



# Sắp xếp



# Nội dung

- Độ phức tạp thuật toán
- Tiếp cận chia để trị
- Sắp xếp trộn
- Sắp xếp nhanh quicksort

# Độ phức tạp thuật toán

- Tài nguyên cần thiết để giải bài toán
  - Thời gian thực hiện
  - Bộ nhớ sử dụng
- Khảo sát độ phức tạp
  - Phụ thuộc vào kích thước dữ liệu
  - Trường hợp xấu nhất
  - Trường hợp trung bình

# Thời gian thực hiện

- Số lượng phép toán cơ bản
  - Xấp xỉ bằng một hàm của kích thước dữ liệu
- Tìm kiếm nhị phân
  - Số lượt chia đôi:  $\log(n)$
- Thuật toán sắp xếp cơ bản
  - Phép so sánh, đổi chỗ
  - Thời gian thực hiện  $t(n) = c \times n^2$

# Độ phức tạp


- Nếu độ phức tạp lớn hơn hàm tuyến tính, ta có thể thấy mối quan hệ

$$t(a + b) > t(a) + t(b)$$

- Nếu tổng hợp lời giải từ các phần nhỏ không phức tạp (*trivial problem*) thì việc chia nhỏ bài toán sẽ giảm độ phức tạp
  - Tiếp cận chia để trị (*divide and conquer*)

# Chia để trị: ứng dụng cho sắp xếp

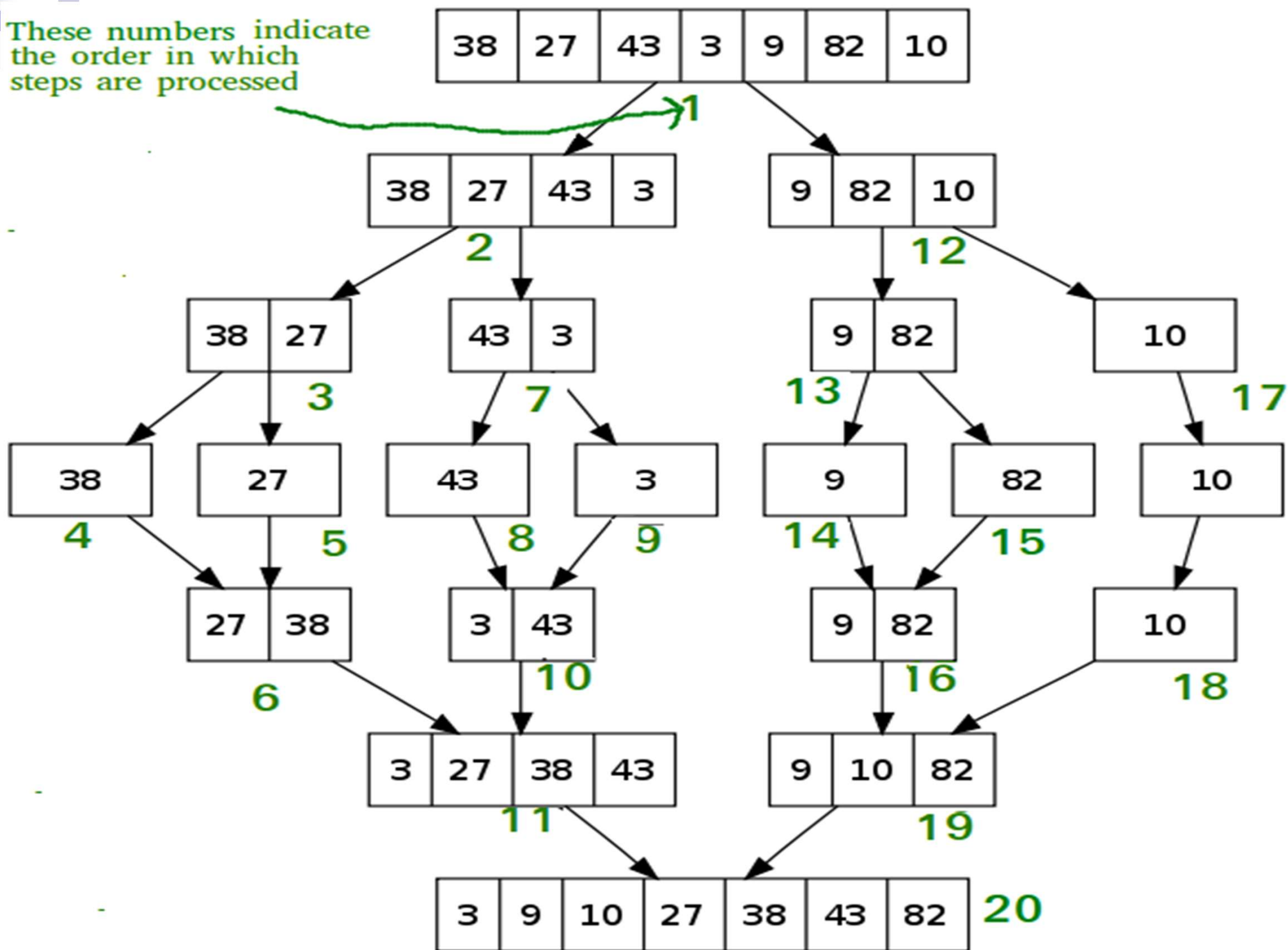
- Chia đôi một dãy (chia nhỏ bài toán) và ghép 2 dãy đã sắp xếp là bài toán không phức tạp
- Hình thành cách tiếp cận đệ qui
  - Chia dãy
  - Sắp xếp các dãy con (bằng cách lặp lại việc chia dãy nếu số phần tử  $> 1$ )
  - Ghép 2 dãy con đã sắp xếp
- Các thuật toán thông dụng
  - Sắp xếp trộn (merge sort)
  - Sắp xếp nhanh quicksort



