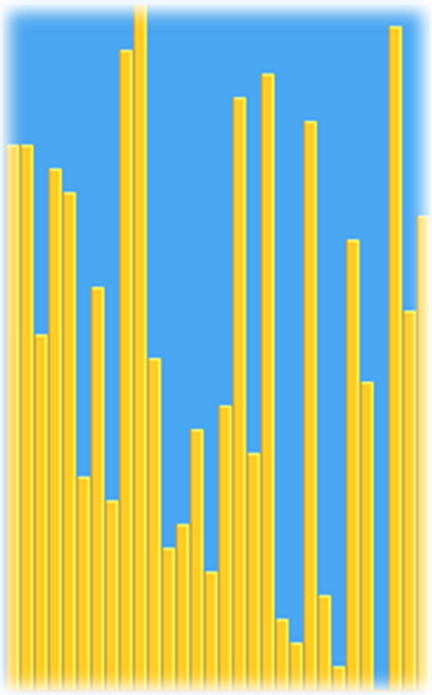


Insertion Sort User Manual

North Western University, Khulna

Insertion Sort User Manual

Table of contents

1. Introduction

Insertion sort is a simple and efficient comparison sort.

In this article, we will discuss the Insertion sort Algorithm. The working procedure of insertion sort is also simple. This article will be very helpful and interesting to students as they might face insertion sort as a question in their examinations. So, it is important to discuss the topic.

Insertion sort works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

The same approach is applied in insertion sort. The idea behind the insertion sort is that first take one element, iterate it through the sorted array. Although it is simple to use, it is not appropriate for large data sets as the time complexity of insertion sort in the average case and worst case is $O(n^2)$, where n is the number of items. Insertion sort is less efficient than the other sorting algorithms like heap sort, quick sort, merge sort, etc.

2. Advantages:

- Simple implementation
- Efficient for small data
- Adaptive: If the input list is presorted [may not be completely] then insertions sort takes O(n+d), where d is the number of inversions.
- Practically more efficient than selection and bubble sorts, even though all of them have O(n2) worst case complexity.
- Stable: Maintains relative order of input data if the keys (temp variable) are same.
- In-place: It requires only a constant amount O (1) of additional memory space.
- Online: Insertion sort can sort the list as it receives it.

3. Insertion Sort Algorithm:

The simple steps of achieving the insertion sort are listed as follows –

Step 1 - If the element is the first element, assume that it is already sorted. Return 1.
Step 2 - Pick the next element, and store it separately in a key.
Step3 - Now, compare the key with all elements in the sorted array.
Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
Step 5 - Insert the value.
Step 6 - Repeat until the array is sorted.

4. Insertion sort complexity

Time Complexity:

Best Case	O(n)
Average Case	$O(n^2)$
Worst Case	$O(n^2)$

Best Case Complexity - It occurs when there is no sorting required, i.e. The array is already sorted. The best-case time complexity of insertion sort is O (n).

Average Case Complexity - It occurs when the array elements are in jumbled order that is not properly ascending and not properly descending. The average case time complexity of insertion sort is O (n^2).

Worst Case Complexity - It occurs when the array elements are required to be sorted in reverse order. That means suppose you have to sort the array elements in ascending order, but its elements are in descending order. The worst-case time complexity of insertion sort is $O(n^2)$.

5. Space Complexity

Space Complexity Stable

The space complexity of insertion sort is O(1). It is because, in insertion sort, an extra variable is required for swapping.

6. Working of Insertion sort Algorithm

To understand the working of the insertion sort algorithm, let's take an unsorted array. It will be easier to understand the insertion sort via an example.

Let the elements of array are –

12	31	25	8	32	17

Initially, the first two elements are compared in insertion sort.

12	**31**	25	8	32	17

Here, 31 is greater than 12. That means both elements are already in ascending order. So, for now, 12 is stored in a sorted sub-array

12	**31**	**25**	8	32	17

Now, move to the next two elements and compare them.
Here, 25 is smaller than 31. So, 31 is not at correct position. Now, swap 31 with 25. Along with swapping, insertion sort will also check it with all elements in the sorted array

12	**25**	31	8	32	17

For now, the sorted array has only one element, i.e. 12. So, 25 is greater than 12. Hence, the sorted array remains sorted after swapping.

12	25	**31**	**8**	32	17

Now, two elements in the sorted array are 12 and 25. Move forward to the next elements that are 31 and 8.

12	**25**	**8**	31	32	17

Both 31 and 8 are not sorted. So, swap them. After swapping, elements 25 and 8 are unsorted. So, swap them.

