PROJECT REPORT-EC351

VIRTUAL HACKATHON on

Sorting Images based on size attribute &

Searching if an image is present in our data

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Submitted to

Prof. Uma Sheshadri

Dept. of Computer Science and Engineering

IIIT Dharwad

PROJECT TEAM

Team:1

18BEC024 - Kummara Bhargavi

18BEC037 - Pasupuleti Chandana

18BEC038 - Peddoju Sathvik

18BEC016 - Gurijala Sai Harsha

18BEC012 - Ellandula Shiva sai Krishna

ACKNOWLEDGMENT

The successful completion of this project during the pandemic does much credit to Indian Institute of Informational Technology Dharwad (IIIT DWD) and is indeed a matter of pride for IIIT DWD and IIIT DWD students.

We are grateful to our Professor **Dr. Uma Sheshadri** for giving us an opportunity to participate in a group project and for being an active source of support and encouragement. We also want to thank her for her valuable suggestions, feedback and advice throughout the semester.

PROJECT DETAILS

Problem statement:

1. An Array of 10 to 15 Images: Select images from the Group of [Flowers or Insects or Fruits or any suitable objects]
2. No group is allowed to take the same type of Images.

To do tasks:

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
5. No HARD code is allowed in the Program

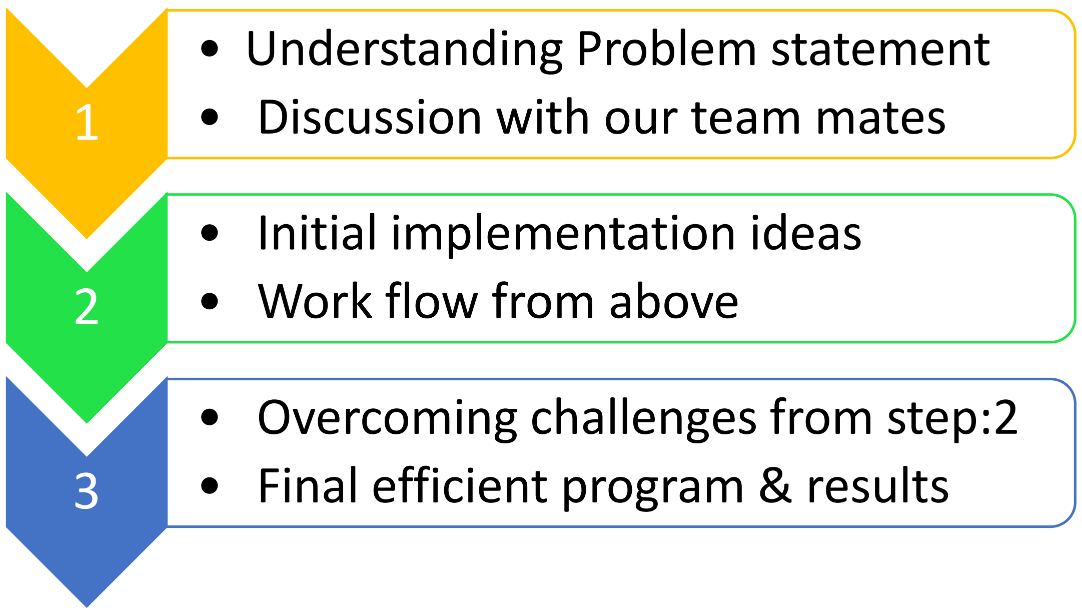
Note:

1. Use any OS and any programming language
2. Team submission is allowed
3. Create a GitHub account and upload the assignments and observation report
4. Use a suitable SEARCH, SORT algorithms

