

**To design and implement all searching algorithms using visualization in Python**

**INT 213 report**

**Submitted by**

Aman Kumar (12106661)

&

Sk Afridi (12106772)

## B-Tech CSE (3rd semester)

**School of Computer Science and Engineering**

Submitted on: 11 November 2022

**Index**

**Topic Page No.**

**Introduction 3**

**Abstract 3**

**Scope 4**

**Design 5-7**

**Algorithms 8**

**Creating the Visualizer 9-11**

**Flow Chart 12**

**Screenshots 13**

**Conclusion 14**

**References 14**

**GitHub Repository 14**

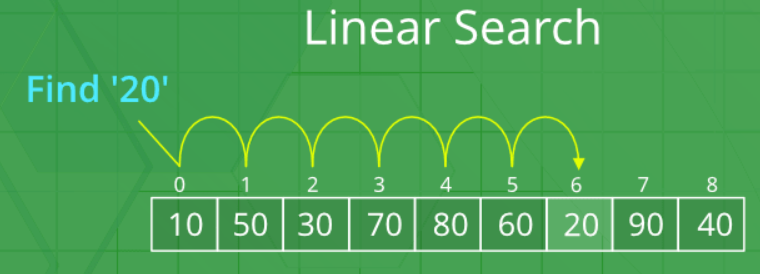
**Introduction**

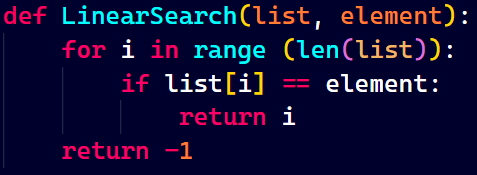
Searching for data stored in different data structures is a crucial part of pretty much every single application. There are many different algorithms available to utilize when searching, and each have different implementations and rely on different data structures to get the job done.

Being able to choose a specific algorithm for a given task is a key skill for developers and can mean the difference between a fast, reliable and stable application.

**Abstract**

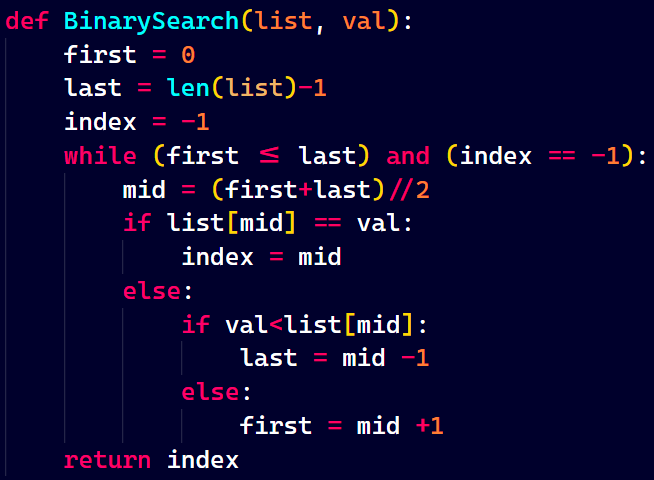
**Linear Search**

Linear search is one of the simplest searching algorithms, and the easiest to understand. The algorithm consists of iterating over an array and returning the index of the first occurrence of an item once it is found:



**Binary Search**

Binary search follows a divide and conquer methodology. It is faster than linear search but requires that the array be sorted before the algorithm is executed.



## Scope

Project Objectives: The main objective of this project is to visualize how linear and binary search work on lists in python with help of GUI.

Resources:

(i) Team of 2 people

(ii) We have mainly used tkinter and **PIL** to develop this GUI interface. The tkinter package

(“Tk interface”) is the standard python interface to the Tcl/Tk GUI toolkit.

(iii) We have also used the messagebox module. It is used to display message

boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.



(iv) For loading images into our GUI, we have used the Image and Imagetk modules. They can be imported from the Pillow library which we have to download by ourselves from the command prompt or on the terminal of our code editor itself.

Command to put into the cmd: **pip install pillow**

(v) Lastly, we have imported the time module which allows us to handle various operations regarding time, its conversions and representations.

