Worst – case preformance of InsertionSort

Worst - case number of comparisons while inserting
i - th element ($1 \le i \le n$) to a sorted sequence of ($i - 1$) elements :

i - 1
(* the meaning of i in this file is different than
the meaning of i in the previous file - average performance *)

So, the worst - case number of comparisons done by insertion sort is

$$T(n) = \sum_{i=1}^{n} (i - 1)$$

Factor$\left[\sum_{i=1}^{n} (i - 1)\right]$

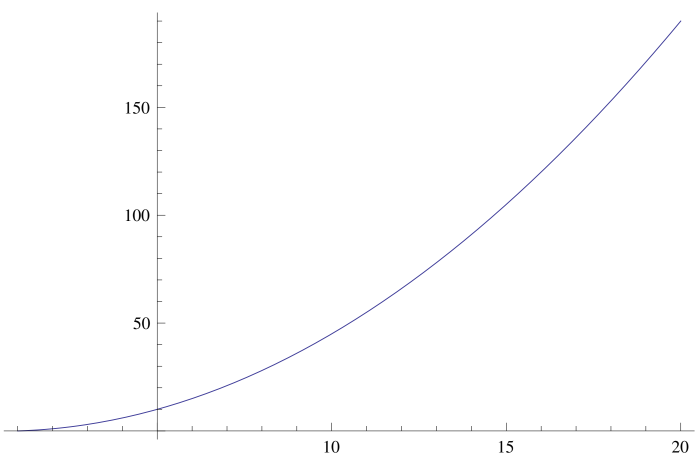
$$\frac{1}{2} (-1 + n)\, n$$

Thus

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

Exercise : Find two (2) different permutations of the set
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
that make worst cases of size 10 for InsertionSort.
(There are more than 2. Can you figure out how many different permutations of the set
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10} make all the worst cases for size 10 are there?)

Plot$\left[\frac{1}{2} n (n - 1), \{n, 1, 20\}\right]$



$$\text{Limit}\left[\frac{\frac{1}{2} n (n-1)}{n^2}, \{n \to \infty\}\right]$$

$$\left\{\frac{1}{2}\right\}$$

So,

$T(n) \in \Theta\left(n^2\right)$

$$\text{Limit}\left[\frac{\frac{1}{2} n (n-1)}{n\,\text{Log}[n]}, \{n \to \infty\}\right]$$

$\{\infty\}$

So,

$T(n) \notin O(n \log n)$

Interesting fact (optional for all students)

Consider BinaryInsertionSort that works like InsertionSort except that it runs the standard binary search algorithm in order to determine the place for an insertion of the next element.

This does not save any moves of keys that still need to be shifted up in order to make a room for insertion but it saves quite a lot of comparisons of keys in the worst case.

Since the number of comparisons of keys performed in the worst case while inserting a key into an ordered array of i keys is now the same as the number of comparisons of keys that the Binary Search performes in the worst case while searching for a key in an ordered array of i keys, the former is equal to

$\lfloor \text{Log2}[i] \rfloor + 1 = \lceil \text{Log2}[i+1] \rceil$.

Thus the number of comparisons of keys performed in the worst case by BinaryInsertionSort is the sum of the above with i ranging from 0 (inserting the first key into an empty array) to n - 1 (inserting the last n - th key into an n - 1 - element array. This is equal to

$\sum_{i=0}^{n-1} \lceil \text{Log2}[i+1] \rceil = \sum_{i=1}^{n} \lceil \text{Log2}[i] \rceil =$

$n \lceil \text{Log2}[n] \rceil - 2^{\lceil \text{Log2}[n] \rceil} + 1$.