

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Vũ Gia Bảo - 25520168

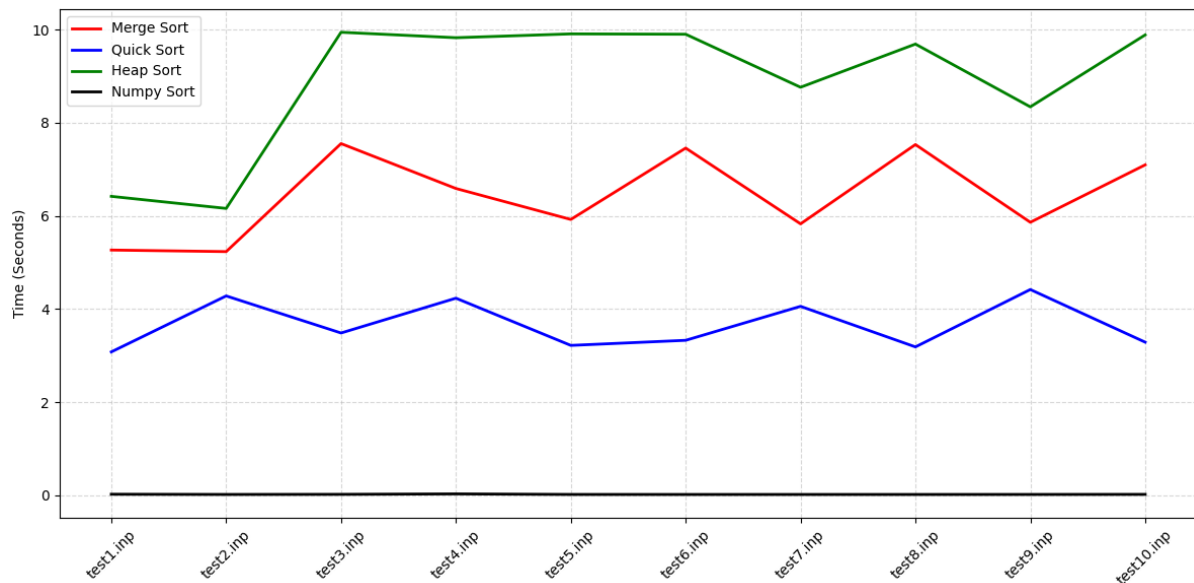
Nội dung báo cáo:

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

Dữ liệu	Thời gian thực hiện (s)			
	Merge Sort	Quick Sort	Heap Sort	sort (numpy)
1 (số thực tăng dần)	5.266541	3.080724	6.419083	0.025817
2 (số thực giảm dần)	5.232998	4.281989	6.159895	0.021055
3 (số thực ngẫu nhiên)	7.551447	3.486160	9.941009	0.023329
4 (số thực ngẫu nhiên)	6.587637	4.233437	9.823873	0.032915
5 (số thực ngẫu nhiên)	5.923088	3.220121	9.906065	0.021224
6 (số nguyên ngẫu nhiên)	7.455210	3.329411	9.898502	0.020520
7 (số nguyên ngẫu nhiên)	5.828925	4.058461	8.760461	0.020650
8 (số nguyên ngẫu nhiên)	7.530568	3.187732	9.687443	0.020648
9 (số nguyên ngẫu nhiên)	5.862585	4.419238	8.340154	0.021084
10 (số nguyên ngẫu nhiên)	7.094742	3.289530	9.884117	0.022689

2. Biểu đồ (đường) 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 nên thời gian thực thi nhanh hơn hai thuật toán sắp xếp truyền thống còn lại.
- Merge Sort thể hiện độ ổn định tương đối 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 và đều sử dụng các hàm đệ quy nên thời gian chạy lên tới vài giây.
- 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 hoá phần cứng khác nên có tốc độ vượt trội.

III. Thông tin chi tiết – link github

<https://github.com/icyalmond6727/Sorting-Algorithms>