1. **Quicksort**

Quicksort là thuật toán chia để trị.

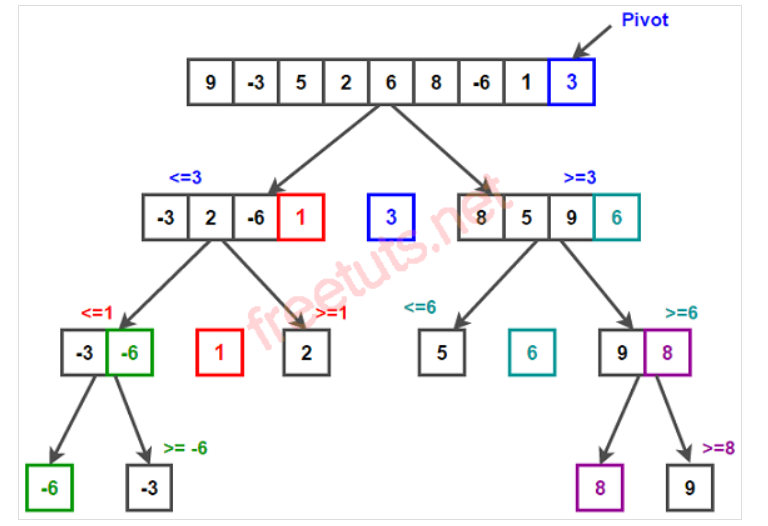
Cách hoạt động:

* Chọn 1 phần tử pivot (đánh dấu) và phân vùng mảng dựa theo pivot đã chọn
* Nếu phần tử nhỏ hơn xếp trước pivot, lớn hơn xếp sau pivot

Cách chọn pivot:

* Phần tử đầu
* Phần tử cuối
* Phần tử giữa
* Phần tử ngẫu nhiên

Ví dụ minh họa:



* Chọn số 3 làm pivot
* Mảng được chia làm hai phần:
* Phần bên trái nhỏ hơn hoặc bằng 3
* Phần bên phải lớn hơn hoặc bằng 3
* Tiếp tục chọn pivot cho mỗi phần.
* Phần bên trái có pivot là số 1 và phần bên phải có pivot là số 6.
* Cứ như vậy chúng ta sẽ chia phần ra cho đến khi không chia được nữa và sắp xếp theo đúng thứ tự.
* Hiện tại em chọn cách chọn pivot ngẫu nhiên vì: Tránh được trường hợp xấu nhất là khi mảng đã sắp xếp hoặc gần như sắp xếp

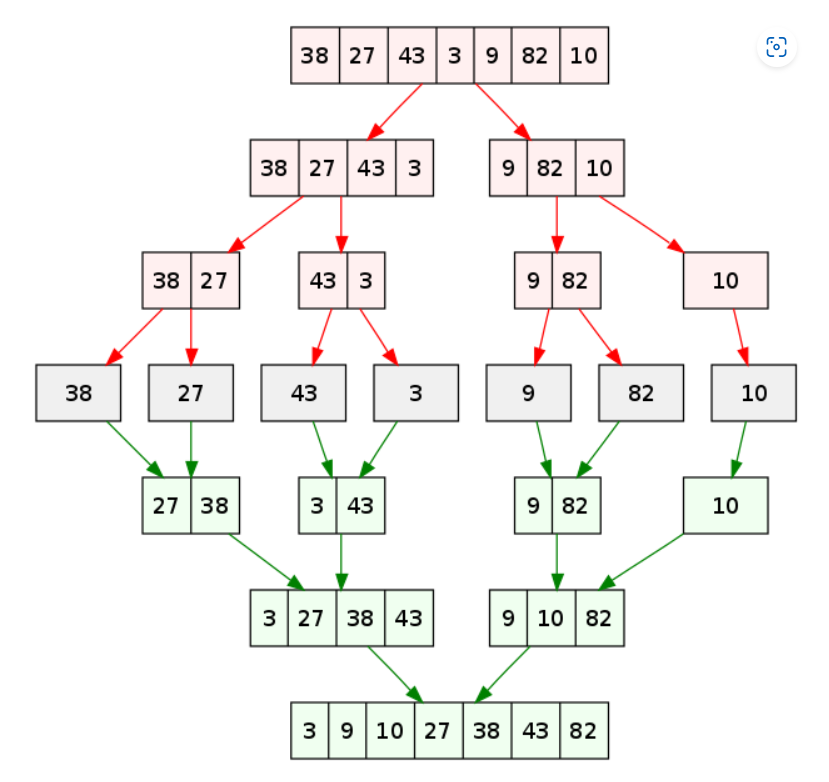
Độ phức tạp tính toán là O(n log n)

1. **MergeSort**

MergeSort là thuật toán chia để trị.

