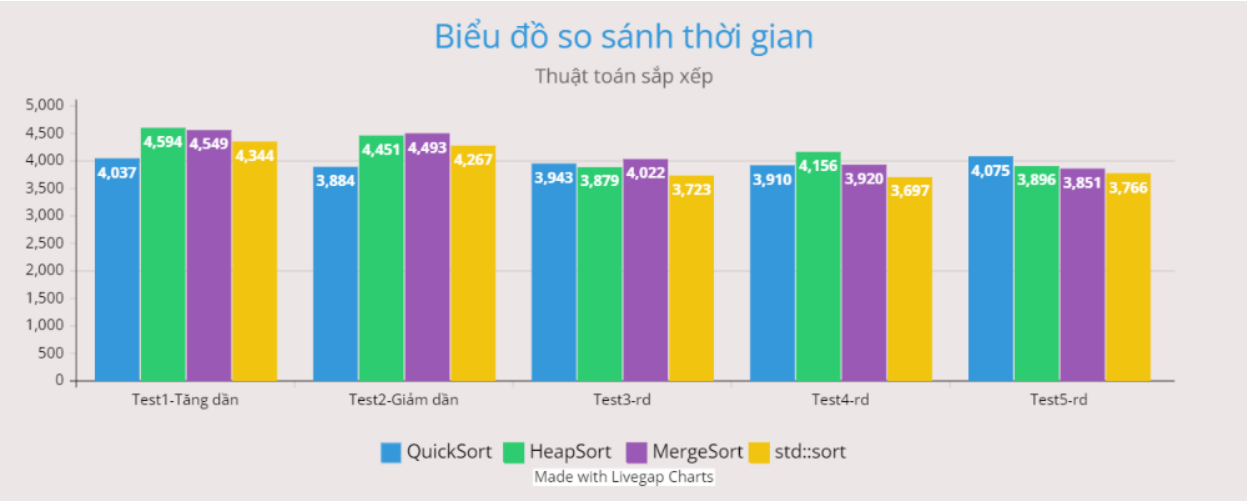
Báo cáo thử nghiệm

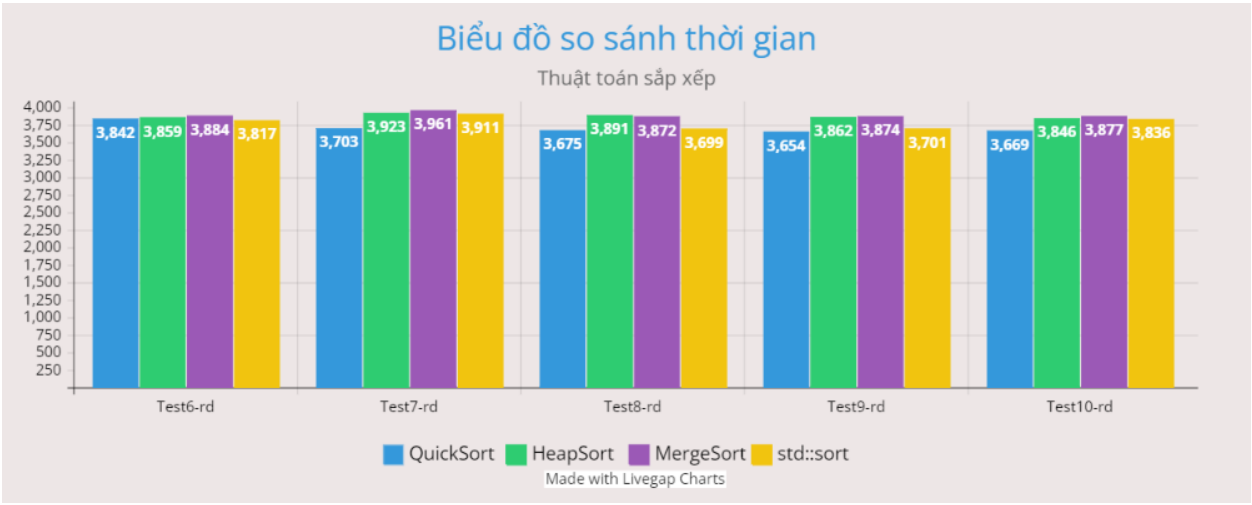
Họ và tên: Hồ Hoàng Diệp
MSSV: 22520249
Link GitHub: https://github.com/Diephho/it003