12	**8**	25	31	32	17

Now, elements 12 and 8 are unsorted. So, swap them too.

8	**12**	25	31	32	17

Now, the sorted array has three items that are 8, 12 and 25. Move to the **next** items that are 31 and 32.

8	12	25	**31**	**32**	17

Hence, they are already sorted. Now, the sorted array includes 8, 12, 25 and 31.

Move to the next elements that are 32 and 17.

8	12	25	31	**32**	**17**

17 is smaller than 32. So, swap them. Swapping makes 31 and 17 unsorted. So, swap them too.

8	12	25	**31**	**17**	32

Now, swapping makes 25 and 17 unsorted. So, perform swapping again.

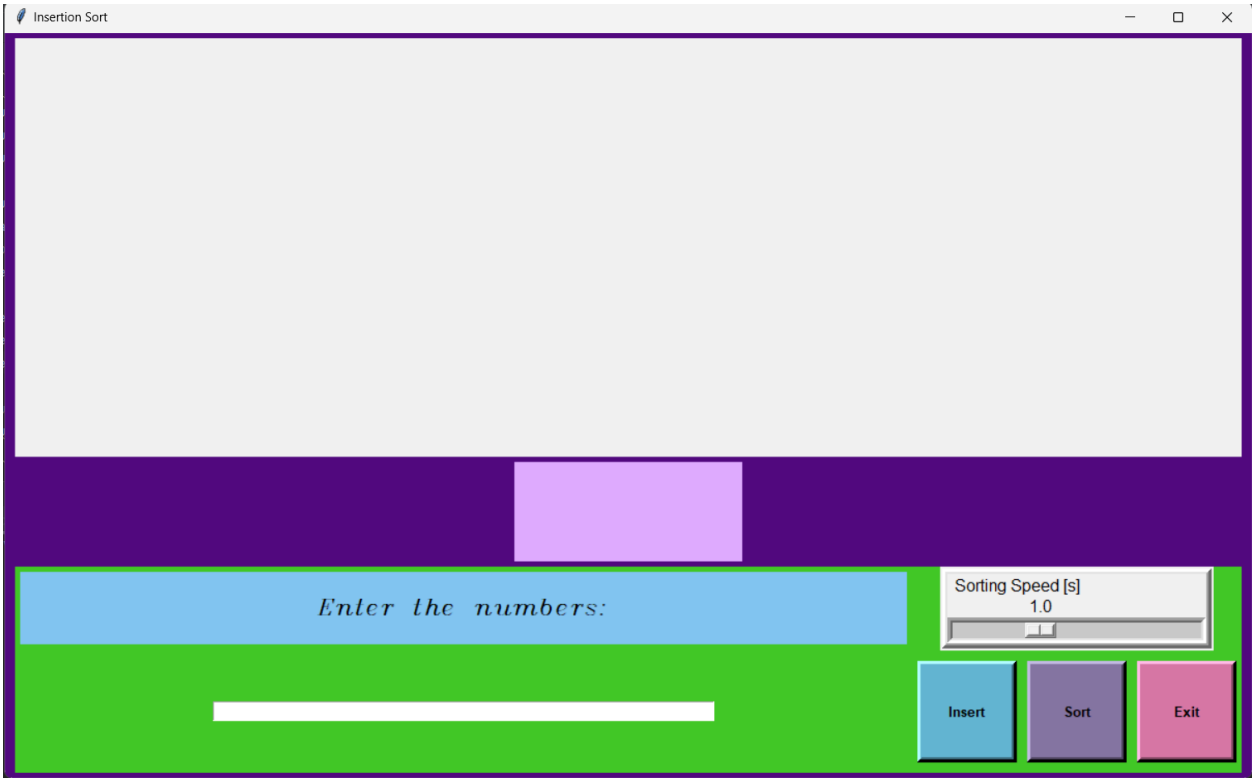
8	12	**25**	**17**	31	32

Now, the array is completely sorted.

