KỊCH BẢN

Chương trình được viết trên Dev-C++. Tạo chương trình **CreateData.c** để tạo dãy số gồm **x** số ngẫu nhiên rồi lưu vào file **x\_number.txt**. Chương trình đánh giá sắp xếp sẽ đọc dữ liệu từ file để tiến hành đánh giá các thuật toán sắp xếp sau: chèn, chọn, nhanh.

SẮP XẾP CHÈN

Thuật toán sắp xếp chèn thực hiện chèn từng phần tử vào vị trí đúng trong mảng. đảm bảo các phần tử vẫn được sắp xếp theo thứ tự. Duyệt từ phần tử thứ hai đến phần tử cuối cùng. Gán **k** là giá trị của phần tử thứ ‘i’ .Khi duyệt đến phần tử thứ ‘i’ nó sẽ so sánh với phần từ thứ ‘i-1’. Nếu nó bé hơn, thực hiện đổi chỗ hai phần tử đó. Cứ như thế thực hiện đến khi tất cả các phần tử được sắp xếp.

void insertionSort(int arr[], int n)

{

    int i, key, j;

    for (i = 1; i < n; i++)

    {

        key = arr[i];

        j = i - 1;

        while (j >= 0 && arr[j] > key)

        {

           arr[j + 1] = arr[j];

           j = j - 1;

        }

       arr[j + 1] = key;

   }

}

Số phép so sánh được thực hiện: n\*(n-1)/2 (được tính bởi (n-1)+(n-2)+….+1= n\*(n-1)/2

Từ đó độ phức tạp của thuật toán là O(n^2).

Độ phức tạp thời gian: Trường hợp xấu nhất: O(n^2).

Trường hợp tốt nhất: O(n).

Trường hợp trung bình nhất: O(n^2).

Độ phức tạp không gian: O(1) vì chỉ dùng biến **key.**

SẮP XẾP CHỌN

Thuật toán sắp xếp chọn thực hiện chọn phần tử nhỏ nhất từ mảng chưa được sắp xếp trong mỗi lượt lăp và đặt phần tử đó ở đầu mảng chưa sắp xếp. Sau đó tìm phần tử nhỏ tiếp theo trong mảng và đổi chỗ với phần tử thứ hai trong mảng. Cứ như thế thực hiện đến khi tất cả các phần tử được sắp xếp.

void selectionSort(int arr[], int n) {

    int i, j, min\_pos;

    for (i = 0; i < n-1; i++) {

        min\_pos = i;

        for (j = i+1; j < n; j++) {

            if (arr[j] < arr[min\_pos]) {

                min\_pos = j;

            }

        }

        int temp = arr[min\_pos];

        arr[min\_pos] = arr[i];

        arr[i] = temp;

    }

}

Số phép so sánh được thực hiện: n\*(n-1)/2 (được tính bởi (n-1)+(n-2)+….+1= n\*(n-1)/2

Từ đó độ phức tạp của thuật toán là O(n^2).

Độ phức tạp thời gian: Trường hợp xấu nhất: O(n^2).

Trường hợp tốt nhất: O(n^2).

Trường hợp trung bình nhất: O(n^2).

Độ phức tạp không gian: O(1) vì chỉ dùng biến **temp.**

SẮP XẾP NHANH

Thuật toán chọn một phần tử trong mảng và đánh dấu, gọi là **pivot**. Sau đó chỉa mảng đó thành hai mảng con bằng cách so sánh với **pivot** đã chọn. Một mảng sẽ bao gồm các phần tử nhỏ hơn hoặc bằng pivot và mảng còn lại luôn lớn hơn hoặc bằng **pivot**. Sau đó, quá trình này được lặp lại đủ số lần cho đến khi các mảng nhỏ có thể được sắp xếp một cách dễ dàng để tạo ra một tập dữ liệu được sắp xếp đầy đủ.

void swap(int\* a, int\* b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

int quickSort(int arr[], int low, int high)

{

    int pivot = arr[high];

    int i = (low - 1);

    for(int j = low; j <= high - 1; j++)

    {

        if(arr[j] < pivot)

        {

            i++;

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return (i + 1);

}

Số phép so sánh được thực hiện: n\*(n-1)/2 (được tính bởi (n-1)+(n-2)+….+1= n\*(n-1)/2

Từ đó độ phức tạp của thuật toán là O(n^2).

Độ phức tạp thời gian: Trường hợp xấu nhất: O(n^2).

Trường hợp tốt nhất: O(n log n).

Trường hợp trung bình nhất: O(n log n).

