PHÂN TÍCH THUẬT TOÁN

REVIEW

- Bài toán: Cho một tập các số nguyên $S = \{s1, s2, . . . , sn\}$, và một giá trị đích T. Tìm một tập con của S sao cho tổng các số trong tập con đó đúng bằng T.
  Ví dụ, Tồn tại một tập con trong $S = \{1, 2, 5, 9, 10\}$ mà tổng là $T = 22$ nhưng lại không tồn tại với $T = 23$.

Tìm phản ví dụ cho các thuật toán sau

REVIEW

- (a) Lần lượt chọn các phần tử trong S theo thứ tự từ trái qua phải nếu chúng phù hợp (thuật toán first-fit).
- (b) Lần lượt chọn các phần tử trong S theo thứ tự từ nhỏ đến lớn (thuật toán best-fit).
- (c) Lần lượt chọn các phần tử trong S theo thứ tự từ lớn nhất đến nhỏ nhất.

NỘI DUNG

- Phân tích thuật toán
- Mô hình RAM
- Tốt nhất, tồi nhất, trung bình
- Ký hiệu O-lớn
- Phân tích tiệm cận
- Tốc độ tăng và tính thống trị
- Một số tính chất của phân tích O-lớn

PHÂN TÍCH THUẬT TOÁN?

Bài toán: tìm phần tử lớn nhất thứ *k*

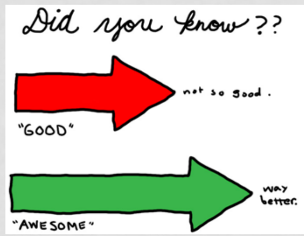
- Đầu vào: Dãy số gồm *n* số nguyên $a_1, a_2, .., a_n$, và số nguyên *k* (0< *k* ≤ *n*)
- Đầu ra: Giá trị phần tử lớn nhất thứ *k* trong dãy.

Có 2 thuật toán A, B để giải bài toán.

Với n = 100,000, k=100

- A cài đặt bằng C chạy mất 12 s
- B cài đặt bằng java chạy mất 19 s

Thuật toán A tốt hơn B ?



PHÂN TÍCH THUẬT TOÁN?

- Đánh giá hiệu quả của thuật toán mà không cần cài đặt.
- 2 mô hình :
  - Mô hình RAM (***Random Access Machine***)
  - Phân tích tiệm cận độ phức tạp trong trường hợp tồi nhất

- **Đánh giá thuật toán**: dự đoán các tài nguyên mà thuật toán cần.
- **Tài nguyên**: Thời gian CPU, bộ nhớ, băng thông, phần cứng…

MÔ HÌNH RAM

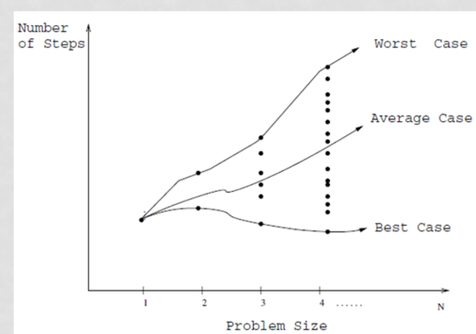
- Thực hiện thuật toán trên một máy tính giả định gọi là *Random Access Machine* hoặc RAM.
  - Mỗi phép tính đơn giản (+, *, –, =, if, call) thực hiện trong 1 đơn vị thời gian (hoặc 1 bước).
  - Vòng lặp, hàm, thủ tục: là kết hợp của nhiều phép tính đơn lẻ
  - Mỗi bước truy cập bộ nhớ mất 1 đơn vị thời gian
  - Luôn có đủ bộ nhớ cần thiết để thực hiện thuật toán

- Đánh giá thời gian thực hiện thuật toán bằng cách đếm số đơn vị thời gian cần.

