Let us as an example consider a hypothetical list represented by the image below.



I(k) , the numbers above the squares, represent how many elements to the left of the chosen element that are in fact greater than the chosen element. It can be seen that each element will be required to move I(k) to the left. Element A for example will move one spot to the left, and take the left of C. Element B will move 3 spots to the left. Unlike more sophisticated sorting algorithms, insertion sort searches by moving iteratively one step at a time until it encounters an element that is smaller.

We can expand on this further and demonstrate that I(L) work is done. Here is an exhaustive demonstration of the entire algorithm for the list above.



As an example, let’s take D. D has two elements that are greater to its left. By the time it reaches D, it has already been sorted. It will take the left of F then it will take the left of E, where it will remain. Corresponding to I(k)=2.

We will say that I(L)=. Which is the total amount of work doing replacements searches included.

In addition, the movement of the identifier from one location to the next from a to b to c etc takes order 1. This is true whether there is a replacement or not. Thus it is O(n).

So we can say that insertion sort takes O(n+I(L))