Báo cáo: So sánh thuật toán sắp xếp

Tên sinh viên: Võ Hồng Quân
Môn: Thuật Toán và Cấu Trúc Dữ Liệu

Ngày 26 tháng 9 năm 2025

Mục lục

1 Yêu cầu và Mục tiêu

1. Đọc dữ liệu sản phẩm từ file CSV (có thể sinh dữ liệu ngẫu nhiên nếu cần).
2. Hiển thị n sản phẩm đầu tiên, n được nhập từ bàn phím.
3. Cài đặt các chức năng sắp xếp theo: `price` , `sold`, `rating`, `name` .
4. Triển khai nhiều thuật toán: Bubble, Selection, Insertion, Interchange, Merge, Quick, Counting, Radix.
5. Kiểm tra tính ổn định (stability) của thuật toán trên bộ dữ liệu nhỏ.
6. Đo thời gian chạy với $N = 10\,000$ sản phẩm, mỗi thuật toán chạy 5 lần — lấy trung bình.
7. Mở rộng: Khi sắp xếp theo số lượng bán, nếu số lượng bằng nhau thì hiển thị theo thứ tự tên sản phẩm A-Z. Lọc các sản phẩm có giá trong đoạn từ [a, b], a và b được nhập từ bàn phím và sắp xếp kết quả theo thứ tự giá từ thấp đến cao.

Trong báo cáo này, ngôn ngữ lập trình Python được sử dụng để cài đặt và đo thời gian các thuật toán sắp xếp. Việc lựa chọn Python không nhằm mục đích đánh giá hiệu năng tuyệt đối của ngôn ngữ, mà nhằm xây dựng một môi trường thử nghiệm đơn giản, dễ cài đặt và dễ mở rộng.

Python cũng cung cấp các thư viện hữu ích (như `pandas`,`seaborn`, `numpy`, `matplolib`) giúp thuận tiện trong việc sinh dữ liệu, đo đạc, và trực quan hóa kết quả thử nghiệm.

2 Cấu hình phần cứng (thí nghiệm)

CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz (6 lõi, 12 luồng)

RAM: 16.0 GB DDR4, tốc độ 2933 MHz (SODIMM)

GPU: NVIDIA GeForce GTX 1650 with Max-Q Design

Hệ điều hành: Windows 10 Pro (phiên bản 22H2)

Python: `Python 3.10` với các thư viện:

- `pandas` (v2.1.0)
- `numpy` (v1.26.0)
- **Ghi chú**: Thí nghiệm chạy trên môi trường Conda, sử dụng IDE VS Code.

3 Dữ liệu

3.1 Định dạng

File CSV gồm các cột: `id, name, price, sold, rating`. Ví dụ mẫu:

```
id,name,price,sold,rating
1,Razer,188451,592025,3.2
2,Jio,313131,270988,2.7
3,Acer,394640,611943,2.69
4,Wings,206279,960777,2.4
5,Apple,292745,378158,2.68
...
```

3.2 Chuẩn bị dữ liệu cho benchmark

- Sinh 10 000 bản ghi ngẫu nhiên nếu dataset gốc không đủ, đảm bảo seed cố định để tái lập kết quả.
- Kiểu dữ liệu theo yêu cầu: `id` (int), `price` (int), `sold` (int), `rating` (float), `name` (string).

4 Thiết kế và Cài đặt

- Lớp/Module `Search`: linear search, binary search.
- Lớp `Sort` base: xử lý dataset, validator, đo thời gian.
- Lớp con `ElementSort`: Interchange, Bubble, Selection, Insertion.
- Lớp con `DivideConquerSorting`: Merge Sort, Quick Sort.
- Lớp con `NonComparisonSort`: Counting Sort, Radix Sort.
- Sử dụng pandas để đọc/ghi CSV và để kiểm tra nhanh. Không tính thời gian đọc file trong benchmark.
- Mỗi thuật toán được bọc bằng decorator để đo thời gian (dùng `time.time()`.

