1/. Nhận xét hiệu năng của 3 thuật toán với các kiểu dữ liệu (n) khác nhau:

* Kích thước dữ liệu 1000 phần tử

| Thời gian (millisecond) | Bubble Sort | Selection Sort | Insertion Sort |
| --- | --- | --- | --- |
| Mảng tăng dần (best case) | 0.1653 | 13.6104 | 0.233 |
| Mảng giảm dần (worst case) | 29.0163 | 5.4755 | 9.7098 |
| Mảng ngẫu nhiên (average) | 32.7918 | 46.7217 | 1.8714 |

* Nhận xét:

Với kích thước dữ liệu nhỏ:

* Ở giải thuật Bubble Sort: thời gian sắp xếp ở mảng tăng dần nhanh nhất, mảng giảm dần xấp xỉ mảng ngẫu nhiên
* Ở giải thuật Selection Sort: thời gian sắp xếp ở mảng giảm dần nhanh nhất, xấp xỉ mảng tăng dần và nhanh hơn mảng ngẫu nhiên
* Ở giải thuật Insertion Sort: thời gian sắp xếp ở mảng tăng dần tương đương mảng ngẫu nhiên và nhanh hơn mảng giảm dần
* Kích thước dữ liệu 10000 phần tử

| Thời gian (millisecond) | Bubble Sort | Selection Sort | Insertion Sort |
| --- | --- | --- | --- |
| Mảng tăng dần (best case) | 0.5959 | 194.4355 | 0.3096 |
| Mảng giảm dần (worst case) | 246.99101 | 198.1811 | 157.30609 |
| Mảng ngẫu nhiên (average) | 250.8684 | 207.42259 | 58.7205 |

* Nhận xét:

Với kích thước dữ liệu trung bình:

* Ở giải thuật Bubble Sort: thời gian sắp xếp ở mảng tăng dần nhanh nhất, mảng giảm dần xấp xỉ mảng ngẫu nhiên
* Ở giải thuật Selection Sort: thời gian sắp xếp ở cả 3 mảng tương đương nhau
* Ở giải thuật Insertion Sort: thời gian sắp xếp ở mảng tăng dần nhanh nhất, mảng ngẫu nhiên nhanh hơn mảng giảm dần
* Kích thước dữ liệu 100000 phần tử

| Thời gian (millisecond) | Bubble Sort | Selection Sort | Insertion Sort |
| --- | --- | --- | --- |
| Mảng tăng dần (best case) | 0.7886 | 20184.572 | 1.2079 |
| Mảng giảm dần (worst case) | 24825.984 | 20101.275 | 10111.629 |
| Mảng ngẫu nhiên (average) | 29829.709 | 19971.834 | 4825.0664 |

* Nhận xét:

Với kích thước dữ liệu lớn:

* Ở giải thuật Bubble Sort: thời gian sắp xếp ở mảng tăng dần nhanh nhất, mảng giảm dần xấp xỉ mảng ngẫu nhiên
* Ở giải thuật Selection Sort: thời gian sắp xếp ở cả 3 mảng tương đương nhau
* Ở giải thuật Insertion Sort: thời gian sắp xếp ở mảng tăng dần nhanh nhất, mảng ngẫu nhiên nhanh hơn mảng giảm dần
* Nhận xét chung:

Kích thước dữ liệu càng lớn thì thời gian chạy càng lâu ⇒ Độ phức tạp tuyến tính: O(n). Số phép tính/thời gian chạy/dung lượng bộ nhớ có xu hướng tỉ lệ thuận với độ lớn đầu vào.

Insertion Sort: thời gian thực hiện tốt hơn khi mảng đã sắp xếp.

Selection Sort: thời gian thực hiện hầu như không phụ thuộc vào các phần tử đang được sắp xếp.

So sánh cả 3 phương pháp thì sắp xếp bằng giải thuật Insertion Sort cho ta kết quả nhanh nhất.

2/. Tính thời gian thực hiện của các thủ tục theo các trường hợp

1. Tính thời gian thực hiện của Bubble Sort

for (int i = 0; i < n - 1; i++) { {1}

swapped = false; {2}

for (int j = 0; j < n - 1 - i; j++) {3}

if (arr[j] > arr[j + 1]) { {4}

float x = arr[i]; {5}

arr[i] = arr[k]; {6}

arr[k] = x; {7}

swapped = true; {8}

}

if (!swapped) { {9}

break; {10}

}

}

* Trường hợp tốt nhất:

- Lệnh gán {2} và lệnh dừng {10} tốn O(1) thời gian, chương trình chạy vòng lặp {1} từ 0 đến n - 2 nên thời gian thực hiện của vòng lặp {1} cũng là độ phức tạp của giải thuật là: T(n) = (n -1).1 = O(n).

