**So sánh thời gian chạy của thuật toán với dữ liệu bằng thực nghiệm:**

Dữ liệu: Mảng một chiều với 10 phần tử với các giá trị từ 1 đến 10.

Mãng ngẫu nhiên sẽ được dùng để thực hiện sắp xếp:

[5, 8, 1, 3, 9, 4, 2, 10, 6, 7]

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sắp xếp nổi bọt (Bubble sort)** | **Sắp xếp chọn (Selection sort)** | **Sắp xếp chèn (Insertion Sort)** |
| **Dữ liệu sắp xếp theo thứ tự ngược lại** | 86ms | 66ms | 58ms |
| **Dữ liệu đã được sắp xếp** | 45ms | 72ms | 53ms |
| **Dữ liệu có xáo trộn ngẫu nhiên** | 88ms | 63ms | 70ms |

Nhận xét:

* Với dữ liệu được sắp xếp theo thứ tự ngược lại, sắp xếp chèn sẽ cho thời gian sắp xếp nhanh nhất, sau đó là sắp xếp chọn và cuối cùng là sắp xếp nổi bọt
* Với dữ liệu đã được sắp xếp: sắp xếp nổi bọt sẽ cho kết quả nhanh nhất, sau đó đến sắp xếp chèn và cuối cùng là sắp xếp chọn.

**Giải thích kết quả thực nghiệm.**

* Các thuật toán sắp xếp đều có hai vòng lặp được chạy lồng nhau, do đó độ phức tạp trung bình của các thuật toán sẽ là O(n2). Tuy nhiên các thuật toán này vẫn có những điểm khác biệt như sau, tương ứng với từng trường hợp cụ thể:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sắp xếp nổi bọt (Bubble sort)** | **Sắp xếp chọn (Selection sort)** | **Sắp xếp chèn (Insertion Sort)** |
| **Nhận xét:** | Là một thuật toán sắp xếp dễ hiểu, nhưng kém hiệu quả và ít khi được sử dụng trong thực tế | Thuật toán này tốt hơn thuật toán sắp xếp nổi bọt, và thuật toán này có thời gian chạy **không** phụ thuộc vào thứ tự của mảng ban đầu, do chúng ta luôn luôn chạy với 1 con trỏ từ phần tử đầu tiên đến phần tử cận cuối cùng, và một con trỏ chạy từ con trỏ đầu tiên đến phần tử sau cùng để tìm vị trí của phần tử nhỏ nhất trong những phần tử còn lại của mảng và thay đổi vị trí với phần tử ở vị trí bên phải | Đây là một thuật toán sắp xếp linh hoạt, hiệu quả ở những mảng ít phần tử, và thuật toán này chạy khá nhanh trên những mảng đã gần như được sắp xếp vì các bước để “đẩy” các phần tử để tạo ra slot cho item mới sẽ được giảm đi rất nhiều. |
| **Tình huống tốt nhất (best case)** | Độ phức tạp: O(n), đi qua đúng n phần tử, đối với mảng đã được sắp xếp hoặc gần như đã được sắp xếp, khi đó thuật toán sẽ dừng ở lần lặp đầu tiên, và không có sự hoán đổi vị trí nào được thực hiện | Như đã nói, thuật toán này có thời gian chạy không thực sự phụ thuộc vào thứ tự của mảng như thế nào, tuy nhiên với mảng đã được sắp xếp, thuật toán này sẽ chạy nhanh hơn vì không mất thời gian cho việc “swap” các phần tử | Best case: O(n), đối với mảng đã được sắp xếp, khi đó sắp xếp chèn sẽ chỉ duyệt mảng đúng một lần mà thôi. |
| **Tình huống xấu nhất:** | Độ phức tạp O(n2), mảng bị sắp xếp ngược, khi đó với mỗi lần duyệt của con trỏ đầu tiên, con trỏ thử hai sẽ duyệt để đưa phần tử nhỏ nhất về vị trí của nó (về bên trái) | Gần như không có worst case, thời gian chạy sẽ gần như nhau trong mọi trường hợp | O(n2), trường hợp mảng bị sắp xếp ngược, khi đó với mỗi lần duyệt từ con trỏ thứ hai về phần từ đầu tiên, nó luôn luôn phải duyệt tất cả phần tử để “đẩy” mảng lên, thay thế phần tử ở vị trí con trỏ đầu tiên cho phần tử đầu tiên này. |