**1.a./ Kết quả test: (chi tiết tại class TestAlgorithm)**

Worst Cases:

Bubble Sort Running Time: 9.156ms

Selection Sort Running Time: 4.5544ms

Insertion Sort Running Time: 3.645101ms

Best Cases:

Bubble Sort Running Time: 0.0224ms

Selection Sort Running Time: 4.0882ms

Insertion Sort Running Time: 0.0451ms

Average Cases:

Bubble Sort Running Time: 7.017901ms

Selection Sort Running Time: 3.6029ms

Insertion Sort Running Time: 2.711901ms

-----------------------------------------------------------------------------------------------

**1.b./ Nhận xét hiệu năng của 3 thuật toán với các kiểu dữ liệu khác nhau:**

- Dữ liệu sắp xếp theo thứ tự ngược lại (worst case):

* Bubble sort: chạy chậm nhất so với tất cả các thuật toán khác cũng như các case khác – chậm.
* Selection sort: chạy nhanh hơn so với bubble sort – trung bình.
* Insertion sort: chạy nhanh hơn 2 thuật toán còn lại – nhanh.

- Dữ liệu đã được sắp xếp (best case)::

* Bubble sort: chạy rất nhanh - nhanh.
* Selection sort: chạy chậm hơn so với 2 thuật toán còn lại – trung bình.
* Insertion sort: chạy rất nhanh - nhanh.

- Dữ liệu có xáo trộn ngẫu nhiên (average case)::

* Bubble sort: chạy chậm hơn so với 2 thuật toán còn lại – chậm.
* Selection sort: chạy nhanh hơn bubble sort – trung bình.
* Insertion sort: chạy nhanh hơn 2 thuật toán còn lại – nhanh.

-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------

**2.a/ Nhận xét và giải thích kết quả thực nghiệm**

* Bubble sort: chạy nhanh hơn đối với best case – khi các phần tử đã được sort trước, các case khác chạy chậm tương đương nhau.
  + Best case chạy nhanh hơn vì khi thứ tự đã được sắp xếp đúng thì chỉ cần chạy một vòng lặp đầu tiên trên toàn bộ array để so sánh, không cần phải bắt đầu những vòng lặp khác.
* Selection sort: tất cả các case đều chạy chậm tương đương nhau.
* Insertion sort: chạy nhanh hơn đối với best case – khi các phần tử đã được sort trước, các case khác chạy chậm tương đương nhau.
  + Best case chạy nhanh hơn vì khi thứ tự đã được sắp xếp đúng thì chỉ cần chạy một vòng lặp đầu tiên trên array để so sánh, không cần phải bắt đầu những vòng lặp khác.

**2.b/ Phân tích công thức tính độ phức tạp thời gian**

Công thức tính O(n^2): khi nesting 2 vòng lặp

n + (n-1) + (n-2) + … + 2 + 1 = n\*(n-1)/2 = 0.5\*n^2 + 0.5\*n (công thức tính Triangular Number)

* O(n^2)
* Bubble sort:
  + Best case: Time complexity O(n) - xảy ra khi array đã được sort cùng chiều, function chỉ lặp qua 1 lần trên array size n để so sánh các phần tử mà không cần swap.
  + Worst case: Time complexity O(n^2) - xảy ra khi array được sort ngược chiều, function lặp qua array, trong mỗi vòng lặp này sẽ có những vòng lặp bên trong để so sánh và swap các phần từ.
  + Average case: Time complexity O(n^2) - trong trường hợp bình thường với các phần tử ngẫu nhiên, cũng phải chạy 2 vòng lặp tương tự như worst case.
* Selection sort:
  + Tất cả các case đều có time complexity O(n^2) – vì đều cần chạy một vòng lặp trên array, trên mỗi vòng lặp này đều phải có những vòng lặp bên trong để so sánh và tìm giá trị max/min, chức năng swap cũng chỉ xảy ra tối đa là n lần nên không được tính.
* Insertion sort:
  + Best case: Time complexity O(n) - xảy ra khi array đã được sort cùng chiều, function chỉ lặp qua 1 lần trên array từ index 1 đến index (n – 1) để so sánh các phần tử mà không cần triển khai vòng lặp bên trong do không đáp ứng điều kiện.
  + Worst case: Time complexity O(n^2) - xảy ra khi array được sort ngược chiều, function lặp qua array, trong mỗi vòng lặp này sẽ có những vòng lặp bên trong để so sánh và shift các phần từ.
  + Average case: Time complexity O(n^2) - trong trường hợp bình thường với các phần tử ngẫu nhiên, cũng phải chạy 2 vòng lặp tương tự như worst case.