BÀI 8 LẬP TRÌNH MỘT SỐ THUẬT TOÁN SẮP XẾP

Học xong bài này, em sẽ:

- Phát biểu được bài toán sắp xếp.
- Viết được chương trình cho một vài thuật toán sắp xếp.

Trình quản lí tệp của hệ điều hành cho phép lựa chọn hiển thị nội dung của thư mục được sắp xếp thứ tự theo vài cách khác nhau. Em hãy cho biết một trong số các lựa chọn này và giải thích rõ thêm tiêu chí (yêu cầu) sắp xếp tương ứng.

1 Bài toán sắp xếp

Một số bài toán sắp xếp như sau:

– Cho dãy các số, yêu cầu sắp xếp "theo thứ tự tăng dần (giảm dần)".

– Cho dãy các xâu kí tự, yêu cầu sắp xếp "theo thứ tự bảng chữ cái", "theo độ dài tăng dần",...

– Sắp xếp các hàng trong một bảng gồm nhiều cột (hay bản ghi trong bảng cơ sở dữ liệu) theo một cột nào đó. Ví dụ, có bảng kết quả học tập gồm các cột *Họ và tên*, *Điểm Toán*, *Điểm Ngữ văn*, *Điểm Tin học*,... yêu cầu sắp xếp theo điểm môn Tin học giảm dần. Các hàng trong bảng có dạng như sau:

```
File  Edit  Format  Run  Options  Window  Help
['Nguyễn Văn An', 7.5, 8.0, 6.5,...]
['Trần Thị Bình', 6.5, 9.0, 4.5,...]
['Phạm Minh Chi', 3.5, 9.5, 8.5,...]
```

Trong tin học, thuật ngữ sắp xếp đề cập đến việc tổ chức lại một tập hợp dữ liệu theo một tiêu chí sắp xếp, tức là đáp ứng một yêu cầu cụ thể về trình tự. Kết quả sắp xếp là một *danh sách* theo đúng thứ tự yêu cầu. Sắp xếp giúp tìm kiếm nhanh hơn. Yêu cầu sắp xếp cần chỉ rõ *cách so sánh* hai mục dữ liệu để quyết định *thứ tự*.

Dưới đây trình bày bài toán sắp xếp đơn giản và minh hoạ bằng sắp xếp dãy số.

– Đầu vào: Dãy n số $a_0, a_1, ..., a_{n-1}$.

– Đầu ra: Dãy được sắp theo thứ tự tăng dần (không giảm).

Sắp xếp tại chỗ và không tại chỗ

Một thuật toán được gọi là sắp xếp tại chỗ khi không phải dùng thêm một dãy khác ở bên ngoài dãy ban đầu để thực hiện sắp xếp. Nó chỉ đổi chỗ các phần tử trong dãy ban đầu. Yêu cầu này rất quan trọng khi dãy cần sắp xếp rất dài.

Nếu thuật toán sử dụng một dãy khác ở bên ngoài dãy ban đầu để chứa kết quả thì gọi là sắp xếp không tại chỗ.

Các thuật toán được trình bày trong bài học đều có yêu cầu sắp xếp tại chỗ và thực hiện với cấu trúc mảng, phải dịch chuyển để lấy chỗ trống khi thao tác chèn để thay đổi vị trí.

Nghịch thế

Nếu $i < j$ mà $a_i > a_j$ thì cặp hai phần tử (a_i, a_j) gọi là một nghịch thế. Dãy số chưa sắp đúng thứ tự khi còn ít nhất một nghịch thế. Một số thuật toán sắp xếp dựa trên ý tưởng giảm dần và tiến đến triệt tiêu hết các nghịch thế trong dãy. Dãy được sắp xếp xong khi không còn nghịch thế nào.

2 Thuật toán sắp xếp nổi bọt (Bubble Sort)

Dựa trên minh hoạ diễn biến từng bước của thuật toán sắp xếp nổi bọt được trình bày như ở *Hình 1*, em hãy nêu tóm tắt ý tưởng của thuật toán này.

Hình 1 trình bày diễn biến từng vòng lặp khi thực hiện thuật toán sắp xếp nổi bọt. Kí hiệu ↻ thể hiện thao tác đổi chỗ khi có nghịch thế (cặp phần tử màu đỏ).

