# Bài 1. Big O và mã giả

Khái niệm: Độc phức tạp của thuật toán là thuật ngừ thường dùng để chỉ khoảng thời gian tieu hao để chạy một thuật toán. Dùng để so sánh mức độ hiệu quả của thuật toán.

## Tính Big O

O(1):

* Phép gán, kiểm tra, so sánh
* Phép tính: +, -, \*, /, …
* Cấu trúc rẽ nhánh

O(logn):

* Mỗi lần lặp giảm đi 1 nửa số lượng phần tử duyệt

VD: Tìm kiếm nhị phân, …

O(n):

* Vòng lặp qua các phần tử
* Các thao tác trên mảng: sao chép, chèn, xóa, lặp lại, duyệt

O(nlogn):

* VD: Mỗi lần duyệt các phần tử trong mảng thì sử dụng tìm kiếm nhị phân

O(n^2):

* Các vòng lặp lồng nhau

O(2^n):

* Tăng trưởng nhân đôi - VD tính fbnc bằng đệ quy

O(n!):

* Hoán vị các phần tử.

Một số phép toán với Big O:

* Quy tắc hằng:
  + O(c \*f(n)) = O(f(n))

VD: O(2n) = O(n)

* Quy tắc cộng:
  + O(f(n) + g(n)) = O(max(f(n), g(n))

VD: O(2 + n) = O(max(2, n)) = O(n)

* Quy tắc nhân:
  + O(f(n)) \* O(g(n)) = O(f(n) \* g(n))

VD: O(3n) \* O(n^2)= O(3n \* n^2) = O(3n^3) = O(n^3)

## Bài tập áp dụng

1. Tính: O(f(n)) với

a) f(n) = 2 + 5n + n^2 => O(n^2)

b) f(n) = n\*(n - 2) = n^2 => O(n^2)

c) f(n) = 2\*(n + 6)\*(n - 5) => 2\*n^2 = O(n^2)

d) f(n) = 3log2(n) + 2 = 3log(n) => O(logn)

f) f(n) = n\*(log2(5) + 1) + 2log2(n) = n + 2log(n) => O(n)

2. Phân tích

Ảnh có chứa văn bản, màn hình

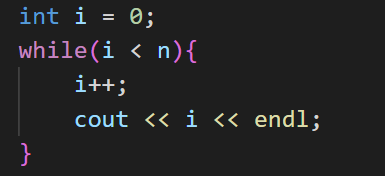
Mô tả được tạo tự động

O1 + O1 + O1\*O3 = O1

Ảnh có chứa văn bản

Mô tả được tạo tự động

O3 + O10\*O(n)\*O1 = O(n)



O1 + On\*O2 = On

Ảnh có chứa văn bản

Mô tả được tạo tự động

i = 1 => 1 2 4 8 16 => số lần lặp log2(n) => O(logn)

Ảnh có chứa văn bản

Mô tả được tạo tự động

n\*(n - i)\*1\*3 = n^2 => O(n^2)

n\*(n-i)\*1\*1 = O(n^2)

Ảnh có chứa văn bản

Mô tả được tạo tự động

O(2^n)

Ảnh có chứa văn bản

Mô tả được tạo tự động

1 + 1 + 1 + n\*n\*2=O(n^2)

Ảnh có chứa văn bản, màn hình, trong nhà

Mô tả được tạo tự động

1 + 1 + n = O(n)

Ảnh có chứa văn bản, màn hình

Mô tả được tạo tự động

O(logn)

# Mã giả:

* Mã giả được sử dụng để mô tả giải thuật nên không thể biên dịch cũng như thực thi như mã thật.
* Mỗi đoạn bắt đầu bằng BEGIN/START và kết thúc bằng END/STOP
* Để hiển thị giá trị nào đó ta sử dụng DISPLAY hoặc WRITE
* Để nhập INPUT/READ
* Không cần khai báo biến
* 1 số cấu trúc cơ bản:
  + IF <đk> THEN: <câu lệnh 1> ELSE <câu lệnh 2>
  + WHILE <đk> DO: <câu lệnh>
  + FOR <biến> TO <đích> DO:
  + AND / OR, && vs || , >, <, ≥

VD:

BEGIN  
INPUT A, B  
C = A + B  
DISPLAY C  
END

VD: Viết mã giả sắp xếp

SELECTION\_SORT(A, n)

BEGIN

FOR i = 0 TO n - 1 DO:

FOR j = i + 1 TO n - 1 DO:

IF A[i] > A[j] THEN:

BEGIN

t = A[i]

A[i] = A[j]

A[j] = t

END

END

VD: Hàm thêm phần tử vào cuối list

Node{

data;

\*next;

}

Push(\*&head, x)

BEGIN

IF head == NULL THEN:

BEGIN

\*p = new Node;

p->data = x;

p->next = NULL;

head = p;

END

ElSE:

Push(head->next, x)

END

VD: Hàm tính x^n:

pow(x, n){

IF n == 0 THEN:

RETRUN 1

ELSE:

RETURN x\*pow(x, n-1);

}

x^4 = x\*pow(x, 3) =.. x^4

* O(n)

pow(x, n){

IF n == 0 THEN:

RETURN 1

IF n mod 2 == 0 THEN

RETURN pow(x, n/2)^2;

RETURN x\*pow(x, n/2)^2;

}

x^4 = (x^2)^2 => 16 => 8 => 4 => 2 log2(n)

x^5 = x\*x^4

* O(logn)

Bài tập:

1. Viết mã giả thuật toán kiểm tra số nguyên tố : -2 -3

Check(n)

BEGIN

IF ABS(n) < 2 THEN:

RETURN FALSE

IF n < 0 THEN:

n = n\*(-1)

FOR i = 2 TO SQRT(n) DO:

IF n mod i == 0 THEN:

RETURN FALSE

RETURN TRUE

END

2. Viết mã giả thuật toán giải phương trình bậc 2 // ax^2 + bx + c = 0

GIAIPT(a, b, c)

BEGIN

IF a == 0 THEN:

BEGIN

IF b == 0 THEN:

BEGIN

IF c == 0 THEN:

DISPLAY “vo so Nghiem”

ELSE:

DISPLAY “vo Nghiem”

END

ELSE:

DISPLAY -c/b

END

ELSE:

BEGIN

denta = b\*b - 4\*a\*c

IF denta < 0 THEN:

DISPLAY “vo Nghiem”

ELSE IF denta == 0 THEN:

DISPLAY -b/(2\*a);

ELSE:

BEGIN

DISPLAY “X1 = “ (-b + sqrt(denta))/(2\*a)

DISPLAY “X2 = “ (-b - sqrt(denta))/(2\*a)

END

END

END