Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini Project Report on

"Merge Sort Visualization"

[Course Code: COMP 342]

Submitted By:

Gaurav Singh Thagunna (55)

Submitted To

Mr. Dhiraj Shrestha

Department of Computer Science and Engineering

Submission Date

December 14, 2020

OBJECTIVE - To write a program to visualize the merge sort algorithm

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. The merge() function is used for merging two halves. With worst-case time complexity being O(n log n), it is one of the most respected algorithms.

Steps for Merge Sort algorithm

1. Divide: partition the list into two roughly equal parts S1 and S2 called the left and right sublists.

2. Conquer: recursively sort S1 and S2

3. Combine: merge the sorted sublists.

Sourcecode

main.py

```
from tkinter import *
from tkinter import ttk
import random

from mergesort import merge_sort

root = Tk()
root.title('Sorting Algorithm Visualisation')
root.maxsize(900, 600)
root.config(bg='black')

#variables
selected_alg = StringVar()
data = []

#function
def drawData(data, colorArray):
    canvas.delete("all")
    c_height = 380
    c_width = 600
    x_width = c_width / (len(data) + 1)
    offset = 30
    spacing = 10
    normalizedData = [ i / max(data) for i in data]
    for i, height in enumerate(normalizedData):
        #top left
        x0 = i * x_width + offset + spacing
```

```
        y0 = c_height - height * 340
        #bottom right
        x1 = (i + 1) * x_width + offset
        y1 = c_height

        canvas.create_rectangle(x0, y0, x1, y1, fill=colorArray[i])
        canvas.create_text(x0+2, y0, anchor=SW, text=str(data[i]))

    root.update_idletasks()


def Generate():
    global data

    minVal = int(minEntry.get())
    maxVal = int(maxEntry.get())
    size = int(sizeEntry.get())

    data = []
    for _ in range(size):
        data.append(random.randrange(minVal, maxVal+1))

    drawData(data, ['red' for x in range(len(data))]) #['red', 'red' ,....]

def StartAlgorithm():
    global data
    if not data: return

    elif algMenu.get() == 'Merge Sort':
        merge_sort(data, drawData, speedScale.get())

    drawData(data, ['green' for x in range(len(data))])


#frame / base lauout
UI_frame = Frame(root, width= 600, height=200, bg='grey')
UI_frame.grid(row=0, column=0, padx=10, pady=5)

canvas = Canvas(root, width=600, height=380, bg='white')
canvas.grid(row=1, column=0, padx=10, pady=5)

#User Interface Area
#Row[0]
Label(UI_frame, text="Algorithm: ", bg='grey').grid(row=0, column=0, padx=5, p
ady=5, sticky=W)
```

```
algMenu = ttk.Combobox(UI_frame, textvariable=selected_alg, values=['Merge
Sort'])
algMenu.grid(row=0, column=1, padx=5, pady=5)
algMenu.current(0)

speedScale = Scale(UI_frame, from_=0.1, to=5.0, length=200, digits=2,
resolution=0.2, orient=HORIZONTAL, label="Select Speed [s]")
speedScale.grid(row=0, column=2, padx=5, pady=5)
Button(UI_frame, text="Start", command=StartAlgorithm, bg='red').grid(row=0,
column=3, padx=5, pady=5)

#Row[1]
sizeEntry = Scale(UI_frame, from_=3, to=25, resolution=1, orient=HORIZONTAL,
label="Data Size")
sizeEntry.grid(row=1, column=0, padx=5, pady=5)

minEntry = Scale(UI_frame, from_=0, to=10, resolution=1, orient=HORIZONTAL,
label="Min Value")
minEntry.grid(row=1, column=1, padx=5, pady=5)

maxEntry = Scale(UI_frame, from_=10, to=100, resolution=1, orient=HORIZONTAL,
label="Max Value")
maxEntry.grid(row=1, column=2, padx=5, pady=5)

Button(UI_frame, text="Generate", command=Generate, bg='white').grid(row=1,
column=3, padx=5, pady=5)

root.mainloop()
```