## Design

# **Creating GUI**

* From tkinter import \* from tells the compiler to import something. Tkinter is a library in python and \* means to import everything.
* window = Tk() To initialize tkinter, we have to create a Tk window widget, which is a window with a title bar and other decoration provided by the window manager.
* window.mainloop() is simply a method in the main window that executes what we wish to execute in an application. As the name implies it will loop forever until the user exits the window or waits for any events from the user.
* window.geometry( “900x700”) sets the default size of the GUI window to 900 by 700
* window.maxsize(900,700) sets the maximum size that can be given to the GUI window to 900 by 700
* window.minsize(900,700) sets the minimum size that can be given to the GUI window to 900 by 700
* window.title(“Searching Algorithms Using Visualizer”) sets the title of the GUI window in the title bar to “Searching Algorithms Using Visualizer”.
* window.config(bg = “white”) sets the background color of our GUI interface to white.

# **Widgets Used**

* Label: The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
* Button: The Button widget is used to display buttons in your application.
* Entry: The Entry widget is used to display a single-line text field for accepting values

from a user.

* Text: The Text widget is used to display text in multiple lines.
* Messagebox: This module is used to display message boxes in your applications.
* PhotoImage: This widget is used to display an image for a Label or a Button.

# **Geometry Management**

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

* The pack() Method: This geometry manager organizes widgets in blocks before placing them in the parent widget.
* The grid() method: This geometry manager organizes widgets in a table-like structure in the parent widget.

# **Attributes used**

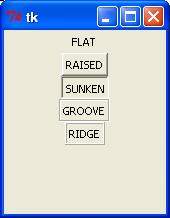
* + bg or background: Used to set the background color. It can hexadecimal value

octal value, rgb or some predefined color name.

* Fg or foreground: Used to set the foreground color.
* relief: The relief style of a widget refers to certain simulated 3-D effects around

the outside of the widget.The possible values which can be used for relief attribute

FLAT, RAISED, SUNKEN, GROOVE, RIDGE.



* Font Used to set the font size, style and formatting of the text. Ex: ("Helvetica",

"16", “bold”, “underline”) for a 16-point Helvetica regular. (Times 24 bold italic)

for a 24-point Times bold italic.

* borderwidth: Used to set the width of the border.
  + width: Used to set the width of a widget in tkinter.
  + height : Used to set the height of a widget in tkinter.

## **Algorithms**

## **Linear Search**

Linear Search(a, n, val)

// 'a' is the given array, 'n' is the size of given array, 'val' is the value to search

Step 1: set pos = -1

Step 2: set i = 1

Step 3: repeat step 4 while i <= n

Step 4: if a[i] == val

set pos = i

print pos

go to step 6

[end of if]

set ii = i + 1

[end of loop]

Step 5: if pos = -1

print "value is not present in the array "

[end of if]

Step 6: exit

Binary Search

Binary Search(a, lower\_bound, upper\_bound, val)

// 'a' is the given array, 'lower\_bound' is the index of the first array element, 'upper\_bound' is the index of the last array element, 'val' is the value to search

Step 1: set beg = lower\_bound, end = upper\_bound, pos = - 1

Step 2: repeat steps 3 and 4 while beg <=end

Step 3: set mid = (beg + end)/2

Step 4: if a[mid] = val

set pos = mid

print pos

go to step 6

else if a[mid] > val

set end = mid - 1

else

set beg = mid + 1

[end of if]

[end of loop]

Step 5: if pos = -1

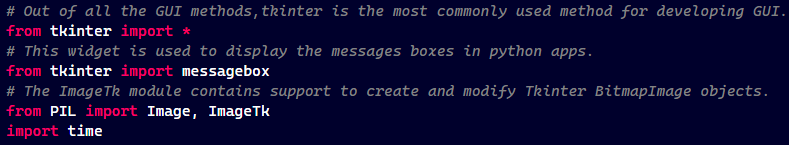
print "value is not present in the array"

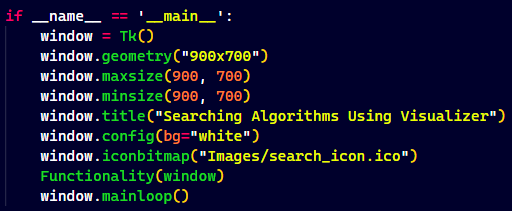
[end of if]

