

Data Structure & Algorithm

Big O

(Độ phức tạp của thuật toán)

1. Định nghĩa

Làm thế nào để **đánh giá** 1 chương trình có **tốt / hiệu quả** hay không?

- **Bộ nhớ | Space complexity**
- **Thời gian | Time complexity**

2. Bộ nhớ | Space Complexity

Là **bộ nhớ** mà chương trình / thuật toán cần **sử dụng** khi **thực thi / run** chương trình đó.

- **$O(1)$** : Nếu chỉ sử dụng 1 bộ nhớ **cố định**.
- **$O(n)$, $O(n^2)$, ...**: Thay đổi theo **input** của bài toán.

Mục tiêu:

Tìm được một phương án tối ưu (**bộ nhớ**) cho một vấn đề nào đó.

2. Bộ nhớ | Space Complexity



Daily LeetCode Challenge [11]: 283. Move Zeroes. (#array)

The Brown Box • 139 lượt xem • 2 tháng trước

[DailyLeetCodeChallenge][11]: 283. Move Zeroes. (#array) Id: 283 Name: Move Zeroes. Url: <https://leetcode.com/problems/move-zeroes/> Tags: #array Source Code: <https://bit.ly/2XLxLZT> Slide:

Problem: Given an array **nums**, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Input: [0,1,0,3,12]

Output: [1,3,12,0,0]

2. Bộ nhớ | Space Complexity

Cách 1: Sử dụng thêm 1 mảng T để lưu các giá trị khác 0, rồi ghi lại vào mảng **nums**.

```
int n = nums.length;  
int[] T = new int[n];  
int iT = 0;  
for (int i = 0; i < n; i++) {  
    if(nums[i] != 0){  
        T[iT++] = nums[i];  
    }  
}
```

$O(n)$

2. Bộ nhớ | Space Complexity

Cách 2: Xử lý trực tiếp trên mảng `nums`.

```
int n = a.length;  
int curIndex = 0;  
for (int i = 0; i < n; i++)  
{  
    if(a[i] != 0) {  
        a[curIndex++] = a[i];  
    }  
}
```

$O(1)$


2. Bộ nhớ | Space Complexity

Giải quyết vấn đề về bộ nhớ:

- Tối ưu cách làm / phương pháp.
- Nâng cấp bộ nhớ. (*)

2. Bộ nhớ | Space Complexity


Các bài luyện tập **tối ưu bộ nhớ: #array**




The Brown Box
2.05K subscribers


SUBSCRIBE

HOMEVIDEOSPLAYLISTSCOMMUNITYCHANNELSABOUT

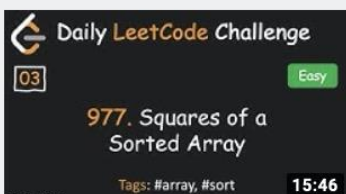
 **#array**




Cấu trúc dữ liệu & Giải thuật [01]: Array - Mảng. #array
The Brown Box • 1.4K views • 3 months ago
===== Data Structure: Array ===== +) Source Code:
https://github.com/thebrownbox/data_structure_algorithm/tree/master/01_Array +) Video về C++ STL



Daily LeetCode Challenge [05]: 88. Merge Sorted Array. (#array)
The Brown Box • 255 views • 3 months ago
[DailyLeetCodeChallenge][05]: 88. Merge Sorted Array. (#array) Id: 88 Name: Merge Sorted Array Url:
<https://leetcode.com/problems/merge-sorted-array/> Tags: #array Source Code: <https://bit.ly/3euPv...>



Daily LeetCode Challenge [03]: 977. Squares of a Sorted Array. (#array, #sort)
The Brown Box • 354 views • 3 months ago
Id: 977 Name: Squares of a Sorted Array Url: <https://leetcode.com/problems/squares-of-a-sorted-array/>
Tags: #array, #sort Source Code: <https://bit.ly/2XLUNjt> Slide: <https://bit.ly/2XwHUCJ> =====...



[DailyLeetCodeChallenge][08]: 1346. Check If N and Its Double Exist. (#array)
The Brown Box • 236 views • 3 months ago

2. Thời gian | Time Complexity

2. Thời gian | Time Complexity

Bài toán Fibonacci.



