KING FAHD UNIVERSITY FOR PETROEULM AND MINERALS

College of Mathematics & Computing

Information & Computer Science Department

ICS202 – Data Structures and Algorithms

ICS202 – Course Project: Quad-Trees

1- Introduction

Quad Trees is a critical data structure involved in various fields including image processing and computer graphics. It could be used as an image compression tool especially for images that have big single-colored areas.

In this project the main objective is to construct the main transitions from a quad tree to an image and vice versa. Knowing this, the following is each task and the corresponding problem-solving strategies and the faced challenges regarding solving the problem.

2- Problem-Solving Strategies and Faced Challenges

a. Task 1: Complete QTNode Class

i. **Problem-Solving**: create no-arg constructor with intensity -1 and null value for the node children, another constructor with intensity parameter, instance variables and since each

node has intensity variable and 4 nodes, create 2 setters for the children and the intensity variable, and a getter of the intensity, and finally override the toString() function.

ii. **Challenges**: no challenges faced since the required task at this point is a repeated concepts of Object-Oriented-Programming.

b. Task 2: Complete The Constructor of ImgQuadTree.java and buildQT() Method

i. **Problem-Solving**: creating a constructor for the file started with exception handling for creating Scanner object. Then initialize the root node with the whole tree using a method called buildQT() to enhance encapsulation, then close the input in order to prevent data leak.

BuildQT() method is a recursive method that build a quad tree by checking the next input of a preorder traversal of quad tree file called walkingtothesky.txt. If the next input of the file is -1 then create a node and call 4 recursive calls for the node children, otherwise create a node with the intensity of the next input.

ii. **Challenges**: the hardest part of the problem is constructing the recursive call of the buildQT() method.

c. Task 3: Building getNumNodes, getNumLeaves, getImgArray.

i. **Problem-Solving**: getNumNodes: is a recursive method that call a helper with a node parameter in order to return call of the nodes children plus one which is the current node.

   getNumLeaves: is also a recursive methods that call a helper method with a node parameter which checks if one of the node's children, since a node can only have 0 or 4 nodes, if the child is null then return one else call the method recursively for each child.

   getImgArr: it is a recursive method which has 5 parameters: instance array of the object, x-coordinate, y-coordinate, dimension of the specified area and the current node. Firstly, the method starts with checking the intensity of the node, if it -1 call 4 recursive methods to each quadrant by adjusting x, y and dimension values, else if the intensity is not -1 then use a double for-loop in order to fill the specified region.

ii. **Challenges**: constructing the getImgArr was the hardest part since the dimension changes in each recursion and also the quadrant changes but, however the main array is the same and it is only a dimension manipulation.

d. Task 4: Create ImgQuadTreeCreator.java file:

i. **Problem-Solving**: the file should convert the image pixles values to a preorder traversal of a quad tree of the image. Firstly, create a constructor that takes the input and output file names as parameters. Then, by using exception handling, initialize the input and output objects defined at the top. Finally, call buildArrFromImg method then

outputQT methods in order to enhance encapsulation and minimize the code.

buildArrFromImg: it gets a scanner object of the file intended and then do double for-loop to populate the array with input text which is the first 256 pixels is the first row. Then close the input to prevent data leak.

outputQT(): is a recursive method that takes the x coordinate, y coordinate, the dimension of the region and the PrintWriter object, then check if the region has the same intensity then print the intensity otherwise print -1 and call the function for the 4 children with each modified parameters according to the quadrant of each node.

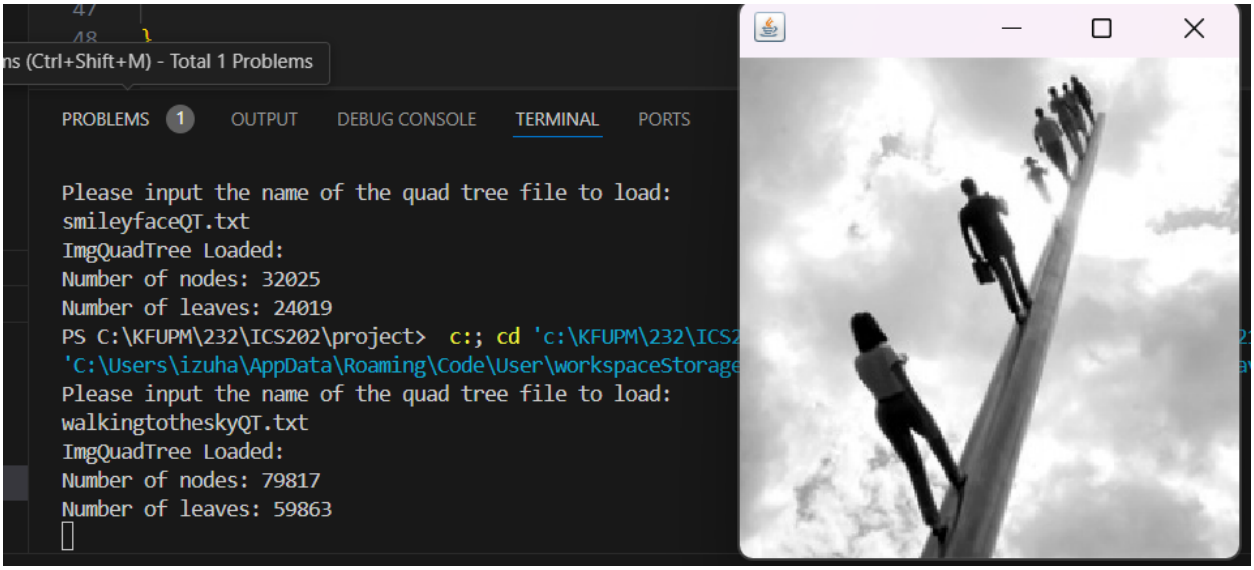
In the main method ask the user for the file name and then process the name to be in the format of "*filename*QT.txt" and pass them to the constructor of the class.

ii. **Challenges**: outputQT was the hardest point since it is recursive and took time to figure out the best way to construct it.

