Lớp: IT003.M21.ANTN

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Thời gian thực hiện: 02/03/2022 – 16/03/2022

Sinh viên thực hiện: Phạm Hoàng Phúc Thịnh **MSSV:** 21521473

Nội dung báo cáo:

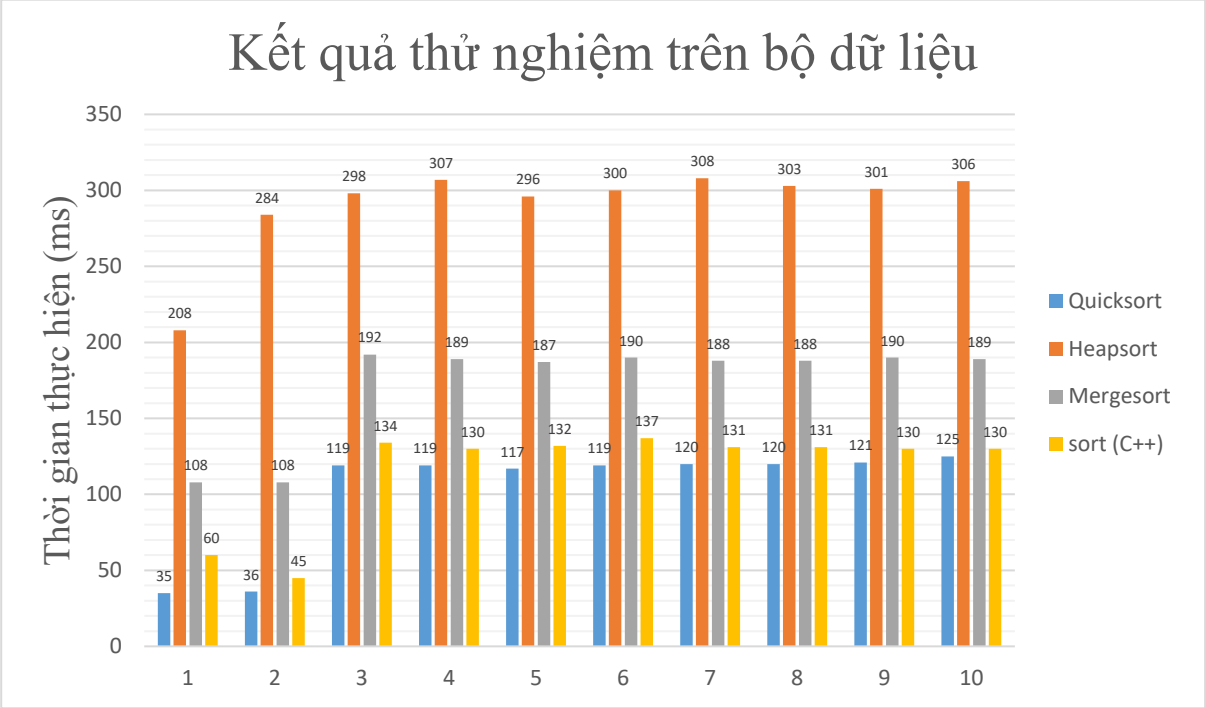
Thử nghiệm chương trình sắp xếp dãy tăng dần theo các thuật toán QuickSort, HeapSort, MergeSort và chương trình gọi hàm sort của C++.

I. *Kết quả thử nghiệm:*

1. *Bảng thời gian thực hiện:*

Dữ liệu	Thời gian thực hiện (ms)			
	Quicksort	Heapsort	Mergesort	sort (C++)
1	35	208	108	60
2	36	284	108	45
3	119	298	192	134
4	119	307	189	130
5	117	296	187	132
6	119	300	190	137
7	120	308	188	131
8	120	303	188	131
9	121	301	190	130
10	125	306	189	130
Trung bình:	103.1	291.1	172.9	116

2. *Biểu đồ (cột) thời gian thực hiện*



II. Kết luận:

Thời gian thực hiện (ms)	**Quicksort**	**Heapsort**	**Mergesort**	**sort (C++)**
Dữ liệu 1 (thứ tự tăng dần)	35	208	108	60
Dữ liệu 2 (thứ tự giảm dần)	36	284	108	45
Nhanh nhất	35	208	108	45
Chậm nhất	125	308	192	137
Trung bình	103.1	291.1	172.9	116

- Nhìn chung, tốc độ xử lý dữ liệu của cả 4 thuật toán đều tương đối nhanh.

- Thuật toán QuickSort:
  + Tốc độ xử lý phụ thuộc vào dữ liệu đầu vào: trường hợp tốt nhất với độ phức tạp là o(n.log(n)) (xảy ra khi mỗi lần phân đoạn, hai mảng con có độ dài bằng nhau); trường hợp xấu nhất là $O(n^2)$ (xảy ra khi mỗi lần phân đoạn, ta chọn phải phần tử lớn nhất hoặc nhỏ nhất làm chốt).
  + Độ phức tạp trung bình là Ω(n.log(n)). Trong một số trường hợp, QuickSort là thuật toán có tốc độ xử lý tốt nhất.

- Thuật toán HeapSort:
  + Không tối ưu trong mọi trường hợp;
  + Độ phức tạp trung bình là Ω(n.log(n)) trong mọi trường hợp;
  + Ít bị ảnh hưởng bởi dữ liệu đầu vào.

- Thuật toán MergeSort:
  + Độ phức tạp trung bình là Ω(n.log(n)); Tốc độ xử lý trung bình khá nhanh;
  + Không bị ảnh hưởng nhiều bởi dữ liệu đầu vào.

- Chương trình gọi hàm sort của C++:
  + Tối ưu hơn so với các thuật toán sắp xếp thông thường, không tối ưu trong một số trường hợp;
  + Độ phức tạp thuật toán xấu nhất là O(n.log(n)); Có sẵn trong thư viện STL;

Mỗi thuật toán sắp xếp đều có những ưu điểm và hạn chế khác nhau, cách tiếp cận vấn đề cũng khác nhau, vì thế nên không có thuật toán nào là tối ưu cho mọi trường hợp. Tùy vào bài toán với bối cảnh cụ thể, với giới hạn về thời gian, tài nguyên và đặc điểm dữ liệu mà ta có thể áp dụng các thuật toán sort khác nhau hoặc cải tiến 1 số thuật toán để phù hợp với yêu cầu bài toán.

III. Thông tin chi tiết:

1. Báo cáo:
   https://github.com/tswt2650/SortCmp/blob/main/SortingComparisonReport-21521473.pdf

2. Mã nguồn:
   https://github.com/tswt2650/SortCmp/blob/main/Source%20Code

3. Dữ liệu thử nghiệm:
   https://github.com/tswt2650/SortCmp/tree/main/test