1. Độ phức tạp thuật toán

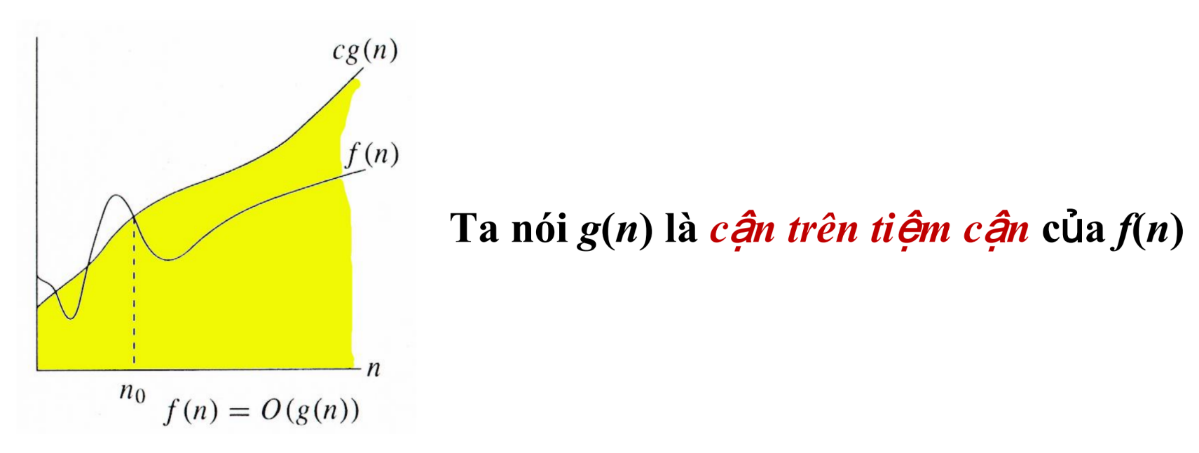
* Thời gian chạy:
* Thực tế: Phụ thuộc input / phụ thuộc ngôn ngữ lập trình / phụ thuộc máy tính cụ thể
* Ước lượng với phép toán(Lí thuyết): “Bất biến” - không phụ thuộc input cụ thể.

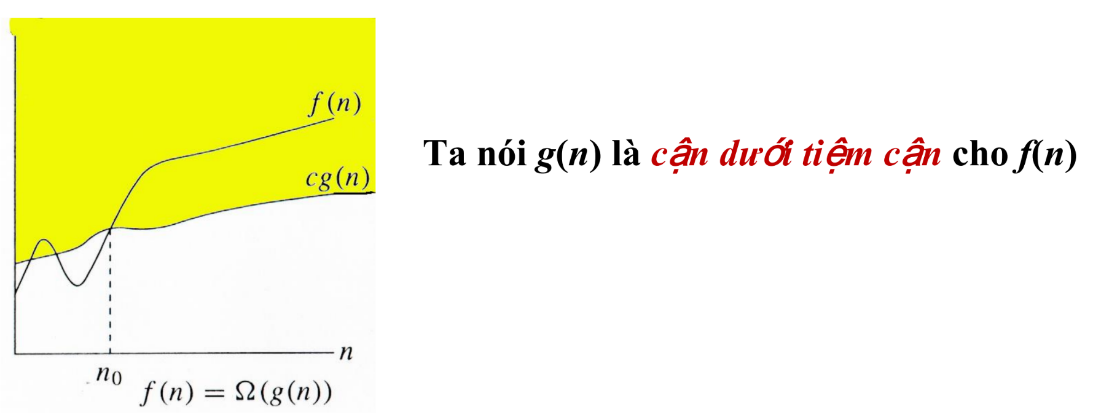
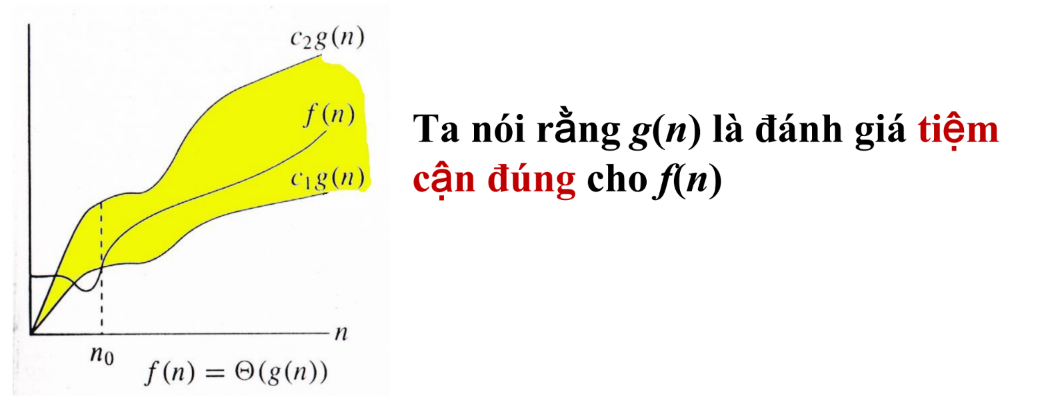
1. Một số quy ước

