Chương 9

SẮP XẾP

9.1 Đặt vấn đề

Sắp xếp (Sorting) là quá trình bố trí lại các phần tử của một tập đối tượng nào đó, theo một thứ tự ấn định. Chẳng hạn thứ tự tăng dần (hay giảm dần) đối với một dãy số, thứ tự từ điển đối với một dãy chữ.v.v...

Yêu cầu về sắp xếp thường xuyên xuất hiện trong các ứng dụng tin học, với những mục đích khác nhau: sắp xếp dữ liệu lưu trữ trong máy tính để tìm kiếm cho thuận lợi, sắp xếp các kết quả xử lý để in ra trên bảng biểu...

Nói chung, dữ liệu có thể xuất hiện dưới nhiều dạng khác nhau, nhưng ở đây ta quy ước: tập đối tượng được sắp xếp là tập các bản ghi (records), mỗi bản ghi bao gồm một số trường (fields) dữ liệu, tương ứng với những thuộc tính (attributs) khác nhau.

Trong chương này ta chỉ xét tới các phương pháp *sắp xếp trong* (internal sorting), nghĩa là các phương pháp tác động trên một tập các bản ghi lưu trữ đồng thời ở bộ nhớ trong, mà ta gọi là *bảng* (table) để phân biệt với tệp (file) hiện nay thường được dùng để chỉ một tập lớn các bản ghi lưu trữ ở bộ nhớ ngoài, và sắp xếp tương ứng với tệp sẽ được gọi là *sắp xếp ngoài* (external sorting).

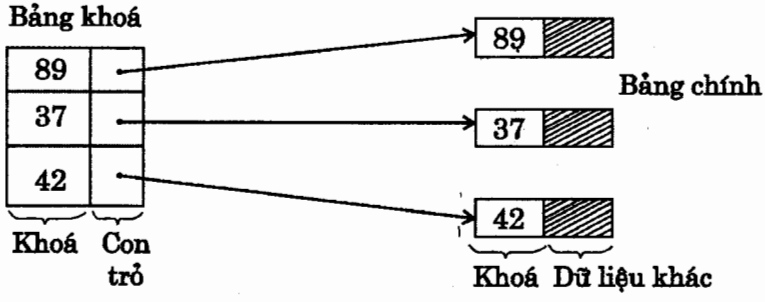
Việc xét các phương pháp sắp xếp, cũng như tìm kiếm, đối với tệp sẽ được đề cập ở chương 11.

Như vậy bài toán được đặt ra ở đây là sắp xếp đối với một bảng gồm n bản ghi $R_1, R_2,..., R_n$.

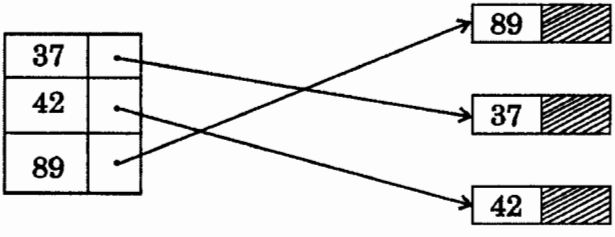
Tuy nhiên, ta thấy: Không phải toàn bộ các trường dữ liệu trong bản ghi đều được xem xét đến trong quá trình sắp xếp mà chỉ một trường nào đó (hoặc một vài trường nào đó - nhưng trường hợp này ta sẽ không đề cập đến) dược chú ý tới thôi. Trường như vậy gọi là *khoá* (key). Sắp xếp sẽ được tiến hành dựa vào giá trị của khoá nầy.

Ví dụ: Bảng danh mục điện thoại các cơ quan nhà nước ở Hà Nội, bao gồm các bản ghi ứng với từng cơ quan. Mỗi bản ghi gồm có ba trường: Tên cơ quan, địa chỉ, số điện thoại. Ở đây khoá sắp xếp chính là tên cơ quan.

