Lớp: IT003.O21.CTTN

BÁO CÁO KẾT QUẢ THỬ NGHIỆM

Thời gian thực hiện: 27/02 – 12/03/2024

Sinh viên thực hiện: Cáp Kim Hải Anh

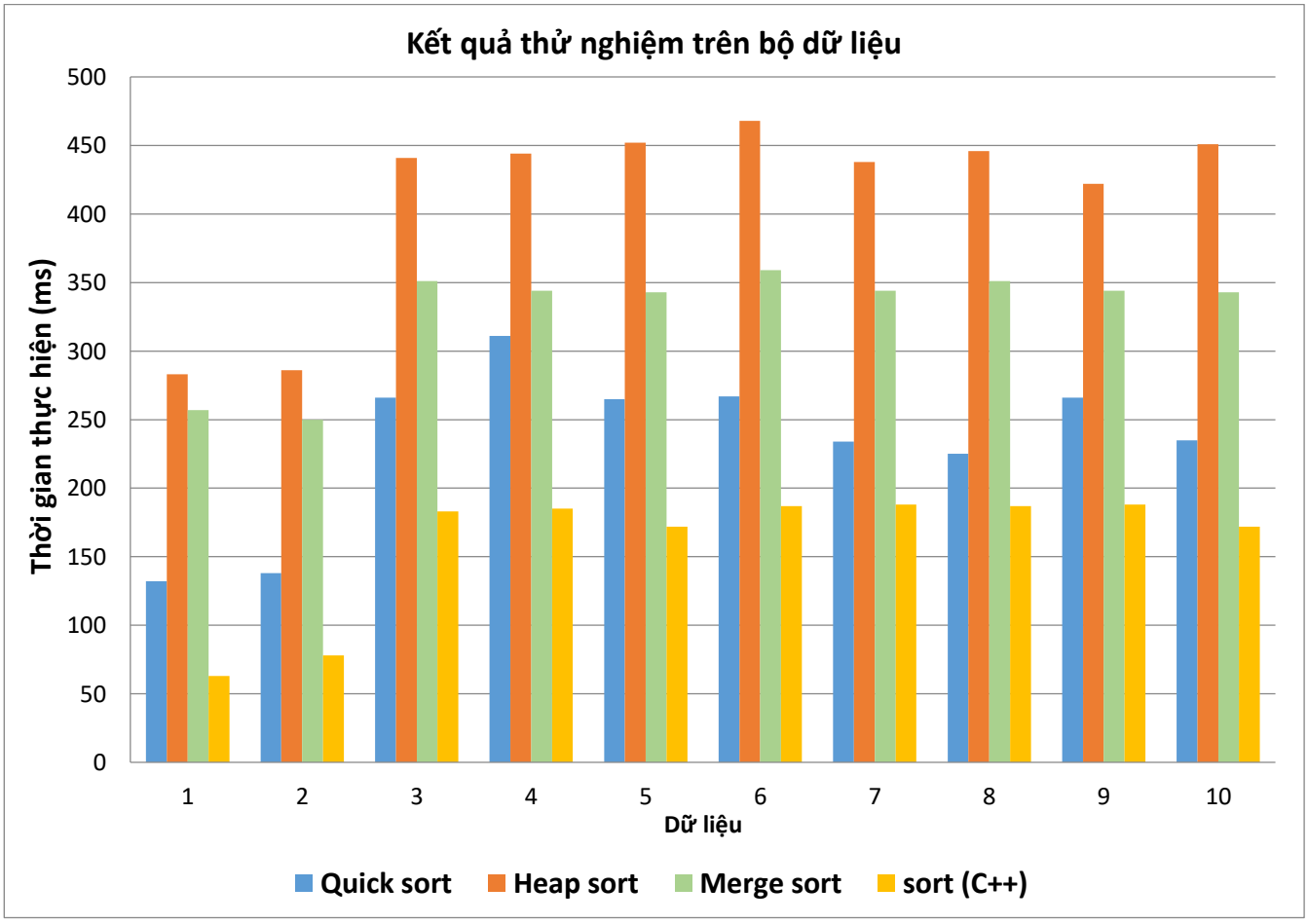
Nội dung báo cáo:

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	132	283	257	63	
2	138	286	250	78	
3	266	441	351	183	
4	311	444	344	185	
5	265	452	343	172	
6	267	468	359	187	
7	234	438	344	188	
8	225	446	351	187	
9	266	422	344	188	
10	235	451	343	172	
Trung bình	233.9	413.1	328.6	160.3	

2. Biểu đồ (cột) thời gian thực hiện:



[1] Số liệu chỉ mang tính minh họa

II. Kết luận:

Phân tích, so sánh các thuật toán sắp xếp giúp ta tiếp cận giải quyết vấn đề bằng cách sử dụng các giải thuật sắp xếp: chia để trị (Quick sort, Merge sort), tiếp cận hai con trỏ (Merge sort, Heap sort), … và sử dụng cấu trúc dữ liệu (heap).

