Mini Project On –

“Time Complexity Analyzer of Non- Recursive algorithm for C”

The Background:

In computer science, the time complexity is the computational complexity that measures or estimates the time taken for running an algorithm. Time complexity is commonly estimated by counting the number of elementary operations performed by the algorithm, supposing that an elementary operation takes a fixed amount of time to perform. Thus, the amount of time taken and the number of elementary operations performed by the algorithm differ by at most a constant factor.

Since an algorithm's running time may vary with different inputs of the same size, one commonly considers the worst-case time complexity, which is the maximum amount of time taken on inputs of a given size. Less common, and usually specified explicitly, is the average-case complexity, which is the average of the time taken on inputs of a given size (this makes sense, as there are only a finite number of possible inputs of a given size).

The Implementation

- For more easiness for the end users, we have used the library from python known as tkinter which provides marvellous tools for GUI
- The user just has to select the file from brose button and a dialogue will be displayed and after selection the time complexity of higher order will be shown
- As previously mentioned the GUI has been created with the help of tkinter library, some part of the file module has been implemented with the help of filedailog module
- One part of the module from this project required pattern matching to search/detect for the loops inside a c program, and hence it is done by re library offered by python which provides necessity tools for pattern matching by means of regular expression
- After the selection of a file various exception handling has been considered, whether user didn’t select any of the files, whether user would have selected irrelevant extension of the file like .java,.py. appropriate message has been generated to provide best form of user experience
- From our project inside, first all the loop is being detected, from a file, after that required time for that loop that how much time it will run its calculated and finally higher order has been displayed to the end user
- For storing contents and manipulation of computation, string and list are used

