INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

INTRODUCTION TO ALGORITHMS

Course Code: EC351

Project Report

Submitted by: **TEAM 7**

Abdul Rahman	18BEC001
Ajmal A	18BEC003
Aryan Kumar	18BEC004
B.K Likhith Kumar	18BEC005
Safwan Mohammad	18BEC028

Submitted To:

DR. UMA S

20 Nov 2020

Project Objective:-

An Array of 10 to 15 Images:
1. Write an Algorithm to SORT the image array using any suitable Sorting algorithms.
2. Sort them in an Ascending Order of their Size.
3. Given a new unknown image, search the new image in the array and display the result as found or not along with the image.
4. Write a Program and find out the Time complexity for Searching and Sorting Algorithm implementation of selected image Arrays.

==========XXXXXXXXXXX==========

Language Used: PYTHON 3

Python Modules Used:
- OpenCV2
- Matplotlib
- Numpy
- PIL (for Image opening/closing)
- Psutil
- Time

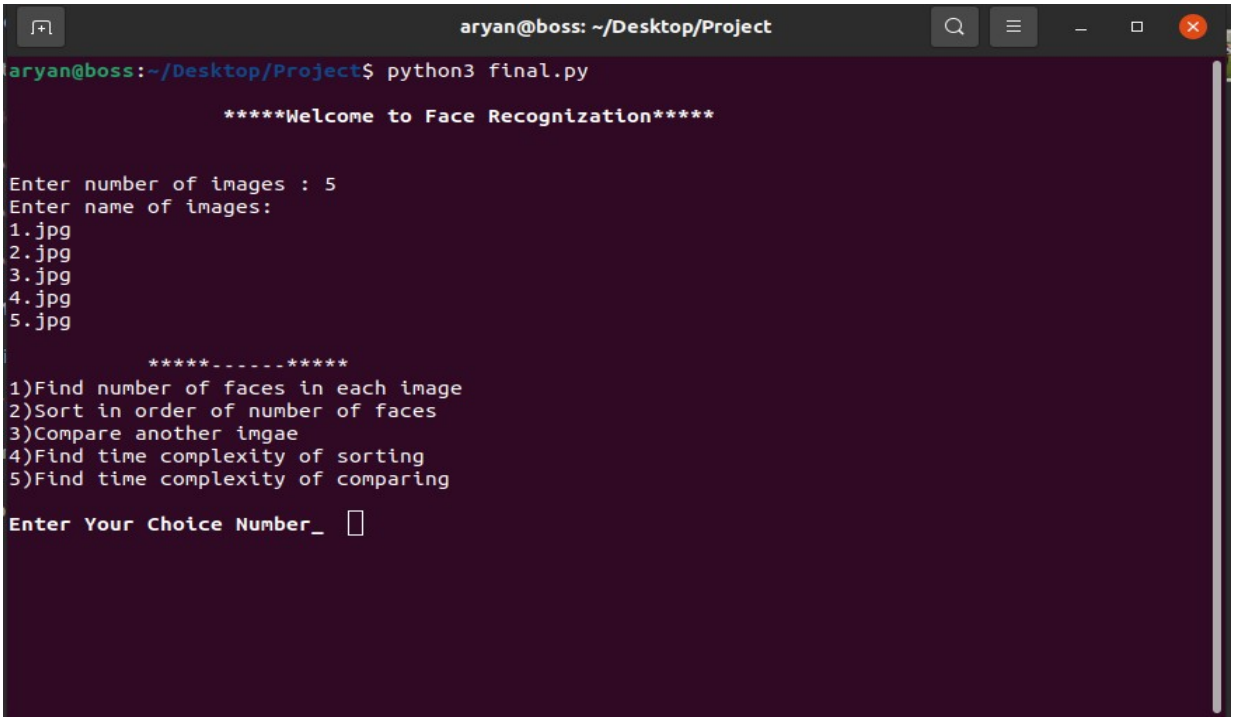
Image Format Used:
- 1.jpg , 2.jpg , 3.jpg, ..., 15.jpg

(Example jpgs)

Segment_1: Finding faces in a given image/array of images.

