

# Hàm np.sort() trong NumPy

## 1. Bản chất của hàm sort() trong NumPy

Thay vì chỉ dùng một thuật toán duy nhất, NumPy sử dụng các **thuật toán lai (Hybrid Algorithms)**. Tùy vào cấu trúc dữ liệu và yêu cầu về hiệu năng, NumPy cho phép lựa chọn phương pháp tối ưu:

- Mặc định:** NumPy sử dụng **Quicksort**.
- Python thuần (list.sort()):** Sử dụng **Timsort** (lai giữa Merge và Insertion Sort).
- Đặc điểm:** Tối ưu hóa ở tầng thấp (ngôn ngữ C) để đạt tốc độ xử lý mảng cực lớn.

## 2. Khi nào sử dụng các thuật toán khác nhau?

Việc lựa chọn thuật toán phụ thuộc vào ba yếu tố: **Tốc độ**, **Bộ nhớ** và **Tính ổn định (Stable)**.

Thuật toán	Tham số kind	Sử dụng khi nào?	Đặc điểm chính
Quicksort	'quicksort'	Dữ liệu ngẫu nhiên, cần tốc độ nhanh nhất.	Nhanh nhất trên thực tế nhưng không ổn định (Unstable).
Mergesort	'mergesort'	Cần tính <b>Stable</b> (giữ nguyên thứ tự phần tử bằng nhau).	Luôn ổn định $O(n \log n)$ nhưng tốn thêm RAM.
Heapsort	'heapsort'	Khi bộ nhớ hạn chế hoặc cần độ an toàn tuyệt đối.	Đảm bảo $O(n \log n)$ trong mọi trường hợp xấu nhất.
Timsort	'stable'	Dữ liệu thực tế thường đã sắp xếp một phần.	Kết hợp ưu điểm của Merge & Insertion Sort, tính ổn định cao.

## 3. Quy tắc lựa chọn nhanh (Rule of Thumb)

- Tốc độ ưu tiên:** Giữ mặc định (Quicksort).
- Dữ liệu có thứ tự sẵn:** Timsort/Stable là lựa chọn số 1.
- Yêu cầu bảo toàn thứ tự gốc:** Bắt buộc dùng Mergesort hoặc Stable.
- Hệ thống nhúng/RAM thấp:** Ưu tiên Heapsort vì không tốn bộ nhớ phụ trợ.

