Indian Institute of Information Technology and Management – Kerala (IIITM-K)

M.Sc. in Computer Science

Data Structures and Algorithms

(Project 2)

ENTITLED

(An Album of Abstract Art using Stack)

Submitted By:

Ravi Prakash

Group 1

M.Sc. Computer Science (Cyber Security)

Dated on: November 24th, 2020

ACKNOWLEDGEMENT

It is my privilege to express my sincerest regards to my project coordinator, ***Mr. Alex P James****, for his valuable input, able guidance, encouragement, whole-hearted cooperation, and constructive criticism throughout the duration of my group project. I deeply express my sincere thanks to* ***Miss Aswani A R*** *&* ***Miss Jessy Scaria*** *for being there for any adverse situation while the development of the project on the topic* ***"An Album of Abstract Art using Stack"*** *towards the Mini-Project 1 of the subject as the Data Structures and Algorithms. I take this opportunity to thank all my lecturers who have directly or indirectly helped in the project. Last but not least I express my thanks to my whole team for their cooperation and for supporting & each other throughout the development of the project.*

Introduction

We were provided with the following problem statements:

Problem Statement

"Images convey a lot of interesting information. In this project, you will take a photo of yourself and many different photos of your neighborhood (at max there should not be more than 10 photos). You might also want to keep the resolution of the images fixed. You need to write a program to read these photos and store them in a stack. You should be able to retrieve any of the images you want using an index value that you assign while storing.

Read randomly any n blocks of pixels from the image stack and create new images that contain blocks from all the images captured. Store these new images on a stack and display the created images to form an album of abstract art."

Required parts of submission:

1. Come up with a suitable solution to this problem within your group.
2. *You are expected to discuss the solutions and suggestions in the group chat. Arrive at the best solution with the group.*
3. *Write a report containing the codes of the project, and an explanation of why this particular solution was selected. Also include the screen capture of the results. The reports should be individually prepared and submitted.*

Planning

During the planning phase, we (the whole team) came across many questions and doubts. Som most frequent questions were:

Frequently Asked Questions:

1. How will we keep the count of maximum of 10 images?
2. What are the operations that should be applied for the stack class?
3. How many classes do we need to have for the solution?
4. On what basis, we are going to select the best solution?
5. Do we need some count dedicated function to get the n-th element from the image stack?
6. What format of images are we going to use for the solution?
7. What should be the resolution of input images?
8. What should be the resolution of output images?
9. Are we going to store the output as well in the stack?
10. How can we print the image without using any special library?
11. Can we print the binary file of the image?
12. How can we get any n-th element from the album?
13. How to get random n pixels from every image?

etc.

After finding the answers to these questions, we came up with many different solutions for the given problem.

Solution

We came up with many different solutions, which:

1. We kept a counter in the image stack class and tried to get the n-th image by traversing through the images available in the stack
2. We tried to read the random number (n) of pixels from the images
3. We tried to keep a dedicated method to print in the stack album
   and many more...

Due to many different cons, we left all of that and finally arrived at one optimal and best-suited solution for the given problem. Our solution has the following features:

1. We can load the provided images either ***one by one*** in the stack or we can give the path of our images' directory and the program will ***automatically*** import a maximum of 10 images.
2. Stack does not include any other feature than just main 3:
   a. push(node)
   b. pop()
   c. empty()
3. ***Extensive*** and smart use of ***stacks***
4. ***Good use of recursion*** hence gives the *best searching time complexity* for any n-th element from the provided stack of image
5. ***The best*** method of ***selecting any random number of pixels*** that always takes some pixels from every available / fed image
6. ***Flexible size*** of the album ***stack***
7. It's all up to the user - how many pics he/she wants to create in the new album of *abstract art.*
8. The program can ***automatically*** load the images, too.
   etc...

