BỘ GIÁO DỤC VÀ ĐÀO TẠO

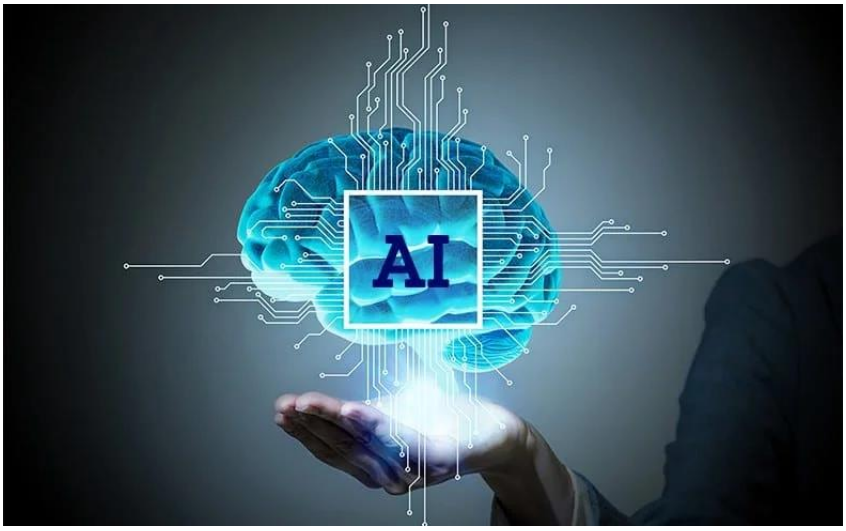
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA TOÁN - TIN HỌC

-----📖-----

BÀI BÁO CÁO THỰC HÀNH TUẦN 8

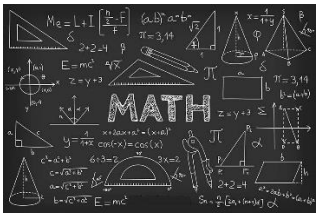


MÔN HỌC: **Phân Tích Thuật Toán**

Sinh Viên: **Trần Công Hiếu - 21110294**

Lớp: **21TTH**

TP.HCM, ngày 02 tháng 06 năm 2024



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA TOÁN – TIN HỌC

BÀI BÁO CÁO THỰC HÀNH TUẦN 8

HK2 - NĂM HỌC: 2024-2025

MÔN: PHÂN TÍCH THUẬT TOÁN

SINH VIÊN: TRẦN CÔNG HIẾU

MSSV: 21110294

LỚP: 21TTH

NHẬN XÉT

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp.HCM, Ngày……Tháng……Năm 2021

Giảng viên Bộ môn

Mục Lục

Bài 1. 5

1.1. Trình bày đoạn mã. 5

1.2. Đánh giá độ phức tạp của thuật toán đưa ra. 7

Bài 1.

1.1. Trình bày đoạn mã.

(File: bai1.py)

```
1   f_0 = 'abc'
2   f_1 = 'def'
3   k = -1
4   while(k<=0):
5       k = int(input("Nhap k (k>=1): "))
6
7   def fibonacci(n):
8       if(n==0):
9           return 'abc'
10      if(n==1):
11          return 'def'
12      else:
13          return fibonacci(n-1) + fibonacci(n-2)
14
```

“f_0 = ‘abc’”: Khởi tạo biến f_0 để lưu trữ chuỗi tại n = 0 của dãy Fibonacci là abc.

“f_1 = ‘def’”: Khởi tạo biến f_0 để lưu trữ chuỗi tại n = 0 của dãy Fibonacci là abc.

“k = -1”: Khởi tạo biến k = -1 để thỏa điều kiện vòng lặp while tiếp đó.

“while(k<=0):”: Tạo vòng lặp while với điều kiện k nhỏ hơn hoặc bằng 0. Điều này đồng nghĩa điều kiện để dừng vòng lặp là khi người dùng phải nhập k (k = int(input(“Nhap k (k>=1)”))) thỏa lớn hơn 1.

