Hành trình chinh phục Big O

Thước đo hiệu năng thuật toán

Đọc Sách - Learning How to Learn

Ngày 16 tháng 1 năm 2026

Chào mừng bạn!

Triết lý học tập

Chúng ta sẽ không học Big O như một mớ công thức toán học khô khan. Chúng ta sẽ coi nó như một **"thước đo hiệu năng"** để đánh giá xem thuật toán của bạn ngốn bao nhiêu tài nguyên khi dữ liệu lớn dần lên.

Lộ trình khám phá Big O

1. **Chương 1:** Big O là gì? – Phép ẩn dụ về Đội quân vận chuyển.
2. **Chương 2:** $O(1)$ – Tốc độ của một cái búng tay.
3. **Chương 3:** $O(n)$ – Cuộc đi bộ đường dài.
4. **Chương 4:** $O(n^2)$ – Thảm họa vòng lặp lồng nhau.
5. **Chương 5:** $O(\log n)$ – Phép màu của việc chia đôi.
6. **Chương 6:** Tổng kết và Cách tính nhanh Big O.

Chương 1: Big O là gì? (Bức tranh toàn cảnh)

Ẩn dụ: Đội quân Robot

Big O không cho bạn biết chính xác số giây. Nó cho bạn biết **"tốc độ tăng trưởng"** của thời gian khi số lượng hàng hóa (n) tăng lên.

Ví dụ: Truy cập mảng $O(1)$

```
1 int lay_phan_tu(int mang[], int index) {
2     return mang[index]; // May tinh chi ton dung 1 buoc
3 }
```

Kiểm tra sự hiểu biết

Nếu thuật toán chạy mất 10 giây với 100 con số, và 100 giây với 1.000 con số. Bạn thấy thời gian tăng có tỉ lệ thuận không?

Kiểm tra sự hiểu biết

Nếu thuật toán chạy mất 10 giây với 100 con số, và 100 giây với 1.000 con số. Bạn thấy thời gian tăng có tỉ lệ thuận không?

Đáp án

Có! Khi dữ liệu tăng 10 lần mà thời gian cũng tăng 10 lần, đó chính là đặc điểm của **Tuyến tính** $O(n)$.

Chương 2: $O(1)$ – Tốc độ hằng số

Ẩn dụ: Siêu năng lực dịch chuyển

Dù điểm đến xa 1 mét hay 1.000 cây số, bạn chỉ cần búng tay là tới nơi. Khoảng cách (n) không làm bạn chậm đi.

Code minh họa

```
1 void kiem_tra_so_chan(int n) {
2     if (n % 2 == 0) {
3         cout << "Day la so chan";
4     } else {
5         cout << "Day la so le";
6     }
7 }
```

Chương 3: $O(n)$ – Cuộc đi bộ đường dài

Ẩn dụ: Quét dọn hành lang

Mỗi mét hành lang là một phần tử (n). Bạn càng có nhiều mét hành lang, bạn càng tốn nhiều sức theo đúng tỉ lệ đó.

Code minh họa

```
1 void in_day_so(int n) {
2     for (int i = 1; i <= n; i++) {
3         cout << i << " ";
4     }
5 }
```

Kiểm tra nhanh (Quiz)

Tình huống

Bạn tìm bạn "An"trong danh sách n học sinh bằng cách hỏi từng người từ đầu đến cuối.

1. Đây là $O(1)$ hay $O(n)$?

Kiểm tra nhanh (Quiz)

Tình huống

Bạn tìm bạn "An"trong danh sách n học sinh bằng cách hỏi từng người từ đầu đến cuối.

1. Đây là $O(1)$ hay $O(n)$? $\rightarrow O(n)$.
2. Nếu lớp có 400 bạn, bạn hỏi tối đa bao nhiêu lần?

Kiểm tra nhanh (Quiz)

Tình huống

Bạn tìm bạn "An"trong danh sách n học sinh bằng cách hỏi từng người từ đầu đến cuối.

1. Đây là $O(1)$ hay $O(n)$? $\rightarrow$ $O(n)$.
2. Nếu lớp có 400 bạn, bạn hỏi tối đa bao nhiêu lần? $\rightarrow$ **400 lần**.

Chương 4: $O(n^2)$ – Thảm họa vòng lặp lồng nhau

Ẩn dụ: Diện tích hình vuông

Nếu bạn tăng cạnh hình vuông lên gấp đôi, diện tích thực tế (công việc) tăng lên gấp **bốn** lần!

Code minh họa

```
1 void in_hinh_vuong_sao(int n) {
2     for (int i = 0; i < n; i++) {
3         for (int j = 0; j < n; j++) {
4             cout << "* ";
5         }
6         cout << endl;
7     }
8 }
```

Chương 5: $O(\log n)$ – Phép màu chia đôi

Ẩn dụ: Cuốn từ điển

Mỗi bước bạn mở trang giữa, loại bỏ một nửa số trang không cần thiết. Thanh chocolate khổng lồ biến mất cực nhanh vì bạn liên tục bẻ đôi nó.

Tìm kiếm nhị phân

```
int tim_kiem_nhi_phan(int mang[], int n, int x) {
    int trai = 0, phai = n - 1;
    while (trai <= phai) {
        int giua = (trai + phai) / 2;
        if (mang[giua] == x) return giua;
        if (mang[giua] < x) trai = giua + 1;
        else phai = giua - 1;
    }
    return -1;
}
```

Chương 6: Cách "nhìn code đoán Big O"

Quy tắc nhìn nhanh

- **Không có vòng lặp phụ thuộc** n: $O(1)$.
- **1 vòng lặp từ** 0 **đến** n: $O(n)$.
- **2 vòng lặp lồng nhau**: $O(n^2)$.
- **Biến** i **tăng gấp đôi/giảm một nửa**: $O(\log n)$.

Quy tắc Bảng xếp hạng

Trong Big O, chỉ quan tâm kẻ mạnh nhất (chạy chậm nhất):
$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$.

Bí kíp cho thi lập trình (Competitive Programming)

Khi đọc giới hạn n, hãy ước lượng thuật toán:

- $n \leq 500$: Có thể dùng $O(n^3)$.
- $n \leq 5000$: Có thể dùng $O(n^2)$.
- $n \leq 10^5$ hoặc 10^6: Phải dùng $O(n \log n)$ hoặc $O(n)$.
- $n \geq 10^9$: Chỉ có thể dùng $O(\log n)$ hoặc $O(1)$.

Chương Đặc Biệt: Phép Toán Log

Định nghĩa cho học sinh

$\log_2(n)$ là: **"Số lần bạn cần chia n cho 2 để kết quả bằng 1"**.

Code minh họa tư duy Log

```
int dem_so_lan_chia_doi(int n) {
    int count = 0;
    while (n > 1) {
        n = n / 2;
        count++;
    }
    return count;
}
```

Thử thách cuối cùng

Nếu bạn viết một chương trình có **3 vòng lặp for lồng nhau**, Big O là gì và nó tốt hay tệ?

Thử thách cuối cùng

Nếu bạn viết một chương trình có **3 vòng lặp for lồng nhau**, Big O là gì và nó tốt hay tệ?

Đáp án

Đó là $O(n^3)$. Đây là một thuật toán "nặng nề". Nếu $n = 1000$, bạn sẽ có **1 tỷ** phép tính, máy tính có thể mất 10 giây để xử lý!

Chúc bạn học tốt!

Bạn có muốn thực hành "hô biến"code từ $O(n^2)$ sang $O(n \log n)$ không?