Trường Đại học Công nghệ Thông tin
Đại học Quốc Gia TP.HCM

PHƯƠNG PHÁP THIẾT KẾ THUẬT TOÁN

CHIA ĐỂ TRỊ

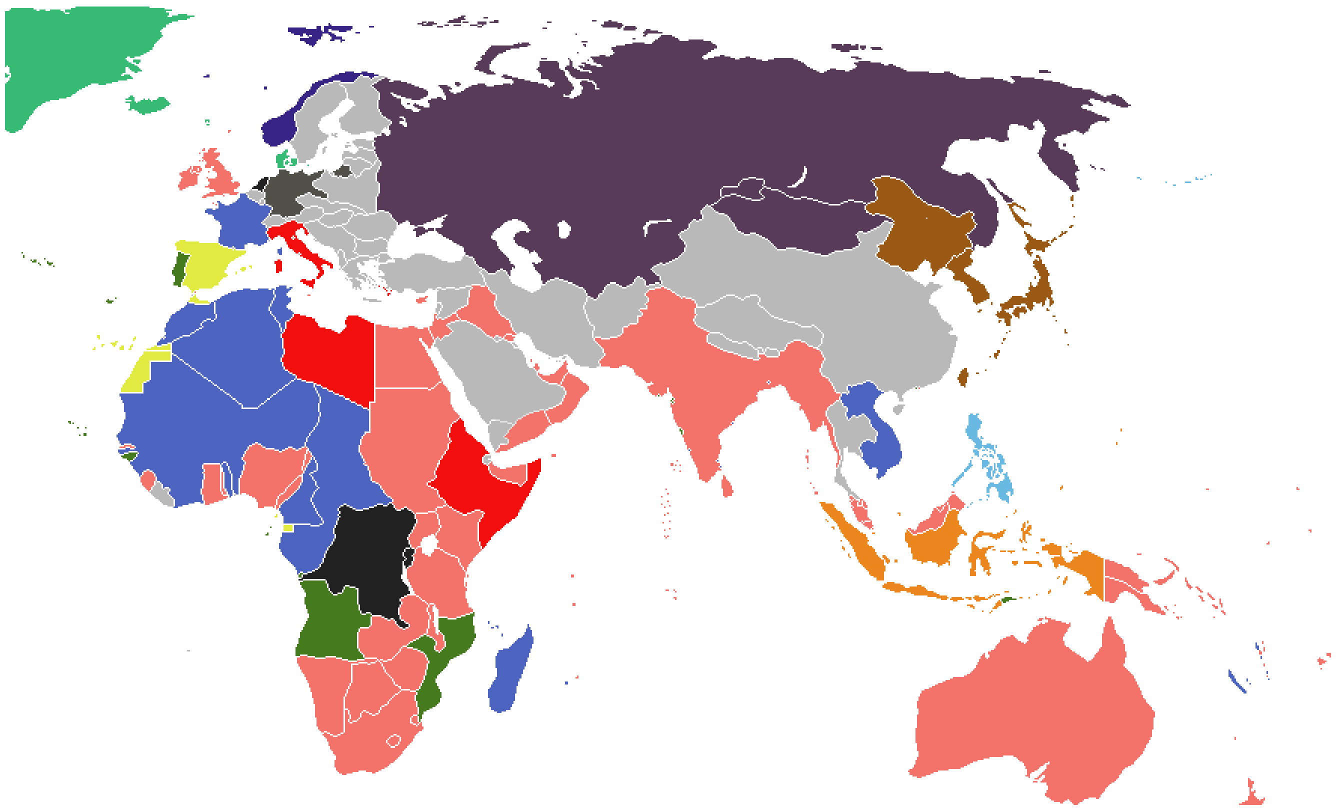
Lê Đoàn Phúc Minh
Nguyễn Duy Đạt

Nội dung

- Kỹ thuật chia để trị
- Các ứng dụng của kỹ thuật chia để trị
- Một số vấn đề khác

Kĩ thuật chia để trị

Các bạn hiểu Chia Để Trị là gì?



Khái niệm

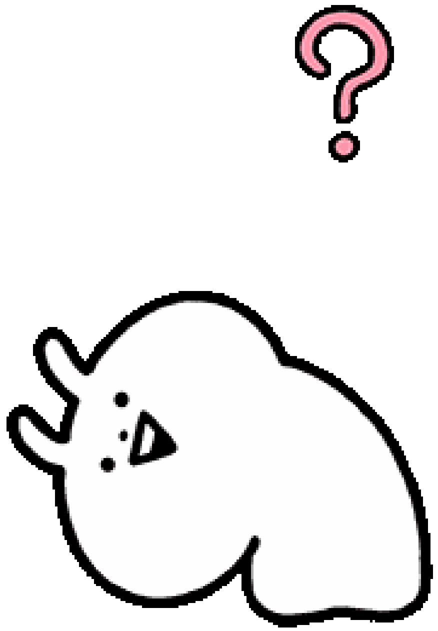
CHIA ĐỂ TRỊ

Chia (Divide): Chia bài toán ra thành các bài toán nhỏ hơn (subproblems). Về cơ bản thì những bài toán nhỏ này giống với bài toán ban đầu.

