1. **Chia để trị**

**-** Nhiều thuật toán mang tính đệ quy

- Để giải quyết: gọi lại chính nó cho bài toán con của nó

**=>** Thuật toán như vậy gọi là chia để trị

* 1. **Khái quát**

**Chia để trị** là một trong những phương pháp thiết kế giải thuật cơ bản bao gồm các thao tác:

- **Chia** bài toán thành nhiều bài toán nhỏ

- **Trị:** giải bằng cách gọi đệ quy cho đến khi bài toán con có thể được giải quyết 1 cách trực tiếp

- **Kết hợp:** tổng hợp nghiệm các bài toán con thành nghiệm của bài toán ban đầu

* 1. **Mô hình**

Vấn đề ban đầu

Vấn đề con

Giải đệ quy các vấn đề con

Giải pháp cho vấn đề ban đầu

Vấn đề con

Giải đệ quy các vấn đề con

***Chia***

***Trị***

***Kết hợp***

* 1. **Ví dụ**

**-** Binary search

- Quicksort

- Mergesort

…

**2.QuickSort:**  là **[một thuật toán sắp xếp](https://www.programiz.com/dsa/sorting-algorithm)** dựa trên cách tiếp cận phân **chia để trị,** trong đó:

- Một mảng được chia thành các mảng con bằng cách chọn một phần tử tổng hợp (phần tử được chọn từ mảng).  
 + Trong khi phân chia mảng, phần tử pivot phải được định vị theo cách sao cho các phần tử nhỏ hơn pivot được giữ ở bên trái và các phần tử lớn hơn pivot nằm ở bên phải của pivot.

- Các mảng con bên trái và bên phải cũng được chia theo cách tiếp cận tương tự. Quá trình này tiếp tục cho đến khi mỗi mảng con chứa một phần tử duy nhất.

- Tại thời điểm này, các phần tử đã được sắp xếp. Cuối cùng, các phần tử được kết hợp để tạo thành một mảng được sắp xếp.

**2.1 Nguyên lí hoạt động**

**B1. Chọn phần tử chốt (pivot)**  
Trong khi phân chia mảng, phần tử pivot phải được định vị theo cách sao cho các phần tử nhỏ hơn pivot được giữ ở bên trái và các phần tử lớn hơn pivot nằm ở bên phải của pivot.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 1 | 0 | 9 | 2 |

**pivot**

**B2. Sắp xếp lại mảng**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 1 | 0 | 9 | 2 |

**8 > 2 1 < 2**

**Swap(8,1)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 7 | 6 | 8 | 0 | 9 | 2 |

**7 > 2 1 < 2**

**Swap(7,0)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 6 | 7 | 8 | 9 | 2 |

**6 > 2**

**Swap(6,2)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 2 | 8 | 7 | 9 | 6 |

**B3.** **Phân chia các mảng con**

Các pivot lại được chọn cho các mảng phụ bên trái và bên phải một cách riêng biệt. Và, **bước 2** được **lặp lại**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 8 | 7 | 9 | 6 |  | 0 | 1 | 2 | 8 | 7 | 9 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 7 | 9 | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 7 | 9 | 8 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 6 | 7 | 9 | 8 |

|  |  |  |
| --- | --- | --- |
| 7 | 9 | 8 |

|  |  |  |
| --- | --- | --- |
| 7 | 8 | 9 |

|  |  |  |
| --- | --- | --- |
| 7 |  | 9 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 6 | 7 | 8 | 9 |

**2.2 Pivot**

**Pivot** là 1 phần tử trong mảng được chọn làm chốt. Thuật toán sẽ thực hiện chia mảng thành các mảng con dựa vào pivot đã chọn. Việc lựa chọn pivot ảnh hưởng rất nhiều tới tốc độ sắp xếp

**2.3 Độ phức tạp của quick sort**

**1. Tốt nhất O(nlog(n))**

Nó xảy ra khi phần tử chốt luôn là phần tử ở giữa hoặc gần với phần tử ở giữa.

