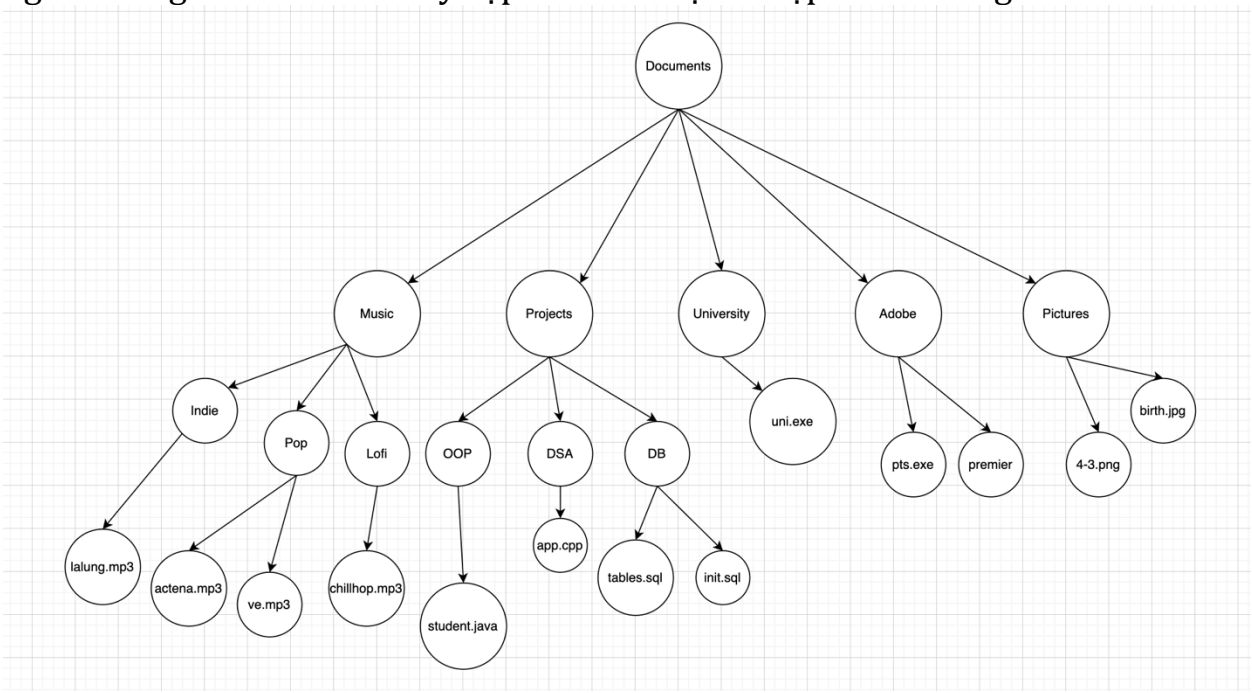
CHUẨN BỊ THỰC HÀNH
Chương 3

Họ và tên: Nguyễn Anh Tuấn
MSSV: 20200400
Lớp: 2

Bài làm:

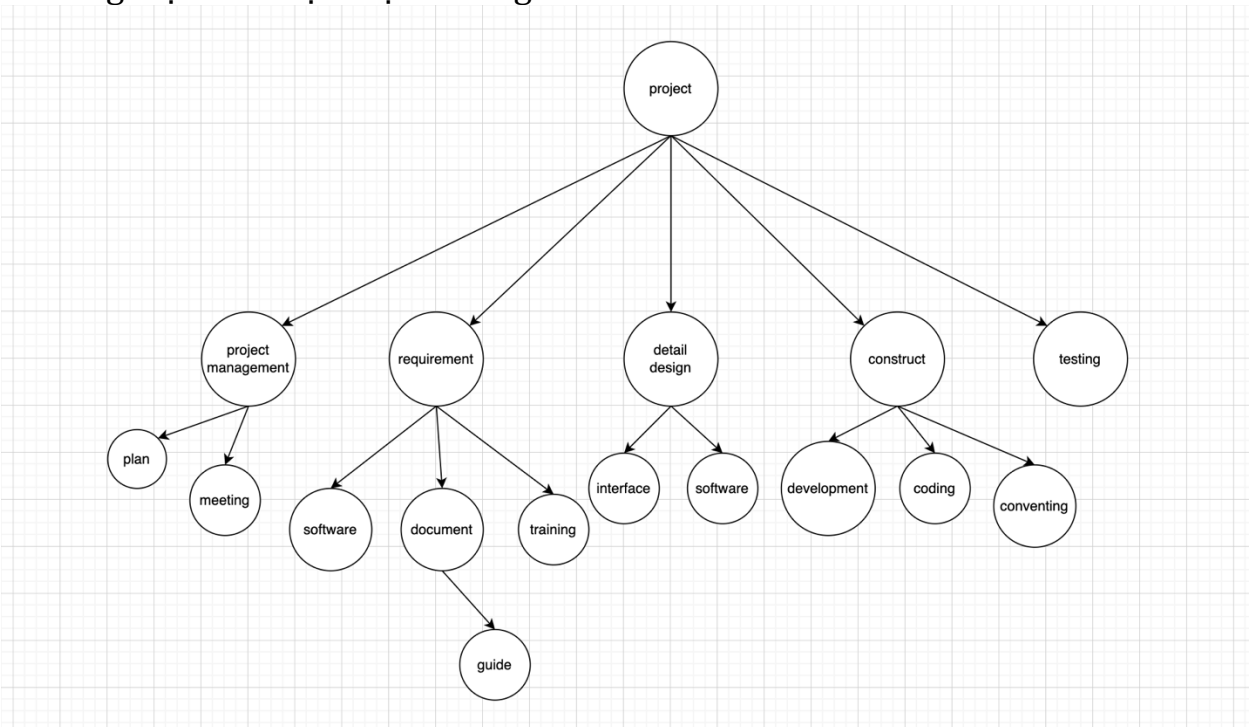
1. Cho một số bài toán dùng cấu trúc cây trong thực tế:

- Hệ thống quản lý thư mục và tập tin: Trong hệ thống này, cây được sử dụng để hiển thị và tổ chức các thư mục và tập tin trên máy tính. Mỗi thư mục và tập tin được đại diện bởi một nút trong cây, và các nút được xếp chồng lên nhau để tạo thành các cấp độ khác nhau. Cây cũng cho phép người dùng tìm kiếm và truy cập các thư mục và tập tin dễ dàng hơn.



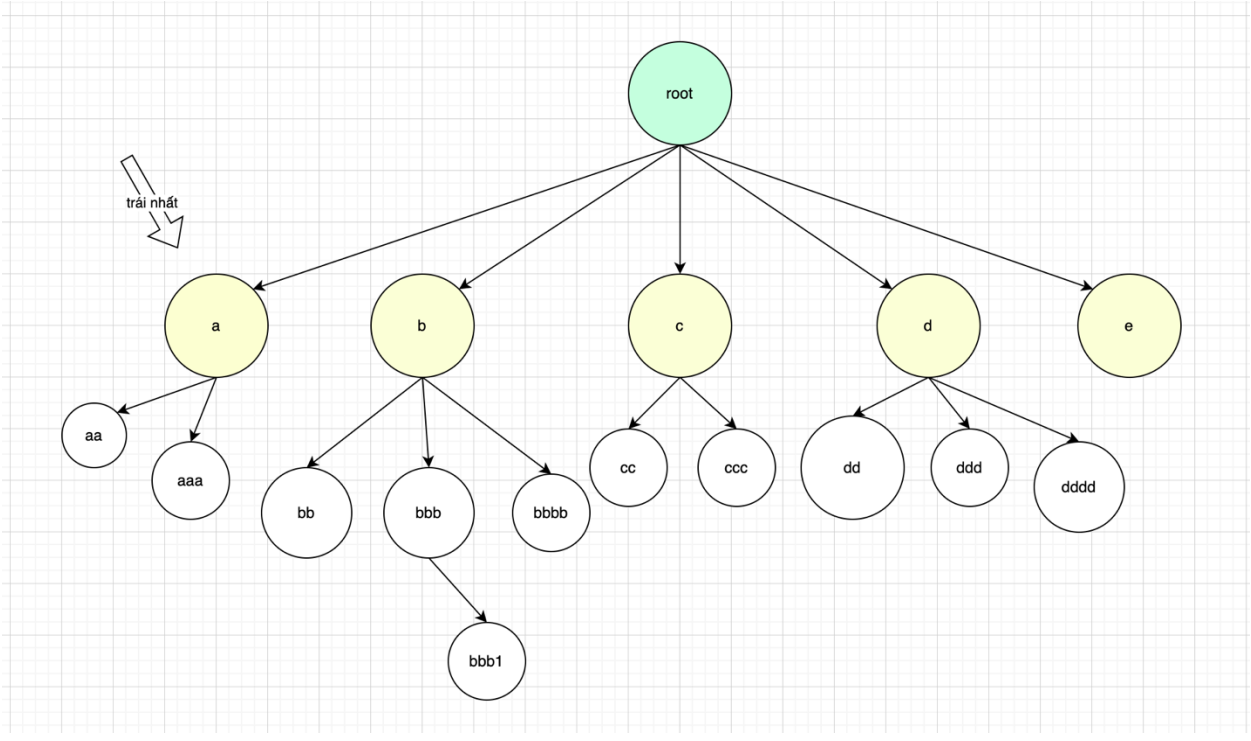
- Hệ thống quản lý dự án: Trong hệ thống này, cây được sử dụng để tổ chức và hiển thị các công việc và nhiệm vụ trong dự án theo cấp độ và mức độ liên quan. Mỗi công việc hoặc nhiệm vụ được đại diện bởi một

