Lớp: IT003.Q21.TTNT

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Lê Trịnh Quỳnh Như

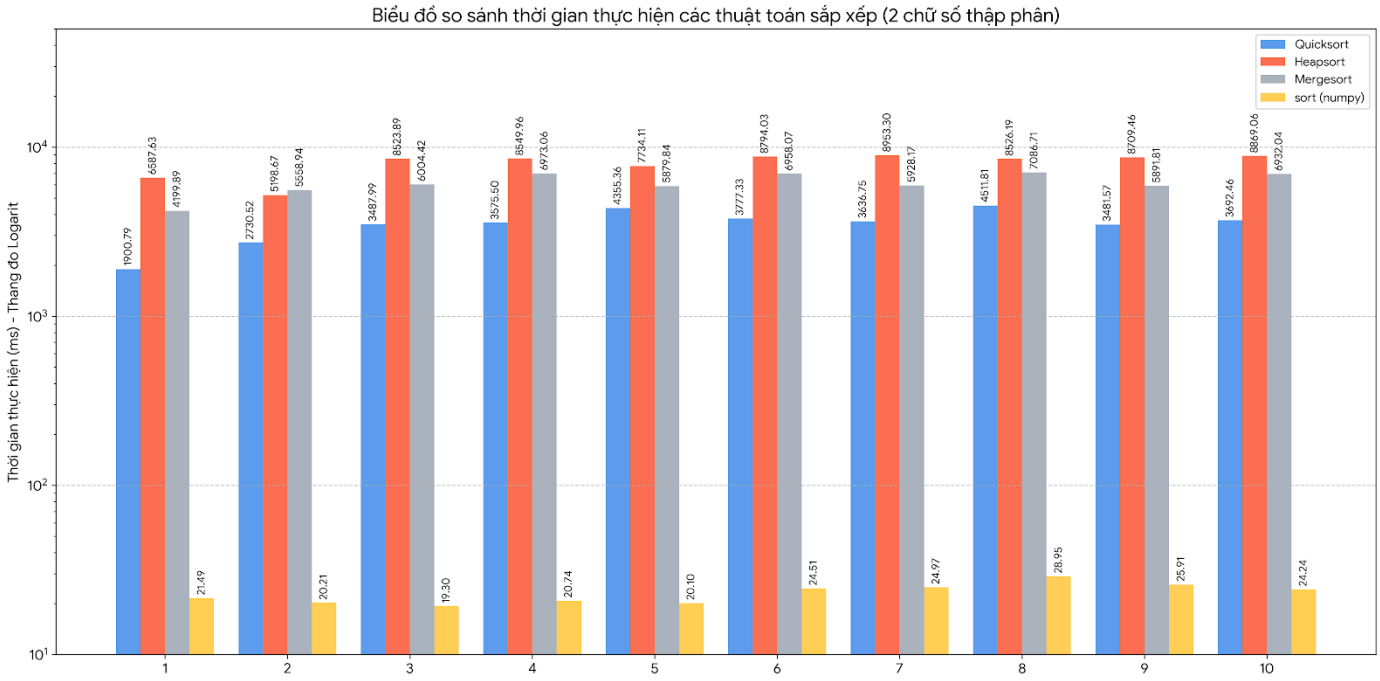
Nội dung báo cáo: Đánh giá và so sánh thời gian thực hiện của các thuật toán sắp xếp bao gồm: Quicksort, Heapsort, Mergesort và chương trình gọi hàm sort của Python (Numpy). Thử nghiệm được tiến hành trên bộ dữ liệu có 10 dãy gồm 1 triệu phần tử (bao gồm 5 dãy số thực và 5 dãy số nguyên). Các dãy được phân bố như sau: dãy 1 là dãy số thực ngẫu nhiên được sắp xếp tăng dần; dãy 2 là số thực ngẫu nhiên được sắp xếp giảm dần; dãy 3, 4 và 5 là dãy số thực có trật tự ngẫu nhiên; dãy 6, 7, 8, 9, 10 là dãy các số nguyên có trật tự ngẫu nhiên.

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện.