mergesort.py

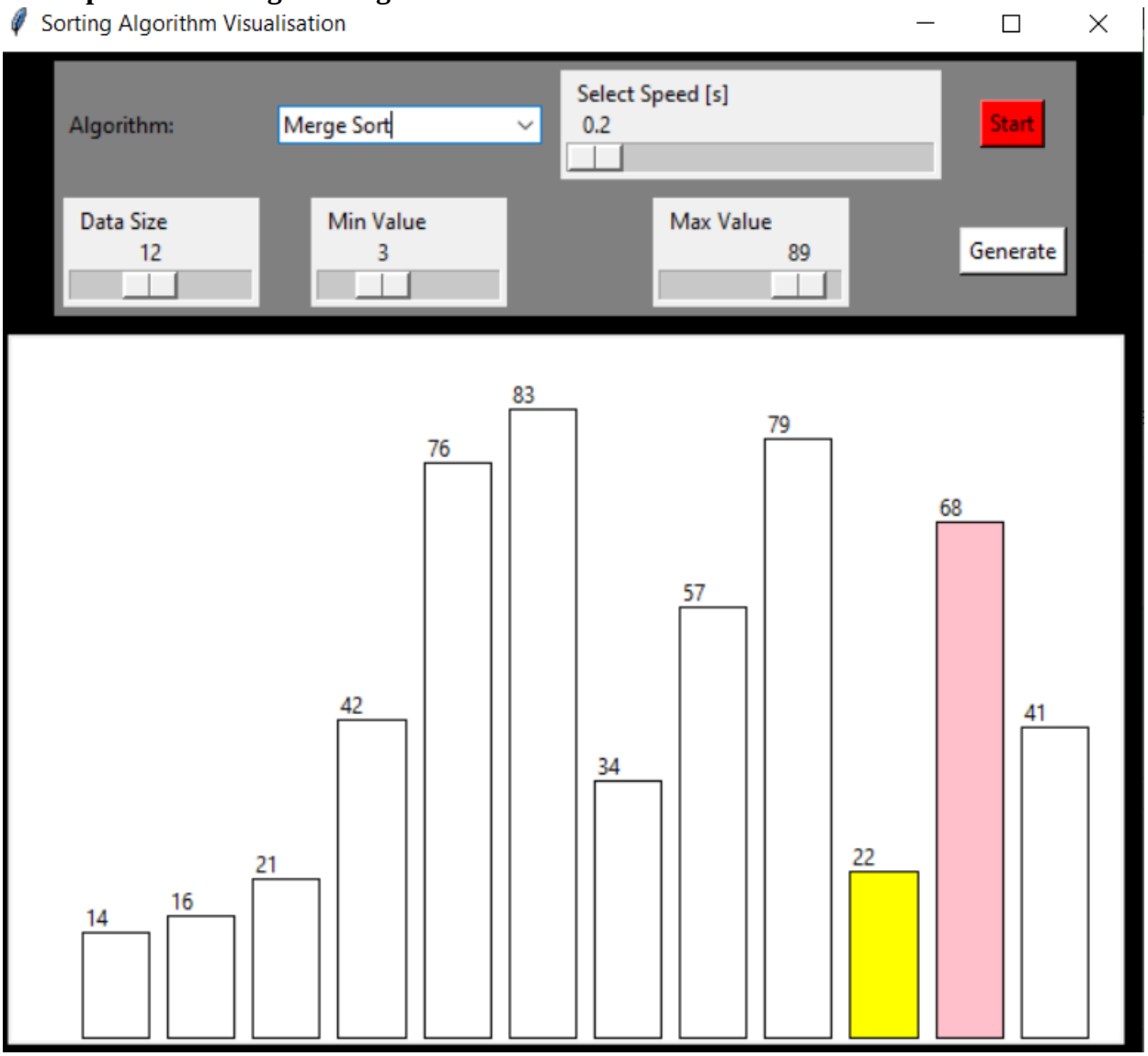
```
import time

def merge_sort(data, drawData, timeTick):
    merge_sort_alg(data,0, len(data)-1, drawData, timeTick)


def merge_sort_alg(data, left, right, drawData, timeTick):
    if left < right:
        middle = (left + right) // 2
        merge_sort_alg(data, left, middle, drawData, timeTick)
        merge_sort_alg(data, middle+1, right, drawData, timeTick)
        merge(data, left, middle, right, drawData, timeTick)
```

```
def merge(data, left, middle, right, drawData, timeTick):
    drawData(data, getColorArray(len(data), left, middle, right))
    time.sleep(timeTick)

    leftPart = data[left:middle+1]
    rightPart = data[middle+1: right+1]

    leftIdx = rightIdx = 0

    for dataIdx in range(left, right+1):
        if leftIdx < len(leftPart) and rightIdx < len(rightPart):
            if leftPart[leftIdx] <= rightPart[rightIdx]:
                data[dataIdx] = leftPart[leftIdx]
                leftIdx += 1
            else:
                data[dataIdx] = rightPart[rightIdx]
                rightIdx += 1

        elif leftIdx < len(leftPart):
            data[dataIdx] = leftPart[leftIdx]
            leftIdx += 1
        else:
            data[dataIdx] = rightPart[rightIdx]
            rightIdx += 1

    drawData(data, ["green" if x >= left and x <= right else "white" for x in
range(len(data))])
    time.sleep(timeTick)

def getColorArray(leght, left, middle, right):
    colorArray = []

    for i in range(leght):
        if i >= left and i <= right:
            if i >= left and i <= middle:
                colorArray.append("yellow")
            else:
                colorArray.append("pink")
        else:
            colorArray.append("white")

    return colorArray
```

