Phân Tích Thuật Toán
Trường Đại Học Khoa Học Tự Nhiên
Nguyễn Thanh Bình

Report lab 8

Nguyễn Quốc Bảo - 18110053

Bài 1. Xây dựng dãy Fibonnaci như sau

$$\begin{cases} f_0 = "abc", f_1 = "def" \\ f_{n+1} = strcat(f_n, f_{n-1}), \forall n \geq 1 \end{cases}$$

trong đó , hàm strcat là hàm nối hai chuỗi thành một chuỗi mới.

a. Hãy xây dựng thuật toán để tìm chính xác ký tự thứ k của từ fn, $\forall n \geq 0$

b. Đánh giá độ phức tạp của thuật toán đưa ra.

Để thực hiện bài toán này ta cần tìm dãy fibonacci như mô tả ở trên và tìm ký thứ k trong dãy fibonacci. Ở đây ta có hàm **find-char-kth-fn** dùng để thực hiện điều này.

```
def find_char_kth_fn(n,k):
    global counter_assign
    global counter_compare

    fn = Fibonnaci_char(n)
    print('f{} = {}'.format(n,fn))
    print('character '+str(k)+'th in f{} is'.format(n) , fn[k-1])
```

Trước đó ta cần tìm dãy fibonacci bằng hàm **Fibonacci-char** như bên dưới.

Cùng ý tưởng với dãy fibonnaci số thì bây giờ sẽ là chuỗi.

```
def Fibonnaci_char(n):
    global counter_assign
    global counter_compare

    f0 = 'abc'
    f1 = 'def'

    counter_assign += 2
    counter_compare += 1

    if (n == 1):
        return f1

    for i in range(0,n):

        f = f1 + f0
        f0 = f1
        f1 = f
        counter_compare += 1
        counter_assign += 3

    return f0
```

Theo như thuật toán đã cài ở trên thì thuật toán này có độ phức tạp là $O(n)$. Vì trong thuật toán này chỉ có một vòng lặp và số lần lặp của nó là n. Ta có thể dễ dàng đếm được số lần thực hiện phép so sánh sẽ là n + 1 và số lần thực hiện phép gán sẽ là 3n + 2. vậy tổng lại sẽ là 4n + 3. Vì thế ta có thể kết luận thuật toán trên là $O(n)$.

Sau khi chạy có kết quả như sau:

```
f4 = defabcdefdefabc
character 3th in f4 is f
When N = 4 , then count_compare: 5 and count_assign: 14
```