SEARCHING Algos

Search & Sort

Page No:- 01
Date :- 21 / 07 / 23

Coding Ninjas → Python Course → Parikh Bhaiya.

code → VS code → coding → Python → CN → Linear search

① * Linear Search :-

arr = | 20 | 30 | 40 | 50 | 90 | 100 | → Size → 6

Index → 0 1 2 3 4 5
-6 -5 -4 -3 -2 -1

↳ if we wanna index of 50 then we will create a variable i and iterate in array from 0th index tone by one.

i.e.
```
for i in range(len(arr)):
    if arr[i] == 50:
        # print(index(50)) (arr.index(50))
        print(i)
```

→ So whenever we are trying to iterate a variable in the array ~~or~~ from one end to another and searching element, this searching technique is known as Linear Search.

— x — x —

function code

LinearSearch1.py → explained properly

```
def linearsearch(array, element):
    for i in range(len(array)):
        if array[i] == element:
            return("Index of {}: {}".format(element, i))
    return "Element not found!"
```

** (2) Binary search :-

↳ This algo works only on the sorted array **
↳ Hence, sort the array first **

arr = | 31 | 27 | 3 | 20 | 1 | 21 | 7 | 18 |

↓ arr.sort()

0	1	2	3	4	5	6	7
1	3	7	18	20	21	27	31

↑ start (s) — ↑ end (e)

0	1	2	3	4	5	6	7

e.g.1 ↳ arr = 1 3 7 18 20 21 27 31

element = 21

← (code)

start = 0, end = 7, mid = 3

arr[mid] = 18 , 21 > 18 → start = mid+1 (element lies on right of arr[mid])

start = 4, end = 7, mid = 5

arr[mid] = 21 == element → return mid

↓ index of that element

e.g.2 element = 31

start = 0, end = 7, mid = 3

arr[mid] = 18 < 31 → start = mid+1

start = 4, end = 7, mid = 5

arr[mid] = 21 < 31 → start = mid+1

start = 6, end = 7, mid = 6

arr[mid] = 27 < 31 → start = mid+1

start = 7, end = 7, mid = 7

arr[mid] = 7 == 7 → return mid

e.g.3

Function code

BinarySearch2.py → explained properly

```
def BinarySearch(array, element):
    array.sort()
    start = 0
    end = len(array)-1
    mid = 0

    while (start <= end):
        mid = (start + end)//2
        if array[mid] == element:
            return ("Index of {}:{}".format(element, mid
        elif (array[mid] < element):
            start = mid + 1
            # Element lies on the right side of mid.
        elif (array[mid] > element):
            start end = mid - 1
            # Element lies on the left side of mid.
    if start > end:
        return ("Element not exist!")
```

e.g 3 → element = 3

start = 0, end = 7, mid = 3

arr[mid] = 18 > 3 → end = end - 1 (element lies on the left of ~~mid~~ arr[mid])

start = 0, end = 2, mid = 1

arr[mid] = 3 == 3 → return mid.

