ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐIỆN – ĐIỆN TỬ

BÁO CÁO BÀI TẬP LỚN

Môn: Cấu trúc dữ liệu và giải thuật

Sinh viên thực hiện

Tên	MSSV	Lớp
Đặng Công Hoàng Nam	20224447	ET – E9 01 K67

Giáo viên hướng dẫn: Tạ Thị Kim Huệ

Hà Nội, tháng 1 năm 2025

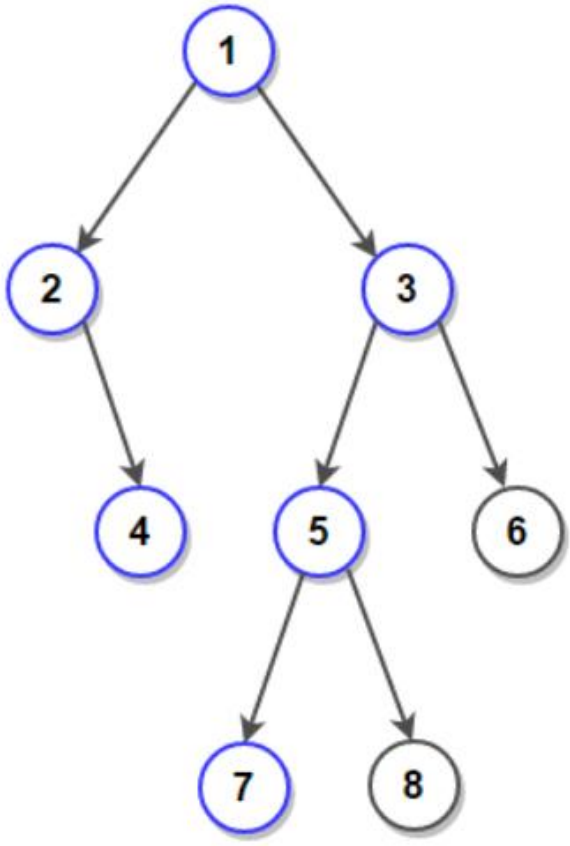
Đề tài: Find the diameter of a binary tree
Tìm đường kính của cây nhị phân

1. Giới thiệu

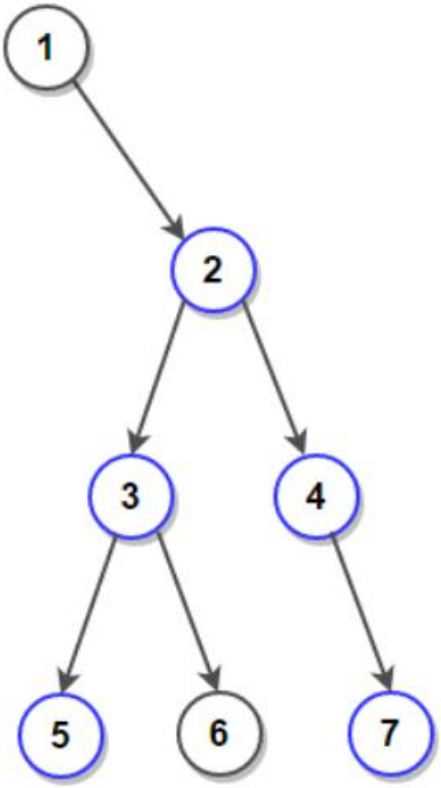
Đề bài toán:

Tính đường kính (diameter) của cây nhị phân. Đường kính của cây nhị phân là số nút lớn nhất trong đường đi dài nhất giữa hai nút bất kỳ trong cây. Đường đi này có thể:

- **Đi qua gốc** (root):



- **Không đi qua gốc**, nằm trong một cây con:



2. Phân tích sơ bộ

2.1. Đề xuất giải pháp:

Để giải quyết bài toán, ta cần:

- Tính chiều cao của cây con trái và cây con phải tại mỗi nút.
- Tính đường kính qua nút đang xét:
- Bằng tổng chiều cao của cây con trái và cây con phải, cộng thêm 1 (nút hiện tại).

    ⇨ Cập nhật đường kính lớn nhất trong quá trình duyệt hậu tự.
- Trả về chiều cao cho nút cha để tiếp tục tính toán.

2.2. Trường hợp

Đi qua gốc: Đường kính là chiều cao của cây con trái + chiều cao của cây con phải + 1.

Không đi qua gốc: Đường kính nằm trong cây con trái hoặc cây con phải.

3. Phân tích hai cách tiếp cận: $O(n^2)$ và $O(n)$

3.1. Giải pháp $O(n^2)$

Ý tưởng:

- Để tính đường kính tại một nút:
  - Tính chiều cao của cây con trái.
  - Tính chiều cao của cây con phải.
  - Tổng chiều cao của cây con trái và cây con phải (cộng thêm 1 cho nút hiện tại) sẽ là đường kính qua nút đó.
- Lặp lại việc tính đường kính tại tất cả các nút, và lấy giá trị lớn nhất.

