Gebze Technical University
Computer Engineering

CSE 222 - 2019 Spring

HOMEWORK 5 REPORT

Hamza YOĞURTCUOĞLU
171044086

Course Assistant: Özgü GÖKSU

1 INTRODUCTION

1.1 Problem Definition

- Loading a pixels of picture in 2D array that must have 3D Vector.
- Pixel Class must keep more variable or just 3 variables
- Underlying PriorityQueue must use implementation of your priority queue.
- How to decide MaxPixel when inserting or extract (root) a Pixel
- A node has a left , right child or parent where they are . (Equation)
- How to comparators would been releated with PQLEX, PQEUC and PQBMX (Different comparated Max Heaps)
- How to apply Lexicographical comparison (LEX) to a MAXHEAP
- How to apply Euclidean norm based comparison (EUC) to a MAXHEAP
- How to apply Bitmix comparison (BMX) to a MAXHEAP
- Properly all comparison situation with Queue
- If threads will run with synchronized , how is it used in java for synchronize for all 4 threads
- Some threads don't deal with same source at the same time. This problem must solve.
- All extracting threads don't extract any Pixel in a queue until inserted 100 Pixels to 3 Unique queue by Thread-1
- Then other 3 threads must start for extract Pixel according to their comparison.
- If any size of queue would be '0' , which extracting thread must be wait until thread-1 inserts a new Pixel.
- Then Inserting new Pixel , waiting thread must continue.
- Inserting thread that is thread-1 when quequ is full , it must wait too. But other thread should send a signal for wake up.
- All Missions of Threads are finished. Threads should exit.

1.2 System Requirements

+ Total of Algoirthm Complexity is work with O(n) (Meaning is depend of your data (Pixels in Image)).

+ All Operating Systems handle this program.

+ Doesnt need a lot of memory acually if have alot of data , memory usage will raise linearly.

+ Program can work 128KB of memory (That can be change your data.)

+ This program doesn't make properly in smartphone it can be work in computer.

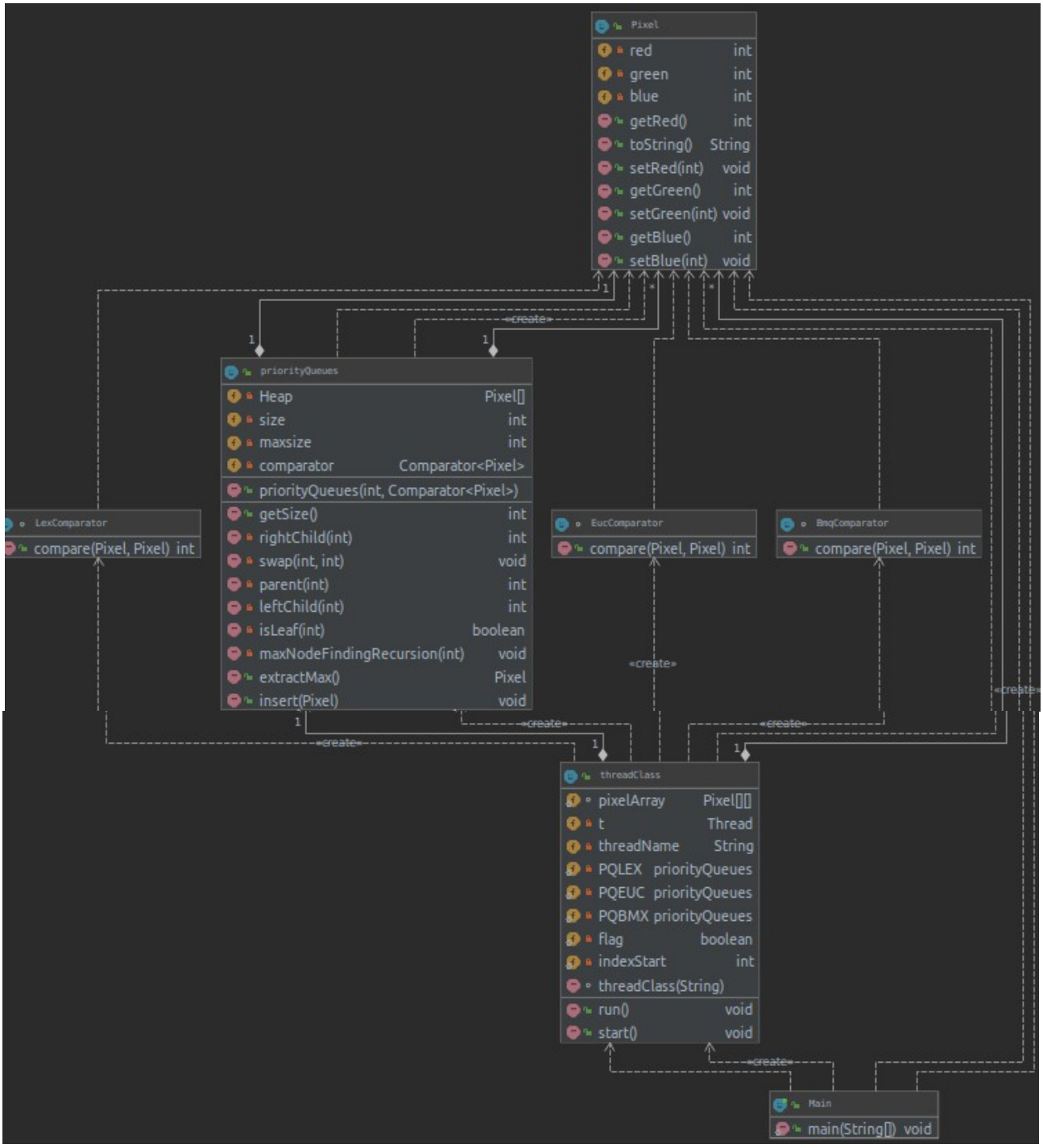
+ Doesn't need a specific of hardware just a computer work.

