**CHƯƠNG 3 – TRẢ LỜI CÂU HỎI**

*Họ và tên: Nguyễn Công Thành*

*MSSV:1654050102*

*Nhóm: 8*

**Câu 1: Trong các phương pháp xếp thứ tự đã học, phương pháp nào tối ưu nhất, và kém tối ưu nhất? Tại sao?**

- Xét độ phức tạp của 6 phương pháp sắp xếp:

|  |  |
| --- | --- |
| **THUẬT TOÁN SẮP XẾP** | **ĐỘ PHỨC TẠP** |
| Selection Sort | O(n2) |
| Insertion Sort | O(n2) |
| Interchange Sort | O(n2) |
| Bubble Sort | O(n2) |
| Quick Sort | O(n log n) |
| Heap Sort | O(n log n) |

- Xét về thời gian sắp xếp của dãy dữ liệu Random (ngẫu nhiên):

Quick Sort > Heap Sort > Insertion Sort > Selection Sort > Interchange Sort> Bubble Sort

=> Quick Sort là thuật toán sắp xếp nhanh nhất trong nhiều trường hợp.

=> Việc lựa chọn thuật toán cần dựa vào nhiều yếu tố: dữ liệu đầu vào số lượng bao nhiêu?, có được sắp xếp sẵn hay không?, dung lượng bộ nhớ và tốc độ xử lý như thế nào? Vì nếu Quick Sort là nhanh nhất thì những thuật toán khác sẽ không được ra đời.

- Ví dụ:

* Nếu dữ liệu đã được sắp xếp sẵn thì nên chọn Insertion Sort.
* Nếu không rơi vào trường hợp (case) xấu nhất thì có thử sử dụng Heap Sort thay cho Quick Sort.
* Nếu dữ liệu có ít phần tử (10 – 20 phần tử) thì Selection Sort sẽ nhanh hơn Quick Sort.

**Câu 2: Trong phương pháp tìm kiếm (Tìm kiếm tuần tự và tìm kiếm nhị phân) trường hợp nào thì cả 2 phương pháp đều như nhau? Giải thích tại sao?**

- Cả 2 phương pháp đều khác nhau về thuật giải, cách xử lý dữ liệu sẽ khác nhau. Tuy nhiên, trong một số trường hợp đặc biệt thì cả 2 phương pháp sẽ tìm kiếm giá trị tại một vị trí và có cùng số phép so sánh.

- Trường hợp: dữ liệu đầu vào đã được xếp thứ tự

Điều kiện: x cần tìm nằm đầu dãy (bên trái), khi phân hoạch thì x cần tìm nhỏ hơn giá trị trung bình được phân hoạch

1. n = 1

0

|  |
| --- |
| 1 |

1. n = 2, x cần tìm tại vị trí i = 0 ( x = 1)

0 1

|  |  |
| --- | --- |
| 1 | 2 |

1. n = 8, x cần tìm tại vị trí i = 1 (**x = 2**)

0 1 2 3 4 5 6 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

* Tìm kiếm tuần tự: số lần so sánh là 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phần tử | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Vị trí i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Lần so sánh 1 | x ≠ 1 |  |  |  |  |  |  |  |
| Lần so sánh 2 |  | **x = 2** |  |  |  |  |  |  |

* Tìm kiếm nhị phân: số lần so sánh là 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phần tử | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Vị trí i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Lần so sánh 1 |  |  |  | x ≠ 4 |  |  |  |  |
| Lần so sánh 2 |  | **x = 2** |  |  |  |  |  |  |

……..

- Trường hợp: xấu nhất là giá trị x cần tìm không có trong dãy dữ liệu hoặc giá trị x có vị trí là cuối dãy.

=> Cả 2 phương pháp sẽ không như nhau.

**Câu 3: Ngoài các phương pháp xếp thứ tự đã học, hãy tìm hiểu thêm một phương pháp xếp thứ tự khác, giới thiệu và giải thích.**

- MERGE SORT:

* Thuật toán Merge Sort là thuật toán có độ phức tạp trung bình nhưng đạt hiệu quả trong nhiều trường hợp.
* Thuật toán Merge Sort là loại thuật toán sắp xếp theo phương pháp trộn có độ phức tạp trung bình (O(n.logn)).

- Ý tưởng:

* Dãy cần sắp xếp sẽ có những đoạn dãy con đã có thứ tự tăng dần hoặc giảm dần. Thuật toán làm giảm dần các dãy con, chỉ còn một dãy con duy nhất có chiều tăng giảm đúng với yêu cầu bằng cách chia dãy ban đầu ra 2 dãy phụ theo nguyên tắc luân phiên, cứ một phần tử vào dãy thứ nhất thì phần tử tiếp sau vào dãy thứ hai và ngược lại.
* Trộn từng cặp dãy con trong hai dãy phụ vào dãy ban đầu ta sẽ được dãy mới có các phần tử tương tự dãy ban đầu nhưng có trật tự thay đổi, số đoạn dãy con giảm dần, ít nhất là giảm đi một nửa theo hướng sắp xếp của người viết. Tuần tự tăng dần độ lớn của dãy con và thực hiện tương tự các bước trên cho đến khi độ lớn của dãy con vượt quá số phần tử của dãy ban đầu.

- Ví dụ:

