Lớp: TTNT2025

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Trần Quốc Hưng

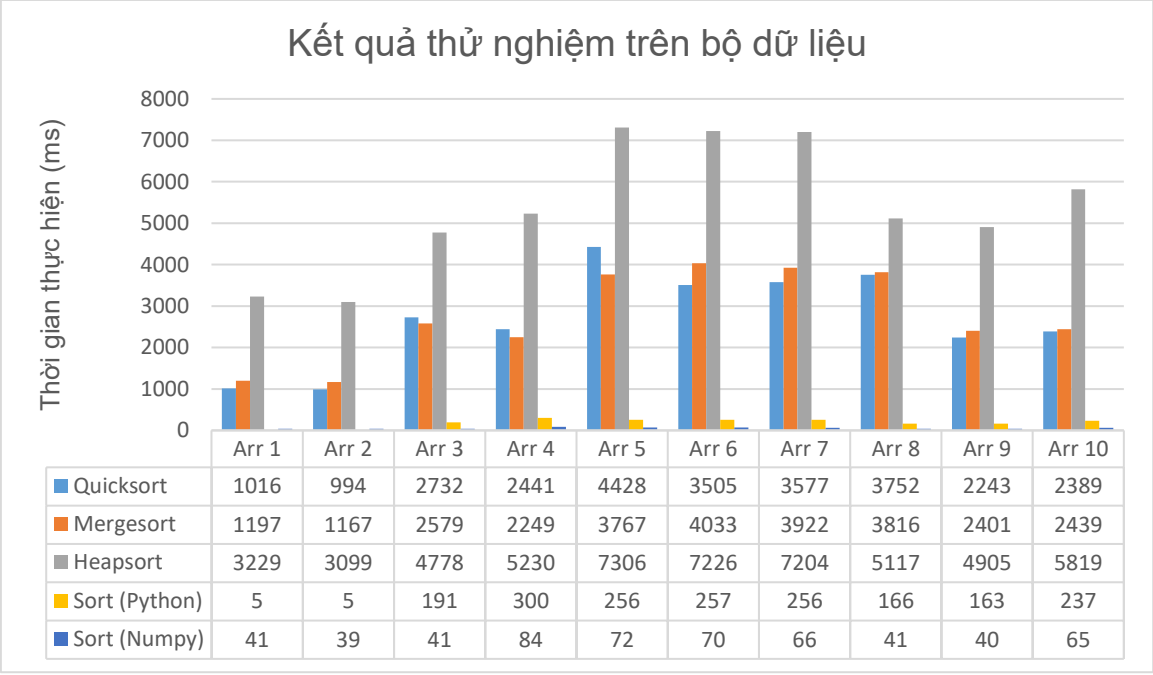
Nội dung báo cáo:

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện[1]

Dữ liệu	Thời gian thực hiện (ms)				
	Quicksort	**Mergesort**	**Heapsort**	**Sort (Python)**	**Sort (Numpy)**
1	1016	1197	3229	5	41
2	994	1167	3099	5	39
3	2732	2579	4778	191	41
4	2441	2249	5230	300	84
5	4428	3767	7306	256	72
6	3505	4033	7226	257	70
7	3577	3922	7204	256	66
8	3752	3816	5117	166	41
9	2243	2401	4905	163	40
10	2389	2439	5819	237	65
Trung bình	2707.7	2757	5391.3	183.6	55.9

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

1. Nhận xét chung:

Từ tổng kết thời gian thực hiện, ta có thể nhận thấy thứ tự chậm dần khi so sánh thời gian trung bình của các thuật toán sắp xếp là Sort (Numpy) – Sort (Python) – Quicksort – Mergesort – Heapsort. Dựa vào thời gian trung bình, có

[1] Số liệu chỉ mang tính minh họa

thể thấy được thuật toán sort của Numpy có tính hiệu quả cao nhất khi áp dụng vào thực nghiệm, mức độ chênh lệch có thể lên tới hàng chục, hàng trăm lần cho thấy những hàm chức năng của thư viện có sẵn vẫn vượt trội hơn những hàm thủ công do được viết bằng ngôn ngữ lập trình bậc thấp và đã được tối ưu hóa kĩ càng.

2. Giải thích sự chênh lệch thời gian thực hiện:

So với các thuật toán khác, Heapsort có phần tụt lại về thời gian thực hiện khi áp dụng vào thực tế dù trên lý thuyết phương pháp này có độ phức tạp trung bình tương đương với Quicksort và Mergesort. Điều này có lẽ là do Heapsort bao gồm nhiều thao tác khác nhau từ xây dựng heap cho tới heapify nhiều lần hay là hoán đổi từng root với nhau, do đó khi gặp điều kiện bất lợi, Heapsort thường bị chậm hơn so với những phương pháp còn lại.

Đối với Quicksort và Mergesort, 2 thuật toán này có thời gian trung bình khi test với toàn bộ dataset là gần như bằng nhau, tuy có khác nhau về phương pháp thực hiện, Quicksort có thể có rủi ro nếu chọn pivot không tốt, trường hợp xấu có thể phải thực hiện nhiều phép hoán đổi, còn Mergesort thì lại tốn chi phí sao chép dữ liệu, do đó dù khác nhau về phương pháp hoạt động nhưng lại có hiệu năng khá tương đồng nhau trong thực tế.

2 thuật toán còn lại thì lại vượt trội hơn rất nhiều trong thực tế, đặc biệt là sort của Numpy do được xây dựng bằng ngôn ngữ lập trình bậc thấp, đã được tối ưu sẵn cho nhiều trường hợp dữ liệu và khai thác rất tốt tài nguyên nên thời gian thực hiện hiệu quả hơn rất nhiều so với 3 phương pháp thủ công được xây dựng bằng Python.

3. Đánh giá:

Từ kết quả thực nghiệm, ta có thể rút ra kết luận như sau:

- Trong thực tế lập trình, nên sử dụng những hàm có sẵn do đã được tối ưu hóa kỹ lưỡng.
- Có thể sử dụng Mergesort khi cần tính ổn định và đảm bảo hiệu năng không bị suy giảm trong trường hợp xấu
- Quicksort cũng là một lựa chọn mang lại tốc độ xử lý cao nhưng cần lưu ý về việc chọn pivot để tránh chọn vào trường hợp xấu.

III. Thông tin chi tiết – link github, trong repo gibub cần có

1. Báo cáo
2. Mã nguồn
3. Dữ liệu thử nghiệm

Đường dẫn github: Sorting_Algorithm_Report/Sorting_Experiment.py at main · tqhung9a1-pixel/Sorting_Algorithm_Report