Lớp: IT003.Q21.CTTN

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Sinh viên thực hiện: Huỳnh Anh Nhật

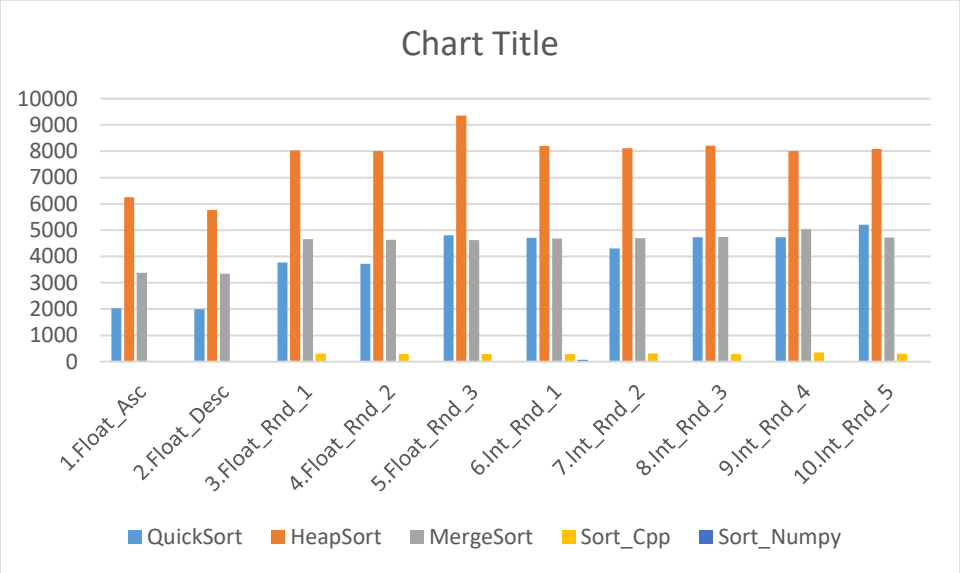
Nội dung báo cáo: Kết quả thử nghiệm các thuật toán sắp xếp

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	2031	6251	3383	5	17
2	1992	5766	3348	6	18
3	3772	8023	4658	299	17
4	3719	8000	4633	293	19
5	4810	9359	4621	294	19
6	4702	8194	4687	290	66
7	4304	8114	4690	312	11
8	4731	8207	4744	294	10
9	4731	8002	5033	347	12
10	5201	8092	4715	299	9
Trung bình	3999	7801	4451	244	20

2. Biểu đồ (cột) thời gian thực hiện



II. Kết luận:

1. Sự vượt trội của các thư viện chuẩn (Numpy và C++ Sort):

- Sort (Numpy): Đạt tốc độ xử lý "thần tốc" với thời gian trung bình chỉ 20 ms. Đây là thuật toán nhanh nhất, nhanh hơn thuật toán tự cài đặt chậm nhất (HeapSort) tới gần 400 lần.

[1] Số liệu chỉ mang tính minh họa

- Sort (C++): Đứng thứ hai với trung bình 244 ms. Đặc biệt, với dữ liệu đã sắp xếp (Dataset 1 & 2), thời gian thực thi chỉ là 5-6 ms, chứng tỏ khả năng nhận diện chuỗi con (runs) cực tốt của thuật toán Timsort.
- Nhận xét: Các thư viện có backend bằng ngôn ngữ C/C++ (như Numpy và Python built-in sort) luôn cho hiệu năng vượt xa code Python thuần túy nhờ khả năng quản lý bộ nhớ và tối ưu hóa cấp thấp.

2. Đánh giá QuickSort (Trung bình ~3999 ms):

- Là thuật toán nhanh nhất trong nhóm các thuật toán tự cài đặt.
- Điểm đáng chú ý: Trên dữ liệu đã sắp xếp (Dataset 1 và 2), QuickSort chạy rất nhanh (~2000 ms), chỉ bằng một nửa thời gian so với khi chạy trên dữ liệu ngẫu nhiên (~4700 ms).
- Kết luận: Việc chọn Pivot ở giữa (Middle Pivot) đã phát huy tác dụng tối đa, giúp thuật toán không những tránh được trường hợp xấu nhất $O(N^2)$ trên mảng đã sắp xếp mà còn tận dụng được tính thứ tự để chạy nhanh hơn.

3. Đánh giá MergeSort (Trung bình ~4451 ms):

- Giữ vị trí cân bằng với hiệu năng ổn định. Thời gian chạy dao động trong khoảng hẹp (từ 3300 ms đến 5000 ms) bất kể dữ liệu đầu vào là gì.
- Tuy chậm hơn QuickSort một chút (khoảng 10-15%), nhưng MergeSort cho thấy tính ổn định (stability) cao, là lựa chọn an toàn khi không thể kiểm soát được cấu trúc dữ liệu đầu vào.

4. Đánh giá HeapSort (Trung bình ~7801 ms):

- Là thuật toán chậm nhất trong bài kiểm tra này.
- Thời gian thực thi trung bình cao gần gấp đôi so với QuickSort (7801 ms so với 3999 ms).
- Nguyên nhân: Dù cùng độ phức tạp $O(N\ log\ N)$, nhưng HeapSort trên Python chịu ảnh hưởng nặng nề bởi việc truy cập bộ nhớ không tuần tự (cache locality kém) và chi phí hoán đổi phần tử (swap) liên tục.

TỔNG KẾT XẾP HẠNG TỐC ĐỘ:

Sort (Numpy) > Sort (C++) >> QuickSort > MergeSort > HeapSort

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

1. Báo cáo
2. Mã nguồn
3. Dữ liệu thử nghiệm

Github repo: https://github.com/ZacLiebert/DSA

Mã nguồn nằm trong file compare.py (gồm các thuật toán sắp xếp và sinh ngẫu nhiên dữ liệu).

Final_Results.csv là kết quả thử nghiệm.