Chương 4

CÁC THUẬT TOÁN SẮP XẾP

4.1. Các thuật toán sắp xếp cơ bản

4.1.1. Sắp xếp chọn (Selection Sort)

Giải thuật

Đây là phương pháp sắp xếp đơn giản nhất được tiến hành như sau:

• Đầu tiên chọn phần tử có khóa nhỏ nhất trong n phần tử từ a[1] đến a[n] và hoán vị nó với phần tử a[1].

• Chọn phần tử có khóa nhỏ nhất trong n-1phần tử từ a[2] đến a[n] và hoán vị nó với a[2].

• Tổng quát ở bước thứ i, chọn phần tử có khoá nhỏ nhất trong n-i+1 phần tử từ a[i] đến a[n] và hoán vị nó với a[i].

• Sau n-1 bước này thì mảng đã được sắp xếp.

Phương pháp này được gọi là phương pháp chọn bởi vì nó lặp lại quá trình chọn phần tử nhỏ nhất trong số các phần tử chưa được sắp.

Ví dụ 2-1: Sắp xếp mảng gồm 10 mẩu tin có khóa là các số nguyên: 5, 6, 2, 2, 10, 12, 9, 10, 9 và 3

Bước 1: Ta chọn được phần tử có khoá nhỏ nhất (bằng 2) trong các phần tử từ a[1] đến a[10] là a[3], hoán đổi a[1] và a[3] cho nhau. Sau bước này thì a[1] có khoá nhỏ nhất là 2.

Bước 2: Ta chọn được phần tử có khoá nhỏ nhất (bằng 2) trong các phần tử từ a[2] đến a[10] là a[4], hoán đổi a[2] và a[4] cho nhau.

Tiếp tục quá trình này và sau 9 bước thì kết thúc.

Bảng sau ghi lại các giá trị khoá tương ứng với từng bước.

Khóa / Bước	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
Ban đầu	5	6	2	2	10	12	9	10	9	3
Bước 1	**2**	6	5	2	10	12	9	10	9	3
Bước 2		**2**	5	6	10	12	9	10	9	3
Bước 3			**3**	6	10	12	9	10	9	5
Bước 4				**5**	10	12	9	10	9	6
Bước 5					**6**	12	9	10	9	10
Bước 6						**9**	12	10	9	10
Bước 7							**9**	10	12	10
Bước 8								**10**	12	10
Bước 9									**10**	12
Kết quả	**2**	**2**	**3**	**5**	**6**	**9**	**9**	**10**	**10**	**12**

Bảng 4.1: Các bước thực hiện sắp xếp chọn

Chương trình:

```
PROCEDURE SelectionSort;
VAR
i,j,LowIndex: integer;
LowKey: KeyType;
BEGIN
{1}    FOR i := 1 TO n-1 DO BEGIN
{2}    LowIndex := i;
{3}    LowKey := a[i].key;
{4}    FOR j := i+1 TO n DO
{5}    IF a[j].key < LowKey THEN
BEGIN
{6}    LowKey := a[j].key;
{7}    LowIndex := j;
END;
{8} Swap(a[i],a[LowIndex]);
END;
```

END;

Đánh giá: Phương pháp sắp xếp chọn lấy O(n) để sắp xếp n phần tử.

Trước hết ta có thủ tục Swap lấy một hằng thời gian như đã nói ở mục 2.2.3.

Các lệnh {2}, {3} đều lấy O(1) thời gian. Vòng lặp for {4} – {7} thực hiện n-i lần, vì j chạy từ i+1 đến n, mỗi lần lấy O(1), nên lấy O(n-i) thời gian. Do đó thời gian tổng cộng là: $O(n^2)$.

4.1.2. Sắp xếp chèn (Insert Sort)

Giải thuật

Trước hết ta xem phần tử a[1] là một dãy đã có thứ tự.

Bước 1, xen phần tử a[2] vào danh sách đã có thứ tự a[1] sao cho a[1], a[2] là một danh sách có thứ tự.

Bước 2, xen phần tử a[3] vào danh sách đã có thứ tự a[1], a[2] sao cho a[1], a[2], a[3] là một danh sách có thứ tự.

Tổng quát, bước i, xen phần tử a[i+1] vào danh sách đã có thứ tự a[1],a[2],..a[i] sao cho a[1], a[2],.. a[i+1] là một danh sách có thứ tự.

