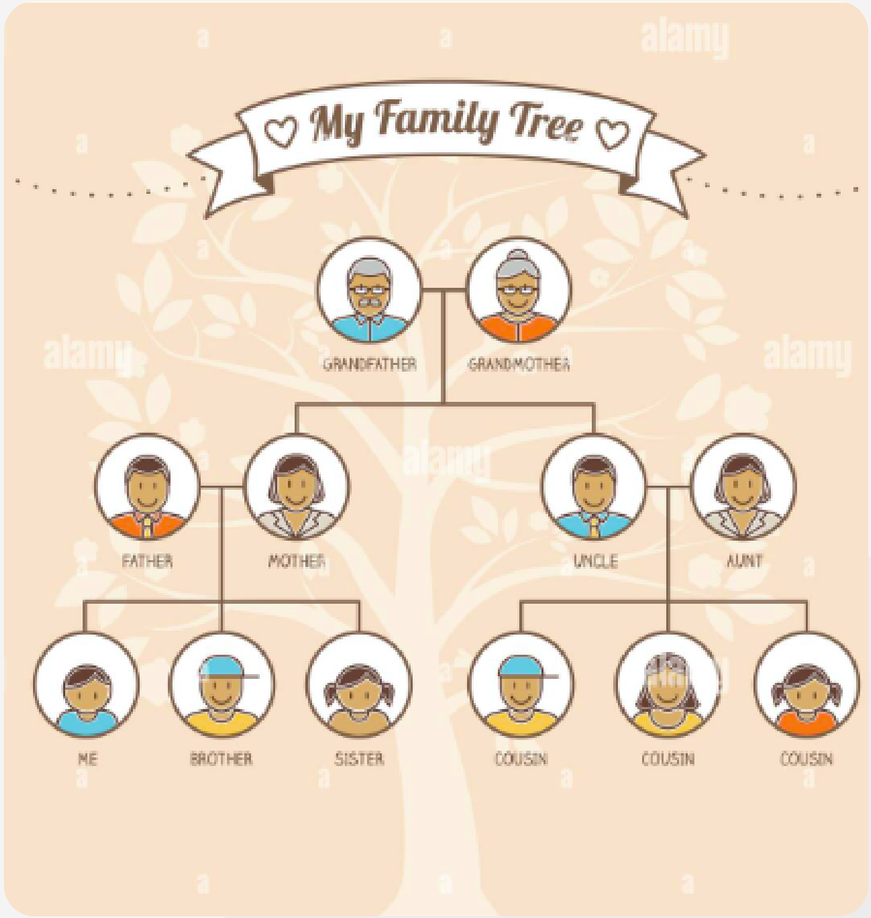


Figure 4.14 Expression tree for (a + b * c) + ((d * e + f) * g)