8	12	17	25	31	32

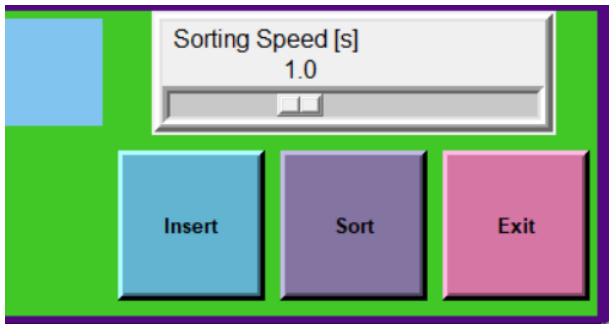
7. Visual diagram:

a. Homepage/ window

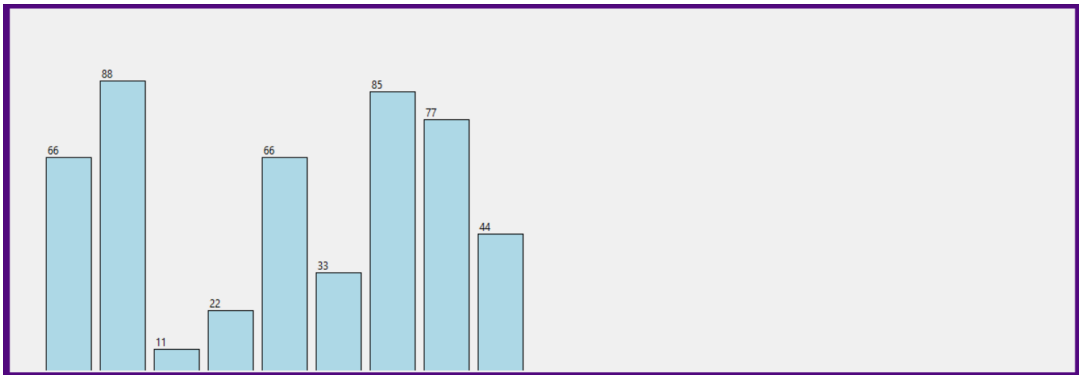


b. Insert the number

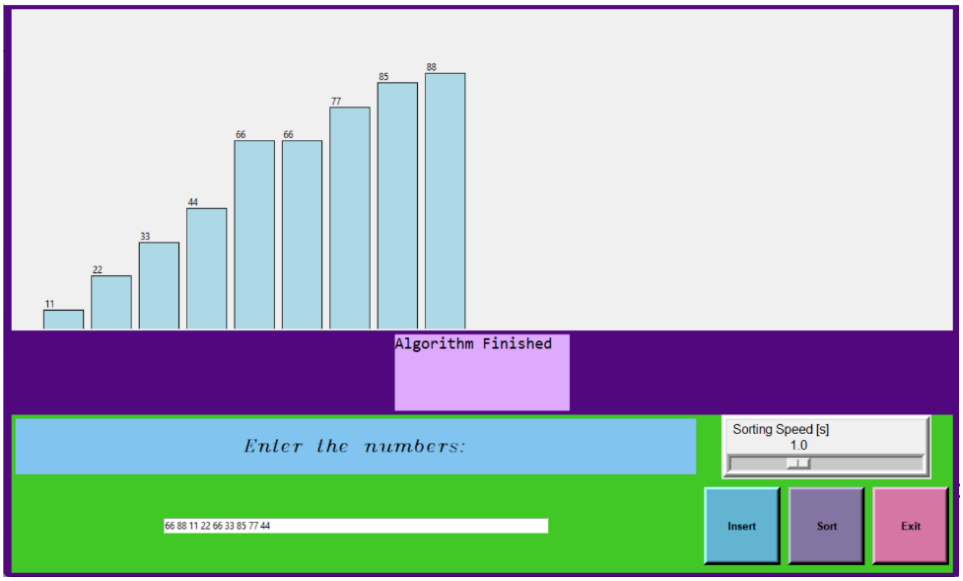
c . Insert input



d. Insert the value.



e. Sorting the value.



