Lớp:

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện:

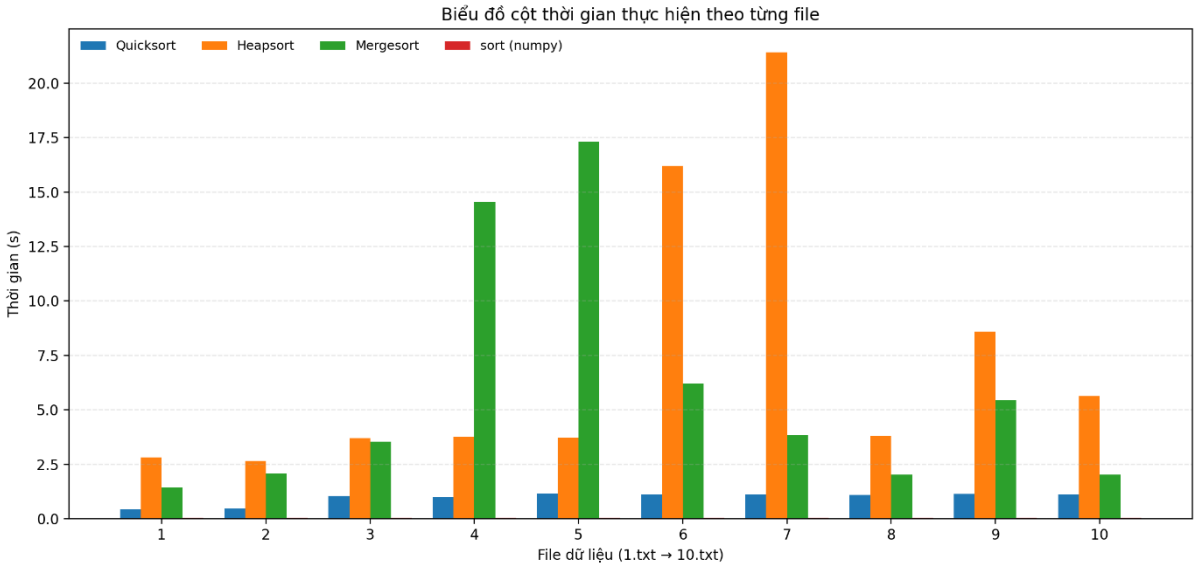
Nội dung báo cáo:

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện[1]

Dữ liệu	Thời gian thực hiện			
	Quicksort (s)	Heapsort (s)	Mergesort (s)	sort (numpy)
1	0.450300	2.808648	1.443946	0.048516
2	0.454594	2.650386	2.072027	0.048094
3	1.048836	3.701373	3.539100	0.045896
4	0.989718	3.760218	14.557731	0.047846
5	1.165053	3.721693	17.315488	0.046649
6	1.108334	16.199169	6.209370	0.046980
7	1.111454	21.426313	3.833174	0.049942
8	1.096743	3.795416	2.018298	0.046314
9	1.150283	8.598128	5.451875	0.046567
10	1.121385	5.624049	2.014111	0.048290
Trung bình	0.969670	7.228539	5.845512	0.047509

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

Qua quá trình chạy thử nghiệm trên 10 bộ dữ liệu, ta nhận thấy sự khác biệt rõ rệt về thời gian thực thi giữa bốn thuật toán sắp xếp: **Quicksort**, **Heapsort**, **Mergesort** và **sort(numpy)**.

[1] Số liệu chỉ mang tính minh họa

- **Quicksort** cho thời gian chạy tương đối ổn định và luôn nằm ở mức trung bình thấp. Đây là thuật toán có hiệu suất tốt trong đa số trường hợp thử nghiệm, không xuất hiện giá trị đột biến.
- **Heapsort** nhìn chung có thời gian thực hiện cao hơn Quicksort và dao động mạnh. Đặc biệt ở một số file (như file 6 và 7), thời gian tăng vọt cho thấy thuật toán nhạy cảm hơn với đặc điểm phân bố dữ liệu đầu vào.
- **Mergesort** thể hiện kết quả không đồng đều: đôi lúc nhanh, nhưng cũng có những file cho thời gian thực thi rất lớn, khiến hiệu suất trung bình của thuật toán cao hơn nhiều so với hai thuật toán trên.
- **sort(numpy)** vượt trội hoàn toàn khi thời gian xử lý ở tất cả các file đều rất thấp và ổn định. Đây là kết quả dễ lý giải vì numpy được viết bằng C và tối ưu hóa mạnh cho thao tác trên mảng.

Từ các kết quả trên, có thể rút ra rằng:

- Đối với các yêu cầu sắp xếp thông thường và dữ liệu ngẫu nhiên, **Quicksort** là lựa chọn cân bằng giữa tốc độ và ổn định.
- Nếu ưu tiên tốc độ tuyệt đối và làm việc với mảng số học lớn, **sort(numpy)** là phương án hiệu quả nhất.
- **Heapsort** và **Mergesort** tuy có tính chất tốt về lý thuyết (ổn định, độ phức tạp đảm bảo), nhưng trong thực nghiệm lại thiếu ổn định hơn và có thời gian chạy cao hơn đáng kể ở nhiều trường hợp.

Do đó, đối với bộ dữ liệu thử nghiệm hiện tại, **sort(numpy)** cho hiệu năng tốt nhất, trong khi **Quicksort** là thuật toán truyền thống có kết quả ổn định và phù hợp cho đa số ứng dụng lập trình thông thường

III. Thông tin chi tiết - https://github.com/bqa100507-spec/SortingAlgorithm