* Gán, đọc, ghi, so sánh, … độ phức tạp = const
* Độ phức tạp:
* Một đoạn lệnh: tổng độ phức tạp của từng lệnh.
* Cấu trúc rẽ nhánh(if else, swtich): Lệnh phức tạp nhất trong khối if else, swtich
* Thời gian thực hiện vòng lặp: tổng thời gian tất cả các lần lặp của thân vòng lặp.
* Từ đó đếm/ước lượng độ phức tạp

1. Kí hiệu

* Big-O: Cận trên của số phép toán (có thể hiểu là xét tình huống xấu nhất có thể xảy ra)



* Big-: Cận dưới của số phép toán (có thể hiểu là xét tình huống tốt nhất có thể xảy ra)
* Big-: Xấp xỉ chặt trên và dưới của số phép toán

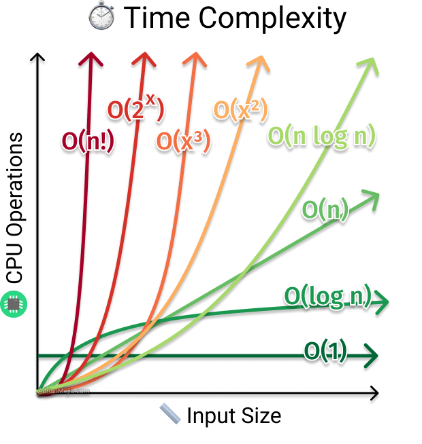
1. Chứng minh độ phức tạp

* Khi chứng minh các công thức tiệm cận ta chỉ cần tìm các hằng số c và nào đó thỏa mãn bất đẳng thức trong định nghĩa công thức tiệm cận.
* Ví dụ: Xác định các hằng số trong các định nghĩa để chứng minh các đánh giá tiệm cận sau:
* = 1

1. Tính độ phức tạp(đanh giá thời gian chạy)

* Quy tắc cơ bản:
* Nhân hằng số:
* Cộng:
* Nhân:
* Ví dụ: Độ phức tạp của thuật toán là gì?

­

1. Độ tăng của một số hàm

– O(1): hằng số (constant)

– O(log n): logarithmic

– O(n): tuyến tính (linear)

– O(n log n): trên tuyến tính (superlinear)

– O(): bình phương (quadratic)

