**TRẢ LỜI CÂU HỎI – CHƯƠNG 3**

**Họ và tên:** Nguyễn Thị Yến Khương

**MSSV:** 1654050052 – **Nhóm: 04**

**Bài làm:**

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

Trong các phương pháp sắp xếp thứ tự đã học như Selection Sort, Insertion Sort, Interchange Sort, Bubble Sort, Quick Sort, Heap Sort, thì Quick Sort được đánh giá là phương pháp sắp xếp thứ tự tối ưu nhất trong 6 phương pháp trên, bên cạnh đó, phương pháp sắp xếp thứ tự kém tối ưu nhất lại là Bubble Sort.

|  |  |  |
| --- | --- | --- |
| THUẬT TOÁN SẮP XẾP | ĐỘ PHỨC TẠP | XẾP THỨ TỰ THỜI GIAN THỰC HIỆN |
| Selection Sort | O(n2) | 4 |
| Insertion Sort | O(n2) | 3 |
| Interchange Sort | O(n2) | 5 |
| Bubble Sort | O(n2) | 6 |
| Quick Sort | O(n logn) | 1 |
| Heap Sort | O(n logn) | 2 |

Nhìn bảng thống kê ở trên, ta có thể dễ dàng nhìn thấy, nếu xét về độ phức tạp, thì Quick Sort và Heap Sort có độ phức tạp tương đương nhau và thấp hơn hơn so với các thuật toán sắp xếp khác (O(n log n) < O(n2)). Thế nhưng, để lựa chọn giữa Quick Sort và Heap Sort, ta nên chọn Quick Sort. Vì Quick Sort thực hiện việc sắp xếp thứ tự nhanh hơn Heap Sort. Chính vì thế, Quick Sort là thuật toán tối ưu nhất với độ phức tạp thấp, mà lại không tốn nhiều thời gian để thực hiện.

Tương tự như trên, thuật toán sắp xếp kém tối ưu nhất sẽ là Bubble Sort. Tuy có độ phức tạp bằng với Selection Sort, Insertion Sort, Interchange Sort, nhưng Bubble Sort lại có thời gian thực hiện lâu nhất (xếp thứ 6). Do đó, đây là thuật toán sắp xếp kém tối ưu nhất vì có độ phức tạp cao và tốn nhiều thời gian để thực hiện.

Thế nhưng trong thực tế, việc lựa chọn thuật toán sắp xếp nào là tối ưu nhất, hay kém tối ưu nhất là điều khó có thể thực hiện. Vì ta có thể thấy, đối với mỗi kiểu dữ liệu khác nhau thì sẽ có một thuật toán sắp xếp chiếm ưu thế. Ví dụ, đối với một kiểu dữ liệu đã gần như có sẵn thứ tự, thì Insertion Sort là nhanh nhất, nhưng đối với những kiểu dữ liệu phức tạp hơn thì Insertion Sort lại không phù hợp. Như vậy, ta khó có thể kết luận được đâu mới là thuật toán sắp xếp tối ưu nhất, vì nó còn phụ thuộc vào nhiều yếu tố khác nhau như dữ liệu đầu vào, số lượng phần tử, dữ liệu đã được sắp xếp sẵn hay chưa, dung lượng bộ nhớ, tốc độ xử lý CPU,…. Đó cũng chính là lý do vì sao có sự xuất hiện của nhiều loại thuật toán sắp xếp được ra đời. Tóm lại, về lý thuyết thì Quick Sort vẫn thực sự là thuật toán sắp xếp nhanh nhất trong phần lớn các trường hợp.

**Câu 2: Trong 2 phương pháp tìm kiếm đã học, trường hợp nào thì cả 2 phương pháp đều như nhau, giải thích tại sao?**

Trong 2 phương pháp tìm kiếm đã học là tìm kiếm tuần tự và tìm kiếm nhị phân, tuy mỗi phương pháp đều có cách thức xử lý và tìm kiếm dữ liệu khác nhau, nhưng trong một số trường hợp đặc biệt, cả 2 phương pháp này đều cho ra kết quả như nhau với cùng số lần so sánh và tại một vị trí cụ thể.