#Source code

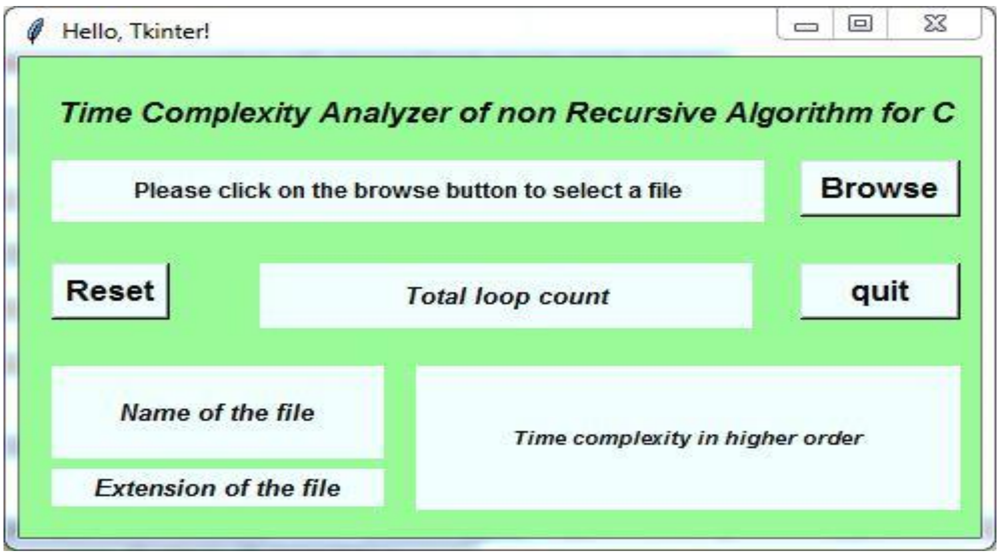
```
from tkinter import *
from tkinter import filedialog
import re
import os                          #importing all the required library
root = Tk()
root.title('Hello, Tkinter!')
root.geometry('480x280')
root.configure(bg='palegreen')     #creating the window of specified size and color

def resfun():                      #reset function
    pathlabel.config(text="Please click on the browse button to select a file")
    loopenclabel.config(text="Total loop count")
    fname.config(text="Name of the file")
    fext.config(text="Extension of the file")
    ftime.config(text="Time complexity in higher order")


def browsefunc():           #main module of the code
    try:
            absfile=""
            filename = filedialog.askopenfilename(initialdir=os.pardir,filetypes =(("C
            File", "*.c"),("All Files","*.*")),title = "Choose a file.")
            pathlabel.config(text="Full path = {}".format(filename))
            absfile=os.path.basename(filename)
            extension = os.path.splitext(filename)[1]
            fname.config(text="File name = {}".format(absfile))
            fext.config(text="File Extension = {}".format(extension))
            #getting filename extension and displaying

            s=""
            k=""
            l=""
            count=0

            file = open(filename,'r')
            for line in file.readlines():
                    if re.search('\s*for\s*\((.*)\)(.*)',line):
                            s=s+line
                    if re.search('\s*while\s*\((.*)\)(.*)',line):
                            s=s+line
                    if re.search('^\s*\{',line):
                            s=s+line
                    if re.search('\}',line):
                            s=s+line
                            #performing pattern match and storing is string s
```

```
#print(s)


k=s.split('\n')
        #splitting string with line by line
count1=0
for line1 in range(len(k)):
        if re.search('\s*for\s*\((.*)\)(.*)',k[line1]):
                #print(k[line1])
                count1=count1+1
        if re.search('\s*while\s*\((.*)\)(.*)',k[line1]):
                #print(k[line1])
                count1=count1+1
loopenclabel.config(text="total number of loop encountered =
{}".format(count1))    #for displaying total count of loop
#print(count1)


a=1
temp1=""
temp2=""
count=0

for text in range(len(k)):
        temp=k[text];
        for subtext in range(0,len(temp)):
                if(k[text][subtext] is '{'):
                        count=1
                if(k[text][subtext] is '}'):
                        count=0
                if(count!=0):
                        if(k[text][subtext]=='<' or k[text][subtext]=='>'):
                                temp1=temp1+k[text][subtext+a]
                                while(temp1 is ' ' or temp1 is '='):
                                        a=a+1
                                        temp1=k[text][subtext+a]
                                        if(temp1 is not ' ' and temp1 is not '='):
                                                break

#calculating and manipulating the time complexity

list1=list(temp1)                        #storing the result in list


for i in range(len(list1)-1):
        if(list1[i] is '='):
```

```
                    del(list1[i])           #correcting irrelevant complexity


        list1=list('*'.join(list1))
        temp2=''.join(list1)            #converting list to string

        ftime.config(text="Higher order time complexity =is {}".format(temp2))
        #displaying result
        #print(temp2)

        if(extension == '.java' or extension == '.py'): #if irrelevant extension is selected
            pathlabel.config(text=" file format not supported")
            loopenclabel.config(text="Nothing to show")
            fname.config(text="File name = {}".format(absfile))
            fext.config(text="File Extension = {}".format(extension))
            ftime.config(text="Unsupported file")
    except:                                     #if no file is selected
        pathlabel.config(text="No file chosen Please select a file")
        loopenclabel.config(text="Nothing to show")
        fname.config(text="Nothing selected")
        ftime.config(text="Nothing to display")
        fext.config(text="Nothing selected")


        #print("no file chosen")

titlelabel = Label(root,text="Time Complexity Analyzer of non Recursive Algorithm for C",fg='gray3' ,bg='palegreen',font=('Arial', 12, 'bold', 'italic'))
titlelabel.pack()
titlelabel.place(x=16,y=20)

pathlabel = Label(root,height=2,width=50,fg='gray3' ,bg='azure',font=('Arial', 9, 'bold'),text="Please click on the browse button to select a file")
pathlabel.pack()
pathlabel.place(x=16,y=60)

browsebutton = Button(root, text="Browse", width='7',command=browsefunc,fg='gray3' ,bg='azure',font=('Arial', 12, 'bold'))
browsebutton.pack()
browsebutton.place(x=390,y=60)

reset = Button(root, text="Reset", command=resfun,fg='gray3' ,bg='azure',font=('Arial', 12, 'bold'))
reset.pack()
reset.place(x=16,y=120)
```

```
quitw = Button(root, text="quit", width='7',command=quit,fg='gray3' ,bg='azure',font=('Arial', 12, 'bold'))
quitw.pack()
quitw.place(x=390,y=120)


loopenclabel = Label(root,height=2,width=30,fg='gray3' ,bg='azure',font=('Arial', 10, 'bold', 'italic'),text="Total loop count")
loopenclabel.pack()
loopenclabel.place(x=120,y=120)


fname = Label(root,height=3,width=20,fg='gray3' ,bg='azure',font=('Arial', 10, 'bold', 'italic'),text="Name of the file")
fname.pack()
fname.place(x=16,y=180)

fext = Label(root,height=1,width=20,fg='gray3' ,bg='azure',font=('Arial', 10, 'bold', 'italic'),text="Extension of the file")
fext.pack()
fext.place(x=16,y=240)

ftime = Label(root,height=6,width=38,fg='gray3' ,bg='azure',font=('Arial', 8, 'bold', 'italic'),text="Time complexity in higher order")
ftime.pack()
ftime.place(x=198,y=180)

mainloop()
```

#Output (Initialized window)


Hello, Tkinter!