# Sắp xếp trộn (merge sort)

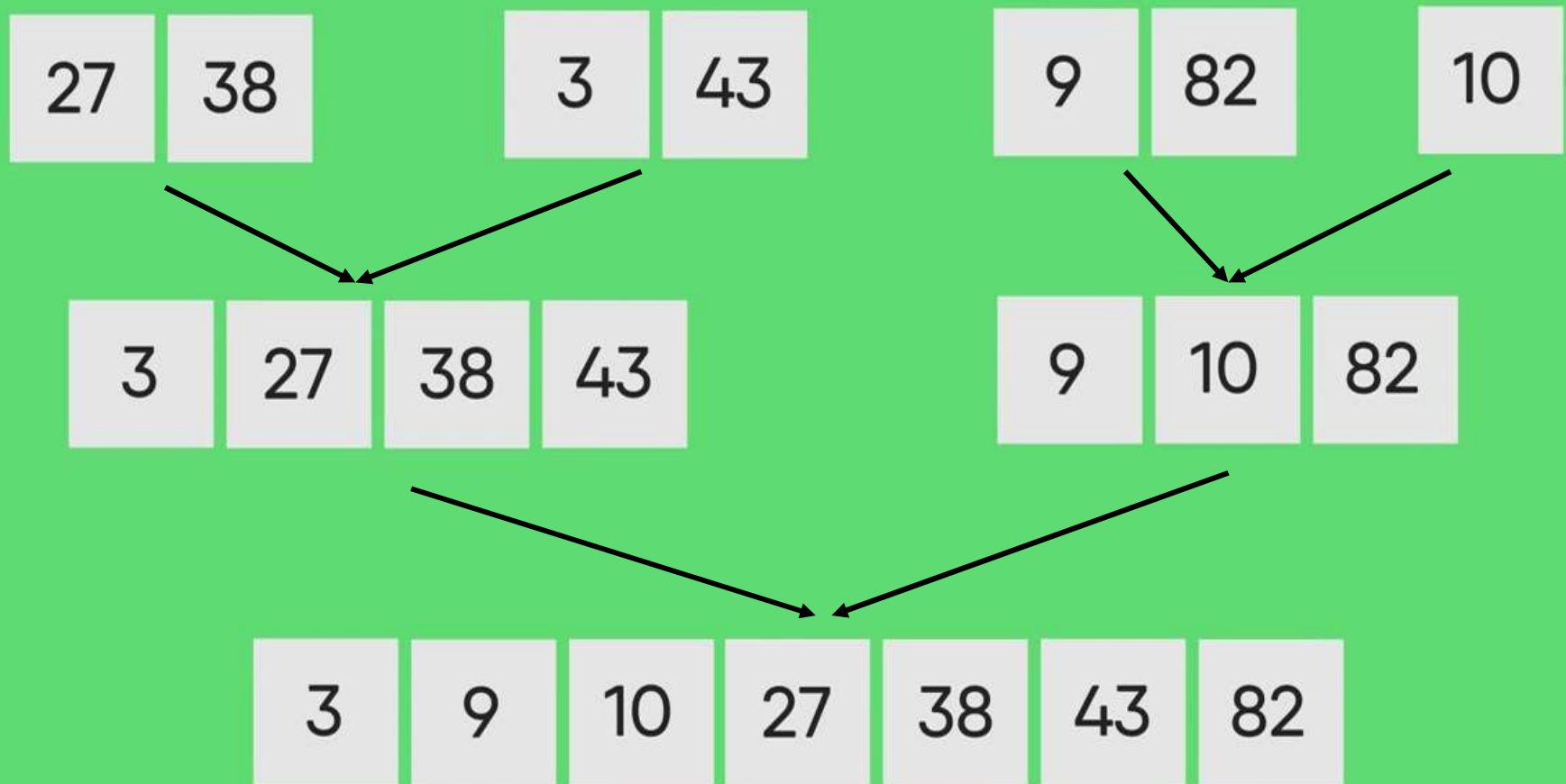
- Nếu mảng có nhiều hơn một phần tử thì chia đôi mảng thành hai mảng A, B
- Sắp xếp 2 mảng A, B bằng cùng thuật giải
- Trộn hai mảng A, B đã được sắp xếp

