1. Đánh giá hiệu năng thuật toán

* Khái niệm

Thời gian mà [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh" \o "Máy tính) khi thực hiện một [thuật toán](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n) không chỉ phụ thuộc vào bản thân thuật toán đó, ngoài ra còn tùy thuộc từng máy tính. Để đánh giá hiệu quả của một thuật toán, có thể xét số các phép tính phải thực hiện khi thực hiện thuật toán này. Thông thường số các phép tính được thực hiện phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp thuật toán là một hàm phụ thuộc đầu vào

* Xác định độ phức tập của thuật toán

•         Độ phức tạp tính toán của giải thuật: O(f(n))

•         Việc xác định độ phức tạp tính toán của giải thuật trong thực tế có thể tính bằng một số quy tắc đơn giản sau:

–        Quy tắc bỏ hằng số:

            T(n) = O(c.f(n)) = O(f(n)) với c là một hằng số dương

–        Quy tắc lấy max:

            T(n) = O(f(n)+ g(n)) = O(max(f(n), g(n)))

–         Quy tắc cộng:

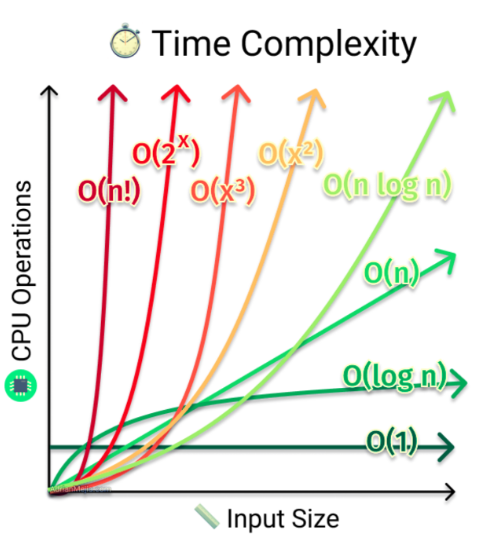
            T1(n) = O(f(n))                     T2(n) = O(g(n))

            T1(n) + T2(n) = O(f(n) + g(n))

–         Quy tắc nhân:

            Đoạn chương trình có thời gian thực hiện T(n)=O(f(n))

            Nếu thực hiện k(n) lần đoạn chương trình với k(n) = O(g(n)) thì độ phức tạp sẽ là O(g(n).f(n))



* **Độ phức tạp thời gian**
* Định nghĩa: Độ phức tạp thời gian của thuật toán là thước đo số lượng phép tính tối đa mà thuật toán cần thực hiện để giải quyết một bài toán với bộ dữ liệu đầu vào có kích thước n.

 Độ phức tạp thời gian thường được biểu diễn bằng ký hiệu O(f(n)), trong đó f(n) là một hàm số thực không âm, nghĩa là thời gian thực hiện của thuật toán sẽ tỉ lệ thuận với f(n) khi kích thước dữ liệu đầu vào n tăng lên.

* Để đánh giá thời gian thực hiện thuật toán, ta xuất phát từ các lệnh đơn trong chương trình, rồi tới các câu lệnh có cấu trúc, các khối lệnh phức tạp hơn, sau đó hợp lại thành thời gian thực hiện cả chương trình. Cụ thể ta có các quy tắc:
* Các lệnh đơn (lệnh khai báo, gán, nhập xuất dữ liệu, phép toán số học,...): Thời gian O(1).
* Các khối lệnh: Giả sử một khối lệnh gồm các câu lệnh S1, S2 ,..., Sm có thời gian thực hiện lần lượt là (f1(n)), O(f2(n)) ,..., O(fm(n)) thì thời gian thực hiện của cả khối lệnh là: O(max(f1(n),f2(n),...,fm(n))).
* Câu lệnh rẽ nhánh: Ta có cú pháp lệnh rẽ nhánh là:   
   Giả sử thời gian thực hiện của câu lệnh 1 và câu lệnh 2 lần lượt là O(f1(n)) và O(f2(n)) thì thời gian thực hiện lệnh rẽ nhánh là: O(max(f1(n), f2​(n))).
* Câu lệnh lặp: Giả sử thời gian thực hiện phần thân của lệnh lặp là O(f1(n)) và số lần lặp tối đa của vòng lặp là f2(n) thì thời gian thực hiện của cả vòng lặp là O(f1(n).f2(n)). Điều này áp dụng cho tất cả các vòng lặp for, while và do...while.
* Sau khi đánh giá được thời gian thực hiện của tất cả các câu lệnh trong chương trình, thời gian thực hiện của toàn bộ chương trình sẽ là thời gian thực hiện của câu lệnh có thời gian thực hiện lớn nhất. Ngoài ra, nếu như độ phức tạp tính toán là O(c×f(n)) với c là một hằng số nhỏ, ta có thể bỏ qua c và coi như thuật toán có độ phức tạp là O(f(n)) - chẳng hạn như O(3n), O(4n) có thể coi như O(n).

* **Độ phức tạp không gian**
* Định nghĩa: Độ phức tạp không gian của thuật toán là thước đo dung lượng bộ nhớ tối đa mà thuật toán cần sử dụng để lưu trữ dữ liệu và thực hiện phép tính với bộ dữ liệu đầu vào có kích thước n.

Độ phức tạp không gian cũng được biểu diễn bằng ký hiệu O(g(n)), trong đó g(n) là một hàm số thực không âm, nghĩa là dung lượng bộ nhớ mà thuật toán sử dụng sẽ tỉ lệ thuận với g(n) khi kích thước dữ liệu đầu vào n tăng lên.

* Độ phức tạp không gian của một thuật toán sẽ được tính toán thông qua hàm O(g(n)) trước, rồi mới đổi ra giá trị dung lượng cụ thể. Nó là tổng của tất cả bộ nhớ sử dụng trong việc nhập dữ liệu đầu vào và bộ nhớ phụ sử dụng khi thực hiện thuật toán. Các quy tắc tính toán cơ bản như sau:
  + Các biến đơn khi khai báo (một hoặc nhiều biến): O(1).
  + Khai báo mảng một chiều kích thước n: O(n).
  + Khai báo mảng nhiều chiều có kích thước các chiều lần lượt là *n*1,*n*2,…,*nk* : *O*(*n*1×*n*2×⋯×*nk*).
  + Lời gọi đệ quy: Phụ thuộc vào số lượng lời gọi đệ quy lưu đồng thời trong phân vùng bộ nhớ call stack (sẽ học ở bài Hàm đệ quy).

 Tổng bộ nhớ sử dụng trong toàn bộ chương trình sẽ hợp thành độ phức tạp không gian của chương trình là O(g(n)). Sau khi tính được, O(g(n)), ta sẽ quy đổi nó ra dung lượng bộ nhớ tương ứng với kiểu dữ liệu của input để tính ra được bộ nhớ sử dụng một cách tương đối chính xác.