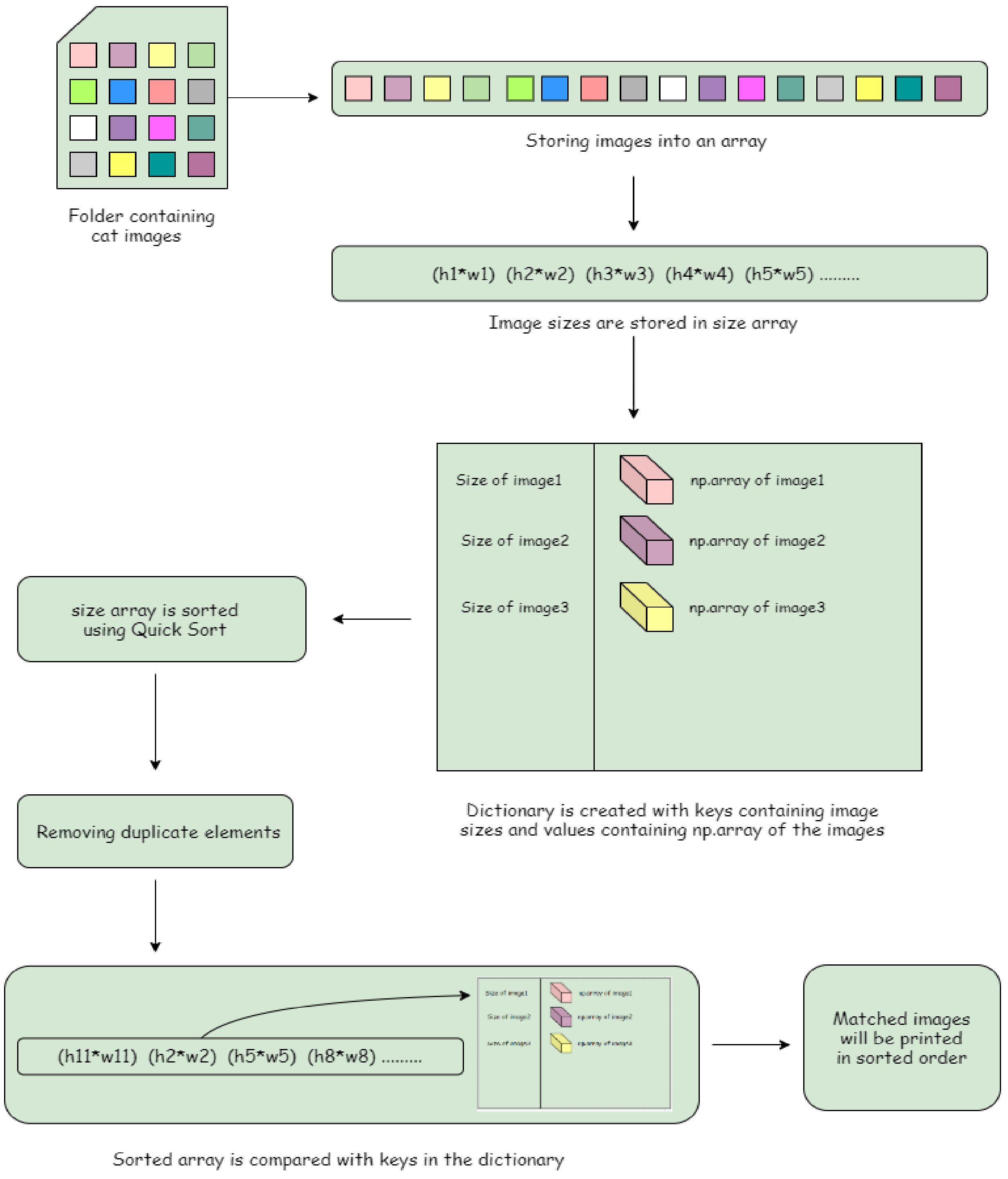
Project Scope:

In today’s world, Technology has made everything easy around us without any chaotic work. Suppose there is a situation where you have some loads of data containing images and you need to arrange them in either ascending or descending order. Then it becomes hectic to manually go through the size of each image and arrange accordingly, this is the place where sorting plays main role. Similarly, if you have got an image and you want to know if it is present in your data set then it would be a tedious task for us to manually check. So Algorithms play a crucial role in these circumstances.

