ĐẠI HỌC QUỐC GIA TPHCM
Trường Đại học Công nghệ Thông tin
Khoa Khoa học Máy tính

BÀI TẬP PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN ĐỆ QUY

Thành viên 1: 21520117 - Phan Trường Trí
Thành viên 2: 21520373 - Nguyễn Đức Nhân
Lớp: CS112.N21.KHTN
Giảng viên: Nguyễn Thanh Sơn

Mục lục

1 Bài 1

1.1 Đề bài: Tower of Hanoi

a. In the original version of the Tower of Hanoi puzzle, as it was published in the 1890s by Édouard Lucas, French mathematician, the world will end after 64 disks have been moved from a mystical Tower of Brahma. Estimate the number of years it will take if monks could move one disk per minute. (Assume that monks do not eat, sleep, or die.)

b. How many moves are made by the i-th largest disk ($1 \leq i \leq n$) in this algorithm?

c. Find a non-recursive algorithm for the Tower of Hanoi puzzle and implement it in the language of your choice

1.2 Bài giải:

Định nghĩa bài toán: Cho 3 cọc và n đĩa tròn được đặt trên các cọc này. Trạng thái ban đầu của bài toán là tất cả các đĩa tròn được đặt trên cọc 1 và sắp xếp theo độ lớn tăng dần với đĩa có kích thước lớn nhất nằm ở dưới cùng.

Yêu cầu bài toán: Chuyển tất cả các đĩa từ cọc 1 sang cọc 3 với thứ tự tăng dần như trạng thái ban đầu.

a. Estimate the number of years it will take if monks could move one disk per minute.

Thuật toán: Để chuyển toàn bộ đĩa từ cọc 1 sang cọc 3 thì ta thực hiện chuỗi 3 bước sau:

1. Chuyển $n-1$ đĩa từ cọc 1 sang cọc 2.
2. Chuyển đĩa có kích thước lớn nhất từ cọc 1 sang cọc 3.

3. Chuyển $n-1$ đĩa từ cọc 2 sang cọc 3.

Gọi $F(n)$ là số bước để chuyển n đĩa từ cọc ban đầu sang 1 cọc khác những vẫn đảm bảo thứ tự tăng dần (đĩa lớn nhất sẽ ở dưới). Xét $n <= 3$ ta có:

- $n = 1 : F(n) = 1$
- $n = 2 : F(n) = 3$
- $n = 3 : F(n) = 7$

Nên ta có công thức: $F(n) = 2F(n-1) + 1$ với $n > 1, F(1) = 1$.
$\Leftrightarrow F(n) = 2(2F(n-2)+1)+1$
$= 2^2F(n-2) + 2 + 1$
$= 2^3F(n-3) + 2^2 + 2 + 1$
$= 2^iF(n-i) + 2^{i-1} + ... + 2 + 1$
$= 2^{n-1}F(n-(n-1)) + 2^{n-2} + ... + 2 + 1$
$= 2^{n-1} + 2^{n-2} + 2^{n-3} + ... + 2 + 1$
$= 2^n - 1$

Dựa vào công thức trên ta tính được số bước để di chuyển 64 đĩa từ cọc 1 sang cọc 3 là $2^{64} - 1$ phút tương đương 35096 tỷ năm.

b. How many moves are made by the i-th largest disk ($1 \leq i \leq n$) in this algorithm?