Phần tử đang xét a[j] sẽ được xen vào vị trí thích hợp trong danh sách các phần tử đã được sắp trước đó a[1],a[2],..a[j-1] bằng cách so sánh khoá của a[j] với khoá của a[j-1] đứng ngay trước nó. Nếu khoá của a[j] nhỏ hơn khoá của a[j-1] thì hoán đổi a[j-1] và a[j] cho nhau và tiếp tục so sánh khoá của a[j-1] (lúc này a[j-1] chứa nội dung của a[j]) với khoá của a[j-2] đứng ngay trước nó...

Ví dụ 2-2: Sắp xếp mảng gồm 10 mẩu tin đã cho trong ví dụ 2-1.

Bước 1: Xen a[2] vào dãy chỉ có một phần tử a[1] ta được dãy hai phần tử a[1]..a[2] có thứ tự. Việc xen này thực ra không phải làm gì cả vì hai phần tử a[1], a[2] có khoá tương ứng là 5 và 6 đã có thứ tự.

Bước 2: Xen a[3] vào dãy a[1]..a[2] ta được dãy ba phần tử a[1]..a[3] có thứ tự. Việc xen này được thực hiện bằng cách : so sánh khoá của a[3] với khoá của a[2], do khoá của a[3] nhỏ hơn khoá của a[2] (2<6) nên hoán đổi

a[3] và a[2] cho nhau. Lại so sánh khoá của a[2] với khoá của a[1], do khoá của a[2] nhỏ hơn khoá của a[1] (2<5) nên hoán đổi a[2] và a[1] cho nhau.

Tiếp tục quá trình này và sau 9 bước thì kết thúc.

Bảng sau ghi lại các giá trị khoá tương ứng với từng bước.

Khóa / Bước	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	A[8]	a[9]	a[10]
Ban đầu	5	6	2	2	10	12	9	10	9	3
Bước 1	**5**	**6**								
Bước 2	**2**	**5**	**6**							
Bước 3	**2**	**2**	**5**	**6**						
Bước 4	**2**	**2**	**5**	**6**	**10**					
Bước 5	**2**	**2**	**5**	**6**	**10**	**12**				
Bước 6	**2**	**2**	**5**	**6**	**9**	**10**	**12**			
Bước 7	**2**	**2**	**5**	**6**	**9**	**10**	**10**	**12**		
Bước 8	**2**	**2**	**5**	**6**	**9**	**9**	**10**	**10**	**12**	
Bước 9	**2**	**2**	**3**	**5**	**6**	**9**	**9**	**10**	**10**	**12**

Bảng 4.2: Các bước sắp xếp chèn

Chương trình

```
PROCEDURE InsertionSort;
VAR
i,j: integer;
BEGIN
{1} FOR i := 2 TO n DO BEGIN
{2} J := i;
{3} WHILE (j>1) AND (a[j].key < a[j-1].key) DO BEGIN
{4} swap(a[j], a[j-1]);
{5} j := j-1;
END;
END;
END;
```

Đánh giá: Phương pháp sắp xếp xen lấy O(n) để sắp xếp n phần tử.

Ta thấy các lệnh {4} và {5} đều lấy O(1). Vòng lặp {3} chạy nhiều nhất i-1 lần, mỗi lần tốn O(1) nên {3} lấy i-1 thời gian. Lệnh {2} và {3} là hai lệnh nối tiếp nhau, lệnh {2} lấy O(1) nên cả hai lệnh này lấy i-1.

Vòng lặp {1} có i chạy từ 2 đến n nên nếu gọi T(n) là thời gian để sắp n phần tử thì ta có

$$T(n) = \sum_{i=2}^{n} (i-1) = \frac{n(n-1)}{2} \text{ tức là } O(n^2).$$

4.1.3. Sắp xếp nổi bọt (Bubble Sort)

Giải thuật

Chúng ta tưởng tượng rằng các mẩu tin được lưu trong một mảng dọc, qua quá trình sắp, mẩu tin nào có khóa "nhẹ" sẽ được nổi lên trên. Chúng ta duyệt tòan mảng, từ dưới lên trên. Nếu hai phần tử ở cạnh nhau mà không đúng thứ tự tức là nếu phần tử "nhẹ hơn" lại nằm dưới thì phải cho nó "nổi lên" bằng cách đổi chỗ hai phần tử này cho nhau. Cụ thể là:

Bước 1: Xét các phần tử từ a[n] đến a[2], với mỗi phần tử a[j], so sánh khoá của nó với khoá của phần tử a[j-1] đứng ngay trước nó. Nếu khoá của a[j] nhỏ hơn khoá của a[j-1] thì hoán đổi a[j] và a[j-1] cho nhau.

