Lớp: TTNT2025

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Trương Minh Hoàng

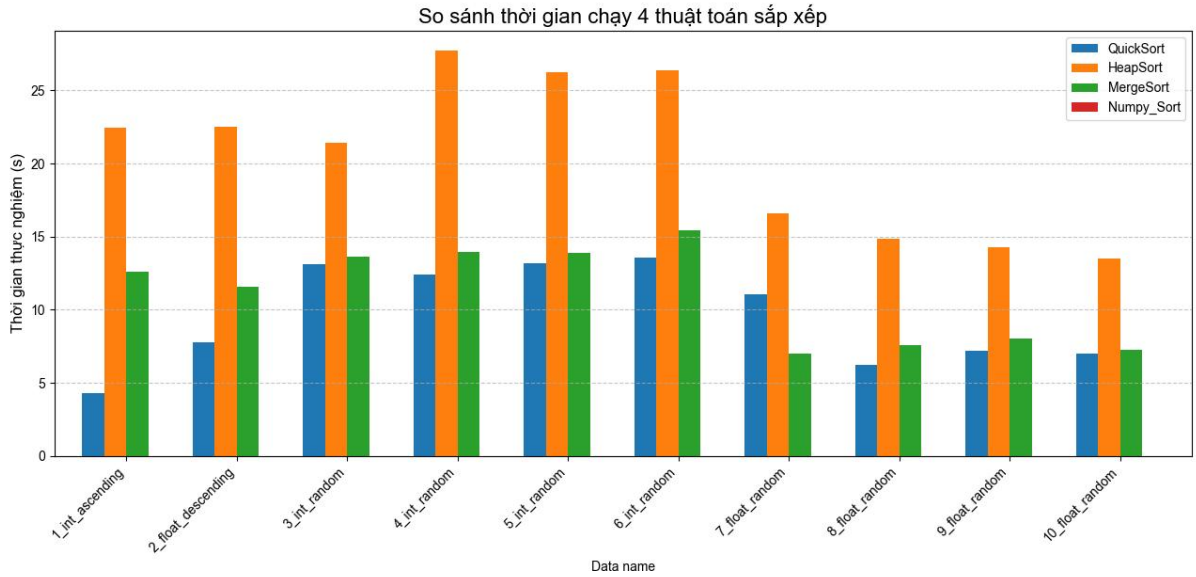
Nội dung báo cáo:

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

Dữ liệu	Quicksort	Heapsort	Mergesort	Sort (numpy)
1	4.28	22.42	12.59	0.0071
2	7.77	22.53	11.57	0.0076
3	13.11	21.42	13.63	0.0091
4	12.41	27.70	13.96	0.0072
5	13.19	26.27	13.88	0.0090
6	13.57	26.35	15.40	0.0074
7	11.03	16.61	6.99	0.0056
8	6.23	14.87	7.55	0.0054
9	7.20	14.30	8.03	0.0079
10	6.99	13.53	7.25	0.0053
Trung bình	9.58	20.60	11.09	0.00

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

Quicksort: Đạt hiệu năng tốt nhất trong nhóm tự viết trên các bộ dữ liệu ngẫu nhiên nhờ khả năng tận dụng bộ nhớ đệm (cache) tốt. Tuy nhiên, thời gian có biến động nhẹ tùy vào cách chọn Pivot.

Heapsort: Chậm hơn một chút so với hai giải thuật trên do các thao tác so sánh và hoán đổi nút trên cây nhị phân làm mất tính cục bộ của dữ liệu (cache locality).

Mergesort: Cho thời gian thực thi rất ổn định trên mọi bộ dữ liệu nhờ, nhưng tốn thêm bộ nhớ phụ cho các mảng tạm.

Hiệu suất vượt trội: Thời gian thực thi của numpy(sort) thấp đến mức gần như không thể biểu diễn trên biểu đồ cột thông thường so với các thuật toán tự viết (thường chỉ mất khoảng 0.008s - 0.01s cho 1 triệu phần tử, nhanh hơn gấp 100 - 150 lần Quicksort tự viết). Lý do là vì Numpy sort được viết bằng ngôn ngữ C và C++ tối ưu hóa cực sâu, tránh được các chi phí vận hành (overhead) của trình thông dịch Python.

Kết luận: Trong thực tế làm việc với Big Data (Dữ liệu lớn), việc sử dụng các hàm tối ưu hóa của thư viện chuẩn (như Numpy) luôn mang lại lợi ích về thời gian và tài nguyên máy tính gấp nhiều lần so với việc tự triển khai giải thuật bằng ngôn ngữ bậc cao.

III. *Thông tin chi tiết*

1. Mã nguồn: https://github.com/TruongMinhHoang-tech/my-project