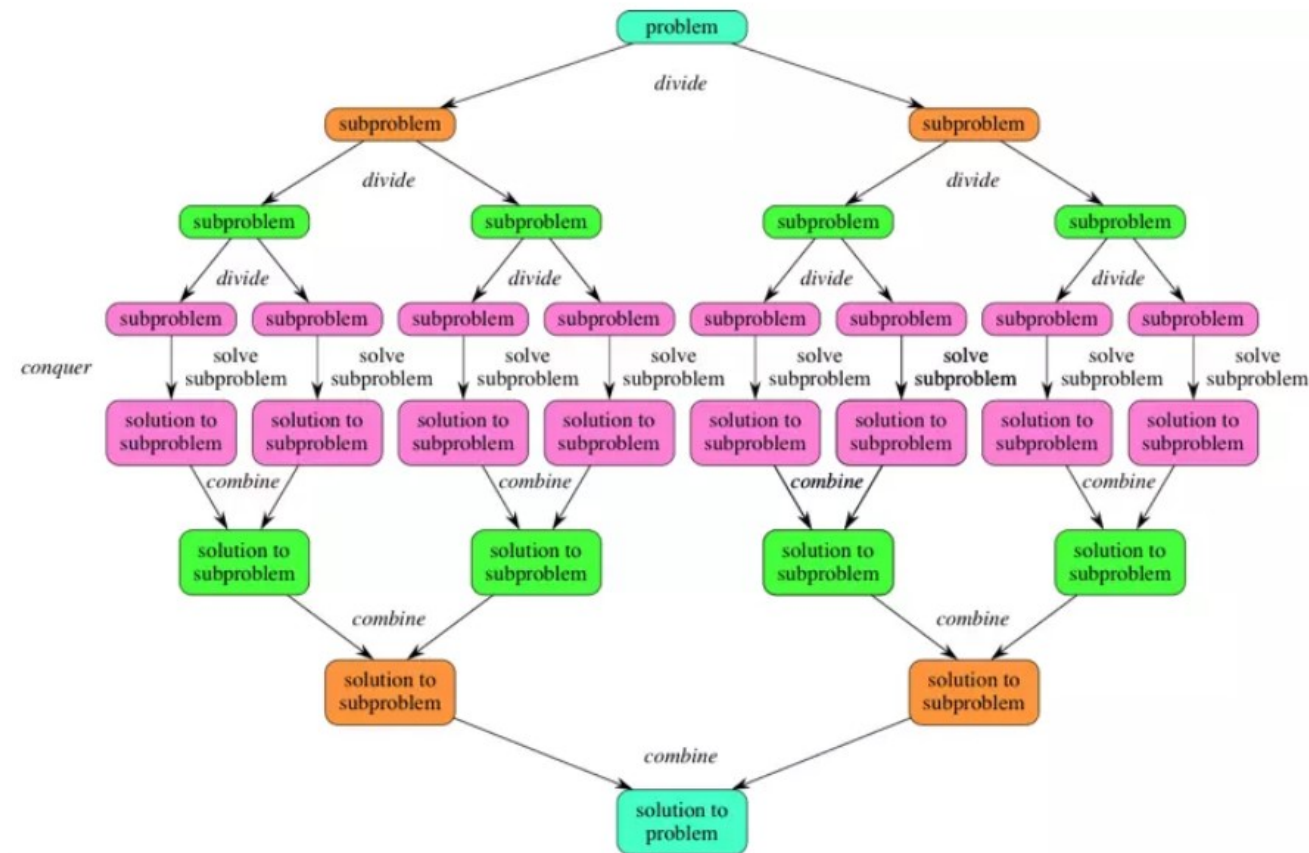


## Thuật toán sắp xếp nhanh (Quick sort)

# Chia và trị - Divide and conquer



- **Chia và trị** là phương pháp thiết kế thuật toán theo kiểu:
  - **Phân chia**: Chia dữ liệu đầu vào  $S$  của bài toán thành 2 tập con rời nhau  $S_1$  và  $S_2$
  - **Đệ quy**: Giải bài toán với dữ liệu vào là các tập con  $S_1$  và  $S_2$
  - **Trị**: kết hợp các kết quả của  $S_1$  và  $S_2$  thành kết quả của  $S$
- Trường hợp cơ sở cho thuật toán đệ quy ở đây là các bài toán có kích thước 0 hoặc 1



