MODULE 17

SORTING OF NUMBERS USING INSERTION SORT

Introduction

- The insertion sort works very well the numbers of elements are very less.
- This technique is similar to the way a librarian keeps the books in self.
- Initially all the books are placed in shelf according to their access number.
- When a student returns the book to the librarian, he compares the access number of this book with all other books of access numbers and inserts it into the correct position, so that all books are arranged in order with respect to their access numbers.

Algorithm: Insertion Sort

It works the way you might sort a hand of playing cards:

1. We start with an empty left hand [sorted array] and the cards face down on the table [unsorted array].
2. Then remove one card [key] at a time from the table [unsorted array], and insert it into the correct position in the left hand [sorted array].
3. To find the correct position for the card, we compare it with each of the cards already in the hand, from right to left.

Note that at all times, the cards held in the left hand are sorted and these cards were originally the top cards of the pile on the table.

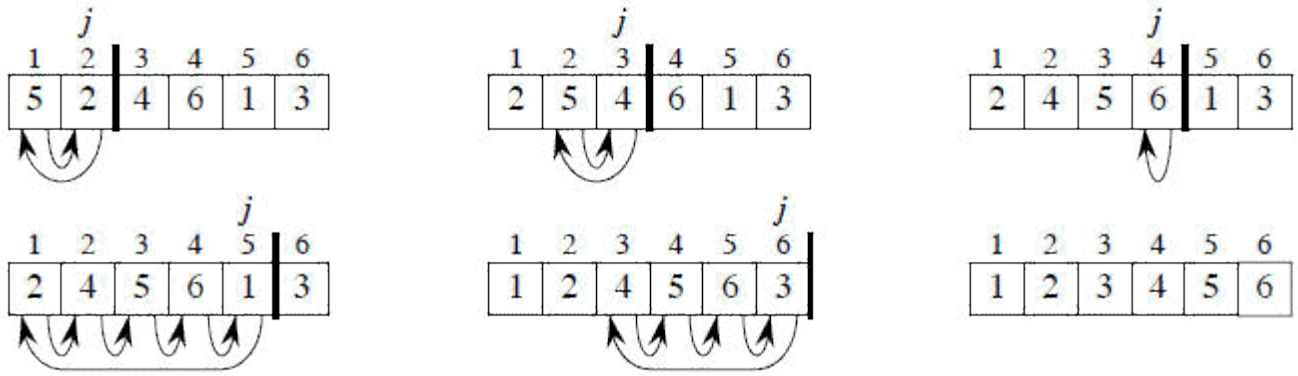
Pseudo code

We use a procedure INSERTION_SORT. It takes as parameters an array $A[1..n]$ and the length n of the array. The array A is sorted in place: the numbers are rearranged within the array, with at most a constant number outside the array at any time.

INSERTION_SORT (A)

```
1.    FOR j ← 2 TO length[A]
2.          DO  key ← A[j]
3.               {Put A[j] into the sorted sequence A[1 . . j − 1]}
4.               i ← j − 1
5.               WHILE i > 0 and A[i] > key
6.                         DO A[i +1] ← A[i]
7.                                i ← i − 1
8.               A[i + 1] ← key
```

Example: Following figure (from CLRS) shows the operation of INSERTION-SORT on the array A= (5, 2, 4, 6, 1, 3). Each part shows what happens for a particular iteration with the value of *j* indicated. *j* indexes the "current card" being inserted into the hand.



Read the figure row by row. Elements to the left of *A*[*j*] that are greater than *A*[*j*] move one position to the right, and *A*[*j*] moves into the evacuated position.

Exercise problem:

Sort {2,6,3,9,2,6,3,4,56} using insertion sort.

Hints to solve:

Pass1: The first element in the array is itself is sorted.

Pass2: 2nd element is inserted either before or after 1st element, so that 1st and 2nd elements are sorted.

Pass3: 3rd element is inserted into its proper place, which is before 1st and 2nd, or between 1st and 2nd elements, or after 1st and 2nd elements.

Pass4: 4th element is inserted into its proper place in 1st, 2nd, and 3rd so that 1st, 2nd, 3rd, 4th elements are sorted.

PassN: Nth, element is inserted into its proper place so that total list is sorted.

Advantages:

- Simple implementation.
- Efficient for (quite) small data sets.

Disadvantages:

- It is less efficient on list containing more number of elements.
- As the number of elements increases the performance of the program would be slow.
- Insertion sort needs a large number of element shifts