ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC

THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

LAB 6: SẮP XẾP

NGUYỄN HOÀNG THÁI – 19120650

Giảng Viên: Lê Đình Ngọc

---- TP Hồ Chí Minh ----

04/01/2021

MỤC LỤC

PHẦN 1: CÁC THUẬT TOÁN SẮP XẾP

Selection Sort

- **Ý tưởng**: Cho 1 mảng gốc có n phần tử, ta sẽ chia nó ra làm hai mảng con, mảng bên trái là mảng đã sắp xếp, mảng bên phải là mảng chưa sắp xếp. Ban đầu, mảng bên trái chưa có phần tử nào nghĩa là rỗng và mảng bên phải chứa toàn bộ phần tử mảng ban đầu. Ta thực hiện duyệt mảng, nếu phát hiện phần tử nhỏ nhất trong mảng bên phải (unsorted array), ta sẽ hoán vị nó cho phần tử trái nhất của mảng gốc, và phần tử nhỏ nhất đó sau khi hoán vị sẽ trở thành phần tử của mảng bên trái. Sau đó, ta sẽ không quan tâm đến phần tử đó nữa, và xem mảng bên phải chỉ còn n-1 phần tử, nghĩa là bắt đầu từ vị trí thứ 2 và tiếp tục xem trong mảng bên phải đó phần tử nhỏ nhất để hoán vị với phần tử trái nhất của mảng bên phải nữa (lúc này đang ở vị trí thứ 2) và thêm nó vào mảng bên phải. Qúa trình cứ tiếp tục như thế cho đến khi mảng bên phải chỉ còn 1 phần tử.

- **Các bước thuật toán**:
  - Bước 1: Cho i=0;
  - Bước 2: Gán min=i;
  - Bước 3: Duyệt mảng từ i đến n-1 để tìm phần tử nhỏ nhất a[nhỏ_nhất], hoán vị a[nhỏ nhất] và a[min].
  - Bước 4: Nếu i<n-1 thì i++, trở lại bước 2. Nếu không kết thúc thuật toán.

- **Đánh giá thuật toán**:
  - Vì có 2 vòng lặp lồng nhau nên ở lượt thứ i, bao giờ ta cũng cần (n-i-1) số lần so sánh.
  - Không phụ thuộc vào tình trạng dãy số ban đầu

- Trong mọi trường hợp, số lần so sánh là:$\sum_{i=0}^{n-1}(n-i-1)=\frac{n(n-1)}{2}$ cho nên độ phức tạp về thời gian ở trường hợp tốt nhất, trung bình và xấu nhất đều là $O(n^2)$ và độ phức tạp về không gian $O(1)$.

Merge Sort

- **Ý tưởng**: Dựa trên kĩ thuật chia để trị để sắp xếp mảng thành 2 phần bằng nhau hoặc chênh nhau 1 đơn vị. Thực hiện lại việc này với các mảng con đó nghĩa là tiếp tục chia các mảng con này thành hai mảng con nữa và tiếp tục lặp như thế. Sau cùng, ta gộp các nửa mảng đó lại với nhau bằng cách so sánh các phần tử, lấy phần tử đó cho vào dãy mới. Cuối cùng sau khi các nửa mảng đã được chia rỗng, ta sẽ có một dãy con không giảm -> dãy đã được sắp xếp.

- **Thuật toán**:
  - Bước 1: Chia mảng ban đầu thành 2 mảng con (mảng trái và phải) có chiều dài tương đương nhau.
  - Bước 2: Sắp xếp mảng bên trái và Sắp xếp mảng bên phải.
  - Bước 3: Gộp 2 mảng đã được sắp xếp lại với nhau.
  - Bước 4: Trong khi phần tử first < last thì gọi đệ quy lại bước 1. Nếu không thì dừng thuật toán, mảng lúc này đã được sắp xếp.

- **Đánh giá**:
  - Vì thời gian của việc chia mảng ở bước 2 cũng như gộp mảng ở bước 3 đều bằng $O(log_2 n)$ cho nên nó không phụ thuộc vào thứ tự (đặc tính của dãy)-> chi phí thuật toán là không đổi, còn chi phí thực hiện việc trộn hai dãy con đã sắp xếp tỷ lệ thuận với n. Vậy trong trường hợp tốt, trung bình, xấu nhất thì độ phức tạp thời gian đều là $O(n\ log_2 n)$. Độ phức tạp về không gian là: $O(n)$.

Heap Sort

- **Ý tưởng**: Dựa vào cấu trúc Binary Heap (cây nhị phân hoàn chỉnh) xây dựng max heap để sắp xếp các phần tử theo thứ tự tăng dần. Sau khi đã xây dựng xong heap, ta sẽ hoán vị node gốc với node lá cuối của cây và xóa node cuối đó ra khỏi heap, khi đó node đã được xóa đó sẽ ở vị trí n-1 của mảng sắp xếp và tiếp tục lặp lại bước ở trên.

- **Thuật toán**:
  - Bước 1: Hiệu chỉnh mảng ban đầu thành max heap.
  - Bước 2: Đưa phần tử lớn nhất ra khỏi về vị trí cuối của dãy.
  - Bước 3: Xóa phần tử lớn nhất ra khỏi heap, số phần tử của mảng n-1 và hiệu chỉnh lại dãy.
  - Bước 4: Lặp lại bước 2. Nếu chỉ còn một phần tử ta dừng thuật toán.

