Lớp: KHTN2024

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Thời gian thực hiện: 011/03 – 16/03/2022

Sinh viên thực hiện: Lê Phạm Thành Nhân

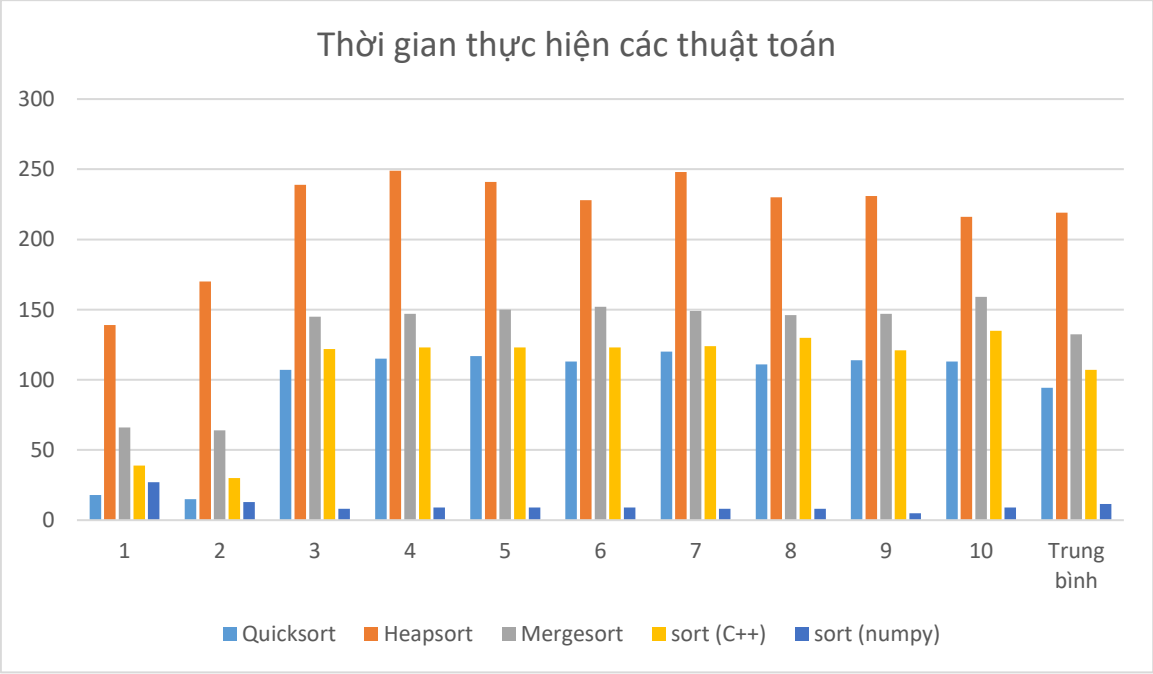
Nội dung báo cáo: Thực nghiệm các giải thuật sắp xếp nội

I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện[1]

Dữ liệu	Thời gian thực hiện (ms)				
	Quicksort	Heapsort	Mergesort	sort (C++)	sort (numpy)
1	18	139	66	39	27
2	15	170	64	30	13
3	107	239	145	122	8
4	115	249	147	123	9
5	117	241	150	123	9
6	113	228	152	123	9
7	120	248	149	124	8
8	111	230	146	130	8
9	114	231	147	121	5
10	113	216	159	135	9
Trung bình	94.3	219.1	132.5	107	11.5

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

[1] Số liệu chỉ mang tính minh họa

Kết quả thực nghiệm cho thấy mặc dù các thuật toán sắp xếp như Quicksort, Heapsort, Mergesort đều có độ phức tạp trung bình là O(n log n), nhưng về mặt thực nghiệm lại có những khác biệt rõ rệt về hiệu năng:

- ***Sort (numpy): Thực nghiệm cho thấy đây là thuật toán nhanh nhất (trung bình 11.5 ms). Điều này phần lớn nhờ tối ưu hóa mạnh mẽ của thư viện NumPy, vốn được xây dựng bằng C và tận dụng tối đa khả năng xử lý mảng của Python.***
- ***Quicksort: Với thời gian trung bình khoảng 94.3 ms, Quicksort cho hiệu năng khá tốt trong điều kiện trung bình. Tuy nhiên, nhược điểm của thuật toán này là có thể gặp trường hợp xấu khi mảng dữ liệu không được phân chia đều, mặc dù trường hợp này hiếm khi xảy ra trong thực tế.***
- ***Sort (C++): Thuật toán sắp xếp có sẵn trong C++ cho kết quả ổn định (trung bình 107 ms). Đây là lựa chọn đáng tin cậy nhờ vào tối ưu hóa của trình biên dịch và thư viện chuẩn.***
- ***Mergesort: Với thời gian trung bình khoảng 132.5 ms, Mergesort hoạt động ổn định và có ưu điểm là không phụ thuộc vào dữ liệu vào, nhưng lại cần bộ nhớ phụ để lưu trữ tạm thời, điều này có thể là bất lợi trong một số trường hợp.***
- ***Heapsort: Mặc dù đảm bảo thời gian thực thi không vượt quá O(n log n) trong mọi trường hợp, nhưng kết quả thử nghiệm (trung bình 219.1 ms) cho thấy hiệu năng của Heapsort kém hơn so với các thuật toán khác, điều này có thể do chi phí cao trong việc duy trì cấu trúc heap.***

Từ đó, có thể rút ra nhận xét rằng, trong ứng dụng thực tế, việc lựa chọn thuật toán sắp xếp không chỉ dựa vào độ phức tạp lý thuyết mà còn phụ thuộc vào hiệu quả tối ưu hóa của thư viện, cấu trúc dữ liệu và ngữ cảnh sử dụng. Nếu có thể, việc sử dụng các hàm sắp xếp có sẵn trong các thư viện chuẩn (như numpy trong Python hoặc sort trong C++) sẽ là lựa chọn hợp lý, vì chúng đã được tối ưu hóa và kiểm chứng qua thời gian.

III. Thông tin chi tiết

https://github.com/lenhanbo/sort_benchmark/tree/master