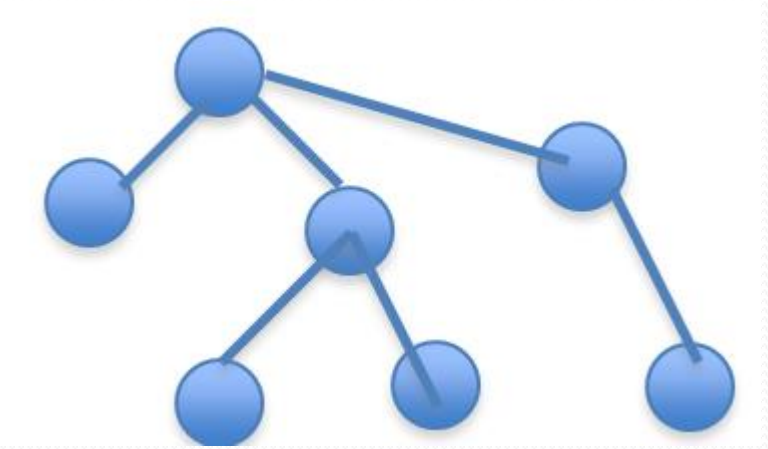
Cây nhị phân

Nội dung

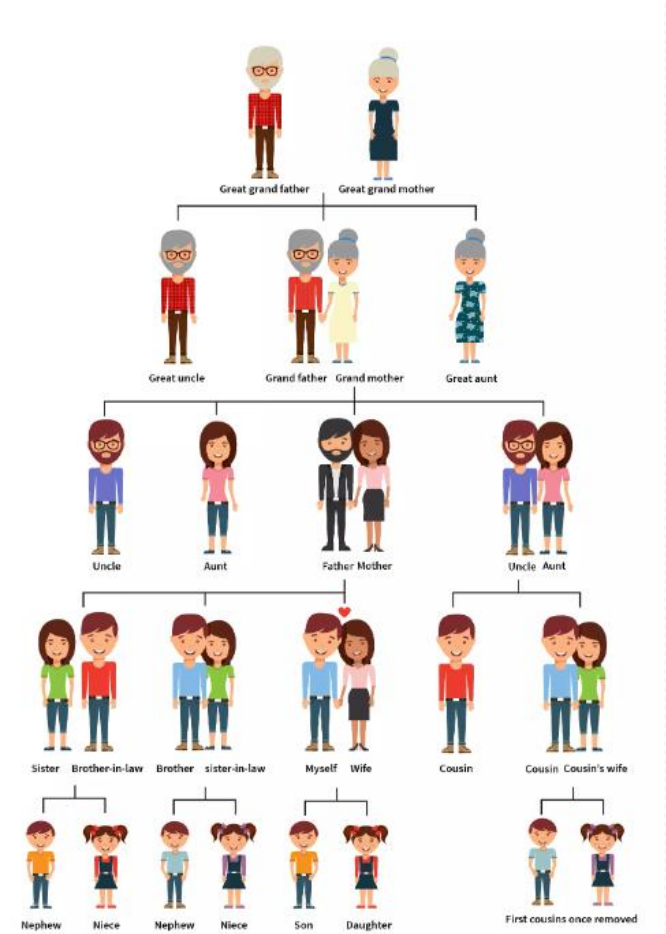
1. Khái niệm cây
2. Cây nhị phân
    1. Tổ chức dữ liệu
    2. Duyệt cây
3. Bài tập

1. Khái niệm cây

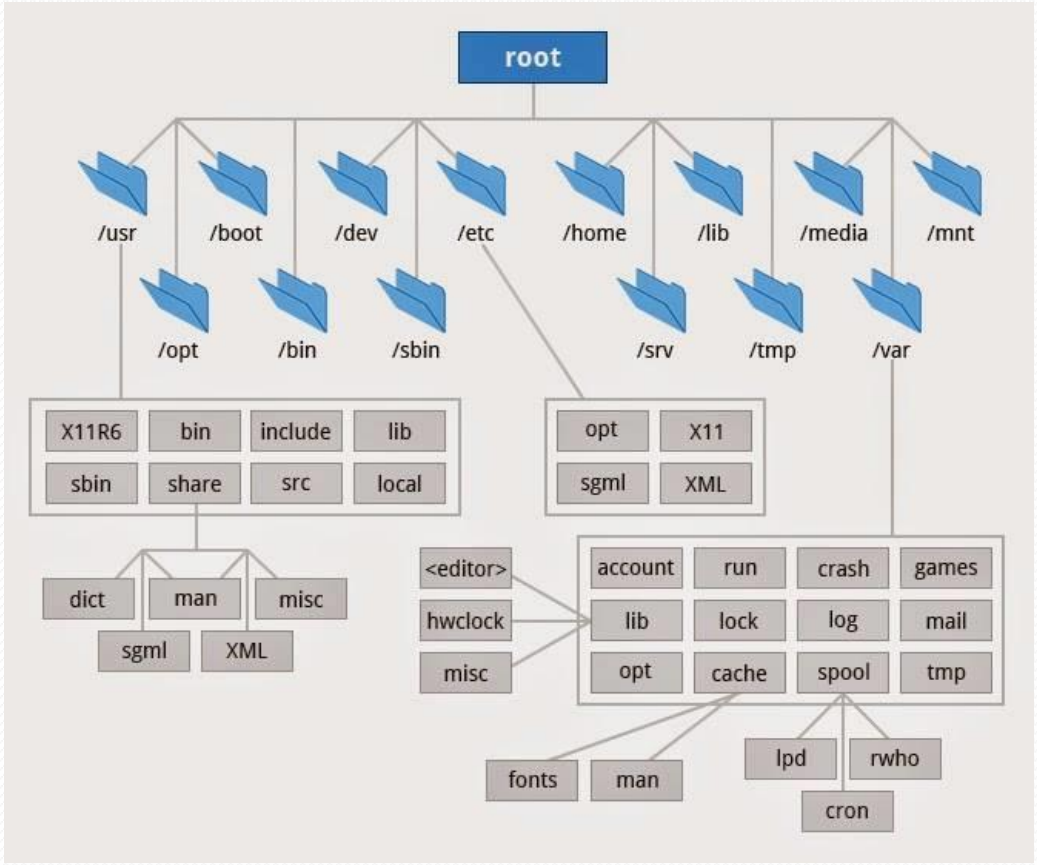
- Cây:
  - Gồm nhiều nút.
  - Quan hệ giữa các nút là quan hệ phân cấp.
  - Nút ở cấp cao nhất gọi là nút gốc.
- Ví dụ: cây gia phả, mục lục hay tổ chức của 1 cuốn sách, cây thư mục,...