NOTE → arr =

0	1	2	3	4	5	6	7
1	3	7	18	20	21	27	31

element = 31

Linear Search

No. of time loop executed = 8

Index of 31 is 7

Binary Search

NO. of time loop executed = 4 → page no. 03

Index of 31 is 7

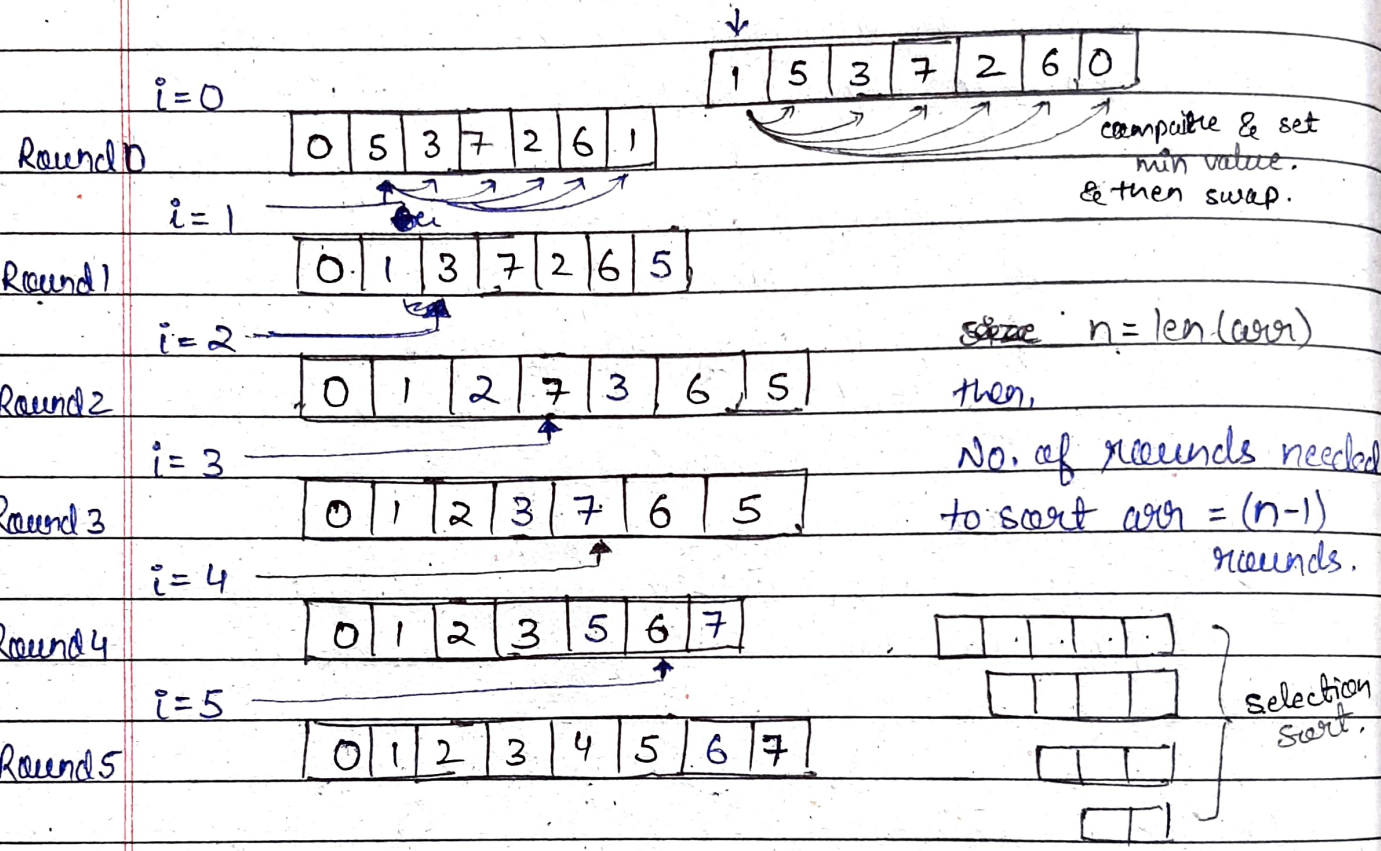
↳ Linear Vs Binary.py → explained properly.

SORTING Algos

Page No:- 06
Date :- 22/07/23

① SELECTION SORT

↳ to sort the given array
↳ we will find out the min no. in every pass & then swap with the first element in every pass.



n = len(arr)
then,
No. of rounds needed to sort arr = (n-1) rounds.

↳ selectionSort2.py

```
def selectionsort(arr):
    length = len(arr)
    # put the correct element at ith position
    for i in range(length-1):
        minIndex = i
        # calculating the index of minimum element
        # for this iteration
        for j in range(i+1, length):
            if (arr[j] < arr[minIndex]):
                minIndex = j
        ## swaping
        arr[i], arr[minIndex] = arr[minIndex], arr[i]
    return arr
```

——— x ——— x ——— x ———

↳ for n elements, we need to execute (i) loop (n-1) times.

② BUBBLE SORT :-

↳ for n elements, we need to iterate (n-1) times.

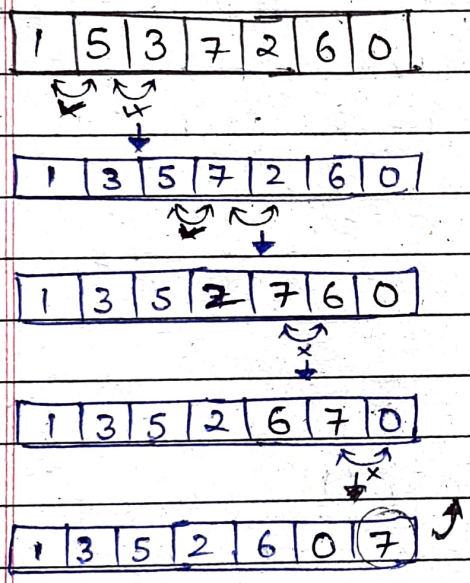
1	5	3	7	2	6	0

↳ iterating loop len(arr)-1 times.

