**8.4. THUẬT TOÁN SẮP XẾP KIỂU CHÈN**

Xét dãy khoá k1, k2, …, kn. Ta thấy dãy con chỉ gồm mỗi một khoá là k1 có thể coi là đã sắp xếp rồi. Xét thêm k2, ta so sánh nó với k1, nếu thấy k2 < k1 thì chèn nó vào trước k1. Đối với k3, ta lại xét dãy chỉ gồm 2 khoá k1, k2 đã sắp xếp và tìm cách chèn k3 vào dãy khoá đó để được thứ tự sắp xếp. Một cách tổng quát, ta sẽ sắp xếp dãy k1, k2, …, ki trong điều kiện dãy k1, k2, …, ki-1 đã sắp xếp rồi bằng cách chèn ki vào dãy đó tại vị trí đúng khi sắp xếp.

procedure InsertionSort;

var

i, j: Integer;

tmp: TKey; {Biến giữ lại giá trị khoá chèn}

begin

for i := 2 to n do {Chèn giá trị ki vào dãy k1,…, ki-1 để toàn đoạn k1, k2,…, ki trở thành đã sắp xếp}

begin

tmp := ki; {Giữ lại giá trị ki}

j := i - 1;

while (j > 0) and (tmp < kj) do {So sánh giá trị cần chèn với lần lượt các khoá kj (i-1j0)}

begin

kj+1 := kj; {Đẩy lùi giá trị kj về phía sau một vị trí, tạo ra "khoảng trống" tại vị trí j}

j := j - 1;

end;

kj+1 := tmp; {Đưa giá trị chèn vào "khoảng trống" mới tạo ra}

end;

end;

Đối với thuật toán sắp xếp kiểu chèn, thì chi phí thời gian thực hiện thuật toán phụ thuộc vào tình trạng dãy khoá ban đầu. Nếu coi phép toán tích cực ở đây là phép so sánh tmp < kj thì: Trường hợp tốt nhất ứng với dãy khoá đã sắp xếp rồi, mỗi lượt chỉ cần 1 phép so sánh, và như vậy tổng số phép so sánh được thực hiện là n - 1.

Trường hợp tồi tệ nhất ứng với dãy khoá đã có thứ tự ngược với thứ tự cần sắp thì ở lượt thứ i, cần có i - 1 phép so sánh và tổng số phép so sánh là:

(n - 1) + (n - 2) + … + 1 = n \* (n - 1) / 2.

Trường hợp các giá trị khoá xuất hiện một cách ngẫu nhiên, ta có thể coi xác suất xuất hiện mỗi khoá là đồng khả năng, thì có thể coi ở lượt thứ i, thuật toán cần trung bình i / 2 phép so sánh và tổng số phép so sánh là:

(1 / 2) + (2 / 2) + … + (n / 2) = (n + 1) \* n / 4.

Nhìn về kết quả đánh giá, ta có thể thấy rằng thuật toán sắp xếp kiểu chèn tỏ ra tốt hơn so với thuật toán sắp xếp chọn và sắp xếp nổi bọt. Tuy nhiên, chi phí thời gian thực hiện của thuật toán sắp xếp kiểu chèn vẫn còn khá lớn. Và xét trên phương diện tính toán lý thuyết thì **cấp của thuật toán sắp xếp kiểu chèn vẫn là O(n2).**

Có thể cải tiến thuật toán sắp xếp chèn nhờ nhận xét: Khi dãy khoá k1, k2, …, ki-1 đã được sắp xếp thì việc tìm vị trí chèn có thể làm bằng thuật toán tìm kiếm nhị phân và kỹ thuật chèn có thể làm bằng các lệnh dịch chuyển vùng nhớ cho nhanh. Tuy nhiên điều đó cũng không làm tốt hơn cấp độ phức tạp của thuật toán bởi trong trường hợp xấu nhất, ta phải mất n - 1 lần chèn và lần chèn thứ i ta phải dịch lùi i khoá để tạo ra khoảng trống trước khi đẩy giá trị khoá chèn vào chỗ trống đó.

procedure InsertionSortwithBinarySearching;

var

i, inf, sup, median: Integer;

tmp: TKey;

begin

for i := 2 to n do

begin

tmp := ki; {Giữ lại giá trị ki}

inf := 1; sup := i - 1; {Tìm chỗ chèn giá trị tmp vào đoạn từ kinf tới ksup+1}

repeat {Sau mỗi vòng lặp này thì đoạn tìm bị co lại một nửa}

median := (inf + sup) div 2; {Xét chỉ số nằm giữa chỉ số inf và chỉ số sup}

if tmp < k[median] then sup := median - 1

else inf := median + 1;

until inf > sup; {Kết thúc vòng lặp thì inf = sup + 1 chính là vị trí chèn}

<Dịch các phần tử từ kinf tới ki-1 lùi sau một vị trí>

kinf := tmp; {Đưa giá trị tmp vào "khoảng trống" mới tạo ra}

end;

end;