5 Benchmark

5.1 Phương pháp

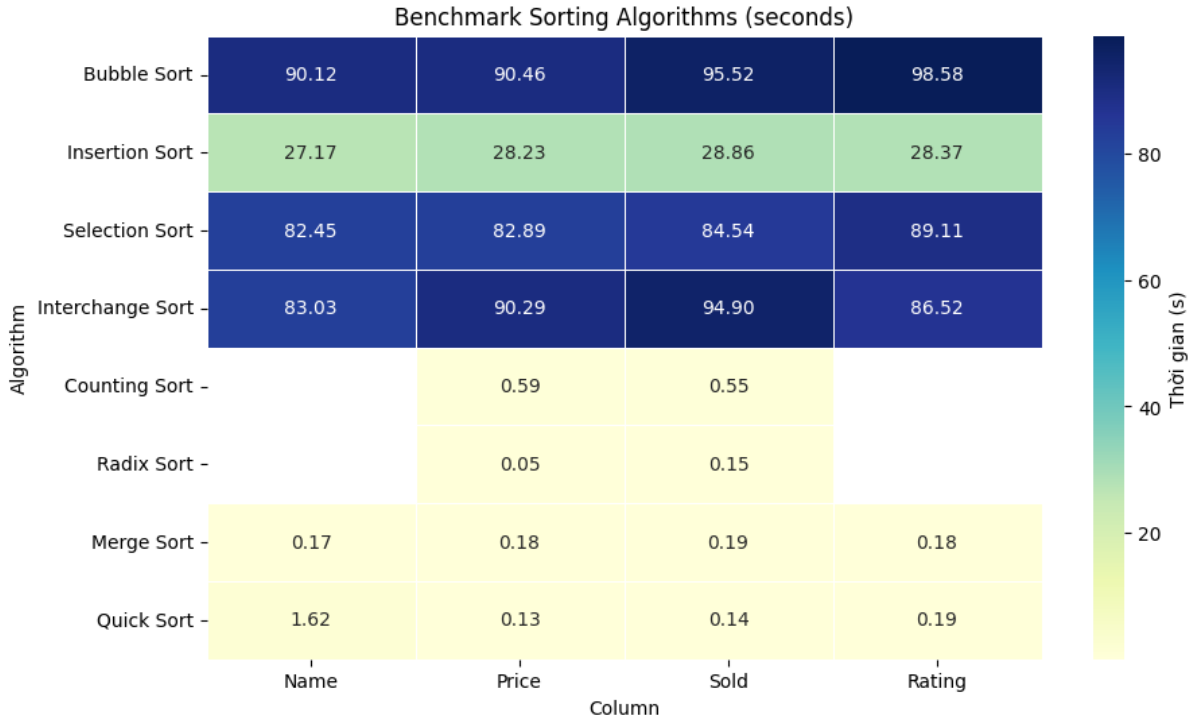
- Sử dụng $N = 10\,000$ sản phẩm (giữ input giống nhau cho tất cả thuật toán).
- Mỗi thuật toán chạy $\times 5$ lần; kết quả báo cáo là thời gian trung bình.
- Không tính thời gian đọc file hoặc sinh dữ liệu; chỉ đo thời gian chạy hàm sắp xếp.
- Với những thuật toán không hỗ trợ kiểu dữ liệu (ví dụ Counting/Radix không hỗ trợ `name` string), kết quả để `None`.

5.2 Kết quả

Bảng kết quả

Thuật toán	Name	Price	Sold	Rating
Bubble Sort	90.12	90.46	95.52	98.58
Insertion Sort	27.17	28.23	28.86	28.37
Selection Sort	82.45	82.89	84.54	89.11
Interchange Sort	83.03	90.29	94.90	86.52
Counting Sort	–	0.59	0.55	–
Radix Sort	–	0.05	0.15	–
Merge Sort	0.17	0.18	0.19	0.18
Quick Sort	1.62	0.13	0.14	0.19

Bảng 1: Kết quả benchmark thời gian sắp xếp (giây)



Hình 1: Ma trận kết quả

5.3 Kết luận

- **Hiệu năng thực nghiệm**: Merge Sort ổn định và nhanh nhất ($\approx 0.18s$). Quick Sort cũng hiệu quả ($\approx 0.2{-}1s$) nhưng kém ổn định hơn.
- **Counting / Radix**: Chỉ áp dụng cho dữ liệu số nguyên. Radix Sort rất nhanh ($< 0.2s$) khi miền giá trị nhỏ. Counting Sort phù hợp khi miền giá trị hẹp, nhưng không áp dụng cho chuỗi.
- **Các thuật toán** $O(n^2)$: Bubble Sort, Insertion Sort, Selection Sort, Interchange Sort mất hàng chục giây → không phù hợp cho dữ liệu lớn, chỉ dùng để minh hoạ.
- **Ổn định**: Merge, Bubble, Insertion, Radix là ổn định; Quick, Selection, Interchange, Counting không ổn định.
- **Kết luận chung**: **Merge Sort** là lựa chọn cân bằng nhất (nhanh và ổn định). Với

dữ liệu số nguyên, **Radix Sort** có thể vượt trội. Các thuật toán $O(n^2)$ chỉ dùng cho tập nhỏ hoặc mục đích học thuật.