However, the main issue that was faced is that all nodes were not printed despite having the print statements running the same number of times as the number of nodes. After deep investigation, the problem was that the output object was not closed after the method finished which caused data leak.

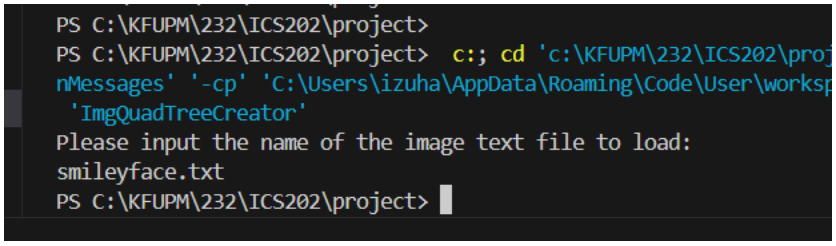
3- Test case screenshots

a. Converting from Quad Tree File to an Image

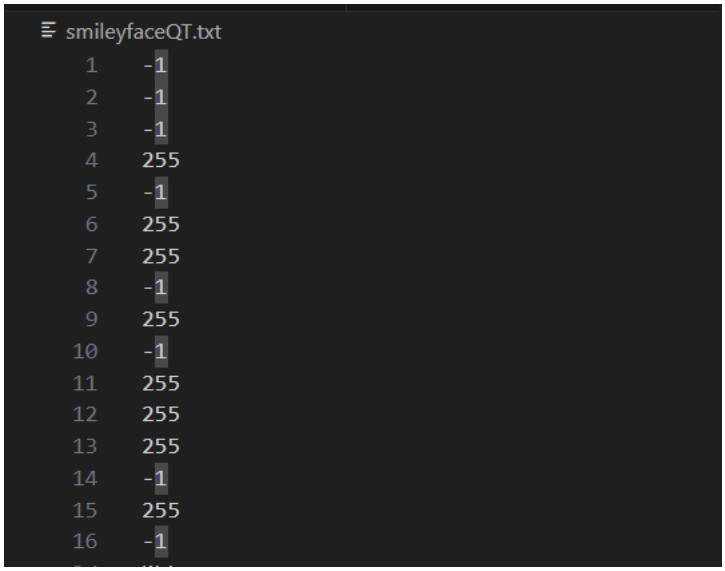


Running ImgQuadTreeDisplayer.java, insert “walkingtotheskyQT.txt”

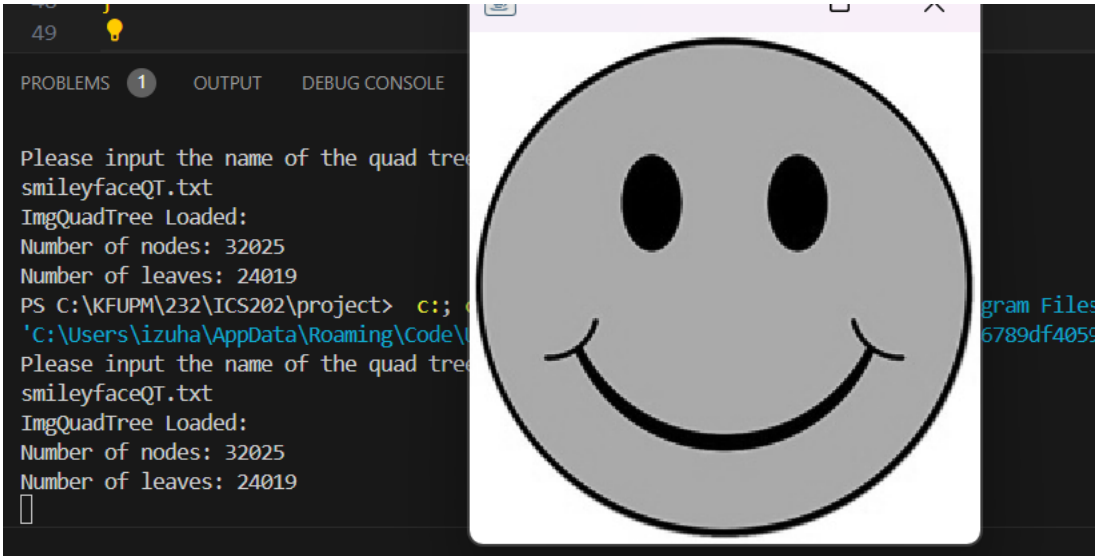
b. Converting From Image to a Quad Tree



Running ImgQuadTreeCreator.java, insert “smileyface.txt” image getting “smileyfaceQT.txt”



"smileyfaceQT.txt" file



Running ImgQuadTreeDisplayer.java, insert "smileyfaceQT.txt"