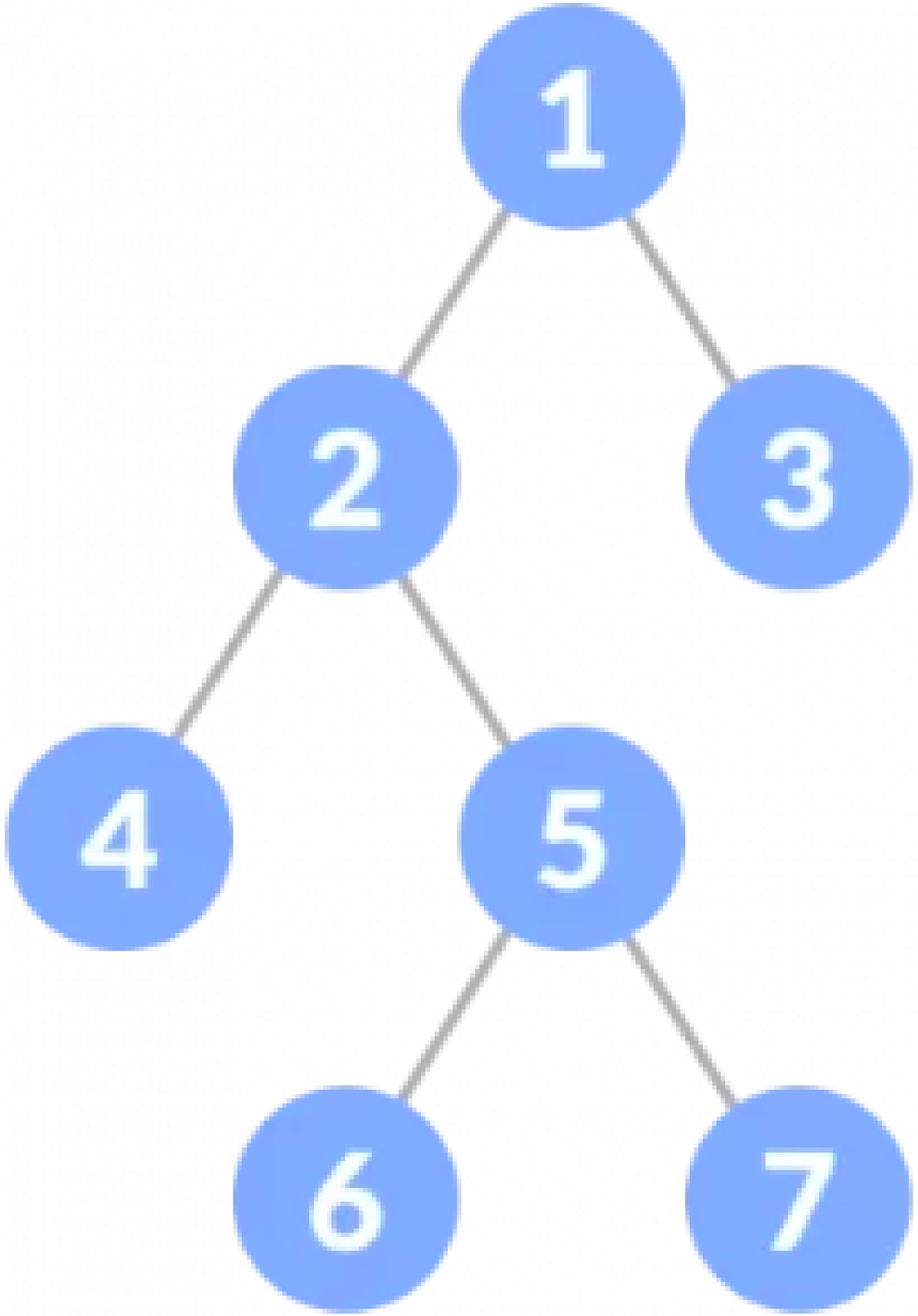
VÍ DỤ



GIA PHẢ CỦA MỘT DÒNG HỌ

Contents

MỤC LỤC CỦA MỘT CUỐN SÁCH



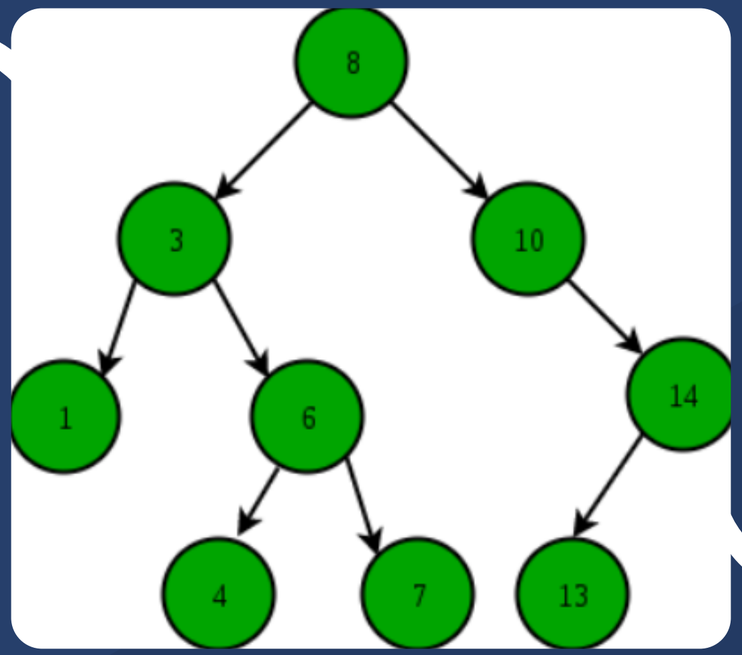
II. CÂY NHỊ PHÂN

- Khái niệm
- Phân loại
- Tính chất

KHÁI NIỆM CỦA CÂY NHỊ PHÂN (BINARY TREE)

KHÁI NIỆM

Hai con của một nút được phân biệt thứ tự và quy ước nút trước gọi là nút con trái và nút sau được gọi là nút con phải



Mỗi node có tối đa 2 cây con

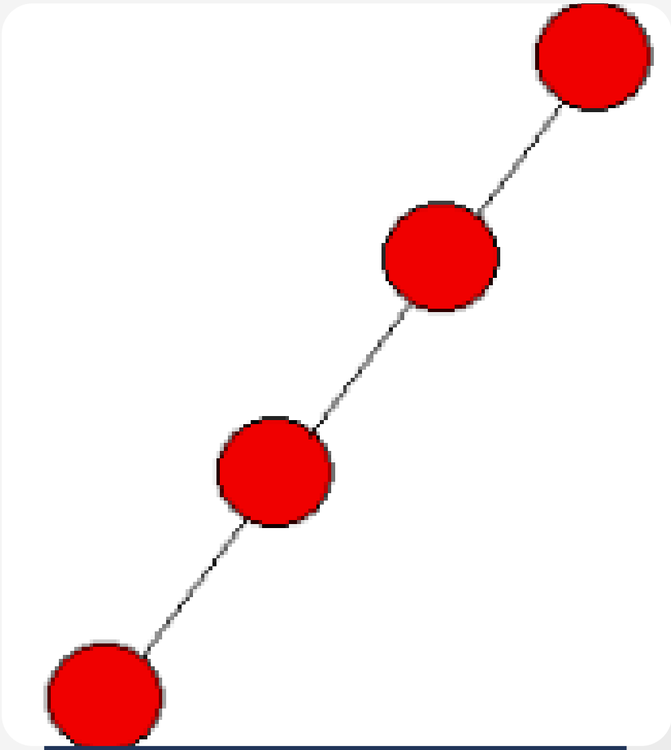
MỘT SỐ DẠNG ĐẶC BIỆT CỦA CÂY NHỊ PHÂN

CÂY SUY BIẾN (DEGENERATE BINARY TREE)

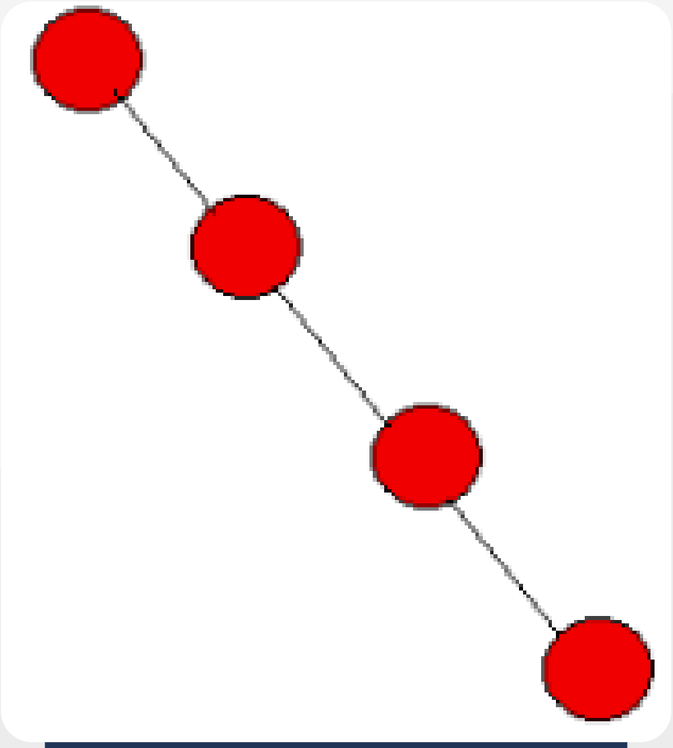
ĐỊNH NGHĨA

- Là cây nhị phân mà mỗi nút chỉ có tối đa một con. Nó đã bị suy biến về cấu trúc danh sách.
- Như vậy, danh sách chỉ là dạng suy biến, trường hợp đặc biệt của cấu trúc cây, và nó vẫn giữa được tính chất đệ quy của cây

