Chương 4: Giải thuật sắp xếp và tìm kiếm đơn giản

1. Sắp xếp chọn (Selection Sort)

1.1. Phương pháp

- Giả sử cần sắp xếp tăng dần một dãy khoá $a_1, a_2,..., a_n$.
- Ý tưởng của thuật toán như sau:
  - Chọn phần tử có khoá nhỏ nhất .
  - Đổi chỗ nó với phần tử a_1.
  - Sau đó lặp lại thao tác trên với n-1 phần tử còn lại, rồi lại lặp lại như trên với n-2 phần tử còn lại,..., cho tới khi chỉ còn 1 phần tử.

1.1. Phương pháp (*tiếp*)

- Ví dụ:

Cho dãy khoá ban đầu là: 6, 10, 1, 8, 9

với n=5.

i=1 1, 10, 6, 8, 9

i=2 1, 6, 10, 8, 9

i=3 1, 6, 8, 10, 9

i=4 1, 6, 8, 9, 10

1.1. Phương pháp (*tiếp*)

```
Procedure  selectionSort(a,n);
      For i:= 1 to n-1 Do
       Begin
            {Tìm phần tử nhỏ nhất ở vị trí k }
            k:=i;
            For  j:=i+1 To  n  Do
                  If  a[j] < a[k]  then  k:=j
            {Đổi chỗ phần tử nhỏ nhất k cho phần tử i}
            If k ≠ i then a[k]↔a[i];
       End
Return
```

2.2. Đánh giá giải thuật

- Với giải thuật trình bày ở trên thì phép toán tích cực là phép so sánh (a[j]<a[k]).
- Gọi C là số lượng phép so sánh, C được tính như sau: Ở lượt thứ i (i=1, 2,... , n-1), để tìm khoá nhỏ nhất cần n-i phép so sánh. Số lượng phép so sánh này không phụ thuộc vào tình trạng ban đầu của dãy khoá. Do đó ta có:

- Vậy, độ phức tạp tính toán là $O(n^2)$

2. Sắp xếp chèn (Insert Sort)

2.1. Phương pháp

- Phương pháp này được những người chơi bài hay dùng.
- Giả sử cần sắp xếp tăng dần dãy khoá a_1, a_2,..., a_n. Ý tưởng thuật toán như sau:
  - Các phần tử được chia thành dãy đích: a_1,..., a_{i-1} (kết quả) và dãy nguồn a_i,..., a_n.
  - Bắt đầu với i=2, ở mỗi bước phần tử thứ i của dãy nguồn được lấy ra và chèn vào vị trí thích hợp trong dãy đích sao cho dãy đích vẫn tăng dần. Sau đó i tăng lên 1 và lặp lại.

2.1. Phương pháp

- Ví dụ: Cho dãy khoá 6, 10, 1, 7, 4 với n=5 (dãy số có 5 phần tử).

	Dãy đích	Dãy nguồn
	6	10, 1, 7, 4
i=2	6, 10	1, 7, 4
i=3	1, 6, 10	7, 4
i=4	1, 6, 7, 10	4
i=5	1, 4, 6, 7, 10	

Thủ tục chèn

```
Procedure insertSort(a,n)
    1) a[0]:=-∞
    2) For i:=2 to n Do
       Begin
            tg:=a[i];  j:=i-1;
            While  tg<a[j] Do
              Begin
                   a[j+1]:=a[j];  j:=j-1;
              End;
            a[j+1]:=tg;  {chèn tg vào sau a[j]}
       End;
Return
```

2.2. Đánh giá thuật toán

- Phép toán tích cực trong thuật toán này là phép so sánh (tg<a[j]). Số phép toán so sánh C được tính như sau:
  - Trường hợp thuận lợi nhất là dãy khoá $a_1, a_2,..., a_n$ đã được sắp, như vậy mỗi lần chỉ cần 1 phép so sánh. Do vậy

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

2.2. Đánh giá thuật toán

- Trường hợp xấu nhất nếu dãy khoá sắp theo thứ tự ngược với thứ tự sắp xếp thì ở lượt i cần có: C= (i-1) phép so sánh. Do vậy

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

- Trường hợp trung bình: Giả sử mọi giá trị khoá đều xuất hiện đồng khả năng thì trung bình phép so sánh ở lượt thứ i là $C_i = i/2$, do đó số phép so sánh trung bình của giải thuật này là:

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

- $O(n^2)$

3. Sắp xếp nổi bọt (Bubble Sort)

3.1. Phương pháp

- Giả sử cần sắp xếp tăng dần dãy khoá $a_1, a_2,..., a_n$. Ý tưởng thuật toán như sau:
  - So sánh từng cặp khóa liền kề, gối nhau từ phải qua trái, nếu khóa đứng sau nhỏ hơn khóa đứng trước thì đổi chỗ. Loạt so sánh thứ nhất thì khóa nhỏ nhất của dãy được đẩy lên vị trí đầu tiên (gọi là phần tử được sắp).
  - Tiếp tục so sánh và đổi chỗ các phần tử liền kề gối nhau của dãy chưa sắp, lần thứ 2 ta được số nhỏ nhất của dãy chưa sắp được đưa lên đầu dãy chưa sắp (ví trí 2).
  - Cứ tiếp tục làm tương tự như trên cho đến khi dãy chỉ còn 1 phần tử.

