Lớp: IT003.O21.CTTN

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Thời gian thực hiện: 27/02/2024 – 12/03/2024

Sinh viên thực hiện: Trần Trung Nhân

Nội dung báo cáo: Kết quả thực nghiệm các giải thuật sắp xếp nội (thực hiện trên hệ điều hành ubuntu)

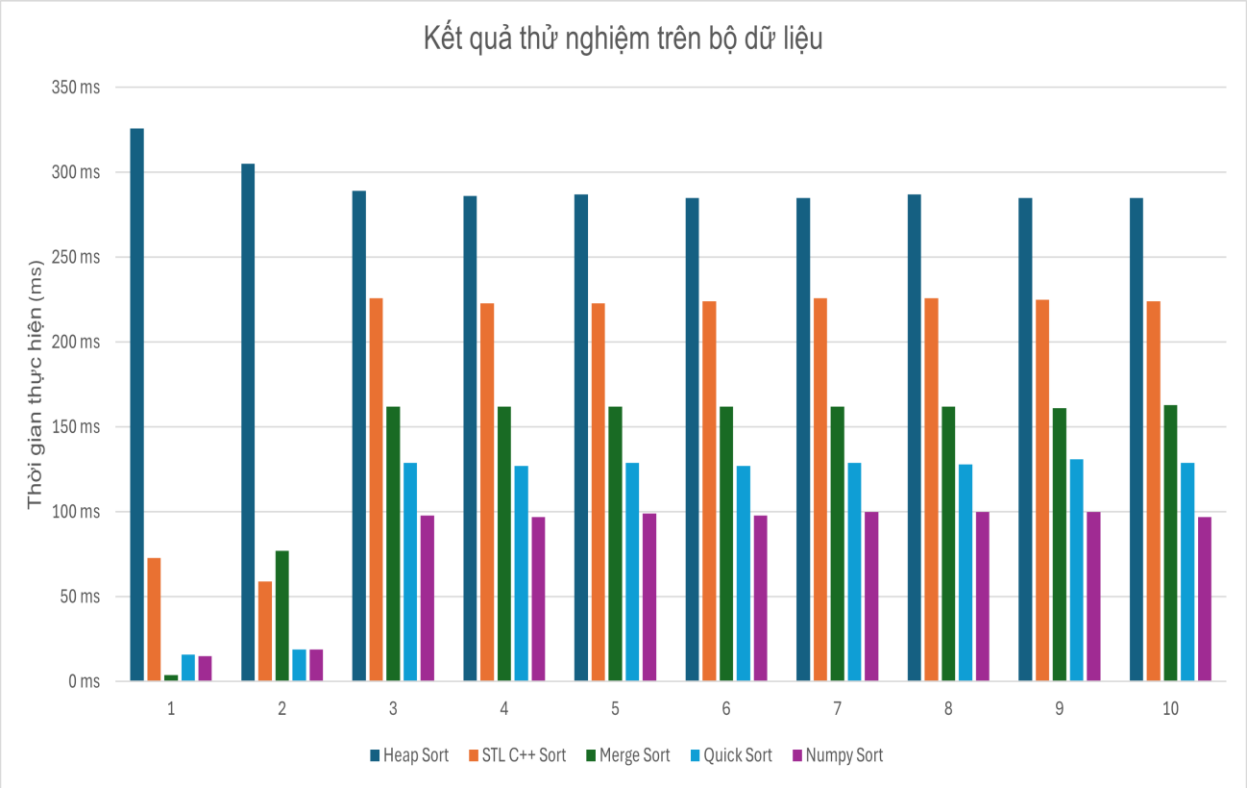
I. Kết quả thử nghiệm

1. Bảng thời gian thực hiện

TEST	Heap Sort	STL C++ Sort	Merge Sort	Quick Sort	Numpy Sort
1	326 ms	73 ms	4 ms	16 ms	15 ms
2	305 ms	59 ms	77 ms	19 ms	19 ms
3	289 ms	226 ms	162 ms	129 ms	98 ms
4	286 ms	223 ms	162 ms	127 ms	97 ms
5	287 ms	223 ms	162 ms	129 ms	99 ms
6	285 ms	224 ms	162 ms	127 ms	98 ms
7	285 ms	226 ms	162 ms	129 ms	100 ms
8	287 ms	226 ms	162 ms	128 ms	100 ms
9	285 ms	225 ms	161 ms	131 ms	100 ms
10	285 ms	224 ms	163 ms	129 ms	97 ms