Khi sắp xếp, các bản ghi trong bảng sẽ được bố trí lại các vị trí sao cho giá trị khoá sắp xếp tương ứng với chúng có đúng thứ tự ấn định. Ta thấy kích thước của khoá thường khá nhỏ so với kích thước bản ghi. Sắp xếp nếu thực hiện trực tiếp trên các bản ghi của bảng sẽ kéo theo sự chuyển đổi vị trí của các bản ghi và việc đó có thể đòi hỏi phải sao chép lại toàn bộ thông tin của bản ghi vào chỗ mới, gây ra tốn phí thời gian khá nhiều. Thường người ta khắc phục tình trạng này bằng cách xây dựng một bảng phụ, cũng gồm n bản ghi như bảng chính nhưng mỗi bản ghi ở đây chỉ có hai trường: (khoá, con trỏ). Trường "khoá" chứa giá trị của khoá ứng với từng bản ghi trong bảng chính, trường "con trỏ" chứa con trỏ, trỏ tới bản ghi tương ứng. Bảng phụ này được gọi là *bảng khoá* (key table), sắp xếp sẽ thực hiện trên bảng khoá đó. Như vậy, trong quá trình sắp xếp, bảng chính không hề bị ảnh hưởng gì còn việc truy nhập vào bản ghi nào đó của bảng chính, khi cần thiết, vẫn thực hiện được bằng cách dựa vào con trỏ của bản ghi tương ứng thuộc bảng khoá này.



a) Trước khi sắp xếp



b) Sau khi sắp xếp

Hình 9.1

Do khoá có vai trò đặc biệt như vậy nên sau này, khi trình bày các phương pháp cũng như giải thuật hay trong các ví dụ minh hoạ, ta sẽ coi khoá như đại diện cho các bản ghi và để cho đơn giản ta chỉ nói tới giá trị khoá thôi. Thực ra phép đổi chỗ được tác động vào các bản ghi nhưng ở đây ta cũng chỉ nói tới phép đổi chỗ đối với khoá, và bài toán sắp xếp bây giờ coi như được đặt ra một cách đơn giản với một bảng các khoá (dãy khoá) $K_1, K_2,...,K_n$ (tương ứng với các bản ghi $R_1, R_2, ..., R_n$) và $K_i \neq K_j$ nếu $i \neq j$.

Tất nhiên giá trị của khoá có thể là số, là chữ và thứ tự sắp xếp cũng được quy định tương ứng với khoá. Nhưng ở đây, để minh hoạ cho các phương pháp ta sẽ coi giá trị khoá là số và thứ tự sắp xếp là thứ tự tăng dần. Để nhất quán, ta sẽ dùng một dãy khoá sau đây:

42 23 74 11 65 58 94 36 99 87

để làm ví dụ.

9.2 Một số phương pháp sắp xếp cơ bản

9.2.1 Sắp xếp kiểu lựa chọn (selection sort)

Một trong những phương pháp đơn giản nhất để thực hiện sắp xếp một bảng khoá là dựa trên phép lựa chọn.

Nguyên tắc cơ bản của phương pháp sắp xếp này là "ở lượt thứ i (i = 1, 2, ..., n) ta sẽ chọn trong dãy khoá $K_i, K_{i+1}, ..., K_n$ khoá nhỏ nhất và đổi chỗ nó với K_i".

Như vậy thì rõ ràng là sau j lượt, j khoá nhỏ hơn đã lần lượt ở các vị trí thứ nhất, thứ hai,..., thứ j theo đúng thứ tự sắp xếp. Ví dụ

i	K_i	lượt	1	2	3	4	...	9
1	42		11	11	11	11		11
2	23		23	23	23	23		23
3	74		74	74	36	36		36
4	11		42	42	42	42		42
5	65		65	65	65	65		58
6	58		58	58	58	58		65
7	94		94	94	94	94		74
8	36		36	36	74	74		87
9	99		99	99	99	99		94
10	87		87	87	87	87		99

Sau đây là giải thuật:

Proceduce SELECT-SORT (K,n)

{Cho dãy khoá K gồm n phần tử. Giải thuật này thực hiện sắp xếp các phần tử của K theo thứ tự tăng dần, dựa vào phép chọn phần tử nhỏ nhất trong mỗi lượt}

```
1.  for i: = 1 to n-1 do
2.          begin
                    m := i;
3.                  for j: = i+1 to n do
                    if K[j] < K[m] then m: = j;
4.                          if m ≠ j then
                            begin { đổi chỗ }
                                    X:= K[i];
                                    K[i]: = K[m]
                                    K[m]: = X
                            end
            end
5.  return
```

9.2.2 Sắp xếp kiểu thêm dần (insertion sort)

Nguyên tắc sắp xếp ở đây dựa theo kinh nghiệm của những người chơi bài. Khi có i-1 lá bài đã được sắp xếp đang ở trên tay, nay rút thêm lá bài thứ i nữa thì sắp xếp lại như thế nào? - Có thể so sánh lá bài mới lần lượt với lá bài thứ (i-1), thứ (i-2)... để tìm ra "chỗ" thích hợp và "chèn" nó vào chỗ đó.