3.1. Phương pháp (*tiếp*)

- Ví dụ: Cho dãy khoá ban đầu là: 6, 3, 7, 10, 1, 8 với n=6.

	6, 3, 7, 10, 1, 8
i=1	1, 6, 3, 7, 10, 8
i=2	1, 3, 6, 7, 8, 10
i=3	1, 3, 6, 7, 8, 10
i=4	1, 3, 6, 7, 8, 10
i=5	1, 3, 6, 7, 8, 10

Thủ tục sắp xếp nổi bọt

```
Procedure   bubbleSort(a,n)
    For  i:= 1  to  n-1  Do
     For  j:= n downto i+1  Do
          If  a[j]<a[j-1]  then
                a[j] <-> a[j-1];
Return
```


3.2. Đánh giá thuật toán

- Giải thuật này tương tự như giải thuật sắp xếp bằng cách chọn trực tiếp (mục 1), do đó có:

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

- Nhận xét: Với 3 phương pháp sắp xếp trên, nếu n vừa và nhỏ thì phương pháp chèn trực tiếp (insert sort) tỏ ra tốt hơn, nếu với n lớn thì cả 3 phương pháp đều có cấp $O(n^2)$, đây là một chi phí thời gian khá cao.

4. Tìm kiếm tuần tự (Sequence Search)

4.1. Bài toán tìm kiếm

Cho dãy khóa là các số nguyên có n phần tử. Tìm khóa có giá trị bằng x.

Gọi x là khoá tìm kiếm hay giá trị tìm kiếm. Công việc tìm kiếm sẽ hoàn thành khi có một trong 2 tình huống sau xảy ra:

1- Tìm được khóa có giá trị bằng x. Lúc đó ta nói phép tìm kiếm được thoả.

2- Không tìm được khóa nào có giá trị bằng x. Khi đó ta nói phép tìm kiếm không thoả.

Sau phép tìm kiếm không thoả nếu có yêu cầu bổ sung x vào dãy khóa. Giải thuật này gọi là " Tìm kiếm có bổ sung".

4.2. Tìm kiếm tuần tự (Sequential Searching)

4.2.1. Phương pháp

Đây là giải thuật đơn giản, cổ điển.

Cách thức làm như sau: Bắt đầu từ khóa thứ nhất, lần lượt so sánh khoá tìm kiếm với các khóa trong dãy cho đến khi tìm thấy khóa mong muốn hoặc đã hết dãy mà chưa thấy.

4.2.2. Giải thuật

Cho dãy khoá K có n phần tử. Tìm xem có khoá nào bằng x, nếu có đưa ra thứ tự của khoá đó, nếu không có thì đưa ra giá trị 0. Trong giải thuật sử dụng khoá phụ k_{n+1}=x.


```
Function  sequenceSearch(k,n,x)
    1. { Khởi đầu }
    i:=1; k[n+1]:=x;
    2. { Tìm kiếm trong dãy}
    While  k[i] <> x  Do  i:=i+1;
    3. { Không tìm thấy }
    If  i=n+1  then  Return(0)  Esle  Return(i);
Return
```

4.2.3. Đánh giá

Trong giải thuật này phép toán tích cực là phép so sánh k[i] <> x .

- Trường hợp thuận lợi ta tìm thấy ngay ở phần tử đầu nên $C_{min} = 1$
- Trường hợp xấu nhất ta phải so sánh n+1 lần nên $C_{max} = n+1$
- Giả sử hiện tượng khoá tìm kiếm x trùng với một khoá nào đó của bảng là đồng khả năng thì $C_{tb} = (n+1)/2$

Do đó $T_{tb} = O(n)$.

- Nếu trường hợp hiện tượng khoá tìm kiếm x trùng với một khoá nào đó của bảng là không đồng khả năng, mà xác suất để xuất hiện $k_i = x$ là p_i, $p_i \neq 1/n$. Khi đó $C_{tb}=1*p_1 + 2p_2 + \ldots + n*p_n$

với $\sum_{i=1}^{n} p_i = 1$

Từ công thức trên ta nhận thấy với $p_1>p_2 > \ldots >p_n$ thì thời gian trung bình sẽ nhỏ nhất. Như vậy ta phải sắp xếp trước khi tìm kiếm.