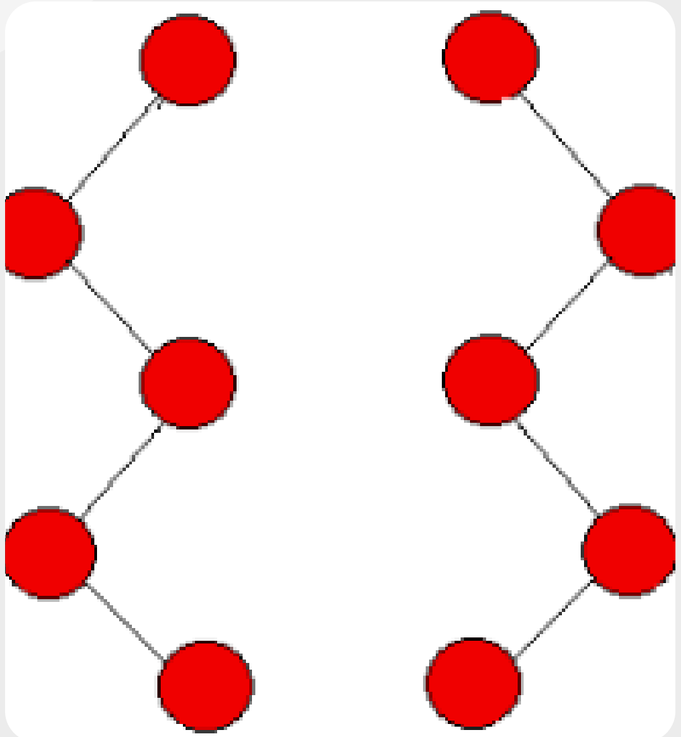
CÂY SUY BIẾN (DEGENERATE BINARY TREE)



CÂY LỆCH TRÁI



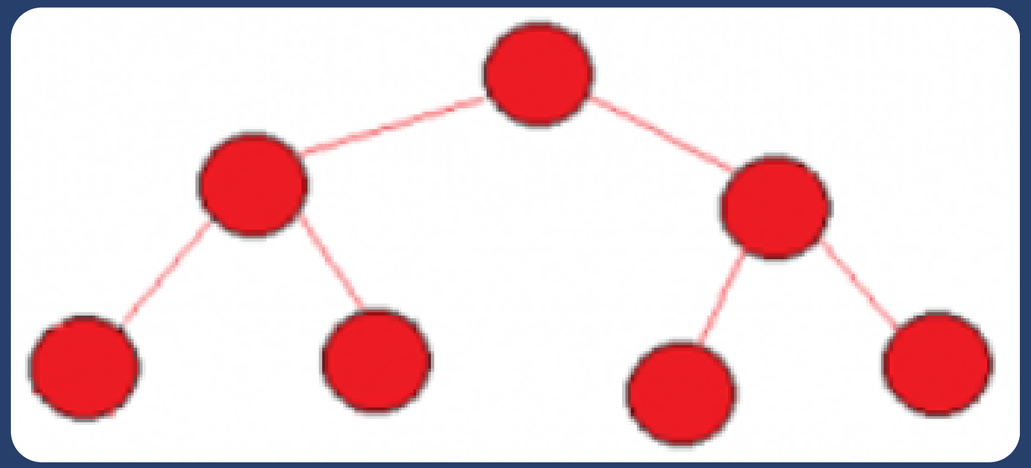
CÂY LỆCH PHẢI



CÂY ZICZAC

CÂY NHỊ PHÂN ĐẦY ĐỦ (FULL BINARY TREE)

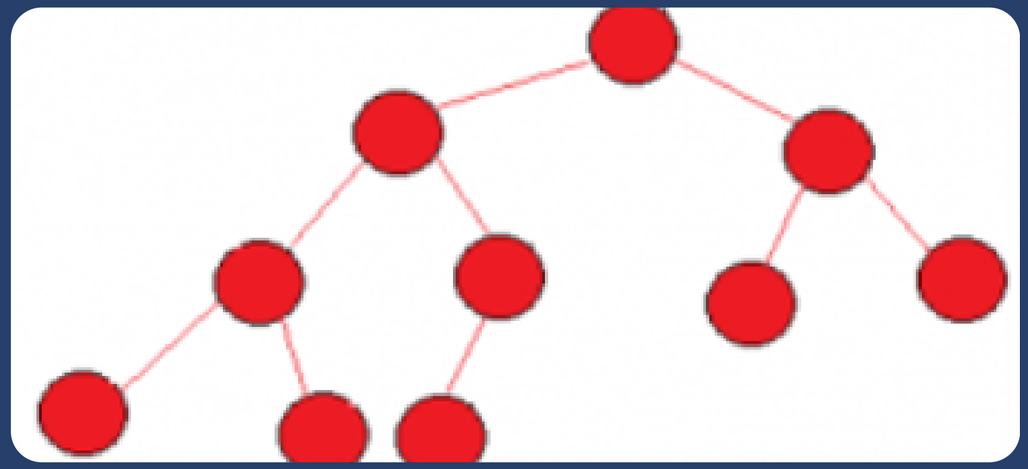
ĐỊNH NGHĨA



Mọi nút có mức nhỏ hơn hoặc bằng $h-1$ đều có đúng 2 nút con. Đây có thể xem là trường hợp đặc biệt của cây nhị phân hoàn chỉnh.