TỐT NHẤT, TỒI NHẤT VÀ TRUNG BÌNH

- Phân tích thuật toán trong trường hợp tổng quát, với một đầu vào bất kỳ thỏa mãn

➔Phân tích trường hợp: tốt nhất, tồi nhất và trung bình



TỐT NHẤT, TỒI NHẤT VÀ TRUNG BÌNH

- **Độ phức tạp trong trường hợp tồi nhất (*worst-case complexity*):** Là số lượng bước ***lớn nhất*** thuật toán cần thực hiện với bất cứ đầu vào kích thước *n* nào.
- **Độ phức tạp trong trường hợp tốt nhất (*best-case complexity*):** Là số lượng bước ***nhỏ nhất*** thuật toán cần thực hiện với bất cứ đầu vào kích thước *n* nào.
- **Độ phức tạp trong trường hợp trung bình (*average-case complexity*):** Là số lượng bước ***trung bình*** thuật toán cần thực hiện trên tất cả các trường hợp đầu vào kích thước *n*.

- Mỗi độ phức tạp là một hàm của ***thời gian*** và ***kích thước*** đầu vào

$$T(n) = 120n^3 + 12.4n^2 - 43nlogn + 9n$$

KÝ HIỆU O LỚN

- Khó xác định chính xác các hàm đánh giá độ phức tạp trong trường hợp tốt nhất, tồi nhất, trung bình
  - Có rất nhiều điểm lồi: thời gian thực hiện biến đổi trong một số trường hợp đầu vào đặc biệt. VD tìm kiếm nhị phân nếu đầu vào $n = 2^k - 1$
  - Để chính xác thì cần phân tích rất tỉ mỉ.

$$T(n) = 120n^3 + 12.4n^2 - 43nlogn + 9n$$

- **Ký hiệu O lớn**: đơn giản phân tích, bỏ qua những thành phần mà không ảnh hưởng đến khi so sánh các thuật toán

  Trong phân tích O lớn $f(n) = 2n^2$ và $g(n) = 3n^2 + 5n$ là tương đương

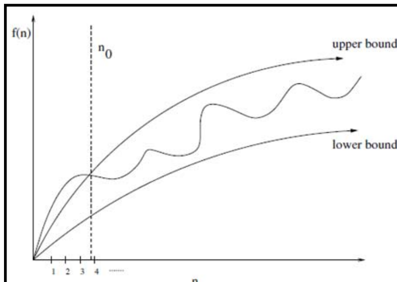
- Các ký hiệu tiệm cận (O, Θ, Ω) dùng để phân tích độ phức tạp trong thực tế

PHÂN TÍCH TIÊM CẬN

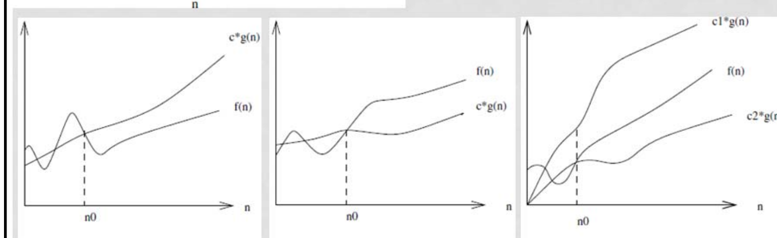
Định nghĩa O lớn chính thức:

- $f(n) = O(g(n))$: nghĩa là $c.g(n)$ là giới hạn trên của $f(n)$. Do vậy tồn tại hằng số c sao cho $f(n) \le c.g(n)$ luôn đúng với mọi $n \ge n_0$
- $f(n) = \Omega(g(n))$: nghĩa là $c.g(n)$ là giới hạn dưới của $f(n)$. Do vậy tồn tại hằng số c sao cho $f(n) \ge c.g(n)$ luôn đúng với mọi $n \ge n_0$
- $f(n) = \Theta(g(n))$: nghĩa là $c_1.g(n)$ là giới hạn trên, và $c_2.g(n)$ là giới hạn dưới của $f(n)$. Do vậy tồn tại hằng số c_1 và c_2 sao cho $f(n) \le c_1.g(n)$ và $f(n) \ge c_2.g(n)$ luôn đúng với mọi $n \ge n_0$. Nói cách khác $g(n)$ là giới hạn chặt của $f(n)$

