Lớp: IT003.Q21.TTNT

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Tạ Cao Nguyên Bảo

MSSV: 25520159

Nội dung báo cáo: Báo cáo này trình bày kết quả thử nghiệm 4 thuật toán sắp xếp phổ biến: **QuickSort**, **HeapSort**, **MergeSort** và **NumPySort** (hàm sort tích hợp của thư viện NumPy).

Thử nghiệm được thực hiện trên bộ dữ liệu gồm 10 dãy, mỗi dãy khoảng 1.000.000 phần tử, bao gồm:

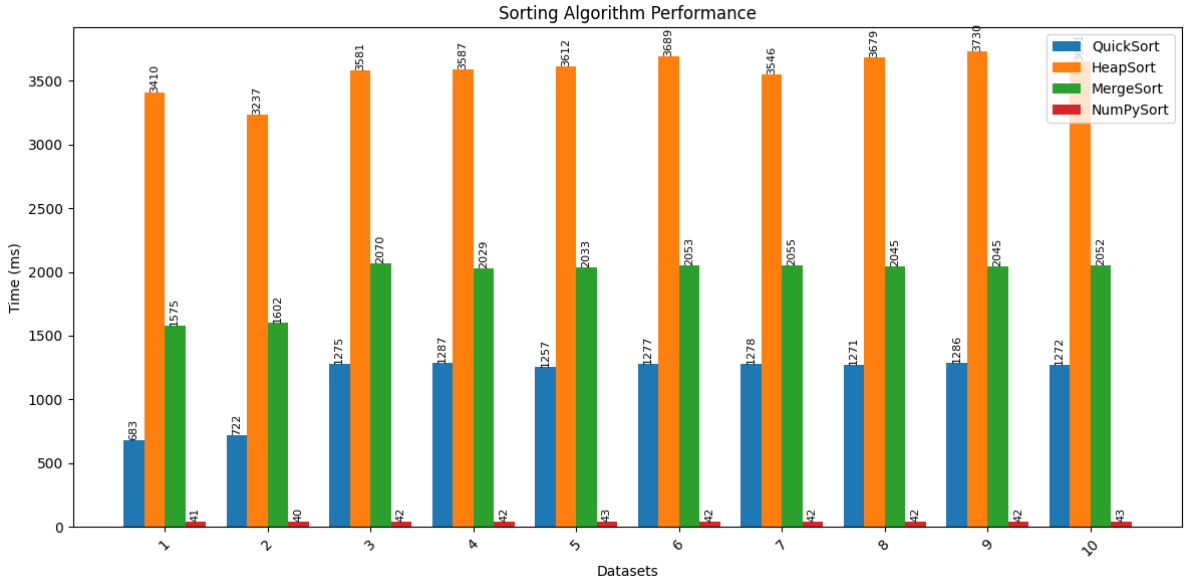
- Dãy 1: số thực, phân phối tăng dần
- Dãy 2: số thực, phân phối giảm dần
- Dãy 3 – 5: số thực, phân phối ngẫu nhiên
- Dãy 6 – 10: số nguyên, phân phối ngẫu nhiên

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện[1]

Dữ liệu	Thời gian thực hiện (ms)			
	Quicksort	**Heapsort**	**Mergesort**	**NumPySort**
1	683.21	3409.88	1575.03	41.14
2	721.56	3236.53	1602.22	40.17
3	1275.37	3581.16	2069.62	41.98
4	1287.02	3587.22	2028.77	42.23
5	1257.07	3612.27	2033.37	42.6
6	1277.06	3688.9	2053.25	42.17
7	1278.12	3546.09	2055.02	42.2
8	1270.93	3679.15	2045.42	42.33
9	1285.92	3729.64	2045.08	42.29
10	1271.87	3651.36	2052.46	42.82
Trung bình	1160.81	3582.22	1956.02	41.99

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

NumPySort - Vượt trội hoàn toàn

NumPySort cho thời gian thực thi cực kỳ thấp và ổn định trên toàn bộ 10 dãy, dao động chỉ trong khoảng 40–43 ms, nhanh hơn các thuật toán còn lại từ 30 đến 90 lần. Điều này đến từ việc NumPy được tối ưu hóa ở tầng C với thuật toán introsort, tận dụng tốt bộ nhớ đệm (cache) và vector hóa phần cứng.

QuickSort - Tốt nhất trong nhóm thuật toán Python

QuickSort cho thấy sự khác biệt rõ rệt giữa các loại dữ liệu. Trên dãy đã có thứ tự (dãy 1 tăng dần: ~683 ms, dãy 2 giảm dần: ~722 ms), thời gian nhanh hơn đáng kể so với dãy ngẫu nhiên (~1.257–1.287 ms). Điều này phản ánh đặc tính của pivot được chọn tốt khi dữ liệu có cấu trúc. Trên dữ liệu ngẫu nhiên (dãy 3–10), thời gian QuickSort tương đối ổn định và đồng đều nhau.

MergeSort - Ổn định, không bị ảnh hưởng bởi phân phối

MergeSort duy trì thời gian khá ổn định trong khoảng 1.575–2.070 ms trên mọi dãy. Đáng chú ý là MergeSort cũng nhanh hơn trên dãy đã có thứ tự (dãy 1–2: ~1.575–1.602 ms) so với dãy ngẫu nhiên (~2.000–2.070 ms), nhưng mức chênh lệch ít hơn so với QuickSort. Đây là minh chứng cho tính ổn định của MergeSort với độ phức tạp O(n log n) trong mọi trường hợp.

HeapSort - Chậm nhất, kém ổn định nhất

HeapSort có thời gian cao nhất trong tất cả các trường hợp, dao động từ 3.237 ms đến 3.889 ms. Đặc biệt, HeapSort không được hưởng lợi từ dữ liệu đã có thứ tự (dãy 1–2 vẫn ~3.237–3.410 ms). Nguyên nhân chính là cấu trúc heap khiến truy cập bộ nhớ không tuần tự (cache-unfriendly), dẫn đến nhiều cache miss hơn so với QuickSort và MergeSort.

Số thực vs. số nguyên - Không có sự khác biệt đáng kể

So sánh dãy 3-5 (float ngẫu nhiên) với dãy 6-10 (int ngẫu nhiên), thời gian thực thi của cả 4 thuật toán đều tương đương nhau. Điều này cho thấy chi phí xử lý không nằm ở kiểu dữ liệu mà chủ yếu đến từ số lần so sánh và hoán đổi phần tử.

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

1. Báo cáo: https://github.com/poundvn07/sorting-benchmark/blob/main/report/sorting_report.pdf
2. Mã nguồn: https://github.com/poundvn07/sorting-benchmark/blob/main/src/sorting_benchmark.py
3. Dữ liệu thử nghiệm: dữ liệu sinh ngẫu nhiên trong code.