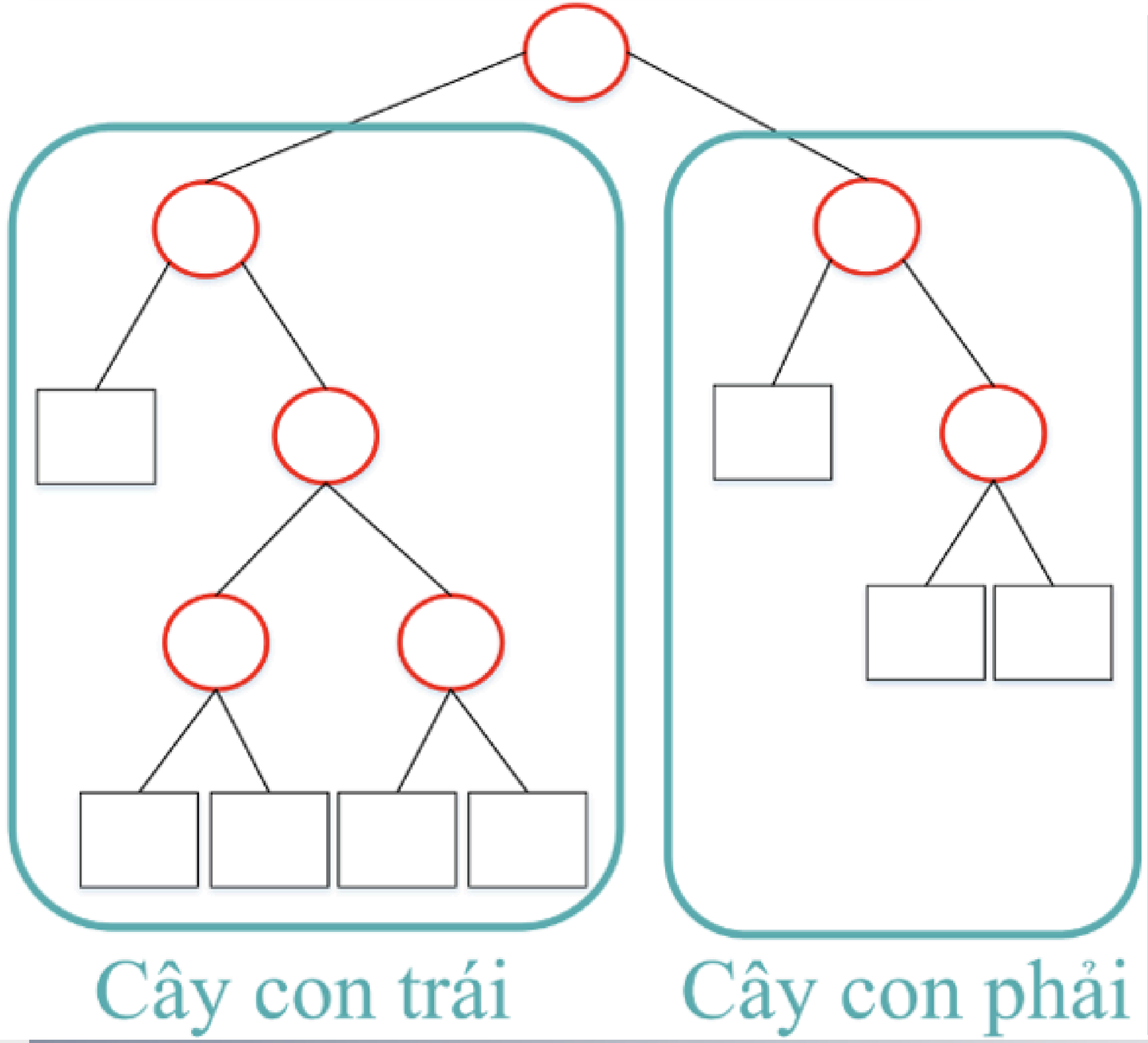
CÂY HOÀN CHỈNH (COMPLETE BINARY TREE)

ĐỊNH NGHĨA



-Gọi h là chiều cao của cây T. Ta gọi nó là cây hoàn chỉnh nếu:

- T là cây đầy đủ đến chiều cao h-1
- Các nút có chiều cao h thì dồn hết về bên trái.



– Loại nút:

- Nút có đủ hai con được gọi là nút kép
- Nút chỉ có một con gọi là nút đơn
- Nút không có con được gọi là nút lá

– Cây con có gốc là nút con trái/phải được gọi là cây con trái/phải.

CÁC TÍNH CHẤT CỦA CÂY NHỊ PHÂN

TÍNH CHẤT CHUNG CỦA CÂY NHỊ PHÂN

-Trong các cây nhị phân có cùng số lượng nút như nhau thì cây nhị phân suy biến có chiều cao lớn nhất, còn cây nhị phân hoàn chỉnh có chiều cao nhỏ nhất.

->Gọi h và n lần lượt là chiều cao và kích thước của cây nhị phân thì ta luôn có: log2(n+1) ≤ h+1 ≤ n

SỐ NÚT, NODE

-Số node nằm ở mức i: 1 ≤ n ≤ 2^i

-Số nút lá ≤ 2^h, với h là chiều cao của cây

-Cây nhị phân có chiều cao h (h ≥ 0) sẽ có tối đa 2^(h+1) -1 node

TÍNH ĐỆ QUY

Nếu ta chặt một nhánh bất kì của cây nhị phân thì ta sẽ thu được hai cây con cũng đều là cây nhị phân.