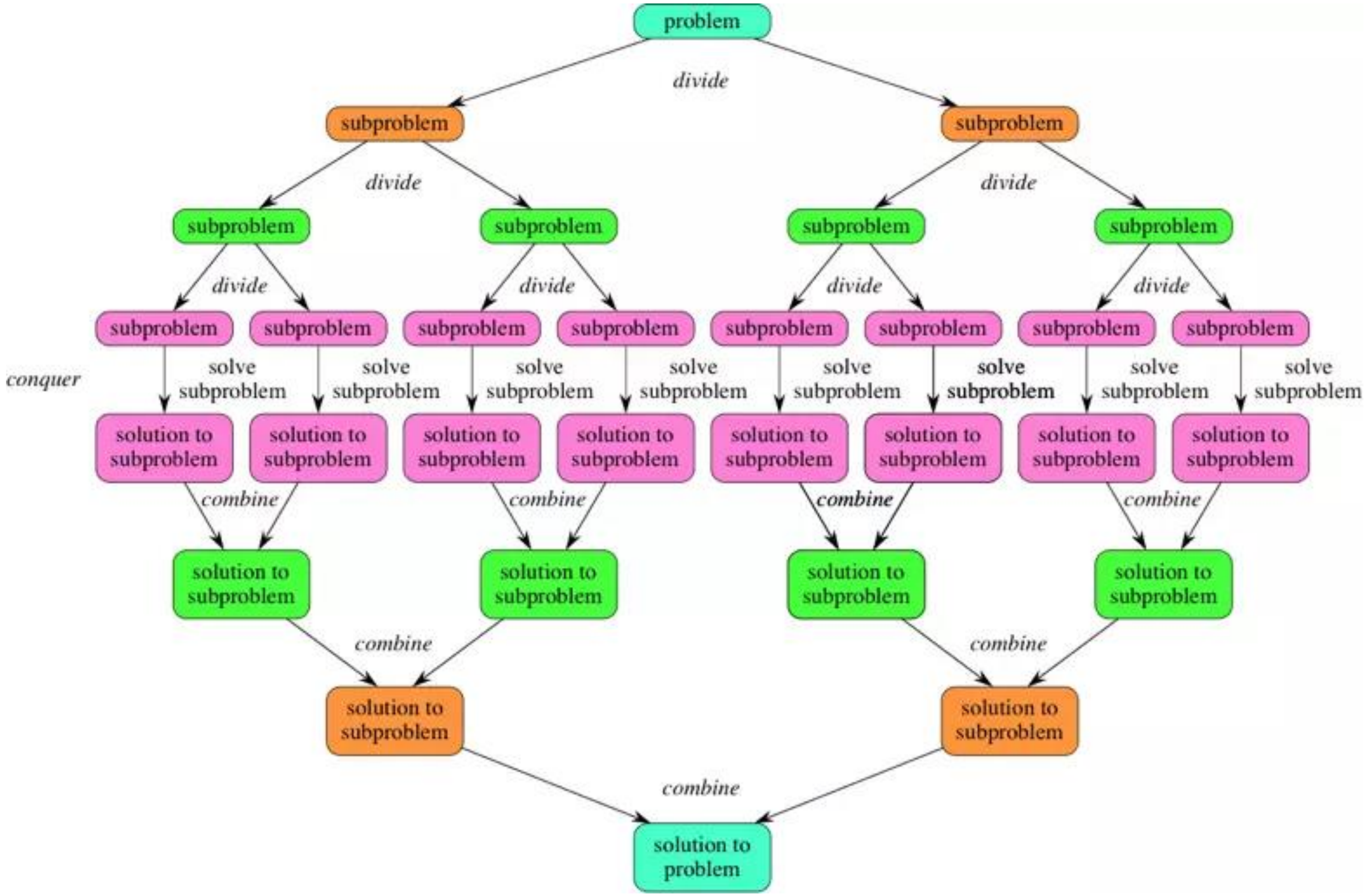
Trị (Conquer): Giải quyết bài toán con trong trường hợp nó đủ nhỏ, còn không thì tiếp tục tiến hành chia tách nó ra thành những bài toán con nhỏ hơn nữa.

Kết hợp (Combine): Kết hợp các kết quả từ bài toán con nhỏ nhất, để ra lời giải cho các bài toán con (subproblems), và cứ thế cuối cùng ra được lời giải cho bài toán ban đầu.

Khi nào thì mình dùng chia để trị?



Hình ảnh miêu tả giải thuật chia để trị



Template

```
DAC(a) {
    // Nếu bài toán đủ nhỏ thì giải thẳng luôn! Còn không thì phải chia nhỏ ra!
    if (small(a))
        return solution(a)
    else {
        // Chia bài toán thành n bài toán con (n là bao nhiêu tùy vào thuật toán)
        part = divide(a)

        // Xử lí bài toán con
        // Tùy vào số lượng bài toán con mà số lượng bài toán phải trị có thể nhiều hơn
        part[0] = DAC(part[0])
        part[1] = DAC(part[1])
        //...

        // Kết hợp kết quả từ bài toán con để suy ra kết quả bài toán trước đó
        result = combine(part)
    }
```

Master Theorem

$$T(n) = aT(n/b) + f(n),$$

+ a >= 1, b > 1
+ n là lũy thừa của b
+ f(n) là hàm tính thời gian phân chia n thành n/b và gộp các kết quả lại

+ Nếu f(n) ∈ O(n^d) , d >= 0

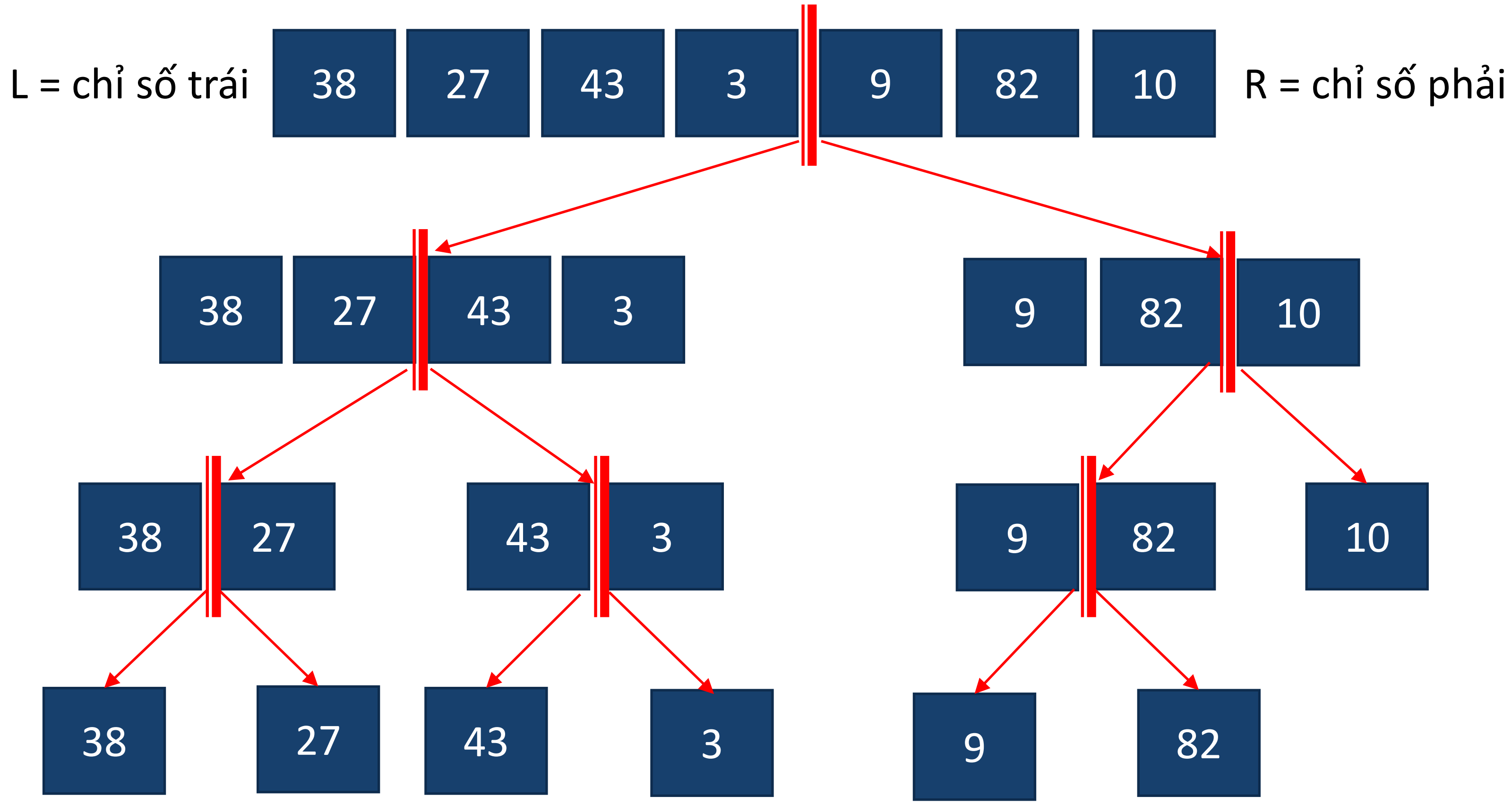
$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d, \\ \Theta(n^d \log n) & \text{if } a = b^d, \\ \Theta(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