“def fibonacci(n):”: Định nghĩa hàm fibonacci với đối số truyền vào là n. Hàm sẽ định nghĩa dãy fibonacci với bước cơ sở là khi n = 0 và n = 1, ứng với từng n đó thì ta return về chuỗi abc và def. Bước đệ qui sẽ là nối 2 chuỗi với n-1 và n-2.

```
15  temp = 0
16  a = 0
17  b = 3
18  n = 0
19  while(k>temp):
20      temp = a + b
21      a = b
22      b = temp
23      n += 1
24  if(k<=3):
25      n-=1
26
```

Ở đoạn này, chức năng chính là khi người dùng nhập k, ta phải tìm n tương ứng (lớn hơn k nhỏ nhất) để từ đó tìm chuỗi ứng với fibonacci theo n thông qua độ dài của chuỗi.

Khởi tạo các biến temp = 0, a = 0, b = 3 và n = 0 bằng các câu lệnh: “temp = 0”, “a = 0”, “b=0” và “n=0”.

“while(k>temp)”: Tạo vòng lặp while với điều kiện là k lớn hơn temp, tức điều kiện dừng là khi độ dài chuỗi của dãy fibonacci tại n lớn hơn hoặc bằng k. Trong vòng lặp, với mỗi bước ta sẽ lần lượt cập nhật biến temp bằng tổng a và b, sau đó gán a bằng b và b bằng temp, cập nhật n thêm 1 đơn vị, cứ như vậy thì sẽ thu được độ dài độ dài chuỗi tương ứng với n để tìm ra n lớn hơn k nhỏ nhất.

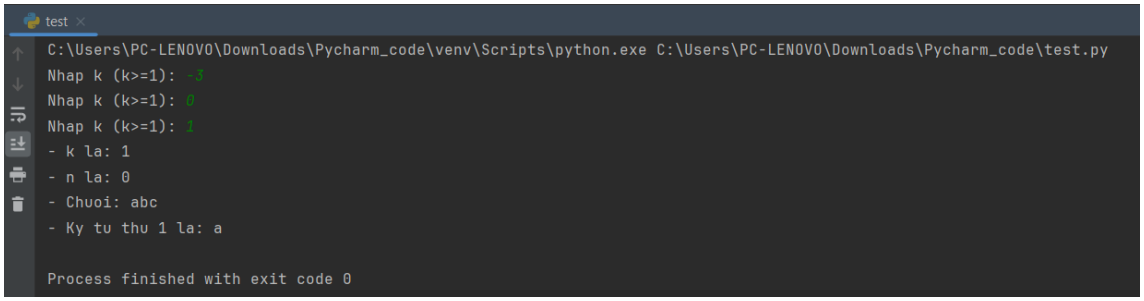
“if(k<=3): n-=1”: Với trường hợp k nhỏ hơn hoặc bằng 3 thì vấn đề là n = 0 sau khi thoát khỏi vòng lặp sẽ vẫn cập nhật lên 1 đơn vị, do đó ta phải kiểm tra khi k nhỏ hơn hoặc bằng 3 và trừ đi 1 đơn vị để kết quả đúng.

```
27  print("- k la:",k)
28  print("- n la:",n)
29  print("- Chuoi:",fibonacci(n))
30  print("- Ky tu thu",k,"la:",fibonacci(n)[k-1])
31
```

Sau cùng là in kết quả ra để quan sát bằng các lệnh print(). Ta sẽ in lần lượt k, n, chuỗi fibnacci thu được ứng với n và thông báo ký tự thứ k là chuỗi trên với index k-1 (vì trong chuỗi index bắt đầu là 0).

*** Kết quả:**

- Thử nhập k nhỏ hơn hoặc bằng 1 và nhập k = 1.



- Thử k khác trường hợp đặc biệt trên.

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

Ý tưởng để tìm độ phức tạp cho phần fibonacci này là thay vì tính trực tiếp, ta sẽ tính thông qua độ dài (len()) của dãy. Thuật toán tìm chính xác ký tự thứ k trong f_n trên có 2 phần cần tính toán, gọi là $T_1(n)$ (dãy fibonacci) và $T_2(k)$ (vòng lặp while tìm n lớn hơn k nhỏ nhất, nó phụ thuộc vào k).

