EEM 480 Homework-3
Report

The homework was developed in Apache NetBeans IDE 12.0 using JDK8-301. The total Beyazıt_Isık_HW3 package contains 9 Java classes with main class and 1 java interface declaration. Some java files consist of 2 java classes.

In the main class there is a "hand" object declaration from Handlerclass. With the "hand" object I called methods which are ReadShapeFile(), OutputShapes() and OutputShapesToFile(). In the data structure, there is a linked list and a queue data structure which generated from a linked list. CustomLinkedList is a linked list that holds "temp_matrix"which is a char 2D array. The second one is a queue list that holds visited index values.

```
public class Handler implements HW3_Interface
```

In the constructor block finder, linkedlist objects created respectively from MatrixFinder, CustomLinkedList classes. Blank integer matrix [11][14] and string matrix_file_object created.There is a pre-defined interface used in the Homework-3.pdf.

1. `public void ReadShapeFile(String path)`
   Created a "reader" object from the ReadandParse class with String path given in main class. From reader object get_matrix method used to get the matrix value from file as an integer 2D array form. There is no loop in this method so that complexity is Θ(1).
2. `private void OutputShapes()`
   Uses the finder object to use ShapeFind method. It gets the linkedlist object created in finder object. Then, updates the matrix_file_string variable with the printList method's return variable. There is no loop in this method so that complexity is Θ(1).
   public void OutputShapesToFile(String path)
3. `public void OutputShapesToFile(String path)`
   Uses the linkedlist object returned from previous method to print matrices to the file with given path and returned string from previous method. There is no loop in this method so that complexity is Θ(1).

```
class MatrixFinder
```

In the constructor block 2D char array temp_matrix and linkedlist object created respectively from char and CustomLinkedList classes. MatrixFinder is a region finder with breadth-first search algorithm.

1. `boolean isSafe(int mat[][], int i, int j, boolean vis[][])`
   Check if the matrix index values "i and j" overflow the matrix. If the values dont overflow and the matrix with that indices equals to one and not visited returns true otherwise returns false. There is no loop in this method so that complexity is $\Theta(1)$.

2. `void Search(int mat[][], boolean visit[][], int visiti, int visitj)`
   Searches the matrix index's neigbours. Generates a "queue" object from Queuex class. Enqueue the new object from ikili class for storing the visited index values. In the while loop checks if the queue is empty or not. Peek method gets the ikili type object in front of the list and takes the first and second indices. Inside the while loop there is a k loop where checks all the neighbours are safe. If they are safe store the char temp_matrix with the value of "*". Then stores the Boolean type visit matrix true that means this index already visited don't check twice. Adds the indices to the queue. There is a nested loop in this method so that complexity is $\Theta(n^2)$.

3. `public Object ShapeFind(int mat [][])`
   Creates visit Boolean matrix for storing the visited values. Res variable stores the regions found but it is not used in the code it is for debugging purposes. Checks all the matrix indices and push them to the Search method in the class. Temp_matrix variable gets reset for other uses after appended to the linkedlist. Linkedlist object returned to the handler class. There is a nested loop in this method so that complexity is $\Theta(n^2)$.

`public class CustomLinkedList`

In the constructor block head object created from Node class, a linked list which holds the char 2D arrays. Node class is a class that holds data and link respectively from char and Node classes. An object that can be created from CustomLinkedList class is a linked list that holds all the 2D char arrays in the database.

1. `public void Append(char[][] new_data)`
   Node created and the data which is char type put inside the node. If the Linked List is empty, then the new node set as head. Return statemens used so that the code will exit after if. If the previous statement is not verified then new_node's pointer will set as null then last node will be created to find last object in the linkedlist. Finally, the last will point to the new node. In worst case the method goes to the nth node. There is a single loop in this method so that complexity is $\Theta(n)$.

2. `public String printList()`
   String variable created and returned with the returned valur from printMatrix method. String variable concatenates in every loop and a new shape added to it. This methods iterates in the nodes so that all the shape can be pushed in to the printMatrix method. There is a single loop in this method so that complexity is Θ(n).

3. `public String printMatrix(char char_matrix[][])`
   Min, maxenter, temp, entercount, string and exit_flag variables created. In the first nested loop all the indices checked and the min values and entercount values hold. In the second nested loop the loop iterates from entercount to the maxenter value. When the "*" value confirmed (backslash b) printed to remove blank spaces. This part of code is a little buggy for in below rows. It works fine with the top row. There are two nested loops in this method so that complexity is $\Theta(2n^2)$.

4. `public void MatrixToFile(String path, String matrix_file_string)`
   Takes a string object and create a filewrite object from FileWriter class. Filewrite.write operation used with the string to print out a txt file filled with shapes. There is no loop in this method so that complexity is Θ(1).

`public class ReadandParse`

In the constructor block myfile and reader objects created respectively from File and Scanner classes. Myfile is generated with the path given in object creation part. There are more methods written in the class file, but they are in comment. They were for debugging purposes.

1. `private String get_line()`
   Uses the reader object to use nextLine method. It gets the lines one by one and return the data every time the method called. There is no loop in this method so that complexity is Θ(1)
2. `public int[][] get_matrix()`
   Nested for loop used in every row for getting the data from get_line method then parses the data with split method from string class. Parts from that method stored in String array called parts[]. For every column parseInt method used from Integer class to parse the values in part[j] index. Returned value stored in temp_value variable and pushed in to the integer 2D array in the class clad matrix. There is a nested loop in this method so that complexity is $\Theta(n^2)$.

`public class QueueNode<E> and class Queuex<ikili>`

In the constructor block of QueueNode<E> generic type of key variable created, and the next variable holds the next QueueNode type.
In the constructor block of Queuex <ikili> accepts a ikili type and constructs the front and rear of the queue as null.

1. `void enqueue(ikili key)`
   Adds a ikili type of key to the rear of the queue. There is no loop in this method so that complexity is Θ(1).
2. `void dequeue()`
   Returns the front node in queue. And iterates the queue.There is no loop in this method so that complexity is Θ(1).
3. `public boolean isEmpty()`
   Checks the front and rear if any of them is null or not. If null return True otherwise False.There is no loop in this method so that complexity is Θ(1).
4. `public ikili peek()`
   Returns the front node in queue without removing it from the queueThere is no loop in this method so that complexity is Θ(1).