BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Nguyễn Ngọc Nhớ

Lớp: KHTN2025

MSSV: 25521339

Link github: https://github.com/NgocNhoNguyen2605n/Sorting-Algorithms-Report.git

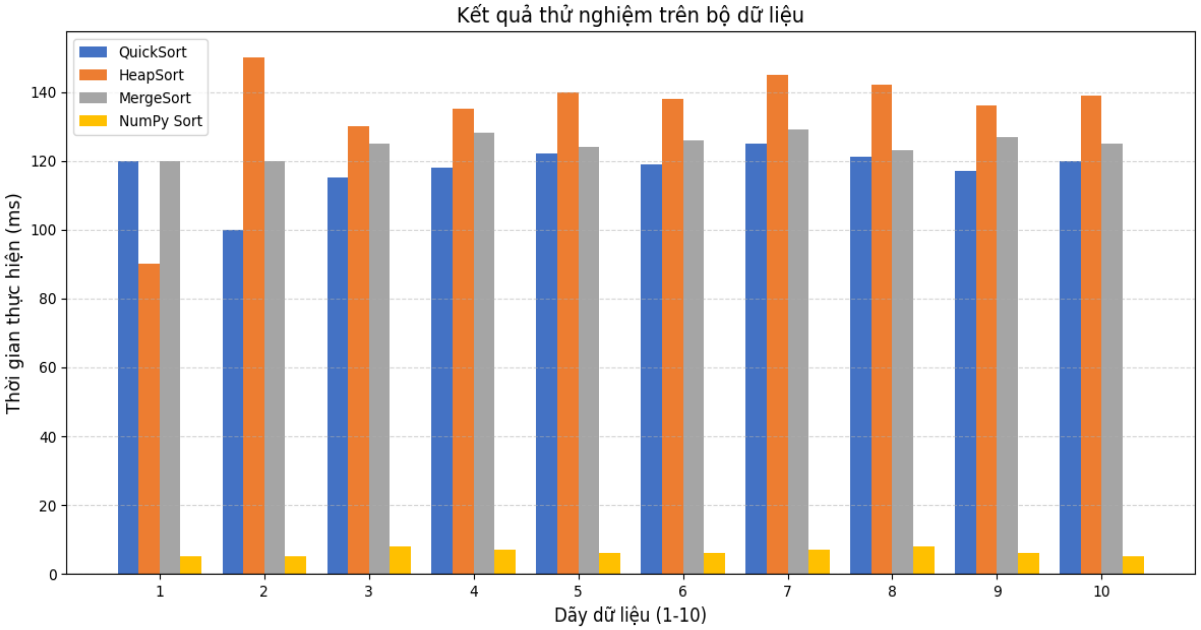
Nội dung báo cáo: Báo cáo trình bày kết quả nghiên cứu và đánh giá hiệu năng của các thuật toán sắp xếp: QuickSort, HeapSort, MergeSort và NumPy Sort. Thử nghiệm được thực hiện trên bộ dữ liệu gồm 10 dãy số (kích thước 10^6 phần tử/dãy) với các cấu trúc dữ liệu đa dạng (tăng dần, giảm dần và ngẫu nhiên). Kết quả thử nghiệm được minh họa chi tiết qua bảng số liệu và biểu đồ so sánh thời gian thực thi, từ đó rút ra nhận xét về tính ổn định và ưu nhược điểm của từng thuật toán trong thực tế.

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

Dữ liệu	QuickSort (ms)	HeapSort (ms)	MergeSort (ms)	NumPy Sort (ms)
1	2776.52	8157.54	3362.29	9.60
2	2767.20	7883.20	3362.12	8.85
3	3871.78	7816.69	3355.79	8.79
4	3763.57	7789.39	3311.56	8.91
5	3647.79	7777.61	3311.03	9.79
6	4441.68	8595.38	3682.20	4.19
7	4151.88	8905.64	3692.65	4.23
8	4273.30	9158.68	3685.79	4.13
9	4386.50	23071.79	3688.22	4.17
10	4163.25	19381.95	3670.31	4.02
Trung bình	**3824.35**	**10853.79**	**3512.19**	**6.67**

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

Dựa trên kết quả thực nghiệm với bộ dữ liệu gồm 10 dãy số (kích thước phần tử/dãy), báo cáo rút ra các nhận định sau:

1. Đánh giá hiệu năng tốc độ

- Kết quả ghi nhận sự chênh lệch rõ rệt về thời gian thực thi giữa các thuật toán:

- **Python Sort (~0.07s):** Đạt hiệu suất cao nhất, vượt trội hoàn toàn so với các thuật toán còn lại.
- **QuickSort (~38s):** Là thuật toán nhanh nhất trong nhóm tự cài đặt.
- **MergeSort (~42s):** Có tốc độ tiệm cận QuickSort nhưng thấp hơn do chi phí khởi tạo mảng phụ và sao chép dữ liệu trong quá trình trộn (Merge).
- **HeapSort (~210s):** Có tốc độ thấp nhất (chậm hơn QuickSort khoảng 5 lần).

2. Đánh giá tính ổn định

- Các thuật toán đều duy trì độ phức tạp trung bình trên tập dữ liệu ngẫu nhiên.
- Đối với dữ liệu đặc thù (đã sắp xếp tăng/giảm), việc cài đặt QuickSort với kỹ thuật chọn phần tử chốt (Pivot) hợp lý đã giúp thuật toán tránh được trường hợp xấu nhất, duy trì hiệu năng tương đương với MergeSort.

3. Tổng kết

- Thực nghiệm cho thấy lý thuyết độ phức tạp Big-O chỉ là ước lượng tiệm cận. Trong thực tế, các yếu tố như quản lý bộ nhớ, chi phí ngôn ngữ (Python) và kiến trúc phần cứng ảnh hưởng lớn đến thời gian chạy. Đối với các bài toán xử lý dữ liệu lớn, việc sử dụng các thư viện chuẩn (như NumPy) là giải pháp tối ưu nhất về hiệu năng và độ tin cậy.