Step 6: exit

Creating the Visualizer

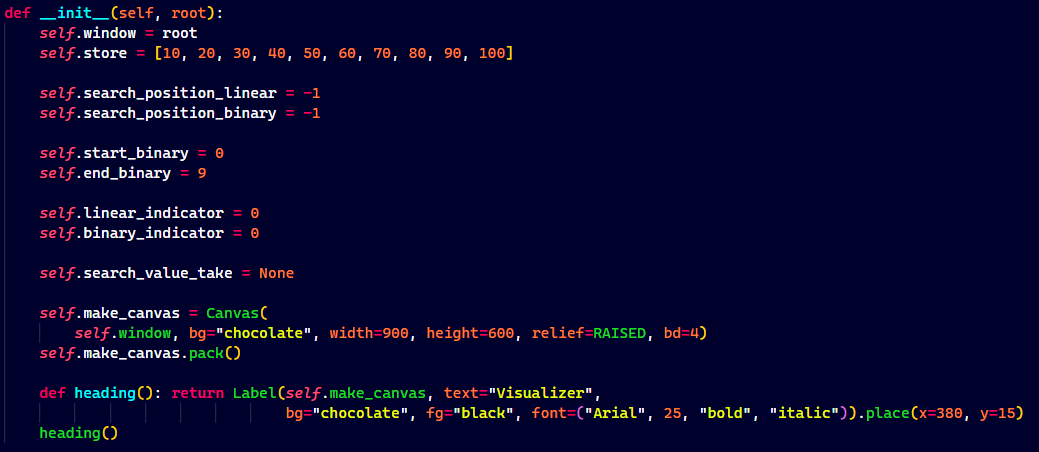
* Program home screen’s UI layout is implemented using the libraries mentioned in below screenshot. Functionality function has been passed window as an argument.



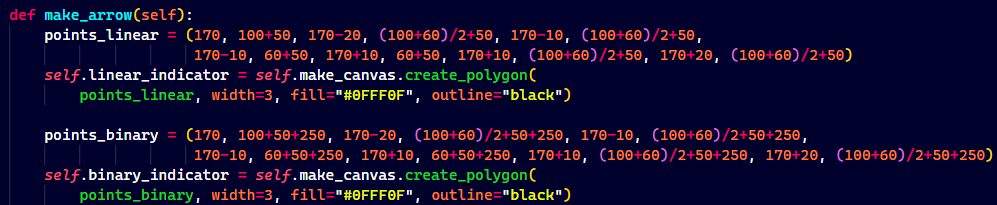


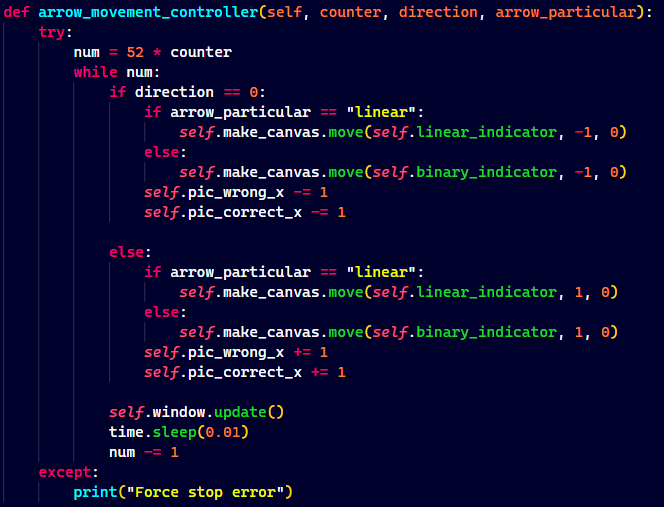
* This function initializes everything related to the program to run correctly and sets the canvas. It is member function of class named as Initialization.

There is a pre-defined list containing numbers from 10 to 100.

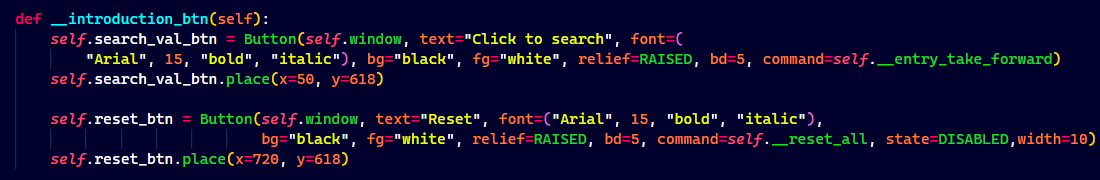


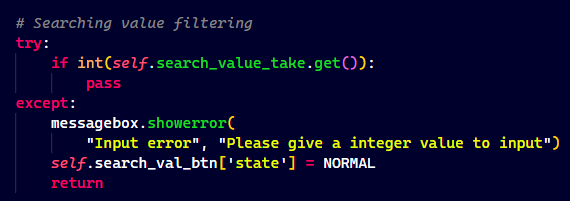
* This function makes the arrow and controls its movement while traversing the list.