Ví dụ cây

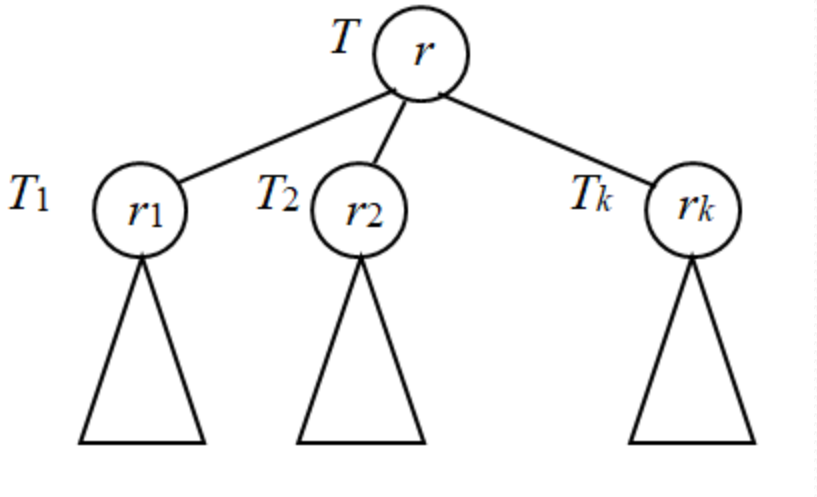


- Copyright
- Contents
- Preface
- Part I: Introduction
  - Chapter 1 Basic Concepts
    - 1.1 Pseudocode
    - 1.2 The Abstract Data Type
    - 1.3 Model for an Abstract Data Type
    - 1.4 ADT Implementations
    - 1.5 Generic Code for ADTs
    - 1.6 Algorithm Efficiency
    - 1.7 Key Terms
    - 1.8 Summary
    - 1.9 Practice Sets
    - Exercises
    - Problems
    - Projects
  - Chapter 2 Recursion
    - 2.1 Factorial—A Case Study
    - 2.2 Designing Recursive Algorithms
    - 2.3 Recursive Examples
    - 2.4 Key Terms
    - 2.5 Summary
    - 2.6 Practice Sets
    - Exercises
    - Problems
    - Projects
- Part II: Linear Lists

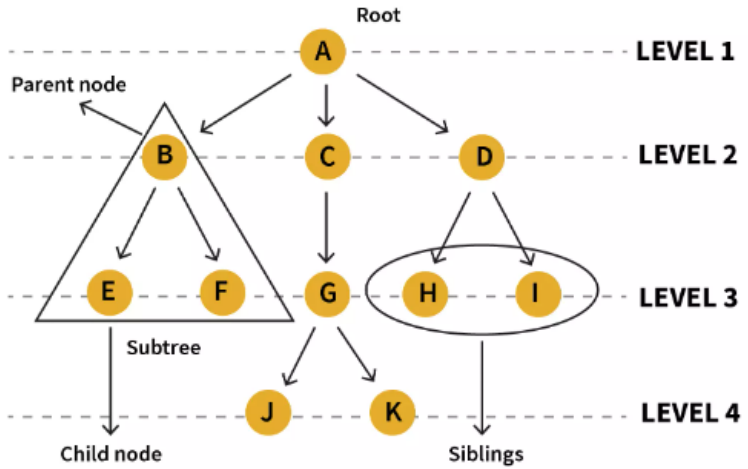


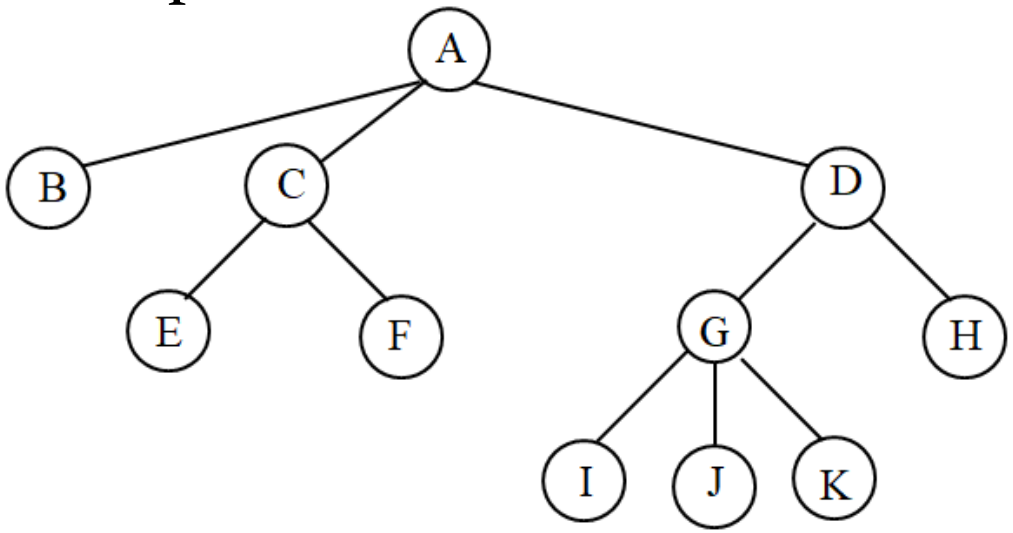
Định nghĩa cây

- Tập gồm 1 phần tử là 1 cây
- Cho T_1, T_2, ..., T_k là các cây với nút gốc là r_1, r_2, ..., r_k. r là một nút có quan hệ phân cấp (cấp trên) với r_1, r_2, ..., r_k. Khi đó tập gồm các phần tử của T_1, T_2, ..., T_k và r tạo thành một cây T có nút gốc là r.
- r_1, r_2, ... r_k gọi là các nút con của r.
- T_1, T_2, ..., T_k là các cây con của cây T.