N

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 13 | 16 | 9 | 1 | 2 | 15 | 10 |

|  |
| --- |
| 10 |

N/2

|  |  |  |
| --- | --- | --- |
| 9 | 1 | 2 |

|  |  |  |
| --- | --- | --- |
| 13 | 16 | 15 |

N/2

N/4

N/4

|  |
| --- |
| 2 |

|  |
| --- |
| 15 |

N/4

N/4

|  |
| --- |
| 9 |

|  |
| --- |
| 1 |

|  |
| --- |
| 13 |

|  |
| --- |
| 16 |

= 1 => k = log2n lần phân hoạch

Mỗi lần phân hoạch duyệt n phần tử

**=> O(logn \* n)**

1. **Xấu nhất O(n2)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 16 | 15 | 13 | 10 | 9 | 2 | 1 |

N

N - 1 1 16,15,13,10,9,2

N - 2 1 2 16,15,13,10,9

N - 3 1 2 9 16,15,13,10

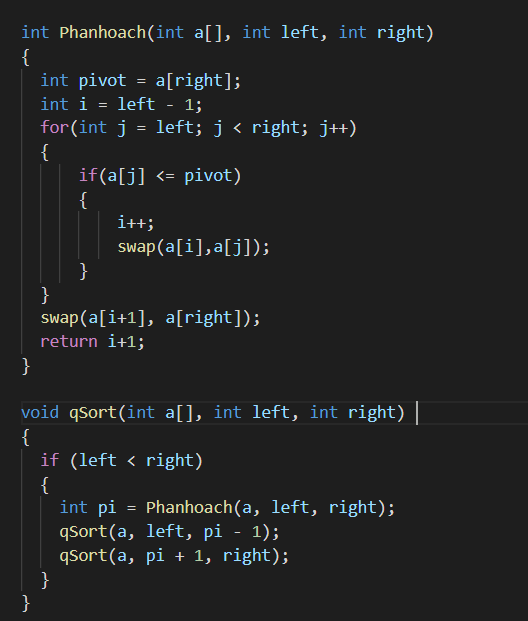
…

…

1. 2 9 10 … 16

n(n+1)/2 => **O(n2)**

**2.4 Code**



1. **Binary Search**

- Tìm kiếm nhị phân là một thuật toán tìm kiếm để tìm vị trí của một phần tử trong một mảng được sắp xếp.

- **Ý tưởng:** Tìm kiếm một mảng được sắp xếp bằng cách chia đôi khoảng thời gian tìm kiếm nhiều lần.

**B1: -** Bắt đầu với một khoảng bao gồm toàn bộ mảng.

- Tìm phần tử ở giữa mảng amid

**B2:**

**-** Nếu key < amid thu hẹp khoảng đó xuống nửa dưới.

- Nếu không, thu hẹp nó ở nửa trên.

**B3:** Lặp lại kiểm tra cho đến khi giá trị được tìm thấy hoặc khoảng thời gian trống

**3.1 Nguyên lí hoạt động**

**Tìm 12**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 8 | 11 | 13  **N** |

L R

**N/2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 8 | 11 | 13 |

L R

**N/4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 8 | 11 | 13 |

R L

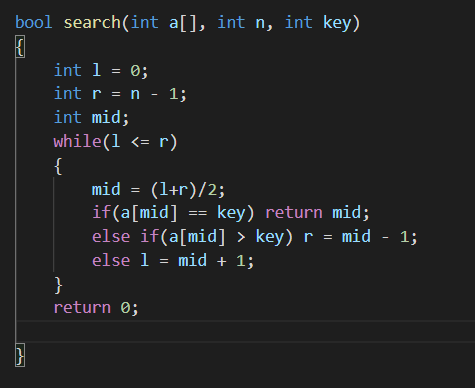
=> **Không tìm thấy 12**

**3.2 Độ phức tạp**

**N N/2 N/4 N/8 … N/2x = 1**

**=> x = log2N => O(logN)**

**3.3 Code**



1. **Giải bài tập**

**4.1 Số bé thứ k**

- Sắp xếp bằng quick sort

- In ra phần tử ở vị trí k (arr[k-1])

**Code: <https://ideone.com/scMqvr>**

**4.2 Help Sudo**

**-** Dùng Binarry Search để tìm kiếm sự xuất hiện của x

- Tìm thấy -> YES, không tìm thấy -> NO

**Code: <https://ideone.com/CLGRZ2>**