**EX NO: 5 LINEAR SEARCH AND BINARY SEARCH**

**Date :**

**AIM**

To write a python program for performing linear search and binary search for the given set of inputs

sample input 0: [5,4,0,29,2]

sample output 0:[0,2,4,5,29]

sample input 1: [5,-4,0,29,2]

sample output 1:[-4,0,2,5,29]

sample input 2: [5,4.3,0,29,4.2]

sample output 2:[0,4.2,4.3,5,29]

sample input 2: ['b','a','c']

sample output 2:['a','b','c']

**ALGORITHM**

**Linear search:**

**Step1:** Start with the first item in the list

**Step2:**  Compare the current item to the key

**Step 3:** If the current value matches the key then update with the location

**Step 4:** Else repeat from 2.

**Binary search:**

**Step1:** Choose the middle element in the list

**Step2:**  If it matches the middle element, its position in the list is returned.

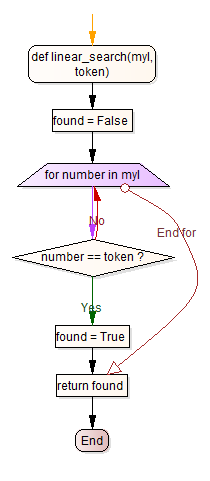
**Step 3:** If the target value is less than or greater than the middle element, the search

continues in the lower or upper half of the array, respectively, eliminating the

other half from consideration

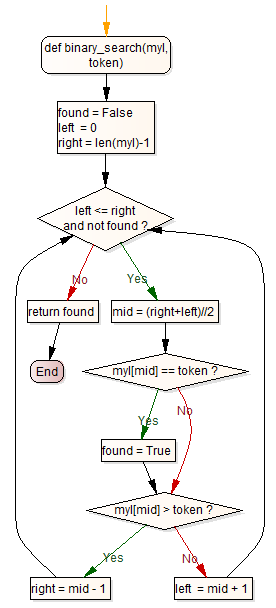
**FLOWCHART**

**Linear search:**



**No**

**Binary search:**

****

**PRE-LAB QUESTIONS**

1. What is the output of selection sort after the 1st iteration given the following sequence of numbers: 14 9 4 18 45 2 37 6
2. What is the worst case complexity for selection sort algorithm?
3. What is the average case complexity for selection sort algorithm?
4. What is the output of selection sort after the 2nd iteration given the following sequence of numbers: 16 3 46 9 28 14?
5. What is the best case complexity for selection sort algorithm?
6. In a selection sort of n elements, how many times is the swap function called in the complete execution of the algorithm?
7. A sorting technique in which successive elements are selected in order and placed into their proper sorted positions is called?
8. In which cases are the time complexities same in selection sort?

**SOURCE CODE**

**Linear search**

def linear\_search(myl, token):

found = False

for number in myl:

if number == token:

found = True

break

return found

In pythonic way:

def linear\_search(mylist, token):

return token in mylist

**Binary search using iteration**

def binary\_search(myl, token):

found = False

left = 0

right = len(myl)-1

while left <= right and not found:

mid = (right+left)//2

if myl[mid] == token:

found = True

if myl[mid] > token:

right = mid - 1

else:

left = mid + 1

return found

**bsearch using recursion**

def bsearch(mylist, token):

if not mylist:

return False

mid = len(mylist)//2

if mylist[mid] is token:

return True

elif mylist[mid] > token:

return bsearch(mylist[:mid], token)

else:

return bsearch(mylist[mid+1:], token)

**OUTPUT**

Enter List: 5,4,0,29,2

Enter Number:5

Output : 0,2,4,5,29

**RESULT**

Thus the python program for performing the linear and binary search was executed and verified successfully.

**POST-LAB QUESTIONS**

1. How to remove the duplicates from the resultant array?
2. Rewrite the selection sort code above to sort in ascending order