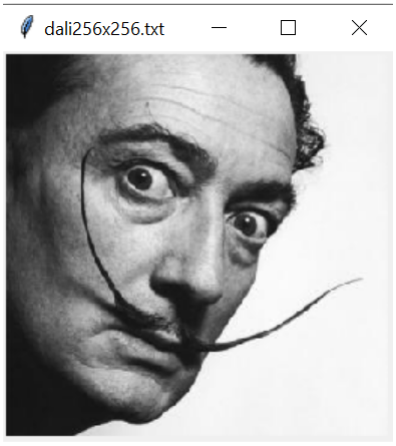
Computational Problem Solving CSCI-603
Image Viewer Lab 8

10/21/2023



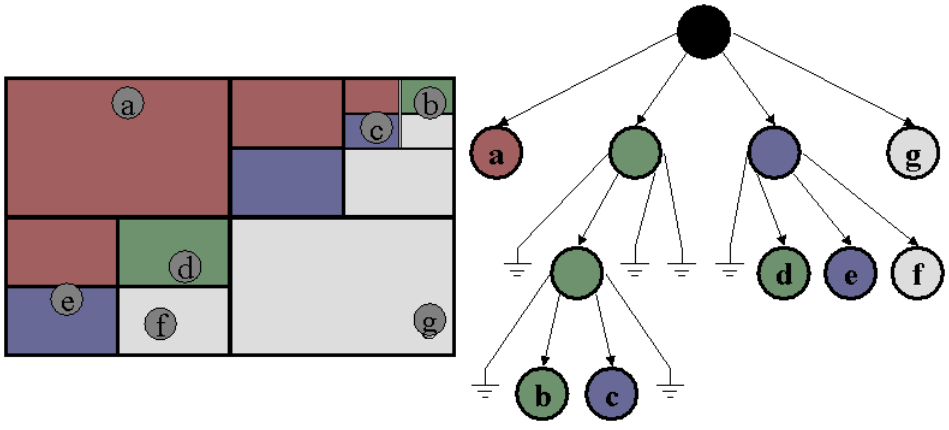
1 Introduction

For this homework, you will implement a program that reads in a compressed image file and visualizes it. All the images are grayscale images whose 8-bit pixel values vary in intensity from 0 (black) to 255 (white).

The images files have been compressed by an algorithm that uses a tree structure known as a quadtree. This algorithm takes advantage of spatial coherence. That is, there is a high degree of likelihood that the value of a given pixel is the same as the values of its neighboring pixels. This allows us to recursively subdivide the image into regions that share the same pixel value.

A quadtree is a tree data structure where each internal node contains exactly four children, and each leaf node represents a single value. It is used to partition the image space into four equally sized, square quadrants.

Your program must read in the file, uncompress it and display the resulting image.



2 Implementation

You will implement a GUI application that read a compressed image file, uncompress it and draw it using the Python interface to Tcl/Tk called `tkinter`.

2.1 Starter code

Download the starter code from here. Here is a brief description of what is provided to you:

1. **images**: A folder containing uncompressed and compressed image files for testing.
2. **src.qtnode**: A class that represents a node in the quadtree.
3. **src.qtexception**: A custom exception for communicating errors with operations involving the quadtree and the files it uses for uncompressing.

2.2 Main program

You will implement a main program called `image_viewer.py` that receives the image file as a command-line argument. The file may come in two different formats: uncompressed (raw file) or compressed.

If the program is run with an incorrect number of arguments, display a usage message and terminates the program.

```
Usage: python image_viewer.py [-c] <filename>
    -c Reads in a compressed image file. If this option is not present,
        the file is considered to be uncompressed.
```

2.2.1 Uncompressed file format

The format of an uncompressed file is very simple. Each line contains a integer value representing a grayscale tone in the range of 0 (black) to 255 (white). The values go from left to right and top to bottom for the image. You may assume the image is guaranteed to be perfectly square.

You are being provided with a collection of uncompressed image files in the `images/uncompressed` directory. Be warned that with the smaller images you will most likely not be able to see them when rendered.

