**Nhận xét về kết quả thực nghiệm tính thời gian chạy của 3 thuật toán sắp xếp: sắp xếp nổi bọt, sắp xếp lựa chọn và sắp xếp chèn**

1. Best case (dữ liệu được sắp xếp theo đúng thứ tự)
   1. Thuật toán có thời gian thực thi chậm nhất : thuật toán sắp xếp lựa chọn (selection sort).
   2. Thuật toán có thời gian thực thi nhanh nhất : thuật toán sắp xếp chèn (insertion sort).
2. Worst case (dữ liệu được sắp xếp theo thứ tự ngược lại)
   1. Thuật toán có thời gian thực thi chậm nhất : thuật toán sắp xếp lựa chọn (selection sort).
   2. Thuật toán có thời gian thực thi nhanh nhất : thuật toán sắp xếp chèn (insertion sort).
3. Average case (dữ liệu được xáo trộn ngẫu nhiên)
   1. Thuật toán có thời gian thực thi chậm nhất : thuật toán sắp xếp lựa chọn (selection sort).
   2. Thuật toán có thời gian thực thi nhanh nhất : thuật toán sắp xếp chèn (insertion sort).
4. Ở thuật toán sắp xếp chèn (Insertion sort) có chênh lệch về thời gian thực thi giữa best case và worst case là lớn nhất, tiếp theo là thuật toán sắp xếp nổi bọt (bubble sort) và cuối cùng là thuật toán sắp xếp lựa chọn (selection sort).
5. Trong 1 số trường hợp, thuật toán sắp xếp nổi bọt (bubble sort) và thuật toán sắp xếp lựa chọn (selection sort) có thời gian thực thi tại average case lớn hơn tại worst case.

*Giải thích :* Có sự chênh lệch như vậy là dựa vào độ phức tạp thời gian (time complexity) và phần cứng máy tính (RAM, CPU).

**Phân tích công thức tính độ phức tạp thời gian (time complexity) của 3 thuật toán sắp xếp**

1. **Sắp xếp nổi bọt (bubble sort)**

Pseudo code:

for i = 0 to a.length – 2:

for j = 0 to a.length – 2 – i:

if (a[j] > a[j+1])

temp = a[j+1]

a[j+1] = a[j]

a[j] = temp

1. Worst case : array a = [n-1, n-2,…..,0] (n elements)
   * I = 0 => n – 1 swap steps, n-1 comparison step
   * I = 1 => n – 2 swap steps, n-2comparison step
   * …
   * I = n – 2 => 1 swap steps, 1 comparison step

* Total steps = ((n-1) + (n-2) + …. + 1) = n\*(n-1)/ 2 = n2 - n.
* **Time complexity : O(n2).**

1. Best case : array a = [0, 1, 2, …., n-1]
   * I = 0 => 0 swap step (because no swap occurred), 1 comparison step
   * I = 1 => 0 swap step (because no swap occurred), 2 comparison step
   * …
   * I = n – 2 => 0 swap step (because no swap occurred), n-1 comparison step

* Total step = n - 1
* **Time complexity : O(n).**

1. Average case : array a = [0, 1, 2…., n/2 - 1, n-1, n-2,…n/2]
   * I = 0 => 0 swap step (because no swap occurred), 1 comparison step
   * …
   * I = n/2 - 1=> (n/2 – 1) swap step, 1 comparison step
   * …
   * I = n/2 => 0 swap step, 1 comparison step

* Total step = (n/2 – 1 + n/2 – 2+ … + 0) = n2 - n.
* **Time complexity : O(n2).**

1. **Sắp xếp lựa chọn (selection sort)**

Pseudo code:

for i = 0 to a.length – 2

int min = i

for j = i + 1 to a.length – 1

if (a[min] > a[j])

min = j;

temp = a[min]

a[min] = a[i]

a[i] = temp

For all cases: best case, worst case and average case, the selection sort has the same time complexity because we need to loop from 0 to array’s length to find the minimum item.

* I = 0 => n – 1 comparison step to find minimum
* I = 1 => n – 2 comparison step to find minimum
* …
* I = n - 1 => 0 comparison step to find minimum
* Total step : (n -1) + …. + 0 = (n-1) \* n / 2 = n2 - n
* **Time complexity : O(n2).**

1. **Sắp xếp chèn (Insertion sort)**

Pseudo code:

for i = 0 to a.length – 1

current = a[i]

j = i – 1

while (j >= 0 && a[j] > current)

a[j+1] = a[j]

j = j – 1

a[j+1] = current

1. Worst case : array a = [n-1, n-2,…., 0]
   * I = 0 => 1 swap step, 1 comparison
   * I = 1 => 2 swap step, 2 comparison
   * …
   * I = n – 2 = > n – 1 swap step, n-1 comparison

* Total step = (n-1) + … + 1 = n\*(n – 1) / 2 = n2 - n
* **Time complexity : O(n2).**

1. Best case : array a = [0,1,…..n-1]
   * I = 0 => 0 swap step, 1 comparison
   * I = 1 => 0 swap step, 2 comparison
   * …
   * I = n – 2 => 0 swap step, n-1 comparison

* Total step = n – 1
* **Time complexity : O(n).**

1. Average case : array a = [0,1,…,n/2,n-1,…n/2-1]
   * I = 0 => 0.5 swap step, 1 comparison
   * I = 1 => 1 swap step, 2 comparison
   * …
   * I = n – 2 => (n – 2) /2 swap step, n -1 comparison

* Total step = 0.5 + 1 + … + (n – 2)/2 = n \* (n – 1) / 4 = n2 - n
* **Time complexity : O(n2).**