Đánh giá hiệu suất các thuật toán sắp xếp trên các bộ test khác nhau

Sinh viên: Nguyễn Viết Duy - MSSV: 22520336

Class: IT003.N21.CTTN

Sắp xếp là một trong những thuật toán cơ bản và quan trọng trong lĩnh vực khoa học máy tính. Với sự phát triển của các công nghệ lưu trữ dữ liệu hiện nay, việc sắp xếp dữ liệu đó trở thành một yêu cầu cần thiết cho nhiều ứng dụng khác nhau. Vì vậy, nghiên cứu và so sánh hiệu suất của các thuật toán sắp xếp là một chủ đề nghiên cứu quan trọng trong lĩnh vực khoa học máy tính.

Thí nghiệm:

Link github: https://github.com/w1n-gl0ry/Data_Structures_and_Algotithms

(Source code, bộ test dữ liệu đều được lưu trữ trên github)

Để so sánh hiệu suất của các thuật toán sắp xếp, em sử dụng 4 thuật toán là Heap Sort, Merge Sort, Quick Sort và C library Sort để sắp xếp 10 bộ test khác nhau. Mỗi bộ test chứa 1 triệu dữ liệu số, trong đó test 1 các số tăng dần, test 2 các số giảm dần và các test còn lại là dữ liệu số ngẫu nhiên.

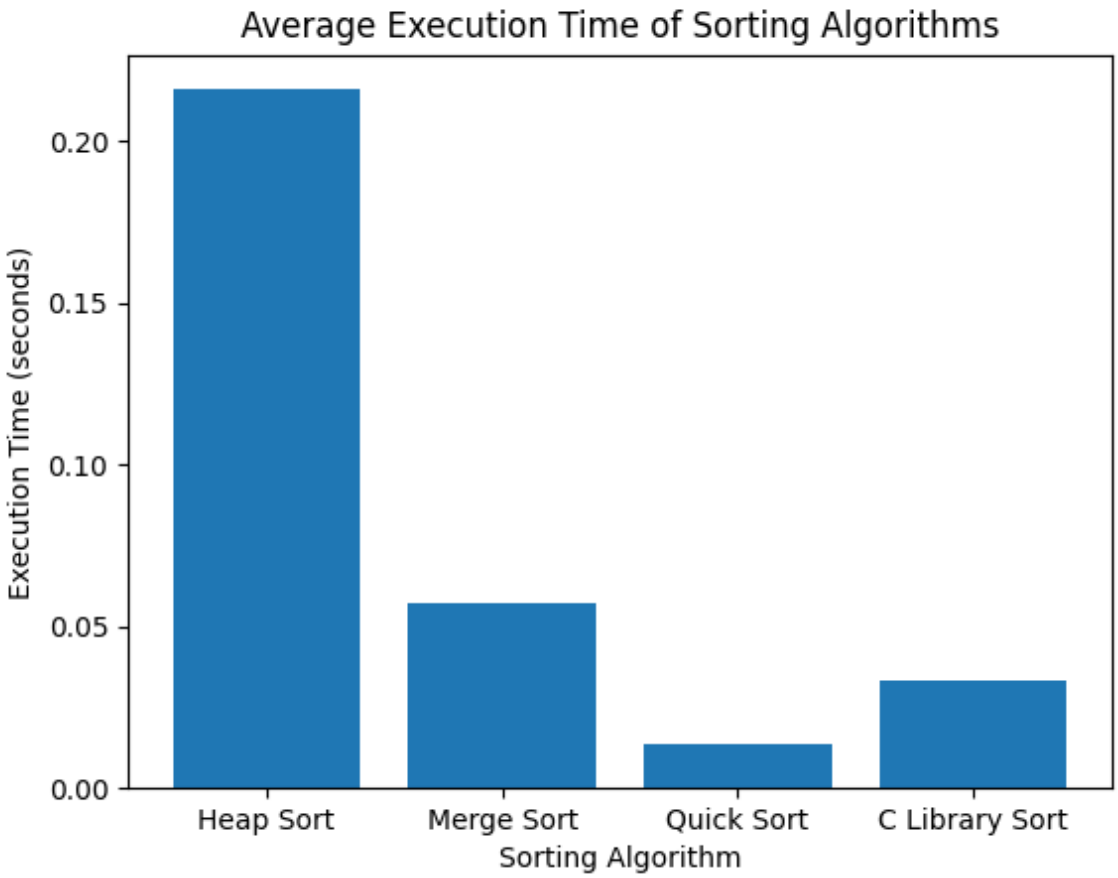
Các thuật toán được triển khai bằng ngôn ngữ lập trình C++ và đo thời gian thực thi bằng thư viện <sys/time.h> trong C++. Tất cả các thử nghiệm đều được em được thực hiện trên cùng một máy tính với cấu hình CPU Intel Core i5 gen 12th và RAM 8GB.