Với c, c_1, c_2 là các hằng số dương không phụ thuộc vào n, và $n_0 > 0$



Cận trên, cận dưới để đánh giá cho độ phức tạp của hàm



(a) O lớn (b) Ω lớn (c) Θ lớn

KÝ HIÊU O LỚN

Ví dụ

- $\boldsymbol{2n^2 - 4n + 5 = O(n^2)}$ vì chọn $c = 2$ thì $2n^2 > 2n^2 - 4n + 5$
- $\boldsymbol{2n^2 - 4n + 5 = O(n^3)}$ vì chọn $c = 1$ thì $n^3 > 2n^2 - 4n + 5$ khi $n > 1$
- $\boldsymbol{2n^2 - 4n + 5 \neq O(n)}$ vì với bất kỳ hằng số c nào thì $cn < 2n^2 - 4n + 5$ khi $n > c$

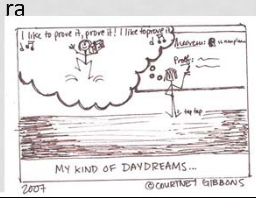
OMEGA LỚN

- $\boldsymbol{2n^2 - 4n + 5 = \Omega(n^2)}$ vì chọn $c = 1.5$ thì $1.5n^2 < 2n^2 - 4n + 5$ khi $n > 50$
- $\boldsymbol{2n^2 - 4n + 5 \neq \Omega(n^3)}$ vì với bất kỳ giá trị c thì $cn^3 > 2n^2 - 4n + 5$ khi n đủ lớn ($n > 100c$ nếu $c > 1$, $n > 10/c$ nếu $c < 1$)
- $\boldsymbol{2n^2 - 4n + 5 = \Omega(n)}$ vì với bất kỳ hằng số c nào thì $cn < 2n^2 - 4n + 5$ khi $n > 5c$

THETA LỚN

- $\boldsymbol{2n^2 - 4n + 5 = \Theta(n^2)}$ vì cả O, Ω đều đúng
- $\boldsymbol{2n^2 - 4n + 5 \neq \Theta(n^3)}$ vì chỉ O đúng
- $\boldsymbol{2n^2 - 4n + 5 \neq \Theta(n)}$ vì chỉ Ω đúng

Để chứng minh $f(n) = \Theta(g(n))$ thì cần chỉ ra
- $f(n) = O(g(n))$
- $f(n) = \Omega(g(n))$

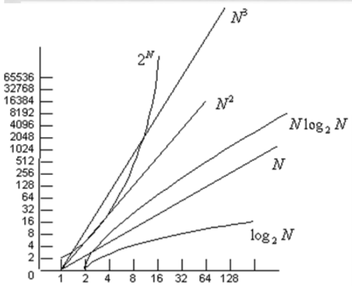


VÍ DỤ

Khẳng định sau đúng hay sai? Tại sao ?

- $2^{n+3} = \Theta(2^n)$
- $3^{2n} = \Theta(3^n)$
- $(x+y)^2 = O(x^2 + y^2)$

TỐC ĐỘ TĂNG VÀ QUAN HỆ THỐNG TRỊ



- **Tốc độ tăng của một số hàm thông dụng**

$n \quad f(n)$	$\lg n$	n	$n \lg n$	n^2	2^n	$n!$
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	3.63 ms
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	1 ms	77.1 years
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	1 sec	8.4×10^{15} yrs
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	18.3 min	
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	13 days	
100	0.007 μs	0.1 μs	0.644 μs	10 μs	4×10^{13} yrs	
1,000	0.010 μs	1.00 μs	9.966 μs	1 ms		
10,000	0.013 μs	10 μs	130 μs	100 ms		
100,000	0.017 μs	0.10 ms	1.67 ms	10 sec		
1,000,000	0.020 μs	1 ms	19.93 ms	16.7 min		
10,000,000	0.023 μs	0.01 sec	0.23 sec	1.16 days		
100,000,000	0.027 μs	0.10 sec	2.66 sec	115.7 days		
1,000,000,000	0.030 μs	1 sec	29.90 sec	31.7 years		