	Heap Sort	STL C++ Sort	Merge Sort	Quick Sort	Numpy Sort
trung bình	292	192.9	137.7	106.4	82.3
độ lệch chuẩn	12.71219887	63.53652493	51.26997172	44.46841576	32.68042227

2. Biểu đồ cột thời gian thực hiện



II. Kết luận

- Về tốc độ thực hiện: hàm sort của thư viện numpy chạy nhanh nhất, thuật toán Heap Sort chạy chậm nhất.
- Về độ ổn định: thuật toán Heap Sort có độ ổn định cao nhất, hàm sort của C++ có độ ổn định thấp nhất.
- Đối với test 1: Thuật toán Merge Sort khi được tối ưu (Natural Merge Sort) thể hiện tốt nhất khi tất cả các phần tử đã được sắp xếp sẵn. Heap Sort tốn nhiều thời gian hơn so với những test còn lại, trong khi những thuật toán sắp xếp còn lại đều thể hiện tốt.
- Đối với test 2: mặt bằng chung các thuật toán sắp xếp đều thể hiện tốt, ngoại trừ Heap Sort.
- Nhìn chung, toàn bộ thuật toán đều thể hiện ổn định, sự chênh lệch chủ yếu đến từ việc toàn bộ phần tử đã được sắp xếp sẵn (theo thứ tự tăng dần hoặc giảm dần).
- Phân tích:
    1. Heap Sort: Heap Sort là một thuật toán sắp xếp dựa trên so sánh sử dụng cấu trúc dữ liệu heap nhị phân. Nó có độ phức tạp thời gian trung bình và tồi nhất là O(n log n). Thời gian thực thi nhất quán trong dữ liệu bất kể thứ tự ban đầu của đầu vào.

2. STL C++ Sort: Hàm sort trong C++ thường triển khai một thuật toán sắp xếp hỗn hợp như Introsort. Nó bắt đầu với Quick Sort và chuyển sang Heap Sort khi độ sâu đệ quy vượt quá một giới hạn nhất định. Điều này cung cấp độ phức tạp thời gian tồi nhất là O(n log n), tránh được trường hợp tồi nhất O(n^2) của Quick Sort. Sự biến đổi trong thời gian thực thi có thể do hiệu quả của Quick Sort tùy thuộc vào thứ tự ban đầu của đầu vào.
3. Merge Sort: Merge Sort là một thuật toán chia để trị với độ phức tạp thời gian là O(n log n) trong tất cả các trường hợp. Thuật toán hoạt động ổn đối với dữ liệu đầu vào 1 triệu phần tử. Việc tối ưu quá trình chia mảng (Natural Merge Sort) giúp tốc độ thực hiện của thuật toán tương đối nhanh.
4. Quick Sort: Quick Sort cũng là một thuật toán chia để trị, nhưng hiệu suất của nó phụ thuộc nhiều vào việc chọn pivot. Độ phức tạp thời gian trung bình là O(n log n), nhưng trường hợp tồi nhất là O(n^2), xảy ra khi đầu vào đã được sắp xếp hoặc sắp xếp ngược. Thời gian trung bình tương đối thấp trong dữ liệu đến từ việc chọn pivot ở vị trí giữa.
5. Numpy Sort: Numpy Sort sử dụng một biến thể của Quick Sort gọi là Introselect theo mặc định, kết hợp Quick Sort, Heap Sort, và Insertion Sort. Nó nhằm cung cấp hiệu suất trung bình nhanh và hiệu suất tồi nhất tối ưu. Thời gian trung bình nhanh nhất trong dữ liệu cho thấy nó được tối ưu hóa cao cho loại dữ liệu đang sắp xếp.

III. Thông tin chi tiết

- Link Github: **Github**
- Repo (branch master): IT003