Dựa trên nguyên tắc này có thể triển khai một cách sắp xếp như sau:

Thoạt đầu K_1 được coi như bảng chỉ gồm có một khoá đã sắp xếp. Xét thêm K_2, so sánh nó với K_1 để xác định chỗ "chèn" nó vào, sau đó ta sẽ có một bảng gồm hai khoá đã được sắp xếp. Đối với K_3 lại so sánh với K_2, K_1 và cứ tương tự như vậy đối với K_4, K_5, K_6.v.v... cuối cùng sau khi xét xong K_n thì bảng khoá đã được sắp xếp hoàn toàn.

Ta thấy ngay phương pháp này rất thuận lợi khi các khoá của dãy được đưa dần vào miền lưu trữ. Đó cũng chính là không gian nhớ dùng để sắp xếp. Có thể minh hoạ qua bảng sau:

Lượt	1	2	3	4		8	9	10
Khoá đưa vào	42	23	74	11		36	39	87
1	42	23	23	11		11	11	11
2	-	42↓	42	23↓		23↓	23	23
3	-	-↓	74	42↓		36↓	36	36
4	-	-	-	74↓		42↓	42	42
5	-	-	-	-		58↓	58	58
6	-	-	-	-		65↓	65	65
7	-	-	-	-		74↓	74	74↓
8	-	-	-	-		94↓	94	87↓
9	-	-	-	-		-	99	95↓
10	-	-	-	-		-	-	99↓

(Dấu - chỉ chỗ trống trong miền sắp xếp

Dấu ↓ chỉ việc phải dịch chuyển khoá cũ để lấy chỗ chèn khoá mới vào)

* Nhưng nếu các khoá đã có mặt ở bộ nhớ trong trước lúc sắp xếp rồi thì sao?

Sắp xếp vẫn có thể thực hiện được ngay tại chỗ chứ không phải chuyển sang một miền sắp xếp khác. Lúc đó các khoá cũng sẽ lần lượt được xét tới và việc xác định chỗ cho khoá mới vẫn làm tương tự, chỉ có khác là: để dành chỗ cho khoá mới nghĩa là phải dịch chuyển một số khoá lùi lại sau, ta không có sẵn chỗ trống như trường hợp nói trên (vì khoá đang xét và các khoá sẽ được xét đã chiếm các vị trí đằng sau này rồi), do đó phải đưa khoá mới này ra một chỗ nhớ phụ và sẽ đưa vào vị trí thực của nó sau khi đã đẩy các khoá cần thiết lùi lại.

Sau đây là giải thuật ứng với trường hợp này.

*** Procedure** INSERT-SORT (K,n)

{Trong thủ tục này người ta dùng X làm ô nhớ phụ để chứa khoá mới đang được xét. Để đảm bảo cho khóa mới trong mọi trường hợp, ngay cả khi vị trí thực của nó là vị trí đầu tiên, đều được "chèn" vào giữa khoá nhỏ hơn nó và khoá lớn hơn nó, ở đây đưa thêm vào một khoá giả K_0, có giá trị nhỏ hơn mọi khoá của bảng, và đứng trước mọi khoá đó. Ta qui ước $K_0 = -\infty$}.

1. K[0] = - ∞
2. **for** i:= 2 to n **do begin**
3. X := K[i]; j:= i-1;

4. {Xác định chỗ cho khoá mới được xét và dịch chuyển các khoá cần thiết}

```
while x < K[j] do begin
        K[j + 1] := K[j];
        j := j - 1;
              end;
```

5. {đưa X vào đúng chỗ} K[j + 1] := X

```
end;
```

6. **return**

Bảng ví dụ minh hoạ tương ứng với các lượt sắp xếp theo giải thuật này, tương tự như bảng đã nêu ở trên, chỉ có khác là không có chỗ nào trống trong miền sắp xếp cả, vì những chỗ đó đang chứa các khoá chưa được xét tới trong mỗi lượt (người đọc có thể tự lập ra bảng ví dụ minh hoạ này).