* This function takes input from the user and gives control back to the arrow movement function



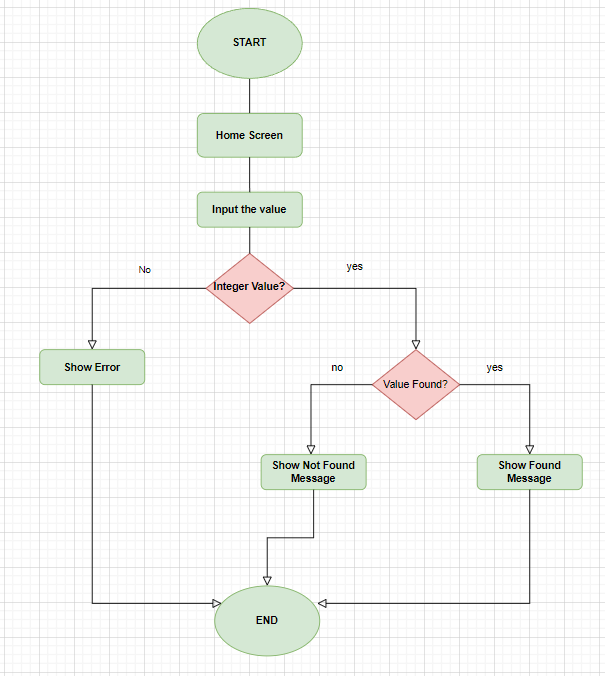
* Error handling is done by implementing this block of code. If the user enters value other than integer then an error message is shown to the user.

## Reset button functionality was implemented using the below mentioned code. It resets all the values to their default state so that program can be used for further test cases.