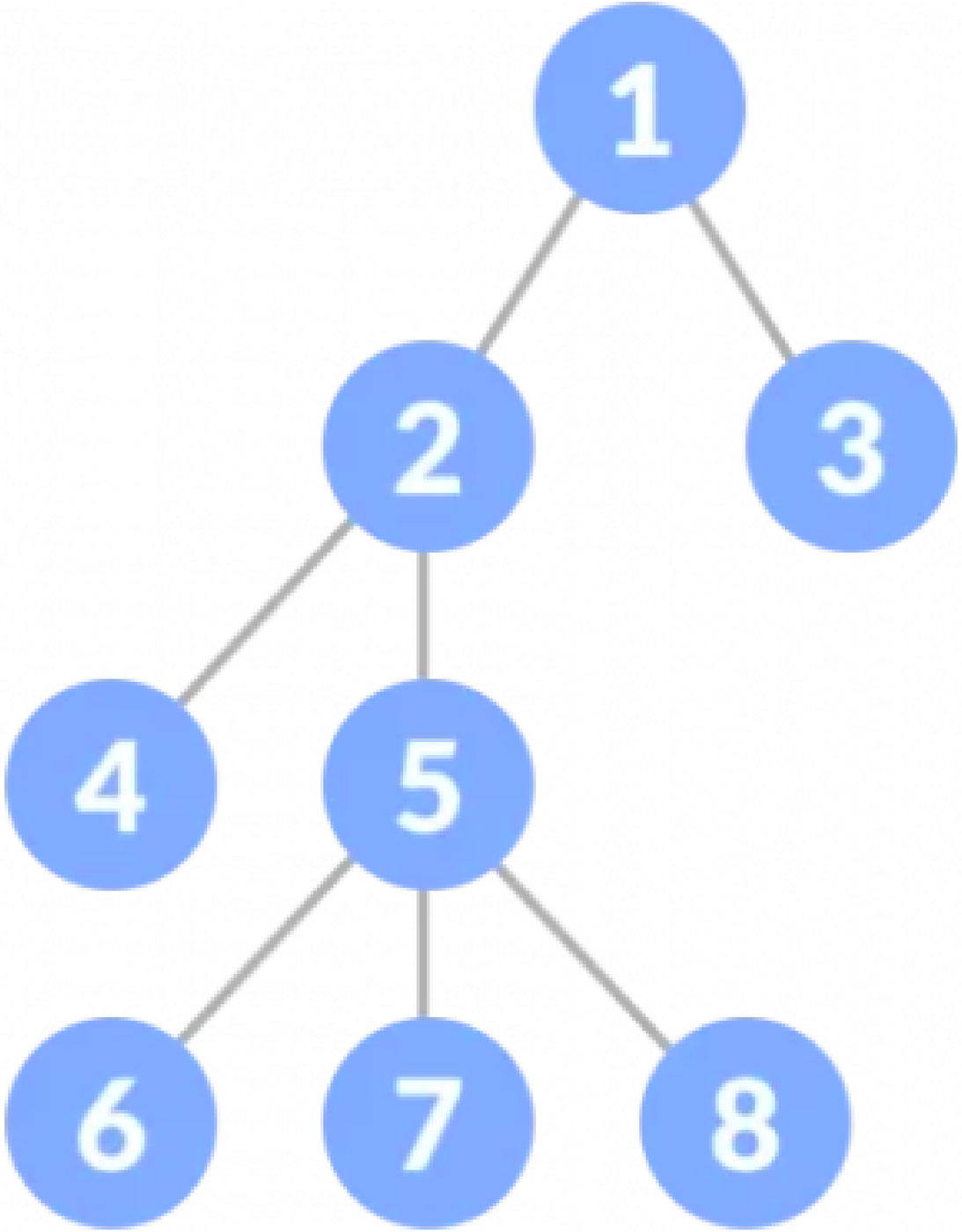
TÍNH CHẤT CÂY HOÀN CHỈNH VÀ CÂY ĐẦY ĐỦ

TẠI NÚT CÓ SỐ THỨ TỰ I

- Nếu 2i>N: nút i là nút lá
- Nếu 2i=N: nút i là nút đơn
- Nếu 2i<N: nút i là nút kép và có hai con trái, phải tương ứng là 2i và 2i+1

TẠI NÚT CÓ SỐ THỨ TỰ J

- Nếu j=1: nút j là nút gốc (không có nút cha)
- Nếu j>1: nút j/2 (nếu j chẵn) hoặc (j-1)/2 (nếu j lẻ) là nút cha của nút j.



III. BIỂU DIỄN CÂY NHỊ PHÂN

- Mảng
- DSLK đơn

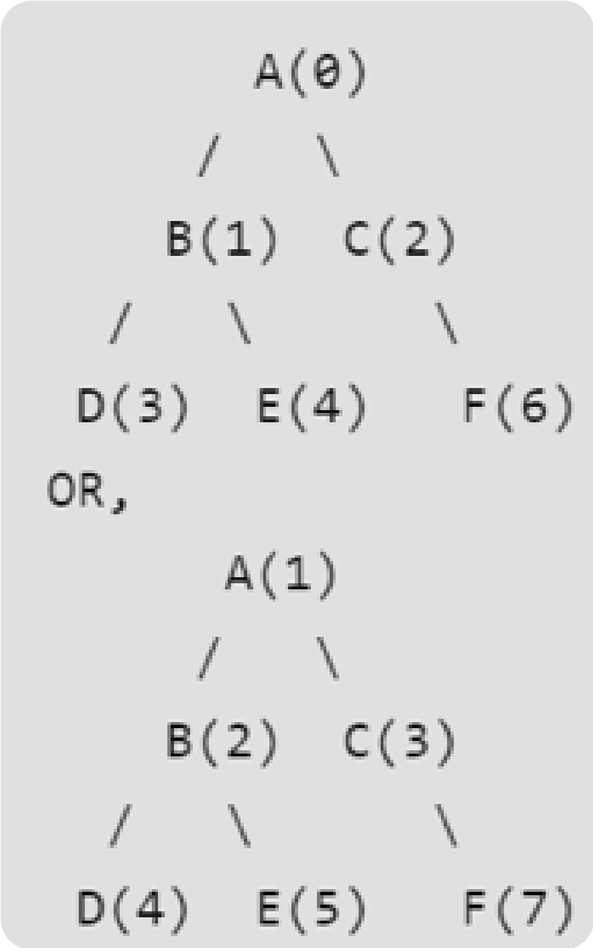
A. Biểu diễn theo mảng

Khi nào cần biểu diễn theo mảng

Cây nhị phân là cây đầy đủ (full binary tree) hoặc cây nhị phân hoàn chỉnh(complete binary tree).

Không có sự chèn/xóa nút trong cây nhị phân. Kích thước của cây là cố định và được xác định trước.

Chỉ số bắt đầu từ 0 -> (n-1)
(Giả sử chỉ số của một phần tử là i)

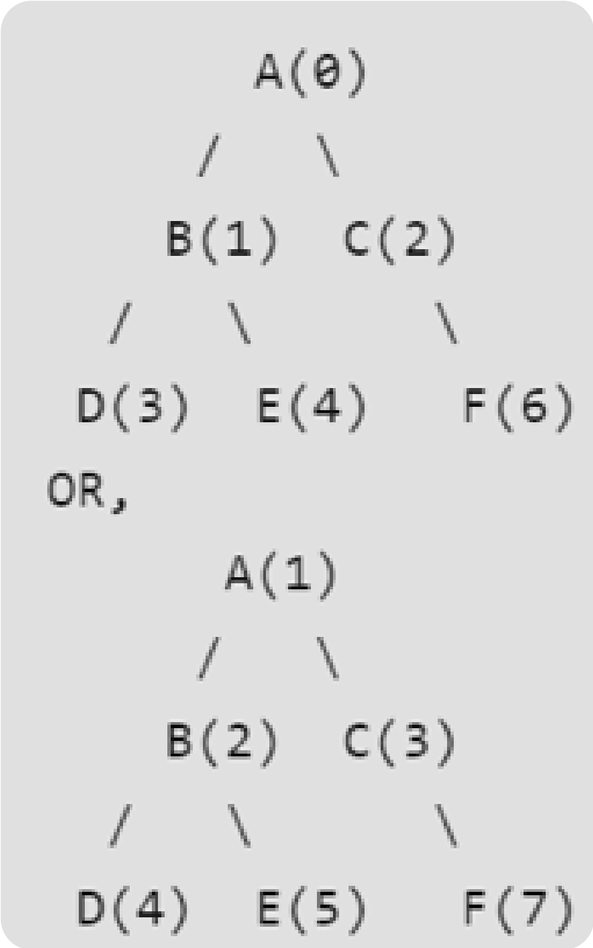


01 CHỈ SỐ CON TRÁI: [(2*I) + 1]

02 CHỈ SỐ CON PHẢI: [(2*I) + 2]

03 CHỈ SỐ NÚT CHA: [(I-1)/2]

Chỉ số bắt đầu từ 1->n
(Giả sử chỉ số của một phần tử là i)



01 CHỈ SỐ CON TRÁI: (2*I)

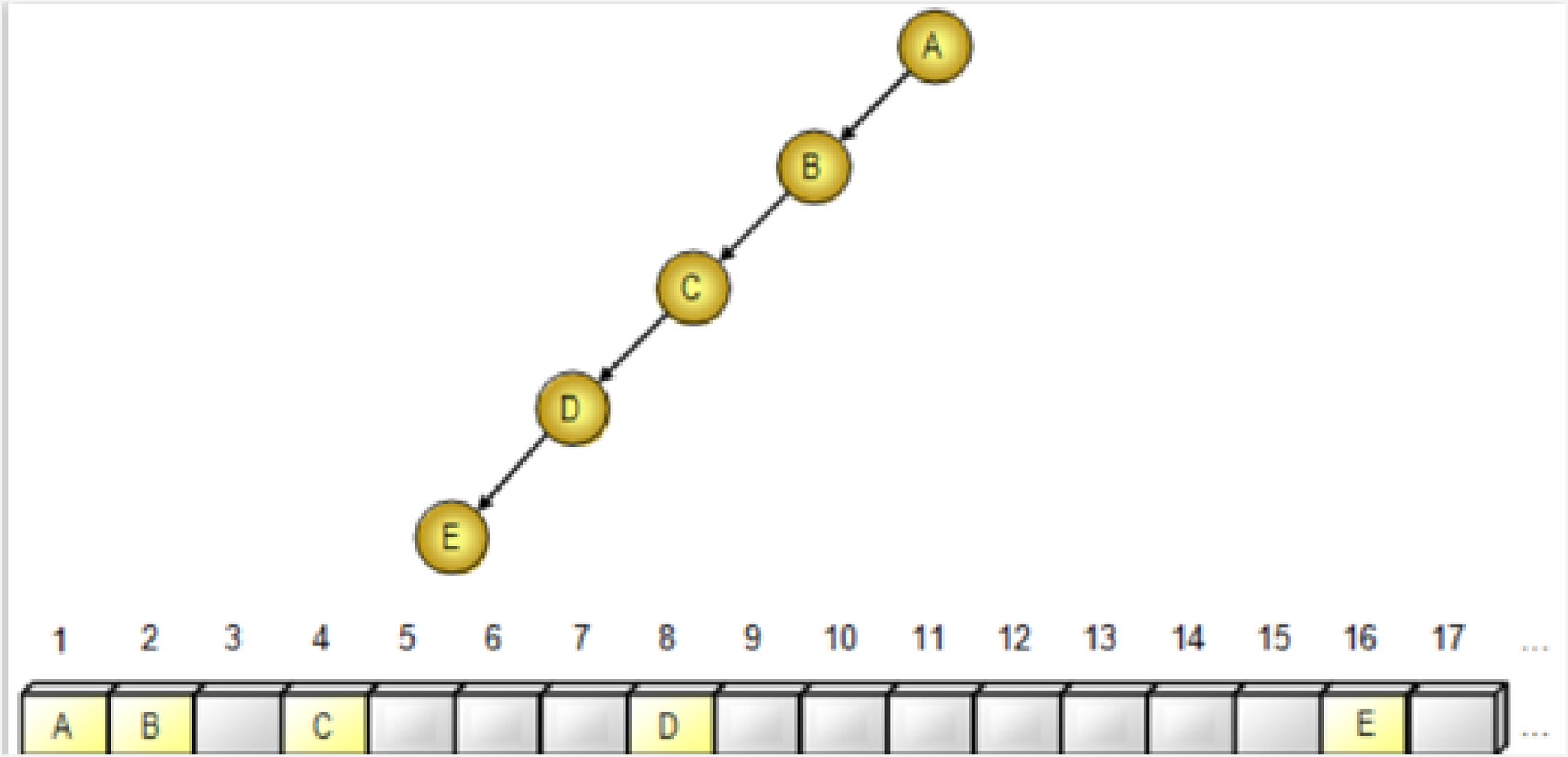
02 CHỈ SỐ CON PHẢI: [(2*I) + 1]

03 CHỈ SỐ NÚT CHA : [I/2]

BIỂU DIỄN THEO MẢNG

LƯU Ý

- Kích thước của mảng cần để biểu diễn một cây nhị phân có chiều cao h = 2^(h+1) - 1
- Đối với cây nhị phân không đầy đủ, ta có thể thêm vào một số nút giả để được cây nhị phân đầy đủ và gán những giá trị đặc biệt cho những phần tử trong mảng T tương ứng với những nút này hoặc dùng thêm một mảng phụ để đánh dấu những nút nào là nút giả tự ta thêm vào



VÍ DỤ: VỚI CÂY NHỊ PHÂN MỨC 4 TA CẦN MẢNG VỚI 2^(4+1) – 1 =31 Ô NHỚ ĐỂ LƯU TRỮ CÂY NHƯNG ĐỐI VỚI CÂY NHỊ PHÂN SUY BIẾN, SỐ Ô NHỚ THỰC SỰ CHỨA DỮ LIỆU CHỈ LÀ 5

BIỂU DIỄN THEO MẢNG

LỢI ÍCH

- **Truy cập dễ dàng: bằng việc tính toán các chỉ số ta có thể thao tác với các nút dễ dành.**
- **Tiết kiệm không gian lưu trữ: Nếu là cây nhị phân đầy đủ mảng tiết kiệm không gian hơn danh sách liên kết. Ta chỉ cần tạo một mảng cố định, không cần phải tạo các con trỏ.**
- **Hiệu suất tốt: Do truy cập vào các phần tử trong mảng là một thao tác có hiệu suất cao.**

BIỂU DIỄN THEO MẢNG

HẠN CHẾ

- **Kích thước cố định: Khi sử dụng biểu diễn mảng, bạn cần xác định kích thước của mảng từ đầu.**
- **Lãng phí không gian: Nếu cây nhị phân không đầy đủ, biểu diễn bằng mảng có thể dẫn đến lãng phí không gian.**

BIỂU DIỄN THEO MẢNG

HẠN CHẾ

- **Khó khắc phục khi chèn/xóa nút: Khi chèn hoặc xóa một nút trong cây, biểu diễn bằng mảng yêu cầu thao tác dịch chuyển các phần tử để tạo vị trí mới cho nút được chèn hoặc loại bỏ. (Điều này có thể làm tăng độ phức tạp và yêu cầu thêm thao tác so với biểu diễn bằng danh sách liên kết).**

```cpp
#include<iostream>
using namespace std;

char tree[10];
int root(char key) {
    if (tree[0] != '\0')
        cout << "Tree already had root";
    else
        tree[0] = key;
    return 0;
}

int set_left(char key, int parent) {
    if (tree[parent] == '\0')
        cout << "\nCan't set child at "
        << (parent * 2) + 1
        << " , no parent found";
    else
        tree[(parent * 2) + 1] = key;
    return 0;
}
```

```cpp
int set_right(char key, int parent) {
    if (tree[parent] == '\0')
        cout << "\nCan't set child at "
        << (parent * 2) + 2
        << " , no parent found";
    else
        tree[(parent * 2) + 2] = key;
    return 0;
}

int print_tree() {
    cout << "\n";
    for (int i = 0; i < 10; i++) {
        if (tree[i] != '\0')
            cout << tree[i];
        else
            cout << "-";
    }
    return 0;
}

// Driver Code
int main() {
    root('A');
    set_left('B', 0);
    set_right('C', 0);
    set_left('D', 1);
    set_right('E', 1);
    set_right('F', 2);
    print_tree();
    return 0;
}
```

BIỂU DIỄN THEO MẢNG

KẾT QUẢ ĐOẠN CODE

```
ABCDE-F---
--------------------------------
Process exited after 0.009277 seconds
Press any key to continue . . .
```

B.Biểu diễn theo DSLK đơn

Khi nào cần biểu diễn theo DSLK đơn

-Cây nhị phân có cấu trúc linh hoạt và thay đổi thường xuyên, bao gồm việc chèn/xóa nút.