These numbers indicate the order in which steps are processed





# Trộn các mảng con





# Sắp xếp trộn

- Chia mảng một cách cơ học (đơn giản)
- Trộn mảng cần dùng mảng phụ
- Cài đặt
  - Đệ qui (top down)
  - Lặp tăng dần (bottom up)

# Sắp xếp trộn: lặp tăng dần

38 27 43 3 9 82 10

27 38 3 43 9 82 10

3 27 38 43 9 10 82

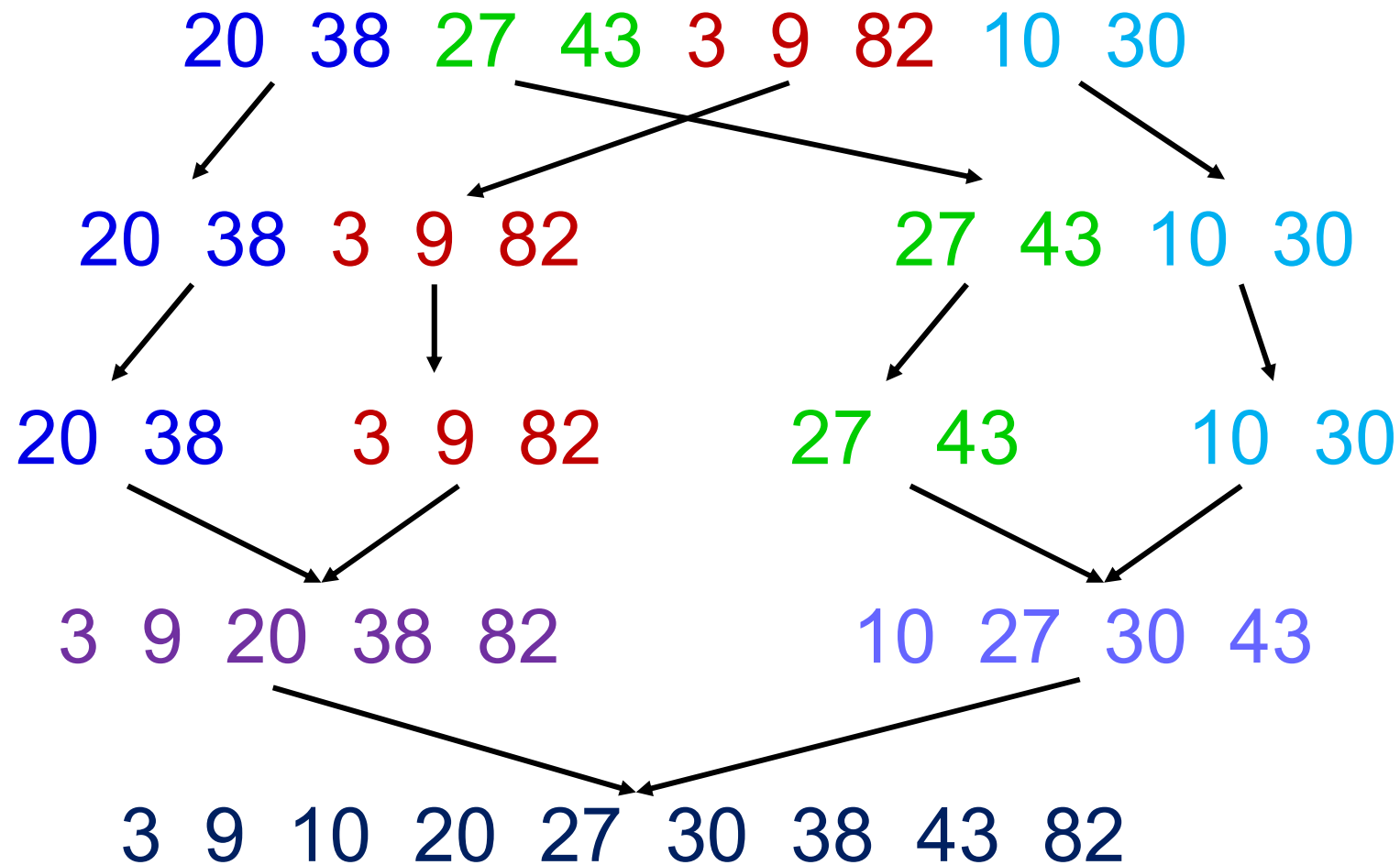
3 9 10 27 38 43 82



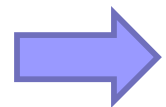
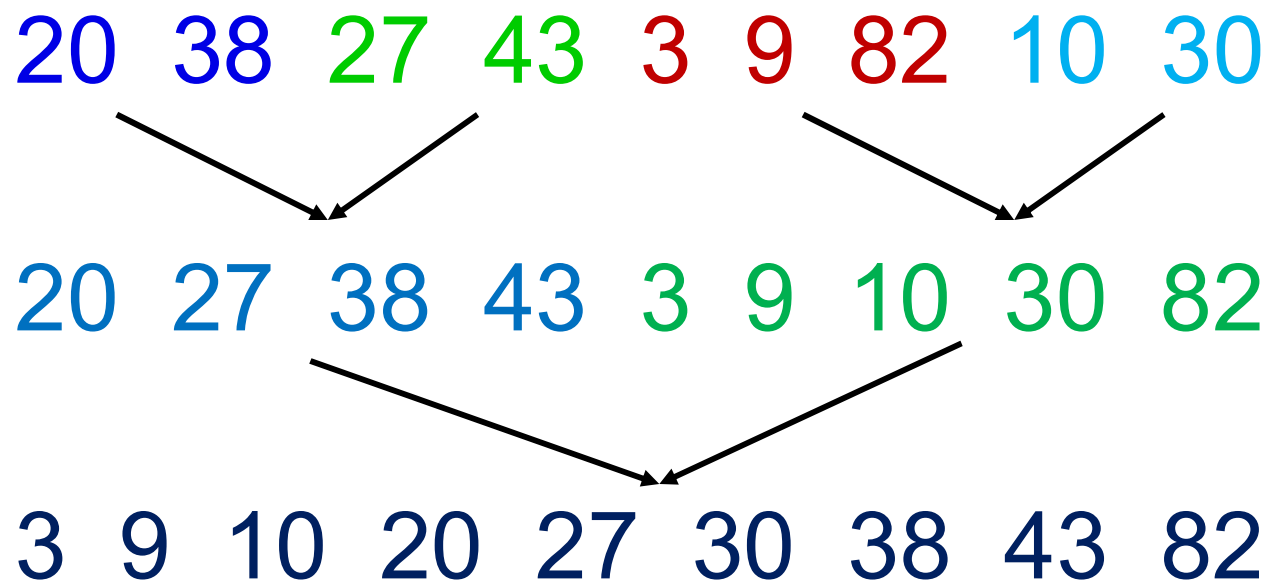
# Natural merge sort