- **Đánh giá**:
  - Khi xây dựng heap sẽ tốn chi phí là $log_2 n$. Mỗi phần tử ta xây dựng lại heap một lần ở bước 3, mà ta có n phần tử. Vậy chi phí cho sắp xếp ở mọi trường hợp *là O(n* $log_2 n$*)* và phức tạp không gian là *O(1).*

Quick Sort

- **Ý tưởng**: Chia mảng gốc thành 2 mảng con bằng cách so sánh từng phần tử của danh sách với một phần tử được chon làm mốc (gọi là pivot). Những phần tử nhỏ hơn hoặc bằng phần tử mốc sẽ đưa về phía trước nghĩa là nằm trong mảng bên trái phần tử mốc, các phần tử lớn hơn phần tử mốc sẽ đưa về phía sau nó và thuộc màng bên phải. Cứ tiếp tục chia như thế cho tới khi các mảng con chỉ còn 1 phần tử.

- **Thuật toán**:
  Cho a[ ]={l,….,r} có n phần tử
  - Bước 1: Chọn phần tử a[k] bất kỳ trong mảng thuộc đoạn [l;r] để làm phần tử mốc (pivot), thường ta chon phần tử trái nhất, phải nhất hoặc ở giữa mảng. Gán pivot=a[k], i=l , j=r. Trong bài, em chọn phần tử ở giữa k= (l+r)/2.
  - Bước 2: Phát hiện và hiệu chỉnh cặp a[i], a[j] nằm sai chỗ.
    - Bước 2.1: Trong khi (a[i] <x) thì i++;
    - Bước 2.2: Trong khi (a[j]>x) thì j--;
    - Bước 2.3: Nếu i<= j thì hoán vị a[i] với a[j]; Tăng I và giảm j;
  - Bước 3: Nếu i< j thì trở lại bước 2, nếu không kết thúc thuật toán.

- **Đánh giá**:
  - Hiệu quả phụ thuộc vào việc chọn mốc pivot của mảng. Lựa chọn tốt nhất khi ta chọn phần tử median (trung vị).
  - Dữ liệu đầu vào khi đã sắp xếp tang dần hoặc giảm dần, pivot là điểm giữa -> đây là trường hợp tốt nhất với độ phức tạp về thời gian là *O(n*log(n)).*
  - Trường hợp xấu nhất có độ phức tạp là*: O(*n^2).
  - Độ phức tạp về không gian: *O(log(n)).*

Bubble Sort

- **Ý tưởng**: Cho mảng gồm có n phần tử và ta muốn sắp xếp tang dần. Ta xem xét lần lượt 2 phần tử liên tiếp nhau (ban đầu là 0 và 1) và hoán vị lẫn nhau nếu phần tử thứ 1 nhỏ hơn phần tử thứ 0. Tiếp tục các bước như thế cho cặp thứ 1 và 2, 2 và 3,… Cuối cùng, ta sẽ được phần tử lớn nhất ở vị trí cuối mảng. Ta không quan tâm đến phần tử lớn nhất đó nữa và lặp lại những bước trên cho đến khi không còn phần tử để xét.

- **Thuật toán**:
  - Bước 1: i=0 //Bước xử lý thứ 1.
  - Bước 2: j=0. . Nếu (j< n-i-1) thì ta thực hiện so sánh (a[j]>a[j+1]) nếu đúng hoán vị 2 phần tử này; tăng j lên 1 đơn vị.
  - Bước 3: i +=1 . Lặp lại bước 2, nếu i> n-1 thì kết thúc thuật toán.

- **Đánh giá**:
  - Thuật toán này không phụ thuộc vào tình trạng ban đầu cùa dãy, nghĩa là các phép so sánh sẽ không phụ thuộc vào dãy đã được sắp xếp hay chưa, số lượng phép hoán vị cũng phụ thuộc vào kết quả so sánh:
    + Trường hợp tốt nhất: là khi mảng tăng dần, lúc này thuật toán chỉ thực hiện phép so sánh $\frac{n(n-1)}{2}$ lần nhưng sẽ không có hoán vị phần tử nào cả nên độ phức tạp của nó là *O(n).*
    + Trường hợp xấu nhất: vừa thực hiện phép so sánh $\frac{n(n-1)}{2}$ lần và hoán vị $\frac{n(n-1)}{2}$ lần nên độ phức tạp của nó là $O(n^2)$.
  - Độ phức tạp về không gian là: *O(1).*

Insertion Sort

- **Ý tưởng**: Giả sử ta có mảng n phần tử, phần tử đầu tiên của mảng đó ta xếp vào mảng đã được sắp xếp (sorted array) và so sánh với tử thứ 2, nếu phần tử thứ 2 nhỏ hơn phần tử thứ 1 thì ta thực hiện chèn nó trước phần tử thứ nhất, tương tự so sánh phần tử thử 3 với 2 phần tử đầu tiên (2 phần tử này đã ở trong sorted array) và chèn nó một cách thích hợp sao cho 3 phần tử này có thứ tự tăng dần. Nói cách khác, ta sẽ chèn phần tử mới a[i] vào vị trí thích hợp giữa a[k-1] va a[k] với (1< k< i] sao cho đoạn đó được sắp xếp tăng dần.