8. Implementation of insertion sort

Program: Write a program to implement insertion sort in python. With visualization.

```
from tkinter import *
import time
from tkinter import ttk, simpledialog

# The Insertion sort algorithm
def insertion_sort(arr, canvas):
    for i in range(1, len(arr)):
        j = i - 1
        key = arr[i]
        listbox.delete(0, END)
        listbox.insert(END, f"Iteration: {i}")
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
            listbox.delete(0, END)
            listbox.insert(END, f"Iteration: {i}")
            listbox.insert(END, f"Comparing: {arr[j + 1]} < {arr[j]}")
            drawArray(arr, canvas,
                      ["#110b38" if a == j or a == j + 1 else "#e06ca2" if a > j + 1 else "#ADD8E6" for a
in
                       range(len(arr))])
            time.sleep(speedScale.get() / 2)
        arr[j + 1] = key
        drawArray(arr, canvas, ['#ADD8E6' for x in range(len(arr))])
        time.sleep(speedScale.get() / 2)

    listbox.delete(0, END)
    listbox.insert(END, "Algorithm Finished")
    drawArray(arr, canvas, ['#ADD8E6' for x in range(len(arr))])


def sort():
    c = canvas
    a = input()
    insertion_sort(a, c)
    print(a)
```

```
# Draw array
# tithy
def drawArray(data, canvas, color):
    canvas.delete("all")
    c_height = 380
    c_width = 550
    x_width = c_width / (len(data) + 1)
    offset = 30
    spacing = 10
    bars = []
    normalizedData = [i / max(data) for i in data]  # normalizing the data to scale with the canvas
    for i, height in enumerate(normalizedData):  # Create boxes
        # top left

        x0 = i * x_width + offset + spacing
        y0 = c_height - height * 340
        # bottom right
        x1 = ((i + 1) * x_width) + offset
        y1 = c_height
        bar = canvas.create_rectangle(x0, y0, x1, y1, fill=color[i])
        canvas.create_text(x0 + 2, y0, anchor=SW, text=str(data[i]))  # Text over boxes
        bars.append(bar)
    window.update_idletasks()
    return bar


# input function
def input():
    input_str = inputBox.get()
    data = list(map(int, input_str.split()))

    return data


def generate():
    c = canvas
    data = input()
    drawArray(data, c, ['#ADD8E6' for x in range(len(data))])

    # edit_button = Button(frame, text='Edit', command=edit_data, bg='#D50032', fg='white',
font=('Arial Rounded MT Bold', 14), width=9, pady=5)
```

```
    # edit_button.grid(row=0, column=4, padx=5, pady=5)


# window
window = Tk()
window.title("Insertion Sort")
window.maxsize(1920, 1080)
window.config(bg="#3b20d6")

# UI
canvas = Canvas(window, width=1000, height=400)
canvas.grid(row=0, column=0, padx=10, pady=5)

listbox = Listbox(window, width=100, height=4)
listbox.grid(row=1, column=0)
listbox.config(font=('Consolas', 15), background="#6ec6ca", fg="white")
listbox.config(highlightthickness=0, borderwidth=0)

frame = Frame(window, width=800, height=300, bg='#41c726')
frame.grid(row=2, column=0, padx=10, pady=5)

# input data
l_label = Label(frame, text="Enter the numbers:", bg="#81c4f0", height=5,
              font=("italic", 14, 'bold'), width=50).grid(row=0, column=0, pady=5, padx=5)
inputBox = Entry(frame, width=100)
inputBox.config()
inputBox.grid(row=1, column=0, padx=5)

input_str = inputBox.get()
data = list(map(int, input_str.split()))
# speed
speedScale = Scale(frame, from_=0, to=3.0, length=250, digits=2, resolution=0.1,
orient=HORIZONTAL,
                   label="Sorting Speed [s]", font=('Cold Warm', 12))
speedScale.set(1.0)
speedScale.configure(sliderrelief="raised", activebackground='#ada8b6', relief="raised", bd=5)
speedScale.grid(row=0, column=1, padx=5, columnspan=3)

# buttons
g_button = Button(frame, text="Insert", bg='#62b4d1', width=10, height=5, relief="raised",
bd=5,
                  font=('arial', 10, 'bold'), command=generate).grid(row=1, column=1, pady=10,
padx=5)
```

```
s_button = Button(frame, text="Sort", bg='#8474a1', width=10, height=5, relief="raised", bd=5,
                  font=('arial', 10, 'bold'), command=sort).grid(row=1, column=2, pady=10, padx=5)

r_button = Button(frame, text="Exit", bg='#d676a4', width=10, height=5, relief="raised", bd=5,
                  font=('arial', 10, 'bold'), command=lambda: [window.destroy()]).grid(row=1,
column=3, pady=10, padx=5)

window.mainloop()
```

9. Dependencies:

1. Pycharm: For built this project we use Python language and Python Built in Packages. Install all the Packages which we have use. The name of the packages are given into ReadeMe.txt file. After Install all packages the project will run successfully.

2. Tkinter: We have used Python Tkinter Module for the User Interface. So, you must have to import the Tkinter Module for successfully executing this project

Special Thanks to:

Md. Shymon Islam

Lecturer

Department Of CSE

North Western University Khulna, Bangladesh

Developed by:

MD Abu Naeem

Student Id:20212016010

Alamin Sheikh

Student Id : 20212029010

Nusrat Jahan

Student Id: 20212014010

Department Of CSE

North Western University Khulna, Bangladesh