Gọi $M(n)$ là số bước mà đĩa lớn thứ n được di chuyển. Dễ thấy $M(n)$ sẽ bằng số lần mà ta chuyển hết toàn bộ n đĩa từ cọc này sang một cọc khác. Nên ta có:
$M(n-1) = 2M(n)$
$\Leftrightarrow M(i) = 2^{n-i}M(n)$

Vậy số bước mà đĩa có độ lớn thứ i phải di chuyển là 2^{n-i}

c. Find a non-recursive algorithm for the Tower of Hanoi puzzle and implement it in the language of your choice

2 Bài 2:

2.1 Đề bài:

Set up a recurrence relation, with an appropriate initial condition, for the number of times the basic operation is executed for Quicksort algorithm. And solve it for the best case, worst case and average case, then conclude the time complexity for each case

2.2 Bài giải:

- **Hệ thức truy hồi độ phức tạp:**

$$T(n) = \begin{cases} 0, & \text{if } x = [0:1]. \\ T(k) + T(n-k-1) + n - 1, & \text{otherwise.} \end{cases} \quad (1)$$

- **Độ phức tạp:**

1. **Worst-case:** Nhận thấy nếu pivot được chọn là phần tử bé nhất hay lớn nhất và mảng đã cho được sắp xếp thì ta có:
   $T(n) = T(n-1) + n - 1$
   $= T(n-2) + (n-2) + (n-1)$
   $= T(0) + 1 + 2 + ... + (n-2) + (n-1) = \frac{n(n-1)}{2}$
2. **Average-case:** $T(n) = 2n \ln n$
3. **Best-case:** Nhận thấy nếu pivot được chọn là phần tử trung vị của mảng thì ta có:
   $T(n) = 2T(\frac{n}{2}) + n - 1$
   $\approx n \log n$

3 Bài 3:

3.1 Đề bài: EXP

1. Design a recursive algorithm for computing 2^n for any non-negative integer n that is based on the formula $2^n = 2^{n-1} + 2^{n-1}$
2. Set up a recurrence relation for the number of additions made by the algorithm and solve it.
3. Draw a tree of recursive calls for this algorithm and count the number of calls made by the algorithm.
4. Is it a good algorithm for solving this problem?

3.2 Bài giải:

a. Design a recursive algorithm

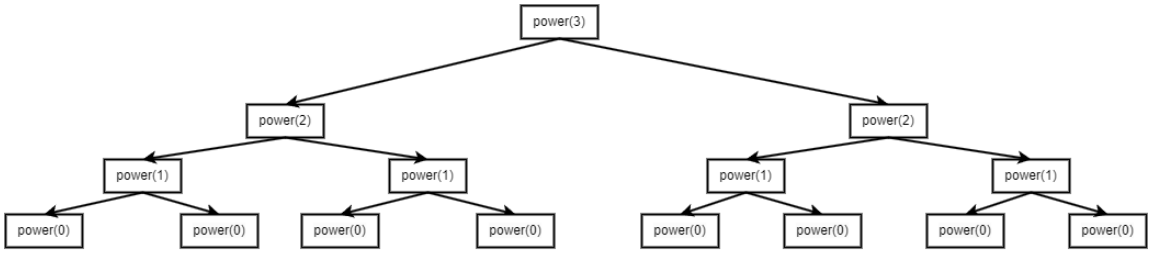
```
Power(n)
    if n = 0 then
        return 1
    end
    return Power(n − 1) + Power(n − 1)
```

b. Set up a recurrence relation for the number of additions made by the algorithm and solve it.

Gọi $T(n)$ là độ phức tạp để tính toán 2^n. Ta có:

$$\begin{aligned} T(n) &= T(n-1) + T(n-1) \\ &= 2T(n-1) \\ &= 2 * 2T(n-2) \\ &= 2^n \end{aligned}$$

c. Draw a tree of recursive calls for this algorithm and count the number of calls made by the algorithm.



Như hình ta có mỗi độ sâu của cây sẽ có 2^i node nên tổng số node trong cây được gọi bởi giải thuật là $\sum_{i=0}^{n-1} 2^i$

d. Is it a good algorithm for solving this problem?

Việc gọi lại hàm độ quy khi $n! = 0$ nhiều lần sẽ khiến thời gian chạy tăng cao. Giải pháp là trả về giá trị $2 * Power(n-1)$ thay vì $Power(n-1) + Power(n-1)$.