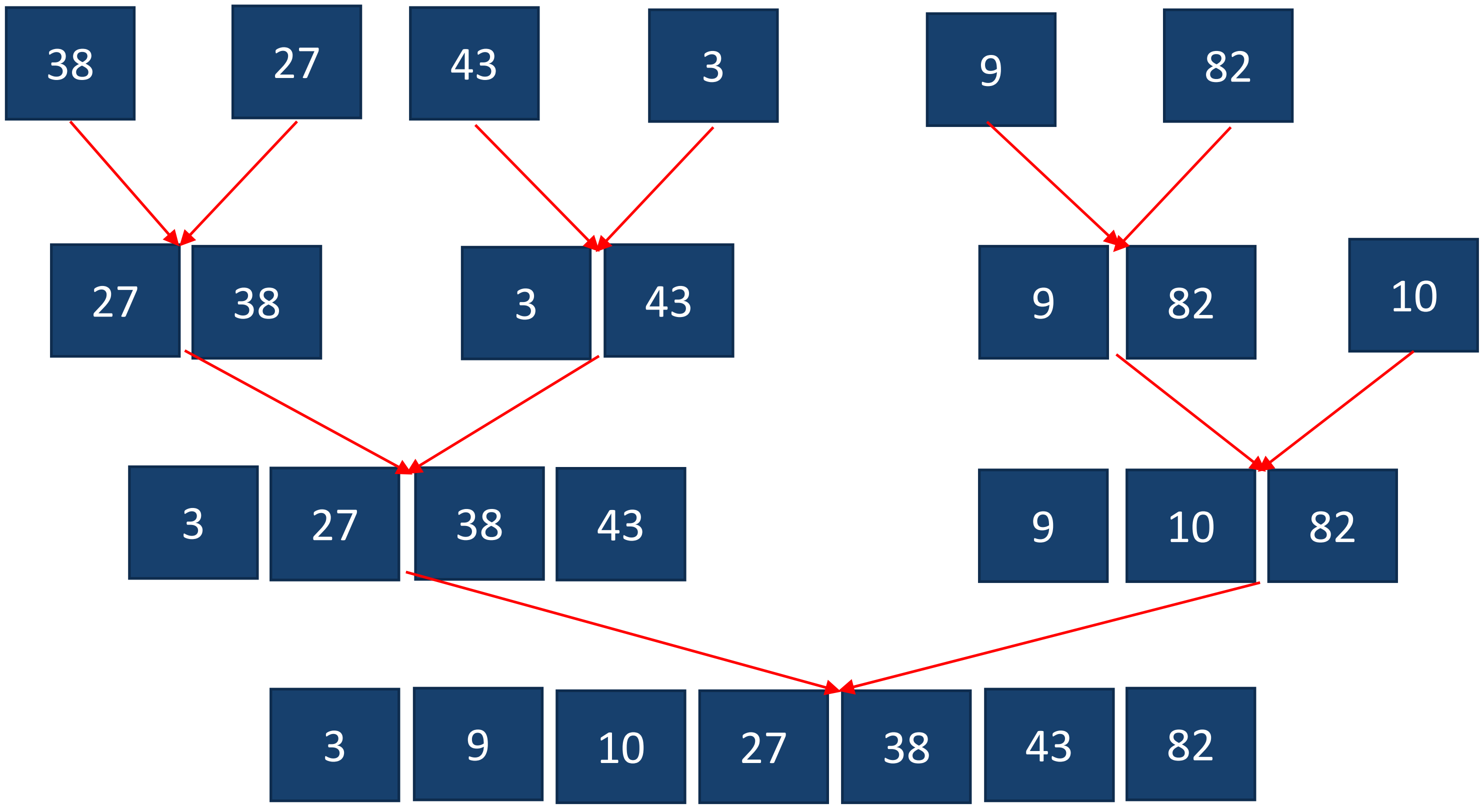
Các ứng dụng của kỹ thuật chia để trị

- **Thuật toán sắp xếp gộp (Merge Sort)**
- **Thuật toán tính lũy thừa nhanh**
- **Thuật toán nhân nhanh Karatsuba**
- **Bài toán tìm cặp đỉnh gần nhau nhất**

Các bạn hãy cho biết thuật toán MergeSort hoạt động như thế nào?

+ Chia : Chia đôi mảng
+ Trị: Sử dụng đệ quy sắp xếp 2 mảng con
+ Gộp: Gộp 2 mảng con với thời gian tuyến tính


L = chỉ số trái
38
27
43
3
9
82
10
R = chỉ số phải
38
27
43
3
9
82
10
38
27
43
3
9
82
10
38
27
43
3
9
82



Thuật toán sắp xếp gộp (Merge Sort)

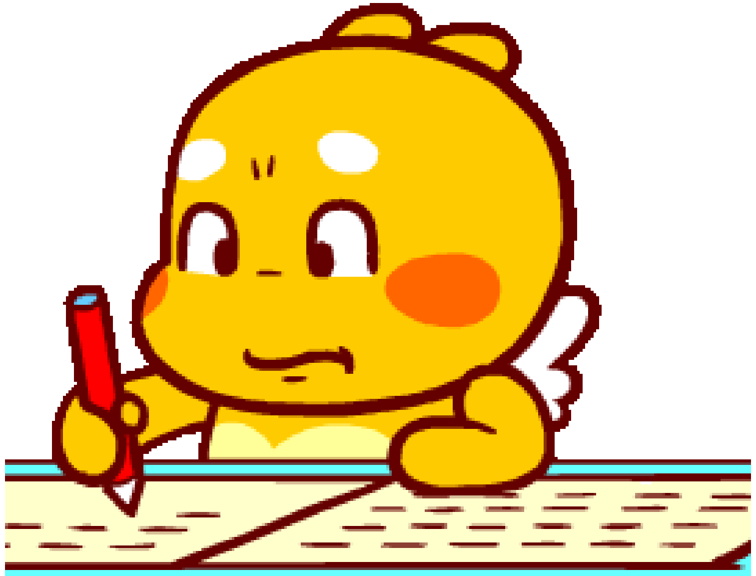
Dựa vào Master Theorem, hãy thử tính độ phức tạp của mergeSort!

$$T(n) = aT(n/b) + f(n),$$

+ a >= 1, b > 1
+ n là lũy thừa của b
+ f(n) là hàm tính thời gian phân chia n thành n/b và gộp các kết quả lại

+ Nếu f(n) ∈ O(n^d) , d >= 0

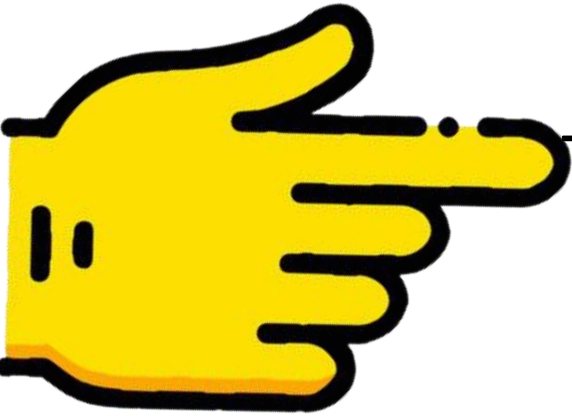
$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$