+ Just take executable file for directly execute program. Then , it works efficently.

2 METHOD

2.1 Class Diagrams

PIXEL: PriorityQueue one to many relation. Pixel can be created many times.
PRIORITYQUEUES: PriorityQ holds all Pixels
LEXCOMPARATOR: Just usage for comparating
EUCCOMPARATOR: Just usage for comparating
BMQCOMPARATOR: Just usage for comparating
THREADCLASS: In created object in main 4 times. Each queue has a thread.



2.2 Use Case Diagrams

User : User just have to real pathname of image such as command line argument.

Then all pixels are uploaded to 2D array (WxH)

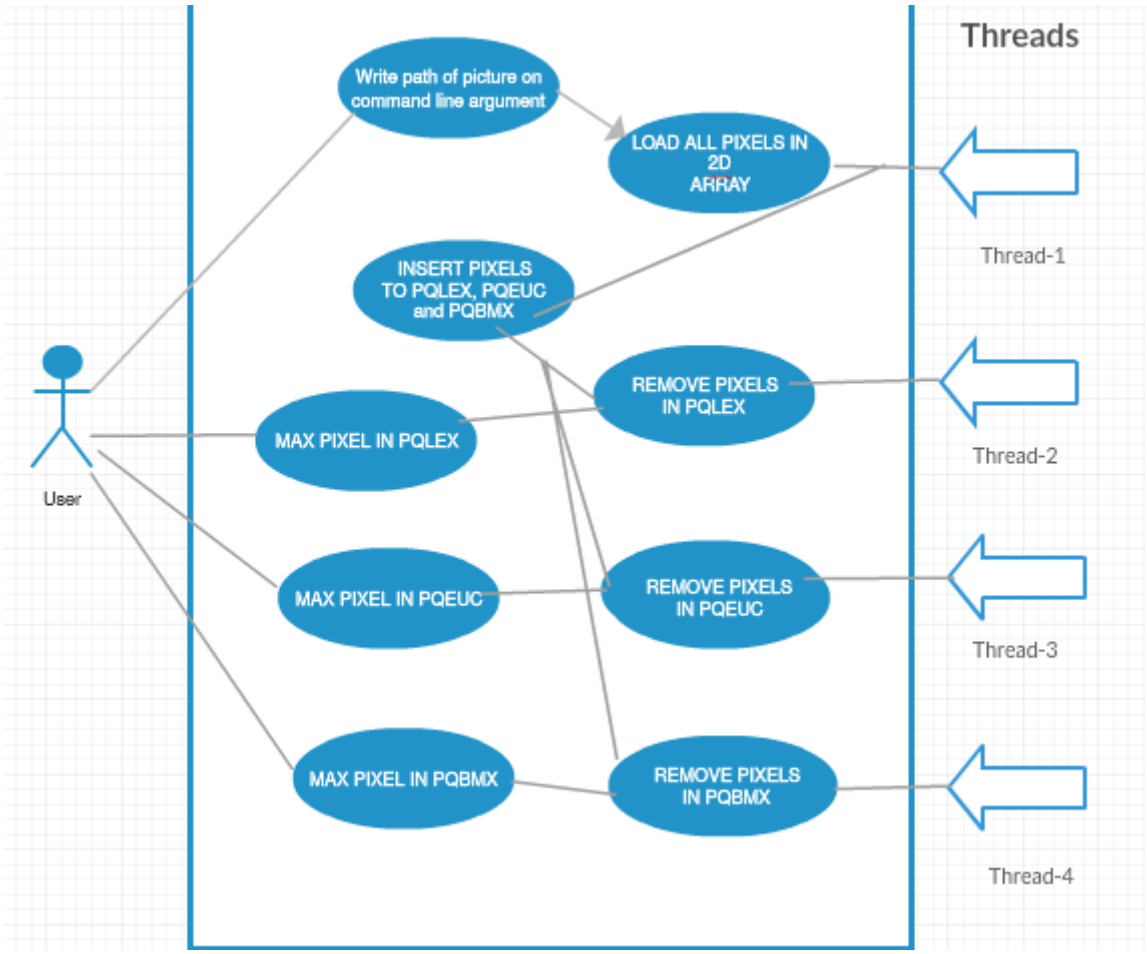
Thread-1 : Thread-1 insert the all pixels in to PQLEX , PQEUC , PQBMX