# Thuật toán sắp xếp nhanh – Quick sort



- **Ý tưởng (sử dụng phương pháp chia và trị):**
  - Thực hiện phân hoạch dãy S cần sắp thành 3 dãy S1, S2, S3. Trong đó:
    - S<sub>2</sub> chỉ có một phần tử
    - Tất cả các phần tử của dãy S3 đều  $>$  phần tử của dãy S2.
    - Tất cả các phần tử của dãy S1 đều  $\leq$  phần tử của dãy S2
    - Dãy S1, S3 có thể là rỗng
  - Tiếp tục phân hoạch dãy S1 và S3 độc lập theo nguyên tắc trên đến khi dãy cần thực hiện phân hoạch chỉ có một phần tử thì dừng lại. Khi đó ta được dãy các phần tử được sắp.

# Thuật toán sắp xếp Quick sort



- ♦ Từ ý tưởng của thuật toán, ta có thể dễ dàng xây dựng thuật toán sắp xếp dưới dạng đệ qui như sau:

**Algorithm *QuickSort* (array  $A$ ,  $i$ ,  $j$ );**

**Input:** Dãy các phần tử  $A[i], \dots, A[j]$  và hai số nguyên  $i, j$

**Output:** Dãy  $A[i], \dots, A[j]$  được sắp.

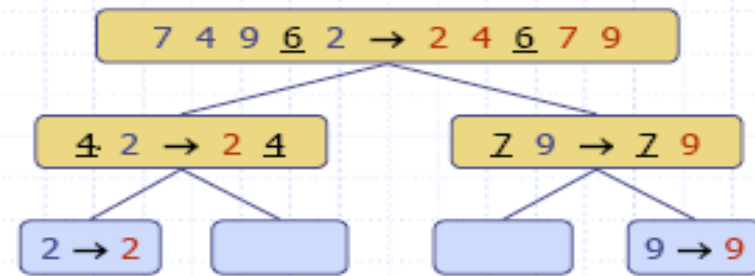
**if**  $i < j$  **then**

    Partition ( $A, i, j, k$ ); //k lấy chỉ số của phần tử làm S2

    Quicksort ( $A, i, k-1$ );

    Quicksort ( $A, k+1, j$ );

## Quick-Sort



Quick-Sort


1

**Vấn đề đặt ra ở đây là phân hoạch dãy S như thế nào?**


# Thuật toán phân hoạch




- Chọn một phần tử bất kỳ của dãy làm dãy S2 (phần tử này được gọi là phần tử chốt - pivot).
- Thực hiện chuyển các phần tử có khóa  $\leq$  phần tử chốt về bên trái và các phần tử  $>$  phần tử chốt về bên phải, sau đó đặt phần tử chốt về đúng vị trí của nó trong dãy.



<u>6</u>	12	32	1	3
----------	----	----	---	---



<u>6</u>	3	32	1	12
----------	---	----	---	----



<u>6</u>	3	1	32	12
----------	---	---	----	----

1	3	<u>6</u>	32	12
---	---	----------	----	----

Sau khi phân hoạch

- Phần tử chốt có thể được chọn là một phần tử bất kỳ của dãy.
  - Phần tử chốt có thể chọn là phần tử đầu hoặc giữa hoặc cuối dãy.
  - Tốt nhất là chọn phần tử chốt mà nó làm cho việc phân hoạch thành hai dãy  $S_1$  và  $S_3$  có số phần tử xấp xỉ bằng nhau.



