# Design and Analysis of Algorithms Lecture 3

Dr. G P Raja Sekhar
Department of Mathematics
Indian Institute of Technology, Kharagpur

rajas@maths.iitkgp.ernet.in

## Selection Sort and Insertion Sort

Two different algorithms for the same task may differ with respect to complexity or efficiency. It depends upon the various statements within the algorithm.

Let's consider sorting of data in ascending order.

### Selection Sort

**Algorithm 1** Selection Sort

```
1: procedure SELECTION-SORT((A(1...n)))
2:     for i ← 1, n − 1 do
3:         J-MIN ← i                          /* c1 */
4:         X-MIN ← A(i)                       /* c2 */
5:         for j ← i + 1, n do
6:             if A(j) < X-MIN then           /* c3 */
7:                 J-MIN ← j                  /* c4 */
8:                 X-MIN ← A(j)               /* c5 */
9:             end if
10:        end for
11:        A(J-MIN) ← A(i)                    /* c6 */
12:        A(i) ← X-MIN
13:    end for
14: end procedure
```

Where $c_1, c_2, \ldots c_7$ are constants.

**For Worst Case (Data in descending order)**

Total time is

$$\begin{aligned} t(n) &= (n-1)(c_1+c_2) + \sum_{1}^{n-1}(n-i)(c_3+c_4+c_5) + (n-1)(c_6+c_7) \\ &= (n-1)a_1 + \frac{(n-1)n}{2}a_2 + (n-1)a_3 \\ &= k_1n^2 + k_2n + k_3 \end{aligned}$$

$t(n) \in O(n^2)$. Quadratic.

**For Best Case (Data in ascending order)**

Here $c_4, c_5$ are zero

$$t(n) = (n-1)a + nb + c$$

$t(n) \in O(n^2)$. Again quadratic.
Therefore, selection sort is insensitive to the instance.

## Insertion Sort

Now we consider another sorting algorithm which is more sensitive and efficient compared to selection sort. It is called *insertion sort*

**Algorithm 2** Insertion Sort

```
1:  procedure INSERTION-SORT((A(1, 2, ..., n)))
2:      for i ← 2, n do
3:          x ← A(i)                          /* c1 */
4:          j ← i − 1                         /* c2 */
5:          while j > 0 and x < A(j) do
6:              A(j + 1) ← A(j)               /* c3 */
7:              j ← j − 1                     /* c4 */
8:          end while
9:          A(j + 1) ← x                      /* c5 */
10:     end for
11: end procedure
```

The statements within the while loop can be executed from zero up to a maximum of $i$ times.

**Worst Case**

Since $i$ goes from 2 to $n$, worst case time of the while loop in this procedure is given by $\sum_{i=2}^{n}(i-1)(c_3+c_4)$. This will happen when the elements of the array

are arranged in reverse order i.e., in decreasing order. So the total running time is

$$\begin{aligned} t(n) &= (n-1)(c_1+c_2) + \sum_{i=2}^{n}(i-1)(c_3+c_4) + (n-1)c_5 \\ &= (n-1)d_1 + (\frac{n(n+1)}{2} - 1)d_2 \\ &= an^2 + bn + c \in O(n^2) \end{aligned}$$

**Best Case**

For the best case i.e., if the elements of the array are already in the ascending order then the while loop does not execute, so the total time

$$t(n) = an + b \in O(n^2)$$

So the insertion sort is sensitive to the input.