# Bài tập CS112: Phân tích độ phức tạp của đệ quy

Nhóm : 12
Hoàng Minh Thái 23521414
Nguyễn Trọng Tát THành

## Problem 1. Given an input string of numbers, find all combinations of numbers that can be formed using digits in the same order.

### Thuật toán :

Ta gọi đệ quy quy lui để tạo ra toàn bộ hoán vị, cuối cùng kiểm tra xem có số 0 ở đầu hay không và kiểm tra xem hoán vị đó đã xuất hiện chưa để đếm kết quả

```
Def Cal(int id)

// Nếu id > n, đã chọn được tất cả các chữ số
    if (id > n) {
        // Nếu chữ cái đầu tiên = 0 thì không hợp lệ
        if (per[1] != 0) {
            // Sử dụng một cấu trúc dữ liệu để kiểm ta xem hoán vị có toàn tại chưa
            if (per not in s) {
                cnt++; // Tăng biến đếm
                cout << per << '\n'; // Ghi ra kết quả
            }
        }
    }
    else {
    // Chọn các số chưa được chọn
        for (int i = 1; i <= n; i++)
            if (a[i] chưa chọn) {
                per[id] = a[i]; // chọn a[i]
                dd[i] = true; // đánh dấu a[i] đã chọn
                cal(id + 1);
                dd[i] = false; // xóa đánh dấu a[i]
            }
    }
```

### Phân tích độ phíc tạp

- Số lần gọi đệ quy:

  Thuật toán này tất cả các hoán vị của $n$ phân tử nên sẽ có tổng cộng $n!$ hoán vị được sinh ra. Và để tạo ra một hoán vị thì ta cần phải gọi đệ quy n lần. Vậy ta gọi đệ quy tổng cộng $O(n! * n)$ lần

- Độ phíc tạo của việc in ra màn hình

  Với việc mỗi hoán vị hợp lệ ta đều in ra màn hình thì sẽ tốn $O(n)$ để in ra nên độ phức tạp của chức năng này sẽ là $O(n! * n)$

- Độ phíc tạp của kiểm tra tính hợp lệ và lưu trữ trong $s$:

  Trong những lần mà các hoán vị đã hoàn thành. Thì ta cần phải kiểm tra xem đã toàn tại một số nào như vậy trước đây chưa. Đề xuất xử dụng cấu trúc dữ liệu set môi lần kiểm tra sẽ mất trung bình khoảng $O(log(n!))$. Và chúng ta cần phải kiểm tra $n!$ lần nên độ phức tạp ở đây rơi vào khoảng $O(n!log(n!))$

- Độ phíc tạp tổng thể:
  Kết hợp các ý ở trên ta có độ phức tạp của bài toán sẽ là

$$MAX(O(n! \cdot n), O(n! \cdot log(n!)$$

## Problem 2. Given a set of characters and a positive integer $k$, print all possible strings of length $k$ that can be formed from the given set.

### Thuật toán:

Tương tự với bài trên ta cũng sẽ gọi để quy để giải quyết bài toán này

```
Def Cal(int id)

// Nếu id > k, đã chọn được k kí tự
    if (id > k) {
        // In ra kết quả
        cout << per;
    }
    else {
    // Chọn các kí tự
        for (int i = 1; i <= n; i++) {
            per[id] = a[i]; // chọn a[i]
            cal(id + 1);
        }
    }
```

### Phân tích độ phíc tạp

- Số lần gọi đệ quy:
  Vỗi mỗi phần tử của chuỗi được tạo ra ta đều có $n$ cách chọn. Vậy số xâu có thể tạo ra sẽ là $n^k$. Và một xâu hoàn chỉnh ta cần gọi đệ quy $k$ lần nên độ phức tạo sẽ là $O(n^k \cdot k)$
- Độ phíc tạo của việc in ra màn hình
  Sau khi đã tạo ra được một chuỗi đủ $k$ phần tử thì ta mỡi in ra màn hình nên công đoạn này cũng chỉ tốn $O(n^k \cdot k)$
- Độ phíc tạp tổng thể:
  Kết hợp các ý ở trên ta có độ phức tạp của bài toán sẽ là

$$O(n^k \cdot k)$$

## Tower of Hanoi

### Thuật toán

```
Def TowerOfHanoi(n, source, auxiliary, destination):
    Nếu n = 1:
        In "Di chuyển đĩa từ" source "sang" destination
        Trả về

    // Bước 1: Di chuyển n-1 đĩa từ cọc nguồn sang cọc trung gian
    TowerOfHanoi(n - 1, source, destination, auxiliary)

    // Bước 2: Di chuyển đĩa còn lại từ cọc nguồn sang cọc đích
    In "Di chuyển đĩa từ" source "sang" destination
```

```
11
12      // Bước 3: Di chuyển n-1 đĩa từ cọc trung gian sang cọc đích
13      TowerOfHanoi(n - 1, auxiliary, source, destination)
```

## Phân tích độ phíc tạp

Giả sử $T(n)$ là số bước cần thiết để di chuyển $n$ đĩa.

- Khi $n = 1$, chỉ cần thực hiện một bước duy nhất để di chuyển đĩa từ cọc nguồn sang cọc đích. Vậy $T(1) = 1$.
- Khi $n > 1$:
  - Đầu tiên, cần $T(n - 1)$ bước để di chuyển $n - 1$ đĩa từ cọc nguồn sang cọc trung gian.
  - Sau đó, di chuyển đĩa thứ $n$ từ cọc nguồn sang cọc đích, mất thêm một bước.
  - Cuối cùng, cần $T(n - 1)$ bước để di chuyển $n - 1$ đĩa từ cọc trung gian sang cọc đích.

Do đó, công thức đệ quy cho $T(n)$ là:

$$T(n) = 2 \cdot T(n - 1) + 1$$

Bằng cách giải phương trình đệ quy này, ta có thể thấy rằng:

$$T(n) = 2^n - 1$$

Kết luận: bài toán tháp hà nội có độ phức tạo là $O(2^n)$

Hết