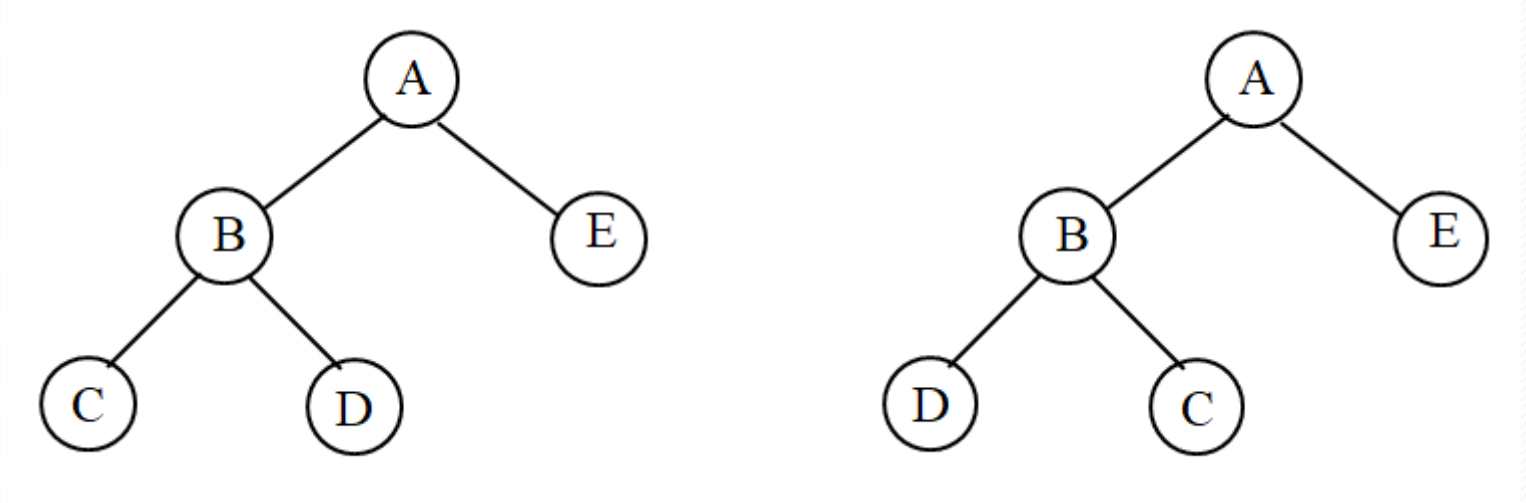


- Bậc của nút: số nút con của một nút.
- Bậc của cây: bậc của nút có bậc lớn nhất.
- Nút lá: là nút có bậc 0.
- Cấp của nút:
  - Nút gốc có cấp là 1.
  - Nút khác nút gốc có cấp là cấp của nút cha cộng thêm 1.
- Chiều cao của cây: là cấp của nút có cấp cao nhất.



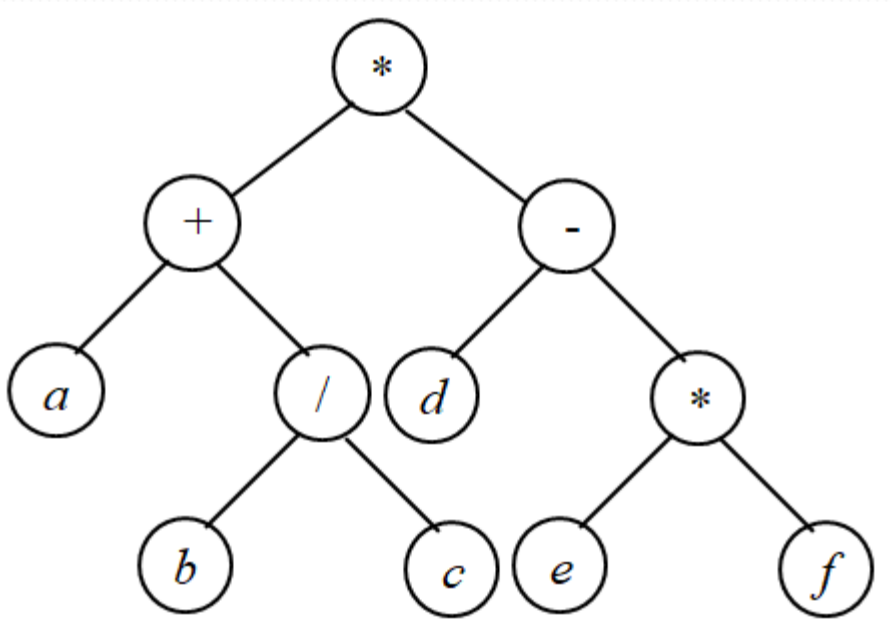


- Cây có thứ tự

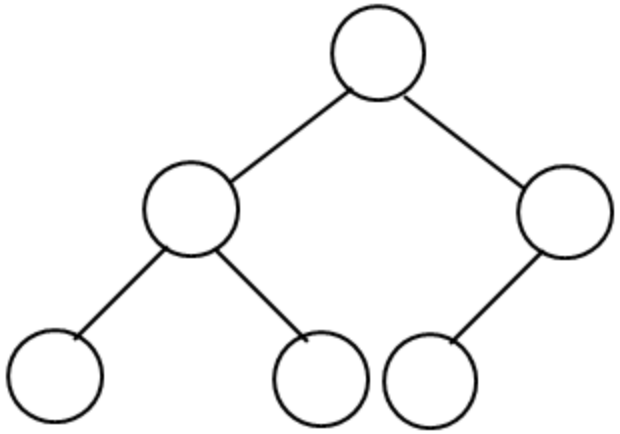


2. Cây nhị phân

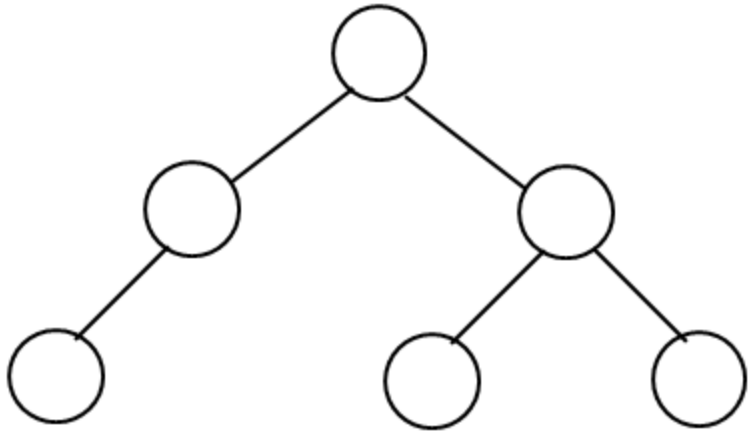
- Cây nhị phân: mỗi nút có không quá 2 nút con.
- Ví dụ: cây biểu diễn biểu thức (a + b/c)*(d-e*f)



- Cây nhị phân đầy đủ: cây nhị phân mà các nút lá trên cây nằm nhiều nhất ở hai mức liên tiếp nhau, và các lá ở mức dưới nằm dồn về phía bên trái