1. Bảng dữ liệu:

Test	QuickSort	HeapSort	MergeSort	std::sort
1	4037	4594	4549	4344
2	3884	4451	4493	4267
3	3943	3879	4022	3723
4	3910	4156	3920	3697
5	4075	3896	3851	3766
6	3842	3859	3884	3817
7	3703	3923	3961	3911
8	3675	3891	3872	3699
9	3654	3862	3874	3701
10	3669	3846	3877	3836

2. Biểu đồ:


Biểu đồ so sánh thời gian
Thuật toán sắp xếp
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
4,037
4,594
4,549
4,344
3,884
4,451
4,493
4,267
3,943
3,879
4,022
3,723
3,910
4,156
3,920
3,697
4,075
3,896
3,851
3,766
Test1-Tăng dần
Test2-Giảm dần
Test3-rd
Test4-rd
Test5-rd
QuickSort
HeapSort
MergeSort
std::sort
Made with Livegap Charts


Biểu đồ so sánh thời gian
Thuật toán sắp xếp
4,000
3,750
3,500
3,250
3,000
2,750
2,500
2,250
2,000
1,750
1,500
1,250
1,000
750
500
250
3,842
3,859
3,884
3,817
3,703
3,923
3,961
3,911
3,675
3,891
3,872
3,699
3,654
3,862
3,874
3,701
3,669
3,846
3,877
3,836
Test6-rd
Test7-rd
Test8-rd
Test9-rd
Test10-rd
QuickSort
HeapSort
MergeSort
std::sort
Made with Livegap Charts

3. Nhận xét:

- Đối với bộ dữ liệu gồm một triệu phần tử, các thuật toán sắp xếp có thời gian giao động từ khoảng 3600-4600 mili giây.
- Trường hợp dữ liệu tăng dần (Test1): HeapSort có tốc độ nhanh hơn các thuật toán khác
- Trường hợp dữ liệu giảm dần (Test2): QuickSort có tốc độ nhanh hơn các thuật toán khác
- Trường hợp dữ liệu ngẫu nhiên (Test3,Test4...): QuickSort là thuật toán có nhiều trường hợp nhanh nhất trong số các thuật toán được đánh giá. std::sort cũng chiếm một vài trường hợp nhanh nhất.
- Từ kết quả đánh giá, ta có thể nhận thấy rằng QuickSort là thuật toán sắp xếp nhanh nhất trong số các thuật toán được đánh giá, và có thể hiệu quả trong hầu hết các trường hợp. std::sort cũng là một lựa chọn tốt với thời gian sắp xếp khá nhanh và đáp ứng được nhiều yêu cầu hiệu suất.
- HeapSort và MergeSort đều đảm bảo thời gian sắp xếp ổn định và không bị ảnh hưởng bởi trường hợp tốt hoặc xấu của bộ dữ liệu, nhưng chúng không thể sắp xếp dữ liệu nhanh như QuickSort và std::sort.
- Với bộ dữ liệu ngẫu nhiên, QuickSort và std::sort là những lựa chọn tốt, vì chúng có thể sắp xếp các phần tử của bộ dữ liệu một cách nhanh chóng và hiệu quả. MergeSort và HeapSort cũng có thể được sử dụng, nhưng có thể không hiệu quả bằng QuickSort và std::sort trong một số trường hợp.

  Tóm lại, việc chọn thuật toán phù hợp nhất để sắp xếp dữ liệu phụ thuộc vào các yêu cầu hiệu suất của ứng dụng và tính chất của bộ dữ liệu.