**Họ và tên:** Nguyễn Phạm Thành Hưng **Mã sinh viên:** 6151071056 **Lớp:** CNTT – K61

**2.1 Khung phân tích**

*Hiệu quả thời gian*, còn được gọi là *độ phức tạp thời gian*, cho biết thuật toán trong câu hỏi chạy nhanh như thế nào. *Hiệu quả không gian* , còn được gọi là *độ phức tạp không gian*, đề cập đến số lượng đơn vị bộ nhớ được yêu cầu bởi thuật toán để bổ sung vào không gian cần thiết cho đầu vào và đầu ra của nó.

## Đo kích thước của đầu vào: Hãy bắt đầu với một quan sát rõ ràng rằng hầu hết các thuật toán chạy dài hơn trên các đầu vào lớn hơn. Việc lựa chọn một số liệu kích thước phù hợp có thể bị ảnh hưởng bởi các hoạt động của thuật toán được đề cập. Một số thuật toán yêu cầu nhiều hơn một tham số để chỉ ra kích thước của đầu vào của chúng.

## Đơn vị đo thời gian chạy: Vấn đề tiếp theo liên quan đến các đơn vị đo thời gian chạy của thuật toán. Một cách tiếp cận có thể là đếm số lần mỗi thao tác của thuật toán được thực thi. Cách tiếp cận này là quá khó và như chúng ta sẽ thấy, thường là không cần thiết. Điều cần làm là xác định hoạt động quan trọng nhất của thuật toán, được gọi là hoạt động cơ bản , hoạt động đóng góp nhiều nhất vào tổng thời gian chạy và tính số lần thao tác cơ bản được thực hiện.

Theo quy định, không khó để xác định hoạt động cơ bản của thuật toán: nó thường là hoạt động tốn nhiều thời gian nhất trong vòng lặp trong cùng của thuật toán. Khung được thiết lập để phân tích hiệu quả thời gian của thuật toán cho thấy việc đo lường nó bằng cách đếm số lần hoạt động cơ bản của thuật toán được thực hiện trên các đầu vào có kích thước *n.* Hãy *c op* là thời gian thực hiện hoạt động cơ bản của một thuật toán trên một máy tính cụ thể, và để cho *C (n)* là số lần này nhu cầu hoạt động được thực hiện cho thuật toán này.

Khung phân tích hiệu quả bỏ qua các hằng số nhân và tập trung vào *thứ tự tăng trưởng của* số đếm trong một bội số không đổi cho các đầu vào kích thước lớn. Ví dụ thời gian chạy thuật toán. giả sử rằng C (n) = 1 /2n (n - 1)

C(n) = 1/2n(n − 1) = 1/2n^2 − 1/2n ≈ 1/2n^2  
T (2n)/T (n) ≈ copC(2n)/copC(n) ≈1/2 \*(2n)2/(1/2n^2) = 4.

**Đơn đặt hàng tăng trưởng:** Tại sao điều này nhấn mạnh vào thứ tự tăng trưởng của số lượng cho kích thước đầu vào lớn? Sự khác biệt về thời gian chạy trên các đầu vào nhỏ không phải là điều thực sự phân biệt các thuật toán hiệu quả với các thuật toán không hiệu quả.   
Trường hợp xấu nhất, Trường hợp tốt nhất và Hiệu quả trường hợp trung bình**:** Trong phần đầu của phần này, chúng tôi đã xác định rằng việc đo lường hiệu quả của thuật toán là một hàm của một tham số cho biết kích thước của đầu vào của thuật toán là hợp lý. Nhưng có nhiều thuật toán mà thời gian chạy không chỉ phụ thuộc vào kích thước đầu vào mà còn phụ thuộc vào chi tiết cụ thể của một đầu vào cụ thể.

Các *hiệu quả trường hợp xấu nhất* của một thuật toán là hiệu quả của nó đối với các trường hợp xấu nhất đầu vào kích thước *n* , mà là một đầu vào (hoặc đầu vào) kích thước *n* mà các thuật toán chạy dài nhất trong số tất cả các đầu vào có thể có của kích thước đó. Nếu không có phân tích trường hợp trung bình, các nhà khoa học máy tính có thể đã bỏ lỡ nhiều thuật toán quan trọng. Một loại hiệu quả khác được gọi là *hiệu quả khấu hao.*

Tóm tắt lại khung phân tích: Cả hiệu quả về thời gian và không gian đều được đo bằng các hàm của kích thước đầu vào của thuật toán. Hiệu quả thời gian được đo bằng cách đếm số lần thao tác cơ bản của thuật toán được thực thi. Hiệu quả không gian được đo bằng cách đếm số lượng đơn vị bộ nhớ phụ được sử dụng bởi thuật toán. Hiệu quả của một số thuật toán có thể khác nhau đáng kể đối với các đầu vào có cùng kích thước. Đối với các thuật toán như vậy, chúng ta cần phân biệt giữa hiệu quả của trường hợp xấu nhất, trường hợp trung bình và trường hợp tốt nhất. Mối quan tâm chính của khung nằm ở thứ tự tăng trưởng thời gian chạy của thuật toán (đơn vị bộ nhớ thêm được tiêu thụ) khi kích thước đầu vào của nó đi đến vô cùng.

