Assignment

Code Link: Click Here

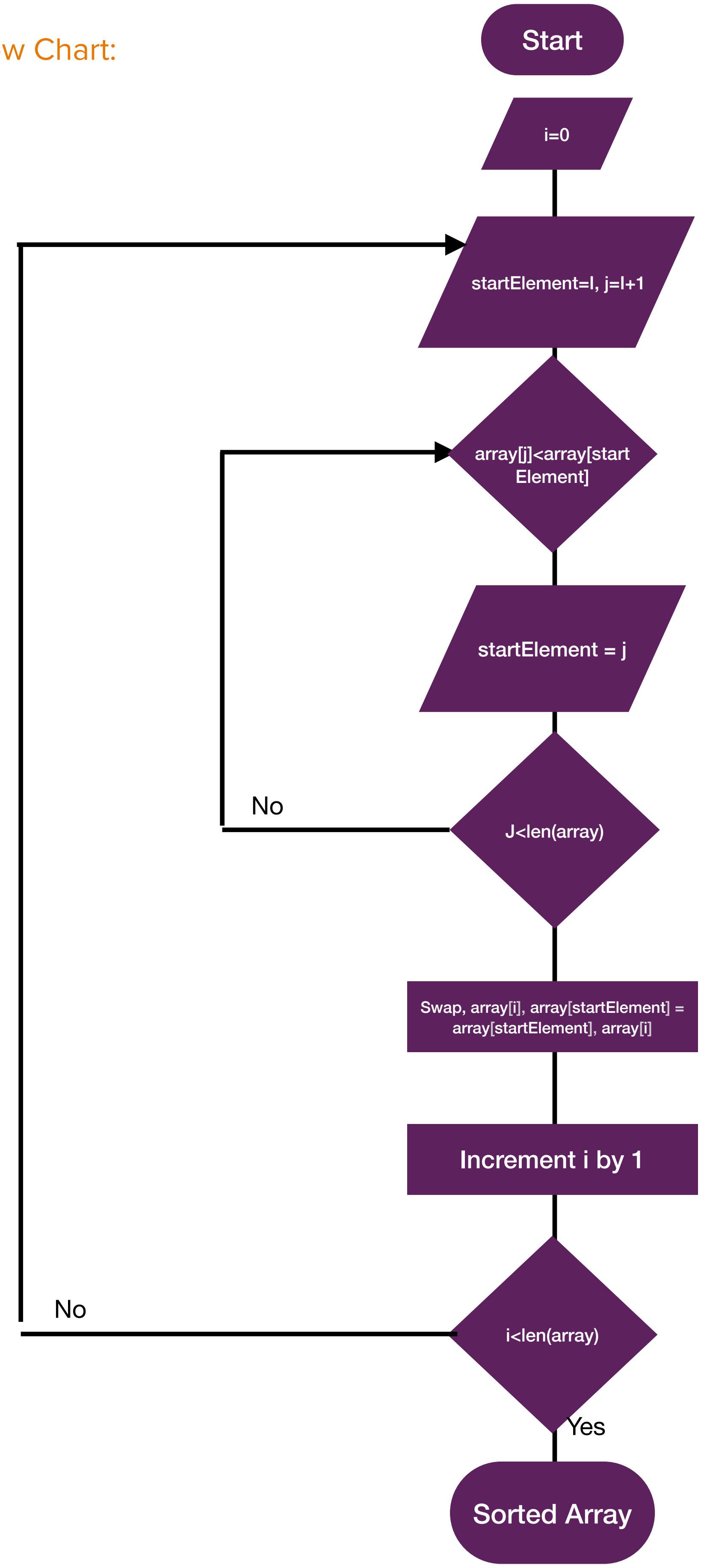
If the above link does not work
Google Colab Link:
https://colab.research.google.com/drive/1ohioCuvQlJADrtwq2m9zOBDT7SY7kRFw?usp=sharing

Selection Sort:

Algorithm:

Step 1: Start
Step 2 : Take the array and consider its left most element as its minimum
Step 3: Compare it to the next element and if the next element is smaller, then update the minimum.
Step 4: Once reached the end of the unsorted array swap the minimum with the first unsorted element.
Step 5: As the first element in the array is sorted move the start point to the next of it.
Step 6: Repeat Steps 2-5

Flow Chart:

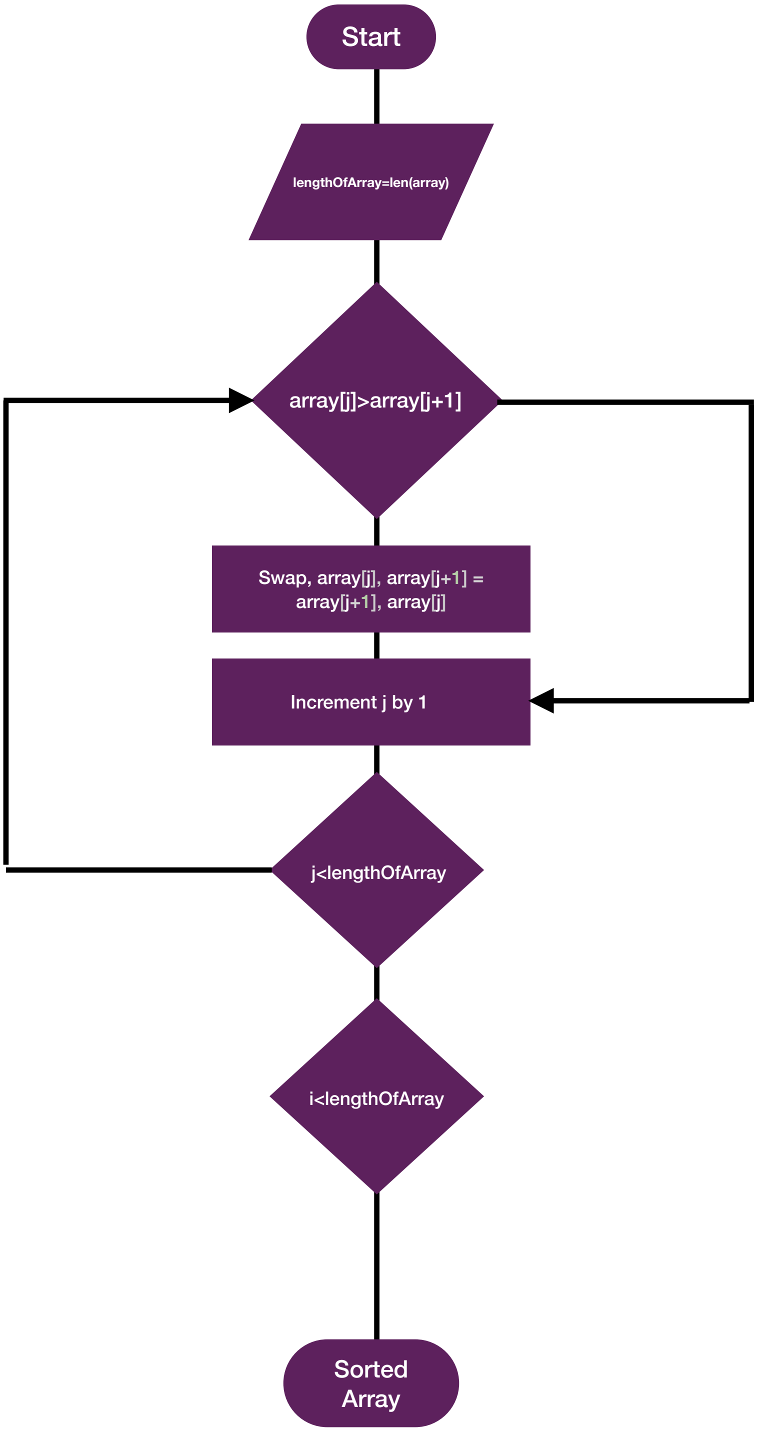

Start
i=0
startElement=I, j=I+1
array[j]<array[start Element]
startElement = j
No
J<len(array)
Swap, array[i], array[startElement] = array[startElement], array[i]
Increment i by 1
No
i<len(array)
Yes
Sorted Array

Bubble Sort:

Algorithm:

Step 1: Start
Step 2: Accept the array, the first element and the element next to it, i.e. first element +1, compare them and swap them if the left element is bigger.
Step 3: Take the next pair of elements while the left most element being the second element in the array, swap them if left is bigger than the right.
Step 4: Continue the steps 2-3 until the largest element is at the end, the first iteration ends
Step 5: Repeat steps 2-4 by reducing the range by 1 as the last element is already at the right position

Flow Chart:


Start
lengthOfArray=len(array)
array[j]>array[j+1]
Swap, array[j], array[j+1] = array[j+1], array[j]
Increment j by 1
j<lengthOfArray
i<lengthOfArray
Sorted Array

Insertion Sort:

Algorithm:

Step 1: Start

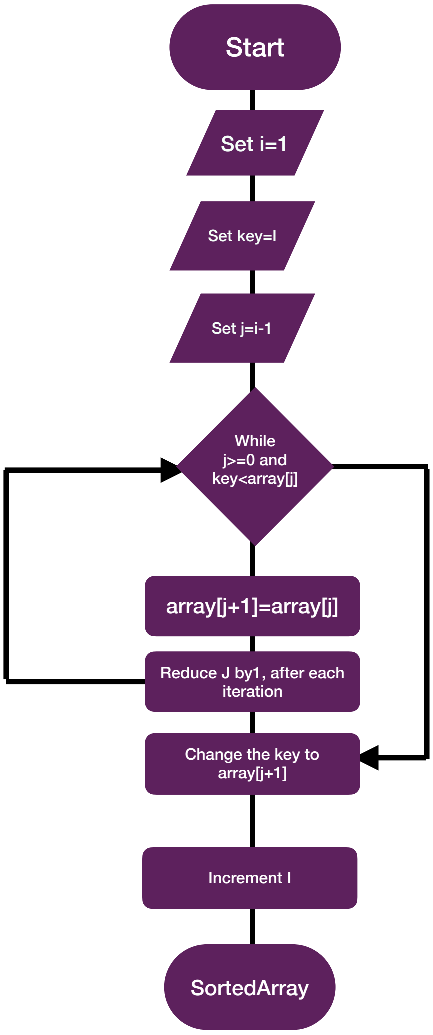
Step 2: Compare the first two elements and swap them if the first element is larger than the second.

Step 3: Take the next pair of elements, i.e. 2 and 3, swap the if the the 2nd element is larger than the 3rd.

Step 4: Compare the first element with the 2nd element(which used to be the 3rd element), swap them if first element is bigger

Step 5:continue steps 2-4 until the array is sorted

Flow Chart:


Start
Set i=1
Set key=I
Set j=i-1
While j>=0 and key<array[j]
array[j+1]=array[j]
Reduce J by1, after each iteration
Change the key to array[j+1]
Increment I
SortedArray

Merge Sort:

Algorithm:

Step 1: Start

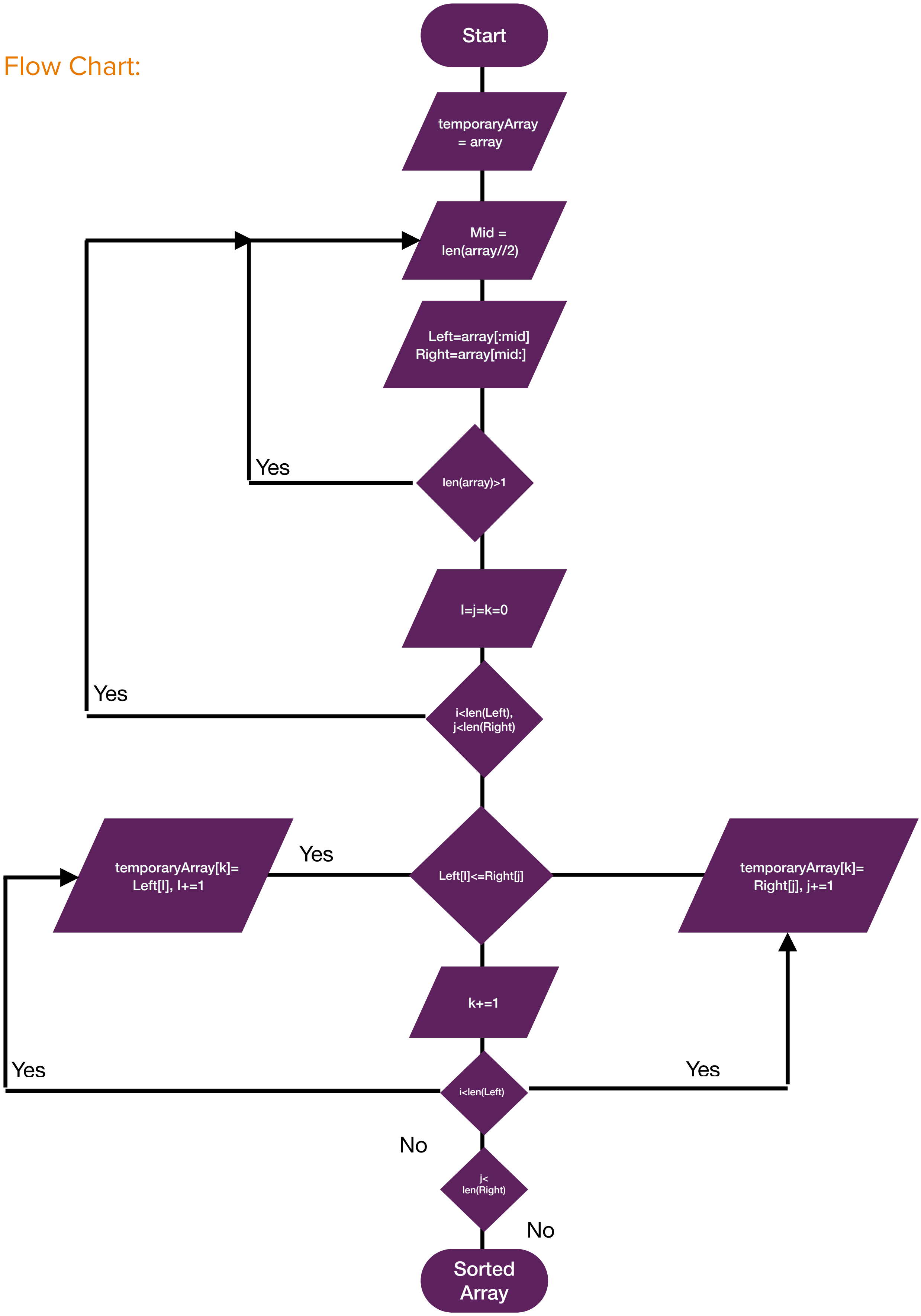
Step 2: Divide the array into length of single array recursively.