nút trong cây, và các nút được xếp chồng lên nhau để tạo thành các cấp độ khác nhau. Cây cũng cho phép người dùng tìm kiếm và truy cập các công việc và nhiệm vụ dễ dàng hơn.

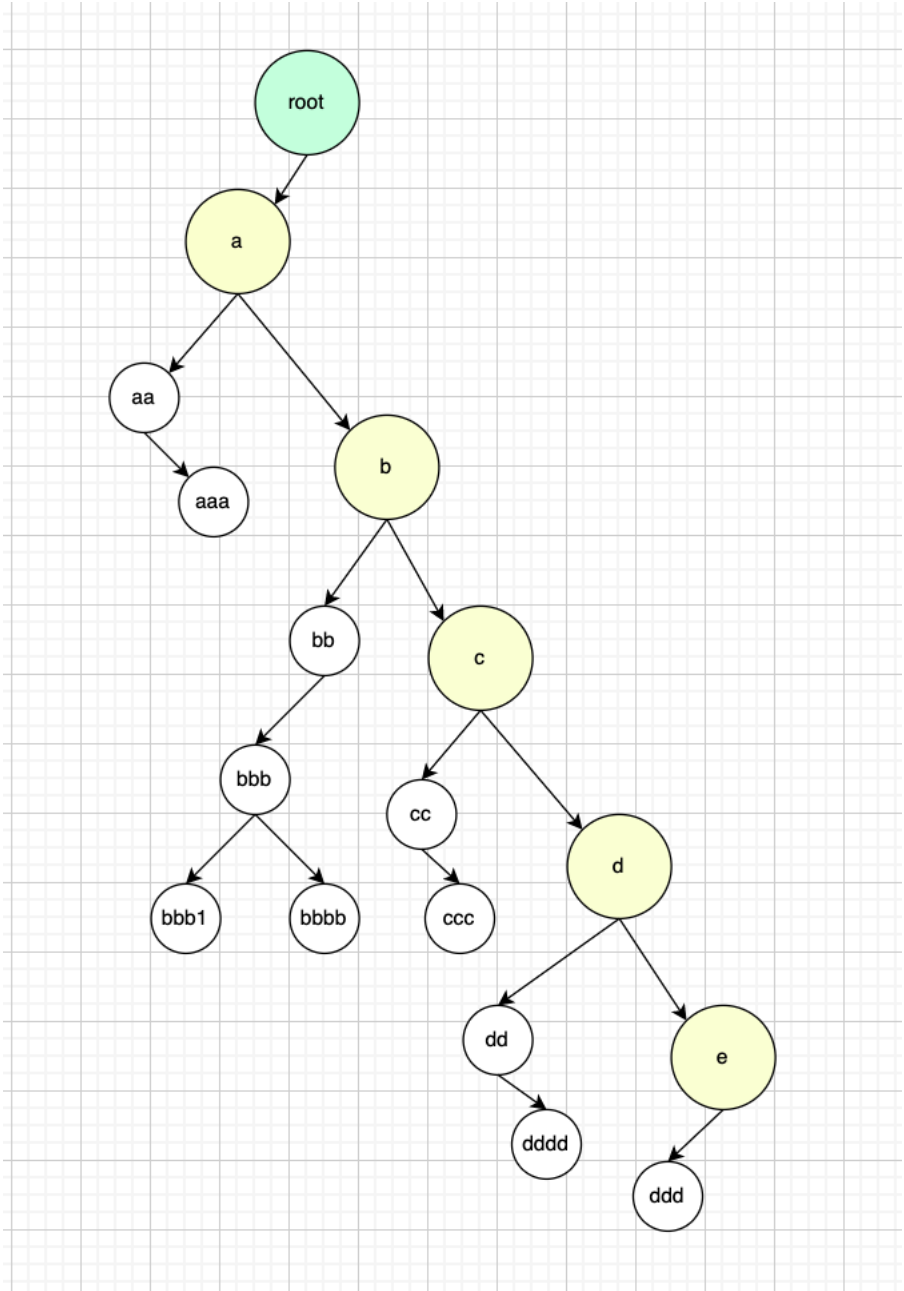


2. Nêu cách chuyển đổi một cây tổng quát thành cây nhị phân:

- Ban đầu ta cần xác định rõ hình thức cây nhị phân từ cây tổng quát với các con trái phải từ các con của cây tổng quát với đặc trưng:
    - root của cây tổng quát giữ nguyên
    - bắt đầu từ node root có các con, các con đồng cấp (gọi là anh em) thì sẽ được assign vào left tính từ con cây tổng quát trái nhất, ví dụ:



- các cháu tính từ node root hoặc sâu hơn sẽ là node bên trái tính từ node cha là node con (màu vàng) của node root (màu xanh), ta được cây nhị phân hoàn chỉnh từ cây tổng quát:



3. Trình bày code tạo cây và duyệt cây theo 3 cách PostOrder, InOrder, PreOrder:

Khởi tạo cây:

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

struct Tree {
    Node* root;
};
```

Tạo node từ giá trị đi kèm :

```
Node* getNode(int value){
    Node* node = new Node;
    node->data = value;
    node->left = NULL;
    node->right = NULL;
    return node;
}
```

Hàm tạo cây từ n phần tử nhập vào:

```
void createTree(Tree* tree, int n){
    cout << "Nhap gia tri tung phan tu: \t";
    for ( int i = 0 ; i < n ; i++){
        int tmp;
        cin >> tmp;
        addNode(*tree, tmp);
    }
}
```

Hàm thêm node vào cây sử dụng hàng đợi để theo dõi thứ tự node tthêm vào từ trên xuống dưới từ trái sang phải:

```
void addNode(Tree& tree, int value) {
    Node* newNode = getNode(value);
    if (tree.root == NULL) {
        tree.root = newNode;
        return;