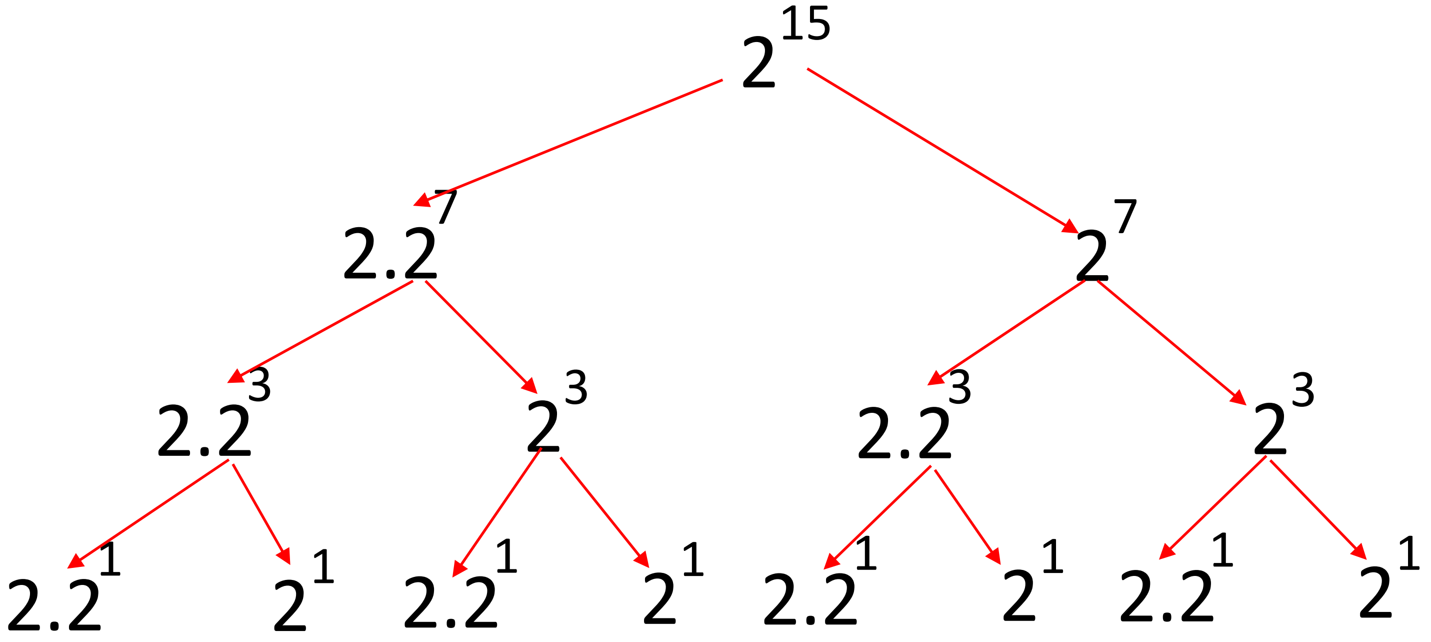
Các bạn hãy thử viết code hàm mergeSort !

Tính 2^9

Các bạn hãy thử viết chương trình tính lũy thừa theo cách đơn giản nhất (sử dụng for hoặc while)



Tính 2^{15}



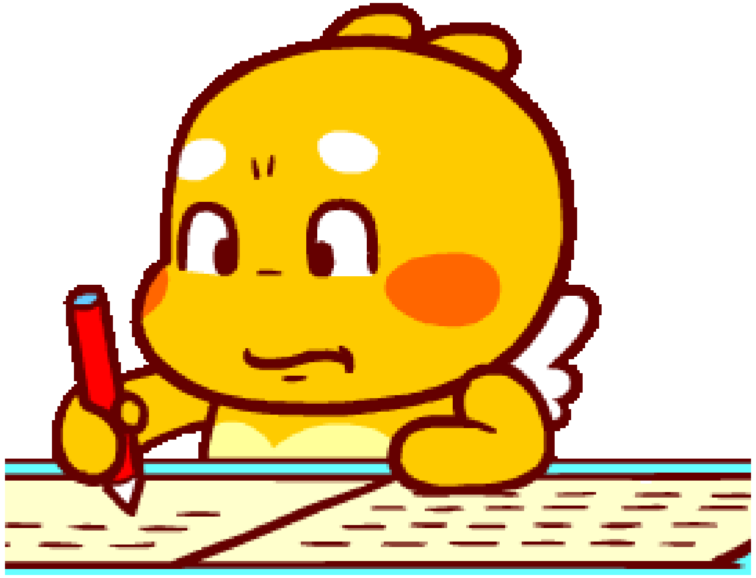
Dựa vào Master Theorem, hãy thử tính độ phức tạp của tính lũy thừa!

$$T(n) = aT(n/b) + f(n),$$

+ a >= 1, b > 1
+ n là lũy thừa của b
+ f(n) là hàm tính thời gian phân chia n thành n/b và gộp các kết quả lại

+ Nếu $f(n) \in O(n^d)$, d >= 0

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$



Các bạn hãy thử viết chương trình tính lũy thừa a^x
bằng chia để trị (a và x rất lớn)

Bài toán: Cho 2 số tự nhiên, hãy cho biết tích của 2 số đó

Phương pháp ngây thơ:

$$
\begin{array}{r}
12345 \\
* \quad 6789 \\
\hline
111105 \\
98760 \\
+ \quad 86415 \\
74070 \\
\hline
= 83810205
\end{array}
$$

(Kết quả)

Độ phức tạp thuật toán: O(n*m)

Kỹ thuật nhân nhanh Karatsuba:

Đầu tiên, ta có x, y là số có n chữ số. $\forall m \in \mathrm{N}$ và $n > 0$, ta viết lại hai số đã cho thành:

$$x = x_1 10^m + x_0 \text{ và } y = y_1 10^m + y_0$$

$$(x_0, y_0 < 10^m)$$

Gọi $z_2 = x_1 y_1$, $z_1 = x_1 y_0 + x_0 y_1$ và $z_0 = x_0 y_0$.

Lúc này, ta có:

$$x * y = (x_1 10^m + x_0)(y_1 10^m + y_0)$$

$$= z_2 10^{2m} + z_1 10^m + z_0$$

Có thể viết lại z_1 thành $z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$.