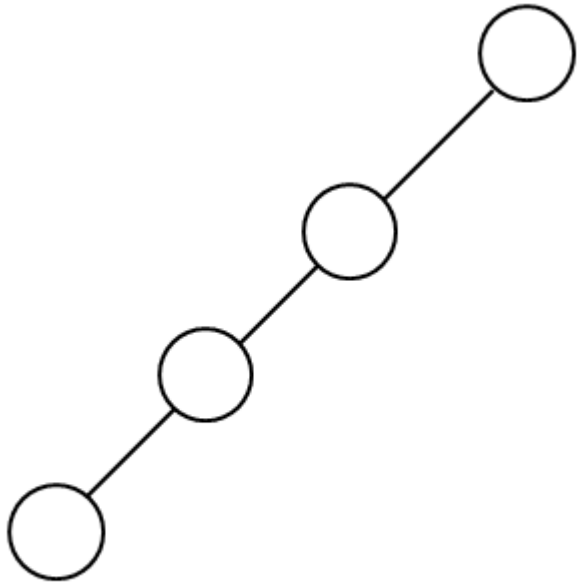


a) Cây nhị phân đầy đủ

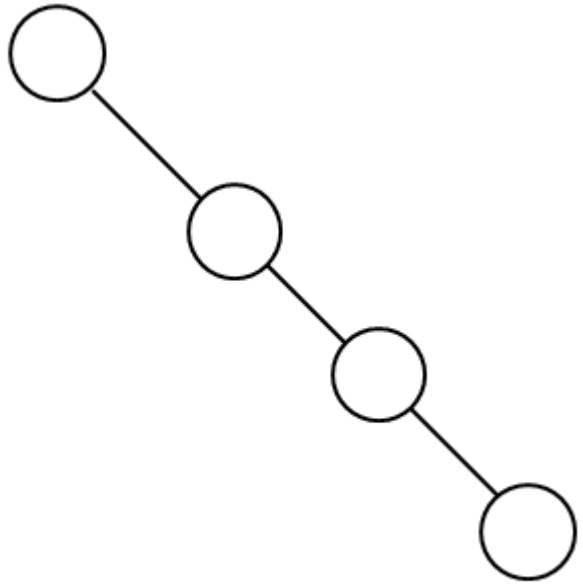


b) Cây nhị phân không đầy đủ

- Cây suy biến



a) Cây lệch trái

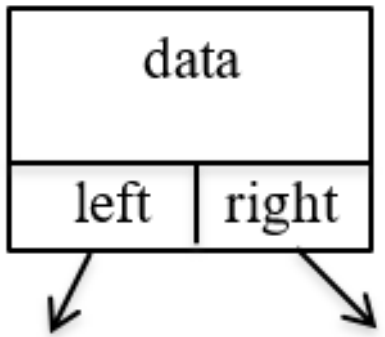


b) Cây lệch phải

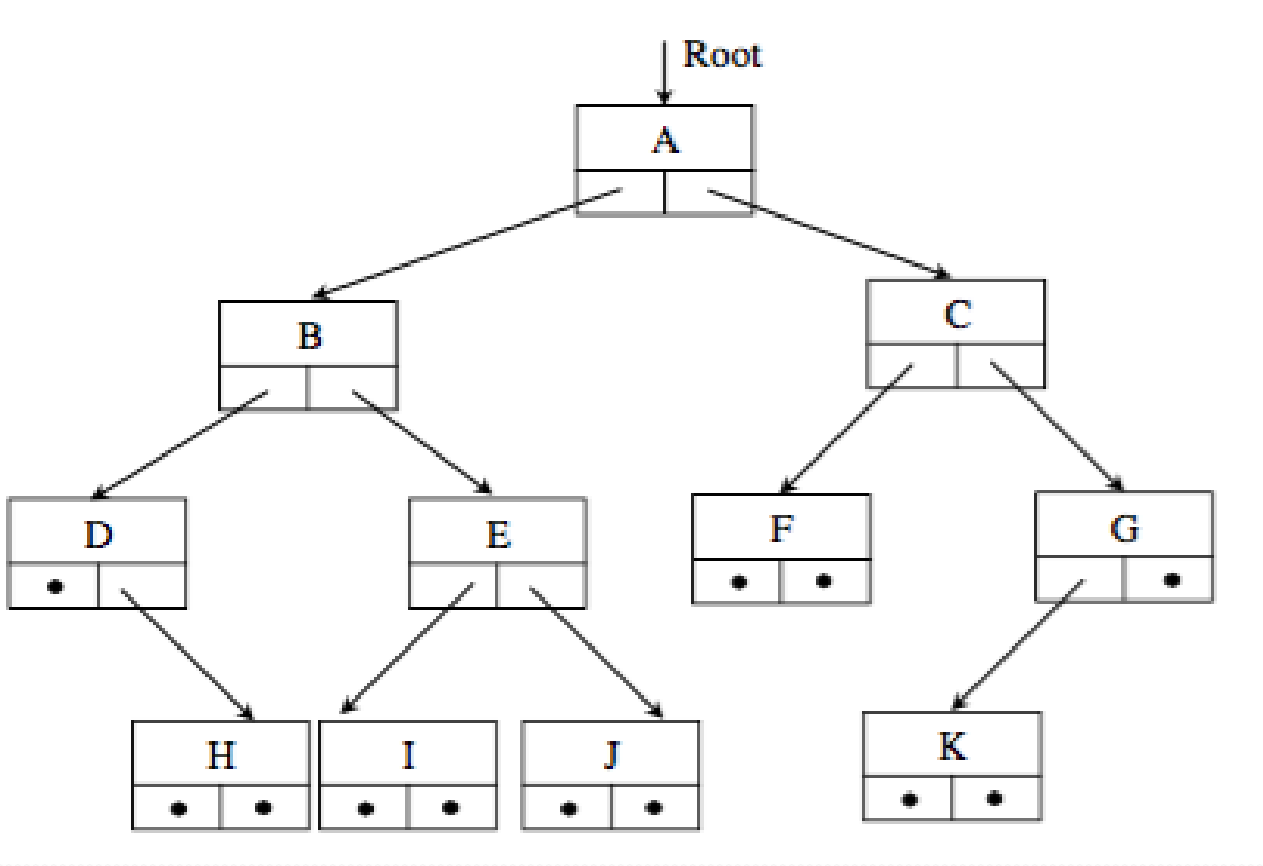
Tổ chức dữ liệu

- Mỗi nút được tổ chức thành 3 phần:
  - data: chứa dữ liệu của 1 phần tử.
  - left: liên kết đến nút con bên trái (nếu có).
  - right: liên kết đến nút con bên phải (nếu có).

hoTen: Nguyen Van A namSinh: 1900	
1000	1100



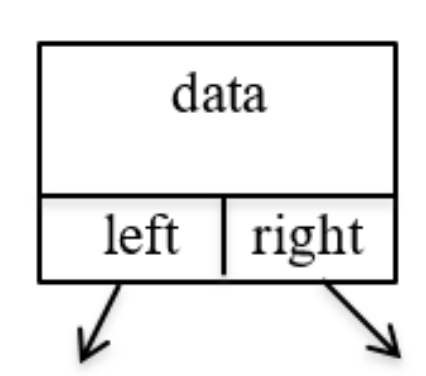
- Để quản lý cây nhị phân cần quản lý nút gốc