Dữ liệu	Thời gian thực hiện làm tròn 2 chữ số thập phân (ms)			
	Quicksort	Heapsort	Mergesort	sort (numpy)
1	1900.79	6587.63	4199.89	21.49
2	2730.52	5198.67	5558.94	20.21
3	3487.99	8523.89	6004.42	19.30
4	3575.50	8549.96	6973.06	20.74
5	4355.36	7734.11	5879.84	20.10
6	3777.33	8794.03	6958.07	24.51
7	3636.75	8953.30	5928.17	24.97
8	4511.81	8526.19	7086.71	28.95
9	3481.57	8709.46	5891.81	25.91
10	3692.46	8869.06	6932.04	24.24
Trung bình	3515.01	8044.63	6141.30	23.04

2. Biểu đồ (cột) thời gian thực hiện.



II. Kết luận:

Dựa trên số liệu thu thập được từ cấu trúc dữ liệu khổng lồ ($N = 10^6$), ta có thể rút ra được những đánh giá về bản chất của từng thuật toán như sau:

- Hàm sort có sẵn của Python (Numpy): Cho kết quả vượt trội và nhanh nhất ở mọi trường hợp (với thời gian trung bình chỉ khoảng 23.04 ms). Bởi vì thư viện Numpy được tối ưu hoá sâu ở tầng kiến trúc máy tính bằng ngôn ngữ C, giúp loại bỏ hoàn toàn độ trễ khi thông dịch ngôn ngữ Python thông thường.
- Quicksort: Đây là thuật toán chạt nhanh nhất trong số 3 thuật toán tự cài đặt, với thời gian trung bình chỉ tốn khoảng 3513.01 ms. Tốc độ thực hiện phụ thuộc vào cách chọn phần tử làm trục (pivot)

  - Với các dãy có trật tự ngẫu nhiên (từ dãy 3 đến dãy 10), thuật toán chạy nhanh và hiệu quả nhờ mảng được chia đồng đều.

  - Với dãy 1 (được sắp xếp tăng dần) và dãy 2 (được sắp xếp giảm dần); nếu chọn phần tử pivot là phần tử cuối cùng (thuật toán gốc), cây đệ quy khiến độ phức tạp thành $O(N^2)$. Điều này dẫn đến vượt quá giới hạn đệ quy (Recursion Error) trên Python. Để khắc phục triệt để và đạt lại tốc độ *O(NlogN),* ta cần cải tiến bằng cách chọn phần tử Pivot là phần tử nằm chính giữa mảng.
- Mergesort: Thể hiện sự ổn định trong toàn bộ tập dữ liệu. Dù dãy đã sắp xếp sẵn (dãy 1, 2) hay theo trật tự ngẫu nhiên, độ phức tạp luôn duy trì ở mức *O(NlogN)* với thời gian chạy trung bình bám sát 6141.30 ms. Đổi lại, thuật toán này tốn nhiều không gian bộ nhớ nhất do phải liên tục cấp phát 2 mảng phụ trong quá trình chia để trị
- Heapsort: Với độ phức tạp *O(NlogN)* ở mọi trường hợp và tối ưu hơn Mergesort là không cần thêm bộ nhớ phụ. Tuy nhiên, trên số liệu thực tế cho thất đây là thuật toán tốn nhiều thời gian nhất (trung bình 8044.63 ms). Nguyên nhân là do quá trình heapify cần nhiều thao tác tráo đổi (swap) dữ liệu liên tục trên cây nhị phân, làm hằng số thời gian tăng lên đáng kể so với Mergesort và Quicksort.

III. Thông tin chi tiết

- **Link GitHub:** https://github.com/daknulak/sorting_algorithms
- **Trong kho lưu trữ (Repository) trên GitHub đã bao gồm đầy đủ các thành phần theo yêu cầu:**

1. **Báo cáo**: File báo cáo định dạng PDF chứa đầy đủ bảng biểu và nhận xét chi tiết.
2. **Mã nguồn**: Các file Python (.py) chứa mã nguồn cài đặt 4 thuật toán sắp xếp.
3. **Dữ liệu thử nghiệm**: File data.pkl chứa bộ dữ liệu gồm 10 dãy số (1 triệu phần tử/ dãy) được sử dụng để chạy thực nghiệm.

IV. Tư liệu tham khảo

1. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.** (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. *(Tài liệu tham khảo lý thuyết cơ sở cho độ phức tạp và cách cài đặt của Quicksort, Heapsort, Mergesort).*
2. **NumPy Developers.** (n.d.). *NumPy Reference: numpy.sort*. Truy cập từ tài liệu chính thức của NumPy: https://numpy.org/doc/stable/reference/generated/numpy.sort.html *(Tài liệu tham khảo về cơ chế hoạt động và tối ưu hóa của hàm sort có sẵn).*

3. **Python Software Foundation.** (n.d.). *The Python Standard Library*. Truy cập từ: https://docs.python.org/3/library/ *(Tài liệu tham khảo về các thư viện chuẩn như* `random`, `pickle`, `sys` *được sử dụng trong mã nguồn).*
4. **GeeksforGeeks.** (n.d.). *Sorting Algorithms*. Truy cập từ: https://www.geeksforgeeks.org/sorting-algorithms/ *(Tài liệu tham khảo hỗ trợ cài đặt và tinh chỉnh cách chọn Pivot ở giữa mảng cho thuật toán Quicksort).*