Mã giả:

```
function height(node):
    if node == null:
        return 0
    return max(height(node.left), height(node.right)) + 1

function diameter(node):
    if node == null:
        return 0
    left_height = height(node.left)
    right_height = height(node.right)
    max_diameter = left_height + right_height + 1
    return max(max_diameter, diameter(node.left), diameter(node.right))
```

Độ phức tạp thời gian:

- Đối với mỗi nút, ta phải tính chiều cao của cây con trái và phải.
- Tính chiều cao mất O(n) thời gian, và thực hiện việc này cho mỗi nút, dẫn đến độ phức tạp $O(n^2)$.

Nhược điểm:

- Không hiệu quả với cây có số lượng nút lớn vì phải tính chiều cao của các cây con lặp đi lặp lại nhiều lần.

3.2. Giải pháp O(n)

Ý tưởng:

- Thay vì tính chiều cao của cây con trái và phải nhiều lần, ta sử dụng duyệt hậu tự để tính chiều cao trong cùng một lần duyệt cây.
- Trong quá trình duyệt:
  - Tính chiều cao của cây con trái.
  - Tính chiều cao của cây con phải.
  - Cập nhật đường kính bằng tổng chiều cao của hai cây con, cộng thêm 1.
- Chiều cao của cây con tại nút hiện tại sẽ được trả về ngay lập tức để tính toán cho các nút cha.

Mã giả:

```
function calculateDiameter(node, diameter):
    if node == null:
        return 0
    left_height = calculateDiameter(node.left, diameter)
    right_height = calculateDiameter(node.right, diameter)
    max_diameter = left_height + right_height + 1
    diameter = max(diameter, max_diameter)
    return max(left_height, right_height) + 1

function findDiameter(root):
    diameter = 0
    calculateDiameter(root, diameter)
    return diameter
```

Độ phức tạp thời gian:

- Mỗi nút trong cây được duyệt đúng một lần.
- Độ phức tạp là O(n), với n là số lượng nút trong cây.

Ưu điểm:

- Hiệu quả và tối ưu về thời gian.
- Không cần tính toán lặp đi lặp lại chiều cao của cây con như trong giải pháp $O(n^2)$.

3.3. So sánh $O(n^2)$ và $O(n)$

Tiêu chí	Giải pháp $O(n^2)$	Giải pháp $O(n)$
Ý tưởng	Tính chiều cao cây con lặp lại nhiều lần.	Tính chiều cao trong một lần duyệt.
Độ phức tạp thời gian	$O(n^2)$	$O(n)$
Số lần duyệt cây	Nhiều lần (lặp tại từng nút).	Một lần duy nhất.
Nhược điểm	Không hiệu quả với cây lớn.	Yêu cầu tư duy tối ưu.
Ưu điểm	Dễ triển khai.	Hiệu quả và tối ưu hơn.

4. Phân tích mã nguồn

4.1. Hàm calculateDiameter

Hàm này là phần cốt lõi của chương trình, được sử dụng đệ quy để duyệt qua từng nút trong cây và tính toán các giá trị cần thiết.

Chức năng chính:

1. Tính chiều cao của cây con trái và cây con phải tại mỗi nút.
2. Cập nhật đường kính lớn nhất nếu đường kính qua nút hiện tại lớn hơn giá trị hiện tại.
3. Lưu lại đường đi dài nhất qua các nút.
4. Trả về chiều cao của cây con để phục vụ tính toán cho nút cha.

Chi tiết các bước:

- **Kiểm tra nếu cây rỗng**:
    - Trả về chiều cao = 0, đường đi tạm thời (được lưu trong tempPath) là rỗng.
- **Tính chiều cao của cây con**:
    - Dùng đệ quy gọi lại calculateDiameter cho cây con trái và cây con phải.
- **Tính đường kính qua nút hiện tại**:

- Công thức: maxDiameter = leftHeight + rightHeight + 1.
- Nếu maxDiameter lớn hơn diameter hiện tại:
    1. Cập nhật diameter.
    2. Lưu đường đi dài nhất gồm:
        - Các nút từ cây con trái.
        - Nút hiện tại.
        - Các nút từ cây con phải (theo thứ tự ngược).

- **Cập nhật đường đi tạm thời:**
    - Dùng để trả về cho nút cha.
    - Chọn cây con trái hoặc phải tùy thuộc vào chiều cao.

4.2. Hàm getDiameter

Hàm này là giao diện không đệ quy để gọi calculateDiameter.

Chức năng chính:

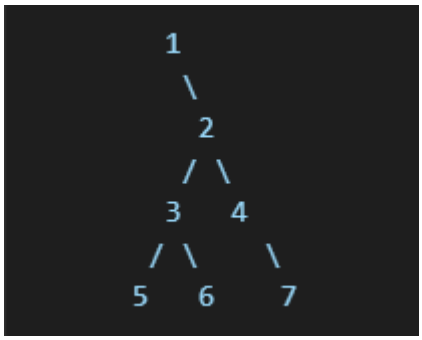
1. Khởi tạo các biến ban đầu như đường kính (diameter = 0), mảng đường đi (path[]), v.v.
2. Gọi hàm calculateDiameter để thực hiện tính toán.
3. Trả về giá trị đường kính lớn nhất.

4.3. Hàm main

- **Tạo cây nhị phân**:
    - Cây 1: Cây không đối xứng.
    - Cây 2: Cây cân bằng.
- **Gọi hàm **``:
    - Tính toán đường kính và lưu đường đi.
- **In kết quả**:
    - Hiển thị đường kính và các nút trong đường đi.

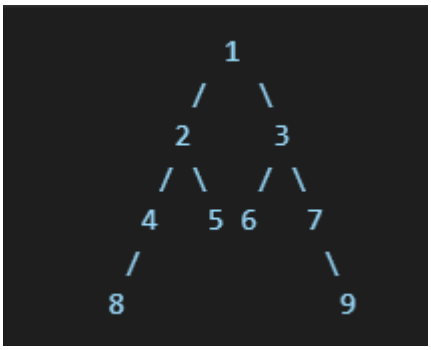
5. Quá trình xử lý từng cây

5.1. Cây 1



- Duyệt theo hậu tự:
  - Nút 5, 6, 7:
    - Chiều cao: 1 (không có con).
  - Nút 3:
    - Chiều cao trái = 1 (từ 5), chiều cao phải = 1 (từ 6).
    - Đường kính qua 3 = 1 + 1 + 1 = 3.
    - Đường đi: 5 -> 3 -> 6.
  - Nút 4:
    - Chiều cao trái = 0, chiều cao phải = 1 (từ 7).
    - Đường kính qua 4 = 0 + 1 + 1 = 2.
    - Đường đi: 4 -> 7.
  - Nút 2:
    - Chiều cao trái = 2 (từ 3), chiều cao phải = 2 (từ 4).
    - Đường kính qua 2 = 2 + 2 + 1 = 5.
    - Đường đi: 5 -> 3 -> 2 -> 4 -> 7.
  - Nút 1:
    - Chiều cao trái = 0, chiều cao phải = 3 (từ 2).
    - Đường kính qua 1 = 0 + 3 + 1 = 4.
    - Đường đi: 5 -> 3 -> 2 -> 4 -> 7 (không thay đổi).
- Kết quả:
  - Đường kính: 5.
  - Đường đi: 5 -> 3 -> 2 -> 4 -> 7.

5.2. Cây 2



- Duyệt theo hậu tự:
  - Nút 8, 5, 6, 9:
    - Chiều cao: 1 (không có con).
  - Nút 4:
    - Chiều cao trái = 1 (từ 8), chiều cao phải = 0.
    - Đường kính qua 4 = 1 + 0 + 1 = 2.
    - Đường đi: 8 -> 4.
  - Nút 7:
    - Chiều cao trái = 0, chiều cao phải = 1 (từ 9).
    - Đường kính qua 7 = 0 + 1 + 1 = 2.
    - Đường đi: 7 -> 9.
  - Nút 2:
    - Chiều cao trái = 2 (từ 4), chiều cao phải = 1 (từ 5).
    - Đường kính qua 2 = 2 + 1 + 1 = 4.
    - Đường đi: 8 -> 4 -> 2 -> 5.
  - Nút 3:
    - Chiều cao trái = 1 (từ 6), chiều cao phải = 2 (từ 7).
    - Đường kính qua 3 = 1 + 2 + 1 = 4.
    - Đường đi: 6 -> 3 -> 7 -> 9.
  - Nút 1:
    - Chiều cao trái = 3 (từ 2), chiều cao phải = 3 (từ 3).
    - Đường kính qua 1 = 3 + 3 + 1 = 7.
    - Đường đi: 8 -> 4 -> 2 -> 1 -> 3 -> 7 -> 9.
- Kết quả:
  - Đường kính: 7.
  - Đường đi: 8 -> 4 -> 2 -> 1 -> 3 -> 7 -> 9.

6. Tổng kết

Bài toán tính đường kính của cây nhị phân được giải quyết hiệu quả bằng cách sử dụng duyệt hậu tự. Hai giải pháp được đưa ra ($O(n^2)$ và $O(n)$) cho thấy sự khác biệt rõ ràng về hiệu năng, với cách $O(n)$ là lựa chọn tối ưu cho các cây lớn. Mã nguồn được triển khai trực quan và hiệu quả, có thể áp dụng cho nhiều ứng dụng thực tế như mạng lưới hoặc hệ thống phân cấp dữ liệu.

Tài liệu tham khảo

- Diameter of a Binary Tree - GeeksforGeeks
- Find the diameter of a binary tree | Techie Delight