**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**BÀI TẬP MÔN PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN**

**HOMEWORK #03: Độ phức tạp và các ký hiệu tiệm cận**

**GV hướng dẫn: Huỳnh Thị Thanh Thương**

**Nhóm thực hiện: Nhóm Ba Con Sói**

**1. Bùi Quốc Thịnh 20520934**

**2. Hoàng Đình Hữu 20521384**

**3. Vũ Quốc Thái Bình 20521119**

**TP.HCM, ngày 12 tháng 4 năm 2022**

**Bài tập 2:**

Chuyển đổi về đơn vị

1 sec =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 sec | 1 min | 1 hour | 1 day | 1month | 1 year | 1 century |
| lgn |  |  |  |  |  |  |  |
|  |  | 36. | 1296. | 746496. | 6718464. | 994519296. | 996827586973696. |
| n |  | 6. | 36. | 864. | 2592. | 31536. | 31556736. |
| nlogn |  | 2801417 | 133378058 | 2755147513 | 71870856404 | 797633893349 | 68654697441062 |
|  |  | 7754 | 60000 | 293938 | 1609968 | 5615692 | 56175382 |
|  |  | 391 | 1532 | 4420 | 13736 | 31593 | 146677 |
|  |  | 25 | 31 | 36 | 41 | 44 | 51 |
| n! |  | 11 | 12 | 13 | 15 | 16 | 17 |

**Bài tập 3**: a) Đúng hay sai, giải thích vì sao?

Ảnh có chứa văn bản

Mô tả được tạo tự động

Phép suy ra ở bên trên là sai. Dấu “=” ở dòng 1,2 có thể được hiểu như là kí hiệu “”thuộc về một tập hợp còn ở dòng 3 là dấu “=” của một đẳng thức .Có cùng độ phức tạp không có nghĩa là 2 đẳng thức của chúng bằng nhau.

b) Xét

Giả sử:

Cho

Theo định nghĩa Big-O, xét và , ta có đccm:



Dễ thấy khi thì

Vì vậy, nên

Theo định nghĩa của Big-O, xét và , ta có đccm:



Dễ thấy khi thì

Vì vậy

Theo định nghĩa Big-O, xét và , ta có đccm:

Giả sử

Xét dấu hàm T(n) =

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | 0 7c | | | | |
| T(n) | + | 0 | - | 0 | + |

Nhìn vào bảng xét dấu, ta chỉ cần cho n=7c thì

Chọn

thì

c) Chứng minh

Giả sử ,

→

Để thì n <

Giả sử

Ảnh có chứa văn bản, ăng-ten

Mô tả được tạo tự động

Vô lí vì n không thể nào vừa lớn với mọi n và vừa nhỏ hơn hoặc bằng .

Giả sử

Ảnh có chứa ăng-ten, đối tượng

Mô tả được tạo tự động

Có n < và vì đoạn [] sẽ không thỏa điều kiện nên vô lí.

Từ 2 giả thuyết trên, ta suy ra được:

Ta có (\*)

Giả sử

Giả sử

Ảnh có chứa văn bản, ăng-ten

Mô tả được tạo tự động

Vô lí vì n không thể nào vừa lớn với mọi n và vừa nhỏ hơn hoặc bằng c.

Giả sử

Ảnh có chứa ăng-ten, đối tượng

Mô tả được tạo tự động

Có n < và vì đoạn [] sẽ không thỏa điều kiện nên vô lí.

Từ 2 giả thuyết trên ta có: (\*\*)

Từ (\*) và (\*\*) suy ra đccm:

Giả sử

(\*)

Để chứng minh thì phải tìm tồn tại ít nhất một hằng số c không phụ thuộc vào n, từ (\*) ta đã có thể thấy sự mẫu thuẫn khi c phụ thuộc vào n biến thiên nên ta có thể kết luận:

Bài tập 4: Sắp xếp dãy tăng dần

**Group 1**

**Group 2**

**Group 3**

**Group 4**

**Bài tập 5:**

* **với c là hằng số**

Xét (\*)

Ta có:

.1

Theo định nghĩa Big-O, xét ta có:

(\*\*)

Xét (\*\*\*)

Ta có:

Theo định nghĩa Big-O, ta xét , ta có:

(\*\*\*\*)

Từ (\*), (\*\*), (\*\*\*), (\*\*\*\*)

* **với C là hằng số**

Xét (\*)

Ta có:

Theo định nghĩa Big-O, xét , ta có:

(\*\*)

Xét (\*\*\*)

Ta có: , đặt

Theo định nghĩa Big-O, xét ta có:

(\*\*\*\*)

Từ (\*),(\*\*),(\*\*\*) và (\*\*\*\*)

* **Nếu và thì**

Ta có

(\*)

Ta có

(\*\*)

Từ (\*) và (\*\*), ta có:

và

( )

Theo định nghĩa Big-O, xét và ,ta có:

* **Nếu và thì**

Ta có

(\*)