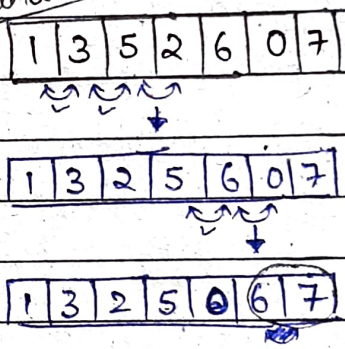
↳ check two adjacent elements &
if arr[i] > arr[i+1]:
then swap them

↳ in every pass/round we got the largest elements at the end.

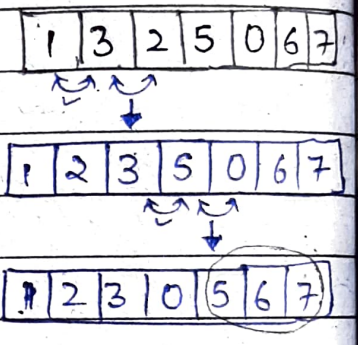
Round1



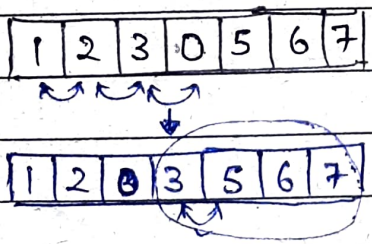
Round2



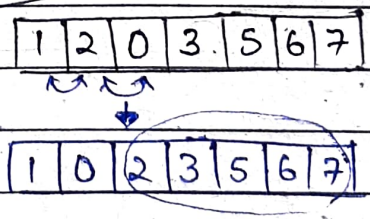
Round3



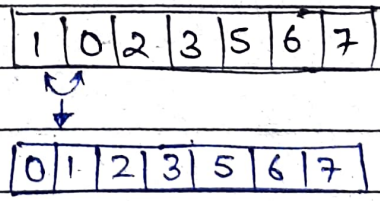
Round4



Round5



Round6



i is for n-1 rounds
j is for in each iteration u need to till length - 1 - i position

4 BubbleSort2.py

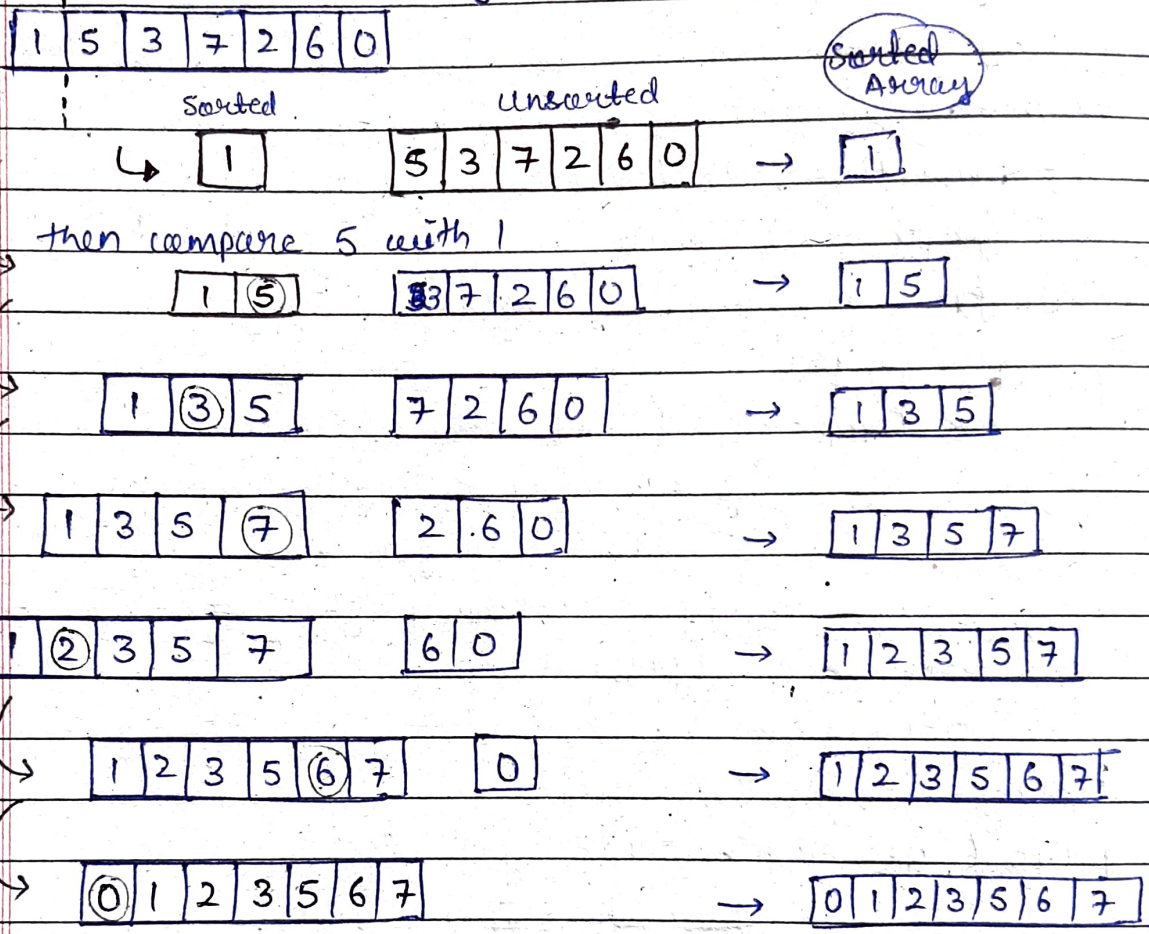
```
def inputArray():
    arr = list(map(int, input().split(" ")))
    return arr

#Bubble Sort
def bubbleSort(arr):
    length = len(arr)
    for i in range(length-1):          # No. of rounds
        for j in range(length-1):      # for 1 round.
            if arr[j] > arr[j+1]:      # swaping
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

#Input
array = inputArray();
#Output
print(f" Input Array : {array} ")
print(f" Sorted Array : { bubbleSort(array)} ")
```

—— x —— x ——

③ INSERTION SORT

↳ Make two array, one is sorted another is unsorted

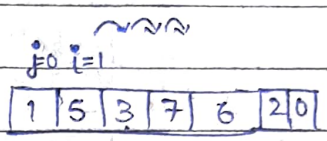
↳ Then compare first element of unsorted array with sorted array and insert at correct position.



NOTE arr[i], arr[j] = arr[j], arr[i] → swaping in python with pop, insert functions.

↳ Insertion Sort.py

```
def inputArray():
    arr = list(map(int, input().split(" ")))
    return arr
```



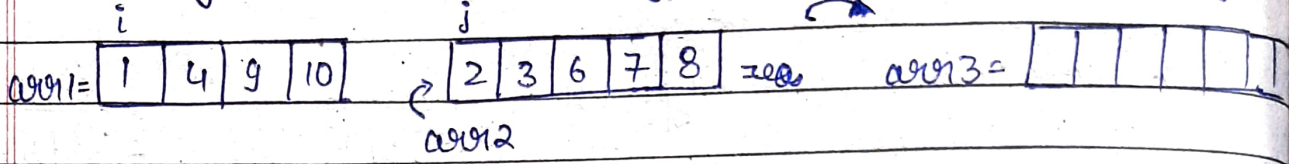

```
# Insertion Sort :-
def insertionSort(arr):
    length = len(arr)
    for i in range(1, length):
        j = i-1
        temp = arr[i]
        while (j>=0 and arr[j]> temp):
            arr[j+1] = arr[j]
            ~~arr[j+1] = temp~~ j = j-1   # Decrementing j
        ~~return~~ arr[j+1] = temp
        return arr
array = [1, 5, 3, 7, 6, 2, 0]
print(insertionSort(array))
```

while loop.

```
def insertionSort(arr):
    for i in range(1, len(arr)):
        j = i-1
        while (j>=0 and arr[j] > arr[j+1]):
            arr[j+1], arr[j] = arr[j], arr[j+1]
            j = j-1
    return arr
```