Classes Used

We have splitted our finally implemented program in 3 Python classes:

Structure of the NODE -

Parameters:	**Datatype:**
1. __image 2. __rawImage 3. __imageIndex (highlighted thing is a User-Input)	1. Location of Image 2. Image raw bytestring 3. Index of node (float)
Methods:	**Significance:**
1. Node() 2. __grabRawImage__(location) 3. getImage() 4. getIndex() 5. printNode()	1. Constructure 2. saves the image in raw format 3. returns image location 4. returns image index 5. prints details of the index

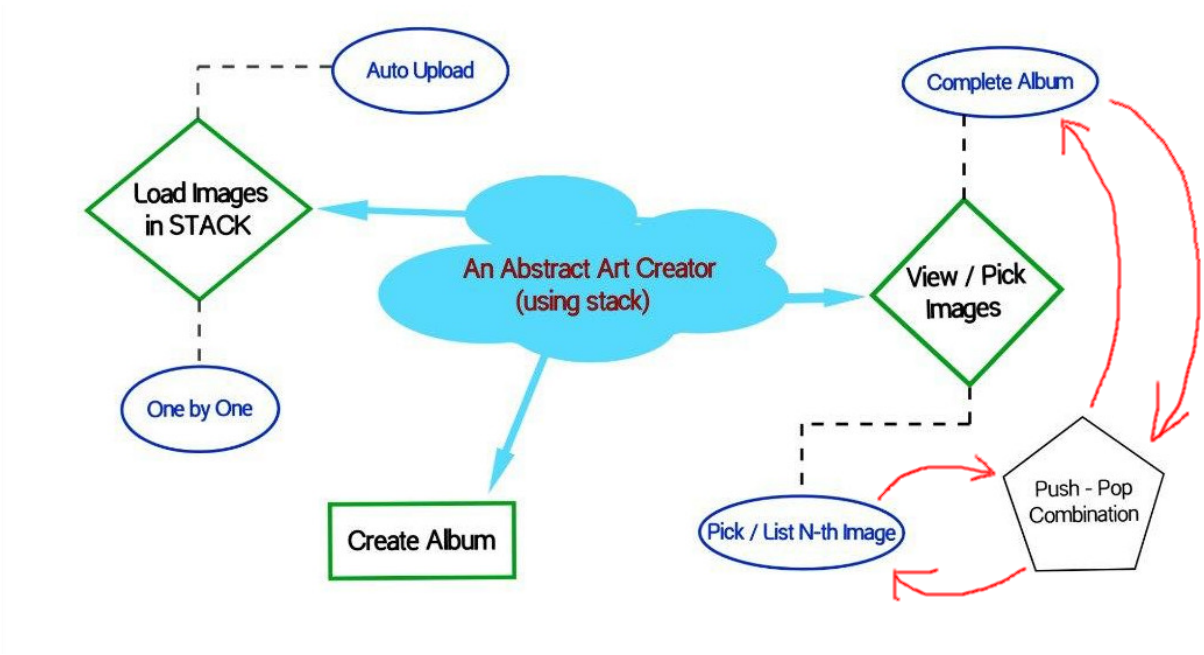
Structure of the imageStack -

Parameters:	**Datatype:**
1. __counter 2. Head 3. Tail	1. To limit max 10 entries 2. NODE 3. NODE
Methods:	**Significance:**
1. imageStack() Private Methods 1. __checkUnderflow__() 2. __checkOverflow__() Public Methods 1. push(node) 2. pop() 3. empty(emptyTo)	1. Constructure 1. Checks if any image exists or not 2. Restricts limit of images to 10 1. Adds an image to the album stack 2. Removes top image from album stack 3. Empties one stack into ‘emptyTo’

Structure of the imageProcessor -

Parameters:	**Datatype:**
1. imgDatabase 2. newAlbum	1. imageStack 2. imageStack
Methods: 1. historyTracker() Public Methods 1. loadImages() 2. grabNthImage() 3. getAllPixels() 4. getFewPixels() 5. createNewPic() 6. createAlbum() 7. saveJPEGImage() 8. printAlbum()	**Significance:** 1. Constructure 1. Loads provided images 2. Picks any n-th element from album 3. Grabs few pixels randomly from all of the pixels 4. Grabs few pixels randomly without special libraries 5. Creates one abstract art 6. Calls createNewPic() multiple times 7. Saves the abstract locally 8. Prints details of available images in an album