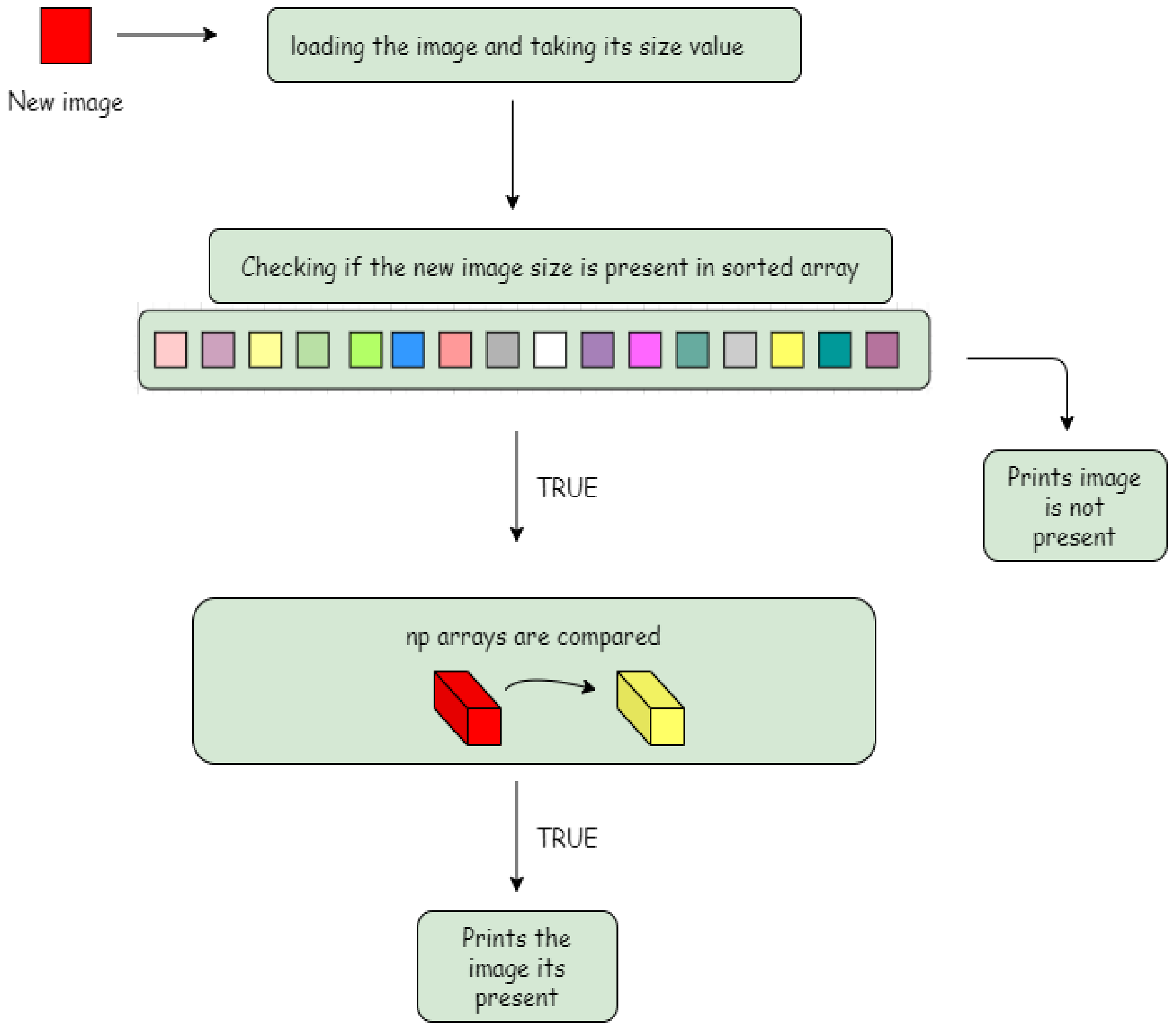
Our approach:


1
• Understanding Problem statement
• Discussion with our team mates
2
• Initial implementation ideas
• Work flow from above
3
• Overcoming challenges from step:2
• Final efficient program & results

FLOWCHART for sorting images:


Folder containing cat images
Storing images into an array
(h1*w1) (h2*w2) (h3*w3) (h4*w4) (h5*w5) ....
Image sizes are stored in size array
Size of image1
np.array of image1
Size of image2
np.array of image2
Size of image3
np.array of image3
Dictionary is created with keys containing image sizes and values containing np.array of the images
size array is sorted using Quick Sort
Removing duplicate elements
(h11*w11) (h2*w2) (h5*w5) (h8*w8) ....
Sorted array is compared with keys in the dictionary
Matched images will be printed in sorted order

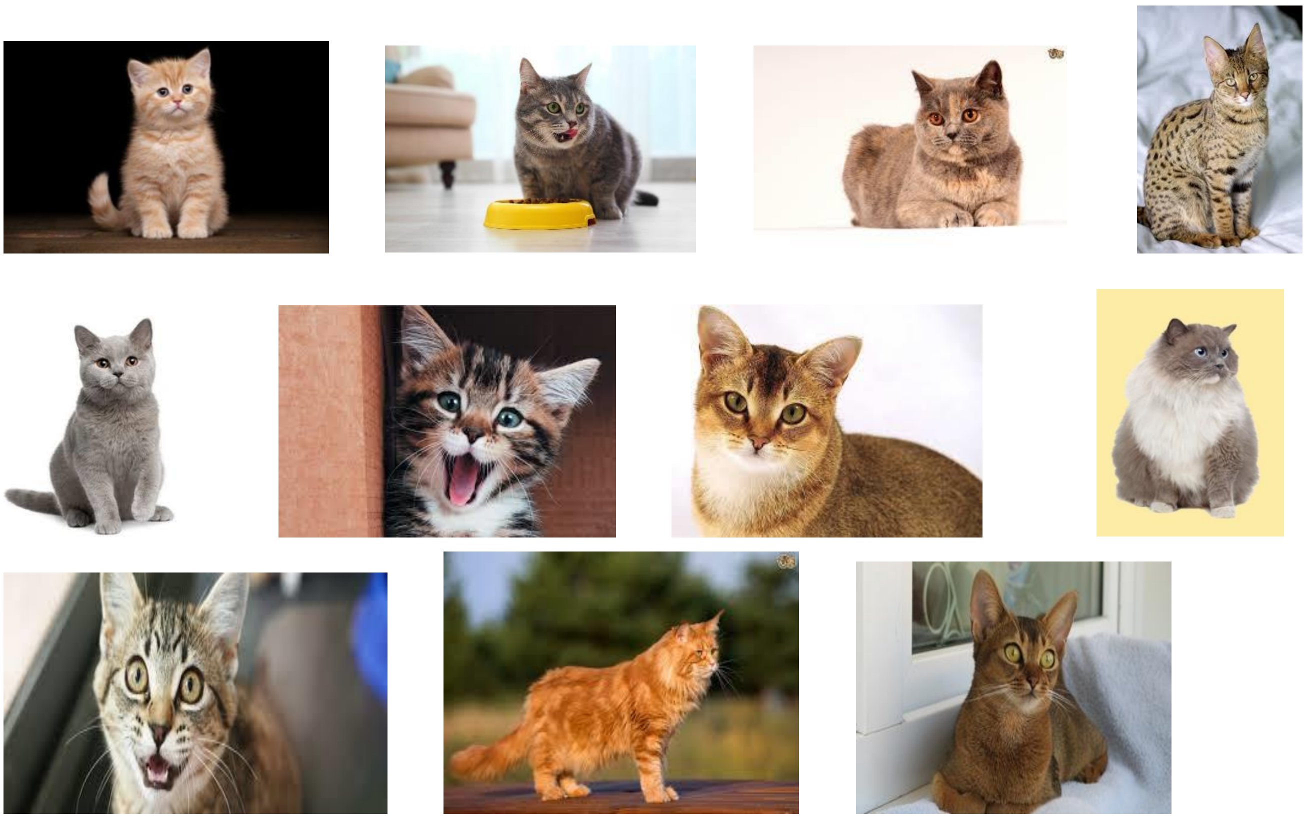
FLOWCHART for searching new image:



New image
loading the image and taking its size value
Checking if the new image size is present in sorted array
TRUE
Prints image is not present
np arrays are compared
TRUE
Prints the image its present