### 2.2. Ký hiệu tiệm cận và các lớp hiệu quả cơ bản

Như đã chỉ ra trong phần trước, khung phân tích hiệu quả tập trung vào thứ tự tăng trưởng của một thuật toán tính toán hoạt động cơ bản như là chỉ số chính của hiệu quả của thuật toán. Để so sánh và xếp hạng các thứ tự tăng trưởng như vậy, các nhà khoa học máy tính sử dụng ba ký hiệu: O (big oh), (omega lớn) và (theta lớn). Đầu tiên, chúng tôi giới thiệu các ký hiệu này một cách không chính thức, và sau đó, sau một vài ví dụ, các định nghĩa chính thức được đưa ra. Trong các cuộc thảo luận sau đây, t (n) và g (n) có thể là bất kỳ hàm không âm nào được xác định trên tập hợp các số tự nhiên. Trong bối cảnh chúng ta quan tâm, t (n) sẽ là một thuật toán thời gian chạy (thường được biểu thị bằng số thao tác cơ bản C (n)) và g (n) sẽ là một hàm đơn giản để so sánh số đếm với.

#### Giới thiệu không chính thức

Một cách không chính thức, O (g (n)) là tập hợp của tất cả các hàm có thứ tự tăng trưởng thấp hơn hoặc giống như g (n) (trong phạm vi bội số không đổi, khi n chuyển sang vô cùng).

Thật vậy, hai hàm đầu tiên là tuyến tính và do đó có thứ tự tăng trưởng thấp hơn g (n) = n2, trong khi hàm cuối cùng là bậc hai và do đó có cùng thứ tự tăng trưởng là n2. Mặt khác, n3 ∉ O (n2), 0,00001n3 ∉ O (n2), n4 + n + 1 ∉ O (n2).

Thật vậy, các hàm n3 và 0,00001n3 đều là khối và do đó có thứ tự tăng trưởng cao hơn n2 và do đó, đa thức bậc 4 n4 + n + 1. Ký hiệu thứ hai, (g (n)), là viết tắt của tập hợp tất cả các hàm có thứ tự tăng trưởng cao hơn hoặc bằng với g (n) (trong phạm vi bội số không đổi, khi n chuyển sang vô cùng).Ví dụ, n3 ∈ Ω (n2), 1/2 \* n (n - 1) ∈ Ω (n2), nhưng 100n + 5∉ Ω (n2).

Cuối cùng, (g (n)) là tập hợp tất cả các hàm có cùng thứ tự tăng trưởng là g (n) (trong phạm vi bội số không đổi, khi n chuyển sang vô cùng). Do đó, mọi hàm bậc hai an2 + bn + c với a> 0 đều nằm trong (n2), nhưng cũng vậy, trong số vô số các hàm khác, n2 + sin n và n2 + log n. Hy vọng rằng, giới thiệu không chính thức này đã làm cho bạn thoải mái với ý tưởng đằng sau ba ký hiệu tiệm cận. Vì vậy, bây giờ đến các định nghĩa chính thức.

#### Ký hiệu O

ĐỊNH NGHĨA: Hàm t (n) được gọi là O (g (n)), ký hiệu là t (n) ∈ O (g (n)), nếu t (n) được giới hạn ở trên bởi một bội số không đổi của g (n ) cho tất cả n lớn, nghĩa là, nếu tồn tại một số hằng số dương c và một số nguyên không âm n0 sao cho t (n) ≤cg (n) với mọi n ≥ n0. Vì mục đích rõ ràng trực quan, n là mở rộng để trở thành một con số thực sự Ví dụ, chúng ta hãy chính thức chứng minh một trong những khẳng định được đưa ra trong giới thiệu: 100n + 5 O (n2). Thật, 100n + 5 ≤ 100n + n (với mọi n ≥ 5) = 101n ≤ 101n2. Do đó, như các giá trị của hằng số c và n0 theo định nghĩa, chúng ta có thể lấy 101 và 5, tương ứng. Lưu ý rằng định nghĩa cho chúng ta rất nhiều tự do trong việc lựa chọn các giá trị cụ thể cho hằng số c và n0. Ví dụ, chúng ta cũng có thể lý do rằng 100n + 5 ≤ 100n + 5n (với mọi n ≥ 1) = 105n để hoàn thành bằng chứng với c = 105 và n0 = 1.

#### Ký hiệu -Ω

ĐỊNH NGHĨA: Một hàm t (n) được gọi là Ω(g (n)), ký hiệu là t (n) ∈ Ω (g (n)), nếu t (n) được giới hạn dưới một số bội số dương của g (n) cho tất cả n lớn, tức là, nếu tồn tại một số hằng số dương c và một số nguyên không âm n0 như vậy đó t (n) ≥ cg (n) với mọi n ≥ n0. Dưới đây là một ví dụ về bằng chứng chính thức rằng n3 (n2):

n3 ≥ n2 với mọi n ≥ 0, tức là, chúng ta có thể chọn c = 1 và n0 = 0.

# chú thích

ĐỊNH NGHĨA Một hàm t (n) được cho là nằm trong (g (n)), ký hiệu là t (n) (g (n)), nếu t (n) được giới hạn ở cả trên và dưới bởi một số bội số dương của g (n) cho tất cả n lớn, tức là, nếu tồn tại một số hằng số dương c1 và c2 và một số số nguyên không âm n0 sao cho c2g (n) ≤ t (n) ≤ c1g (n) với mọi n ≥ n0.

Tài sản hữu ích có liên quan đến các ký hiệu tiệm cận.  
Sử dụng các định nghĩa chính thức của các ký hiệu tiệm cận, chúng ta có thể chứng minh tính chất chung .Đặc tính sau đây, đặc biệt, rất hữu ích trong việc phân tích các thuật toán bao gồm hai phần thực hiện liên tiếp.

ĐỊNH LÝ : If t1(n) ∈ O(g1(n)) and t2(n) ∈ O(g2(n)), then t1(n) + t2(n) ∈ O(max{g1(n), g2(n)}). (Các xác nhận tương tự cũng đúng cho các ký hiệu và ký hiệu.)

BÀI TOÁN Bằng chứng mở rộng cho các đơn đặt hàng tăng trưởng thực tế đơn giản sau đây về bốn số thực tùy ý a1, b1, a2, b2: nếu a1 ≤ b1 và a2 b2, thì a1 + a2 ,2 tối đa {b1, b2}. Vì t1 (n) O (g1 (n)), tồn tại một số hằng số dương c1 và một số không số nguyên âm n1 sao cho t1 (n) c1g1 (n) với mọi n ≥ n1. Tương tự, vì t2 (n) O (g2 (n)), t2 (n) c2g2 (n) với mọi n ≥ n2. Để tôi biểu thị c3 = max {c1, c2} và xem xét n ≥ max {n1, n2} để chúng có thể sử dụng cả hai bất đẳng thức. Thêm chúng mang lại những điều sau đây:

t1 (n) + t2 (n) ≤ c1g1 (n) + c2g2 (n)≤ c3g1 (n) + c3g2 (n) = c3 [g1 (n) + g2 (n)] ≤ c32 tối đa {g1 (n), g2 (n)}.

Do đó, t1 (n) + t2 (n) O (max {g1 (n), g2 (n)}), với các hằng số c và n0 cần thiết theo định nghĩa O lần lượt là 2c3 = 2 max {c1, c2} và max {n1, n2}. Vì vậy, tính chất này ngụ ý gì cho một thuật toán bao gồm hai mục đích phần thực hiện hoàn toàn? Nó ngụ ý rằng hiệu quả tổng thể của thuật toán là yếu tố ngăn chặn được khai thác bởi một phần với thứ tự tăng trưởng cao hơn, tức là phần kém hiệu quả nhất của nó: t1 (n) ∈ O (g1 (n)) t2 (n) ∈ O (g2 (n)) t1 (n) + t2 (n) O (tối đa {g1 (n), g2 (n)}). Sử dụng giới hạn để so sánh các đơn đặt hàng tăng trưởng Ba trường hợp chính có thể phát sinh:

Lưu ý rằng hai trường hợp đầu có nghĩa là t (n) O (g (n)), hai trường hợp cuối có nghĩa là t (n) ∈ (g (n)), và trường hợp thứ hai có nghĩa là t (n) (g (n)). Cách tiếp cận dựa trên giới hạn thường thuận tiện hơn phương pháp dựa trên các định nghĩa bởi vì nó có thể tận dụng các kỹ thuật tính toán mạnh mẽ được phát triển cho các giới hạn tính toán, chẳng hạn như quy tắc L’Hopital’s rule Stirling’s formula . Mặc dù khung phân tích hiệu quả tập hợp tất cả các chức năng có thứ tự tăng trưởng khác nhau bởi một bội số không đổi, vẫn còn vô số những lớp học như vậy. (Ví dụ: các hàm số mũ có các thứ tự khác nhau của tăng trưởng cho các giá trị khác nhau của cơ sở a.) Do đó, có thể gây ngạc nhiên rằng hiệu quả thời gian của một số lượng lớn các thuật toán chỉ rơi vào một vài lớp. Theo quy định, bạn nên mong đợi một thuật toán từ lớp hiệu quả tiệm cận tốt hơn để vượt trội hơn một thuật toán từ một lớp kém hơn ngay cả đối với các đầu vào có kích thước vừa phải.Quan sát này đặc biệt đúng đối với một thuật toán có hàm mũ tốt hơn theo cấp số nhân thời gian chạy so với thuật toán theo cấp số nhân (hoặc tệ hơn).