Họ và tên: Nguyễn Hoàng Hải

MSSV: 21522034

Lớp: KHTN2021

BÁO CÁO KẾT QUẢ THỰC NGHIỆM CÁC THUẬT TOÁN SORT

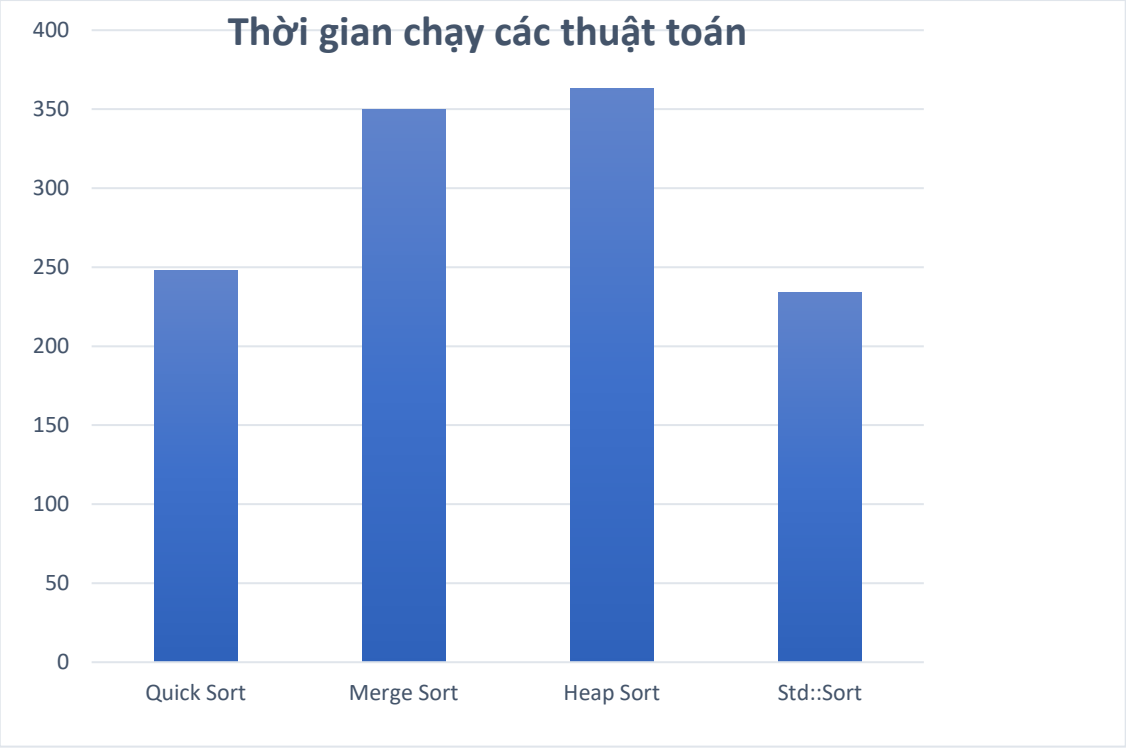
Thời gian thực hiện: 11/03 – 16/03/2022

I. Thống kê kết quả thực nghiệm

1. Bảng kết quả:

Test case	Thời gian chạy (ms)			
	Quick Sort	Merge Sort	Heap Sort	Std::Sort
1	54	250	206	89
2	65	253	241	77
3	351	379	402	294
4	302	369	409	263
5	278	368	379	252
6	286	378	399	265
7	285	373	390	259
8	272	380	425	296
9	279	374	382	257
10	306	373	400	291
Trung bình	247.8	349.7	363.3	234.3

2. Biểu đồ cột kết quả



II. Nhận xét:

- Ở trường hợp đầu tiên, khi mảng đã sắp xếp tăng dần, thuật toán Quick Sort chạy nhanh nhất vì chỉ quét qua mỗi phần tử log(n) lần để xét các điều kiện. Tương tự, std::sort c++ hay còn gọi là Introsort (thuật toán kết hợp cả 3 thuật toán sắp xếp Quick Sort, Heap Sort và Insertion Sort) cũng có thời gian chạy nhanh thứ 2 và không chênh lệch so với Quick Sort quá nhiều. Riêng 2 thuật toán Merge Sort và Heap Sort có thời gian thực thi lớn gấp nhiều lần Quick Sort.
- Ở trường hợp thứ 2, khi mảng đã sắp xếp giảm dần, tốc độ thực hiện các thuật toán tương đối như trường hợp đầu tiên.
- Ở 8 trường hợp còn lại, thời gian thực hiện các thuật toán lần lượt như sau:
  - ❖ Quick Sort: dao động từ 272 – 351 (ms), biên độ: 79(ms)
  - ❖ Merge Sort: dao động từ 368 – 380 (ms), biên độ: 12(ms)
  - ❖ Heap Sort: dao động từ 379 – 425 (ms), biên độ: 46(ms)
  - ❖ Std::Sort: dao động từ 252 – 296 (ms), biên độ: 44(ms)

  → Thuật toán Merge Sort có biên độ thời gian thực hiện rất bé so với Quick Sort (nhờ vào tính ổn định của độ phức tạp thuật toán O(nlogn)).

  → Thuật toán Std::Sort và Heap Sort có biên độ thời gian thực hiện lần lượt là 44ms và 46ms, bé hơn một nửa với Quick Sort (nhờ vào tính ổn định của độ phức tạp thuật toán O(nlogn))

  → Như vậy, thuật toán Quick Sort nhạy cảm với bộ dữ liệu, thời gian thực hiện thuật toán này bị ảnh hưởng bởi thứ tự tương đối giữa các phần tử cần được sắp xếp.

- Từ tốc độ trung bình và các nhận xét nêu trên, có thể thấy rằng:
  - ❖ Std::Sort là một thuật toán hiệu quả và có tính ổn định về mặt thời gian cao
  - ❖ Hai thuật toán Merge Sort và Heap Sort cũng có tính ổn định về mặt thời gian tốt tuy nhiên hiệu quả chạy thuật toán chưa cao bằng Std::Sort
  - ❖ Quick Sort hoạt động không ổn định về mặt thời gian do nhạy cảm với bộ dữ liệu, tuy nhiên tốc độ trung bình của Quick Sort chỉ xếp sau Std::Sort cho thấy rằng đây là một thuật toán đáng được sử dụng trong một số các trường hợp nhất định.
- Việc thực nghiệm thời gian chạy các thuật toán trên còn bị chi phối bởi cách tổ chức cấu trúc dữ liệu trong mã nguồn cũng như hệ điều hành cùng nhiều yếu tố khác nên các giá trị thực nghiệm mang tính tương đối.

III. Kết luận:

Std::sort là thuật toán rất hiệu quả, có tính ổn định cao, là ứng cử viên sáng giá cho các lập trình viên sử dụng trong nhiều vấn đề. Merge Sort và Heap Sort hiệu quả chỉ sau Std::Sort nhưng tính ổn định rất cao, hai thuật toán này có ý tưởng sắp xếp rất hay và thường được dùng bổ trợ cho các thuật toán trong lập trình thi đấu. Quick Sort là thuật toán hoạt động rất hiệu quả ở nhiều trường hợp, tuy nhiên thuật toán này hoạt động lại không ổn định nên phải cân nhắc trước khi quyết định áp dụng Quick Sort để giải quyết vấn đề.

Link Github: ***https://github.com/SKN443/Sort.HW***