- Phân hoạch dãy gồm các phần tử  $A[i], \dots, A[j]$
- Chọn phần tử **đầu** dãy làm chốt
- Sử dụng 2 biến *left* và *right*:
  - left chạy từ trái sang phải bắt đầu từ i.
  - right chạy từ phải sang trái bắt đầu từ j
  - Biến left được tăng cho tới khi  $A[\text{left}].\text{Key} > A[i].\text{Key}$  hoặc  $\text{left} > \text{right}$
  - Biến right được giảm cho tới khi  $A[\text{right}].\text{Key} \leq A[i].\text{Key}$
  - Nếu  $\text{left} < \text{right}$  thì ta đổi  $A[\text{left}]$  và  $A[\text{right}]$
  - Quá trình trên được lặp lại cho tới khi nào  $\text{left} > \text{right}$
  - Cuối cùng trao đổi  $A[i]$  và  $A[\text{right}]$

# Ví dụ phân hoạch



10	3	24	1	4	21	54	5
i							j

?

# Thuật toán phân hoạch



**Algorithm** *Partition (Array A, i, j, &right)*

**Input:** Dãy các phần tử  $A[i], \dots, A[j]$ , 2 số nguyên  $i, j$

**Output:** Dãy  $A[i], \dots, A[j]$  được phân hoạch, right là chỉ số của phần tử làm S2.

$p \leftarrow A[i];$

$left \leftarrow i; right \leftarrow j;$

**while** (  $left < right$  )

**while**(  $A[left].Key \leq p.Key$  &&

$left \leq right$ )

$left \leftarrow left + 1;$

**while**(  $A[right].Key > p.Key$  )  $right$

$\leftarrow right - 1;$

**if**  $left < right$  **then**

**SWAP**( $A[left], A[right]$ ); Sorting

**if**  $i \neq right$  **then**

$A[i] \leftarrow A[right];$

$A[right] \leftarrow p;$

# Ví dụ Sắp xếp dãy số



A= ... 

10	3	24	1	4	21	54	5
----	---	----	---	---	----	----	---

 ...  
i=1 j=8

?

[Videos mô phỏng quá trình sắp xếp bằng thuật toán Quicksort](#)

# Mô tả quá trình Sắp xếp



Quicksort(A,1, 8)

10	3	24	1	4	21	54	5
----	---	----	---	---	----	----	---

i=1 j=8

i < j partition(A,1,8,k)

4	3	5	1	10	21	54	24
---	---	---	---	----	----	----	----

i=1 k=5 j=8

Quicksort(A,1, 4)

4	3	5	1	10	21	54	24
---	---	---	---	----	----	----	----

i=1 j=4

i < j partition(A,1,4,k)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=1 k=3 j=4

# Mô tả quá trình Sắp xếp



Quicksort(A,1, 2)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=1 j=2

i<j partition(A,1,2,k)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=k=1 j=2

Quicksort(A,1,0)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=1 j=0

Quicksort(A,2, 2)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=j=2

Quicksort(A,4, 4)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=j=4

Quicksort(A,6, 8)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=6

j=8

i<j partition(A,6,8,k)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

i=k=6

j=8

# Mô tả quá trình Sắp xếp



Quicksort(A,6,5)

1	3	4	5	10	21	54	24
---	---	---	---	----	----	----	----

j=5 i=6

Quicksort(A,7,8)  
i<j Partition(A,7,8,k)

1	3	4	5	10	21	24	54
---	---	---	---	----	----	----	----

i=7 k=j=8

Quicksort(A,7,7)

1	3	4	5	10	21	24	54
---	---	---	---	----	----	----	----

i=7 j=7

Quicksort(A,9,8)

1	3	4	5	10	21	24	54
---	---	---	---	----	----	----	----

i=9 j=8

- Mô tả quá trình sắp xếp dãy số sau đây bằng thuật toán sắp xếp QuickSort
- 45,24, 432, 23, 876, 34, 789, 4



# Thời gian chạy



- Thủ tục partition kiểm tra tất cả các phần tử trong mảng nhiều nhất một lần, vì vậy nó mất thời gian tối đa là  **$O(n)$** .
- Thủ tục partition sẽ chia phần mảng được sắp thành 2 phần.
- Mặt khác cần chia liên tiếp  **$n$**  phần tử thành hai phần thì số lần chia nhiều nhất là  **$\log_2 n$**  lần.
- Vậy thời gian chạy của thuật toán QuickSort là  **$O(n \log n)$**

# Hết