# Bài tập Phân tích độ phức tạp thuật toán đệ quy

Nhóm 4

CS112.N21.KHTN

**Tóm tắt nội dung**

Trình bày lời giải của nhóm 4 cho các bài tập nhóm 2 đưa ra trong chủ đề Phân tích độ phức tạp thuật toán đệ quy.

**Nhóm 4**

| Thành viên | MSSV |
|---|---|
| Võ Trần Thu Ngân | 21520069 |
| Tô Anh Phát | 21520085 |

Link repository GitHub: CS112.N21.KHTN-Group4

# Mục lục

# 1 Tower of Hanoi

## 1.1 Câu hỏi

**Tower of Hanoi**

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 $i$th 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. Để chuyển $n-1$ chiếc đĩa từ cọc 1 sang cọc 3, ta có giải thuật đệ quy như sau:

1. Chuyển $n-1$ chiếc đĩa bên trên từ cọc 1 sang cọc 2
2. Chuyển chiếc đĩa thứ $n$ (tức chiếc đĩa lớn nhất, nằm dưới cùng) từ cọc 1 sang cọc 3
3. Chuyển $n-1$ chiếc đĩa từ cọc 2 sang cọc 3

Gọi $M(n)$ là số bước chuyển đĩa cần thực hiện để chuyển hết $n$ đĩa từ cọc này sang cọc khác. Dựa vào thuật toán trên, ta có công thức: $M(n) = 2M(n-1) + 1$ với $n > 1$ và $M(1) = 1$. Ta khai triển công thức như sau:

$$\begin{aligned} M(n) =& 2M(n-1) + 1 \\ =& 2^2 M(n-2) + 2^1 + 1 \\ =& 2^3 M(n-3) + 2^2 + 2^1 + 1 \\ =& ... \end{aligned}$$

Tổng quát: $M(n) = 2^i M(n-i) + 2^{i-1} + 2^{i-2} + ... + 1 = 2^i M(n-i) + 2^i - 1$
Với $i = n-1$ và $M(1) = 1$, ta được hệ thức:

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

Vậy, số bước chuyển đĩa mà nhà sư phải thực hiện để chuyển tất cả 64 đĩa là $2^{64} - 1$. Nếu mỗi bước chuyển đĩa mất 1 phút thì tổng cộng nhà sư mất $2^{64} - 1$ phút, tức hơn 3500 tỷ năm.

**2.** Xét mã giả cho giải thuật đệ quy:

```
def Move(n, src, aux, des):
    if n == 1:
        Move disk from src to des
    else:
        Move(n - 1, src, des, aux)
        Move the final disk from src to des
        Move(n - 1, aux, src, des)
```

Theo giải thuật trên, đĩa lớn nhất sẽ chỉ di chuyển 1 lần, đĩa lớn thứ hai sẽ di chuyển 2 lần, đĩa lớn thứ ba sẽ di chuyển 4 lần,... Tổng quát, đĩa thứ $i-1$ sẽ có số lần di chuyển gấp đôi số lần di chuyển đĩa thứ $i$. Vậy số lần di chuyện của đĩa lớn thứ $i$ là $2^{i-1}$.

**3.** Ta có thể sử dụng cấu trúc dữ liệu `stack` để khử đệ quy như sau:

```
/// Move disks from src to dest
struct Move
{
    /// number of disks to move
    int disks;
    /// index of source, destination and auxiliary pegs
    int src, dest, aux;
};

void solve(int n)
{
    stack <Move> st;
    st.push({n, 1, 3, 2});
    while (sz(st))
    {
        auto m = st.top();
        st.pop();
        if (m.disks == 1)
            cout << "Move a disk from peg " << m.src << " to peg " << m.dest << '\n';
        else
        {
            st.push({m.disks - 1, m.aux, m.dest, m.src});
            st.push({1, m.src, m.dest, m.aux});
            st.push({m.disks - 1, m.src, m.aux, m.dest});
        }
    }
}
```