OpenCV (Open Source Computer Vision) is a popular computer vision library . The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms. OpenCV 2.4 now comes with the very new FaceRecognizer class for face recognition.

- As the program starts to run, first it will ask tje user to input the number of images that will be stored in an image ndarray (numpy).

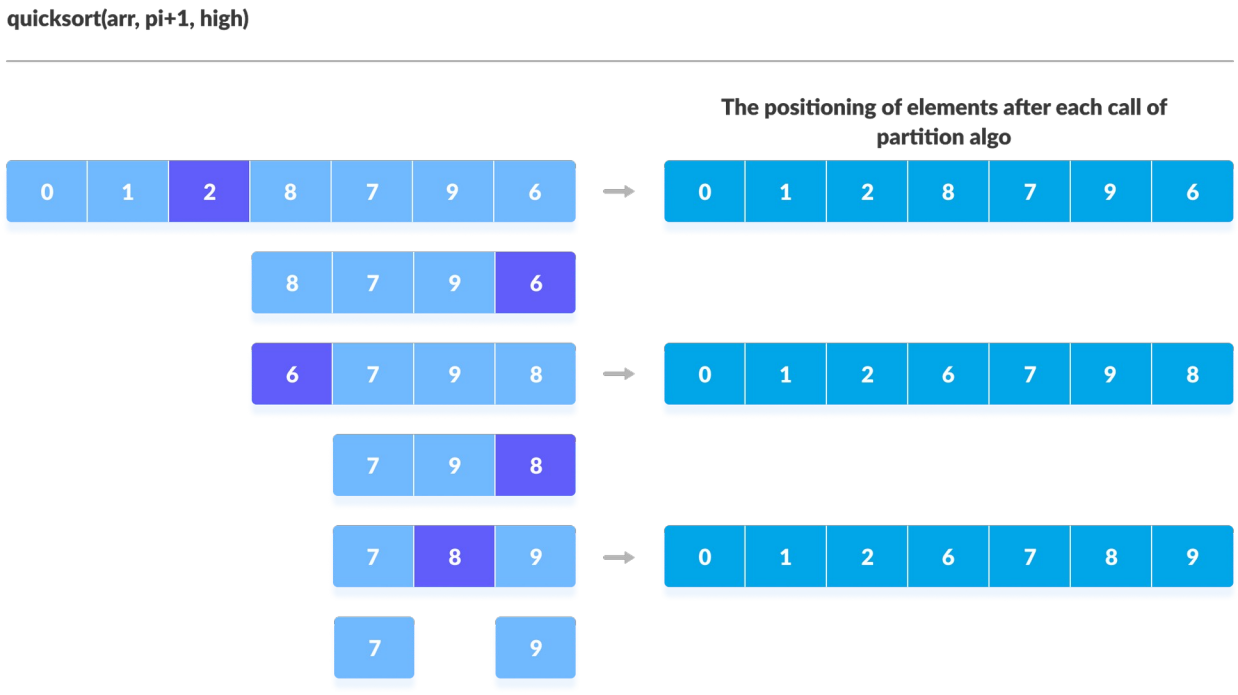


- A function **facereco()** is given input as array of images which then find number of faces in each image and prints it.

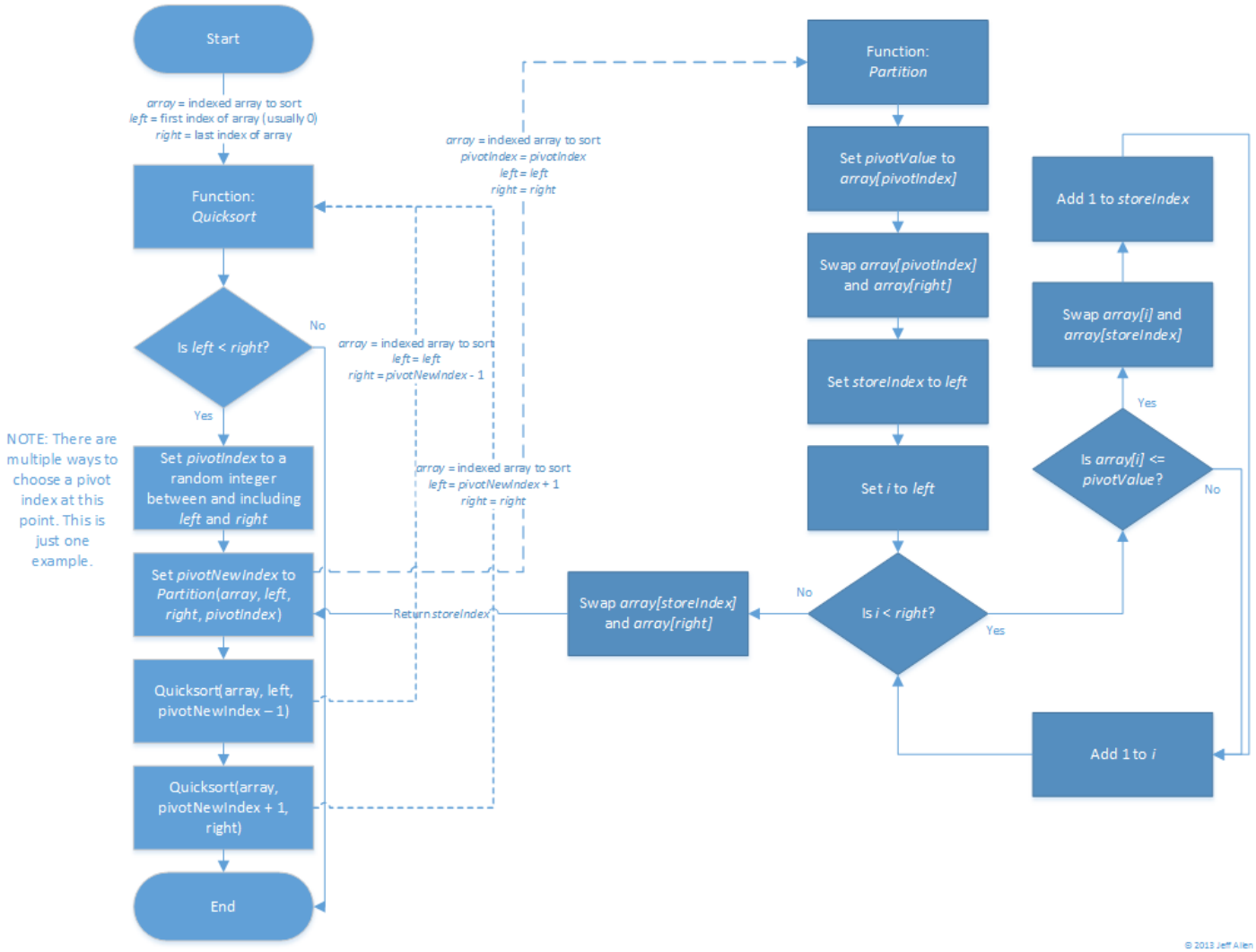
```
aryan@boss: ~/Desktop/Project
aryan@boss:~/Desktop/Project$ python3 final.py

                *****Welcome to Face Recognization*****


Enter number of images : 5
Enter name of images:
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg

          *****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing

Enter Your Choice Number_  1
Faces found in Image  1 is - 7
Faces found in Image  2 is - 15
Faces found in Image  3 is - 3
Faces found in Image  4 is - 1
Faces found in Image  5 is - 20

          *****------*****
```

Segment_2: Sorting in order of number of faces.

After storing number of faces in images in an integer array, the array is given to the input of a **quicksort()** fuction where the array gets sorted by ***Quicksort algorithm***.



Flow Chart of Quick Sort:



QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways. Always pick first element as pivot, Always pick last element as pivot (implemented below), Pick a random element as pivot, Pick median as pivot.