Ứng dụng của kỹ thuật chia để trị và thuật toán đầy đủ:

$xy = ?$

$xy = z_2 10^{2m} + z_1 10^m + z_0$

$z_2 = z_{2'} 10^{2m} + z_{1'} 10^m + z_{0'}$

$z_1 = x_1' y_1' - z_2 - z_0$

$z_0 = z_{2'} 10^{2m} + z_{1'} 10^m + z_{0'}$

…

$x_1' y_1' = z_{2'} 10^{2m} + z_{1'} 10^m + z_{0'}$

…

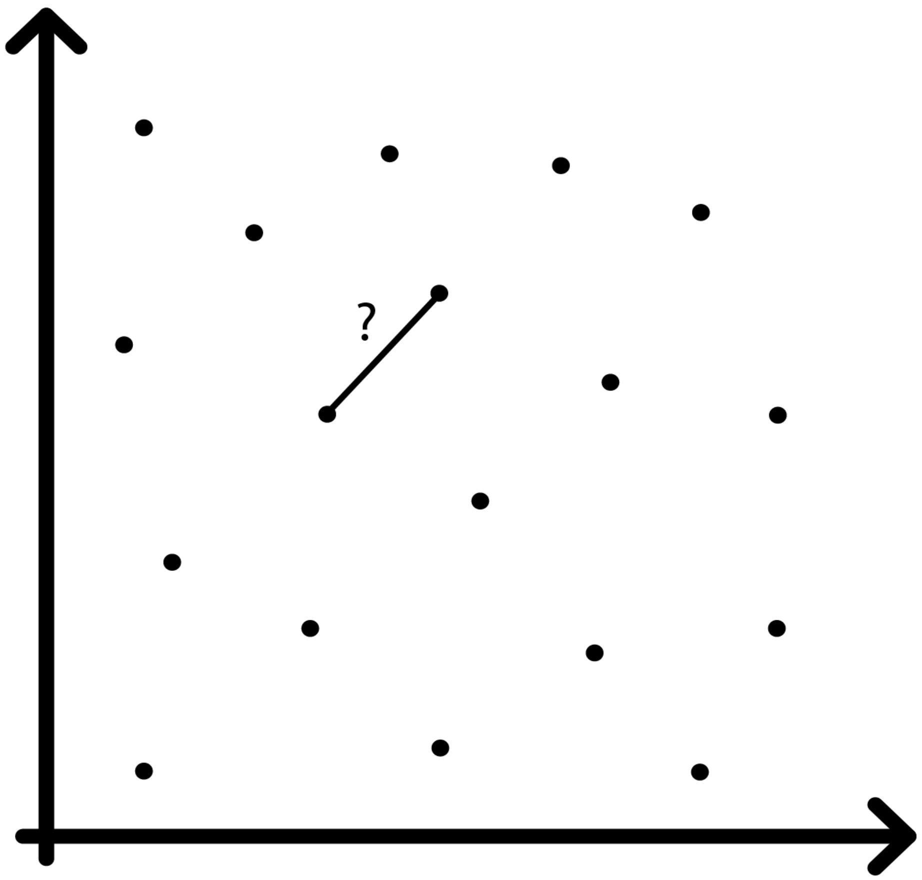
…

Lưu ý: x1'y1' = $(x_1 + x_0)(y_1 + y_0)$

Theo các bạn, với phép tính 1234567890 * 987654321 và m = 3, thuật toán này sẽ phải thực hiện công đoạn chia và công đoạn trị bao nhiêu lần ?

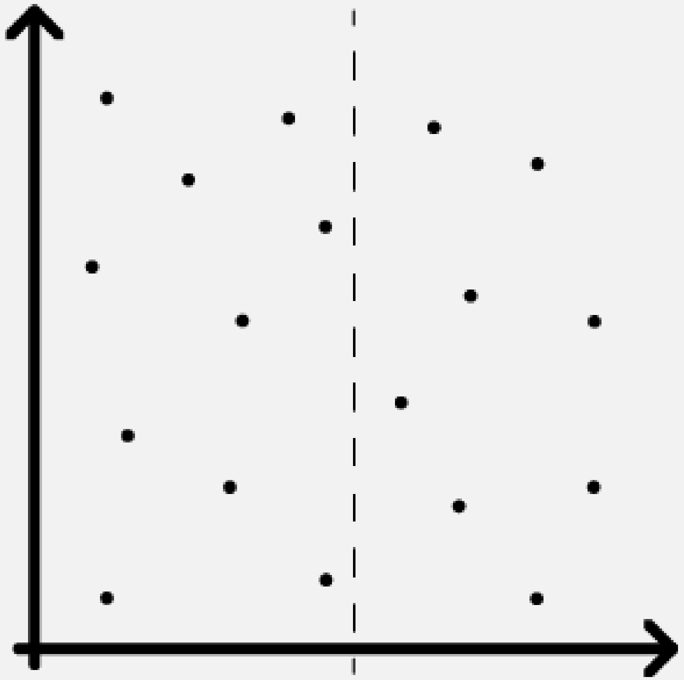
Bài toán:

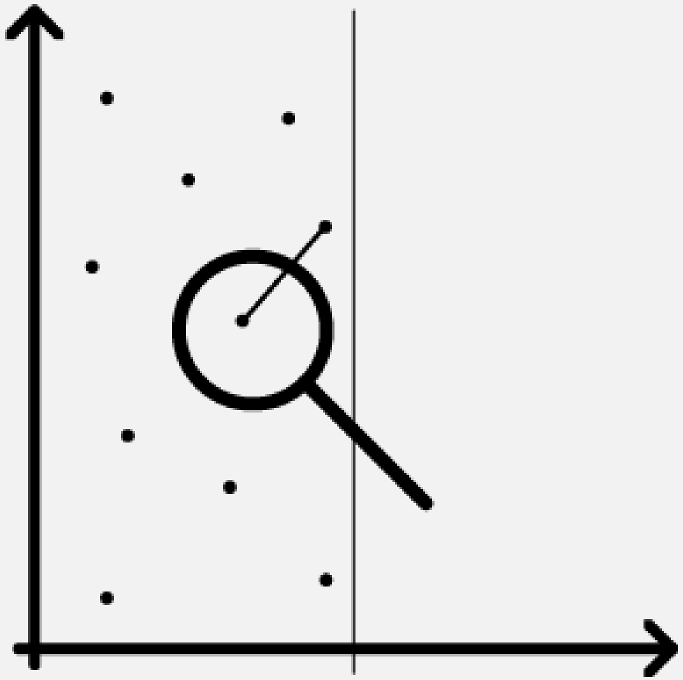
Cho một mặt phẳng gồm N điểm, hãy tìm cặp điểm gần nhau nhất và khoảng cách của chúng.