Ví dụ:

* Cho một dãy các số tự nhiên gồm 8 phần tử với các giá trị như sau: 0, 2, 4, 6, 8, 10, 12, 14. Ta thực hiện tìm phần tử tại vị trí thứ 2 (phần tử có giá trị là 4) và đếm số phép so sánh của 2 phương pháp tìm kiếm.
* Đối với phương pháp **tìm kiếm tuần tự**, ta sẽ tìm thấy phần tử có giá trị là 4 tại vị trí thứ 2 trong dãy sau 3 lần so sánh.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phần tử | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 |
| Vị trí | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| So sánh lần 1 | 4 ≠ 0 |  |  |  |  |  |  |  |
| So sánh lần 2 |  | 4 ≠ 2 |  |  |  |  |  |  |
| So sánh lần 3 |  |  | **4 = 4** |  |  |  |  |  |

* Đối với phương pháp **tìm kiếm nhị phâ**n, ta cũng sẽ tìm thấy phần tử có giá trị là 4 tại vị trí thứ 2 trong dãy sau 3 lần so sánh.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Phần tử | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 |
| Vị trí | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| So sánh lần 1 |  |  |  | 4 ≠ 6 |  |  |  |  |
| So sánh lần 2 |  | 4 ≠ 2 |  |  |  |  |  |  |
| So sánh lần 3 |  |  | **4 = 4** |  |  |  |  |  |

Tóm lại, với số phần tử là 8, ta sẽ luôn tìm được phần tử tại vị trí n/4 (vị trí thứ 2) sau 3 lần tìm kiếm. Tuy nhiên, trường hợp đặc biệt này chỉ xảy ra và đúng khi dãy các phần tử này đã được sắp xếp theo thứ tự tăng dần, còn nếu dãy chưa được sắp xếp thứ tự thì trường hợp này sẽ không còn đúng cho 2 phương pháp tìm kiếm tuần tự và tìm kiếm nhị phân.

**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** hay còn gọi là sắp xếp trộn, là một giải thuật sắp xếp dựa trên giải thuật chia để trị (Divide and Conquer). Với độ phức tạp tại trường hợp xấu nhất là O(n logn) thì đây là một trong các giải thuật đáng được quan tâm nhất.

Giả sử ta có mảng sau:



Đầu tiên, giải thuật Merge Sort chia toàn bộ mảng thành hai nửa và tiến trình chia này tiếp tục diễn ra cho đến khi không còn chia được nữa. Cuối cùng ta sẽ thu được các giá trị tương ứng biểu diễn các phần tử có trong mảng.

* Bước 1: Chia mảng có kích thước là 8 phần tử thành hai mảng, mỗi mảng gồm 4 phần tử. Lưu ý, tiến trình này phải đảm bảo không làm thay đổi thứ tự các phần tử trong mảng ban đầu.



* Bước 2: Ta tiếp tục chia các mảng thành 2 nửa, cuối cùng ta có được 4 mảng, với số phần tử mỗi mảng là 2.



* Bước 3: Tiếp tục chia nhỏ các mảng cho đến khi không còn chia được nữa.



* Bước 4: Sau khi kết thúc bước 3, ta tiến hành tổ hợp các phần tử theo đúng cách thức mà chúng được chia ra. Ví dụ, 14 và 33 là các phần tử đã có thứ thự, nên chúng ta thực hiện so sánh 27 và 10, vì 27 > 10 nên ta hoán đổi vị trí của hai phần tử. Tương tự, chúng ta hoán đổi vị trí của 35 và 19, còn 42 và 44 đã có thứ tự thì giữ nguyên.



* Bước 5: Thực hiện kết hợp từng cặp mảng ở trên, ta so sánh các giá trị và sau đó hợp nhất chúng lại vào một mảng chứa 4 phần tử đã được sắp xếp thứ tự.



* Bước 6: Sau khi lần kết hợp cuối cùng được thực hiện, mảng sẽ có thứ tự như sau.



Tóm lại, ý tưởng của thuật toán Merge Sort được thực hiện như sau:

* Nếu mảng chỉ có một phần tử thì mảng này được xem như là đã được sắp xếp.
* Chia mảng bằng thuật toán đệ quy thành hai nửa cho đến khi không thể chia được nữa.
* Kết hợp các mảng con (đã qua sắp xếp) thành mảng mới với thứ tự tăng dần hoặc giảm dần.