Program Flow



We can broadly identify our solution to have the following main functionality:

- Loading Any Images / Images
- Creating New Album
- Storing / Picking / Viewing Details of any n-th Image

All the above functionality internally implement:

1. Push (to add images to the stack)
2. Pop to view any image's detail or pick n-th element
3. Push-Pop-Empty Combination to get n-th element

Main-Menu

```
Main Menu:
----------

1. Load Raw Images
2. Create New Album
3. Grab n-th Image
4. Print Loaded Images
5. Print Newly Created Album


** Any other key to exit!!


Enter your choice: 1
```

So, for all these different features, we need a structure like the below:

Load images

1. One by one
2. Auto upload from any directory

Create album

- calls <create 1 pic> module multiple times
- calls <get n-th image> module internally

1. maximum images allowed 10
2. minimum images allowed 1
3. Needs a method to grab any n random pixels in sequence from every element

View and list the available images

1. From loaded folder
2. From a newly created album

Pick/list n-th image from the albums

1. implements push()
2. implements pop()
3. implements empty()

Some Main Algorithms

1. List Images' Details in an Album Stack -

*# order proves that data is saved in **STACK***

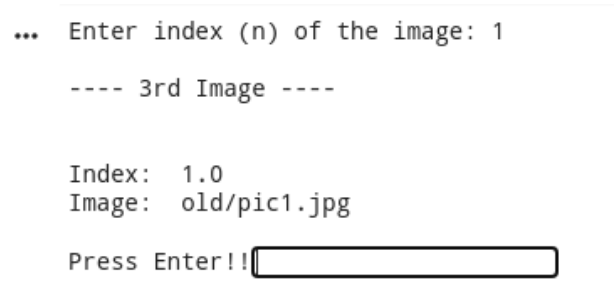
```
...
    ---- Loaded Images ----


    Index:  10.0
    Image:  old/pic1.jpg

    Index:  9.0
    Image:  old/10.jpg

    Index:  8.0
    Image:  old/5.jpg

    Index:  7.0
    Image:  old/8.jpg

    Index:  6.0
    Image:  old/6.jpg

    Index:  5.0
    Image:  old/2.jpg

    Index:  4.0
    Image:  old/9.jpg

    Index:  3.0
    Image:  old/1.jpg

    Index:  2.0
    Image:  old/7.jpg

    Index:  1.0
    Image:  old/4.jpg

    Press Enter!!
```

Algorithm

```
# uses stack functions only

START
tempStack = imageStack()

if masterAlbum.head is None:
  print("Album is empty!!!")

while True:
   popped = masterAlbum.pop()
   if popped:
      tempStack.push(popped)
      popped.printNode()
      im = Image.open(popped.getImage())
      im.show()
   else:
      break
   END IF
END WHILE

tempStack.empty(masterAlbum)
STOP
```

2. Getting n-th Image from the album stack -

This function is internally implemented in creating the album



Algorithm

uses stack functions only

START
declare imageStack **temp**
counter = 1
START WHILE
  while counter <= n:
        popped = **pop()**
        push(popped**)** in **temp**
END WHILE
 # empty temp back in the main stack
temp.**empty(main)**
return popped
STOP

3. Getting n randomly n pixels -

Algorithm

START
rawImg = open(file)
allPixels = size.rawImg(length x width)
grabPixels = allPixels / **randNum(**1,9**)**
startPixel = randNum(0, allPixels - grabPixels)
endPxiel = startPixel + grabPixels

grabbed = rawImg.Pxl range(**startPixel, endPxiel**)
return **grabbed**

STOP

Outputs

Loading Images:

 # loading only one image (manually)

```
<----   Images must either be of JPG or JPEG in format  ---->


1) Load an image by location
2) Auto upload images from a directory
Enter your choice: 1
Enter the location of directory of images (JPG/JPEG): old/pic1.jpg
```