Chi tiết code có thể tham khảo tại repository của nhóm

# 2 Quicksort

## 2.1 Câu hỏi

**Quicksort**

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

- Ta có hệ thức độ phức tạp:

$$T(n) = \begin{cases} 1 & \text{với } n = 0 \text{ hoặc } n = 1 \\ T(n) = T(k) + T(n-k-1) + n - 1 & \text{với } n \geq 2 \text{ và } k = \overline{1, n-1} \end{cases} \quad (1)$$

- **Worst-case:** Nhận thấy rằng, nếu pivot được chọn là phần tử lớn nhất hoặc bé nhất thì khi chia mảng, một mảng sẽ có $n-1$ phần tử, mảng còn lại không có phần tử nào. Khi đó $T(n) = T(n-1) + T(0) + n - 1 = T(n-1) + n$. Vậy trường hợp xấu nhất của giải thuật xảy ra khi mảng đã được sắp xếp tăng dần hoặc giảm dần. Khi đó:

$$\begin{aligned} T(n) =& T(n-1) + n \\ =& n + T(n-2) + n - 1 \\ =& n + (n-1) + (n-2) + (n-3) + ... + 1 + 1 \\ =& \frac{n(n+1)}{2} + 1 \end{aligned}$$

- **Average-case:** $T(n) = 2ln(n)(n+1)$
- **Best-case:** Pivot chính là trung vị của mảng. Khi đó:

$$\begin{aligned} T(n) =& T(n/2) + T(n/2) + n \\ =& 2T(n/2) + n \\ \approx& nlog(n) \end{aligned}$$

# 3 EXP

## 3.1 Câu hỏi

**EXP**

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

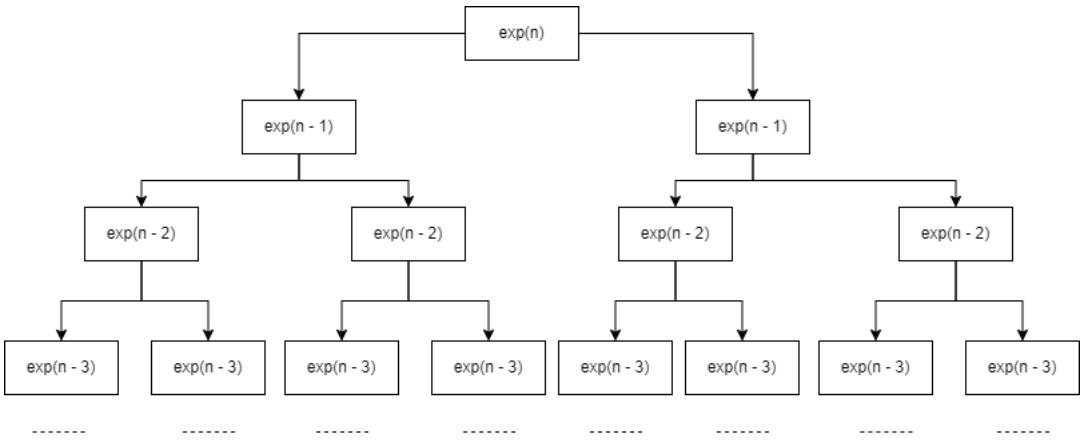
**1.** Thuật toán:

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

**2.** Ta có hệ thức:

$$\begin{aligned} T(n) =& 2T(n-1) = 2^2T(n-2) = 2^3T(n-3) \\ =& ... \\ =& 2^nT(0) = 2^n \end{aligned}$$

**3.** Recursive Tree:




**4.** Nhận thấy, số lần gọi đệ quy là $2^n - 1$. Vậy độ phức tạp tương ứng của giải thuật $O(2^n)$. Đây là một độ phức tạp lớn nên giải thuật này không phải một giải thuật tốt.