- Cải thiện hiệu quả bằng cách tận dụng các đoạn mảng đã có thứ tự
- Chia mảng dựa trên các đoạn mảng đã có thứ tự
- Lặp lại với các mảng con

# Natural merge sort: đệ qui



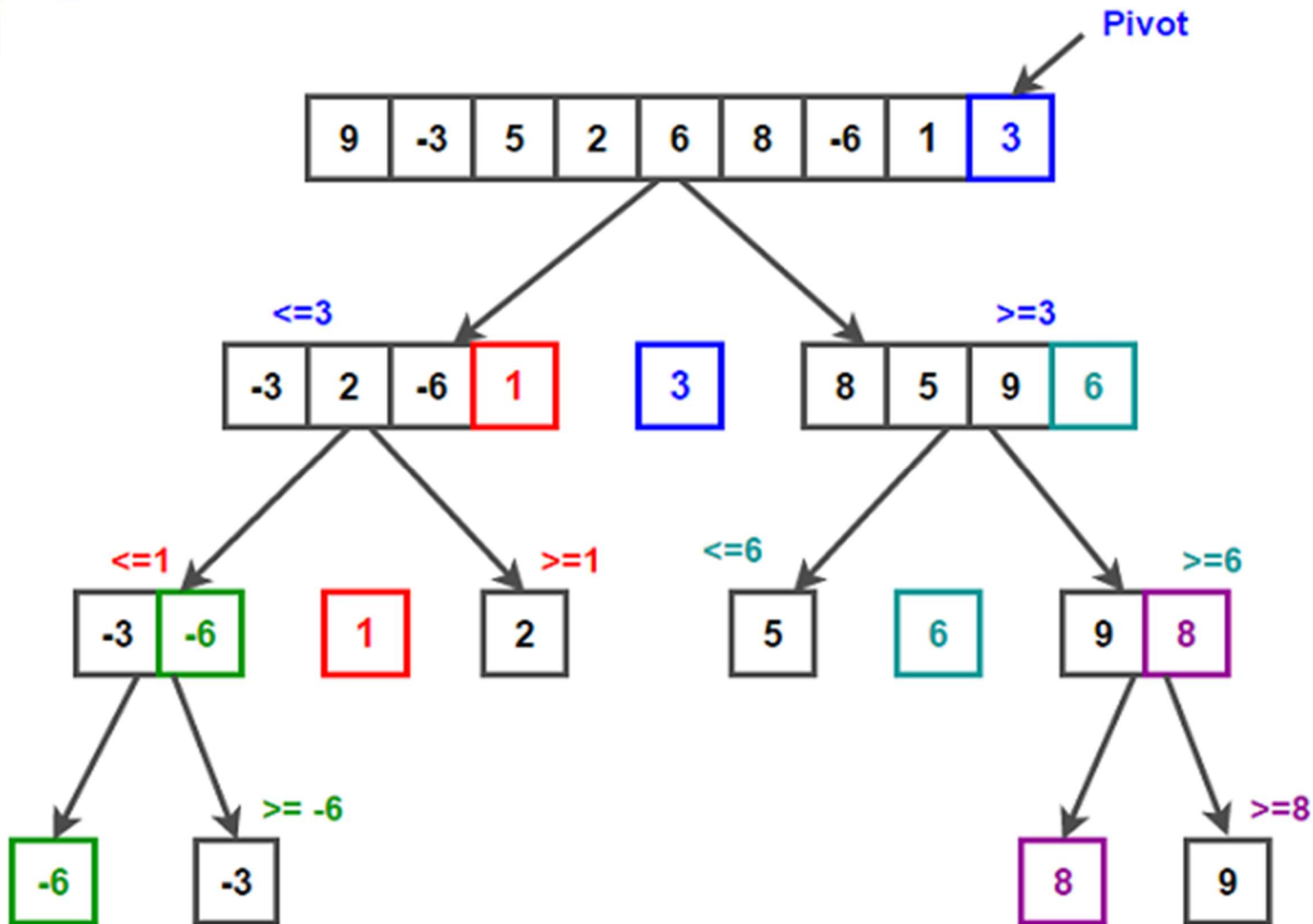
# Natural merge sort: lắp tăng dần



Cách tiếp cận tự nhiên của sắp xếp trộn

# Sắp xếp nhanh QuickSort

- Do giáo sư C. A. R. Hoare phát triển
- Dựa trên tiếp cận chia để trị
- Được đánh giá là thuật toán hiệu quả nhất
- Khái niệm cơ bản
  - Chọn một giá trị chốt  $p$  (pivot)