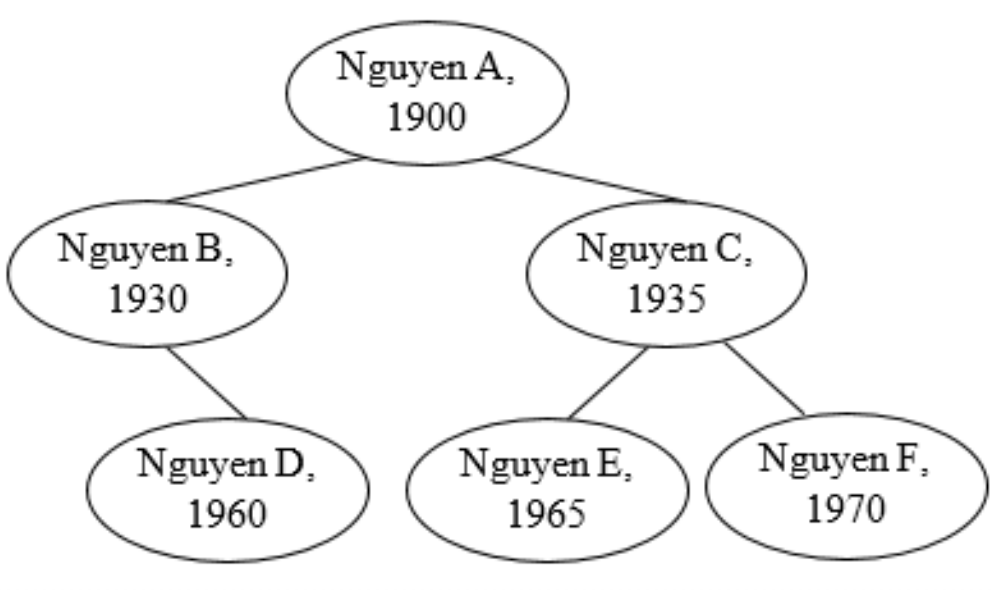


Khai báo

```
struct BinaryTreeNode{
   ElementType        data;
   BinaryTreeNode     *left, *right;
};

struct Person
{
   string name;
   int yearOfBirth;
};
struct BFT  //Binary Family Tree
{
   Person data;
   BFT *left, *right;
};
```





Duyệt cây nhị phân

- Lần lượt thăm tất cả các nút của cây, mỗi nút thăm đúng 1 lần.
- Có 3 cách duyệt cơ bản:
  - Duyệt theo thứ tự trước (PreOrder)
  - Duyệt theo thứ tự giữa (InOrder)
  - Duyệt theo thứ tự sau (PosOrder)

Duyệt theo thứ tự trước

- Thuật toán đệ quy:

Input: Cây nhị phân cần duyệt có nút gốc là root

Output: Thứ tự duyệt các nút của cây theo thứ tự trước

Action:

Nếu cây khác rỗng thì:

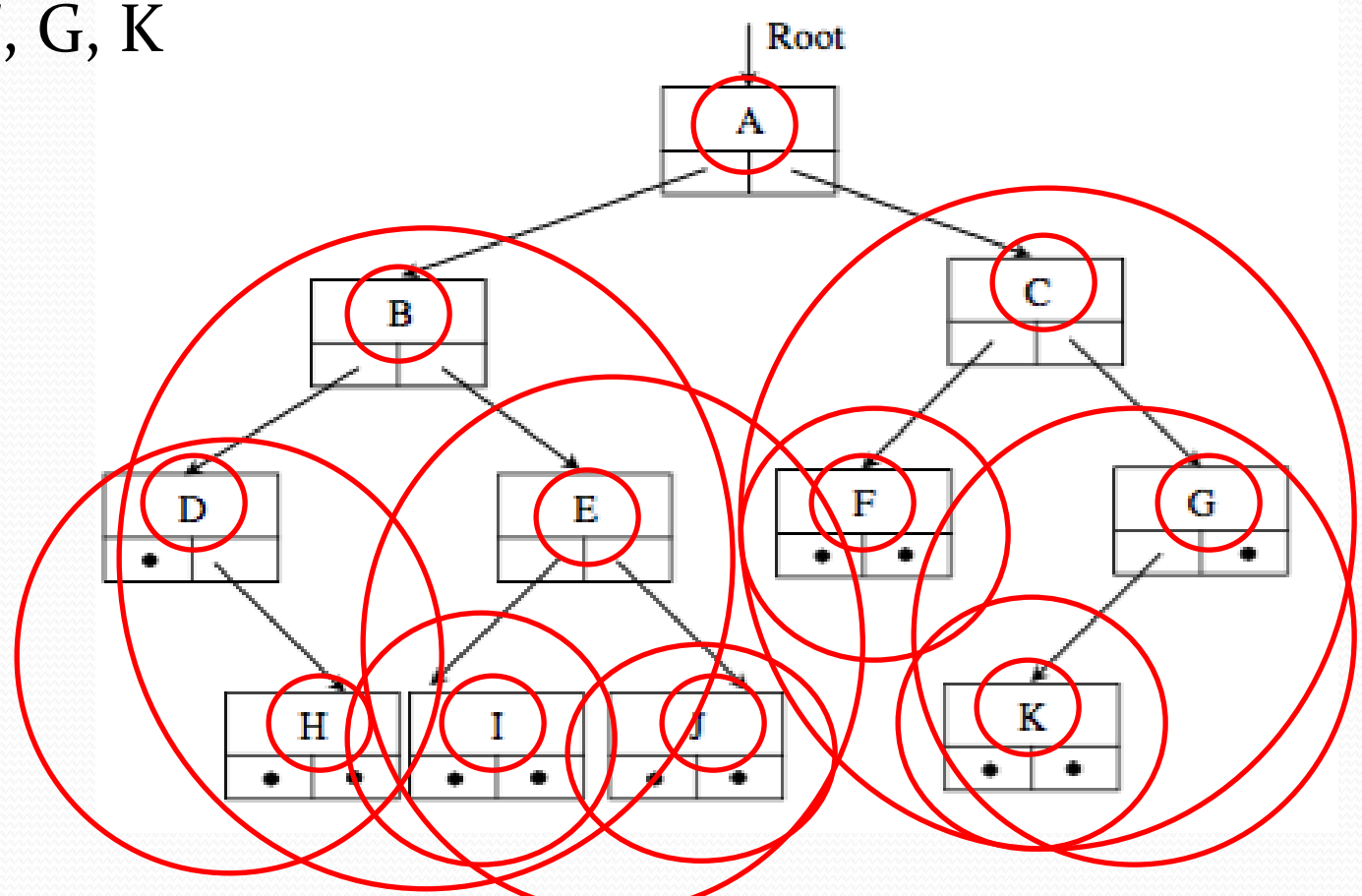
+ Thăm nút gốc

+ Duyệt theo thứ tự trước cây con trái

+ Duyệt theo thứ tự trước cây con phải

Ví dụ

- Duyệt theo thứ tự trước:
- A, B, D, H, E, I, J, C, F, G, K



Cài đặt

```
void preOrder(BinaryTreeNode *root)
{
      if (root != nullptr)
      {
      visit(root);
      preOrder(root->left);
      preOrder(root->right);
      }
}
```

Ví dụ: hàm in cây gia phả lên màn hình.