-Cây nhị phân không đầy đủ (non-full binary tree) hoặc không đầy đủ ở mức dưới cùng (non-complete binary tree).

BIỂU DIỄN THEO DSLK

LỢI ÍCH

- **Tiết kiệm không gian: Sử dụng danh sách liên kết, bạn chỉ cần cấp phát bộ nhớ cho các nút thực sự tồn tại trong cây. Không có không gian bị lãng phí cho các nút không tồn tại.**
- **Dễ dàng mở rộng: Bạn chỉ cần tạo một đối tượng mới cho nút và cập nhật liên kết tương ứng, không cần phải di chuyển dữ liệu hiện có.**

BIỂU DIỄN THEO DSLK

LỢI ÍCH

- **Linh hoạt trong cấu trúc: Biểu diễn cây nhị phân bằng danh sách liên kết cho phép thay đổi cấu trúc của cây dễ dàng.**
- **Cấu trúc không cố định: Bạn có thể tạo ra cây nhị phân với bất kỳ số lượng nút nào mà không cần xác định trước.**

BIỂU DIỄN THEO DSLK

HẠN CHẾ

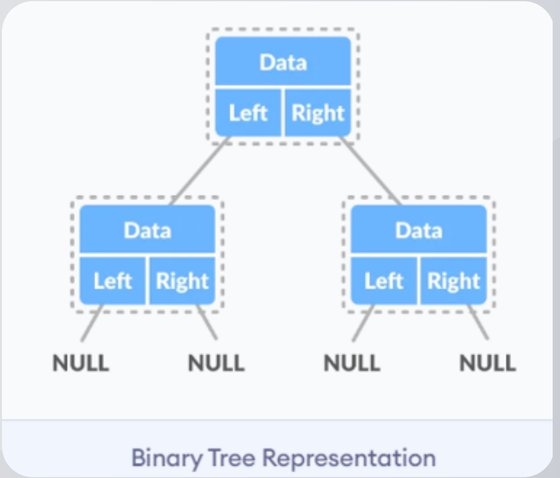
- **Tốn không gian bộ nhớ: So với biểu diễn bằng mảng, biểu diễn bằng danh sách liên kết tốn nhiều không gian bộ nhớ hơn trong trường hợp cùng là cây nhị phân đầy đủ.**
- **Truy cập không hiệu quả: Trong biểu diễn bằng danh sách liên kết, truy cập đến một nút trong cây đòi hỏi đi qua các con trỏ từ nút gốc đến nút cần truy cập.**

BIỂU DIỄN THEO DSLK

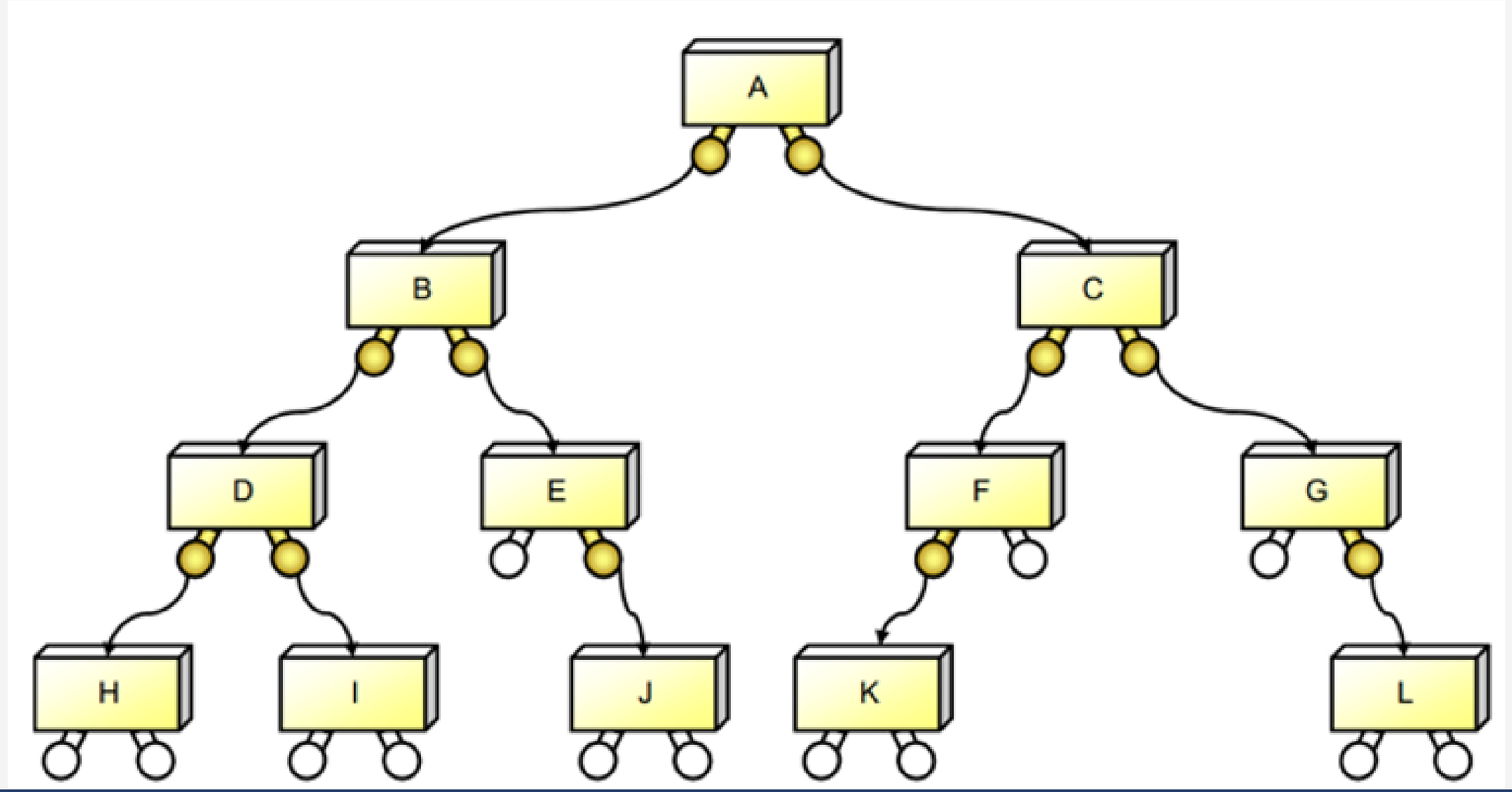
HẠN CHẾ

- **Không thể tiện ích cho các thao tác đồng thời: Biểu diễn bằng danh sách liên kết không tận dụng tốt việc thực hiện các thao tác đồng thời trên cây.**