Dãy (a)	a_0	a_1	a_2	a_3	a_4	a_5	*Giải thích diễn biến từng vòng lặp*
Ban đầu	55	19	42	94	18	67	Vòng lặp 1) 55 > 19 ↻, 55 > 42 ↻, 55 < 94, 94 > 18 ↻, 94 > 67 ↻
Sau vòng lặp 1	19	42	55	18	67	94	Vòng lặp 2) 19 < 42, 42 < 55, 55 > 18 ↻, 55 < 67, 67 < 94
Sau vòng lặp 2	19	42	18	55	67	94	Vòng lặp 3) 19 < 42, 42 > 18 ↻, 42 < 55, 55 < 67, 67 < 94
Sau vòng lặp 3	19	18	42	55	67	94	Vòng lặp 4) 19 > 18 ↻, 19 < 42, 42 < 55, 55 < 67, 67 < 94
Sau vòng lặp 4	18	19	42	55	67	94	Vòng lặp 5) 18 < 19, 19 < 42, 42 < 55, 55 < 67, 67 < 94

Hình 1. Diễn biến từng vòng lặp của thuật toán

Ở vòng lặp 5, không tìm ra nghịch thế nào nên không xảy ra đổi chỗ, dãy đã được sắp xếp đúng thứ tự. Để biết khi nào hết nghịch thế, tức là đã sắp xếp xong, ta dùng một biến logic "có đổi chỗ" nhận giá trị `True` hay `False` tuỳ theo có xảy ra đổi chỗ hay không. *Hình 2* là mã giả của thuật toán sắp xếp nổi bọt phiên bản thô nhất.

Nhận xét:

– Có thể chứng minh sau vòng lặp 1 thì số lớn nhất trong dãy sẽ được chuyển đến vị trí cuối dãy, tức là phần tử $a[n-1]$ đã ở đúng vị trí. Như vậy vòng lặp 2 chỉ cần rà soát nghịch thế và đổi chỗ đến vị trí $n-2$.

– Sau vòng lặp i thì phần tử $a[n-i-1]$ đã ở đúng vị trí.

```
while (True):
    CoDoiCho ← False
    for i in { i | 0 ≤ i < n − 1}:
        if a[i] > a[i + 1]:
            CoDoiCho ← True
            swap (a[i], a[i + 1])
    if(not CoDoiCho):
        return
```

Hình 2. Mã giả của thuật toán sắp xếp nổi bọt

3 Thuật toán sắp xếp chèn tuyến tính (Insertion Sort)

Ý tưởng sắp xếp chèn tuyến tính:

– Vì dãy con a_0 chỉ có một phần tử, nên dãy con này có thứ tự.

– Lặp lại việc chèn a_i với $1 \le i < n$ như sau:

Xét dãy con $a_0, \ldots, a_{i-1}$ đã có thứ tự, ta chèn a_i vào dãy con này sao cho dãy con sau khi chèn sẽ có thứ tự.

Mô tả thuật toán chèn tuyến tính:

Hình 3 minh hoạ diễn biến từng bước của thuật toán sắp xếp chèn tuyến tính.

Dãy (a)	a_0	a_1	a_2	a_3	a_4	a_5	***Giải thích diễn biến từng bước***
Ban đầu	55	**19**	42	94	18	67	1) Xét a_1 = 19. 55 > 19. Lấy ra 19. Dịch chuyển 55 qua phải, chèn 19 vào trước 55.
Sau bước 1	19	55	**42**	94	18	67	2) Xét a_2 = 42. 55 > 42; 19 < 42. Lấy ra 42. Dịch chuyển 55 qua phải, chèn 42 vào trước 55.
Sau bước 2	19	42	55	**94**	18	67	3) Xét a_3 = 94. 55 < 94. 94 đã nằm đúng chỗ.
Sau bước 3	19	42	55	94	**18**	67	4) Xét a_4 = 18. 94 > 18; 55 > 18; 42 > 18; 19 > 18. Lấy ra 18. Dịch chuyển dãy con 19,..., 94 qua phải, chèn 18 vào trước 19.
Sau bước 4	18	19	42	55	94	**67**	5) Xét a_5 = 67. 94 > 67; 55 < 67. Lấy ra 67. Dịch chuyển 94 qua phải, chèn 67 vào trước 94.
Sau bước 5	18	19	42	55	67	94	Sắp xếp xong.

Hình 3. Minh hoạ thuật toán sắp xếp chèn tuyến tính

Mô tả các bước của thuật toán như sau:

Bước 0. $i \leftarrow 1$;

Bước 1.

if $i \geq n$: kết thúc;

else: $val \leftarrow a_i$; $k \leftarrow i - 1$;

Bước 2.

if $k \geq 0$:

if $a_k > val$: $a_{k+1} \leftarrow a_k$; $k \leftarrow k - 1$; đến Bước 2;

Bước 3. $a_{k+1} \leftarrow val$; $i \leftarrow i + 1$; đến Bước 1;

Để "chèn a_i vào đúng chỗ của nó" trong dãy $a_0, a_1, \ldots, a_{i-1}$ cần:

a) Tìm được chỗ đúng thứ tự của a_i: cho k đi từ $i - 1$ qua trái cho đến khi $a_k \leq a_i$ hoặc $k = -1$.

b) Lấy a_i ra khỏi dãy; dịch chuyển dãy $a_{k+1}, \ldots, a_{i-1}$ sang phải một vị trí để có chỗ trống tại a_{k+1}; đưa a_i vào a_{k+1}.