Requirements:

Dataset: Any objects or pictures can be chosen for sorting



Operating system used: Windows 10

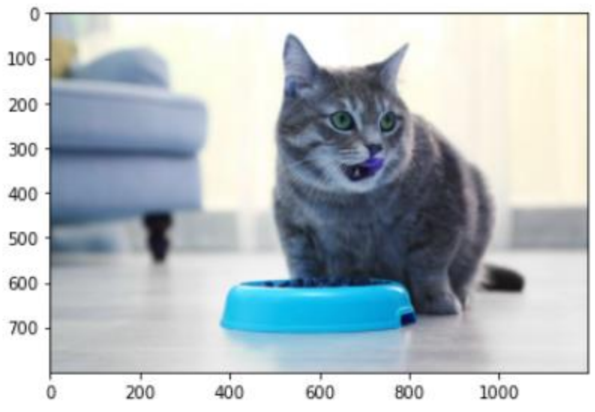
Programming language: Python3

Libraries and modules used: Opencv, PIL, Matlplotlib, glob, numpy, collections, colorthief, statistics

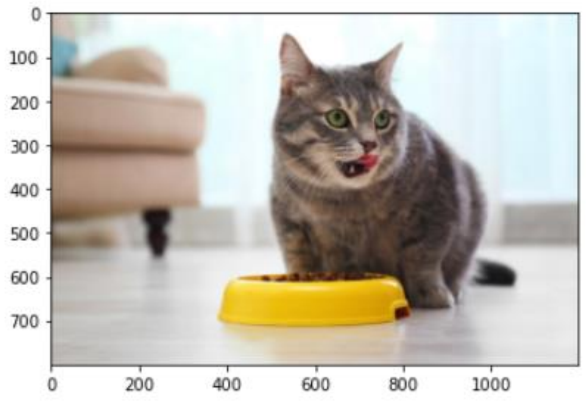
Brief description:

<u>Step 1:</u> Importing images and storing them into a list

Before storing the images into a list, import the images into our working environment by providing the path of the image folder location and then store them into a list. Here if we use opencv to load images into list, then by default it loads them in BGR format. So care is to be taken while using opencv to load images.



BGR format



RGB format

<u>Step 2:</u> Storing image sizes into another list

After storing images into a list, calculate the resolution for each image by finding their shapes using PIL library. Store all the sizes(resolution) into another list.

Step 3: Create a Dictionary

Since we have got two lists, one containg information(numpy arrays) of image and second containing the image sizes, let us create a dictionary with keys as image sizes and values as their respective image information.

Challenge faced here:

There is a problem in creating a dictionary if we have different images with same sizes. To overcome this we created a list for storing values(np arrays) with the same key(image size). So if the images with same sizes are printed at once.

Step 4: Sort image sizes using quick sort

Inorder to sort the images, we need to sort the image sizes. So we chose quick sort algorithm because of it's efficient way of sorting elements based on pivot. Hence, we sorted our image size list using this algorithm and stored the sorted list into another list.

Step 5: Remove duplicate image sizes:

As explained in the step3 while creating a dictionary all the images with same size are already stored in a list under the same key(size). So there is no need of duplicate sizes in our list.