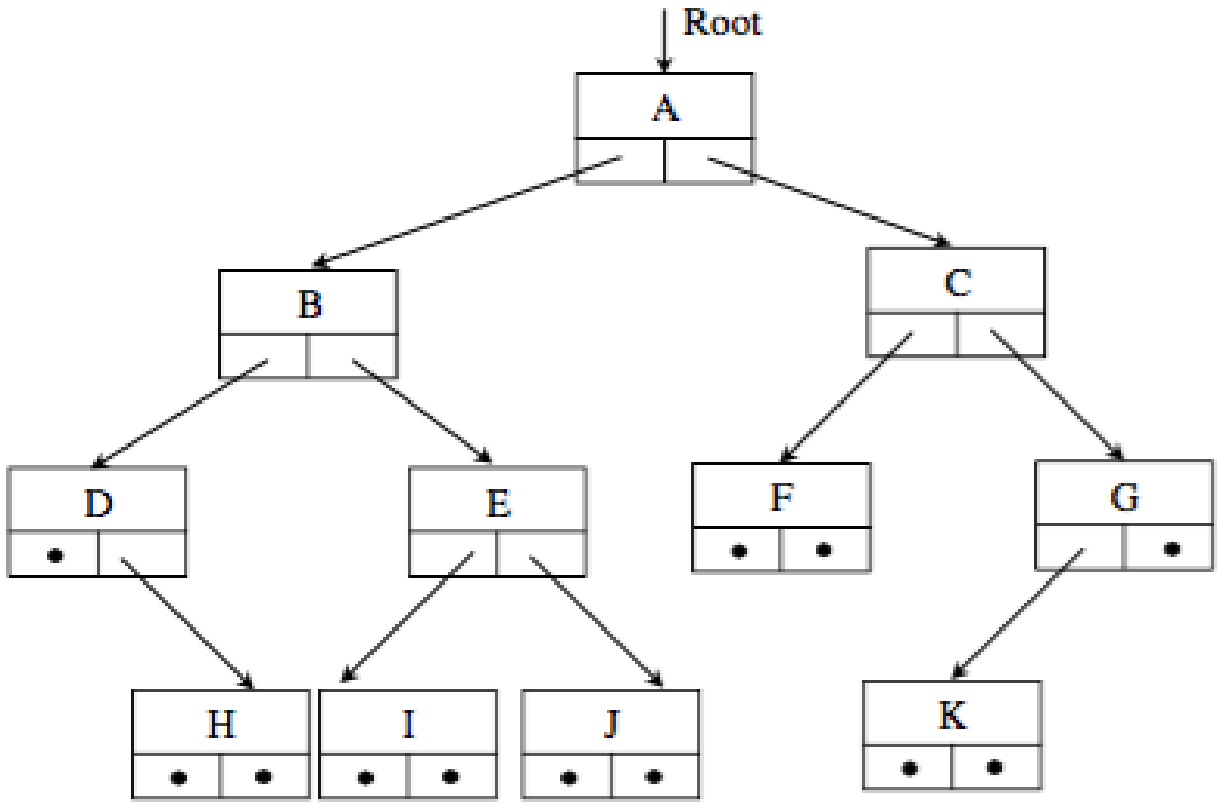
```
void printBFT(BFT *root)
{
   if(root != nullptr)
   {
      cout<<root->data.name<< " " <<root->data.yearOfBirth << endl;
      printBFT(root->left);
      printBFT(root->right);
   }
}
```

Duyệt theo thứ tự trước - Lặp

Input: Cây nhị phân cần duyệt có nút gốc là root
Output: Thứ tự duyệt các nút của cây theo thứ tự trước
Sử dụng 1 ngăn xếp s để lưu các nút trung gian trong quá trình duyệt.
Action:
- Khởi tạo ngăn xếp s rỗng
- Xuất phát từ nút gốc của cây
- Lặp khi nút đang xét khác rỗng
    + Thăm nút đang xét
    + Nếu nút con phải nút đang xét khác rỗng thì đưa nút con phải vào ngăn xếp s
    + Chuyển sang nút con bên trái
    + Nếu nút đang xét rỗng và ngăn xếp khác rỗng thì lấy 1 nút từ ngăn xếp

Minh họa

Nút đang xét	Ngăn xếp	Nút được thăm
A	C	A
B	C E	B
D	C E H	D
H	C E	H
E	C	E
I	C J	I
J	C	J
C	G	C
F	G	F
G		G
K		K
NULL		



```
void preOrderLoop(BinaryTreeNode *root)
{
   stack<BinaryTreeNode*> s;
   BinaryTreeNode *r;
   r = root;
   while (r != nullptr)
   {
      visit(r);
      if (r->right != nullptr)
      s.push(r->right);
      r = r->left;
      if(r == nullptr && !s.empty())
      {
      r = s.top(); s.pop();
      }
   }
}
```

Duyệt theo thứ tự giữa

- Thuật toán:

Input: Cây nhị phân cần duyệt có nút gốc là root

Output: Thứ tự duyệt các nút của cây theo thứ tự giữa

Action :

Nếu cây khác rỗng thì:

+ Duyệt theo thứ tự giữa cây con trái

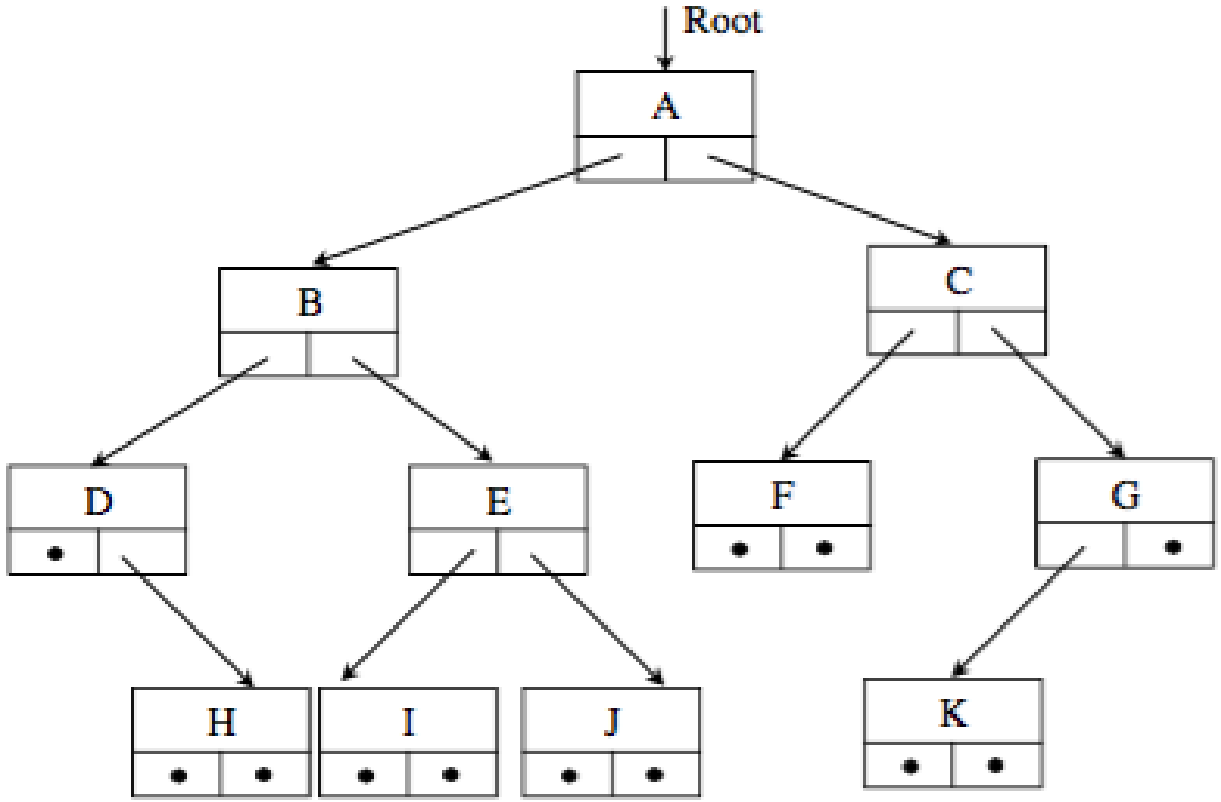
+ Thăm nút gốc

+ Duyệt theo thứ tự giữa cây con phải

- Cài đặt: tương tự.

Ví dụ

- Duyệt theo thứ tự giữa:
- D, H, B, I, E, J, A, F, C, K, G



Duyệt theo thứ tự sau

- Thuật toán:

Input: Cây nhị phân cần duyệt có nút gốc là root

Output: Thứ tự duyệt các nút của cây theo thứ tự sau

Action :

Nếu cây khác rỗng thì:

+ Duyệt theo thứ tự sau cây con trái

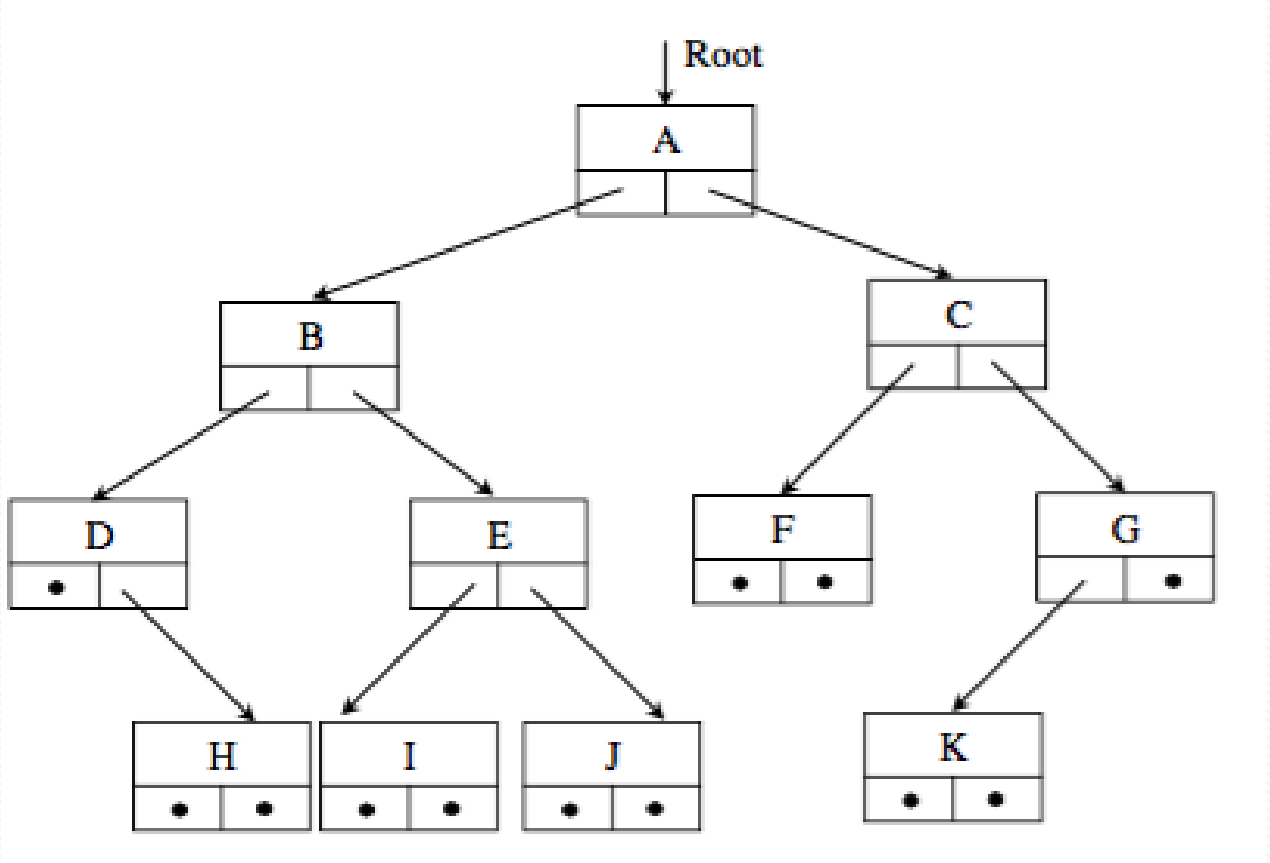
+ Duyệt theo thứ tự sau cây con phải

+ Thăm nút gốc

- Cài đặt: tương tự.