Cách hoạt động:

* Chia một danh sách ban đầu thành các nửa độc lập chia đến khi mảng chỉ còn 1 phần tử.
* Sắp xếp mảng con sử dụng đệ quy.
* Thực hiện merge: hai mảng con đã được sắp xếp được trộn lại với nhau để tạo ra một mảng mới đã được sắp xếp.
* Sau quá trình merge kết thúc sẽ tạo ra mảng cần tìm



Độ phức tạp: O(nlog(n))

1. **BubbleSort**

Thuật toán này hoạt động bằng cách so sánh lần lượt từng cặp phần tử liền kề trong mảng và hoán đổi chúng nếu chúng không được sắp xếp theo thứ tự mong muốn. Quá trình này được lặp lại cho đến khi không còn cặp nào cần hoán đổi nữa.

Cách hoạt động của Bubble Sort:

* Bắt đầu từ phần tử đầu tiên của mảng, so sánh nó với phần tử kế tiếp.
* Nếu phần tử hiện tại lớn hơn phần tử kế tiếp, hoán đổi chúng.
* Tiếp tục so sánh và hoán đổi các phần tử kế tiếp cho đến khi đi đến phần tử cuối cùng của mảng.
* Sau khi hoàn thành một lượt so sánh và hoán đổi, phần tử lớn nhất sẽ được đưa về cuối mảng.
* Lặp lại các bước trên cho tất cả các phần tử còn lại của mảng, trừ phần tử đã được sắp xếp ở cuối.
* Quá trình lặp lại cho đến khi không còn cặp nào cần hoán đổi nữa.

Độ phức tạp: O (n\*n) trong trường hợp mảng đầu vào sắp xếp ngược

1. **Binarysort**

Binarysort là một biến thể của thuật toán Insertion Sort

Cách thức hoạt động:

* Chèn một phần tử vào vị trí đúng trong mảng đã sắp xếp (sử dụng tìm kiếm nhị phân)
* Sử dụng tìm kiếm nhị phân để giảm số lượng so sánh trong kiểu Insertion Sort.
* Thực hiện sắp xếp chèn: sử dụng tìm kiếm nhị phân để tìm vị trí thích hợp để chèn mục đã chọn ở mỗi lần lặp

Trong Insertion Sort, sắp xếp phải mất O(i) (ở lần lặp thứ i) trong trường hợp xấu nhất. chúng ta có thể giảm nó xuống O(logi) bằng cách sử dụng tìm kiếm nhị phân .

Độ phức tạp: O(n mũ 2 ) vì có trường hợp cần có hàng loạt hoán đổi cho mỗi lần chèn.

1. **InsertionSort**

Thuật toán sắp xếp bằng cách so sánh phần tử hiện tại với các phần tử đã được sắp xếp và được đặt chính xác trong phần được sắp xếp. Phần tử chọn đầu tiên sẽ luôn là phần tử thứ 2 trong mảng.

Cách thức hoạt động:

* Bắt đầu từ phần tử thứ hai (vị trí 1), so sánh với các phần tử ở bên trái, trong phần đã được sắp xếp.
* Nếu phần tử đang xét nhỏ hơn một phần tử ở bên trái của nó, nó sẽ được di chuyển sang phải để tạo ra một vị trí trống, và các phần tử lớn hơn sẽ được dịch chuyển sang phải một vị trí.
* Khi không còn phần tử nào bên trái lớn hơn phần tử đang xét hoặc khi tìm thấy một phần tử bên trái nhỏ hơn nó, phần tử đang xét sẽ được chèn vào vị trí trống.



Độ phức tạp: O(n mũ 2 ) vì có trường hợp mảng đầu vào sắp xếp ngược chiều

Tổng kết

* **Quicksort:**
* Độ phức tạp thời giant rung bình là O(nlogn).
* Trường hợp xấu nhất có thể lên đến O(n mũ 2) nếu pivot được chọn không tốt.
* Sử dụng rất tốt cho tập dữ liệu lớn.
* **MergeSort**:
* Độ phức tạp thời gian trung bình và xấu nhất đều là O(nlogn).
* Cần không gian bộ nhớ để lưu trữ các mảng phụ trong quá trình sắp xếp,
* Không phù hợp cho các tập dữ liệu lớn nếu không có đủ bộ nhớ.
* **BubbleSort**:
* Độ phức tạp thời gian trung bình và xấu nhất đều là O(n mũ 2).
* Xử lý chậm trên các tập dữ liệu lớn.
* **BinarySort**:
* Độ phức tạp thời gian trung bình là O(n mũ 2)
* Một số trường hợp có thể nhanh hơn InsertionSort.
* **InsertionSort**:
* Độ phức tạp thời gian trung bình và xấu nhất là O(n mũ 2).
* Phù hợp cho các tập dữ liệu nhỏ hoặc đã gần sắp xếp
* Không hiệu quả trên các tập dữ liệu lớn.
* Dựa trên đánh giá này, nếu cần một thuật toán sắp xếp hiệu quả cho các tập dữ liệu lớn, MergeSort hoặc Quicksort là sự lựa chọn tốt. Tuy nhiên, nếu bộ nhớ là một vấn đề, thể cân nhắc sử dụng Quicksort với việc tối ưu hóa cách chọn pivot.

**Môi trường:**

Back end:

* spring-boot-starter-parent : version 3.0.5
* java jdk 20

Front end: python 3.9.0