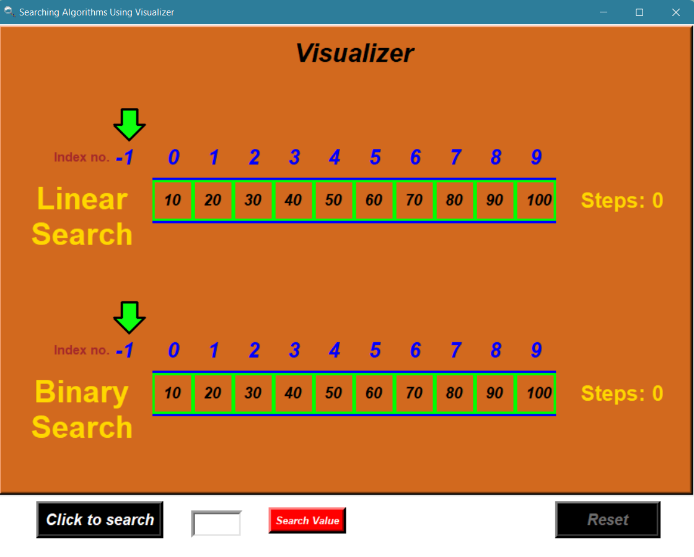
## 

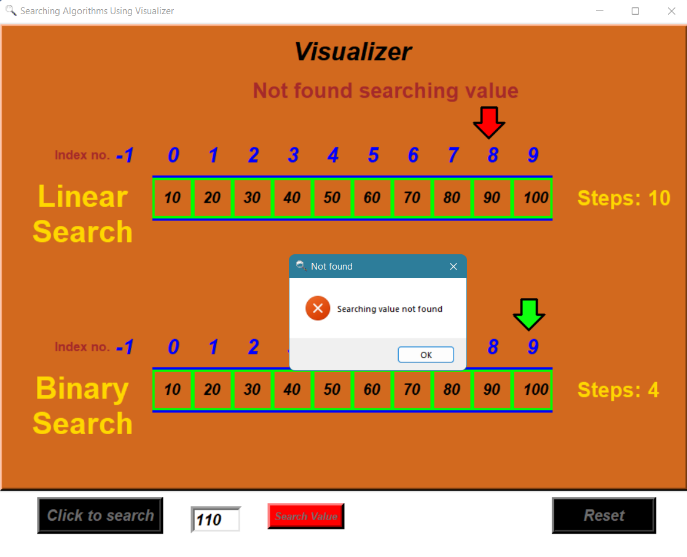
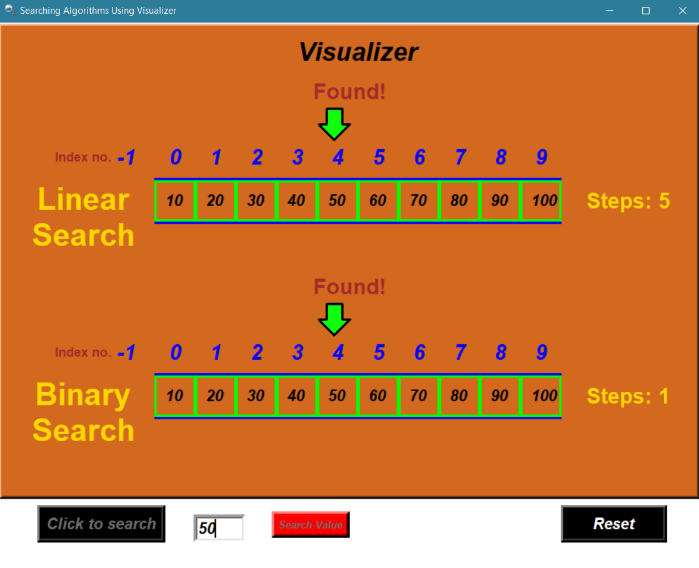
Flow Chart

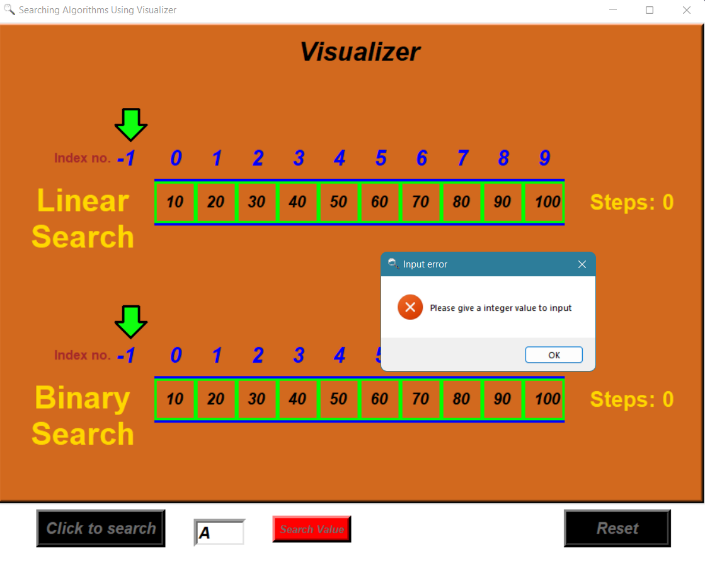
The flow-chart for the translator function is given below



**Screenshots**

****

****



Conclusion

The main objective of this project was to visualize the working of linear and binary search

by the help of GUI interface.

Using this project, someone who doesn’t have any prior idea about searching algorithms will get some idea about how it is done because the interface is user friendly and clearly shows every step before searching the key value that we enter into the interface.

We learn about the different workings of the tkinter module, how to use Label, PhotoImages, Canvas, MessageBoxes and add foreground and background colors, customize our text to the style we want and managing the geometry of our interface.

The program was able to handle all test cases. The project can therefore be judged a success. It fully achieved its goals within the time period provided.

**References**

* <https://stackoverflow.com/>
* <https://stackabuse.com/search-algorithms-in-python/>
* <https://www.geeksforgeeks.org/python-gui-tkinter/>
* <https://www.w3schools.in/python/gui-programming/>
* <https://docs.python.org/3/library/tk.html>
* <https://app.diagrams.net/>
* <https://www.javatpoint.com/binary-search>
* <https://www.javatpoint.com/linear-search>

**GitHub Repository**

* <https://github.com/AmanTheExecutable/Algorithm-Visualizer>

**THANK YOU**