Ví dụ

- Duyệt theo thứ tự sau:
- H, D, I, J, E, B, F, K, G, C, A

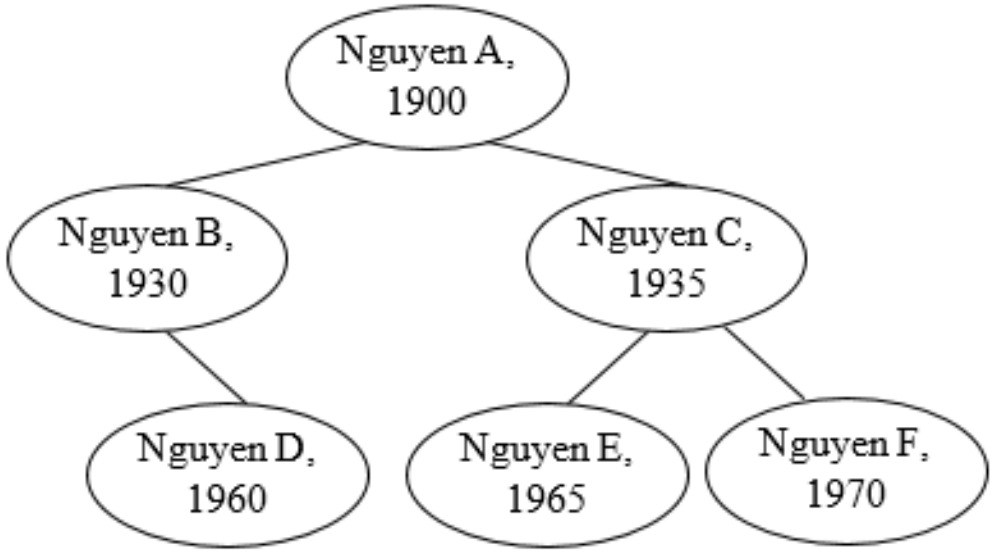


3. Bài tập

Bài 5.1 Trình bày thuật toán và cài đặt các thao tác trên cây gia phả nhị phân:

a) Đếm số người có trong cây
b) Tính số thế hệ của cây
c) Tìm một người có họ tên x trong cây
d) Cho biết người tên y có phải con của người tên x trong cây không?
e) Đếm số người sinh trước năm x
f) Kiểm tra người tên y có phải là con cháu của người tên x không?
g) Cho biết người tên x thuộc thế hệ thứ mấy trong cây gia phả?

```
struct Person
{
   string name;
   int yearOfBirth;
};
struct BFT  //Binary Family Tree
{
   Person data;
   BFT *left, *right;
};
```



Bài 5.1a

- Thuật toán:
- Input: cây gia phả cần đếm
- Output: số người trong cây gia phả
- Action:
- Nếu cây rỗng thì số người là 0
- Ngược lại:
  - Đếm số người ở cây con trái lưu vào biến d1
  - Đếm số người ở cây con phải lưu vào biến d2
  - Số người của cây là d1+d2+1

- Cài đặt:

```
int count(BFT *root)
{
  if(root == nullptr) return 0;
  else
  {
    int d1, d2;
    d1 = count(root->left);
    d2 = count(root->right);
    return d1+d2+1;
  }
}
```

Bài 5.1b

- Thuật toán:
- Input: cây gia phả cần tính số thế hệ
- Output: số thế hệ của cây gia phả
- Action:
- Nếu cây rỗng thì số thế hệ là 0
- Ngược lại:
  - Tính số thế hệ của cây con trái lưu vào biến h1
  - Tính số thế hệ của cây con phải lưu vào biến h2
  - Số thế hệ của cây là max(h1, h2)+1

Bài 5.1b

- Cài đặt: tự làm

Bài 5.1c

- Thuật toán:
- Input: Cây gia phả nhị phân, họ tên người cần tìm x
- Output: Nút chứa người họ tên x hoặc rỗng
- Action:
- Nếu cây rỗng thì không tìm thấy
- Ngược lại:
  - Nếu nút gốc chứa người có họ tên x thì trả về nút gốc
  - Ngược lại:
    - Tìm người tên x ở cây con trái
    - Nếu tìm được thì kết quả tìm thấy
    - Ngược lại thì tìm người tên x ở cây con phải

```
BFT* findPerson(BFT* root, string name)
{
    if (!root) return nullptr;
    if (root->data.name == name) return root;
    BFT* left = findPerson(root->left, name);
    if (left) return left;
    return findPerson(root->right, name);
}
```

Bài 5.1d

- Thuật toán:
- Input: cây gia phả nhị phân, họ tên x, y
- Output: true nếu y con x, false nếu ngược lại
- Action:
- Tìm nút p chứa người có họ tên x
- Nếu không tìm thấy thì trả về false
- Ngược lại:
  - Nếu p có nút con chứa người tên y thì trả về true
  - Ngược lại trả về false

```
bool isParent(BFT* root, string pName, string cName)
{
    BFT* parent;
    parent = findPerson(root, pName);
    if (!parent)
      return false;
    else
      return (parent->left && parent->left->data.name==cName) ||
            (parent->right && parent->right->data.name==cName);
}
```

Bài 5.1e

- e) Đếm số người sinh trước năm x

Bài 5.1f

- f) Kiểm tra người tên y có phải là con cháu của người tên x không?

Bài 5.1g

- g) Cho biết người tên x thuộc thế hệ thứ mấy trong cây gia phả?

Bài tập

Bài 5.2. Cho cây nhị phân mà mỗi nút chứa một số nguyên. Hãy trình bày thuật toán và cài đặt hàm kiểm tra hai cây có giống nhau không?

Bài 5.3.

a) Cho biết kết quả duyệt theo thứ tự trước của một cây nhị phân là: A D F G H K L P Q R W Z, và kết quả duyệt theo thứ tự giữa là: G F H K D L A W R Q P Z. Hãy xây dựng cây nhị phân. Tổng quát: hãy đưa ra thuật toán xây dựng cây từ kết quả duyệt theo thứ tự trước và thứ tự giữa.

b) Hãychỉ ra bằng một ví dụ rằng nếu biết kết quả duyệt theo thứ tự trước và thứ tự sau không xác định một cách duy nhất cây nhị phân. (Hãy chỉ một ví dụ của hai cây nhị phân khác nhau mà kết quả duyệt theo thứ tự trước và thứ tự sau trùng nhau).

Tổng kết

- Cây là cấu trúc phân cấp
- Duyệt là thao tác cơ bản trên cây nhị phân
- Cây nhị phân là cấu trúc cơ bản của cây nói chung
- Nhiều cấu trúc cây mở rộng từ cây nhị phân