  - Chia mảng thành 2 mảng con, một mảng gồm các phần tử nhỏ hơn (hoặc bằng  $p$ ), mảng kia lớn hơn  $p$
  - Đặt  $p$  vào vị trí giữa 2 mảng con
  - Lặp lại thuật toán với các mảng con





# Quicksort

## ■ Thuật toán

```
quicksort(list, left, right) {  
    if (left < right) {  
        partition(list, left, right, j);  
        quicksort(list, left, j-1);  
        quicksort(list, j+1, right);  
    }  
}
```

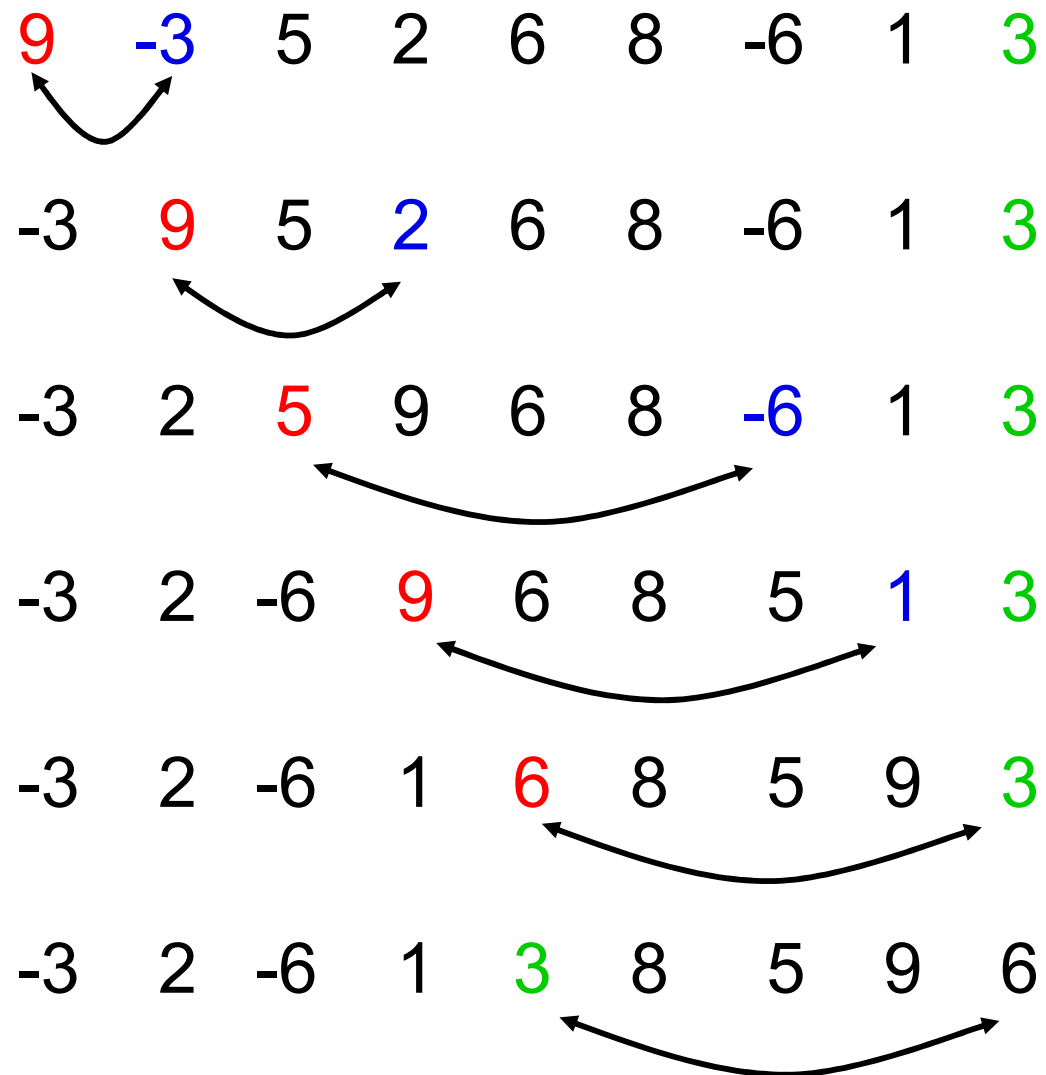
## ■ Mấu chốt ở giải thuật phân chia mảng