* Trường hợp trung bình:

- Cả bốn lệnh gán {5}, {6}, {7} và {8} đều tốn O(1) thời gian, việc so sánh arr[j] > arr[j + 1] cũng tốn O(1) thời gian, do đó lệnh {4} tốn O(1) thời gian.

- Vòng lặp {3} thực hiện (n - i) lần, mỗi lần O(1) do đó vòng lặp {3} tốn O((n-i).1) = O(n-i).

- Vòng lặp {1} lặp có i chạy từ 0 đến n-2 nên thời gian thực hiện của vòng lặp {1} và cũng là độ phức tạp của giải thuật là:

T(n) = n\*(n-1)/2 = O(n^2)

1. Tính thời gian thực hiện của Selection Sort

for (int i = 0; i < n - 1; i++) { {1}

int min = i; {2}

for (int j = i + 1; j < n; j++) {3}

if (arr[j] < arr[min]) {4}

min = j; {5}

if (min != i) { {6}

float x = arr[i]; {7}

arr[i] = arr[k]; {8}

arr[k] = x; {9}

}

}

* Trường hợp tốt nhất chương trình chỉ chạy vòng lặp {1} và lệnh gán {2} nên thời gian thực hiện của vòng lặp {1} cũng là độ phức tạp của giải thuật là: T(n) = (n-1).1 = O(n).
* Trường hợp trung bình: chương trình chạy vòng lặp {1} với lệnh gán {2}, vòng lặp {3} và lệnh {6}

- Lệnh gán {5} tốn O(1) thời gian,việc so sánh arr[j] < arr[min] cũng tốn O(1) thời gian, do đó lệnh {4} tốn O(1) thời gian. Vòng lặp {3} thực hiện (n-i) lần, mỗi lần O(1) do đó vòng lặp {3} tốn O((n-i).1) = O(n-i).

- Cả ba lệnh gán {7}, {8} và {9} đều tốn O(1) thời gian, việc so sánh min != i cũng tốn O(1) thời gian, do đó lệnh {6} tốn O(1) thời gian.

- Do đó thời gian thực hiện của các lệnh nối tiếp trong vòng lặp {1} là O(max(1, n-i, 1) = O(n-i).

- Vòng lặp {1} lặp có i chạy từ 0 đến n-2 nên thời gian thực hiện của vòng lặp {1} và cũng là độ phức tạp của giải thuật là:

T(n) = n\*(n-1)/2 = O(n^2)

1. Tính thời gian thực hiện của Insertion Sort

for (int i = 1; i < n; i++) { {1}

int j = i - 1; {2}

float temp = arr[i]; {3}

while (j >= 0 && arr[j] > temp) { {4}

arr[j + 1] = arr[j]; {5}

j--;

}

arr[j + 1] = temp; {6}

}

* Trường hợp tốt nhất chương trình chạy 3 lệnh gán {2}, {3} và {6} đều tốn O(1) thời gian nên thời gian thực hiện của vòng lặp {1} cũng là độ phức tạp của giải thuật là: T(n) = n-1 = O(n).
* Trường hợp trung bình:

- Cả ba lệnh gán {2}, {3} và {6} đều tốn O(1) thời gian.

- Việc so sánh j >= 0 && arr[j] > temp tốn O(1) thời gian. Vòng lặp {4} thực hiện (i-1) lần, mỗi lần O(1) do đó vòng lặp {2} tốn O((i-1).1) = O(i-1).

- Do đó thời gian thực hiện của các lệnh nối tiếp trong vòng lặp {1} là O(max(1, i-1)) = O(i-1).

- Vòng lặp {1} lặp có i chạy từ 1 đến n-1 nên thời gian thực hiện của vòng lặp {1} và cũng là độ phức tạp của giải thuật là:

T(n) = n\*(n-1)/2 = O(n^2)

**Độ phức tạp trong các thuật toán sắp xếp**

|  | Bubble Sort | Selection Sort | Insertion Sort |
| --- | --- | --- | --- |
| Best Case | O(n) | O(n) | O(n) |
| Average Case | O(n^2) | O(n^2) | O(n^2) |
| Worst Case | O(n^2) | O(n^2) | O(n^2) |