Time Complexity Analyzer of non Recursive Algorithm for C

Please click on the browse button to select a file

Browse

Reset

Total loop count

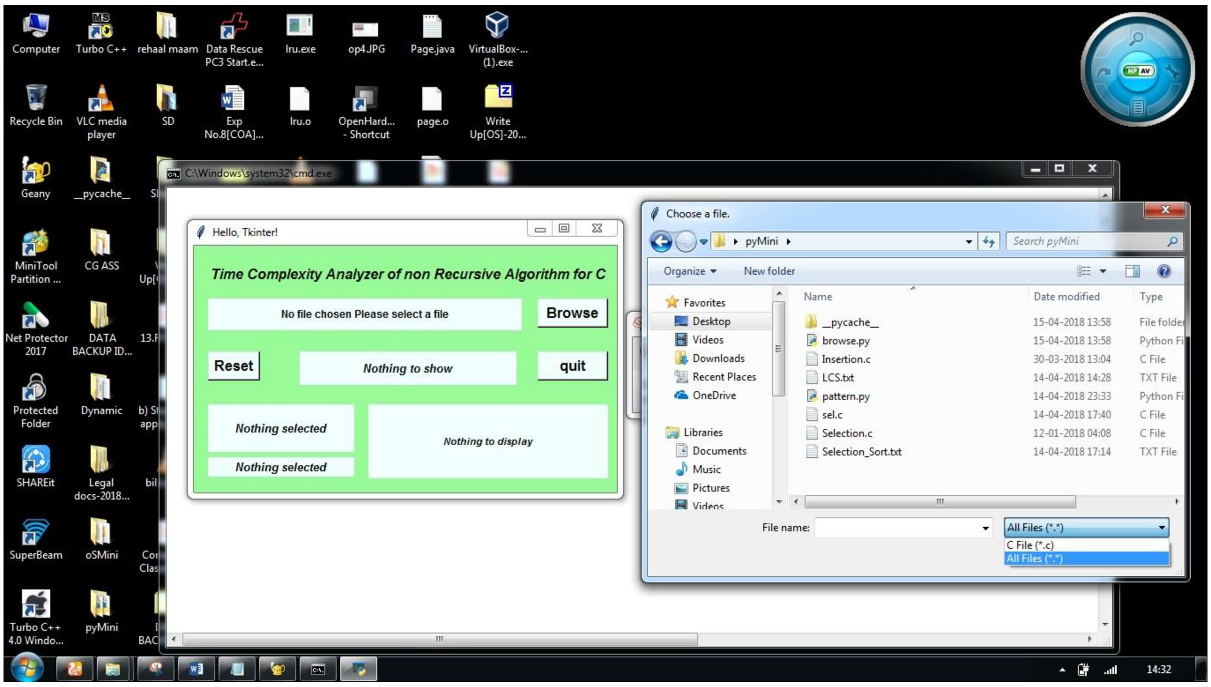
quit

Name of the file

Extension of the file

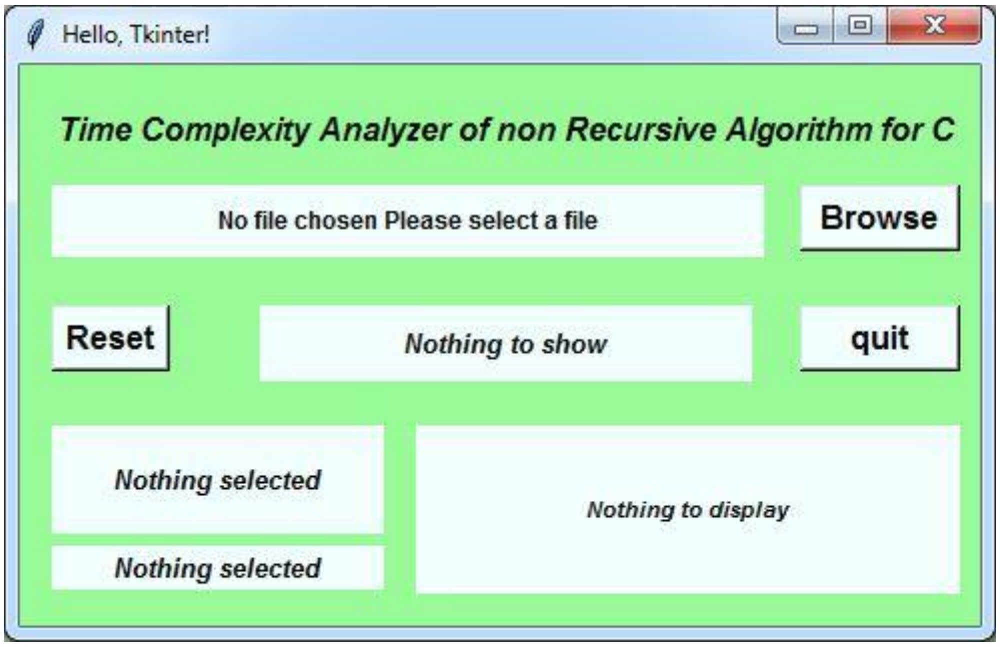
Time complexity in higher order

#Output (Browse)

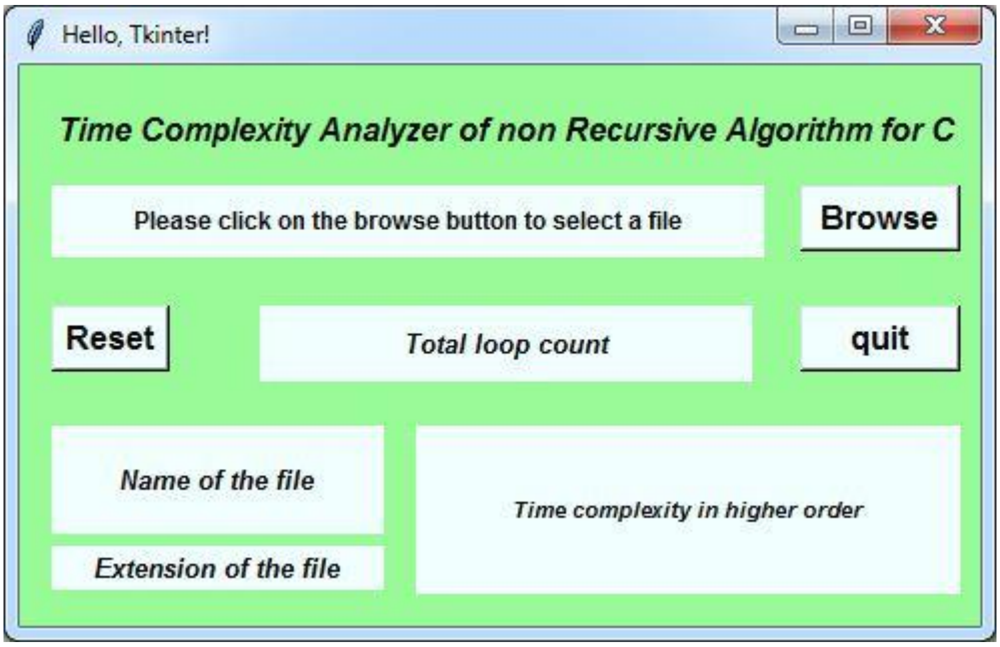


Name	Date modified	Type
__pycache__	15-04-2018 13:58	File folder
browse.py	15-04-2018 13:58	Python Fi
Insertion.c	30-03-2018 13:04	C File
LCS.txt	14-04-2018 14:28	TXT File
pattern.py	14-04-2018 23:33	Python Fi
sel.c	14-04-2018 17:40	C File
Selection.c	12-01-2018 04:08	C File
Selection_Sort.txt	14-04-2018 17:14	TXT File

#Output (No file chosen exception)



#Output (Reset)



