

Big O

Big O Notation là một quy ước toán học được các lập trình viên toàn thế giới sử dụng để mô tả hiệu năng của thuật toán một cách đồng nhất.

Big O là gì?

Big O mô tả **tốc độ tăng** của thời gian và không gian mà một thuật toán yêu cầu trong **trường hợp xấu nhất**, khi kích thước đầu vào n tiến đến vô cùng.

Tốc độ tăng: Big O không cho bạn biết thuật toán chạy chính xác trong thời gian bao lâu. Nó cho bạn biết quy luật mà thời gian chạy sẽ tăng lên.

Trường hợp xấu nhất: Khi phân tích thuật toán, chúng ta thường quan tâm đến kịch bản tệ nhất có thể xảy ra. Vì nó cho chúng ta một sự đảm bảo. Nếu chúng ta biết thuật toán của mình chạy không quá x giây trong trường hợp xấu nhất chúng ta có thể yên tâm rằng nó luôn đáp ứng được trong mọi tình huống thực tế.

Các loại độ phức tạp phổ biến

$O(1)$ - Constant

Thời gian thực thi không phụ thuộc vào kích thước đầu vào n .

Trường hợp thường gặp:

- Truy cập một phần tử trong mảng bằng index
- Lấy một giá trị trong Dictionary thông qua key
- Thêm, xóa một phần tử ở cuối List

$O(\log n)$ - Logarithmic

Thời gian thực thi tăng rất chậm khi n tăng. Khi gấp đôi n , thời gian chỉ tăng thêm một lượng rất nhỏ.

Trường hợp thường gặp:

- Thuật toán tìm kiếm nhị phân trên tập dữ liệu đã sắp xếp

$O(n)$ - Linear

Thời gian thực thi tỷ lệ thuận với kích thước đầu vào n . Nếu n tăng gấp đôi thì thời gian cũng tăng gấp đôi

Trường hợp thường gặp:

- Duyệt qua tất cả các phần tử để tìm một giá trị
- In ra tất cả các phần tử trong mảng

$O(n \log n)$ - Linearithmic

Hiệu quả hơn $O(n^2)$ nhưng kém hiệu quả hơn $O(n)$. Đây là tiêu chuẩn vàng cho các thuật toán sắp xếp dựa trên so sánh.

$O(n \log n)$ tương đương với việc thực hiện một thao tác $O(\log n)$ với mỗi phần tử.

Trường hợp thường gặp:

- Các thuật toán sắp xếp như merge sort, quick sort.
- IntroSort được sử dụng bởi `List<T>.Sort()` trong C#

$O(n^2)$ - Quadratic

Thời gian thực thi tăng theo bình phương của n . Nếu n tăng 10 lần thì thời gian/không gian tăng 100 lần

Trường hợp thường gặp:

- Các vòng lặp lồng nhau
- Các thuật toán Bubble Sort, Selection Sort

$O(2^n)$ - Exponential và $O(n!)$ Factorial

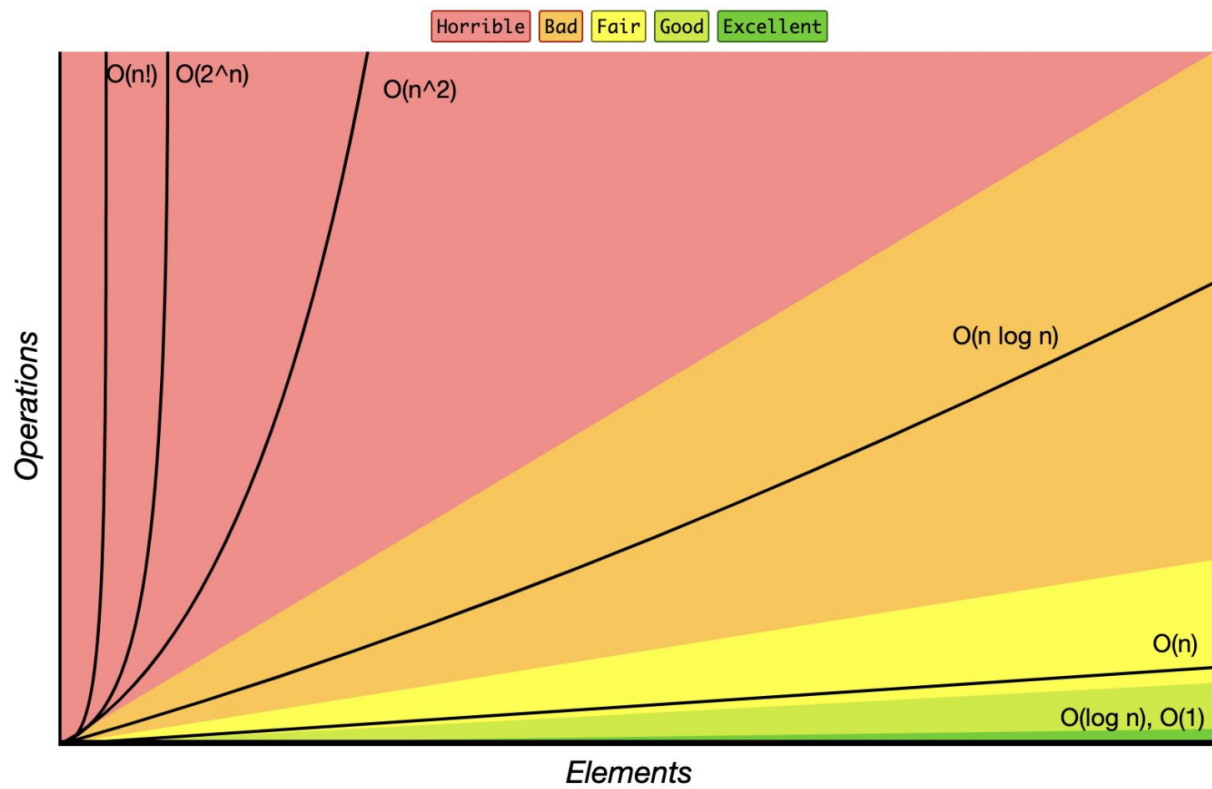
Cực kỳ chậm và nhanh chóng trở nên bất khả thi ngay cả với n tương đối nhỏ

Trường hợp thường gặp:

- $O(2^n)$ Tính số Fibonacci bằng đệ quy đơn giản
- $O(n!)$ Giải quyết bài toán Traveling Salesman Problem bằng cách vét cạn

Xếp hạng trực quan

Big-O Complexity Chart



Revision #1

Created 2025-07-13 13:46:03 UTC by ThanhDV

Updated 2025-07-13 14:20:12 UTC by ThanhDV