④ ~~MERGESORT~~ (Merge Two sorted Arrays)

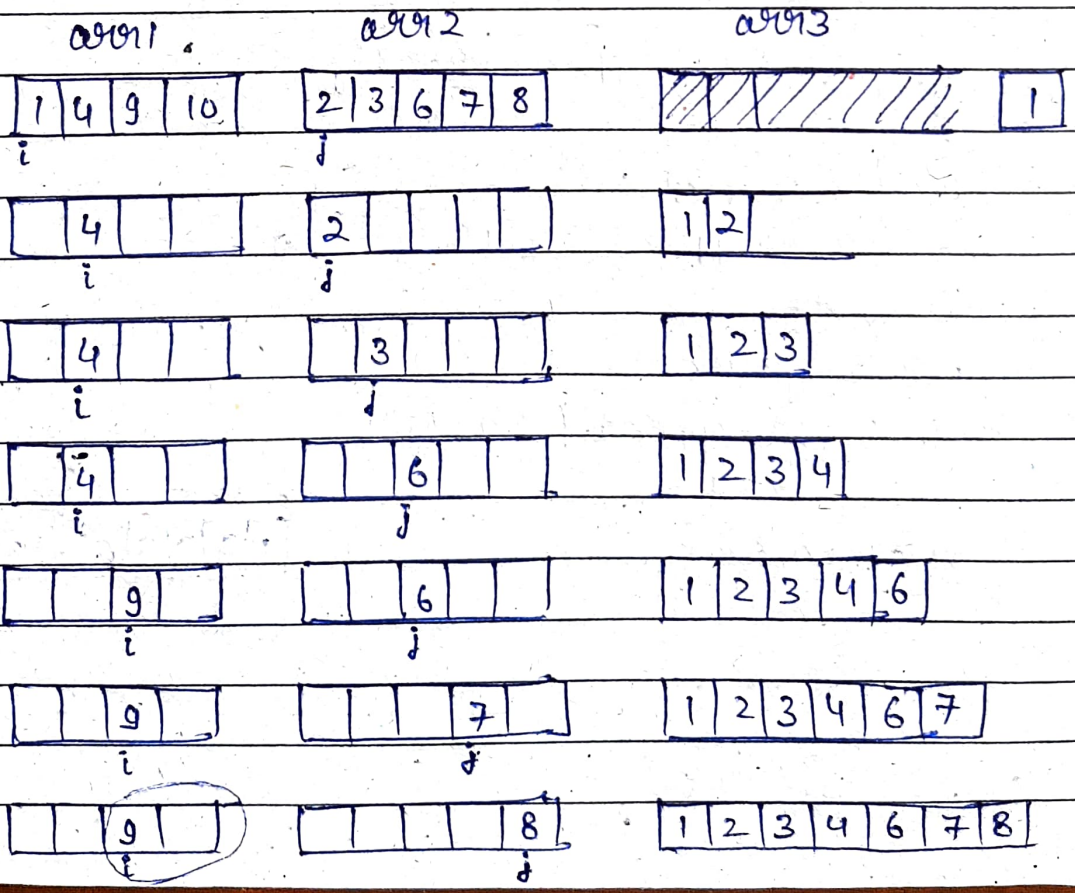
↳ Merge two sorted arrays into an another array.



↳ Compare i and j which is smaller then append that into arr3.

```
↳ if arr[i] < arr[j] :
        arr3.append(arr[i])
        i = i+1
  else :
        arr3.append(arr[j])
        j = j+1
```

} while loop



↳ if i == len(arr1) - 1 or j == len(arr2) - 1 :
then arr3. append (all the remaining items of another list).

—x—x—

↳ MergeTwoSortedArray1.py

i

1	4	9	10

j

2	3	6	7	8

```
def MergeTwoSorted Array (arr1, arr2):
    i=0
    j=0
    len1, len2 = len(arr1), len(arr2)
    arr= []
    while ((i<len1) and (j< len2)):
        if (arr1[i] < arr2[j]):
            arr. append (arr1[i])
            i +=1
        else :
            arr. append (arr2[j])
            j +=1
    while (i< len1):
        arr. append (arr1[i])
        i+=1
    while (j< len2):
        arr. append (arr2[j])
        j +=1
    return arr
```