VIRTUAL HACKATHON GROUP EXERCISE

A PROJECT REPORT

Submitted by

K. LAKSHMI GAYATRI (18BEC019)

N. NIKHIL KUMAR (18BEC031)

NIKHIL S AMBIGAR (18BEC033)

P. BHANU PRAKASH (18BEC035)

P. CHETHAN KRISHNA (18BEC040)

Y. SANTHI SWARUP (18BEC051)

in partial fulfillment for the award of Introduction to Algorithm (EC351)

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

DHARWAD

AUGUST 2020/DECEMBER 2020

TABLE OF CONTENTS:

PROBLEM STATEMENT:

Consider the following 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

APPLICATIONS:

1. In small scale and large scale industries, To sort the products based on various parameters like diameter, material shape etc.
2. If government thinks to develop a mobile applications of particular zoo or forest, then they can use our image sorting algorithm and store particular things under particular sections.
3. This sorting algorithm can be used in mobile application to sort images based on date, time and location.
4. It can be used in medical fields, To detect the scanned reports and sort them based on their sizes.(Ex: Scanned reports of heart.)
5. This algorithm can be used in marriage album creation by sorting the images of Bride and Bridegroom separately.

ALGORITHM TO TAKE IMAGE INPUTS INTO ARRAY & DISPLAY:

Step_1: START

Step_2: Declare f =[]

Step_3: Take integer input n

Step_4: Repeat until in range(n)

a = input(path of required image)

open the image and store it in x

store image size in a y

append [y,x] to f

print image

display image

Step_5: STOP

SORTING ALGORITHM:

Merge Sort (arr[], l, r)

If r > l

1. Find the middle point to divide the array into two halves:

middle m = (l+r)/2

2. Call merge Sort for first half:

Call merge Sort(arr, l, m)

3. Call merge Sort for second half:

Call merge Sort(arr, m+1, r)

4. Merge the two halves sorted in step 2 and 3:

Call merge (arr, l, m, r)

MERGE FUNCTION WORKS AS FOLLOWS:

Create copies of the sub-arrays L ← A[p..q] and M ← A[q+1..r].

Create three pointers i, j and k

i maintains current index of L, starting at 1

j maintains current index of M, starting at 1

k maintains the current index of A[p..q], starting at p.

Until we reach the end of either L or M, pick the larger among the elements from L and M and place them in the correct position at A[p..q]

When we run out of elements in either L or M, pick up the remaining elements and put in A[p..q]

SEARCHING ALGORITHM:

Based of no. of pixels present in image

Previously store all images pixels value in an array g[]

Step_1: START

Step_2: Input (path of image which you want search)

Step_3: Store size of that image in variable s2

Step_4: If s2 in g

Print (image is found)

else:

Print (image is not found)

Step_5: STOP

TIME COMPLEXITY OF SORTING ALGORITHM:

Whenever we divide a number into half in every step, it can be represented using a logarithmic function, which is log(n) and the number of steps can be represented by log(n) + 1(at most)

We perform a single step operation to find out the middle of any sub-array, i.e. O(1).

And to merge the sub-arrays, made by dividing the original array of n elements, a running time of O(n) will be required.

Hence the total time for merge Sort function will become n(log n + 1), which gives us a time complexity of O(n*log n).

Worst Case Time Complexity [Big-O]: O(n*log n)

Best Case Time Complexity [Big-omega]: O(n*log n)

Average Time Complexity [Big-theta]: O(n*log n)

TIME COMPLEXITY OF SEARCHING ALGORITHM:

Our search algorithm does linear search that it traverse the array from first element till the last element and it makes decision based on whether the element is present in array or not

Best case: O(1)

Worst case: O(n)

CODE IMPLEMENTATION:

```
from PIL import Image

# sorting algorithm

def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r- m
```

```
# create temp arrays
L = [0] * (n1)
R = [0] * (n2)
# Copy data to temp arrays L[] and R[]
for i in range(0 , n1):
    L[i] = arr[l + i]
for j in range(0 , n2):
    R[j] = arr[m + 1 + j]
# Merge the temp arrays back into arr[l..r]
i = 0     # Initial index of first subarray
j = 0     # Initial index of second subarray
k = l     # Initial index of merged subarray
while i < n1 and j < n2 :
    if L[i] <= R[j]:
        arr[k] = L[i]
        i += 1
    else:
        arr[k] = R[j]
        j += 1
    k += 1
# Copy the remaining elements of L[], if there
# are any
while i < n1:
    arr[k] = L[i]
    i += 1
```

```
        k += 1

    # Copy the remaining elements of R[], if there
    # are any
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1
# l is for left index and r is right index of the
# sub-array of arr to be sorted
def mergeSort(arr,l,r):
    if l < r:
        # Same as (l+r)//2, but avoids overflow for
        # large l and h
        m = (l+(r-1))//2
        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)
# searching algorithm
def search(x):
 if x in h.keys():
     print("image is found")
     f1['image'].show()
 else:
```

```
    print("\nimage is not found\n")
    print("the size of given image does not match with the size of images present
our images list\n")
    print("the given image is %d" %x)
# main function
f =[]
g =[]
t="Enter the number of images you want to insert : ";
n=int(input(t));
for i in range(n):
   j=i+1;
   a=input("Give the location of image %d :" % j)
   image = Image.open(a)
   [width, length] = image.size
   s1 = width*length
   g.append(s1)
   f.append([s1,image]);
   print(image)
   image.show()
# print images in list
print ("Given images list is")
print(f)
# sorting images using merge sort
mergeSort(f,0,n-1)
# print sorted images and display
print ("\n\nSorted array is")
```

```
print(f)
h = dict(f)
for key in h:
    k = h[key]
    k.show()
# searching for input image in given image list
print("All the images have been sorted based on thier size")
t1="Enter the path of the image you want to search :";
k=input(t1);
image2 = Image.open(k)
print(image2)
[width2,length2]=image2.size
s2 = width2*length2
f1 = {'size':s2,'image':image2}
a2 = f1['size']
search(a2)
```

THANK YOU!