

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Vũ Gia Bảo - 25520168

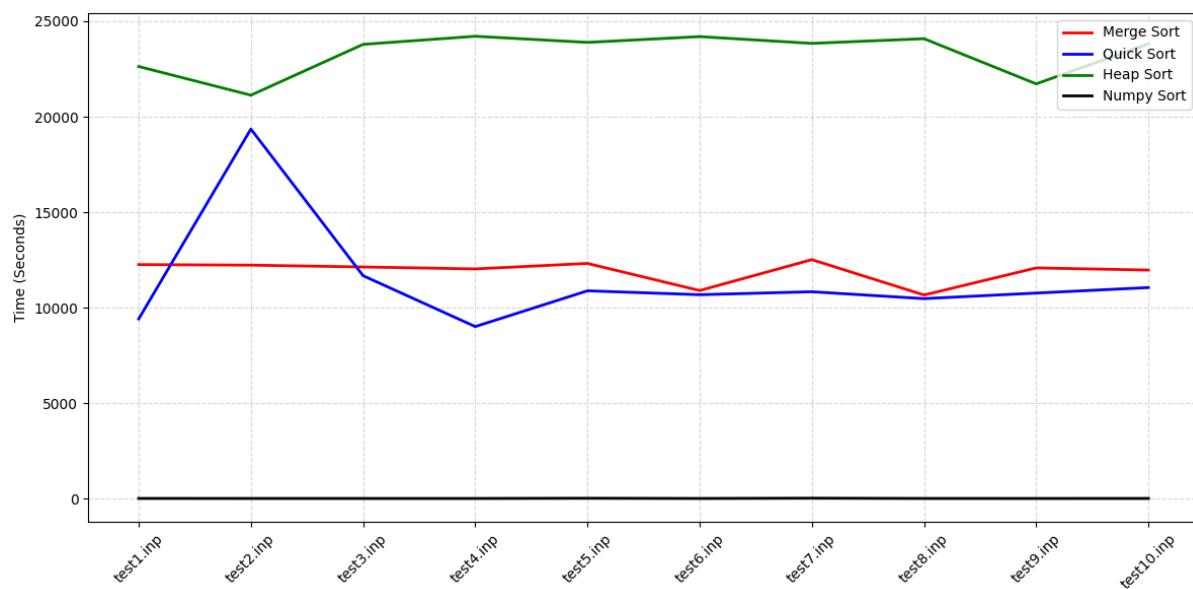
Nội dung báo cáo:

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

Dữ liệu	Thời gian thực hiện (ms)			
	Merge Sort	Quick Sort	Heap Sort	sort (numpy)
1 (số thực tăng dần)	12261.058772	9415.510871	22629.966607	19.689770
2 (số thực giảm dần)	12231.280983	19356.168282	21128.515120	17.877566
3 (số thực ngẫu nhiên)	12135.801229	11679.476538	23788.218038	17.701865
4 (số thực ngẫu nhiên)	12033.386235	9015.880348	24212.991697	17.607474
5 (số thực ngẫu nhiên)	12321.112735	10887.670325	23892.222328	28.624950
6 (số nguyên ngẫu nhiên)	10902.667811	10684.551238	24194.529384	17.707680
7 (số nguyên ngẫu nhiên)	12518.955430	10839.110043	23840.866892	31.792513
8 (số nguyên ngẫu nhiên)	10667.956008	10479.244289	24085.598092	18.057940
9 (số nguyên ngẫu nhiên)	12086.768524	10769.561918	21720.084910	17.490240
10 (số nguyên ngẫu nhiên)	11973.974897	11058.285350	23816.527442	18.008121

2. Biểu đồ (dường) thời gian thực hiện



II. Kết luận

1. Trường hợp dữ liệu có thứ tự (test 1, 2)

- Thời gian Quick Sort thực thi hai trường hợp này không lâu hơn các trường hợp dãy ngẫu nhiên vì chương trình Quick Sort được sử dụng có cách chọn pivot tốt (cụ thể là cách chọn median of three) nên tránh được trường hợp xấu nhất là $O(N^2)$.

- Merge Sort và Heap Sort thực thi hai trường hợp này nhanh hơn so với các trường hợp dãy ngẫu nhiên.

2. Trường hợp dữ liệu ngẫu nhiên (test 3-10)

- Số thực (test 3-5): Trên lý thuyết thì thời gian thực thi của các thuật toán sắp xếp sẽ tăng lên khi dữ liệu là ngẫu nhiên và thực tế đã chứng minh điều đó.
- Số nguyên (test 6-10): Tuy rằng về mặt lý thuyết thì số nguyên sẽ được xử lý nhanh hơn số thực, nhưng kết quả thể hiện rằng sự khác biệt này là không đáng kể.

3. Kết luận chung

- Chương trình Quick Sort được sử dụng có cách chọn pivot tốt (cụ thể là phương pháp median of three) nên thời gian thực thi nhanh hơn hai thuật toán sắp xếp truyền thống còn lại (trừ trường hợp dữ liệu giảm dần).
- Merge Sort thể hiện độ ổn định cao so với hai thuật toán truyền thống còn lại, thuật toán Heap Sort cũng thể hiện độ ổn định khá cao.
- Heap Sort được cài đặt thủ công (không sử dụng max heap từ các thư viện có sẵn, ví dụ như priority_queue từ thư viện STL trong C++) nên thời gian thực thi lâu hơn.
- Ba thuật toán sắp xếp truyền thống được cài đặt bằng Python, sử dụng các thao tác thông thường (nên Python phải chuyển đổi từ numpy array sang list mặc định) và đều sử dụng các hàm đệ quy nên thời gian chạy rất lâu so với hàm sort của thư viện numpy.
- Hàm sort của thư viện numpy được viết bằng C, sử dụng Timsort hoặc Introsort giúp kết hợp ưu điểm của nhiều thuật toán sắp xếp khác nhau (sử dụng mỗi thuật toán tùy vào trường hợp) và có các tối ưu hóa phần cứng khác nên có tốc độ vượt trội.

III. Thông tin chi tiết

- Báo cáo: File "Sorting Algorithms Report.pdf"
- Liên kết github: <https://github.com/icyalmond6727/Sorting-Algorithms>
- Dữ liệu thử nghiệm: Các bộ test được sinh ngẫu nhiên bằng thư viện numpy của Python