- **Thuật toán**:
  - Bước 1: Cho phần tử a[0] đã được sắp xếp, nên sẽ bắt đầu i= 1
  - Bước 2: x=a[i] và tìm vị trí thích hợp pos trong đoạn a[0] đến a[i-1] để chèn a[i] vào.
  - Bước 3: Dời chỗ các phần tử từ pos tới i-1 sang phải 1 đơn vị để thay a[i] vào
  - Bước 4: Gán a[pos]=x;
  - Bước 5: Tăng i lên 1 đơn vị. Nếu i <= n thì lặp lại bước 2. Không thì kết thúc thuật toán.

- **Đánh giá**:
  - Ở trường hợp mảng đã được xếp tăng dần, mảng chỉ thực hiện so sánh n-1 lần (do có n-1 bước) mà không có hoán vị nên độ phức tạp về thời gian là $O(n)$.
  - Thuật toán sử dụng trung bình $n^2/4$ phép so sánh và $n^2/4$ lần hoán vị, $n^2/2$ phép so sánh và $n^2/2$ lần hoán vị trong trường hợp xấu nhất nên độ phức tạp về thời gian là $O(n^2)$.
  - Độ phức tạp về không gian là: $O(1)$.
  - Thuật toán chỉ thích hợp sử dụng đối với mảng được sắp xếp một phần hoặc mảng có số lượng phần tử nhỏ.

Binary Insertion Sort:

- **Ý tưởng**: Thuật này cải tiến của thuật toán insertion sort. Sử dụng tìm kiếm nhị phần binary search để làm giảm số phép so sánh, như thế ta sẽ tìm kiếm vị trí thích hợp để chèn sau mỗi lần lặp.

- **Thuật toán**:
  - Bước 1: Cho phần tử a[0] đã được sắp xếp, nên sẽ bắt đầu i=1
  - Bước 2: Gán selected = a[i] để tìm vị trí thích hợp pos cho nó bằng cách sử dụng hàm binarySearch()

- Bước 3: Dời chỗ các phần tử từ vị trí đã tìm được ở bước 2 tới i-1 sang phải 1 đơn vị để thay selected vào
- Bước 4: Gán a[pos] = selected
- Bước 5: Tăng i lên 1 đơn vị. Nếu I <= n thì lặp lại bước 2, không thì kết thúc thuật toán.

- **Đánh giá**:
  - Insertion sau khi thực hiện tìm kiếm nhị phân sẽ tốn *O(n).*
  - Nếu binary insertion sort chỉ thực hiện n-1 phép so sánh mà không hoán vị thì độ phức tạp thời gian của nó là *O(n*logn).*
  - Trong trường hợp xấu nhất: nó phải thực hiện hoán vị thì độ phức tạp về thời gian là $O(n^2)$.
  - Độ phức tạp về không gian là: *O(1).*

PHẦN 2: TRÌNH BÀY KẾT QUẢ THỰC NGHIỆM

Đây là bảng thời gian thực thi (đơn vị: giây) của các thuật toán sắp xếp trong 4 trường hợp sau khi chạy chương trình debug:

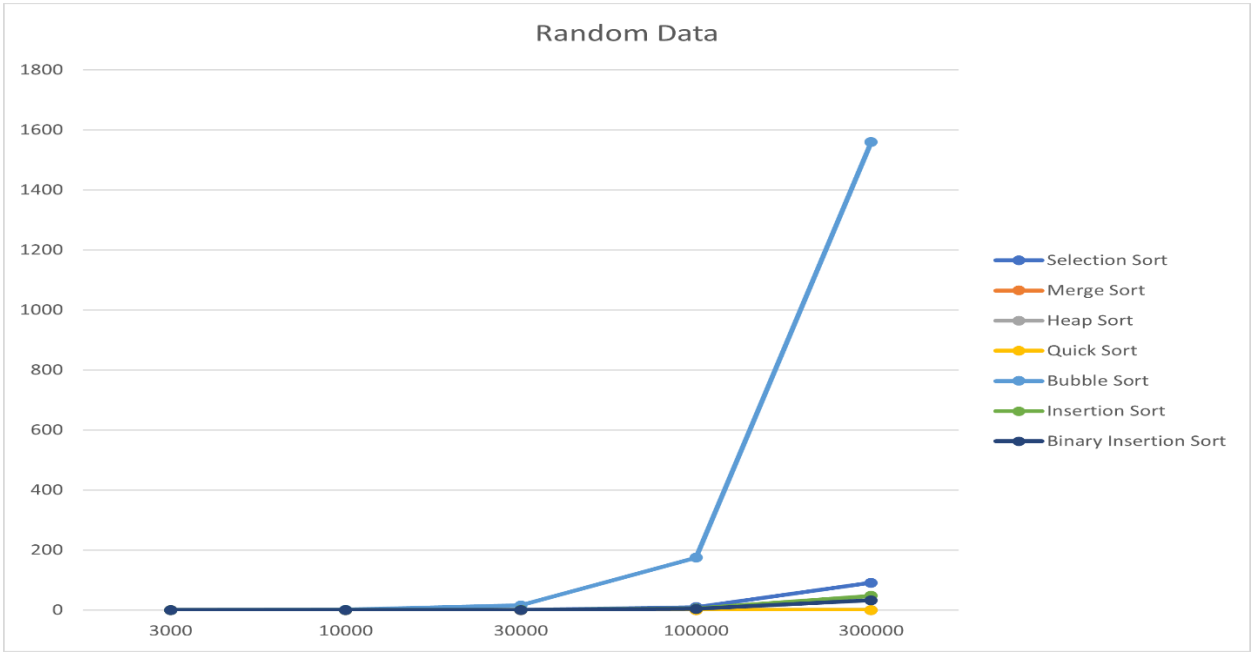
		3000	10000	30000	100000	300000
Random Data	Selection Sort	0.01	0.103	0.893	10.067	90.827
	Merge Sort	0.001	0.003	0.01	0.041	0.13
	Heap Sort	0.002	0.01	0.037	0.146	0.462
	Quick Sort	0.001	0.003	0.011	0.04	0.136
	Bubble Sort	0.156	1.718	15.488	174.751	1559.64
	Insertion Sort	0.005	0.053	0.472	5.543	47.768
	Binary Insertion Sort	0.004	0.038	0.32	3.669	32.631
Sorted Data	Selection Sort	0.09	0.106	0.899	9.849	95.479
	Merge Sort	0.001	0.002	0.006	0.022	0.076
	Heap Sort	0.004	0.011	0.038	0.144	0.481
	Quick Sort	0	0.001	0.002	0.009	0.022
	Bubble Sort	0.01	0.117	1.061	11.658	109.705
	Insertion Sort	0	0	0	0	0.001
	Binary Insertion Sort	0.001	0.002	0.008	0.029	0.099
Reverse Data	Selection Sort	0.008	0.091	0.8	8.828	83.617
	Merge Sort	0.001	0.002	0.006	0.024	0.076
	Heap Sort	0.003	0.01	0.034	0.128	0.428
	Quick Sort	0	0.001	0.003	0.011	0.031
	Bubble Sort	0.28	3.102	27.894	311.153	2817.113
	Insertion Sort	0.01	0.106	0.994	10.794	96.307
	Binary Insertion Sort	0.007	0.071	0.647	7.139	63.491
Nearly Sorted Data	Selection Sort	0.01	0.106	0.909	10.248	87.218
	Merge Sort	0	0.002	0.007	0.023	0.072
	Heap Sort	0.003	0.011	0.038	0.142	0.451
	Quick Sort	0	0.001	0.002	0.008	0.021
	Bubble Sort	0.01	0.12	1.083	12.022	102.118
	Insertion Sort	0	0	0.001	0.001	0.002
	Binary Insertion Sort	0.001	0.003	0.008	0.03	0.097