Step 3: Compare the left and right elements.

Step 4: Swap them if bigger and merge them

Step 5: Repeat steps 2-4 until the array is sorted

Flow Chart:


Start
temporaryArray = array
Mid = len(array//2)
Left=array[:mid] Right=array[mid:]
len(array)>1
Yes
I=j=k=0
i<len(Left), j<len(Right)
Yes
temporaryArray[k]= Left[I], I+=1
Yes
Left[I]<=Right[j]
temporaryArray[k]= Right[j], j+=1
k+=1
Yes
i<len(Left)
Yes
No
j< len(Right)
No
Sorted Array

Linear Search:

Algorithm:

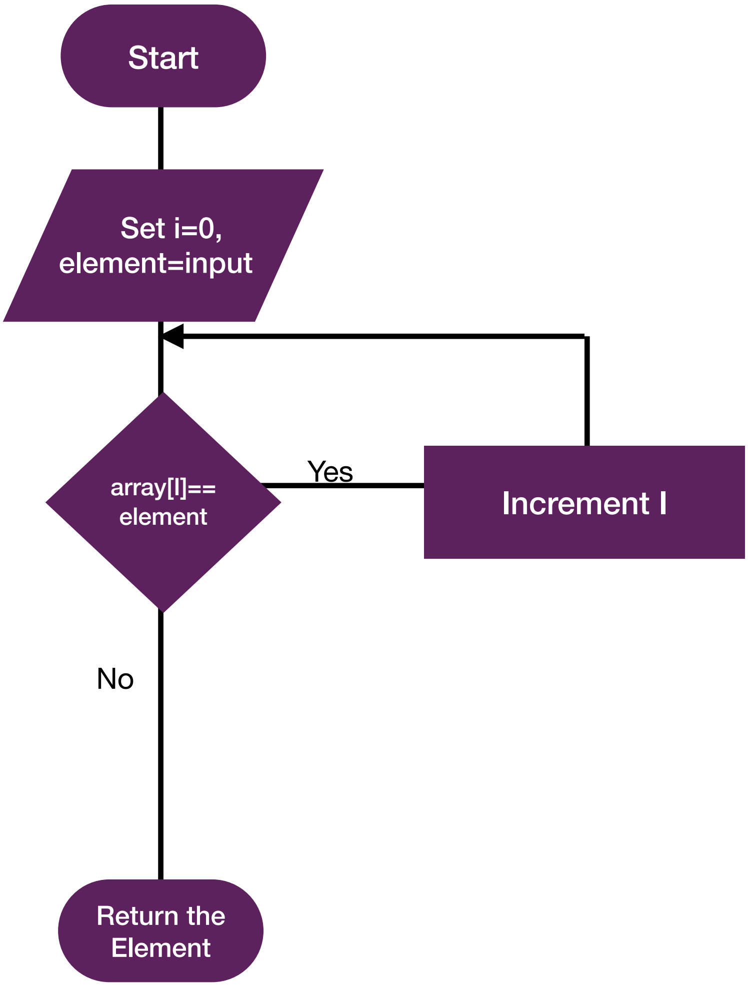
Step 1: Start

Step 2: Set i=0 and the range of loop to length of array, and accept the input.