TỐC ĐỘ TĂNG VÀ QUAN HỆ THỐNG TRỊ

- O lớn nhóm các hàm thành các ***lớp hàm***.
  $n + 4$ và $100.3n - 3$ là thuộc lớp hàm $\Theta(n)$
- Hai hàm f, g thuộc hai lớp khác nhau có quan hệ theo các ký hiệu tiệm cận Ω, O khác nhau
- Các lớp hàm thông dụng:
  - Hàm hằng $f(n) = 1$. Thời gian thực hiện là hằng số VD hàm tính tổng 2 số
  - Hàm loga $f(n) = logn$. VD tìm kiếm nhị phân
  - Hàm tuyến tính $f(n) = n$. VD Tìm giá trị lớn nhất trong dãy số
  - Hàm siêu tuyến tính $f(n) = nlogn$. VD QuickSort, MergeSort
  - Hàm bậc hai $f(n) = n^2$. VD Sắp xếp nổi bọt (bubble sort)
  - Hàm bậc ba $f(n) = n^3$.
  - Hàm mũ $f(n) = c^n$, c là hằng số >1.
  - Hàm giai thừa $f(n) = n!$

TỐC ĐỘ TĂNG VÀ QUAN HỆ THỐNG TRỊ

- Quan hệ thống trị:
  $$n! \gg c^n \gg n^3 \gg n^2 \gg nlogn \gg n \gg \log n \gg 1$$
- Giới hạn và quan hệ thống trị của các hàm
  - $f(n)$ thống trị $g(n)$ nếu $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

VD.

$f(n) = n^2$ không thống trị $g(n) = 3n^2 + 5$ vì $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 3$

$f(n) = n^2$ thống trị $g(n) = n^{1.999999}$ vì $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

$$n! \gg c^n \gg n^3 \gg n^2 \gg n^{1+\epsilon} \gg nlogn \gg n \gg \sqrt{n} \gg \log^2 n \gg \log n \gg \log n/\log\log n \gg \log\log n \gg 1$$

CÁC PHÉP TÍNH VỚI O LỚN

- Cộng hai hàm
  - $O(f(n)) + O(g(n)) \to O(\max(f(n), g(n)))$
  - $\Omega(f(n)) + \Omega(g(n)) \to \Omega(\max(f(n), g(n)))$
  - $\Theta(f(n)) + \Theta(g(n)) \to \Theta(\max(f(n), g(n)))$
- Nhân hàm
  - $O(c \cdot f(n)) \to O(f(n))$
  - $\Omega(c \cdot f(n)) \to \Omega(f(n))$
  - $\Theta(c \cdot f(n)) \to \Theta(f(n))$

  c là một hằng số dương bất kỳ

CÁC PHÉP TÍNH VỚI O LỚN

- Nhân hai hàm
  - $O(f(n)) * O(g(n)) \rightarrow O(f(n) * g(n))$
  - $\Omega(f(n)) * \Omega(g(n)) \rightarrow \Omega(f(n) * g(n))$
  - $\Theta(f(n)) * \Theta(g(n)) \rightarrow \Theta(f(n) * g(n))$

MỘT SỐ TÍNH CHẤT

Tính truyền ứng – transitivity
- Nếu $f(n) = O(g(n))$ và $g(n) = O(h(n))$ thì $f(n) = O(h(n))$
- Nếu $f(n) = \Omega(g(n))$ và $g(n) = \Omega(h(n))$ thì $f(n) = \Omega(h(n))$
- Nếu $f(n) = \Theta(g(n))$ và $g(n) = \Theta(h(n))$ thì $f(n) = \Theta(h(n))$

Tính đối xứng – symmetry
- $f(n) = \Theta(g(n))$ khi và chỉ khi $g(n) = \Theta(f(n))$

Tính đối xứng chuyển vị - transpose symmetry
- $f(n) = O(g(n))$ khi và chỉ khi $g(n) = \Omega(f(n))$