For displaying the image of an uncompressed file, you must run the program as follows:

```
$python image_viewer <filename>
```

2.2.2 Compressed file

The first line in a compressed image file will be the number of pixels in the image. The remaining lines will be the values we get when doing a preorder (parent, left, right) traversal

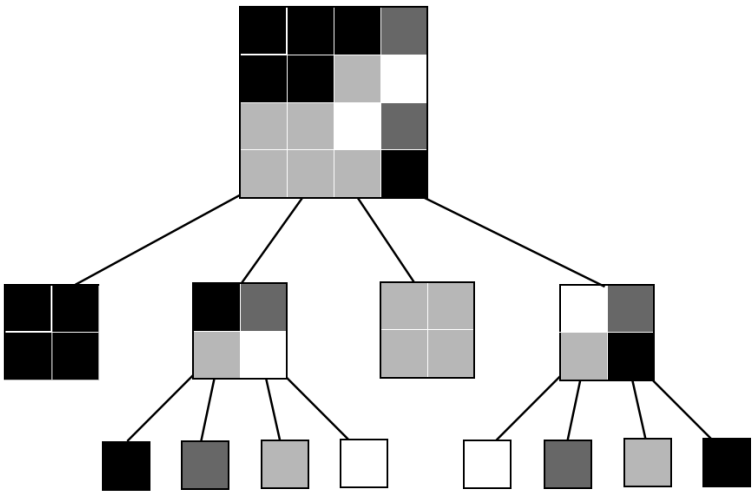
of the tree. We will use -1 to indicate an internal node, and the region's pixel value, 0-255, as the leaf nodes.

Consider the raw image file `simple4x4.txt` and its quadtree representation:

simple4x4.raw:

```
0
0
0
85
0
0
170
255
170
170
255
85
170
170
170
0
```

quadtree representation:



Its corresponding preorder traversal and compressed image file are the followings:

preorder traversal: -1, 0, -1, 0, 85, 170, 255, 170, -1, 255, 85, 170, 0

simple4x4.rit:

```
16
-1
0
-1
0
85
170
255
170
-1
255
85
170
0
```

For displaying the image of a compressed file, you must run the program as follows:

```
$python image_viewer -c <filename>
```

2.3 Design

You should create a class that represents your quadtree data structure. The quadtree should provide the following funtionalities that will be used by the main program:

- Read an uncompressed file into a raw image that will typically be a two dimensional array of integers.

- Read a compressed file, build its corresponding quadtree and parse it into a raw image.
- Create and display the string representation of the quadtree by doing a preorder traversal of the tree.

2.4 Error Handling

Your program must deal with the following errors:

- If the command line argument is not present, display a usage error and exit.
- If the file does not exist or is not readable, display an error message and exit.
- While processing a compressed image file:
  - If the image size is not square (a power of two), display an error message and exit.
  - If an integer value is encountered for a pixel value that is outside the range 0-255, display an error message and exit. Note: -1 is not a pixel value, it used for identifying internal nodes.
  - If a non-integer value is encountered for a pixel value, display an error message and exit.

  All these errors must be reported by throwing an exception of type `QTException` with the appropriate description of the problem.

About uncompressed images files. If the uncompressed file is present, its contents are assumed to be valid and do not need to be error checked.

2.5 Output

If there are no errors and the image file passed in is compressed, the program must display to the standard output the generated quadtree, in preorder fashion. For example, with the `simple4x4.rit` compressed file the output would be:

```
Uncompressing: simple4x4.rit
Quadtree: -1 0 -1 0 85 170 255 170 -1 255 85 170 0
```

Notice that your program do not have to display anything to the standard output for uncompressed image files.

2.6 Working with tkinter

Python provides a tkinter package which provides some basic GUI widgets. We will use this library to draw the images pixel by pixel.

The basic steps for displaying the image are:

- Create a window by initializing an instance of the `Tk` class.
- Create a new `PhotoImage` that is the square dimension of the raw image.
- Loop over the grayscale values of the image.
  - Create the hexadecimal representation of the color whose red, green and blue values are all the same value for this pixel.

  - Put the fill color into the `PhotoImage` at its corresponding coordinate.
- Create a `Canvas` with the square dimension of the raw image.
- Add the `Canvas` to the window by using the `pack()` method, e.g. `canvas.pack()`.
- Add the `PhotoImage` to the `Canvas`.
- Call the `mainloop()` method in the window to keep it opened.

3 Grading

Your grade will be determined as follows:

- 20%: Problem Solving
- 10%: Design
- 65%: Implementation
  - Error handling 10%
  - Visualize uncompressed image file 10%
  - Build quadtree from a compressed file 20%
  - Visualize image from a quadtree 20%
  - Display quadtree in preorder fashion 5%
- 5%: Code Style and Documentation

4 Submission

Create a ZIP file named Lab8.zip with your all your source code. Submit the ZIP file to the MyCourses assignment before the due date (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this lab).