Ta có

(\*\*)

Từ (\*) và (\*\*), ta có:

và

Vì và

Theo định nghĩa Big-O, chọn và ,

ta có đccm:

**Bài tập 6:**

* **if ,then**

Ta có

()

Theo định nghĩa Big-, xét và , ta có đccm:

* ***where***

Xét (\*)

Theo định nghĩa Big-θ, ta có (\*\*)

Xét (\*\*\*)

Theo định nghĩa Big-θ, ta có (\*\*\*\*)

Từ (\*),(\*\*),(\*\*\*) và (\*\*\*\*):

Xét (\*)

(\*\*)

Xét

(\*\*\*\*)

Từ(\*),(\*\*),(\*\*\*)và(\*\*\*\*):

**Cách 1:**

Ta có: và

,

Đặtvà N

Theo định nghĩa Big-O, xét và , ta có:

(\*)

Ta có:

)

Đặt

Theo định nghĩa Big-Ω, xét và , ta có:

(\*\*)

Từ (\*) và (\*\*) và theo định nghĩa của Big-, ta có:

**Cách 2:**

Theo định nghĩa của Big-Θ, ta có f(n) = Θ(g(n)) nếu ∃ c1, c2 ∈ R+, ∃ n0 ∈ N sao cho c1g(n) ≤ f(n) ≤ c2g(n) ∀ n ≥ n0.

Áp dụng cho giả thiết trên ta có: c1(f(n) + g(n)) ≤ max{f(n), g(n)} ≤ c2(f(n) + g(n))

Ta có: f(n) + g(n) ≥ max{f(n), g(n)} luôn đúng (1), f(n) ≤ max{f(n), g(n)} và g(n) ≤ max{f(n), g(n)}

* f(n) + g(n) ≤ 2.max{f(n), g(n)}
* ½ (f(n) + g(n)) ≤ max{f(n), g(n)} (2)

Từ (1), (2) => ½ (f(n) + g(n)) ≤ max{f(n), g(n)} ≤ f(n) + g(n)

* max{f(n), g(n)} = Θ(f(n) + g(n)) (với c1=1/2 và c2=1) (điều phải chứng minh)

**Cách 1:**

Ta có:

Giả sử:

Theo định nghĩa Big-O, xét c=1 và , ta có đccm:

(\*)

Giả sử:

Theo định nghĩa Big-, xét và , ta có đccm:

(\*\*)

Từ (\*), (\*\*) và theo định nghĩa Big-, ta có đccm:

Cách 2:

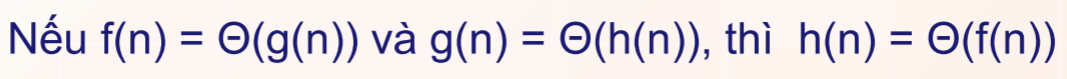
Ta có: log3(n2) = 2log3n ≤ 2log2n ≤ 3log2n = log2(n3) => log3(n2) = O(log2(n3)) ∀ n ≥ n0 và c = 1, n0 = 1

Ta có:

log3(n2) = 2log3n ≥ 2log4n ≥ log2n ≥ 3.1/3.log2n = 1/3.log2(n3) => log3(n2) = Ω(log2(n3)) ∀ n ≥ n0 và c = 1/3, n0 = 1

* log3(n2) = Θlog2(n3) (điều phải chứng minh)

**Bài tập 7:**



Ta có:

(\*)

Ta có:

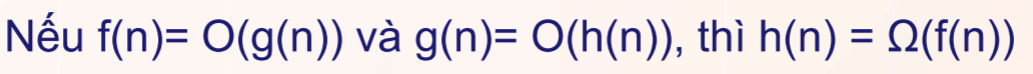
(\*\*)

Từ (\*) và (\*\*), ta có:

Đặt

Theo định nghĩa Big- , ta có:

Nên khẳng định trên là đúng.



Ta có:

(\*)

Ta có:

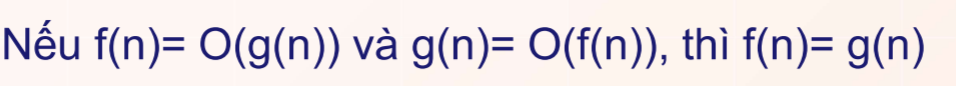
(\*\*)

Từ (\*) và (\*\*), ta có:

Đặt

Theo định nghĩa Big-Ω, ta có:

Khẳng định trên là đúng.



Giả sử và

Dễ thấy rằng thì

Chọn và theo định nghĩa Big-O, ta có được:

(\*)

Xét ,

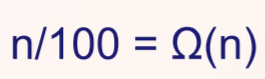
Theo định nghĩa Big-O, xét và ta có:

(\*\*)

Từ (\*) và (\*\*), ta có:

nhưng

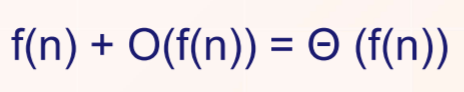
Dù có là độ phức tạp của Big-O của nhau nhưng về mặt bản chất đẳng thức chúng khác nhau. Nên khẳng định trên là sai.