Psudo Code:

```
/* low  --> Starting index,  high  --> Ending index */
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```

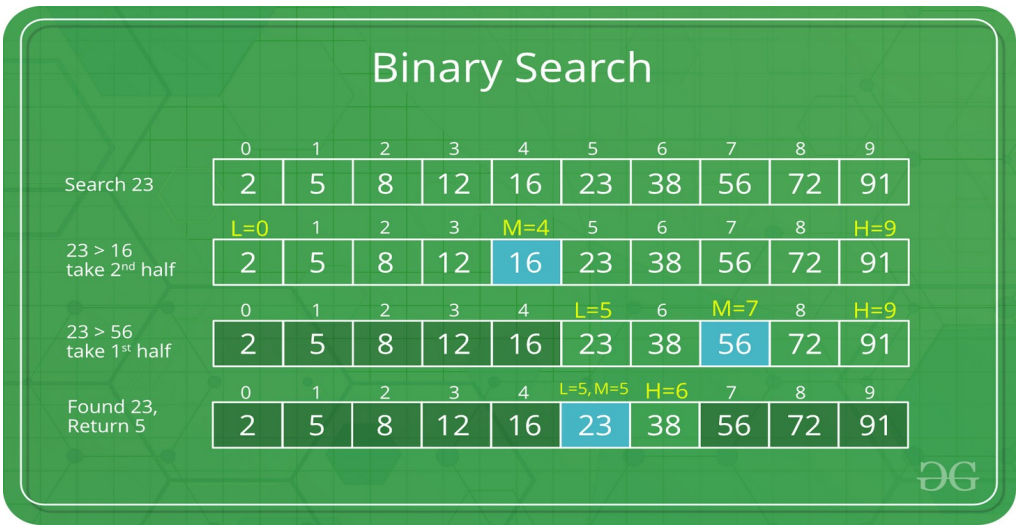
Output of Sorting:

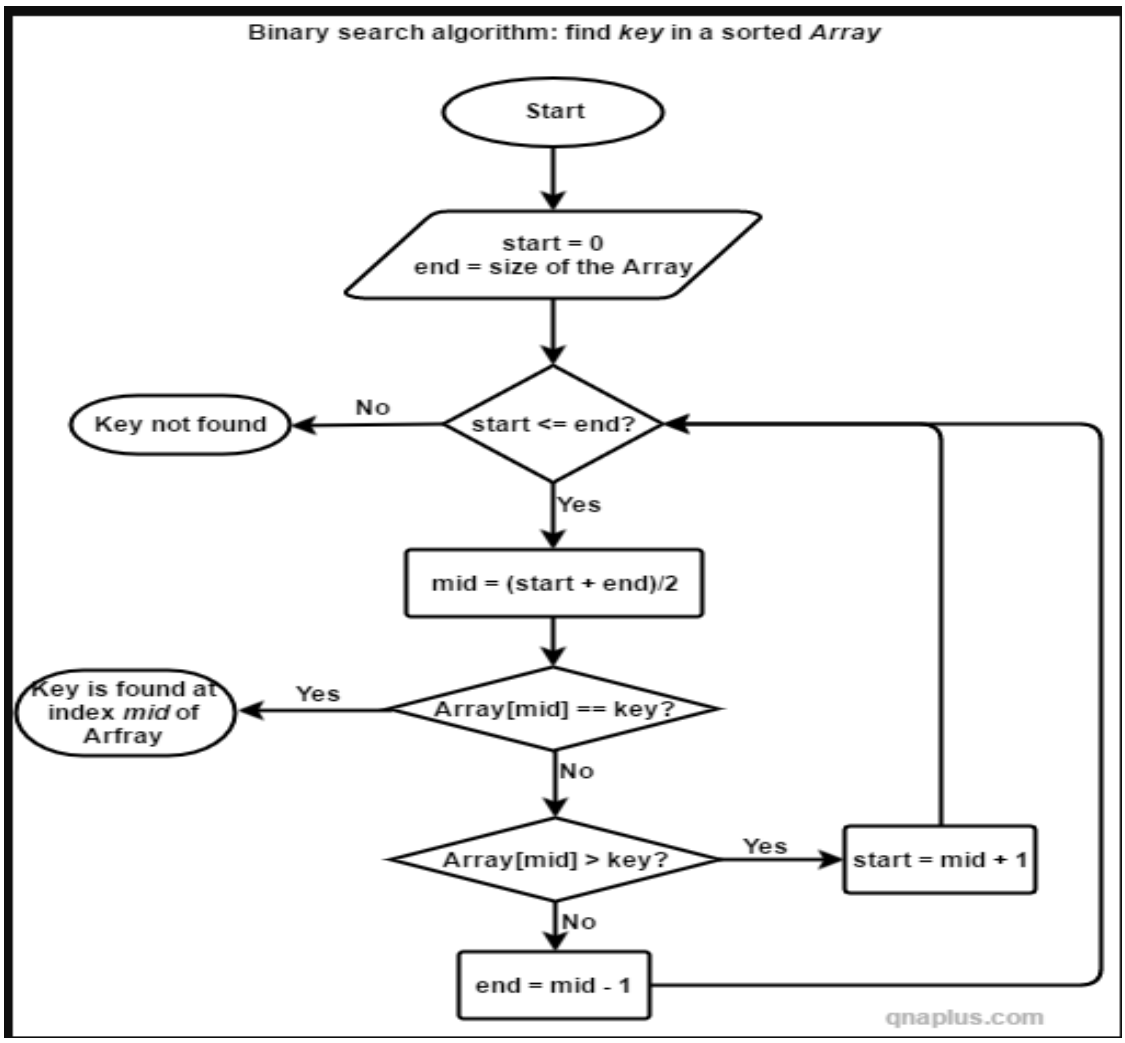
```
aryan@boss: ~/Desktop/Project
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing

Enter Your Choice Number_  1
Faces found in Image  1 is - 7
Faces found in Image  2 is - 15
Faces found in Image  3 is - 3
Faces found in Image  4 is - 1
Faces found in Image  5 is - 20

          *****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing

Enter Your Choice Number_  2
Array of number of faces:  [7, 15, 3, 1, 20]


Sorted list in order of number of faces:[1, 3, 7, 15, 20]
Killed
Killed
Killed
Killed
```

Segment_3: Finding a similar image as given input.

Searching in array: -

Binary Search: Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.




Binary search algorithm: find key in a sorted Array
Start
start = 0
end = size of the Array
start <= end?
No
Key not found
Yes
mid = (start + end)/2
Array[mid] == key?
Yes
Key is found at index mid of Arfray
No
Array[mid] > key?
Yes
start = mid + 1
No
end = mid - 1
qnaplus.com

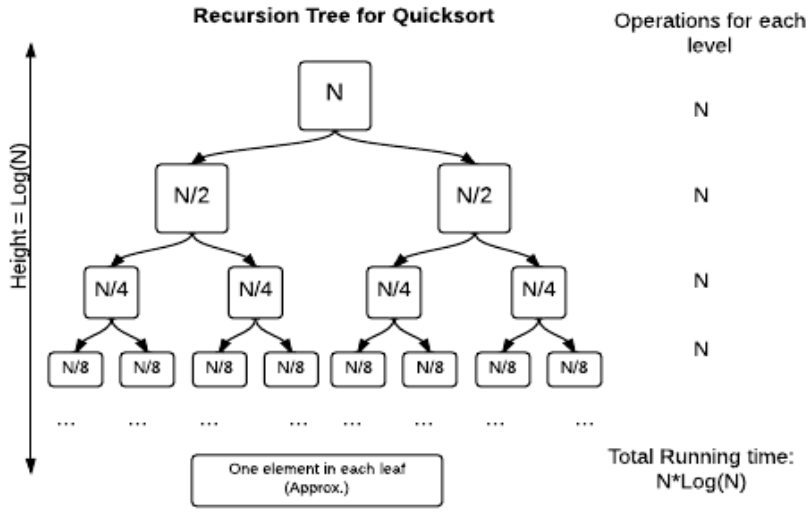

ImageMagick: tmp1mx0dk5w.PNG
ImageMagick: tmpjntrm3ht.PNG
Terminal
4)Find time complexity of sorting
5)Find time complexity of comparing
Enter Your Choice Number_ 3
Enter image_ 7.jpg
Faces found in entered image: 3
====> Same number of faces found!
*****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing
Enter Your Choice Number_ 3
.Enter image_ 3.jpg
*****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing
Enter Your Choice Number_ 3
Enter image_ 3.jpg
Faces found in entered image: 3
====> Same number of faces found!