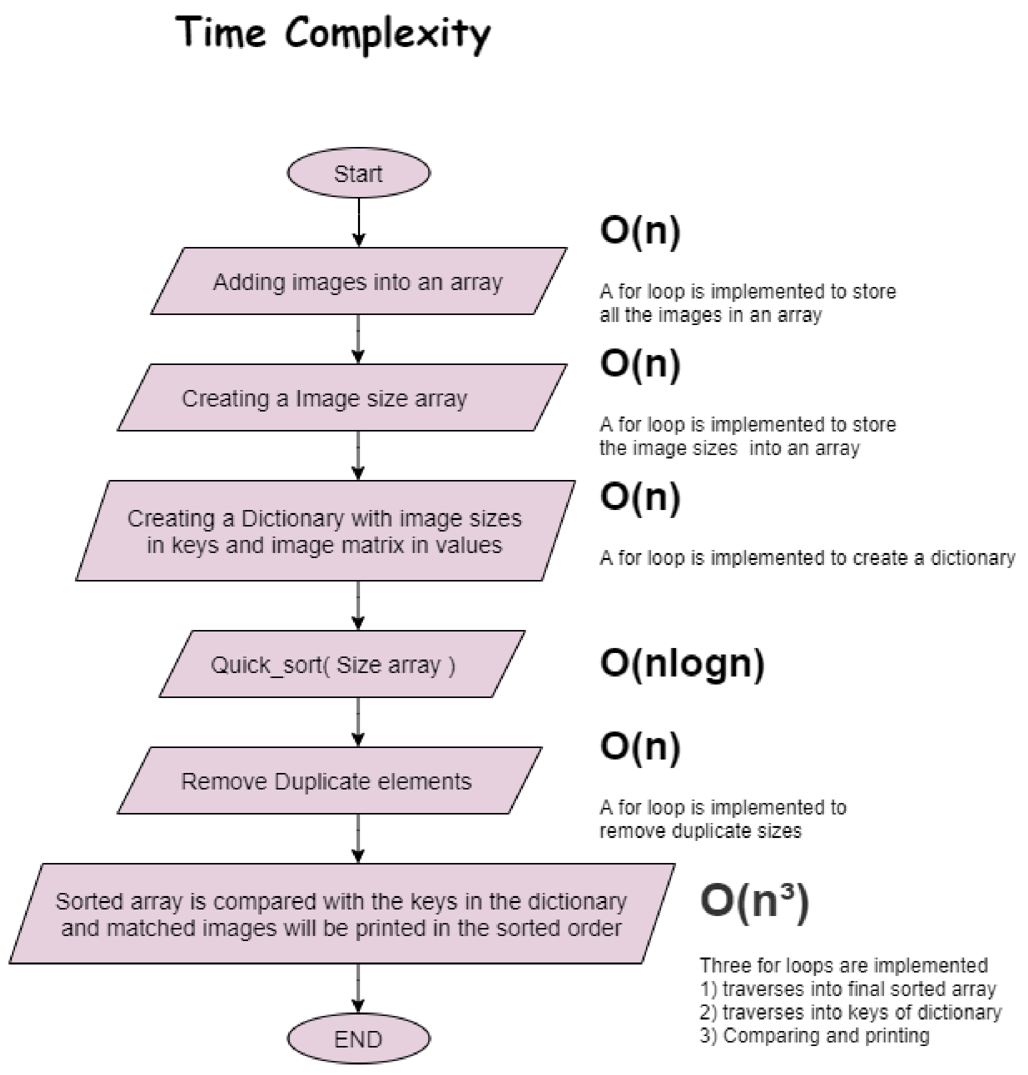
Step 6: Displaying sorted images

As we know the sorted size of images, we can display the images in sorted order easily using the created dictionary. From the sorted size list, we need to search for the key with the size in sorted list and retrieve the images from the corresponding list of values.

Searching whether the given image is present in the array or not:

Firstly, the size of the images is searched linearly . If the size of the new image is matched with any of the image size present in the array then we will compare the pixel values of the matched images. If difference of the pixel values is zero then we can say the new Image is found else it will print the image is not found.

Time Complexity analysis:



Time Complexity
Start
Adding images into an array
O(n)
A for loop is implemented to store all the images in an array
Creating a Image size array
O(n)
A for loop is implemented to store the image sizes into an array
Creating a Dictionary with image sizes in keys and image matrix in values
O(n)
A for loop is implemented to create a dictionary
Quick_sort(Size array)
O(nlogn)
Remove Duplicate elements
O(n)
A for loop is implemented to remove duplicate sizes
Sorted array is compared with the keys in the dictionary and matched images will be printed in the sorted order
O(n³)
Three for loops are implemented
1) traverses into final sorted array
2) traverses into keys of dictionary
3) Comparing and printing
END

Time complexity : $n + n + n + n\log n + n + n^3$

$$\boxed{n^3 + 4n + n\log n}$$

-----End of report----