Dưới đây là kết quả thực nghiệm của 4 kiểu dữ liệu đầu vào với từng kích thước mảng khác nhau. Ta vẽ mỗi trường hợp biểu đồ thể hiện thời gian thực thi của các thuật toán sắp xếp, trong đó trục hoành biểu hiện cho số lượng phần tử đầu vào và trục tung là thời gian thực thi (đơn vị là giây).

Trường hợp dữ liệu chọn ngẫu nhiên (random data):

```
Phan tu mang la: 3000
********   Truong hop du lieu chon ngau nhien  ********

Thoi gian chay cua selection sort: 0.01s
Thoi gian chay cua merge sort: 0.001s
Thoi gian chay cua heap sort: 0.002s
Thoi gian chay cua quick sort: 0.001s
Thoi gian chay cua bubble sort: 0.156s
Thoi gian chay cua insertion sort: 0.005s
Thoi gian chay cua binary insertion sort: 0.004s

Phan tu mang la: 10000
********   Truong hop du lieu chon ngau nhien  ********

Thoi gian chay cua selection sort: 0.103s
Thoi gian chay cua merge sort: 0.003s
Thoi gian chay cua heap sort: 0.01s
Thoi gian chay cua quick sort: 0.003s
Thoi gian chay cua bubble sort: 1.718s
Thoi gian chay cua insertion sort: 0.053s
Thoi gian chay cua binary insertion sort: 0.038s

Phan tu mang la: 30000
********   Truong hop du lieu chon ngau nhien  ********

Thoi gian chay cua selection sort: 0.893s
Thoi gian chay cua merge sort: 0.01s
Thoi gian chay cua heap sort: 0.037s
Thoi gian chay cua quick sort: 0.011s
Thoi gian chay cua bubble sort: 15.488s
Thoi gian chay cua insertion sort: 0.472s
Thoi gian chay cua binary insertion sort: 0.32s

Phan tu mang la: 100000
********   Truong hop du lieu chon ngau nhien  ********

Thoi gian chay cua selection sort: 10.067s
Thoi gian chay cua merge sort: 0.041s
Thoi gian chay cua heap sort: 0.146s
Thoi gian chay cua quick sort: 0.04s
Thoi gian chay cua bubble sort: 174.751s
Thoi gian chay cua insertion sort: 5.543s
Thoi gian chay cua binary insertion sort: 3.669s

Phan tu mang la: 300000
********   Truong hop du lieu chon ngau nhien  ********

Thoi gian chay cua selection sort: 90.827s
Thoi gian chay cua merge sort: 0.13s
Thoi gian chay cua heap sort: 0.462s
Thoi gian chay cua quick sort: 0.136s
Thoi gian chay cua bubble sort: 1559.64s
Thoi gian chay cua insertion sort: 47.768s
Thoi gian chay cua binary insertion sort: 32.631s
```

Hình ảnh 1: Thời gian thực thi của các thuật toán sắp xếp dữ liệu chọn ngẫu nhiên



Hình ảnh 2: Biểu đồ thể hiện thời gian thực thi các thuật toán với bộ dữ liệu ngẫu nhiên (debug) (đv: giây)

Nhận xét:

- Với kích thước đầu vào 3000, 10000, 30000 thời gian chạy của các thuật toán gần như bằng nhau, sự chênh lệch thời gian nhiều nhất cũng chỉ khoảng 0.15 giây, nên không có sự khác biệt quá lớn giữa các thuật toán sắp xếp.
- Với kích thước từ 30000 trở lên thì bắt đầu có sự chênh lệch rõ ràng. Bubble Sort và Selection chạy chậm nhất, Heap Sort, Merge Sort, Quick Sort chạy khá là nhanh.
- Các thuật toán chạy nhanh nhất: Merge Sort, Quick Sort, Heap Sort, Binary-Insertion Sort với độ phức tạp *O(nlogn)*. Các thuật toán này có độ phức tạp logarit và tuyến tính nên chạy rất nhanh.
- Các thuật toán chạy chậm nhất: Bubble Sort, Selection Sort, Insertion Sort với độ phức tạp về thời gian là $O(n^2)$. Các thuật toán này luôn chạy đủ bình phương độ dài mảng các thao tác nên khi khi số lượng phần tử càng nhiều sẽ làm cho thời gian chạy càng cao.

- *Xếp hạng thời gian thực thi các thuật toán sắp xếp từ chậm nhất đến nhanh nhất:*

  Bubble Sort < Selection Sort < Insertion Sort < Binary-Insertion Sort < Heap Sort < Merge Sort < Quick Sort.

Trường hợp dữ liệu có thứ tự (sorted data):

```
Phan tu mang la: 3000
********   Truong hop du lieu tang dan  ********

Thoi gian chay cua selection sort: 0.009s
Thoi gian chay cua merge sort: 0.001s
Thoi gian chay cua heap sort: 0.004s
Thoi gian chay cua quick sort: 0s
Thoi gian chay cua bubble sort: 0.01s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.001s
Phan tu mang la: 30000
********   Truong hop du lieu tang dan  ********

Thoi gian chay cua selection sort: 0.899s
Thoi gian chay cua merge sort: 0.006s
Thoi gian chay cua heap sort: 0.038s
Thoi gian chay cua quick sort: 0.002s
Thoi gian chay cua bubble sort: 1.061s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.008s
```

```
Phan tu mang la: 10000
********   Truong hop du lieu tang dan  ********

Thoi gian chay cua selection sort: 0.106s
Thoi gian chay cua merge sort: 0.002s
Thoi gian chay cua heap sort: 0.011s
Thoi gian chay cua quick sort: 0.001s
Thoi gian chay cua bubble sort: 0.117s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.002s
Phan tu mang la: 100000
********   Truong hop du lieu tang dan  ********

Thoi gian chay cua selection sort: 9.849s
Thoi gian chay cua merge sort: 0.022s
Thoi gian chay cua heap sort: 0.144s
Thoi gian chay cua quick sort: 0.009s
Thoi gian chay cua bubble sort: 11.658s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.029s
```

```
Phan tu mang la: 300000
********   Truong hop du lieu tang dan  ********

Thoi gian chay cua selection sort: 95.479s
Thoi gian chay cua merge sort: 0.076s
Thoi gian chay cua heap sort: 0.481s
Thoi gian chay cua quick sort: 0.022s
Thoi gian chay cua bubble sort: 109.705s
Thoi gian chay cua insertion sort: 0.001s
Thoi gian chay cua binary insertion sort: 0.099s
```

Hình ảnh 3: Thời gian thực thi của các thuật toán sắp xếp dữ liệu có thứ tự

Hình ảnh 4: Biểu đồ thể hiện thời gian thực thi các thuật toán với bộ dữ liệu có thứ tự (debug) (đv: giây)

Nhận xét:

- Tương tự như trên, với dữ liệu 3000, 10000, 30000 thì các thuật toán đều có thời gian chạy rất nhanh với độ chênh lệch rất ít.
- Với kích thước lớn hơn 30000 thì ta cũng thấy được sự chênh lệch của các thuật toán. Bubble Sort và Selection Sort có thời gian chạy khác lâu so với các hàm còn lại.
- Các thuật toán chạy nhanh nhất: Insertion Sort. Vì mảng dữ liệu đầu vào đã sắp xếp, với dộ phức tạp *O(n)* ta thấy thời gian chạy thực thi của nó gần như là 0 giây, xem như Selection Sort chỉ duyệt mảng có một lần nên thời gian nhanh nhất.
- Các thuật toán chạy nhanh: Quick Sort, Merge Sort, Heap Sort. Các thuật toán này có thời gian thực hiện cũng khá nhanh với độ phức tạp *O(nlogn).*
- Các thuật toán chạy chậm nhất: Bubble Sort, Selection Sort . Các thuật toán này không quan tâm đến thứ tự mảng và đều phải chạy đủ bình phương độ dài mảng nên độ phức tạp thời gian trong mọi trường hợp đều là $O(n^2)$.