MỘT SỐ TÍNH CHẤT

Chứng minh

Nếu $f(n) = O(g(n))$ và $g(n) = O(h(n))$ thì $f(n) = O(h(n))$

Ta có
- $f(n) = O(g(n))$ tức là $f(n) \leq c_1 g(n)$ với $n > n_1$
- $g(n) = O(h(n))$ tức là $g(n) \leq c_2 h(n)$ với $n > n_2$

Suy ra
- $f(n) \leq c_1 g(n) \leq c_1 c_2 h(n)$ với $n > \max(n_1, n_2)$

Chọn $c_3 = c_1 c_2$ và $n_3 = \max(n_1, n_2)$ thì $f(n) \leq c_3 h(n)$ khi $n > n_3$

Vậy $f(n) = O(h(n))$

MỘT SỐ VÍ DỤ

Thuật toán sắp xếp lựa chọn – Selection Sort

```
selection_sort(int s[], int n)
{
    int i,j; /* counters */
    int min; /* index of minimum */
    for (i=0; i<n; i++) {
       min=i;
       for (j=i+1; j<n; j++)
          if (s[j] < s[min]) min=j;
       swap(&s[i],&s[min]);
    }
}
```

MỘT SỐ VÍ DỤ

- Lệnh được lặp lại nhiều nhất chính là lệnh if

Phân tích trong trường hợp tồi nhất

- Vòng lặp ngoài lặp n lần (từ 0 tới n-1)
- Vòng lặp trong lặp n lần ứng với mỗi lần lặp của vòng ngoài
- Vậy số lượng bước (thời gian) cần thực hiện trong trường hợp tồi nhất là $n \times n \rightarrow O(n^2)$

MỘT SỐ VÍ DỤ

- **Phân tích chi tiết hơn**
  - $i = 0$ lệnh if lặp $n - 2$ lần (từ 1 tới n-1)
  - $i = 1$ lệnh if lặp $n - 3$ lần (từ 2 tới n-1)

    …
  - $i = n - 2$ lệnh if lặp 1 lần (từ n-1 tới n-1)
  - $i = n - 1$ lệnh if lặp 0 lần
- Số lần lặp của if sẽ là
  $T(n) = (n - 2) + (n - 3) + \cdots + 1 + 0 = (n - 2)(n - 1)/2$
- Vậy $T(n) = \Theta(n^2)$

MỘT SỐ VÍ DỤ

- Sắp xếp chèn – Insertion Sort

```
for (i=1; i<n; i++) {
    j=i;
    while ((j>0) && (s[j] < s[j-1])) {
          swap(&s[j],&s[j-1]);
          j = j-1;
    }
}
```

- Lệnh được lặp nhiều nhất là 2 lệnh bên trong while : lệnh cơ sở

MỘT SỐ VÍ DỤ

- **Phân tích trong trường hợp tồi nhất**
  - $i = 1$ lệnh cơ sở lặp 1 lần
  - $i = 2$ lệnh cơ sở lặp 2 lần

    …
  - $i = n - 2$ lệnh cơ sở lặp $n - 2$ lần
  - $i = n - 1$ lệnh cơ sở lặp $n - 1$ lần
- Số lần lặp của lệnh cơ sở sẽ là
  $T(n) = 1 + 2 + \cdots + (n - 2) + (n - 1) = (n - 1)n/2$
- Vậy $T(n) = O(n^2)$

Môt số công thức hay dùng

- $\sum_{a}^{b} 1 = b - a + 1$
- $\sum_{i=1}^{n} i = 1 + 2 + .. + n = \frac{n(n+1)}{2}$
- $\sum_{j=i}^{n} j = i + (i+1) + .. + n = \sum_{i=1}^{n} i - \sum_{j=1}^{i-1} j = \frac{(n^2+n-i^2+i)}{2}$
- $\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + .. + n^2 = \frac{(2n^3+3n^2+n)}{6}$
- $a^0 + a^1 + .. + a^n = \frac{(a^{n+1}-1)}{(a-1)}$ với $a \neq 1$
- $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$ nếu $|x| < 1$
- $\sum_{i=1}^{n} ic^i = c + 2c^2 + .. + nc^n = \frac{\left((n-1)c^{n+1} - nc^n + c\right)}{(c-1)^2}$