Trên lý thuyết, độ phức tạp của 3 thuật toán sắp xếp (Quick sort, Heap sort, Merge sort) được cho ở bảng dưới đây:

Thuật toán	Tốt nhất	Trung bình	Xấu nhất
Quick sort	O(nlogn)	O(nlogn)	$O(n^2)$
Heap sort	O(nlogn)	O(nlogn)	O(nlogn)
Merge sort	O(nlogn)	O(nlogn)	O(nlogn)

Các thuật toán sắp xếp trên đều sử dụng phép toán so sánh là chủ yếu, vì vậy thao tác so sánh là yếu tố quyết định đến độ phức tạp của thuật toán. Độ phức tạp về thời gian trong trường hợp xấu nhất của các thuật toán sắp xếp ở trên có thể là O(nlogn) hoặc $O(n^2)$, hiệu suất trong trường hợp xấu nhất của Quick sort là $O(n^2)$, tuy nhiên, trung bình Quick sort hoạt động rất nhanh ở độ phức tạp thời gian O(nlogn), trên thực tế, khả năng xảy ra trường hợp xấu nhất là rất thấp khi tất cả các mảng đầu vào đều có khả năng xảy ra như nhau.

Quick sort và Heap sort là thuật toán sắp xếp tại chỗ vì nó có một số lượng không đổi các phần tử mảng đầu vào được lưu trữ bên ngoài mảng, trong khi Merge sort cần một không gian bổ sung.

Xét về độ ổn định, thuật toán Merge sort có độ ổn định tốt hơn Quick sort và Heap sort vì nó không thay đổi thứ tự của các phần tử có cùng giá trị, hai thuật toán còn lại thì ngược lại.

Sau thử nghiệm, với 10 bộ test dữ liệu ngẫu nhiên, trong 3 thuật toán sắp xếp, ta có thể thấy Quick sort có hiệu quả cao nhất (thời gian trung bình 233.9 ms), tiếp đến lần lượt là Merge sort (328.6 ms), Heap sort (413.1 ms).

Quick sort có code chặt chẽ và yếu tố hằng số ẩn trong thời gian chạy của nó là nhỏ, Quick sort là một thuật toán sắp xếp phổ biến để sắp xếp các mảng đầu vào lớn trong hầu hết các tình huống thực tế, ta cũng có thể thế nó cũng hoạt động tốt hơn Heap sort sau quá trình thử nghiệm.

Nếu sự ổn định là cần thiết và không gian có sẵn, Merge sort có thể là lựa chọn tốt nhất để triển khai.

Đối với hàm sort có sẵn trong thư viện algorithm của C++, nó là hàm intro – sort, là sự kết hợp của 2 thuật toán sắp xếp rất hiệu quả là Quick sort và Insertion sort. Độ phức tạp của hàm sort() là O(nlogn). Hàm sort() có thể sắp xếp các dãy dữ liệu lớn một cách nhanh chóng và hiệu quả. Sau quá trình thử nghiệm, ta có thể thấy hàm sort() có thời gian sắp xếp nhỏ (160.3 ms), hiệu quả hơn hẳn so với 3 thuật toán sắp xếp đã nói trên. Hơn nữa, hàm sort() rất dễ sử dụng và tiện lợi, chỉ cần truyền 2 con trỏ đến vị trí bắt đầu và kết thúc của dãy cần sắp xếp.

Nói tóm lại, xét trên lý thuyết và kết quả của quá trình thử nghiệm thực tế, có thể thấy trong 3 giải thuật sắp xếp, Quick sort có hiệu quả cao nhất, tiếp đến là Merge sort và Heap sort. Ngoài ra, hàm sort() có sẵn trong C++ có thời gian chạy nhỏ nhất so với 3 giải thuật sắp xếp trên, có hiệu quả cao nhất và cũng dễ sử dụng trong quá trình giải quyết vấn đề ở thực tế.

III. Thông tin chi tiết – link github:

Link github: https://github.com/ckha1410/IT003/tree/main

1. Báo cáo:
   Báo cáo kết quả thử nghiệm.
2. Mã nguồn:

   Chương trình sắp xếp theo các thuật toán: QuickSort, HeapSort, MegerSort, hàm sắp xếp sort (C++).

3. Dữ liệu thử nghiệm:

   10 bộ test dữ liệu.