SORTING :→

One of the fundamental operation in computer science.

Refers to arrange data in some given order such as ascending order or descending (decreasing) order.

We begin our discussion of sorting algorithms by considering two simple and most popular schemes that are typically studied in introductory programming courses: these are

1. Bubble sort and
2. Selection sort

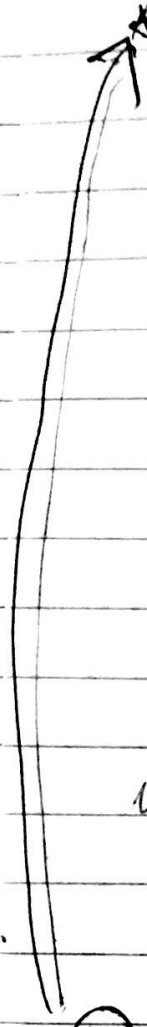
Bubble sort is already discussed in previous sem. Neither of these schemes is very efficient but used because of their simplicity.

Another sorting scheme that we discuss after these two is Insertion sort which is preferred over these two, especially for small lists.

Bubble Sort :-

In bubble sort methods we compares pairs of list element and interchanges as required.

It makes repeated passes through the list and sublists till list get sorted.



After 1st pass and n-1 comparisons we get largest data at the bottom of the list.

Hence if list has got n elements than there can be at most (n-1) comparisons and in ith pass there may be at most (n-i) comparisons.

This case is known as worst case for bubble sort algorithm and it occurs when list elements are in reverse order.

On the first pass through the list (n-1) comparisons and interchanges are made and only the largest element is correctly positioned.

On the second pass only

(n-1) elements will be scanned and (n-1) comparisons and exchanges occur and the next largest element will be in the position.

This continued until the subset consisting of the first two elements are to be scanned, and on this pass, one comparison and interchange occurs.

Thus a total of

$$(n-1)+(n-2)+(n-3)+\cdots+3+2+1$$

$$= \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} = O(n^2)$$

comparisons and interchanges required to sort the list

It follows that the worst case computing time for bubble sort is $O(n^2)$.

The average complexity is also $O(n^2)$ but this is considerably more difficult to show.

Selection Sort :→

The selection sort algorithm works as follows:

Logic

First find the smallest element in the list and put it in the first position. Then find the second smallest element in the list and put it in the second position. This process continues till we get the sorted list.

Basic Idea

The basic idea of a selection sort of a list is to make a number of passes through the list and on each pass select one element and position it correctly (properly) in the list.

Example

For example if list is to be arranged in ascending order then in the first pass the ~~Sma~~ smallest item will be selected and moved to its proper location (that is first location) and so on.

ALGORITHM

SIMPLE SELECTION SORT FOR ARRAY BASED LISTS :-

(1) Set I = 1

(2) while I <= (n-1) do the following steps (3) and (4) Repeat

[This loop compares elements 1st through second last one by one and to find the smallest in the sublist

(3) [First find the smallest element in the list in I^{th} pass]

(a) Set smallpos equal to I

(b) set

smallest = A[smallpos]

(c) Set J := I+1

(d) while J <= n do the following Repeat steps (i) and (ii)

(i) If A[J] < smallest Then

Set

smallpos = J

and

Set smallest = A[smallpos]

else

set J := J+1

(e) [Now interchange the smallest element with the element at the beginning of the Sublist]

ALGORITHM:

① Set A[0] = -∞ [Initialize Sentinel]

② Repeat steps ③ to ⑤ for I := 2 to N

③ ⓐ Set INSELMT := A[I]

ⓑ Set INSPOS := I-1

④ Repeat while INSELMT < A[INSPOS]

ⓐ Set [To make room of element]

A[INSPOS+1] := A[INSPOS]

ⓑ Set

INSPOS := INSPOS-1

⑤ [Now Insert element in proper place]

A[INSPOS+1] := INSELMT.

⑥ EXIT

Complexity

The worst case for insertion sort is once again the case in which the list elements are in reverse order.

Inserting A[2] requires two comparisons, A[3] requires for 3 comparison. and so a.

The total number of comparison is thus:

$$2 + 3 + \cdots + n = \frac{n(n+1)}{2} - 1$$

So the computing time is $O(n^2)$. This is also the average case complexity.