- *Xếp hạng thời gian thực thi các thuật toán sắp xếp từ chậm nhất đến nhanh nhất:*

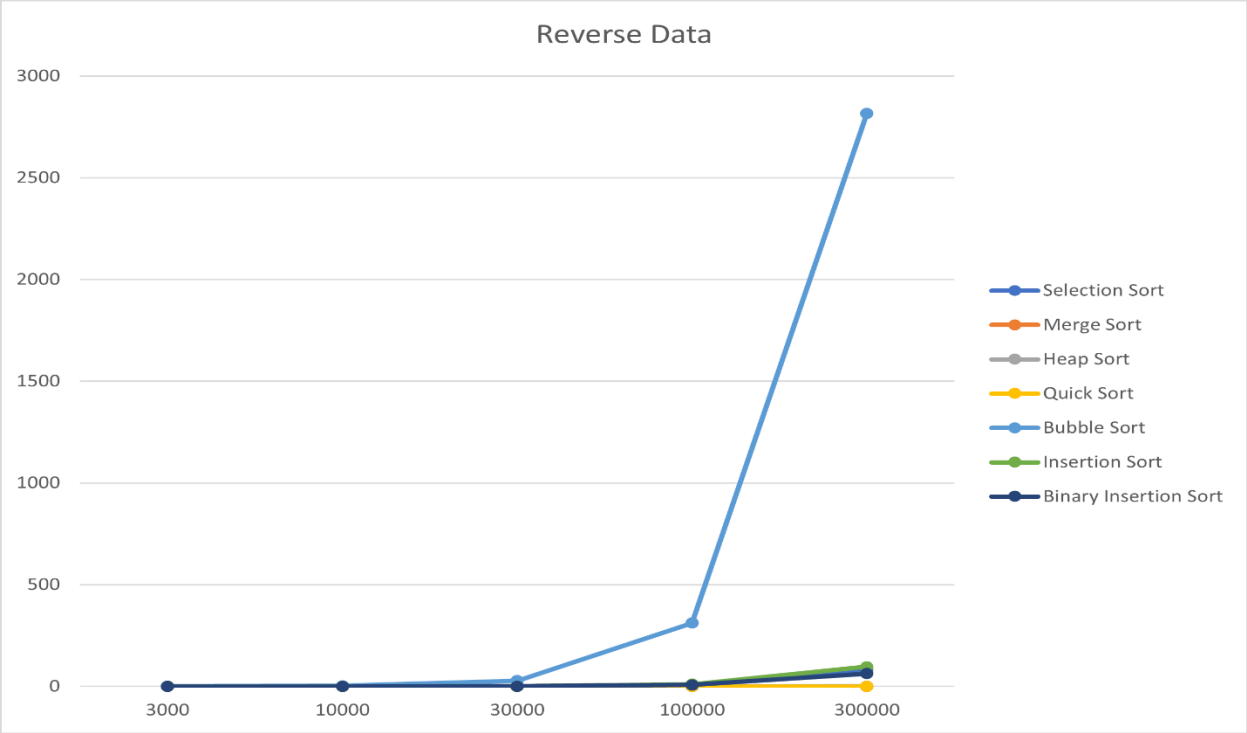
  Bubble Sort < Selection Sort < Heap Sort < Binary-Insertion Sort < Merge Sort < Quick Sort < Insertion Sort

Trường hợp dữ liệu có thứ tự ngược (reverse data):

```
Phan tu mang la: 3000                                   Phan tu mang la: 10000
********   Truong hop du lieu co thu tu nguoc  **********   Truong hop du lieu co thu tu nguoc  ****

Thoi gian chay cua selection sort: 0.008s               Thoi gian chay cua selection sort: 0.091s
Thoi gian chay cua merge sort: 0.001s                   Thoi gian chay cua merge sort: 0.002s
Thoi gian chay cua heap sort: 0.003s                    Thoi gian chay cua heap sort: 0.01s
Thoi gian chay cua quick sort: 0s                       Thoi gian chay cua quick sort: 0.001s
Thoi gian chay cua bubble sort: 0.28s                   Thoi gian chay cua bubble sort: 3.102s
Thoi gian chay cua insertion sort: 0.01s                Thoi gian chay cua insertion sort: 0.106s
Thoi gian chay cua binary insertion sort: 0.007s        Thoi gian chay cua binary insertion sort: 0.071s
Phan tu mang la: 30000                                  Phan tu mang la: 100000
********   Truong hop du lieu co thu tu nguoc  **********   Truong hop du lieu co thu tu nguoc  ****

Thoi gian chay cua selection sort: 0.8s                 Thoi gian chay cua selection sort: 8.828s
Thoi gian chay cua merge sort: 0.006s                   Thoi gian chay cua merge sort: 0.024s
Thoi gian chay cua heap sort: 0.034s                    Thoi gian chay cua heap sort: 0.128s
Thoi gian chay cua quick sort: 0.003s                   Thoi gian chay cua quick sort: 0.011s
Thoi gian chay cua bubble sort: 27.894s                 Thoi gian chay cua bubble sort: 311.153s
Thoi gian chay cua insertion sort: 0.994s               Thoi gian chay cua insertion sort: 10.794s
Thoi gian chay cua binary insertion sort: 0.647s        Thoi gian chay cua binary insertion sort: 7.139s
```

```
Phan tu mang la: 300000
********   Truong hop du lieu co thu tu nguoc  ********

Thoi gian chay cua selection sort: 83.617s
Thoi gian chay cua merge sort: 0.076s
Thoi gian chay cua heap sort: 0.428s
Thoi gian chay cua quick sort: 0.031s
Thoi gian chay cua bubble sort: 2817.13s
Thoi gian chay cua insertion sort: 96.307s
Thoi gian chay cua binary insertion sort: 63.491s
```

Hình ảnh 5: Thời gian thực thi của các thuật toán sắp xếp dữ liệu có thứ tự ngược



Hình ảnh 6: Biểu đồ thể hiện thời gian thực thi các thuật toán với bộ dữ liệu có thứ tự ngược (debug) (đv: giây)

Nhận xét:

- Tương tự như trên, với dữ liệu 3000, 10000, 30000 thì các thuật toán đều có thời gian chạy rất nhanh với độ chênh lệch rất ít. Riêng Bubble Sort có độ chênh lệch khá cao lên đến 27 giây.
- Các thuật toán chạy nhanh nhất: Quick Sort, Merge Sort, Heap Sort . Các thuật toán này chạy thời gian ngắn với độ phức tạp về thời gian là : *O(nlogn).*
- Các thuật toán chậm nhất: Bubble Sort, Insertion Sort, Selection Sort, Binary-Insertion Sort. Trong trường hợp này, các thuật toán có độ phức tạp là $O(n^2)$.