9.2.3 Sắp xếp kiểu đổi chỗ (exchange sort)

Trong các phương pháp sắp xếp nêu trên, tuy kỹ thuật đổi chỗ đã được sử dụng nhưng nó chưa trở thành một đặc điểm nổi bật. Bây giờ ta mới xét tới phương pháp mà việc đổi chỗ một cặp khoá kế cận, khi chúng ngược thứ tự, sẽ được thực hiện thường xuyên cho tới khi toàn bộ bảng các khoá đã được sắp xếp. Ý cơ bản có thể nêu như sau:

Bảng các khoá sẽ được duyệt từ đáy lên đỉnh. Dọc đường, nếu gặp hai khoá kế cận ngược thứ tự thì đổi chỗ chúng cho nhau. Như vậy trong lượt đầu khoá có giá trị nhỏ nhất sẽ chuyển dần lên đỉnh. Đến lượt thứ hai khoá có giá trị nhỏ thứ hai sẽ được chuyển lên vị trí thứ hai.v.v... Nếu hình dung dãy khoá được đặt thẳng đứng thì sau từng lượt sắp xếp các giá trị khoá nhỏ sẽ "nổi" dần lên giống như các bọt nước nổi lên trong nồi nước đang sôi. Vì vậy phương pháp này thường được gọi bằng cái tên khá đặc trưng là: sắp xếp kiểu nổi bọt (buble sort).

Ví dụ:

i	K_i	lượt 1	2	3	4	5	. . .	9
1	42	11	11	11	11	11		11
2	23	42	23	23	23	23		23
3	74	23	42	36	36	36		36
4	11	74	36	42	42	42		42
5	65	36	74	58	58	58		58
6	58	65	58	74	65	65		65
7	94	58	65	65	74	74		74
8	36	94	87	87	87	87		87
9	99	87	94	94	94	94		94
10	87	99	99	99	99	99		99

Sau đây là giải thuật:

Procedure BUBBLE- SORT(K,n)

1. **For** i:=1 **to** n - 1 **do**
2. **for** j:= n **down to** i + 1 **do**
3. **if** K[j] < K[j-1] **then**
   begin
4. x:= K [j];
   K[j] := K[j-1];
   K[j-1] := x
   end;
5. **return**

Giải thuật này rõ ràng còn có thể cải tiến được nhiều. Chẳng hạn xét qua ví dụ ở trên ta thấy: sau lượt thứ ba không phải chỉ có ba khoá 11, 23, 36 vào đúng vị trí sắp xếp của nó mà là 5 khoá. Còn sau lượt thứ tư thì tất cả các khoá đã nằm đúng vào vị trí của nó rồi. Như vậy nghĩa là năm lượt cuối không có tác dụng gì thêm cả. Từ đó có thể thấy: nếu "nhớ" được vị trí của khoá được đổi chỗ cuối cùng ở mỗi lượt thì có thể coi đó là "giới hạn" cho việc xem xét ở lượt sau. Chừng nào mà giới hạn này chính là vị trí thứ n, nghĩa là trong lượt ấy không có một phép đổi chỗ nào nữa thì sắp xếp có thể kết thúc được. Nhận xét này sẽ dẫn tới một giải thuật cải tiến hơn, chắc chắn có thể làm cho số lượt giảm đi và số lượng các phép so sánh trong mỗi lượt cũng giảm đi nữa. Người đọc hãy tự xây dựng giải thuật theo ý cải tiến này (coi như một bài tập).

9.2.4 Phân tích so sánh ba phương pháp

Đối với một phương pháp sắp xếp, khi xét tới hiệu lực của nó ngoài những đánh giá về mặt không gian nhớ cần thiết, người ta thường lưu ý đặc biệt tới chi phí về thời gian. Mà thời gian thì chủ yếu phụ thuộc vào việc thực hiện các phép so sánh giá trị khoá và các phép chuyển chỗ bản ghi, khi sắp xếp. Vì vậy thông thường người ta lấy số lượng trung bình các phép so sánh hoặc các phép chuyển chỗ làm đại lượng đặc trưng cho chi phí về thời gian thực hiện của từng phương pháp. Ở đây phép so sánh sẽ được coi như một "phép toán tích cực" để đánh giá thời gian thực hiện của các giải thuật.

