
Lab Assignment 2: Binary Search

Name: **Arnav Gupta**

Utsa abcID : **Enp615**

Abstract: This lab is about writing a python program to perform binary search. I started with 1) setting up my developer environment. I installed VSCode, git, python3 and configured my VSCode environment with many suitable extensions like python, git, markdown etc. to enable me to make use of all SDLC best practices. futher I used python script that implements a recursive binary search algorithm to find a given value in a sorted integer array. for this I used python's in-built module 'array' which provides a space efficient homogenious storage for arrays.

File	Description
lab_2_binary-search.py	Contains the binary search functions

Execution Steps

```
PS C:\mysharedfolder\git\EE-3233-0AT-2025-SystemsProgramming> & c:/mysharedfolder/git/EE-3233-0AT-2025-SystemsProgramming/.venv/Scripts/python.
exe c:/mysharedfolder/git/EE-3233-0AT-2025-SystemsProgramming/lab_2/lab_2_binary-search.py
myarray:  array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
found 3 at index 3
3
found 7 at index 7
7
val: 11 not found
-1
val: 20 not found
-1
PS C:\mysharedfolder\git\EE-3233-0AT-2025-SystemsProgramming>
```

Conclusion: I have leant the flow of executing python code in VScode (an IDE environment), python in-built array module, writing and calling basic functions in python and recursion concepts. With this I am feeling motivated to contine on my path of continous leaning.

Source Code

EE-3233-0AT-2025-SystemsProgramming > lab_2 > lab_2_binary-search.py > ...

```python
#!/bin/user/env python3
import array as arr
##############################################
# Lab Assignment 2 - Binary Search
##############################################

##############################################
# function to search a val in an array between index:left and right
# my array shoudl be a sorted array
##############################################
def binary_search_recursive(myarray, val, left, right):
    if left > right:
        print("val: {0} not found".format(val))
        return -1

    nmid = (left+right)//2
    mid_val = myarray[nmid]
    #print("nmid: ", nmid, mid_val)

    if val == mid_val:
        #found
        print("found {0} at index {1}".format(val, nmid))
        return nmid
    elif val < mid_val :
        #search left tree
        right=nmid - 1
    elif val > mid_val:
        #right tree
        left= nmid + 1

    return binary_search_recursive(myarray, val, left, right)


##############################################
# function to search a val in an array using binary search method
# my array shoudl be a sorted array
##############################################
def binary_search(myarray, val):
    left=0
    right=len(myarray) - 1
    return binary_search_recursive(myarray, val, left, right)


if __name__ == '__main__':
    myarray = arr.array('i',[n for n in range(10)]) # this will make an array
    # val=10
    print("myarray: ", myarray)
    print(binary_search(myarray, 3))
    print(binary_search(myarray, 7))
    print(binary_search(myarray, 11))
    print(binary_search(myarray, 20))
```

Annexure - Assignment Details

EE-323:Systems Programming for Engineers

Teaching/Lab Assistant: Kriza Baby

kriza.baby@utsa.edu

Lab Student Hour: Friday 11:00am to 12:00pm

Lab Assignment 2

In this assignment your task is to implement a function to do binary search in a sorted array. This is more of a practice assignment for you to start coding with Python3.

Binary Search

Binary search is a relatively quick algorithm to find an index within a sorted (non-decreasing) array, whose value is equal, or almost equal, to a specified value. Thus, you define the function as:

def binary_search(array, val):

Where array is an input array where you want to find the value val.

The idea behind binary search is that you will inspect the element in the middle of array by making use of two auxiliary indexes: left and right. If the value in such middle index equals to val:

if array[middle] == val:

then you just return middle.

If the value in the middle of the array is less than val, then you need to readjust right. Otherwise, if the value in the middle of the array is more than val, then you need to readjust left. If you keep repeating this process, in a while loop, then the window between left and right should start to shrink until the algorithm eventually finds the correct index middle. However, if the algorithm reaches a point where:

left >= right

You stop and return left. If the element is not present either print the element is not present or return -1.

Index

0 1 2 3 4

Value

4 7 11 12 40

```
                    Left          Middle          Right
```

Tips You may eventually want to look for an online tutorial if you are still confused. Don’t worry about it, the binary search algorithm is very popular and easy to implement. The idea of this assignment, however, is for you to familiarize with Python3.

One thing that you may want to consider is that you need to use the floor division operator to calculate the middle index within your while loop. Otherwise, middle may end up getting assigned a floating-point number that should not be used as an index.

Additionally, you can get the length of array as follows:

n = len(array)

Testing After defining the binary search function, use the following lines of code to test your implementation:

array = [n for n in range(10)]

print(binary_search(array, 3))

print(binary_search(array, 7))

print(binary_search(array, 11))

print(binary_search(array, 20))

Which prints 3, 7, -1, -1 on the terminal.

Deliverables Upload a unique PDF file with screenshots of your code and simulation output. Your PDF must include a simple header with your name and your abc123. Submit a file with your work even if you cannot replicate the output shown above.

Grading Full marks will be granted to students that replicate the output, or whose logic is correct.

-1 point if you don’t include the header with your name and abc123 in your file. -1 point if you only submit the image files to Canvas(.jpeg, .png, etc...) Submissions with links to screenshots uploaded to the internet will not be accepted. Make sure that both code and simulation output is attached. No late submissions will be accepted. Refer to the syllabus.