    }
    queue<Node*> q;
    q.push(tree.root);
    while (!q.empty()) {
        Node* tmp = q.front();
        q.pop();
        if (tmp->left == NULL) {
            tmp->left = newNode;
            return;
        } else {
            q.push(tmp->left);
        }
        if (tmp->right == NULL) {
            tmp->right = newNode;
            return;
        } else {
            q.push(tmp->right);
        }
    }
}
```

Hàm khởi tạo cây:

```
void init(Node* root){
    root = NULL;
}
```

Các hàm lần lượt : preorder, inorder, postorder

```
void preOrder(Node* root){
    if ( root == NULL ) return;
    cout << root->data << " ";
    preOrder(root->left);
    preOrder(root->right);
}

// Traverse from left bottom side to parer
void inOrder(Node* root){
    if ( root == NULL ) return;
    inOrder(root->left);
    cout << root->data << " ";
    inOrder(root->right);
}

// Traverse from bottom left level to bott
void postOrder(Node* root){
    if ( root == NULL ) return;
    postOrder(root->left);
    postOrder(root->right);
    cout << root->data << " ";
}
```

Hàm main:

```cpp
int main() {
    Tree* tree = new Tree;
    init(tree->root);

    int n;
    cout << "Nhap so luong phan tu:";
    cin >> n;
    createTree(tree, n);

    cout << "\nPreOrder: \n";
    preOrder(tree->root);
    cout << "\nInOrder: \n";
    inOrder(tree->root);
    cout << "\nPostOrder: \n";
    postOrder(tree->root);

    delete tree;
}
```

Kết quả:

```
(base) macad@nganhhtuann-3 output % ./"binaryTree"
Nhap so luong phan tu:
7
Nhap gia tri tung phan tu:      4
2
5
3
7
4
2

PreOrder:
4 2 3 7 5 4 2
InOrder:
3 2 7 4 4 5 2
PostOrder:
3 7 2 4 2 5 4 %
```