- *Xếp hạng thời gian thực thi các thuật toán sắp xếp từ chậm nhất đến nhanh nhất:*

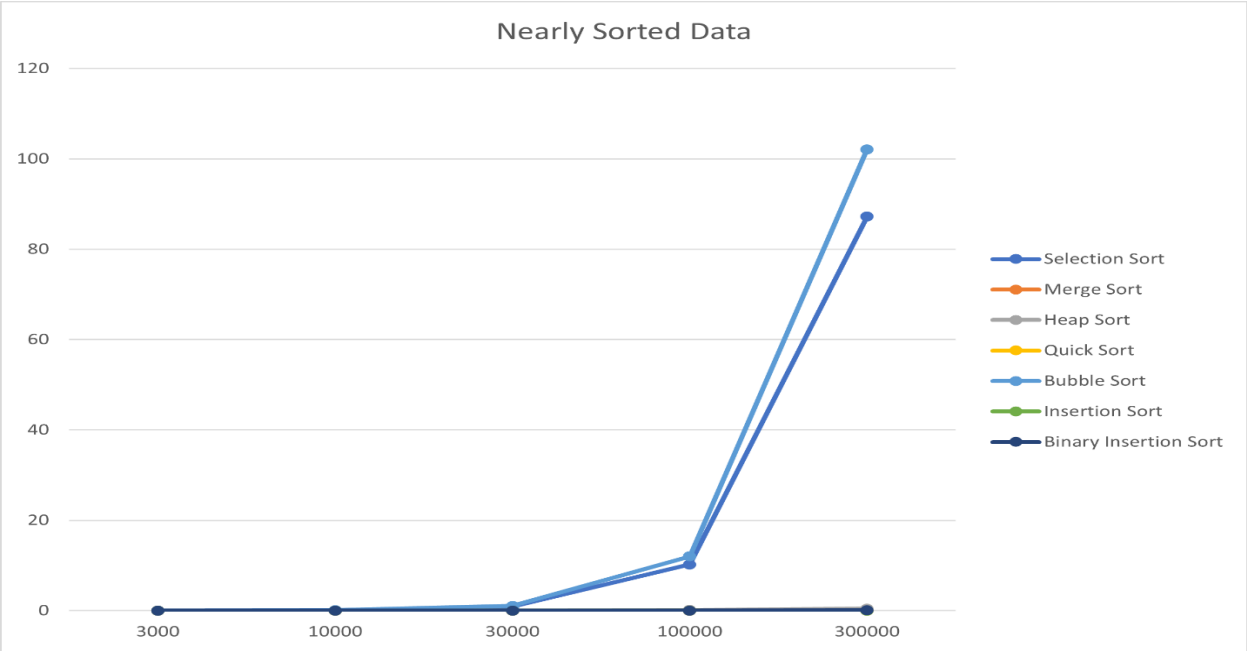
  Bubble Sort < Insertion Sort < Selection Sort < Binary-Insertion Sort < Heap Sort < Merge Sort < Quick Sort

Trường hợp dữ liệu gần như có thứ tự (nearly sorted data):

```
Phan tu mang la: 3000
********   Truong hop du lieu gan nhu co thu tu  ********

Thoi gian chay cua selection sort: 0.01s
Thoi gian chay cua merge sort: 0s
Thoi gian chay cua heap sort: 0.003s
Thoi gian chay cua quick sort: 0s
Thoi gian chay cua bubble sort: 0.01s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.001s
Phan tu mang la: 30000
********   Truong hop du lieu gan nhu co thu tu  ********

Thoi gian chay cua selection sort: 0.909s
Thoi gian chay cua merge sort: 0.007s
Thoi gian chay cua heap sort: 0.038s
Thoi gian chay cua quick sort: 0.002s
Thoi gian chay cua bubble sort: 1.083s
Thoi gian chay cua insertion sort: 0.001s
Thoi gian chay cua binary insertion sort: 0.008s
Phan tu mang la: 300000
********   Truong hop du lieu gan nhu co thu tu  ********

Thoi gian chay cua selection sort: 87.218s
Thoi gian chay cua merge sort: 0.072s
Thoi gian chay cua heap sort: 0.451s
Thoi gian chay cua quick sort: 0.021s
Thoi gian chay cua bubble sort: 102.118s
Thoi gian chay cua insertion sort: 0.002s
Thoi gian chay cua binary insertion sort: 0.097s
```

```
Phan tu mang la: 10000
********   Truong hop du lieu gan nhu co thu tu  ***

Thoi gian chay cua selection sort: 0.106s
Thoi gian chay cua merge sort: 0.002s
Thoi gian chay cua heap sort: 0.011s
Thoi gian chay cua quick sort: 0.001s
Thoi gian chay cua bubble sort: 0.12s
Thoi gian chay cua insertion sort: 0s
Thoi gian chay cua binary insertion sort: 0.003s
Phan tu mang la: 100000
********   Truong hop du lieu gan nhu co thu tu  ***

Thoi gian chay cua selection sort: 10.248s
Thoi gian chay cua merge sort: 0.023s
Thoi gian chay cua heap sort: 0.142s
Thoi gian chay cua quick sort: 0.008s
Thoi gian chay cua bubble sort: 12.022s
Thoi gian chay cua insertion sort: 0.001s
Thoi gian chay cua binary insertion sort: 0.03s
```