*** T_1 = ?**

Với ý tưởng đó, ta thu được:

$$\begin{cases} a_0 = 3, a_1 = 3 \\ a_{n+1} = a_n + a_{n-1}, \forall n \geq 1 \end{cases}$$

Đặt G(z) là hàm sinh cho dãy (a_n). Ta có:

$$G(z) = a_0 + a_1 z + \sum_{k=2}^{\infty} a_k z^k$$

$$= a_0 + a_1 z + \sum_{k=2}^{\infty} (a_{k-1} + a_{k-2}) z^k$$

$$= a_0 + a_1 z + z \sum_{k=2}^{\infty} a_{k-1} z^{k-1} + z^2 \sum_{k=2}^{\infty} a_{k-2} z^{k-2}$$

$$= a_0 + a_1 z + z(G(z) - a_0) + z^2 G(z)$$

$$= a_0 + (a_1 - a_0) z + z G(z) + z^2 G(z)$$

$$\Leftrightarrow (1 - z - z^2) G(z) = 3 + 0z = 3$$

$$\Leftrightarrow G(z) = \frac{3}{(1 - z - z^2)}$$

Phân tích $G(z)$, ta được:

$$G(z) = \frac{3}{(1 - z - z^2)} = \frac{A}{1 - \alpha z} - \frac{B}{1 - \beta z}$$

$$Với\ \alpha = \frac{1 + \sqrt{5}}{2}, \beta = \frac{1 - \sqrt{5}}{2}$$

Ta thu được hệ:

$$\begin{cases} A + B = 3 \\ -\beta A - \alpha B = 0 \end{cases} \Rightarrow \begin{cases} A = \dfrac{13 + 3\sqrt{5}}{10} \\ B = \dfrac{15 - 3\sqrt{5}}{10} \end{cases}$$

$$G(z) = \frac{A}{1 - \alpha z} + \frac{B}{1 - \beta z} = A \sum_{k=0}^{\infty} (\alpha z)^k + B \sum_{k=0}^{\infty} (\beta z)^k$$

Khi:

$$\begin{cases} |z| < \dfrac{1}{|\alpha|} \\ |z| < \dfrac{1}{|\beta|} \end{cases} \Rightarrow \begin{cases} |z| < \dfrac{2}{1+\sqrt{5}} \\ |z| < \dfrac{2}{-1+\sqrt{5}} \end{cases} \Rightarrow |z| < \frac{2}{-1+\sqrt{5}}$$

$$\Rightarrow G(z) = \sum_{k=0}^{\infty} \left(\frac{15+3\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^k + \left(\frac{15-3\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^k z^k$$

$$\Rightarrow a_k = \left(\frac{15+3\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^k + \left(\frac{15-3\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^k$$

Thử: $a_0 = 3, a_1 = 3, a_2 = 6$.
Do đó, công thức tổng quát của dãy trên là:

$$a_n = \left(\frac{15+3\sqrt{5}}{10}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{15-3\sqrt{5}}{10}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n, \forall n \geq 1$$

Suy ra: $T_1(n) = O(logn)$.

$*$ T_2= ?.

Số lần lặp của vòng lặp while phụ thuộc tuyến tính vào k. Mỗi vòng lặp chỉ có các phép gán cơ bản. Do đó, độ phức tạp của đoạn mã này là: $T_2(k) = O(k)$.

*** Kết luận.**

Từ kết quả trên, ta có độ phức tạp của thuật toán trên trong việc tìm ký tự thứ k trong f_n là:

$$T(n,k) = T_1(n) + T_2(k) = O(logn) + O(k) = O(logn + k)$$

*** Nhận xét.**

Có thể thấy độ phức tạp của thuật toán phụ thuộc vào n và k, ta nên cân nhắc giải quyết bài toán theo hướng phụ thuộc vào cùng 1 biến đầu vào. Hoặc dễ dàng thấy, ở vòng lặp while, khi k thay đổi thì n cũng thay đổi, điều đó cho thấy n và k có mối liên hệ nào đó với nhau, chỉ cần tìm được thì khi đó từ $T(n,k)$ ta có thể đưa về dạng $T(k)$ cho bài toán trên phụ thuộc vào giá trị đầu vào k.