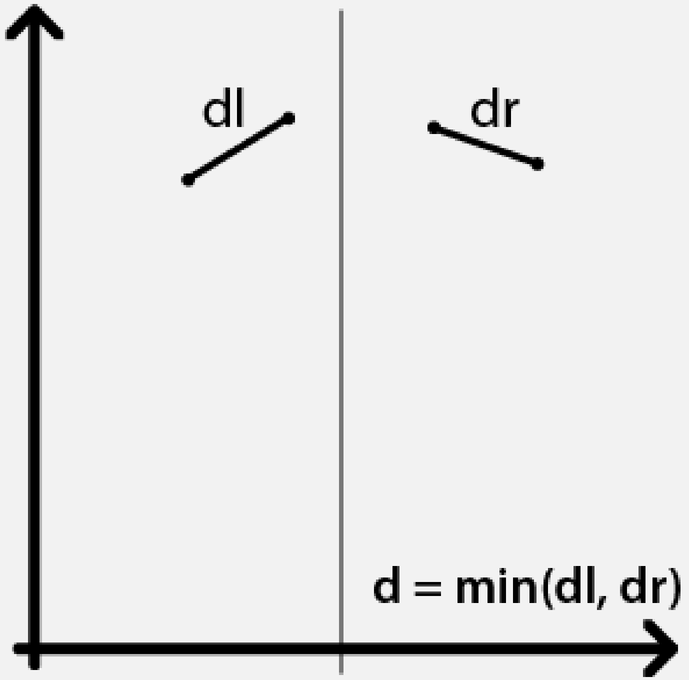


Thuật toán chia để trị:

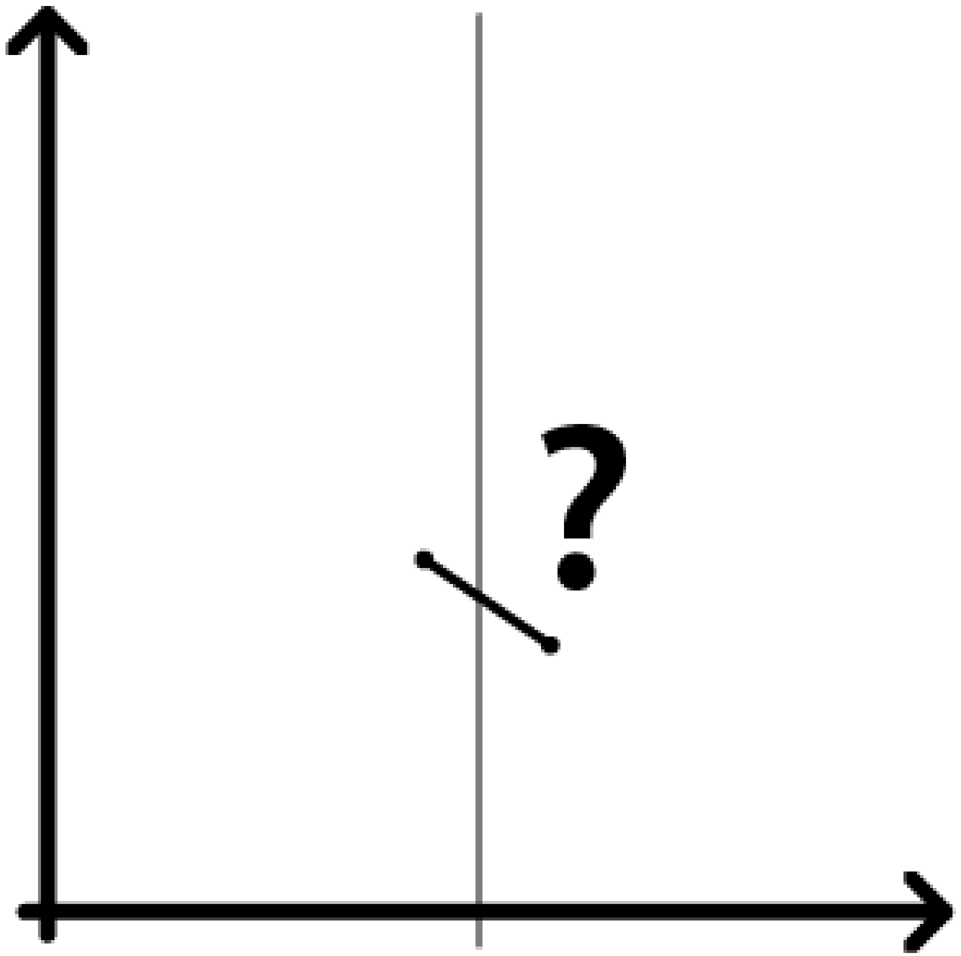
Chia	Trị	Gộp





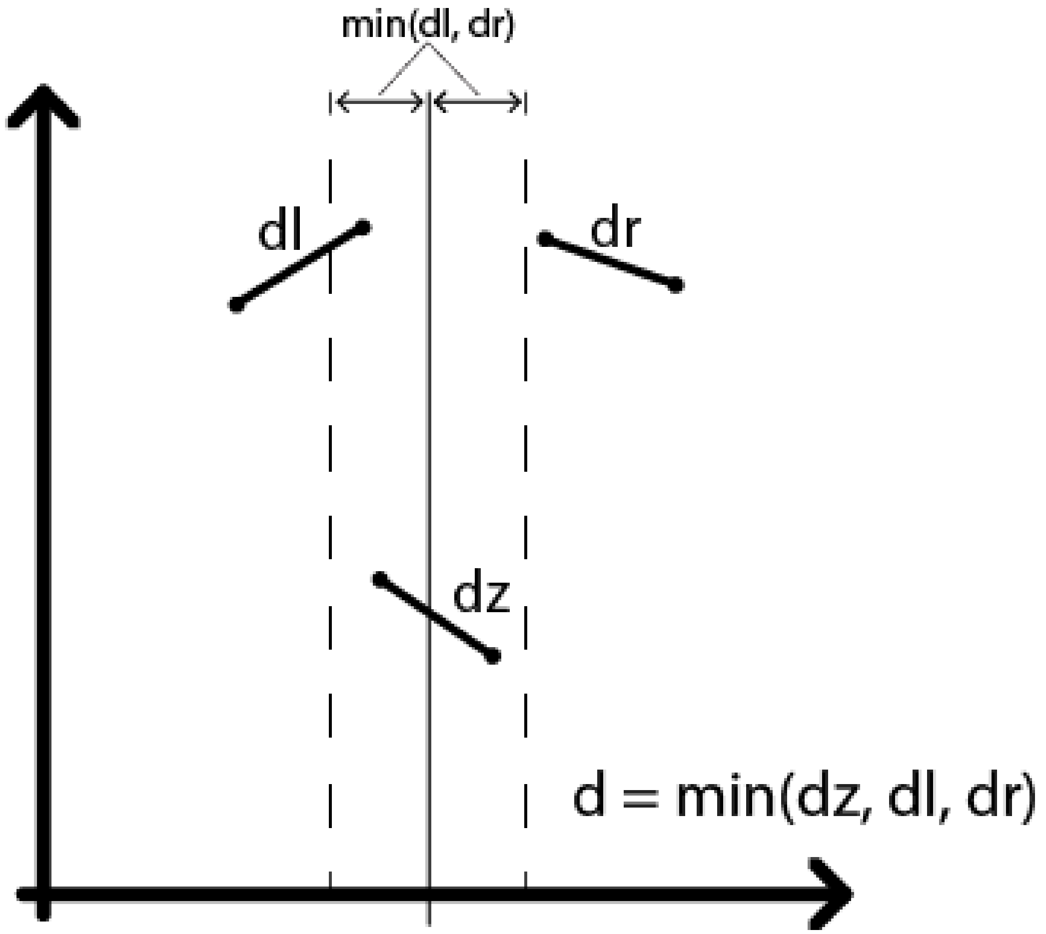


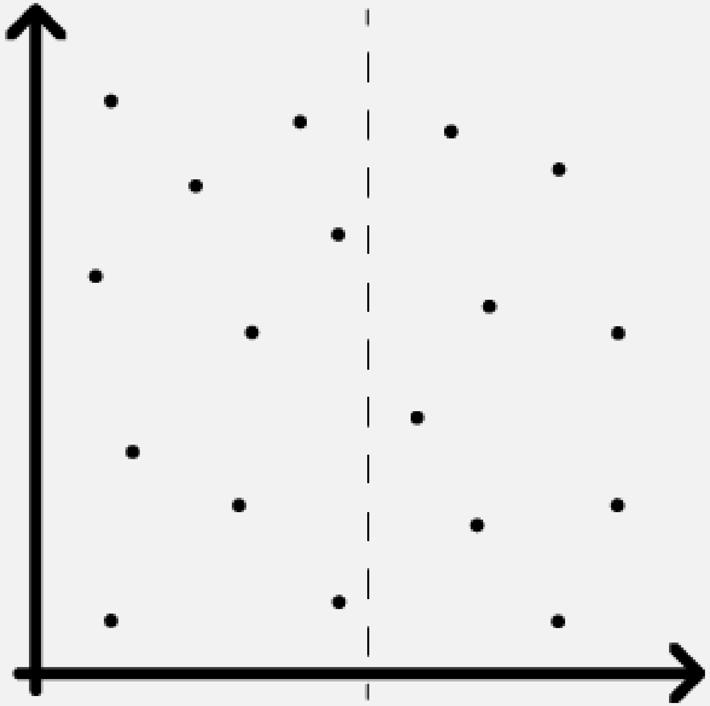
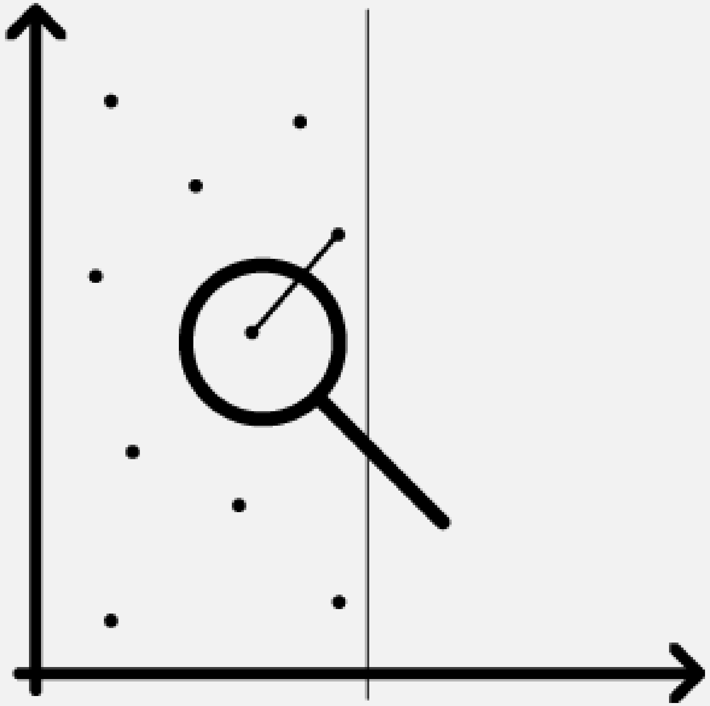
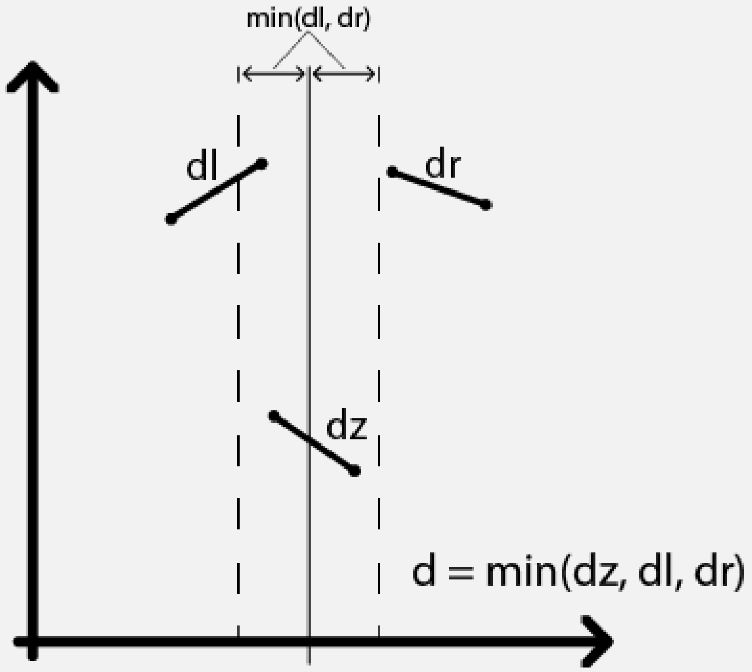
Thế còn trường hợp này ?



Đây là giải pháp:

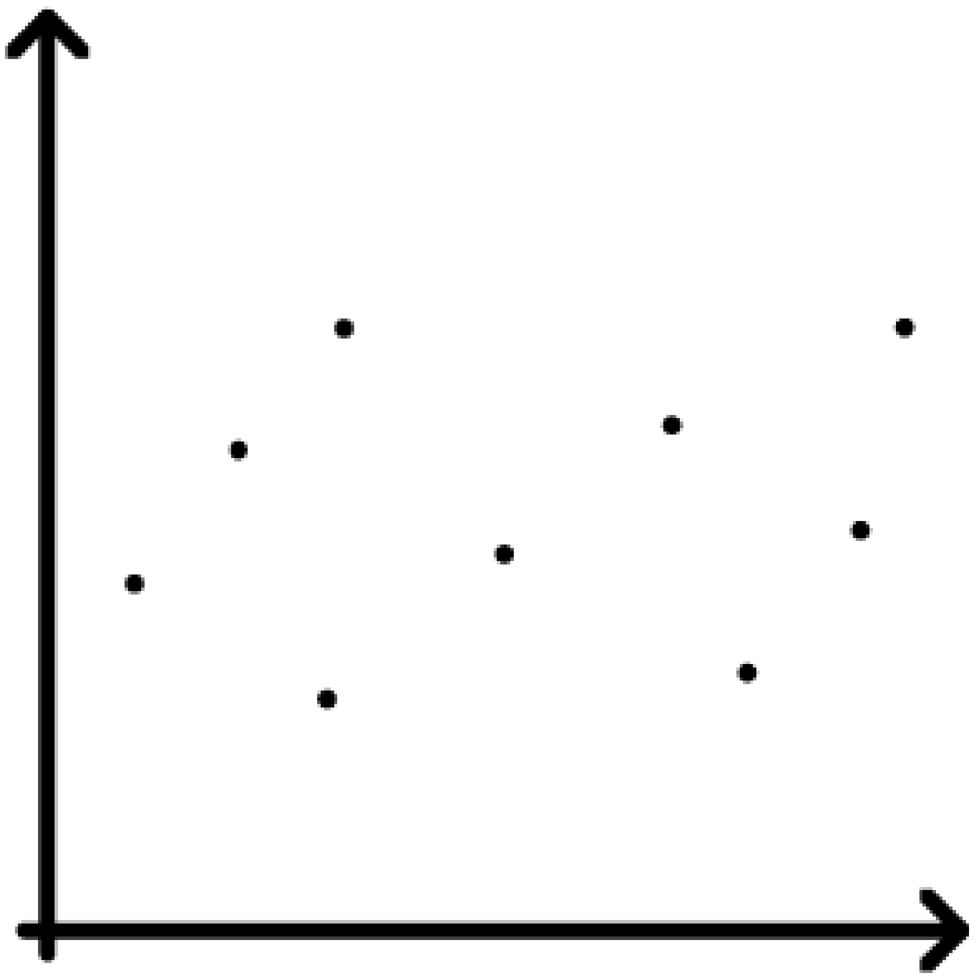
Gọi d_2 = min(dl, dr). Lúc này, ta sẽ khoanh vùng các điểm có tọa độ x nằm trong khoảng từ $[-d_2; d_2]$ và tìm khoảng cách dz nhỏ nhất giữa các điểm nằm trong này và so sánh chúng với min(dl, dr).