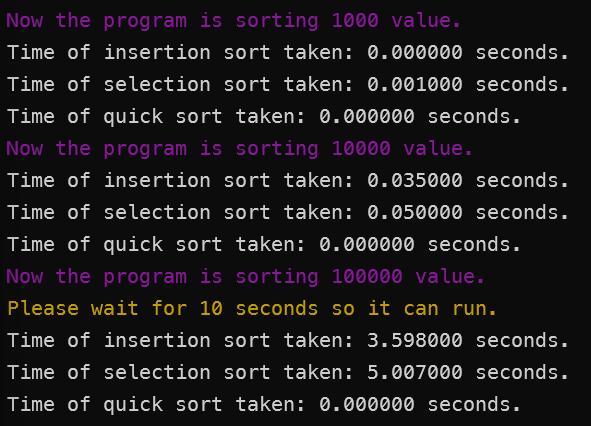
Độ phức tạp không gian: Trường hợp trung bình nhất: O(log n)

Trường hợp xấu nhất: O(n)

Khi chạy thực nghiệm được kết quả sau:

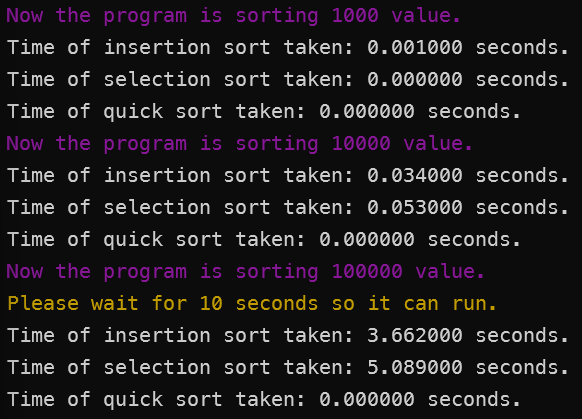
Lần 1 (Ngày 15/4/2023)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1000 phần tử | 10000 phần tử | 100000 phần tử |
| Sắp xếp chọn | 0.000000 s | 0.035000 s | 3.598000 s |
| Sắp xếp chèn | 0.000000 s | 0.050000 s | 5.007000 s |
| Sắp xếp nhanh | 0.000000 s | 0.000000 s | 0.000000 s |

****

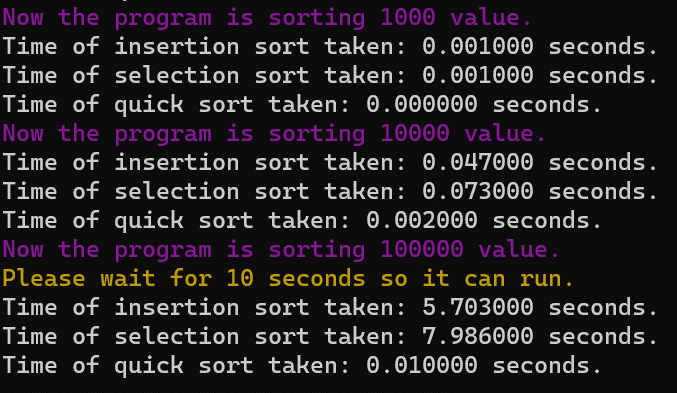
Lần 2 (Ngày 15/4/2023)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1000 phần tử | 10000 phần tử | 100000 phần tử |
| Sắp xếp chọn | 0.000000 s | 0.034000 s | 3.662000 s |
| Sắp xếp chèn | 0.000000 s | 0.053000 s | 5.089000 s |
| Sắp xếp nhanh | 0.000000 s | 0.000000 s | 0.000000 s |

****

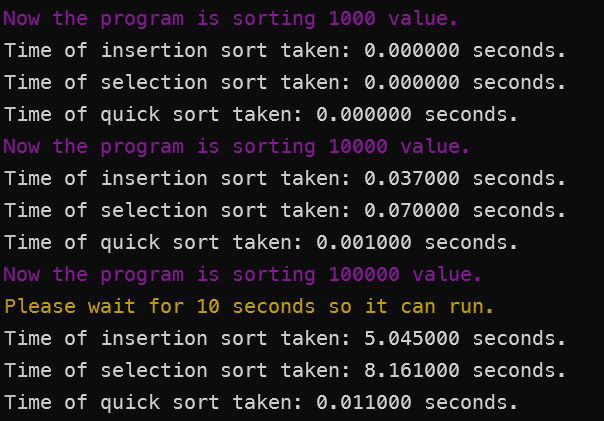
Lần 3 (Ngày 16/4/2023)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1000 phần tử | 10000 phần tử | 100000 phần tử |
| Sắp xếp chọn | 0.001000 s | 0.047000 s | 5.703000 s |
| Sắp xếp chèn | 0.001000 s | 0.073000 s | 7.986000 s |
| Sắp xếp nhanh | 0.000000 s | 0.002000 s | 0.010000 s |

****

Lần 4 (Ngày 16/4/2023)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1000 phần tử | 10000 phần tử | 100000 phần tử |
| Sắp xếp chọn | 0.000000 s | 0.037000 s | 5.045000 s |
| Sắp xếp chèn | 0.000000 s | 0.070000 s | 8.161000 s |
| Sắp xếp nhanh | 0.000000 s | 0.001000 s | 0.011000 s |

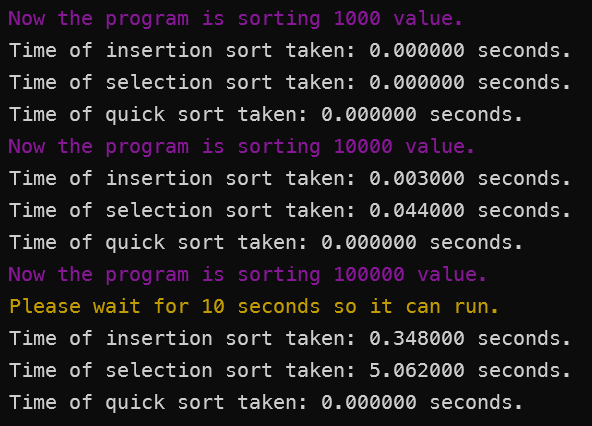
****

Dựa vào kết quả thực nghiệm, ta nhận thấy:

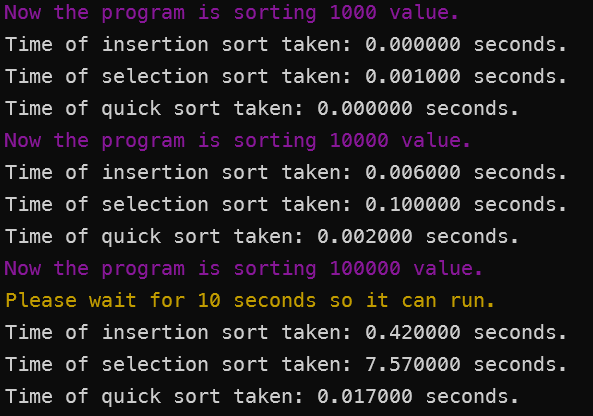
**Với lượng dữ liệu nhỏ ( ~1000 phần tử)** thì thời gian thực hiện của các thuật toán sắp xếp tương đối bằng nhau.

Nếu danh sách đã được sắp xếp gần như đúng thì sắp xếp chèn và chọn có thể hiệu quả hơn sắp xếp nhanh. VD: với danh sách đã được sắp xếp trước 95% ta được:

Ngày 15/4/2023



Ngày 16/4/2023



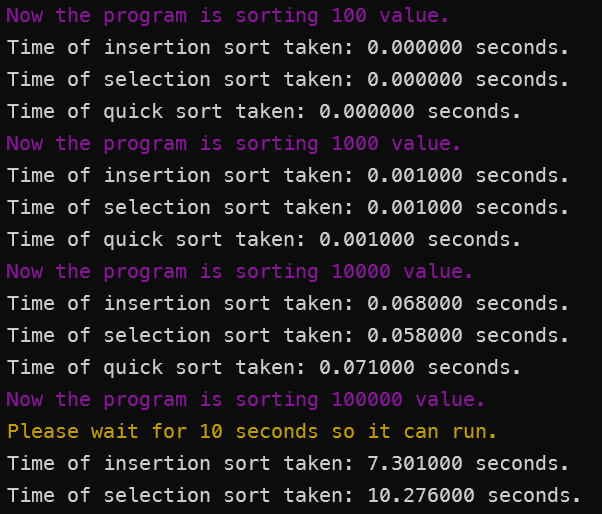
**Lý do ngày 15/4 chạy nhanh hơn nhiều so với ngày 16/4**: Hiện tại mình đang chạy chương trình đồng thời hai máy ảo nên thời gian thực hiện tăng. => điều này giúp chứng minh thuật toán sắp xếp nhanh thiếu sự ổn định

**Với lượng dữ liệu lớn ( >10000 phần tử)** thì thời gian thực hiện của các thuật toán sắp xếp có sự khác nhau rõ rệt.

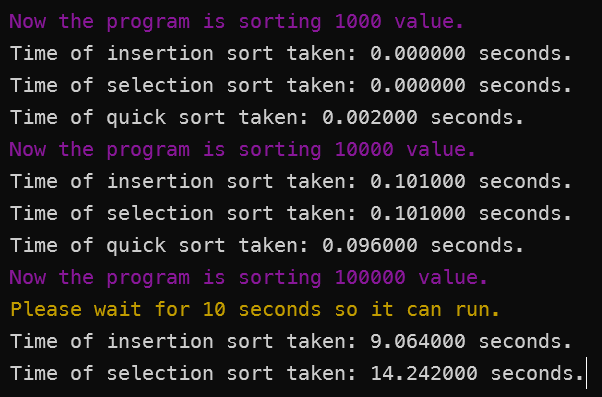
Thời gian của sắp xếp chọn là lâu nhất, nhanh hơn có sắp xếp chọn và nhanh nhất là sắp xếp nhanh.

Đối với trường hợp xấu nhất ( tất cả sắp xếp từ lớn đến bé):

Ngày 15/4/2023



Ngày 16/4/2023

****

Với dữ liệu nhỏ thì thời gian thực hiện của các thuật toán tương đối bằng nhâu nhưng khi dữ liệu lớn, thuật toán sắp xếp nhanh thể hiện sự không ổn định của mình và thời gian thực hiện tại 10000 giá trị lâu hơn hai thuật toán trên. Vì việc lựa chọn pivot không phù hợp và gọi hàm đệ quy quá nhiều lần khiến cho nó thực hiện chậm đi.

Trong ví dụ trên, do gọi hàm đệ quy quá nhiều khiến cho stack bị đầy. Từ đó chương trình không thực thi tiếp được và kết thúc.