Bước 2: Xét các phần tử từ a[n] đến a[3], và làm tương tự như trên.

Sau n-1 bước thì kết thúc.

Ví dụ 2-3: Sắp xếp mảng gồm 10 mẩu tin đã cho trong ví dụ 2-1.

Bước 1: Xét a[10] có khoá là 3, nhỏ hơn khoá của a[9] nên ta hoán đổi a[10] và a[9] cho nhau. Khoá của a[9] bây giờ là 3 nhỏ hơn khoá của a[8] nên ta hoán đổi a[9] và a[8] cho nhau. Khoá của a[8] bây giờ là 3 nhỏ hơn khoá của a[7] nên ta hoán đổi a[8] và a[7] cho nhau. Khoá của a[7] bây giờ là 3 nhỏ hơn khoá của a[6] nên ta hoán đổi a[7] và a[6] cho nhau. Khoá của a[6] bây giờ là 3 nhỏ hơn khoá của a[5] nên ta hoán đổi a[6] và a[5] cho nhau. Khoá của a[5] bây giờ là 3 **không nhỏ hơn** khoá của a[4] nên bỏ qua. Khoá của a[4] là 2 **không nhỏ hơn** khoá của a[3] nên bỏ qua. Khoá của a[3] là 2 nhỏ hơn khoá của a[2] nên ta hoán đổi a[3] và a[2] cho nhau. Khoá của a[2] bây

giờ là 2 nhỏ hơn khoá của a[1] nên ta hoán đổi a[2] và a[1] cho nhau. Đến đây kết thúc bước 1 và a[1] có khoá nhỏ nhất là 2.

Bước 2: Xét a[10] có khoá là 9, nhỏ hơn khoá của a[9] nên ta hoán đổi a[10] và a[9] cho nhau. Khoá của a[9] bây giờ là 9 **không nhỏ hơn** khoá của a[8] nên bỏ qua. Khoá của a[8] là 9 nhỏ hơn khoá của a[7] nên ta hoán đổi a[8] và a[7] cho nhau. Khoá của a[7] bây giờ là 9 nhỏ hơn khoá của a[6] nên ta hoán đổi a[7] và a[6] cho nhau. Khoá của a[6] bây giờ là 9 **không nhỏ hơn** khoá của a[5] nên bỏ qua. Khoá của a[5] bây giờ là 3 **không nhỏ hơn** khoá của a[4] nên bỏ qua. Khoá của a[4] là 2 nhỏ hơn khoá của a[3] nên ta hoán đổi a[4] và a[3] cho nhau. Khoá của a[3] bây giờ là 2 nhỏ hơn khoá của a[2] nên ta hoán đổi a[3] và a[2] cho nhau. Đến đây kết thúc bước 2 và a[2] có khoá là 2.

Tiếp tục quá trình này và sau 9 bước thì kết thúc.

Bảng sau ghi lại các giá trị khoá tương ứng với từng bước.

Khóa / Bước	a[1]	a[2]	a[3]	A[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]
Ban đầu	5	6	2	2	10	12	9	10	9	3
Bước 1	**2**	5	6	2	3	10	12	9	10	9
Bước 2		**2**	5	6	3	9	10	12	9	10
Bước 3			**3**	5	6	9	9	10	12	10
Bước 4				**5**	6	9	9	10	10	12
Bước 5					**6**	9	9	10	10	12
Bước 6						**9**	9	10	10	12
Bước 7							**9**	10	10	12
Bước 8								**10**	10	12
Bước 9									**10**	12
Kết quả	**2**	**2**	**3**	**5**	**6**	**9**	**9**	**10**	**10**	**12**

Bảng 4.3: Các bước sắp xếp nổi bọt

Chương trình

```
PROCEDURE BubbleSort;
VAR
i,j: integer;
BEGIN
```

```
{1} FOR i := 1 to n-1 DO
{2} FOR j := n DOWNTO i+1 DO
{3} IF a[j].key < a[j-1].key THEN
{4} Swap(a[j],a[j-1]);
END;
```

Đánh giá: Phương pháp sắp xếp nổi bọt lấy O(n) để sắp n phần tử.

