Bài tập thực hành

Lab2: Basic sorting

1. MỤC TIÊU

Trong bài tập này, các sinh viên cần ***thiết kế***, ***cài đặt***, và ***thực thi*** một thử nghiệm hoàn chỉnh. Các sinh viên sẽ tự mình ***thử nghiệm*** một nhóm các thuật toán quen thuộc: ***sắp xếp***.

Chúng ta đã được học rằng có các nhóm thực toán với thời gian chạy khác nhau. Chậm nhất là nhóm $O(n^2)$ như bubble sort, kế đến là nhóm $O(n \times log_n)$ tiêu biểu là quick sort, ngoài ra có một nhóm thuật toán đặc biệt như radix sort có thời gian chạy $O(n)$.

Tuy nhiên, đây là đánh giá độ phức tạp được áp dụng trên mức độ thuật toán, trên khái niệm, không phải trên cài đặt cụ thể của thuật toán đó dưới dạng một chương trình. Mặc dù cách đánh giá này là mạnh mẽ và hữu dụng nhưng nó không xem xét một số yếu tố quan trọng khác. Thứ nhất, khi cài đặt cụ thể với các cách cài đặt khác nhau, ngôn ngữ khác nhau, cấu trúc dữ liệu khác nhau sẽ ảnh hưởng đến hiệu năng của chương trình. Ngoài ra khi đánh giá độ phức tạp của thuật toán, người ta thường xem chi phí so sánh và hoán vị là như nhau. Điều kiện này không phải lúc nào cũng xảy ra. Các chi phí khác cũng thường không được quan tâm như chi phí hoạt động của vòng lặp. Với một giá trị n không quá lớn, chưa chắc một thuật toán thuộc lớp $O(n^2)$ chạy chậm hơn thuật toán thuật lớp $O(n \times log_n)$

Qua bài tập này sinh viên sẽ cài đặt các thuật toán so sánh đã học. Sau đó đo lường hiệu năng của các thuật toán thông qua thời gian chạy thực tế với các giá trị input khác nhau. Sau đó sinh viên sẽ phân tích kết quả và rút ra kết luận về các thuật toán sắp xếp khi có sự thay đổi kích thước mảng (***n***).

2. Các thuật toán sắp xếp

Sinh viên cần cài đặt các thuật toán sắp xếp sau đây để sắp xếp một mảng số nguyên theo thứ tự từ nhỏ đến lớn:

- Selection sort
- Insertion sort
- Interchange sort
- Bubble sort
- Shaker sort

Nên có hàm kiểm tra xem sau khi chạy thuật toán sắp xếp, mảng kết quả có đúng là đã có thứ tự hay không.

3. Thử nghiệm

3.1 Khởi tạo mảng ngẫu nhiên

Mảng input cho các thuật toán sắp xếp sẽ được khởi tạo ngẫu nhiên. Sinh viên cần viết hàm cho tác vụ này. Khi người dùng cung cấp hai số ***n*** và ***k***, chương trình sẽ phát sinh mảng gồm ***n*** phần tử, mỗi phần tử là số nguyên có giá trị từ [***0-k***].

Lưu ý: giá trị tối đa của số ngẫu nhiên phải lớn hơn 10^9

3.2 Đo thời gian

Sinh viên cần đo thời gian chạy của các thuật toán sắp xếp (theo ***millisecond***). Lưu ý chỉ đo thời gian chạy thuật toán sắp xếp, không đo các thời gian khác như thời gian phát sinh mảng.

Ở mỗi lần chạy sinh viên phải xuất ra: Tên thuật toán, loại input (xem bên dưới), số lượng input, và thời gian thực thi.

3.3 Thống kê

Xác định thời gian chạy thuật toán này với những giá trị input khác nhau và những kích thước ***n*** khác nhau. Trong bài tập này ta xét 4 loại input:

- Ngẫu nhiên

- Đã có thứ tự
- Có thứ tự ngược
- Gần như có thứ tự (Khoảng 5% số lượng phần tử sai vị trí)

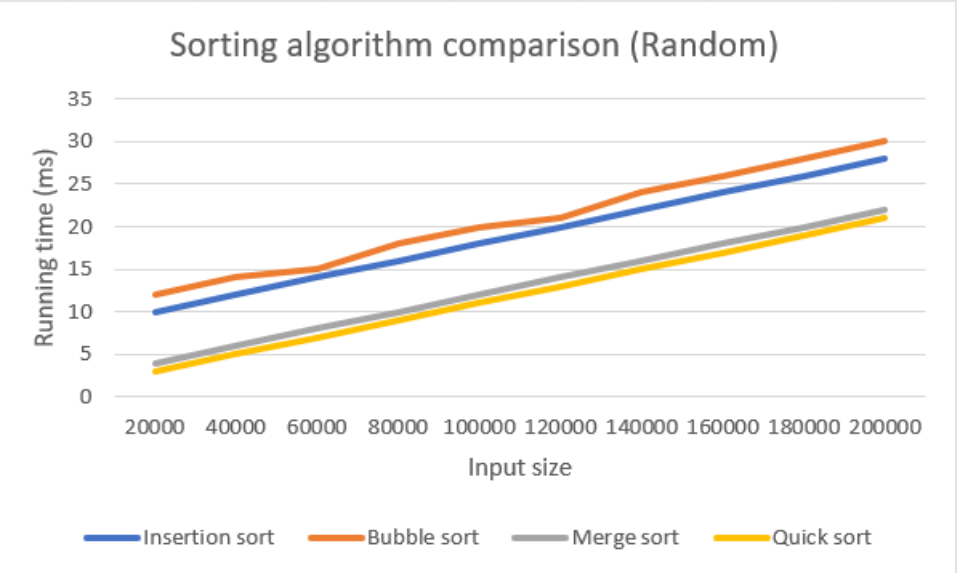
Thực hiện ***thống kê*** thời gian chạy của các thuật toán, ***vẽ biểu đồ***, và ***nhận xét kết quả***.

Thử nghiệm:

Số lượng phần tử ***n*** lần lượt là: 10^4, 2. 10^4, 4.10^4, 6.10^4, 8.10^4, 10.10^4, 12.10^4, 14.10^4, 16.10^4, 20.10^4

Ví dụ:

	20000	40000	60000	80000	100000	120000	140000	160000	180000	200000
Insertion sort	10	12	14	16	18	20	22	24	26	28
Bubble sort	12	14	15	18	20	21	24	26	28	30
Merge sort	4	6	8	10	12	14	16	18	20	22
Quick sort	3	5	7	9	11	13	15	17	19	21



4. Qui định nộp

- Sinh viên nộp một tập tin nén, có tên là <***MSSV***>***.zip*** hoặc <***MSSV***>***.rar*** chứa source code và báo cáo của chương trình.

Bài giống nhau hay nộp file rác sẽ 0 điểm MÔN HỌC.