Daily LeetCode Challenge [35]: 509. Fibonacci Number (#recursion)

The Brown Box • 24 lượt xem • 6 ngày trước

[DailyLeetCodeChallenge][35]: 509. Fibonacci Number (#recursion) Url:

<https://leetcode.com/problems/fibonacci-number/> Tags: #recursion Source Code:

$$F_n = F_{n-1} + F_{n-2}$$

2. Thời gian | Time Complexity

Bài toán Fibonacci.

Cách 1	Cách 2
<pre>public int fib1(int n) { if(n <= 1) return n; return fib1(n-1) + fib1(n-2); }</pre>	<pre>public int fib2(int n) { int[] F = new int[31]; F[0] = 0; F[1] = 1; for (int i = 2; i <= n; i++) { F[i] = F[i-1] + F[i-2]; } return F[n]; }</pre>

>> Thực nghiệm...

2. Thời gian | Time Complexity

- Làm sao để biết 1 chương trình chạy **nhANH** hay chạy **chẬM**?
- Làm sao để **miêu tả mức độ** nhanh/chậm của một chương trình?

>> Big O notation! <<

3. Big O notation

- **Định nghĩa: Độ phức tạp (về thời gian)**
Là **tổng thời gian / số phép tính** toán mà chương trình cần để thực thi chương trình.

- **Example:**

```
for (int i = 0; i < 10; i++) {  
    int t = 2*i;  
    System.out.println(" 2 * " + i + " = " + t);  
}
```

3. Big O notation

```
for (int i = 0; i < 10; i++) {  
    int t = 2*i;  
    System.out.println(" 2 * " + i + " = " + t);  
}
```

i = 0: 1
i < 10: 10
i ++: 10
2 * i: 10
t = : 10
Print: 10 * 4

=====

Sum: $81 = 8 * 10 + 1$

3. Big O notation

```
for (int i = 0; i < n; i++) {  
    int t = 2*i;  
    System.out.println(" 2 * " + i + " = " + t);  
}
```

i = 0: **1**

i < n: **n**

i++: **n**

2*i: **n**

t = : **n**

Print: **n** * **4**

=====

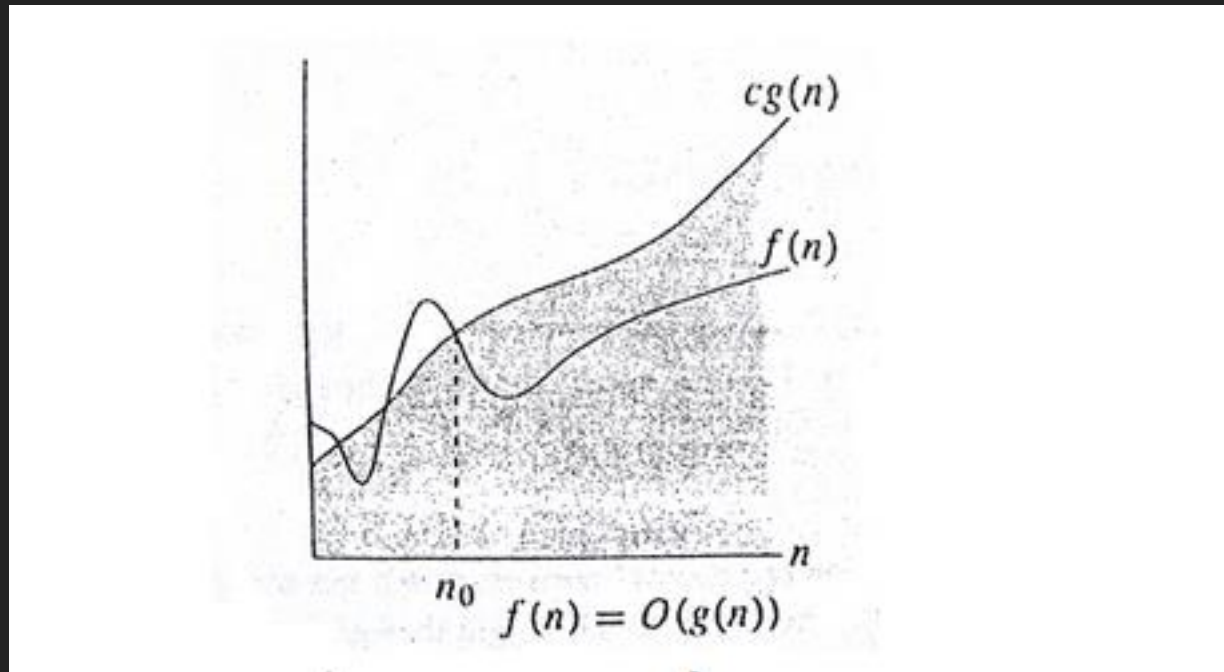
Sum: = 8 * **n** + 1

3. Big O notation

➤ Định nghĩa:

$g(n)$ được gọi là **O của $f(n)$** nếu tồn tại **C** (>0 , không phụ thuộc vào n) và **n_0** sao cho với mọi **$n > n_0$** , ta luôn có:

$$f(n) \leq C \cdot g(n)$$



3. Big O notation

➤ Example:

$$f(n) = 8 * n + 1$$

$$g(n) = n$$

$$C = 9$$

$$n_0 = 1$$

Với mọi $n > n_0$ ta luôn có: $C.g(n) \geq f(n)$

Với mọi $n > 1$ ta luôn có: $9.n \geq 8.n + 1$

Độ phức tạp: $O(n)$

4. Các độ phức tạp cơ bản

$O(1)$	Độ phức tạp hằng số
$O(\log n)$	Độ phức tạp logarit
$O(n)$	Độ phức tạp tuyến tính
$O(n^k)$	Độ phức tạp đa thức
$O(k^n)$	Độ phức tạp hàm mũ

4. Các độ phức tạp cơ bản

➤ Cách xác định ngắn gọn độ phức tạp:

$O(1, 2, 3, \dots)$	$O(1)$
$O(\log_2 n, \log_3 n, \dots)$	$O(\log n)$
$O(n + 100, n + 1000)$	$O(n)$
$O(n^k + n^{k-1}, \dots)$	$O(n^k)$
$O(k^n + k^{n-1}, \dots)$	$O(k^n)$

4. Các độ phức tạp cơ bản

➤ Quy tắc cộng:

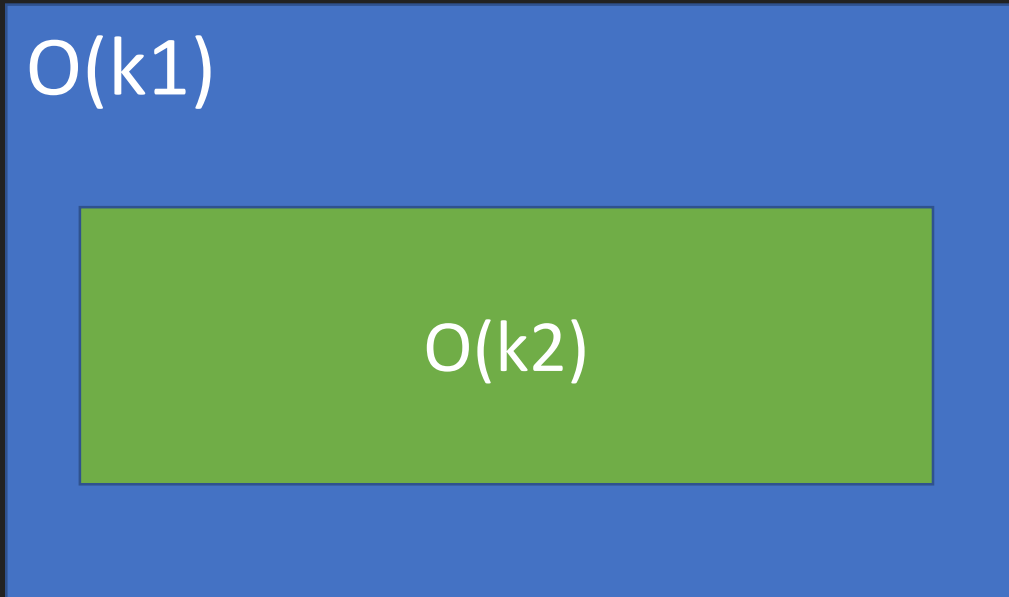
$O(k1)$

$O(k2)$

➤ $O = O(k1) + O(k2) = O(n)$

4. Các độ phức tạp cơ bản

➤ Quy tắc nhân:



➤ $O = O(k1) * O(k2) = O(n^2)$

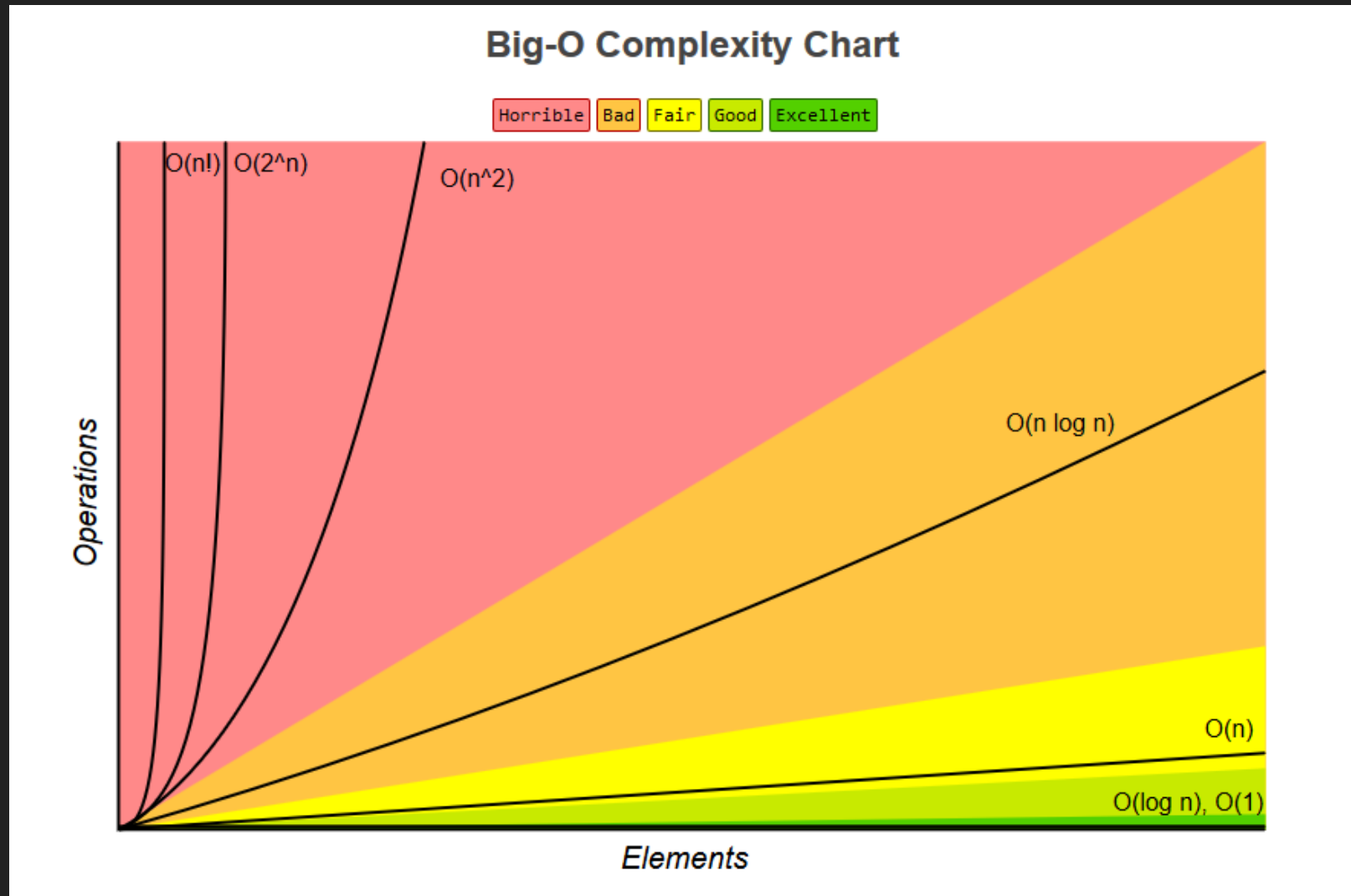
4. Các độ phức tạp cơ bản

➤ Example 2: Tìm giá trị x trong mảng `arr`

```
int[] arr = new int[n];  
int x = 10;  
for (int i = 0; i < arr.length; i++) {  
    if(arr[i] == x)  
    {  
        System.out.println("Found X at " + i);  
        break;  
    }  
}
```

Xác định với tình huống xấu nhất
(worst case)

4. Các độ phức tạp cơ bản



5. Xác định độ phức tạp của một số thuật toán

Tìm kiếm nhị phân	$O(\log n)$
Tìm kiếm tuần tự	$O(n)$
Sắp xếp nổi bọt Duyệt ma trận 2 chiều	$O(n^2)$
Fibonacci	$O(2^n)$
...	...

Data Structure & Algorithm



Please Like and Subscribe