Dòng lệnh {3} lấy một hằng thời gian. Vòng lặp {2} thực hiện (n-i) bước, mỗi bước lấy O(1) nên lấy O(n-i) thời gian. Như vậy đối với toàn bộ chương trình ta có:

$$T(n)= \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2} = O(n^2).$$

4.2. Sắp xếp nhanh (Quick Sort)

4.2.1. Tư tưởng

Chúng ta vẫn xét mảng a các mẩu tin a[1]..a[n]. Giả sử v là 1 giá trị khóa mà ta gọi là chốt (pivot). Ta phân hoạch dãy a[1]..a[n] thành hai mảng con "bên trái" và "bên phải". Mảng con "bên trái" bao gồm các phần tử có **khóa nhỏ hơn chốt**, mảng con "bên phải" bao gồm các phần tử có **khóa lớn hơn hoặc bằng chốt**.

Sắp xếp mảng con “bên trái” và mảng con “bên phải” thì mảng đã cho sẽ được sắp bởi vì tất cả các khóa trong mảng con “bên trái“ đều nhỏ hơn các khóa trong mảng con “bên phải”.

Việc sắp xếp các mảng con “bên trái” và “bên phải” cũng được tiến hành bằng phương pháp nói trên.

Một mảng chỉ gồm một phần tử hoặc gồm nhiều phần tử có khóa bằng nhau thì đã có thứ tự.

4.2.2. Giải thuật

Vấn đề chọn chốt

Chọn **khóa lớn nhất** trong hai phần tử có khóa khác nhau đầu tiên kể từ trái qua. Nếu mảng chỉ gồm một phần tử hay gồm nhiều phần tử có khóa bằng nhau thì không có chốt**.**

Ví dụ 2-5: Chọn chốt trong các mảng sau

Cho mảng gồm các phần tử có khoá là 6, 6, 5, 8, 7, 4, ta chọn chốt là 6 (khoá của phần tử đầu tiên).

Cho mảng gồm các phần tử có khoá là 6, 6, 7, 5, 7, 4, ta chọn chốt là 7 (khoá của phần tử thứ 3).

Cho mảng gồm các phần tử có khoá là 6, 6, 6, 6, 6, 6 thì không có chốt (các phần tử có khoá bằng nhau).

Cho mảng gồm một phần tử có khoá là 6 thì không có chốt (do chỉ có một phần tử).

Vấn đề phân hoạch

Để phân hoạch mảng ta dùng 2 "con nháy" L và R trong đó L từ bên trái và R từ bên phải, ta cho L chạy sang phải cho tới khi gặp phần tử có khóa ≥ chốt và cho R chạy sang trái cho tới khi gặp phần tử có khóa < chốt. Tại chỗ dừng của L và R nếu L < R thì hoán vị a[L],a[R]. Lặp lại quá trình dịch sang phải, sang trái của 2 "con nháy" L và R cho đến khi L > R. Khi đó L sẽ là điểm phân hoạch, cụ thể là a[L] là phần tử đầu tiên của mảng con “bên phải”.

Giải thuật QuickSort

Để sắp xếp mảng a[i]..a[j] ta tiến hành các bước sau:

• Xác định chốt.

• Phân hoạch mảng đã cho thành hai mảng con a[i]..a[k-1] và a[k]..a[j]**.**

• Sắp xếp mảng a[i]..a[k-1] (Đệ quy).

• Sắp xếp mảng a[k]..a[j] (Đệ quy).

Quá trình đệ quy sẽ dừng khi không còn tìm thấy chốt.

```
Procedure quicksoft(t,p:integer);
var i,j,x,m:integer;
```

```
begin
    i:=t;j:=p;
    m:=a[(i+j) div 2];
    While (i<=j) do
    Begin
        while (a[i]<m) do i:=i+1;
        while (a[j]>m) do j:=j-1;
        if (i<=j) then
        begin
            hoanvi(a[i],a[j]);
            i:=i+1;
            j:=j-1;
        end;
        if (t<j) then quicksoft(t,j);
        if(i<p) then quicksoft(i,p);
end;
end;
```

Ví dụ 2-4: Sắp xếp mảng gồm 10 mẩu tin có khóa là các số nguyên: 5, 8, 2, 10, 5, 12, 8, 1, 15 và 4.