- Cách phân chia (duyệt nổi bọt, duyệt 2 chiều)
- Cách chọn chốt (đầu mảng, cuối mảng, giữa mảng)

# Quicksort: phân chia

- Có nhiều cách tiếp cận
- Tiếp cận dạng nổi bọt
  1. Đặt  $pivot = list[right]$ ;  $i = left - 1$ ;
  2. Cho  $j$  chạy từ  $left$  đến  $right - 1$ , nếu  $list[j] < pivot$ 
    1.  $i++$ ;
    2. Đổi chỗ  $list[i] \leftrightarrow list[j]$ ;
  3. Đổi chỗ  $list[i] \leftrightarrow list[right]$

# Quicksort: phân chia

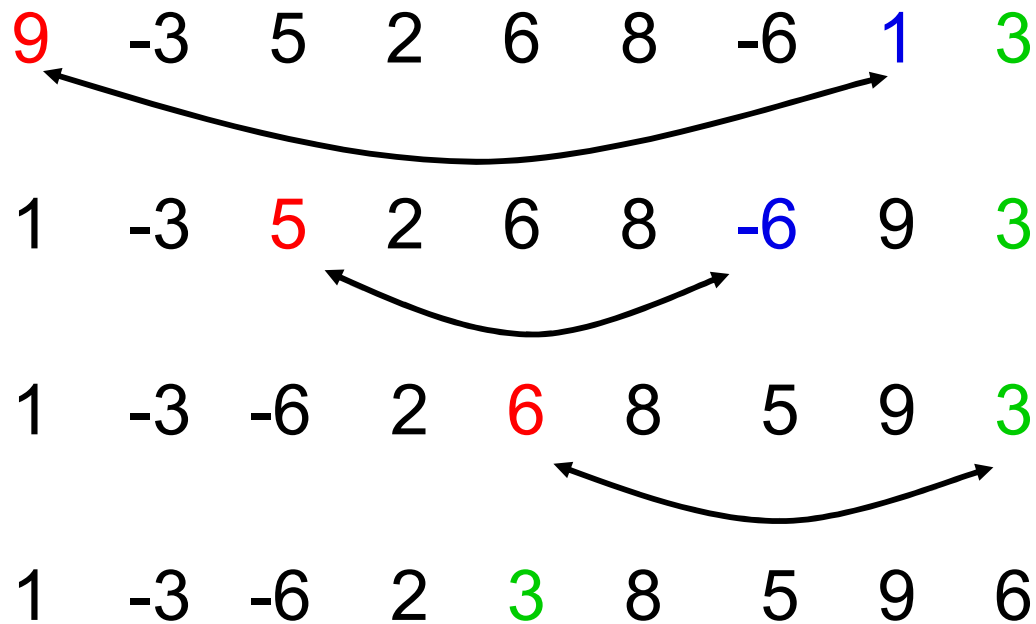


# Quicksort: phân chia

## ■ Tiếp cận duyệt từ hai đầu mảng

1. Đặt  $pivot = list[right]$
2. Dùng 2 biến chỉ số  $i = left$  và  $j = right$ , trong khi  $i < j$ 
  1. tăng  $i$  until  $list[i] \geq pivot$ ;
  2. giảm  $j$  until  $list[j] \leq pivot$ ;
3. Đổi chỗ  $list[i] \leftrightarrow list[j]$  và lặp lại bước 2
4. Đổi chỗ  $list[i] \leftrightarrow list[right]$

# Quicksort: phân chia





# Quicksort vs. merge sort

- Merge sort chia đơn giản, trộn *phức tạp*, phải dùng mảng phụ, có thể cài đặt không đệ qui
- Quicksort chia phức tạp, không cần trộn (ghép) kết quả
- Quicksort được xem là hiệu quả nhất

# Độ phức tạp và ký pháp *big O*

- Tài nguyên cần dùng để giải bài toán được xấp xỉ bằng hàm của kích thước dữ liệu  $f(n)$
- Độ phức tạp được qui ước là bậc lớn nhất của hàm và bỏ qua hệ số

$$f(n) = 3n^2 + 5n \rightarrow O(n^2)$$

- Binary search:  $O(\log(n))$