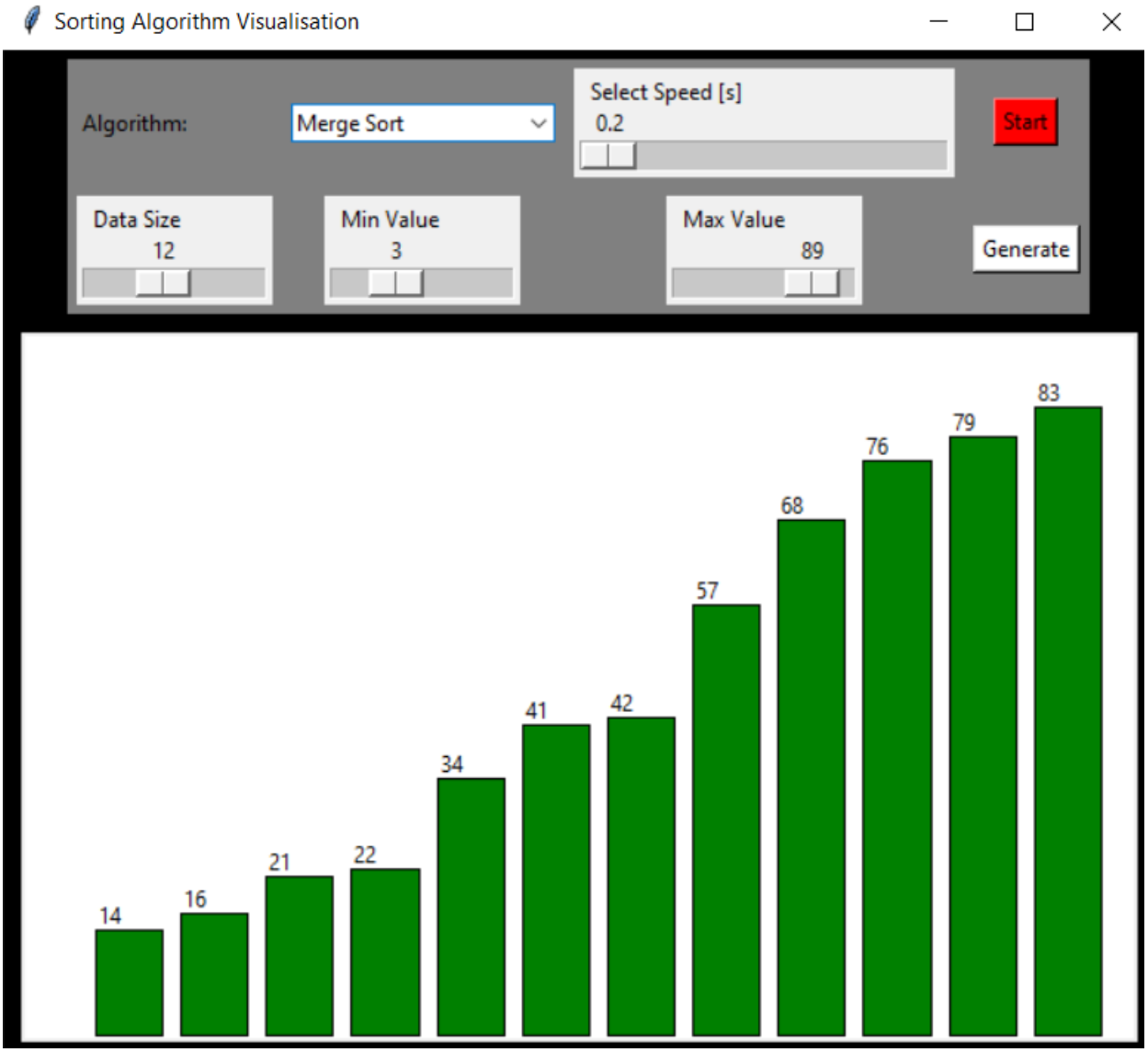
Output:

Before sorting

In the process of merge sorting



Final visualization after sorting



Conclusion: Thus, in this mini project tkinter package is used for building graphical user interface and after the completion of this program we can easily visualizing the merge sort algorithm**.**

githublink- click here