MỘT NODE CỦA CÂY NHỊ PHÂN ĐƯỢC BIỂU DIỄN BỞI MỘT CẤU TRÚC GỒM 3 PHẦN



- Thành phần thông tin (data): chưa giá trị được lưu trữ của node, có thể là bất kỳ kiểu dữ liệu nào.
- Thành phần liên kết bên trái (left): lưu trữ địa chỉ của nút gốc của cây con bên trái. Kiểu dữ liệu là con trỏ trỏ vào node.
- Thành phần liên kết bên phải (right): lưu trữ địa chỉ của nút gốc của cây con bên phải. Kiểu dữ liệu là con trỏ trỏ vào node.


A
B
C
D
E
F
G
H
I
J
K
L

Cách định nghĩa cấu trúc dữ liệu dạng cây

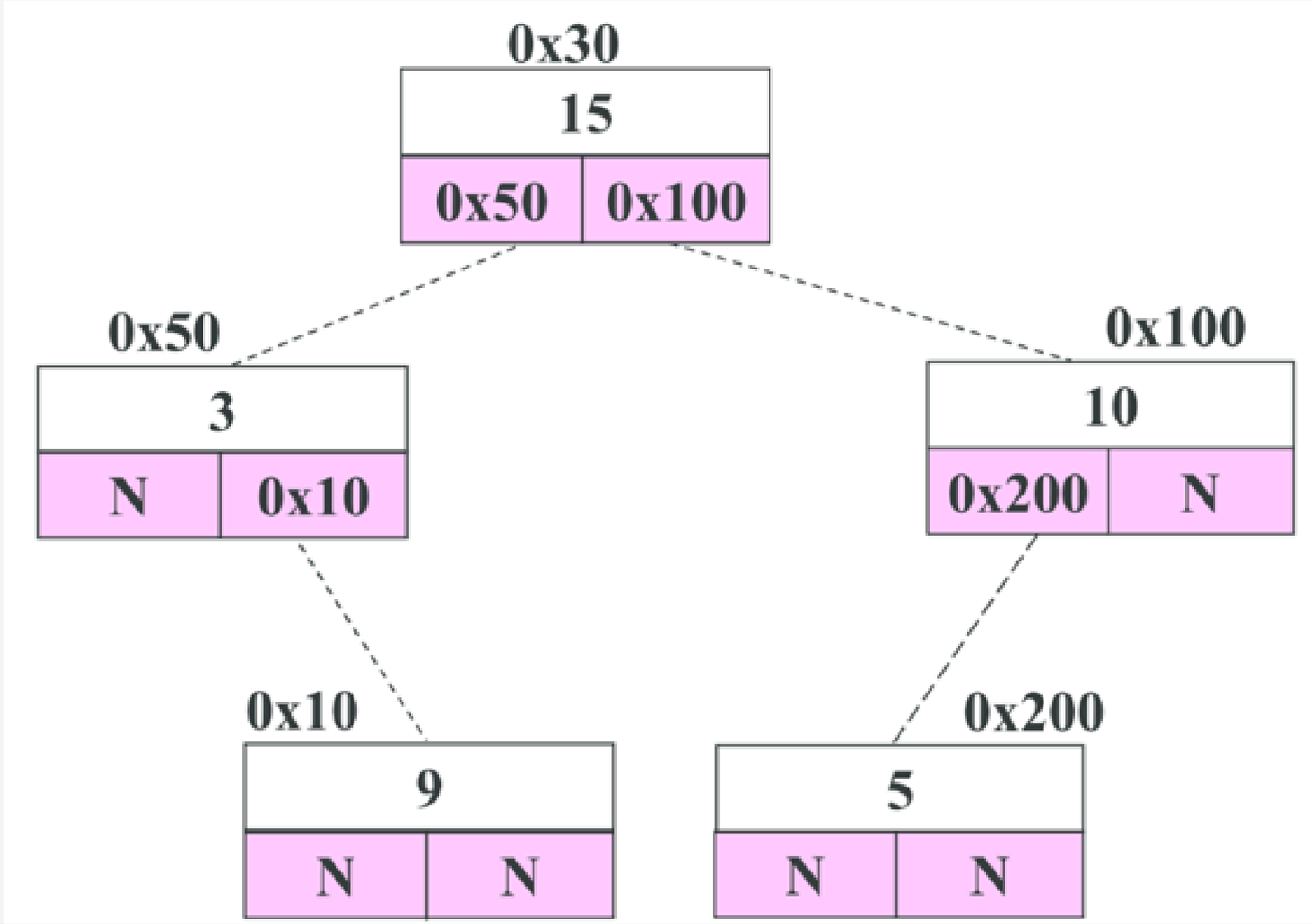
```
struct TreeNode {
    int data;
    TreeNode* left;
    TreeNode* right;
};

struct BinaryTree {
    TreeNode* root;
};
```

Tạo nút mới

```
TreeNode* makeNode(int data) {
    TreeNode* node = new TreeNode();
    node->data = data;
    node->left = node->right = nullptr;
    return node;
}
```

Khởi tạo một cây mới

```
void init(BinaryTree* tree) {
    tree->root = nullptr;
}
```



CÂY NHỊ PHÂN LƯU TRỮ DANH SÁCH CÁC SỐ SAU: 15, 3, 10, 9, 5

MỞ RỘNG

Đôi khi còn quan tâm đến cả quan hệ 2 chiều cha con chứ không chỉ một chiều như định nghĩa. Cấu trúc cây nhị phân như sau

```
typedef struct Node {
    DataType      Key;
    struct Node* pParent;
    struct Node* pLeft;
    struct Node* pRight;
}NODE;
typedef NODE* Tree;
```

THANK YOU

We look forward to working with you

OFFICE

UIT UNIVERSITY

+123-456-7890

www.dsa.com