*****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing
Enter Your Choice Number_

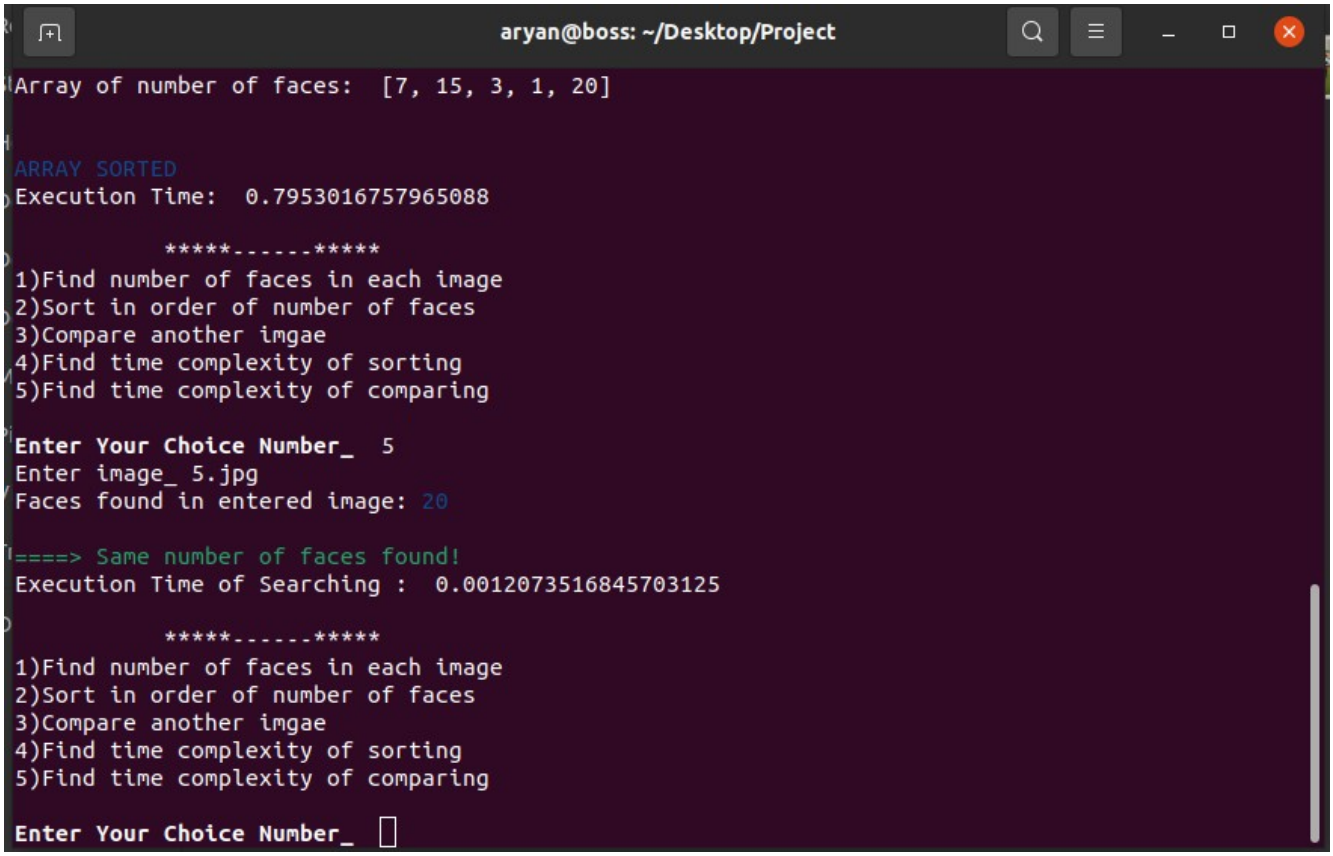
Segment_4: Time Complexity of Quick Sort Algorithm.

➔ For an array, in which partitioning leads to unbalanced subarrays, to an extent where on the left side there are no elements, with all the elements greater than the pivot, hence on the right side. And if keep on getting unbalanced subarrays, then the running time is the worst case, which is O(n2) Where as if partitioning leads to almost equal subarrays, then the running time is the best, with time complexity as O(n*log n).

- Worst Case Time Complexity [Big-O]: **O(n2)**
- Best Case Time Complexity [Big-omega]: **O(n*log n)**
- Average Time Complexity [Big-theta]: **O(n*log n)**
- Space Complexity: **O(n*log n)**

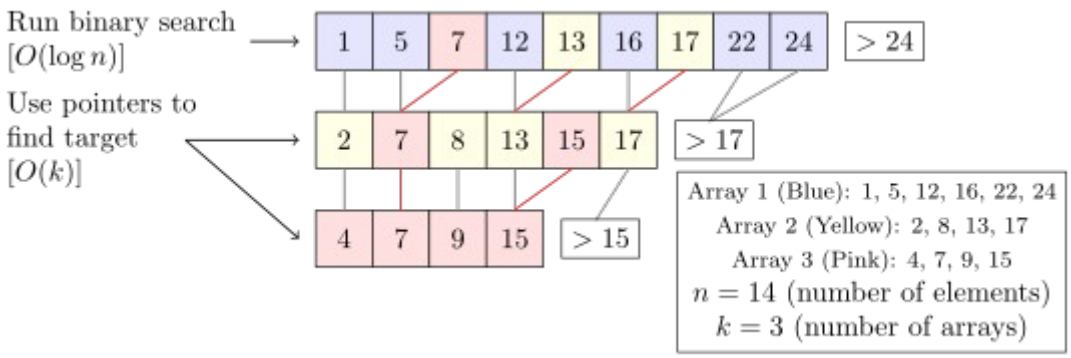
As we know now, that if subarrays partitioning produced after partitioning are unbalanced, quick sort will take more time to finish. If someone knows that you pick the last index as pivot all the time, they can intentionally provide you with array which will result in worst-case running time for quick sort. To avoid this, you can pick random pivot element too. It won't make any difference in the algorithm, as all you need to do is, pick a random element from the array, swap it with element at the last index, make it the pivot and carry on with quick sort. Space required by quick sort is very less, only O(n*log n) additional space is required. Quick sort is not a stable sorting technique, so it might change the occurence of two similar elements in the list while sorting.





Segment_5: Time Complexity of Binary Search Algorithm.

The time complexity of the binary search algorithm is **O(log n)**. The best-case time complexity would be **O(1)** when the central index would directly match the desired value. The worst-case scenario could be the values at either extremity of the list or values not in the list. The space complexity of the binary search algorithm depends on the implementation of the algorithm. The binary search algorithm breaks the list down in half on every iteration, rather than sequentially combing through the list. On large lists, this method can be really useful. *(Example)*




```
aryan@boss: ~/Desktop/Project
Enter Your Choice Number_  5
Enter image_ 5.jpg
Faces found in entered image: 20

====> Same number of faces found!
Execution Time of Searching :  0.0012073516845703125

          *****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing

Enter Your Choice Number_  3
Enter image_ 6.jpg
Faces found in entered image: 9

====> Match not found!

          *****------*****
1)Find number of faces in each image
2)Sort in order of number of faces
3)Compare another imgae
4)Find time complexity of sorting
5)Find time complexity of comparing

Enter Your Choice Number_
```