Giả sử và ,

Theo định nghĩa Big- , xét và ta có:

→ Khẳng định trên là đúng.



Xét (\*)

Ta có:

Theo định nghĩa Big-, xét ta có:

Mà từ (\*)



**Cách 1:**

Giả sử

(\*)

Khẳng định trên là sai vì, để chứng minh được thì phải chọn ra 1 hằng số c nhất định nhưng từ (\*) ta có thể thấy c sẽ phụ thuộc vào n mà n có thể biến thiên nên c cũng có thể biến thiên theo nếu chọn như vậy c sẽ không còn là một hằng số.

**Cách 2:**

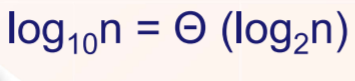
Theo định nghĩa Big-Ω, ta có 210n = Ω (2n) nếu ∃ c ∈ R+, ∃ n0 ∈ N sao cho 210n ≥ c.2n ∀ n ≥ n0.

Theo định nghĩa Big-O, ta có 210n = O(2n) nếu ∃ c ∈ R+, ∃ n0 ∈ N sao cho 210n ≤ c.2n ∀ n ≥ n0.

Gọi f(n) = 210n và g(n) = 2n, ta có:

f(n) ≥ c.g(n) ∀ n ≥ n0

* 210n ≥ c.2n ∀ n ≥ n0
* (2n)10 ≥ c.2n ∀ n ≥ n0
* (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) ≥ c.2n ∀ n ≥ n0
* (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) (2n) ≥ c ∀ n ≥ n0
* log2((2n)10) ≥ log2(c) ∀ n ≥ n0
* 10x ≥ log2(c) ∀ n ≥ n0
* 210n = Ω(2n) => 210n ∉ O(2n) => Khẳng định trên sai



**Cách 1:**

Ta có:

(\*)

Ta có:

(\*\*)

Từ (\*) và (\*\*) và theo định nghĩa Big-, ta chọn: và ta có:

**Cách 2:**

Ta có: log10n = log2n / log210 ≈ log2n/3.32 (1)

Theo định nghĩa Big-Θ, ta có: c1.log2n ≤ log10n ≤ c2.log2n

* c1.log2n ≤ log2n / log210 ≤ c2.log2n (2)

Từ (1), (2), chọn c1=1/2.log210 và c2=1/log210. Theo định nghĩa Big-Θ => log10n = Θ(log2n) => Khẳng định đúng.

**Bài tập 1:**

a) Ý nghĩa của “độ phức tạp” khi đề cập đến thuật toán là “độ phức tạp” là từ dùng chung để nói về bộ ký hiệu/ký pháp tiệm cận: O, Ω, Θ. Không những thế, “độ phức tạp” phân lớp “cấp độ lớn” của hàm T(n) khi n đủ lớn. Giải thuật nào có độ phức tạp ở phân lớp thấp hơn thì hiệu quả hơn.

b) Theo tôi, khi nghiên cứu về các thuật toán thì thường mọi người sẽ quan tâm đến tính hiệu quả về thời gian của chúng nhưng lại quan tâm đến bậc tăng trưởng (order of growth) của hàm thời gian thực hiện của thuật toán hơn là bản thân thời gian thực hiện T(n) vì dù chỉ mang kết quả tương đối và không chính xác tuyệt đối nhưng lại dễ dàng hơn so với việc đo chính xác thời gian thực hiện T(n) và việc đánh giá lại dễ hơn. Ngoài ra, việc đó chính xác cũng rất khó vì chịu nhiều yếu tố khách quan bên ngoài như công cụ và cấu hình của thiết bị đo, …

c) Nói về độ phức tạp tức là đề cập tới các ký hiệu tiệm cận, mà có nhiều ký hiệu khác nhau.

Chúng ta dùng ký hiệu Big-O khi muốn tìm hàm chặn trên của T(n). Big-O được định nghĩa là giới hạn trên và giới hạn trên của một thuật toán là lượng thời gian cần thiết nhiều nhất (để tìm hiệu suất trong trường hợp xấu nhất).

Chúng ta dùng ký hiệu Big-Ω khi muốn tìm hàm chặn dưới của T(n). Big- Ω được định nghĩa là giới hạn dưới và giới hạn dưới của một thuật toán là lượng thời gian cần thiết ít nhất (để tìm hiệu suất trong trường hợp tốt nhất).

Chúng ta dùng ký hiệu Big-Θ khi muốn tìm hàm chặn dưới của T(n). Big-Θ được định nghĩa là ràng buộc chặt chẽ nhất và ràng buộc chặt chẽ nhất là thời điểm tốt nhất trong tất cả các trường hợp xấu nhất mà thuật toán có thể thực hiện.