 # loading automatically from specified directory

```
<----   Images must either be of JPG or JPEG in format  ---->


1) Load an image by location
2) Auto upload images from a directory
Enter your choice: 2
Enter the location of directory of images (JPG/JPEG): old

Press Enter!!
```

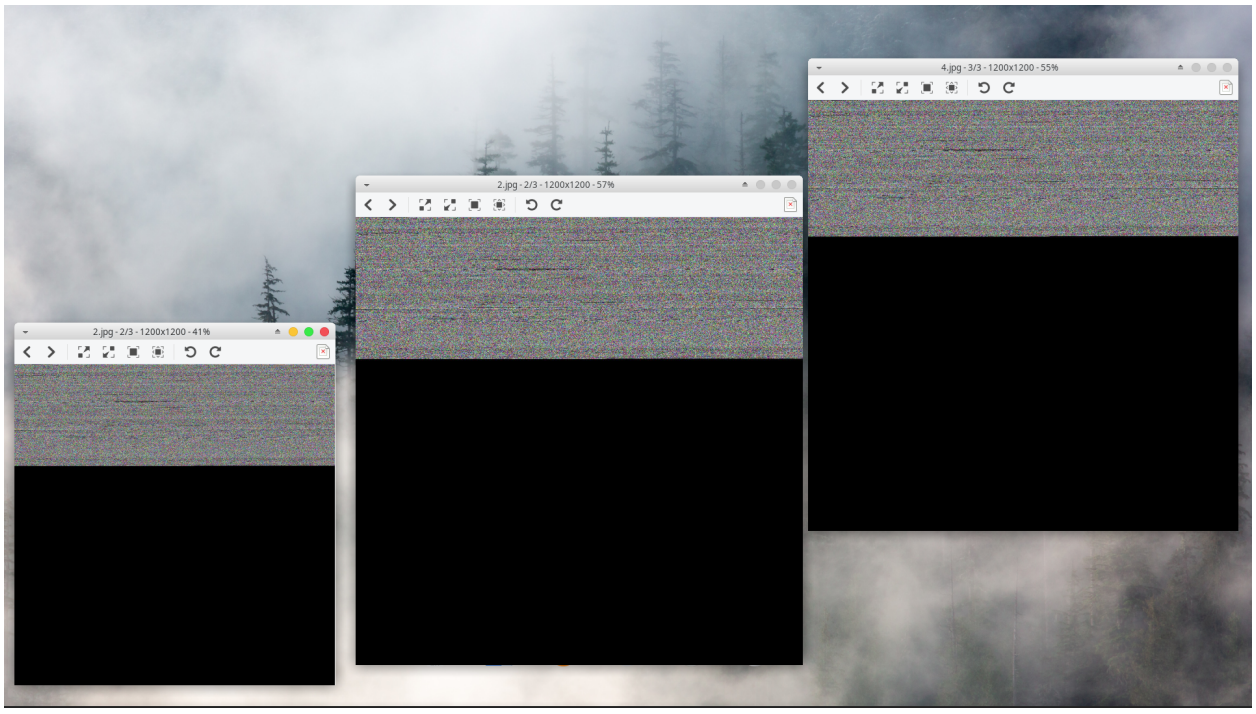
Few (6 of 10) Images that I used in this demo:
(I took them from an Instagram poetry page @says.my.soul)

Creating Album:

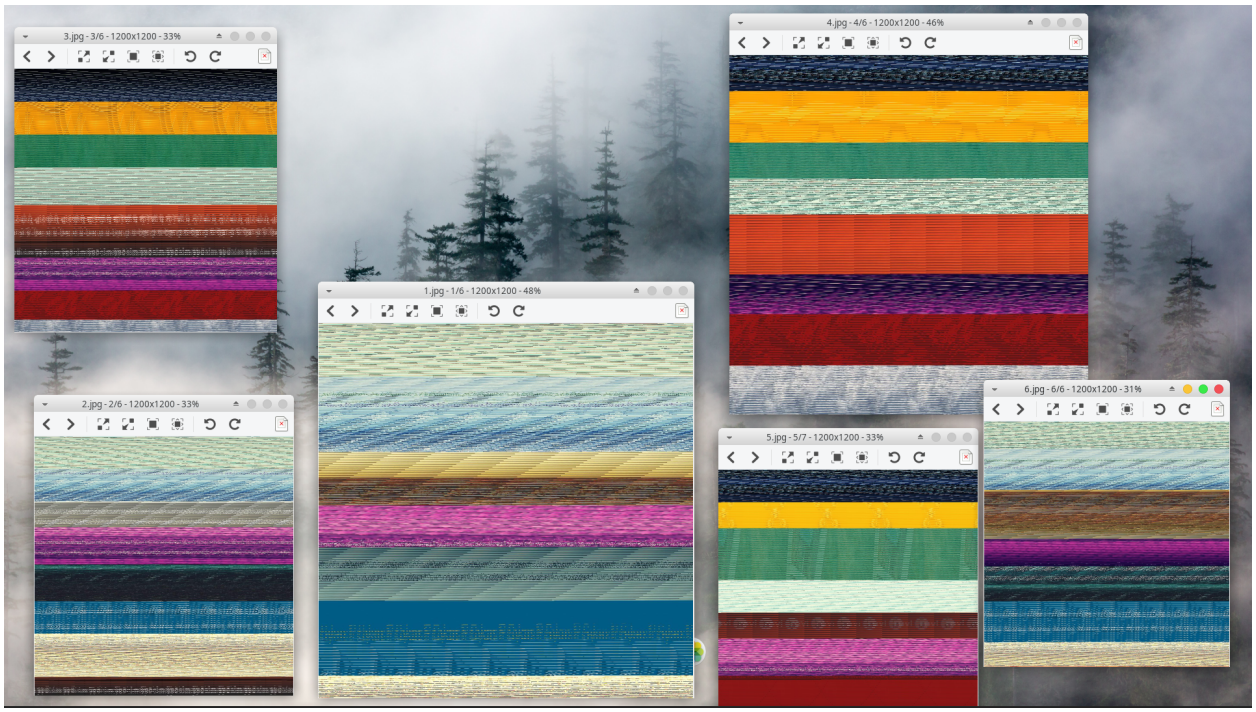
```
...
    Main Menu:
    ----------

    1. Load Raw Images
    2. Create New Album
    3. Grab n-th Image
    4. Print Loaded Images
    5. Print Newly Created Album


    ** Any other key to exit!!


    Enter your choice: 2
    Enter number of images you want to create in the new album (max 10): 6
```

```
...  Enter number of images you want to create in the new album (max 10): 6

     Hold on! Your album of 6 images is being created...
```

*# following **3** images are created by our function **without using any special library***



*# following **6** images are created by our function where **pixels** were extracted **using PIL** library*



It was very challenging to extract pixels without using any library, still we managed to do that at a lot much extents (excepting in case of few special images).

Listing Images in New Album (of abstract art):

```
...
    ---- Newly Created Album ----


    Index:  6.0
    Image:  newAlbum/6.jpg

    Index:  5.0
    Image:  newAlbum/5.jpg

    Index:  4.0
    Image:  newAlbum/4.jpg

    Index:  3.0
    Image:  newAlbum/3.jpg

    Index:  2.0
    Image:  newAlbum/2.jpg

    Index:  1.0
    Image:  newAlbum/1.jpg

    Press Enter!!
```

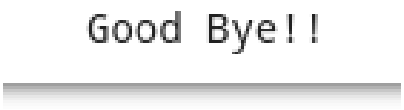
Endnote

Few major pros of our program:

1. There is **no restriction** for the input image **resolution**
2. Every developed image has an **inbuilt copyright** in encrypted format (that can be seen in the bytestring of images i.e. abstract arts after converting them into *.ppm* format)
3. Awesome user friendly interface.

Exiting:

Program greets back to the user on exit as below:



A **running instance** of the given solution is available at the following link :

https://github.com/ravi-prakash1907/Data-Structures-and-Algo/blob/main/Submissions/Group%20Projects/Project2/Abstract_Art_Album_Generator.ipynb

Happy Testing!!!