Đối với phương pháp sắp xếp kiểu lựa chọn, ta thấy: ở lượt thứ i (i = 1, 2, ..., n-1) để tìm khoá nhỏ nhất bao giờ cũng cần $C_i = (n-i)$ phép so sánh. Số lượng phép so sánh này không hề phụ thuộc gì vào tình trạng ban đầu của dãy khoá cả. Từ đó suy ra:

$$C_{min} = C_{max} = C_{tb} = \sum_{i=1}^{n-1}(n-i) = \frac{n(n-1)}{2}$$

Còn đối với phương pháp sắp xếp kiểu thêm dần (giải thuật INSERT-SORT) thì có hơi khác. Rõ ràng số lượng phép so sánh phụ thuộc và dãy khoá ban đầu. Trường hợp thuận lợi nhất ứng với dãy khoá đã được sắp xếp rồi. Như vậy ở mỗi lượt chỉ cần 1 phép so sánh. Do đó:

$$C_{min} = \sum_{i=2}^{n} 1 = n-1$$

Nhưng nếu dãy khoá ban đầu có thứ tự ngược với thứ tự sắp xếp thì ở lượt thứ i phải cần có: $C_i = (i-1)$ phép so sánh. Vì vậy:

$$C_{max} = \sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2}$$

Nếu giả sử mọi giá trị khoá đều xuất hiện đồng khả năng thì trung bình ở lượt thứ i có thể coi như $C_i = \frac{i}{2}$ phép so sánh. Suy ra:

$$C_{tb} = \sum_{i=2}^{n}\frac{i}{2} = \frac{n^2+n-2}{4}$$

Với sắp xếp kiểu nổi bọt theo giải thuật BUBBLE-SORT như trên, nghĩa là chưa hề có cải tiến gì, thì tương tự như phương pháp đầu, ta cũng có:

$$C_{min} = C_{max} = C_{tb} = \frac{n(n-1)}{2}$$

Nhìn vào các kết quả đánh giá ở trên ta thấy INSERT-SORT tỏ ra "tốt hơn" so với hai phương pháp kia. Tuy nhiên, với n khá lớn, chi phí về thời

gian thực hiện được đánh giá qua cấp độ lớn, thì cả ba phương pháp đều có cấp $O(n^2)$ và đây vẫn là một chi phí cao so với một số phương pháp mà ta sẽ xét thêm sau đây:

9.3 Sắp xếp kiểu phân đoạn (Partition - Sort) hay sắp xếp "nhanh" (Quick - Sort)

9.3.1. Giới thiệu phương pháp

Sắp xếp kiểu phân đoạn là một cải tiến của phương pháp sắp xếp kiểu đổi chỗ. Đây là một phương pháp khá tốt, do đó người sáng lập ra nó C.A.R. Hoare, đã mạnh dạn đặt cho nó cái tên hấp dẫn là sắp xếp NHANH.

Ý chủ đạo của phương pháp có thể tóm tắt như sau:

Chọn một khoá ngẫu nhiên nào đó của dãy làm "chốt" (pivot). Mọi phần tử nhỏ hơn khoá "chốt" phải được xếp vào vị trí ở trước "chốt" (đầu dãy), mọi phần tử lớn hơn khoá "chốt" phải được xếp vào vị trí sau "chốt" (cuối dãy). Muốn vậy, các phần tử trong dãy sẽ được so sánh với khoá chốt và sẽ đổi vị trí cho nhau, hoặc cho chốt, nếu nó lớn hơn chốt mà lại nằm trước chốt hoặc nhỏ hơn chốt mà lại nằm sau chốt. Khi việc đổi chỗ đã thực hiện xong thì dãy khoá lúc đó được phân làm hai đoạn: một đoạn gồm các khoá nhỏ hơn chốt, một đoạn gồm các khoá lớn hơn chốt còn khoá chốt thì ở giữa hai đoạn nói trên, đó cũng là vị trí thực của nó trong dãy khi đã được sắp xếp, tới đây coi như kết thúc một lượt sắp xếp.

Ở các lượt tiếp theo cũng áp dụng một kỹ thuật tương tự đối với các phân đoạn còn lại. Lẽ tất nhiên chỉ có một phân đoạn được xử lý ngay sau đó, còn một phân đoạn phải để lúc khác, nghĩa là phải được "ghi nhớ" lại.

Quá trình xử lý một phân đoạn, ghi nhớ phân đoạn còn lại được thực hiện tiếp tục cho tới khi gặp một phân đoạn chỉ gồm có một phần tử thì việc ghi nhớ không cần nữa. Lúc đó một phân đoạn mới sẽ được xác định và đối với phân đoạn này quá trình lặp lại tương tự. Sắp xếp sẽ kết thúc khi phân đoạn cuối cùng đã được xử lý xong.

9.3.2 Ví dụ và giải thuật

Giả sử, ta qui ước chọn khoá "chốt" là khoá đầu tiên của dãy. Như vậy với dãy

42 23 74 11 65 58 94 36 99 87

thì chốt là 42.