Với mảng a[1]..a[10], hai phần tử đầu tiên có khóa khác nhau là là a[1] và a[2] với khoá tương ứng là 5 và 8, ta chọn chốt v = 8.

Để phân hoạch, khởi đầu ta cho L := 1 (đặt L ở cực trái) và R := 10 (đặt R ở cực phải). Do a[L] có khoá là 5 nhỏ hơn chốt nên L := L+1 = 2 (di chuyển L sang phải), lúc này a[L] có khoá là 8 = chốt nên dừng lại. Do a[R] có khoá là 4 nhỏ hơn chốt nên R cũng không chuyển sang trái được. Tại các điểm dừng của L và R ta có L < R (L=2 và R=10) nên hoán đổi a[L] và a[R] (a[2] và a[10]) cho nhau. Sau khi hoán đổi, a[L] lại có khoá là 4 nhỏ hơn chốt nên di chuyển L sang phải (L := L+1 = 3). Khoá của a[L] là 2 nhỏ hơn chốt nên lại di

chuyển L sang phải (L := L+1 = 4). Khoá của a[L] là 10 lớn hơn chốt nên dừng lại. Với R, khoá của a[R] bây giờ là 8 bằng chốt nên di chuyển R sang trái (R := R-1 = 9). Khoá của a[R] là 15 lớn hơn chốt nên di chuyển R sang trái (R := R-1 = 8). Khoá của a[R] là 1 nhỏ hơn chốt nên dừng lại. Tại các điểm dừng của L và R ta có L < R (L=4 và R=8) nên hoán đổi a[L] và a[R] (a[4] và a[8]) cho nhau. Sau khi hoán đổi, a[L] có khoá là 1 nhỏ hơn chốt nên di chuyển L sang phải (L := L+1 = 5). Khoá của a[L] là 5 nhỏ hơn chốt nên lại di chuyển L sang phải (L := L+1 = 6). Khoá của a[L] là 12 lớn hơn chốt nên dừng lại. Với R, khoá của a[R] bây giờ là 10 lớn hơn chốt nên di chuyển R sang trái (R := R-1 = 7). Khoá của a[R] là 8 bằng chốt nên di chuyển R sang trái (R := R-1 = 6). Khoá của a[R] là 12 lớn hơn chốt nên di chuyển R sang trái (R := R-1 = 5). Khoá của a[R] là 5 nhỏ hơn chốt nên dừng lại. Tại các điểm dừng của L và R ta có L > R (L=6 và R=5) nên ta đã xác định được điểm phân hoạch ứng với L = 6. Tức là mảng đã cho ban đầu được phân thành hai mảng con bên trái a[1]..a[5] và mảng con bên phải a[6]..a[10]. Hình ảnh của sự phân hoạch này được biểu diễn như sau:

Chỉ số	1	2	3	4	5	6	7	8	9	10
Khoá Ban đầu	5	8 → 4	2	10 → 1	5	12	8	1 → 10	15	4
					v = 8					
Cấp 1	5	4	2	1	5	12	8	10	15	8

Trong bảng trên, dòng *chỉ số* ghi các chỉ số của các phần tử của mảng (từ 1 đến 10).

Trong dòng *khoá ban đầu*, các giá trị khoá ở dòng trên (5, 8, 2, 10, 5, 12, 8, 1, 15 và 4) là các giá trị khoá của mảng đã cho ban đầu, các giá trị khoá ở dòng dưới (4, 1, 10 và 8) là các giá trị khoá mới sau khi thực hiện hoán đổi a[2] với a[10] và a[4] với a[8].

Giá trị chốt là v = 8.

Dòng cấp c*ấp 1*, biểu diễn hai mảng con sau khi phân hoạch. Mảng bên trái từ a[1] đến a[5] gồm các phần tử có khoá là 5, 4, 2, 1 và 5. Mảng con bên phải từ a[6] đến a[10] gồm các phần tử có khoá 12, 8, 10, 15 và 8.

Tiếp tục sắp xếp đệ quy cho mảng con bên trái và mảng con bên phải.