Phân tích độ phức tạp thuật toán đệ quy

Nguyễn Tuấn Anh, Nguyễn Trần Việt Anh*

Ngày 2 tháng 4 năm 2023

Bài viết này trình bày lời giải cho các bài tập phân tích độ phức tạp thuật toán đệ quy, do **nhóm 2** đề xuất.

Mục lục

*Sinh viên lớp KHTN2021, MSSV: 21520142 - 21520006

§1 Problem 1. Tower Of Hanoi, proposed by Team 2

§1.1 Bài toán

1. In the original version of the Tower of Hanoi puzzle, as it was published in the 1890s by Edouard 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.)
2. How many moves are made by the ith largest disk ($1 \leq i \leq$ n) in this algorithm?
3. Find a nonrecursive algorithm for the Tower of Hanoi puzzle and implement it in the language of your choice.

§1.2 Lời giải

1. Xét bài toán tháp Hà Nội với n chiếc đĩa. Gọi a_n là số lần chuyển n chiếc đĩa từ cọc 1 sang cọc 2. Để chuyển n chiếc đĩa từ cọc 1 sang cọc 2 ta phải thực hiện các công đoạn sau:

1. Chuyển $n-1$ chiếc đĩa bên trên chiếc đĩa lớn nhất sang cọc số 3 sang nguyên tắc trên. Cần thực hiện a_{n-1} lần chuyển như vậy.
2. Chuyển chiếc đĩa lớn nhất sang cọc số 2.
3. Chuyển $n-1$ chiếc đĩa từ cọc số 3 sang cọc số 2. Cần thực hiện a_{n-1} lần chuyển.

Ta thu được hệ thức $a_n = 2a_{n-1} + 1$. Với $a_1 = 1$ thì $a_n = 2^n - 1$.
Để di chuyển 64 đĩa từ cọc nguồn tới đích, phải mất $2^{64} - 1$ phút. Xấp xỉ 35000 tỷ năm.
Minh họa lời giải đệ quy

```
def Move(disks, source, intermediate, dest):
    if disks == 1:
        Move disk from source to destination
    else:
        Move(disk-1, source, destination, intermediate)
        move the remaining disk from source to destination
        Move(disk-1, intermediate, source, destination)
```

2. Nhận thấy, với mỗi lần đệ quy ta chỉ cần di chuyển đĩa lớn nhất đúng 1 lần. Từ thuật toán đệ quy phía trên, dễ dàng thấy số lần gọi hàm Move(n-1) đúng bằng 2 lần số lần gọi hàm Move(n). Vậy số lần di chuyển các đĩa từ lớn tới nhỏ lần lượt là $1,\ 2,\ 4, .., 2^{n-1}$

3. Có thể sử dụng thuật toán ngắn gọn sau:

```
def Move(n):
        stakes = ["source", "immediate", "dest"]
        if n%2==0:
                stakes[1], stakes[2] = stakes[2], stakes[1]
        for i in range(1,2**n):
                print(f"Move a disk from {stakes[(i & i - 1) % 3]}
                    to {stakes[((i | i - 1) + 1) % 3]}")
```

§2 Problem 2. Quick Sort, proposed by Team 2

§2.1 Bài toán

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 Lời giải

- Độ phức tạp:

$$T(0) = T(1) = 1, \quad T(n) = T(k) + T(n-k-1) + n - 1 \quad (\text{với } k = \overline{1, n-1}).$$

- Nhận xét: khi pivot là phần tử lớn nhất hoặc nhỏ nhất của mảng. Khi chia mảng thành 2 mảng con, 1 mảng sẽ không có phần tử nào, mảng còn lại có $n-1$ phần tử. Khi đó, $T(n) = T(n-1) + T(0) + n - 1 = T(n-1) + n$.
  Worst-case: Mảng đã được sắp xếp tăng hoặc giảm dần, khi đó:
  $T(k) = T(k-1) + k, \quad \forall k \in [1, n]$.
  Vì $T(n) - T(n-1) = n$, ta có:

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

- Average-case: $T(n) \leq 2\ln(n)(n+1)$. Phần chứng minh có thể tham khảo ở proof.

- Best-case: Pivot chính là trung vị của mảng, khi đó:

$$T(n) = T(n/2) + T(n/2) + n = 2T(n/2) + n$$
$$\Rightarrow T(n) \approx n\log(n)$$

§3 Problem 3. Exponentiation, proposed by Team 2

§3.1 Bài toán

1. Design a recursive algorithm for computing 2^n for any nonnegative 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 Lời giải

1. Thuật toán:

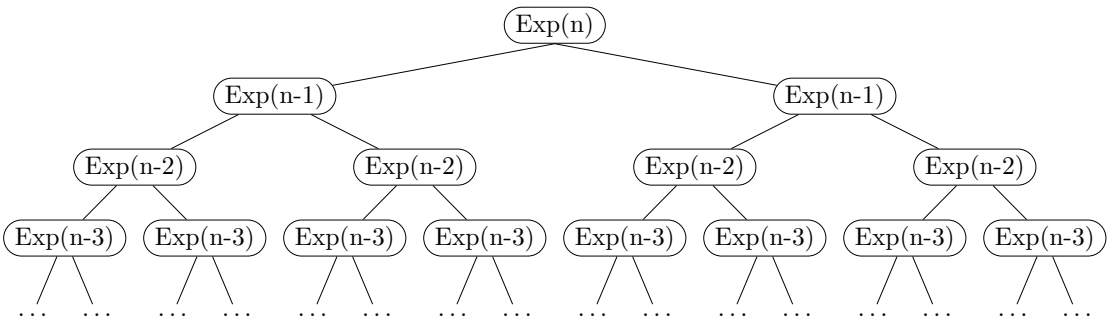
```
def exp(n):
    if (n==0):
        return 1

    return 2*exp(n-1)
```

2. Recurrence:

$$T(n) = 2T(n-1) = \cdots = 2^n T(0) = 2^n$$

3. Recursion Tree:



Số lần gọi đệ quy: $1 + 2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 1$.

4. Thuật toán trên không phải là thuật toán tốt. Bởi vì với mỗi $k \leq n$, Exp(k) được gọi 2^{n-k} lần. Do đó, độ phức tạp của thuật toán này rất lớn.