Mã giả thuật toán sắp xếp chèn tuyến tính

Thuật toán sắp xếp chèn tuyến tính kết hợp làm đồng thời hai việc a) và b) theo cách dịch chuyển dần từng bước sang trái, từ vị trí i tới vị trí $k+1$.

```
for i in range(1, n):
    val ← a_i                      # Chép a_i vào val
    k ← i - 1                      # Xét dãy a_0,..., a_{i-1}

    while k ≥ 0 and a_k > val :
        a_{k+1} ← a_k              # Chép a_k vào a_{k+1}
        k ← k - 1                  # k di chuyển qua trái
    a_{k+1} ← val                  # Chép val vào a_k
```

Hình 4. Mã giả của thuật toán sắp xếp chèn tuyến tính

Vòng lặp **for** bên ngoài kiểm soát việc thực hiện đúng $n - 1$ bước (xem *Hình 4*).

Vòng lặp **while** lồng bên trong thực hiện đồng thời cùng lúc hai việc a) và b) trong mỗi bước (xem *Hình 4*).

4 Thực hành

Nhiệm vụ 1. Em hãy thực hiện các công việc sau:

a) Tính số lần lặp của vòng lặp bên trong của thuật toán sắp xếp chèn tuyến tính.

b) Tính số lần lặp của vòng lặp ngoài của thuật toán sắp xếp chèn tuyến tính.

c) Ước lượng độ phức tạp thời gian của thuật toán sắp xếp chèn tuyến tính.

Nhiệm vụ 2. Em hãy viết chương trình Python thực hiện thuật toán sắp xếp nổi bọt.

Nhiệm vụ 3. Em hãy viết chương trình Python thực hiện thuật toán sắp xếp chèn tuyến tính dựa trên mã giả đã cho trong bài học.

Cho danh sách *Bảng điểm* là kết quả học tập gồm các cột *Họ và tên*, *điểm Toán*, *điểm Ngữ văn*, *điểm Tin học*,... Hãy viết chương trình sắp xếp *Bảng điểm* theo điểm môn Tin học giảm dần.

Gợi ý: Mỗi phần tử của *Bảng điểm* là một danh sách con, ứng với một học sinh. So sánh theo thành phần điểm Tin học của danh sách con để sắp xếp.

Theo em, thuật toán sắp xếp nổi bọt và thuật toán sắp xếp chèn, thuật toán nào đơn giản và dễ cài đặt hơn?

Tóm tắt bài học

- Thuật toán sắp xếp nổi bọt có hai vòng lặp lồng nhau: vòng lặp trong thực hiện đổi chỗ một lượt các cặp phần tử là nghịch thế, vòng lặp ngoài kiểm tra điều kiện "không xảy ra đổi chỗ".
- Việc tìm vị trí chèn đúng chỗ trong thuật toán sắp xếp chèn có thể thực hiện bằng cách dịch dần từng bước.

BÀI TÌM HIỂU THÊM

CÁC HÀM PYTHON SẮP XẾP DÃY SỐ

Python có hai hàm để sắp xếp dãy số:

Hàm `sorted`: Sắp xếp dãy đầu vào theo thứ tự tăng dần. Hàm trả về một dãy mới gồm các phần tử trong dãy cũ nhưng đã sắp xếp lại theo yêu cầu. Dãy cũ vẫn còn đó.

Hàm `sort`: Thực hiện sắp xếp tại chỗ, tức là đổi chỗ các phần tử trong dãy. Hàm trả về `None` để tránh nhầm lẫn.

Chú ý: `sort` chỉ áp dụng cho danh sách; trái lại, `sorted` chấp nhận đầu vào tổng quát hơn.

Cú pháp:

```
sorted (dãy_số, key = ..., reverse = ...)
danh_sách.sort(key = ..., reverse = ...)
```

Trong đó:

`reverse` = `True` nghĩa là xếp theo thứ tự giảm dần, mặc định là `reverse` = `False`.

`key` dùng để xác định một hàm (`key function`) được áp dụng cho mỗi phần tử trong dãy trước khi làm phép so sánh.