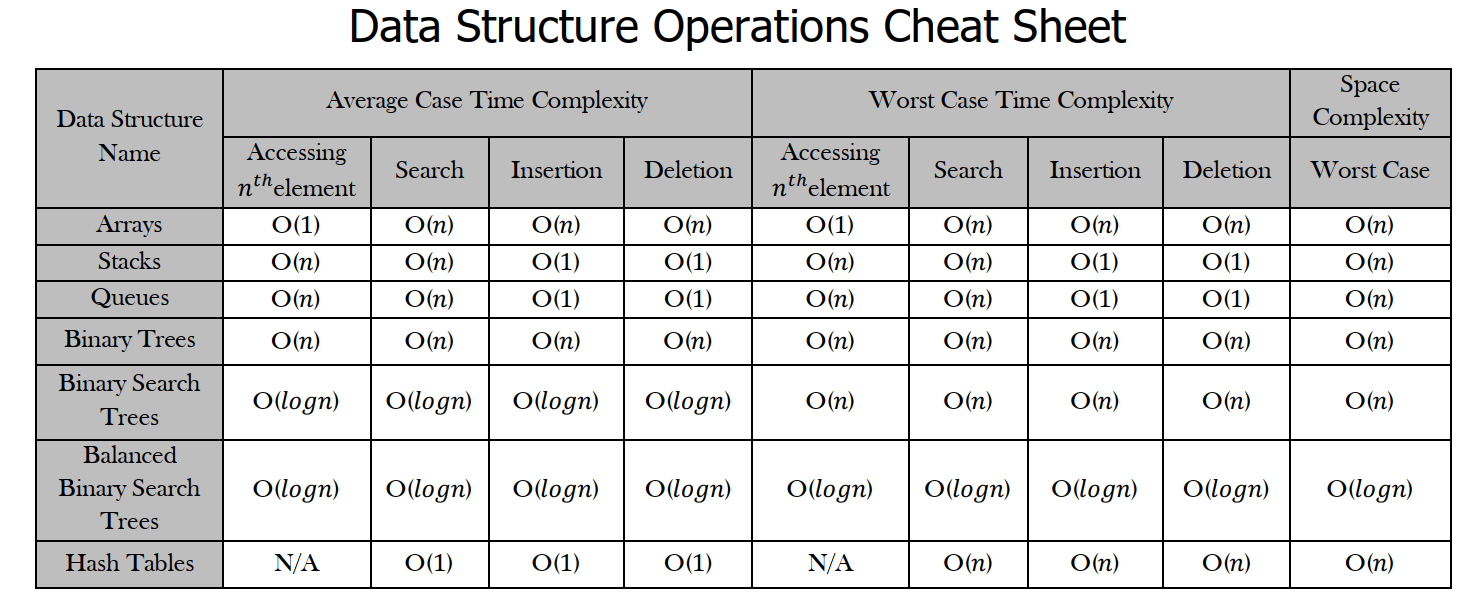
– O(): bậc ba (cubic)

– O(): hàm mũ (exponential ) (a > 1)

– O(n!): Giai thừa (Factorial)

1. Định lý thợ

* Dùng để giúp tính toán độ phức tạp của một số thuật toán đệ quy/chia để trị.

1. Bảng tổng kết độ phức tạp của các cấu trúc dữ liệu thường dùng
2. Mảng, Danh Sách liên kết (Linked List: Singly Linked Lists, Double Linked List, Circular Linked Lists, Linked lists với mảng và mảng động)
3. Array
4. Stack
5. Queues
6. Priority Queue
7. Tree(Binary Tree, Binary search Tree, AVL)
8. Binary Tree

- Mỗi nốt có nhiều nhất 2 nút con.

- Biểu diễn:

+ Mảng:

* + Nếu gốc tại vị trí 0, các nút x có:
    - Con trái ở vị trí 2x + 1.
    - Con phải ở vị trí 2x + 2
  + Gốc tại vị trí 1, các nút x có:
    - Con trái ở vị trí 2x
    - Con phải ở vị trí 2x + 1

+ Danh sách liên kết:

* + Mỗi node gồm có dữ liệu và con trỏ trỏ đến node con trái và node con phải.
  + Node con của một node nhận giá trị null nếu node con đó không liên kết tới node nào nữa.
  + Pseudo code:

Node {

Data;

Node left;

Node right;

}

* + Minh họa:

