Insertion Sort

Basics, Algorithm and Program

Kazi Reyazul Hasan [82] Wasif Jalal Galib [84]
Mubasshira Musarrat [88]

Department of Computer Science Engineering
Bangladesh University of Engineering and Technology

February 25, 2023

Index

1. Introduction
2. Visual Representation
3. Implementation
4. Analysis

What is Insertion Sort?

Insertion Sort is a type of sorting algorithm.

Sorting Alogorithm

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements in computer language.

Example

- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

What is Insertion Sort?

Insertion Sort is a type of sorting algorithm.

Sorting Alogorithm

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements in computer language.

Example

- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

What is Insertion Sort?

Insertion Sort is a type of sorting algorithm.

Sorting Alogorithm

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements in computer language.

Example

- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

Sorting: Rocket Science or Human Nature?

(a) Not Sorted

(b) Sorted

We have been using **sorting** for our own convenience since primitive times without really knowing or realizing any computer science documented algorithms!

Background

Historical Overview

Insertion Sort's real-life application ranges from sorting cards (see 5) to students' exam scripts. It might be hard to find the first person who came up with the idea behind insertion sort, because it is one of the basic ways humans would sort a list of items.

Knuth [1] writes that the variant of using binary insertion was mentioned by John Mauchly as early as 1946, in the first published discussion of computer sorting, being the algorithm's first documentation.

John Mauchly

Background

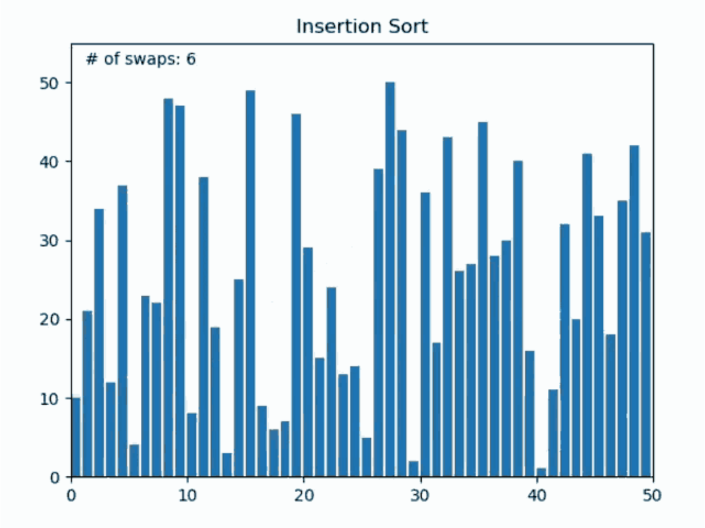
Historical Overview

Insertion Sort's real-life application ranges from sorting cards (see 5) to students' exam scripts. It might be hard to find the first person who came up with the idea behind insertion sort, because it is one of the basic ways humans would sort a list of items.

Knuth [1] writes that the variant of using binary insertion was mentioned by John Mauchly as early as 1946, in the first published discussion of computer sorting, being the algorithm's first documentation.

John Mauchly

Visual Representation



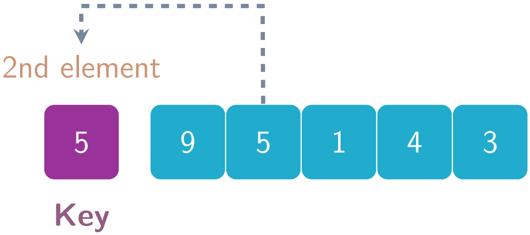
Steps in Illustration

9	5	1	4	3

Initial Array

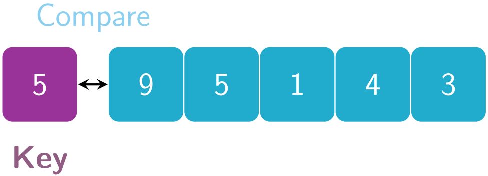
Steps in Illustration

Step 1:



Steps in Illustration

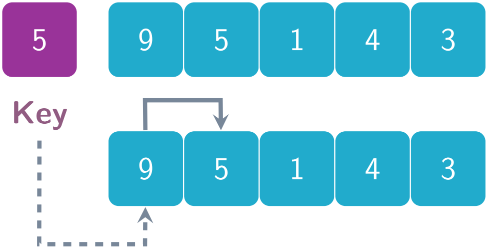
Step 1:



$$5 < 9$$

Steps in Illustration

Step 1:



Steps in Illustration

Step 1:

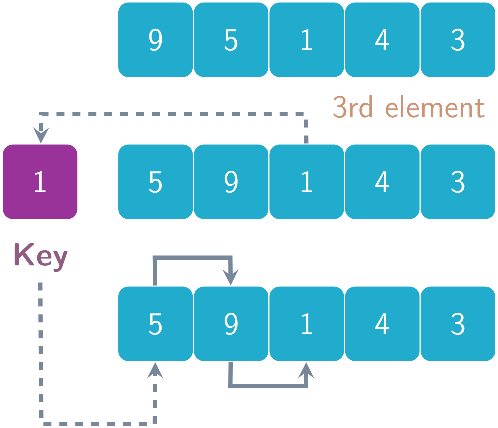
9	5	1	4	3

5	9	1	4	3

Places 5 at the begining

Steps in Illustration

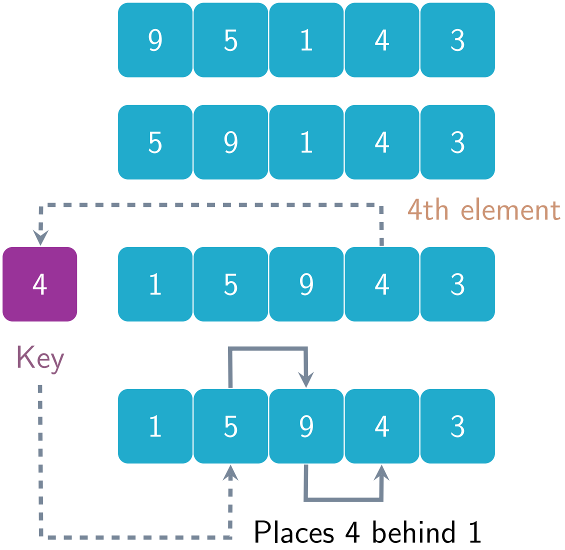
Step 2:



Places 1 at the begining

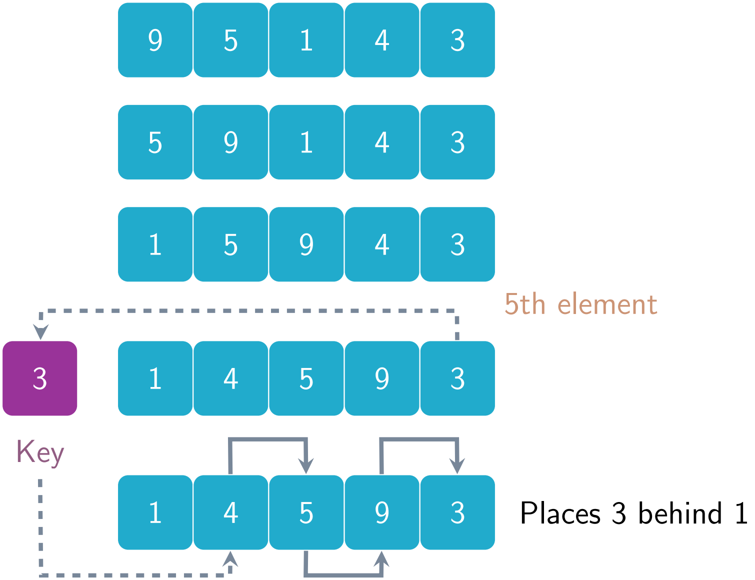
Steps in Illustration

Step 3:



Steps in Illustration

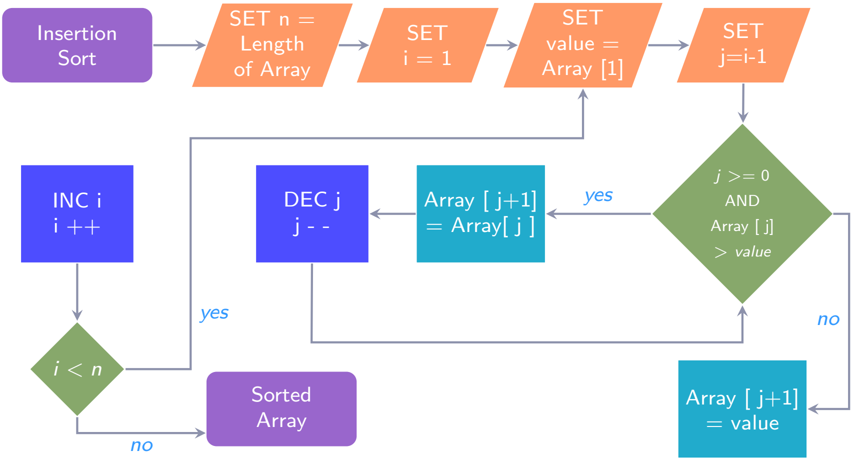
Step 4:



Steps in Illustration

	9	5	1	4	3	*Initial Array*
	5	9	1	4	3	*step 1*
	1	5	9	4	3	*step 2*
	1	4	5	9	3	*step 3*
Sorted Array	1	3	4	5	9	*step 4*

Flowchart


Insertion Sort
SET n = Length of Array
SET i = 1
SET value = Array [1]
SET j=i-1
j >= 0 AND Array [j] > value
yes
Array [j+1] = Array[j]
DEC j
j - -
no
Array [j+1] = value
INC i
i ++
i < n
yes
no
Sorted Array

Algorithm

Algorithm 1 Insertion algorithm

```
procedure INSERTIONSORT(A : array)
    n ← length(A)
    for i:=1 to n-1 do
        j ← i
        while j > 0 and A[j-1] > A[j] do
            swap(A[j], A[j-1])
            j ← j − 1
        end while                                ▷ inner loop end
    end for                                      ▷ outer loop end
end procedure
```

Complete Program in Python

```python
# Insertion sort in Python
def insertionSort(array):

    for step in range(1, len(array)):
        key = array[step]
        j = step - 1

        # Compare key with each element on the left of it until an element smaller than it is found
        while j >= 0 and key < array[j]:
            array[j + 1] = array[j]
            j = j - 1

        # Place key after the element just smaller than it
        array[j + 1] = key

data = [9, 5, 1, 4, 3]
insertionSort(data)
print('Sorted Array in Ascending Order:')
print(data)
```

See [2]

Best Case

The best case of Insertion Sort occurs **if the array is already sorted.** When $j = i - 1$, we always find the key $A[i]$ the first time.
Therefore, the running time of an algorithm equation-

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{2 \le j \le n} (1)$$
$$+ c_6 \sum_{2 \le n} (1-1) + c_7 \sum_{2 \le n} (1-1) + c_8(n-1)$$

simplifies as,

$$T(n) = an + b = \mathcal{O}(n)$$

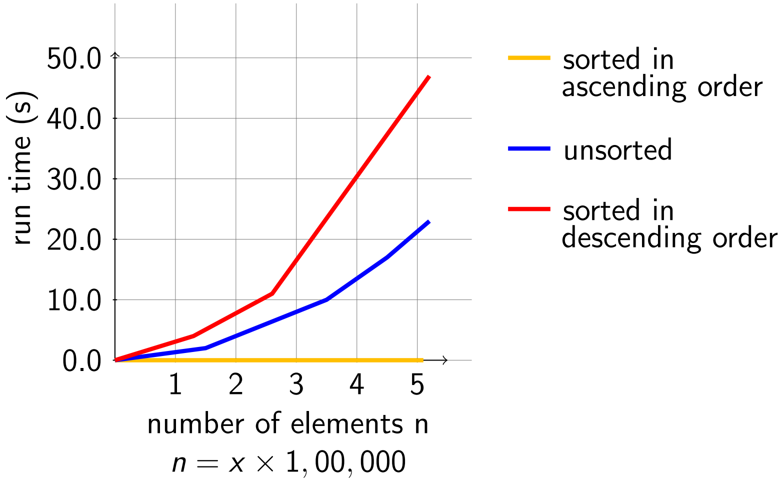
Worst Case

The worst case occurs **if the array is in reverse order of how we need to sort it.** So, we must compare each element $A[j]$ with element in the entire sorted subarray $A[i..j-1]$.
The running term hence simplifies as,

$$T(n) = an^2 + bn + c = \mathcal{O}(n^2)$$

Since we focus more on worst case, the complexity of insertion sort is said to be n^2.

Complexity Increase by Element



Complexity Overview

We can compare different sorting algorithm complexities with insertion sort from the given table.

Sorting Algorithm	**Time Complexity**			**Space Complexity**
	Best Case	**Average Case**	**Worst Case**	**Worst Case**
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Quick Sort	$\Omega(Nlog_2 N)$	$\Theta(Nlog_2 N)$	$O(N^2)$	$O(N)$
Merge Sort	$\Omega(Nlog_2 N)$	$\Theta(Nlog_2 N)$	$O(Nlog_2 N)$	$O(N)$
Heap Sort	$\Omega(Nlog_2 N)$	$\Theta(Nlog_2 N)$	$O(Nlog_2 N)$	$O(1)$

Time and Space Complexity of Sorting Algorithms

References

[1] E Knuth Donald et al. "The art of computer programming". In: *Sorting and searching* 3 (1999), pp. 426–458.

[2] *Programiz*. URL: https://www.programiz.com/dsa/insertion-sort.