Kết quả thực nghiệm :

Bảng dữ liệu:

Thuật toán	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Heap Sort	0.210s	0.212s	0.212s	0.213s	0.211s	0.211s	0.220s	0.218s	0.227s	0.226s
Merge Sort	0.059s	0.054s	0.055s	0.055s	0.056s	0.059s	0.059s	0.060s	0.061s	0.057s
Quick Sort	0.015s	0.014s	0.013s	0.013s	0.015s	0.013s	0.013s	0.013s	0.014s	0.013s
C library Sort	0.034s	0.026s	0.033s	0.034s	0.033s	0.034s	0.034s	0.033s	0.036s	0.035s

Biểu đồ:



Bảng so sánh thời gian trung bình của các thuật toán cho mỗi bộ dữ liệu

Kết quả thực nghiệm cho thấy Quick Sort là thuật toán nhanh nhất với thời gian thực thi trung bình chỉ ở mức khoảng 0.013s. Merge Sort là thuật toán thứ hai nhanh nhất với thời gian thực thi trung bình ở mức khoảng 0.056s. Heap Sort là thuật toán thứ ba nhanh nhất với thời gian thực thi trung bình ở mức khoảng 0.214s và C library Sort là thuật toán chậm nhất với thời gian thực thi trung bình ở mức khoảng 0.033s.

Ngoài ra, kết quả thực nghiệm còn cho thấy, trong các bộ test ngẫu nhiên, Quick Sort là thuật toán có hiệu suất tốt nhất, trong khi Merge Sort là thuật toán tốt nhất trong các bộ test tăng dần và Heap Sort là thuật toán tốt nhất trong các bộ test giảm dần.

Phân tích kết quả:

Từ kết quả thực nghiệm trên, ta có thể thấy rằng trong các thuật toán sắp xếp được so sánh, Quick Sort là thuật toán nhanh nhất. Thời gian thực hiện của Quick Sort nằm trong khoảng từ 0.012 đến 0.015 giây cho tất cả các bộ test, nhanh hơn hẳn so với các thuật toán khác.

Merge Sort là thuật toán sắp xếp thứ hai nhanh nhất trong số các thuật toán này. Thời gian thực hiện của nó ở mức từ 0.054 đến 0.061 giây, tương đối nhanh so với Heap Sort và C library Sort.

C library Sort là thuật toán sắp xếp chậm thứ hai trong số các thuật toán này. Thời gian thực hiện của nó ở mức từ 0.026 đến 0.036 giây, tương đối chậm so với Quick Sort và Merge Sort.

Heap Sort là thuật toán sắp xếp chậm nhất trong số các thuật toán này. Thời gian thực hiện của nó ở mức từ 0.210 đến 0.226 giây, chậm hơn hẳn so với các thuật toán khác.

Đánh giá và nhận xét:

Quick Sort là thuật toán sắp xếp nhanh nhất trong số các thuật toán được so sánh. Nó được ưa chuộng và sử dụng rộng rãi trong các ứng dụng thực tế, đặc biệt trong các ứng dụng cần thời gian xử lý nhanh. Tuy nhiên, nó có thể gặp vấn đề với một số trường hợp xấu nhất, khi mà dãy số cần sắp xếp đã gần như được sắp xếp trước đó.

Merge Sort là thuật toán sắp xếp thứ hai nhanh nhất trong số các thuật toán này. Nó là một thuật toán ổn định và có hiệu quả trong hầu hết các trường hợp. Tuy nhiên, nó sử dụng bộ nhớ đệ quy để thực hiện, do đó có thể gặp vấn đề khi số lượng dữ liệu lớn.

C library Sort là thuật toán được sử dụng phổ biến nhất trong các thư viện sắp xếp của ngôn ngữ C. Nó là một thuật toán sắp xếp tổng quát, được thiết kế để hoạt

động tốt với mọi loại dữ liệu. Tuy nhiên, đối với các trường hợp đặc biệt như sắp xếp dữ liệu đã được sắp xếp trước đó hoặc dữ liệu đã được sắp xếp theo thứ tự ngược lại, Quick Sort lại cho kết quả tốt hơn. Điều này có thể được giải thích bằng cách Quick Sort sử dụng phương pháp chia để trị và hoán đổi dữ liệu, giúp giảm thiểu số lần so sánh và hoán đổi phần tử, do đó sắp xếp dữ liệu nhanh hơn trong các trường hợp đặc biệt này.

Ngoài ra, các thuật toán Heap Sort và Merge Sort có thể được cải tiến để tối ưu hóa hiệu suất. Ví dụ, Heap Sort có thể được cải tiến bằng cách sử dụng Heapify thay vì sắp xếp từng phần tử, và Merge Sort có thể được cải tiến bằng cách sử dụng insertion sort khi kích thước mảng là nhỏ hơn một ngưỡng nhất định.

Trong tổng quát, lựa chọn thuật toán sắp xếp phù hợp phụ thuộc vào loại dữ liệu cần được sắp xếp và các điều kiện đặc biệt trong quá trình sắp xếp. Việc hiểu rõ ưu nhược điểm của các thuật toán sắp xếp sẽ giúp cho nhà phát triển lựa chọn và tối ưu hóa thuật toán phù hợp cho ứng dụng của mình.