Lớp: CS112.P11.CTTN
Nhóm: 14
Sinh viên: Lê Nguyễn Anh Khoa. MSSV: 23520742
Sinh viên: Cáp Kim Hải Anh. MSSV: 23520036

BÀI TẬP
Phương pháp thiết kế thuật toán Divide, Decrease, Transform and Conquer

1. **Bài 1:Hãy đưa ra 3 bài toán có ứng dụng 3 kỹ thuật vừa được học: Divide and conquer, Decrease and conquer, Transform and conquer. Giải thích cụ thể áp dụng như thế nào. (Có thể các bài toán lập trình hoặc các ứng dụng thực tế)**

A. Divide and conquer

- Có thể áp dụng phương pháp trên vào bài toán sắp xếp, cụ thể thuật toán **merge sort**:

+ **Merge_sort(A, l, r)**: thực hiện sắp xếp mảng A với các vị trí từ l đến r.

+ mid = (l + r)/2. Thực hiện sắp xếp 2 phần nhỏ hơn là Merge_sort(A, l, mid) và Merge_sort(A, mid+1, r).

+ Thực hiện sắp xếp trên đoạn l đến r bằng cách gộp 2 phần dùng 2 con trỏ.

- Độ phức tạp: **O(n.log(n))** do duyệt qua mảng log(n) lần mà mỗi lần duyệt qua n phần tử.

B. Decrease and conquer

- Đề bài: **Tháp Hà Nội**

+ Có n đĩa và 3 cột A, B, C. Lúc đầu n đĩa này đang ở cột A. Cần chuyển tất cả các đĩa sang cột C

+ Nguyên tắc: Chỉ sử dụng 3 cọc để chuyển, một lần chỉ được di chuyển một đĩa nằm trên cùng từ cọc này sang cọc khác, một đĩa chỉ được đặt lên một đĩa lớn hơn.

- Lời giải:

+ Hướng giải dễ thấy của bài này là ta sẽ sử dụng đệ quy.

+ Đề bài yêu cầu chúng ta cần phải chuyển **n** đĩa cọc sang cọc **C**. Giả sử bắt đầu xét đĩa to nhất (đĩa thứ **n**), ta sẽ cần phải chuyển đĩa này sang cọc **C** trước tiên, có 2 trường hợp xảy ra:

- Đĩa đã nằm ở cọc C: ta sẽ không cần di chuyển đĩa này nữa và xét đến đĩa n - 1.
- Đĩa không nằm ở cọc C: ta cần dồn tất cả (n - 1) đĩa còn lại sang cọc trung gian (ở đây nếu đĩa n nằm ở cọc A thì đĩa trung gian ở cọc B và ngược lại) để có thể chuyển được đĩa n sang cọc C. Sau đấy thì bắt đầu bước dồn n - 1 đĩa còn lại từ cọc trung gian sang cọc C.
- Từ đó, khi xét đến đĩa thứ x, ta cần quan tâm 2 điều: vị trí hiện tại của đĩa x và cột mục tiêu ta cần chuyển đĩa x sang.

- Độ phức tạp: Với mỗi đĩa, trường hợp xấu nhất là ta cần chuyển x – 1 đĩa sang cọc trung gian và chuyển x – 1 đĩa sang cọc C. Vậy sẽ có 2 thao tác cần làm ở mỗi lần đệ quy nên độ phức tạp là **O(2^n).**
- Sol(x, row): xét đến dĩa thứ **x** và cần đặt vào cột **row.** Cần giải quyết bài toán nhỏ hơn là sol(x-1, row) và sol(x-1, 3-row-a[x]). 2 bài toán trên đã được giảm thiểu kích thước.

C. Transform and conquer

- Có thể áp dụng vào trong việc tìm kiếm trong mảng
- Sắp xếp mảng theo thứ tự và dùng tìm kiếm nhị phân trên mảng này.

⇨ Giảm độ phức tạp từ O(n*n) -> O(n.log(n)).

2. Bài 2: Hãy giải bài toán:

Cho 2 số nguyên **x** và **n** ($x \leq 10^{18}$, $n \leq 10^{18}$)

$S = x^0 + x^1 + x^2 + x^3 + \cdots + x^n$

Yêu cầu:

- Suy nghĩ bài toán trên có mấy cách giải (Gợi ý có ít nhất 2 cách giải) và đối với mỗi cách hãy cho biết ứng dụng những kĩ thuật gì đã học.
- Viết mã giả cho các thuật toán các bạn đã nghĩ ra ở trên.

- **Cách 1: brute force**

+ Duyệt **i** từ 0 đến n và tính x^i rồi cộng lại

+ Độ phức tạp: O(n.log(n)) -> không thể chạy được khi **n** lớn.

+ Mã giả:

```
sol(x, n):
    S = 0
    for i in range(0, n)
        S = S + x^i
    return S
```

- Cách 2: Sử dụng công thức toán

+ Có thể tính tổng trên nếu x khác 1 bằng công thức:

$$S = \frac{x^{n+1}-1}{x-1}$$

+ Độ phức tạp: O(log(n)) do tính lũy thừa của x^{n+1}.

+ Mã giả:

```
sol(x, n):
    if x == 1:
        return n + 1
    else:
        return (x^(n+1) - 1) / (x - 1)
```