- Selection sort, bubble sort:  $O(n^2)$
- Quick sort, merge sort ???



# Độ phức tạp

- Merge sort?

- Quicksort?





# Đánh giá độ phức tạp thời gian

- Câu lệnh đơn
- Khối lệnh
- Các khối lệnh tuần tự
- Rẽ nhánh
- Vòng lặp
- Độ quy



# Đánh giá độ phức tạp bộ nhớ

- Các biến đơn
- Mảng
- Gọi hàm
- đệ quy

# Đặc trưng của thuật toán

- Tính đơn trị (đơn định)
  - Kết quả giống nhau với cùng đầu vào
- Tính hữu hạn (tính dừng)
  - Đảm bảo kết thúc sau hữu hạn bước
- Tính chính xác
  - Diễn giải được chính xác (và đơn nghĩa) thành các bước máy tính thực hiện được
- Tính tổng quát
  - Áp dụng được cho một lớp bài toán



# Tổng kết

- Chiến lược chia để trị
- Khái niệm về độ phức tạp thuật toán
- Các thuật toán sắp xếp hiệu quả
  - Merge sort, quicksort



# Tự học

- Shell sort
- Radix sort
- Bucket sort



# Bài tập

- Phân tích độ phức tạp của các thuật toán merge sort, quicksort
  - Trường hợp trung bình
  - Trường hợp xấu nhất

# Bài tập/thực hành

- Cài đặt các thuật toán sắp xếp trộn, sắp xếp trộn tự nhiên bằng 2 cách đệ quy và lặp tăng dần
- Cài đặt thuật toán quicksort với hai cách phân chia mảng
- Sinh dữ liệu mảng ngẫu nhiên 1k, 10k, 100k, 1M, 10M phần tử và chạy các thuật toán sắp xếp đã học với các dữ liệu này, so sánh thời gian chạy



# Chuẩn bị

- Các cấu trúc dữ liệu
  - ☐ Danh sách liên kết
  - ☐ Hàng đợi
  - ☐ Ngăn xếp