Time Complexity Analyzer of non Recursive Algorithm for C

Please click on the browse button to select a file

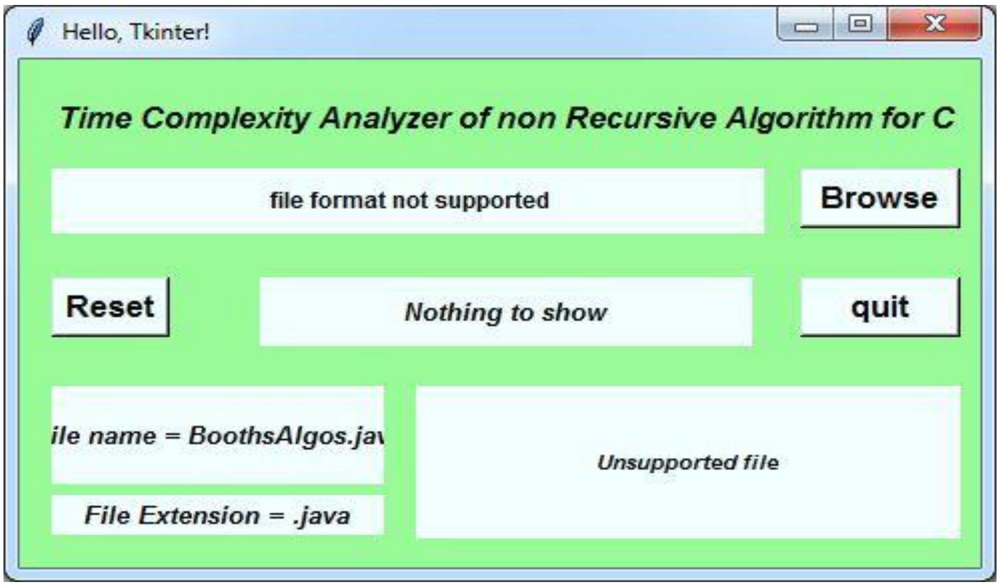
Total loop count

Name of the file

Extension of the file

Time complexity in higher order

#Output (Unsupported file)



Time Complexity Analyzer of non Recursive Algorithm for C

file format not supported

Nothing to show

ile name = BoothsAlgos.jav

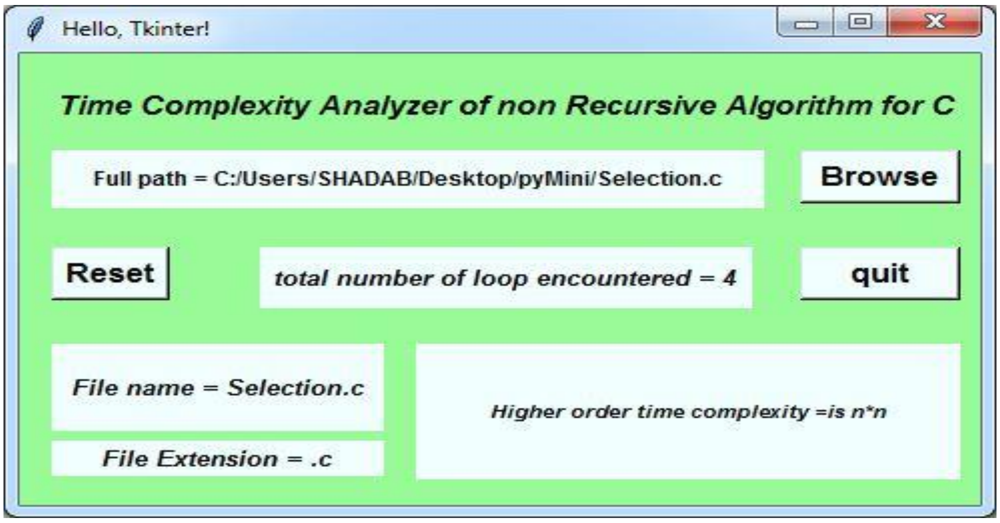
File Extension = .java

Unsupported file

Sample Code Of Selection.C

```
for(i=0;i<n;i++)
{
    min=i;                  //considering the minimum value
    for(j=i+1;j<n;j++)
    {
            if(a[min]>a[j]){
            min=j; //actual minimum value found
    }
}
```

#Output (Time complexity for file Selection.c)



Time Complexity Analyzer of non Recursive Algorithm for C

Full path = C:/Users/SHADAB/Desktop/pyMini/Selection.c

total number of loop encountered = 4

File name = Selection.c

File Extension = .c

*Higher order time complexity =is n*n*

#Output (quit)

```
------------------
(program exited with code: 0)
Press any key to continue . . .
```