Tree data structure

Hoàng-Nguyên Vũ

1 Lý thuyết về cấu trúc dữ liệu cây

1.1 Cấu trúc dữ liệu cây là gì?

Cấu trúc dữ liệu **cây (Tree)** là một tập hợp các nút (*nodes*) có mối quan hệ cha - con (*parent-child*). Đây là một dạng dữ liệu phi tuyến tính, được sử dụng để biểu diễn quan hệ phân cấp giữa các phần tử.

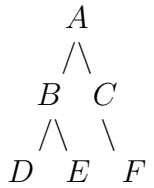
Một số ứng dụng thực tế của cây:

- **Hệ thống tệp tin** trong máy tính (cây thư mục).
- **Cây quyết định (Decision Tree)** trong học máy.
- **Cấu trúc tổ chức doanh nghiệp** (sơ đồ tổ chức).
- **Cây biểu thức (Expression Tree)** trong xử lý toán học.
- **Tìm kiếm và sắp xếp** (cây nhị phân tìm kiếm, cây AVL, cây đỏ-đen).

1.2 Các thành phần chính của cây

- **Nút (Node)**: Phần tử cơ bản của cây.
- **Nút gốc (Root)**: Nút đầu tiên trong cây, không có nút cha.
- **Nút con (Child)**: Các nút có liên kết trực tiếp từ một nút cha.
- **Nút cha (Parent)**: Một nút có các nút con.
- **Nút lá (Leaf)**: Nút không có nút con nào.
- **Cấp (Level)**: Độ sâu của một nút tính từ gốc.
- **Chiều cao của cây (Height)**: Số mức tối đa từ gốc đến nút xa nhất.

Ví dụ về một cây đơn giản:



2 Các ví dụ về cấu trúc cây và code mẫu

2.1 Cài đặt lớp TreeNode

Lớp `TreeNode` đại diện cho một nút trong cây:

```
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []

    def add_child(self, child):
        self.children.append(child)

    def remove_child(self, child):
        self.children = [c for c in self.children if c != child]

    def print_tree(self, level=0):
        print(" " * (level * 4) + "|-- " + self.value)
        for child in self.children:
            child.print_tree(level + 1)
```

2.2 Cài đặt lớp Tree

Lớp `Tree` giúp quản lý toàn bộ cây:

```
class Tree:
    def __init__(self, root_value):
        self.root = TreeNode(root_value)

    def find(self, value, node=None):
        if node is None:
            node = self.root
        if node.value == value:
            return node
        for child in node.children:
            found = self.find(value, child)
            if found:
                return found
        return None

    def insert(self, parent_value, child_value):
        parent_node = self.find(parent_value)
        if parent_node:
            parent_node.add_child(TreeNode(child_value))
        else:
            print(f"Nút cha '{parent_value}' không tồn tại.")

    def print_tree(self):
        self.root.print_tree()
```

3 Duyệt cây theo BFS (Breadth-First Search)

3.1 BFS là gì?

BFS (Breadth-First Search) hay **Tìm kiếm theo chiều rộng** là một thuật toán duyệt cây bằng cách kiểm tra tất cả các nút ở mức hiện tại trước khi chuyển sang mức tiếp theo.

Ví dụ duyệt BFS trên cây:

$$A \to B \to C \to D \to E \to F$$

3.2 Cài đặt thuật toán BFS

```python
from collections import deque

class Tree:
    def __init__(self, root_value):
        self.root = TreeNode(root_value)

    def traverse_bfs(self):
        if not self.root:
            return

        queue = deque([self.root])
        while queue:
            node = queue.popleft()
            print(node.value, end=" ")
            for child in node.children:
                queue.append(child)
        print()
```

3.3 Ví dụ sử dụng BFS

```python
# Khởi tạo cây
tree = Tree("A")

# Thêm các nút con vào cây
tree.root.add_child(TreeNode("B"))
tree.root.add_child(TreeNode("C"))

tree.root.children[0].add_child(TreeNode("D"))
tree.root.children[0].add_child(TreeNode("E"))
tree.root.children[1].add_child(TreeNode("F"))

# In cây theo dạng phân cấp
print("Cây ban đầu:")
tree.root.print_tree()

# Duyệt cây theo BFS
print("\nDuyệt cây theo BFS:")
tree.traverse_bfs()
```

4 Bài tập

4.1 Bài 1: Cài đặt và kiểm thử

- Viết code để tạo một cây có gốc `"Company"`.
- Thêm các phòng ban `"HR"`, `"IT"`, `"Finance"`.
- Thêm nhân viên `"Alice"`, `"Bob"` vào `"HR"`.
- Thêm nhân viên `"Charlie"`, `"David"` vào `"IT"`.
- In toàn bộ cây.

4.2 Bài 2: Cài đặt duyệt BFS

Viết phương thức `traverse_bfs()` trong lớp `Tree` và kiểm tra kết quả.

4.3 Bài 3: Tìm kiếm bằng BFS

Viết phương thức `search_bfs(value)` để kiểm tra xem giá trị `value` có tồn tại trong cây hay không.

4.4 Bài 4: Tính chiều cao của cây

Viết phương thức `tree_height()` để tính chiều cao của cây.