Hình ảnh 7: Thời gian thực thi của các thuật toán sắp xếp dữ liệu gần như có thứ tự



Hình ảnh 8: Biểu đồ thể hiện thời gian thực thi các thuật toán với bộ dữ liệu gần như có thứ tự (debug) (đv: giây)

Nhận xét:

- Tương tự như trên, với dữ liệu 3000, 10000, 30000 thì các thuật toán đều có thời gian chạy rất nhanh với độ chênh lệch rất ít.
- Các thuật toán chạy nhanh nhất: Insertion Sort. Trường hợp này khá giống như trường hợp của dữ liệu mảng đã sắp xếp tăng dần nên thuật toán này chạy nhanh nhất khi nó gần như duyệt mảng chỉ có 1 lần với độ phức tạp thời gian O(n).
- Các thuật toán chạy nhanh: Quick Sort, Merge Sort, Heap Sort với độ phức tạp *O(nlogn)* và , Binary-Insertion Sort với độ phức tạp *O(n).*
- Các thuật toán chạy chậm: Bubble Sort, Selection Sort. Hiển nhiên sẽ chậm vì nó không quan tâm đến thứ tự mà phải chạy đủ bình phương độ dài mảng nên có độ phức tạp là $O(n^2)$.

- *Xếp hạng thời gian thực thi các thuật toán sắp xếp từ chậm nhất đến nhanh nhất:*

  Bubble Sort < Selection Sort < Heap Sort < Binary-Insertion Sort < Merge Sort < Quick Sort < Insertion Sort

PHẦN 3: NHẬN XÉT TỔNG THỂ

- Các thuật toán có thời gian chạy bằng nhau và ổn định nếu kích thước mảng nhỏ, sự chênh lệch thời gian chay giữa các thuật toán là không quá lớn.
- Nếu kích thước mảng lớn sẽ cho ta cái nhìn rõ rệt hơn về thời gian thực thi của các thuật toán, cụ thể:
  + Thuật toán chạy nhanh nhất: Quick Sort, Merge Sort, Heap Sort, Binary-Insertion Sort, Insertion Sort vì nó độ phức tạp logarit hoặc tuyến tính nên thời gian thực hiện thường dưới 1 giây.
  + Thuật toán chạy chậm nhất: Bubble Sort và Selection Sort vì luôn xuất ra thời gian sắp xếp lâu nhất so với các thuật toán khác vì trong mọi trường hợp nó điều có độ phức tap là $O(n^2)$.
- Các thuật toán không ổn định: Selection, Quick Sort (phụ thuộc vị trí chọn mốc), Heap Sort.
- Các thuật toán ổn định: Bubble Sort, Insertion Sort, Merge Sort, Binary – Insertion Sort.
- Qua đó cho ta thấy rằng, khi kích thước mảng lớn ta không nên dùng Bubble Sort và Selection Sort, nên dùng thuật toán Quick Sort vì nó rất nhanh trong mọi trường hợp nhưng cũng phụ thộc vào vào chọn mốc pivot cho nó (trường hợp tốt nhất là chon mốc ở trung vị), sử dung Merge Sort cũng là một lựa chọn hợp lý để thay thế Quick Sort nhưng ta sẽ ưu tiên Quick Sort hơn vì độ phức tạp thời gian của Quick Sort là *O(nlogn)* và của Merge Sort là *O(n).*

PHẦN 4: ĐÁNH GIÁ TIẾN ĐỘ

- Cài đặt và chạy thuật toán thành công.
- Đo được thời gian chạy của các thuật toán trong mỗi trường hợp
- Vẽ biểu đồ biểu thị thời gian chạy các thuật toán và so sánh các trường hợp.
- Hiểu được ý nghĩa, thuật toán, đánh giá được về độ phức tạp và ứng dụng thực tế của các thuật toán sắp xếp.

Đánh giá mức độ: hoàn thành khoảng 90% yêu cầu đề bài.

PHẦN 5: THÔNG TIN VỀ CẤU HÌNH MÁY VÀ CÔNG CỤ HỖ TRỢ

- Cấu hình máy:
  Processor: Intel ® Core ™ i7-8750H CPU @ 2.20 GHz (12 CPUs)
  Ram: 8GB
  System Type: 64-bit Operating System
  GPU: GeForce GTX 1050 Ti 4Gb

- Công cụ hỗ trợ:
  IDE: Visual Studio 2019 (C++).
  Microsoft Excel 365.
  Microsoft Word 365.

PHẦN 6: TÀI LIỆU THAM KHẢO

https://www.geeksforgeeks.org/sorting-algorithms/
https://en.wikipedia.org/wiki/Sorting_algorithm
https://www.youtube.com/watch?v=-OVB5pOZJug
https://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm
https://www.programiz.com/dsa/selection-sort
https://visualgo.net/en
Algorithms in a Nutshell by George T. Heineman, Gary Police, and Stanley Selkow.
Introduction to Algorithms by Thomas H. Cormen, Charles E.Leiserson, Ronald L. Rivest, Clifford Stein.