Chia	Trị	Gộp
		

Lưu ý: Trước khi vào giai đoạn chính của thuật toán, các bạn nên sắp xếp lại các điểm theo tọa độ x để tránh chọn sai điểm.

Với các điểm đã cho dưới đây, thuật toán sẽ thực hiện thao tác chia và thao tác trị bao nhiêu lần ?



Một số vấn đề khác

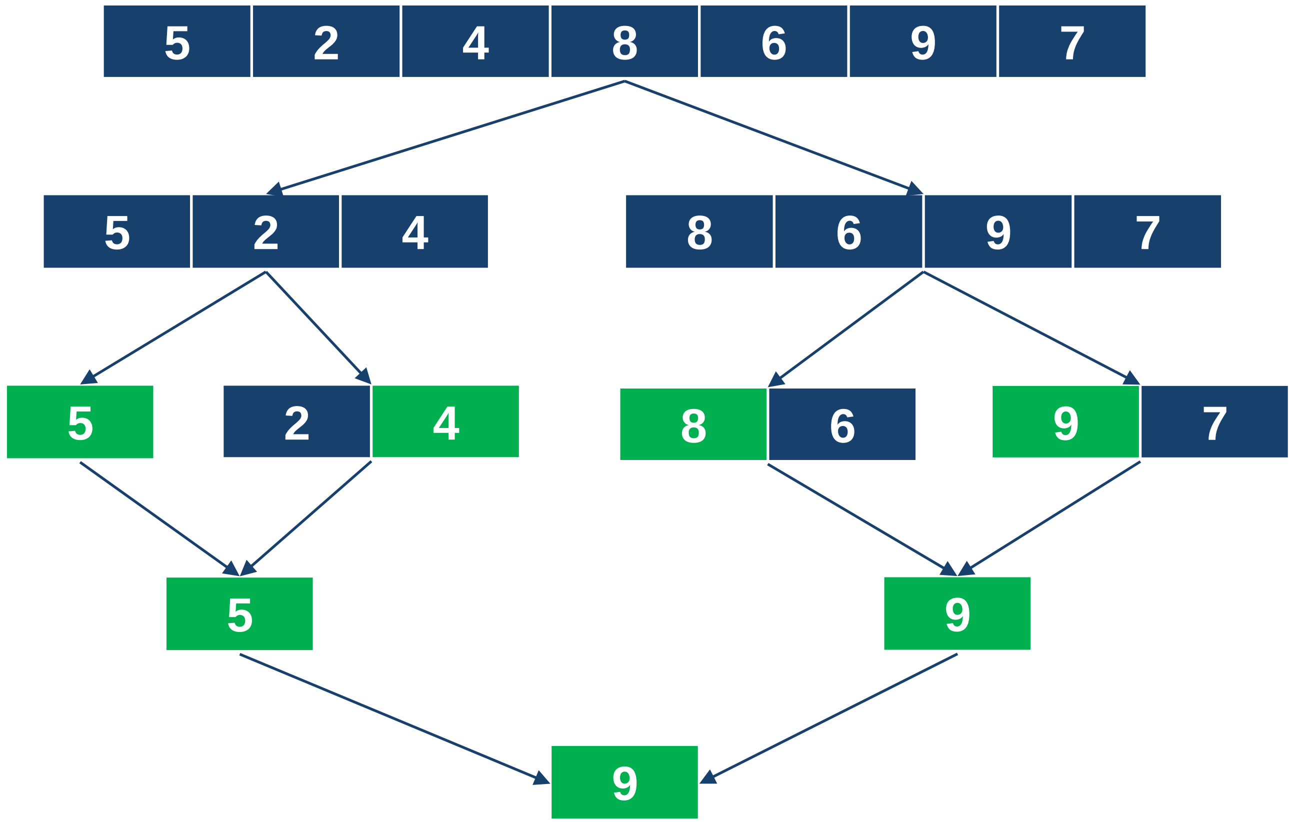
Ưu điểm và nhược điểm:

Ưu điểm	Nhược điểm
• Cho phép chúng ta dễ dàng giải quyết những bài toán khó xử lý và nhìn nhận. • Thường được sử dụng trong các thuật toán hiệu suất cao. • Sử dụng hiệu quả bộ nhớ đệm CPU. • Hiệu năng xử lý tốt hơn Brute Force. • Dễ dàng áp dụng trong việc lập trình ứng dụng Multi-Thread.	• Đệ quy có thể trở thành rào cản hiệu năng. • Không phải bài toán nào cũng sử dụng tốt phương pháp này. • Các bài toán con khi được chia ra có thể bị trùng với bài toán trước đó. • Chia bài toán con không tốt có thể dẫn đến hiệu suất thuật toán bị giảm.

Ví dụ các bài toán không ứng dụng được:

Tìm Min/Max trong dãy số

Nguyên nhân: Phương pháp chia để trị cho ra thuật toán không tốt hơn phương pháp duyệt



Ví dụ các bài toán không ứng dụng được:

Tìm số tại vị trí x trong dãy số Fibonacci

Nguyên nhân: Có phương pháp khác tốt hơn

0	1	2	3	4	5	6	7	…
1	1	2	3	5	8	13	21	…

x = 5

Ví dụ các bài toán không ứng dụng được:

Bài toán xếp lịch

Nguyên nhân: Do bản chất bài toán

Customer #	Start	End
1	7:30 AM	8:30 AM
2	8:00 AM	10:00 AM
3	9:00 AM	10:30 AM
4	11:00 AM	12:00 PM
5	11:30 AM	1:00 PM

Tài liệu tham khảo

- Anany Levitin, Introduction to the Design and Analysis of Algorithms, 3rd Edition, 2014
- Divide and Conquer Algorithm | Introduction - Geeksforgeeks.org
- Divide and Conquer Introduction - javatpoint.com
- Wikipedia:
  - Divide and Conquer
  - Karatsuba Algorithm