Step 3: Compare array[I] to the element, if not the element increment i by 1.

Step 4:Repeat step 3 until element is found

Flow Chart:

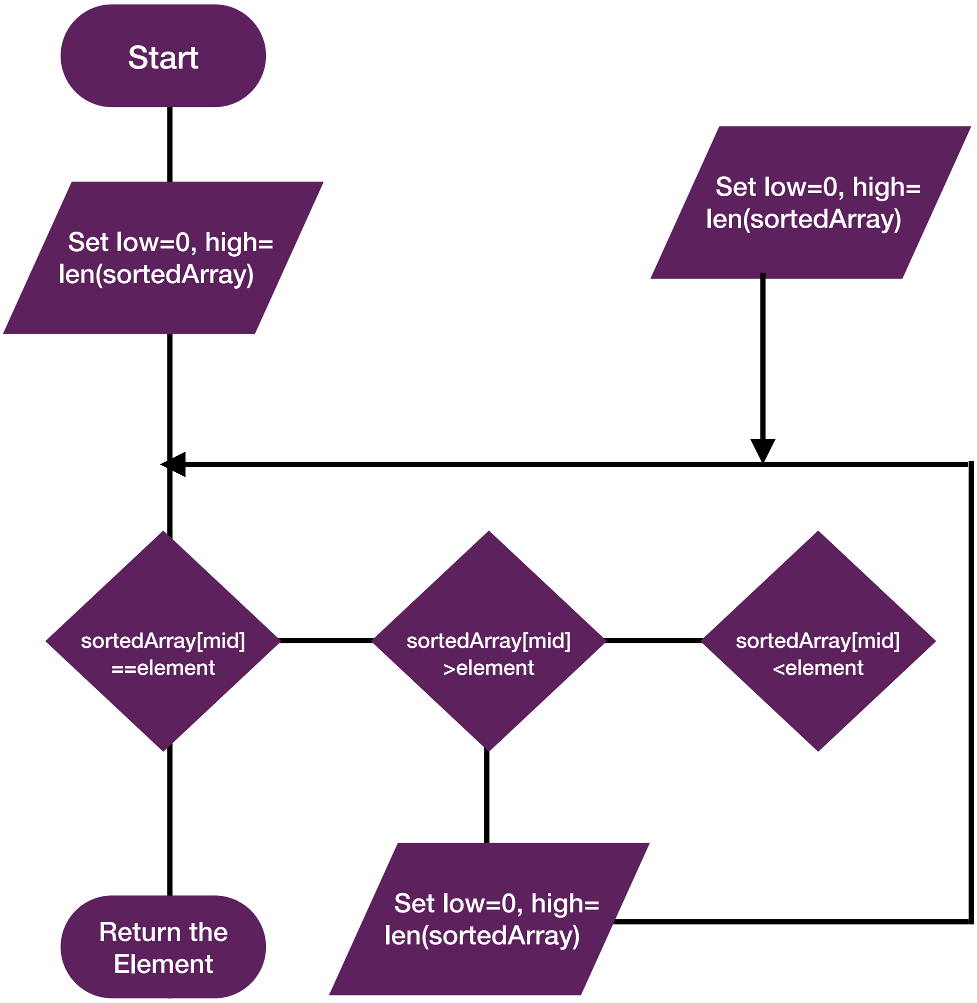


Binary Search:

Algorithm:

Step 1: Start
Step 2: Accept sortedArray and element
Step 3: Set low as 0, high as len(sortedArray)+1
Step 4: if high >=low, set mid=low+(high-low)//2
Step 5: Compare the element and the mid, if mid is equal to the element return mid.
Step 6: if sortedArray[mid]>element, recursively call binarySearch function, this time pass in mid -1 as high.
Step 7: if sortedArray[mid]<element, recursively call binarySearch function, this time pass in mid +1 as high.
Step 8: step 5 will return the element.

Flowchart:


Start
Set low=0, high= len(sortedArray)
Set low=0, high= len(sortedArray)
sortedArray[mid] ==element
sortedArray[mid] >element
sortedArray[mid] <element
Return the Element
Set low=0, high= len(sortedArray)