Thread-2 : Remove the pixels according to comparison LEX until all pixels are removing.

Thread-3 : Remove the pixels according to comparison EUC until all pixels are removing.

Thread-4 : Remove the pixels according to comparison LEX until all pixels are removing.

All removing pixels are represented.



2.3 Problem Solution Approach

- All pixels are loaded “Pixel 2D array”

- for 3D 8bit unsigned integer valued vectors. Pixel class is created , it has 3 attribute which are red , green and blue.

- When we need priority queue. Underlying PriorityQueue used implementation of your priority queue.

- When inserting a pixel in a priority queue. The pixel is inserted in end of array. Then , The pixel and its parent are checked , if the pixel the biggest than parent Swaping between each other.

- Parent : current = currentIndex / 2
  Left Child : 2*currentIndex
  Right Child : 2*currentIndex +1

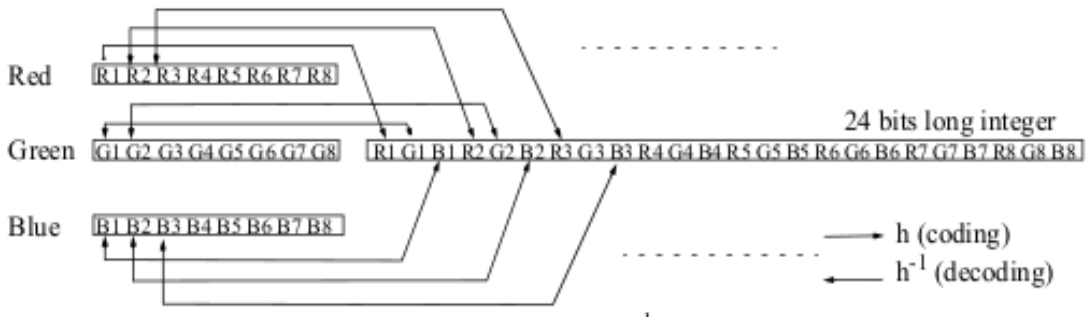
LEX COMPARATOR >> O(1):

```
public int compare(Pixel s1, Pixel s2) {
    if (!(s1.getRed()>s2.getRed()))
        return 0;
    if (!(s1.getGreen()>s2.getGreen()))
        return 0;
    if (!(s1.getBlue()>s2.getBlue()))
        return 0;

    return 1;

}
```

Compare function takes 2 Pixel parameter. Firstly, red value comparing if one red is bigger other colors are checked

EUC COMPARATOR >> O(1) : Each Pixels are getting length, then if first paramter bigger return 1 or 0

BMX COMPARATOR >> O(8) : Bits of colors of Pixel are mixed. Like below screenshot.



(NOTE : N is Pixel numbers “WxH of picture”)

EACH THREADS WORK N TIMES. Time Complexity : ---- O(n) ---
Space complexity : It depends on Operating System
But worst scene O(n)

SYNCHRONIZE : If any thread insert or remove a Pixel to priority queue. This situation must be in “synchronized” area. Then giving such a paramater queue to it.

WAIT : If Queue has no such a Pixel elemen. Removing thread will be waited.
If Queue is full inserting will be waited

NOTIFY : If any thread will be wait mode. It must give signal other thread for continue inserting or removing.

- If All threads worked n (Total pixel number) times, they are finished one by one.

3 RESULT

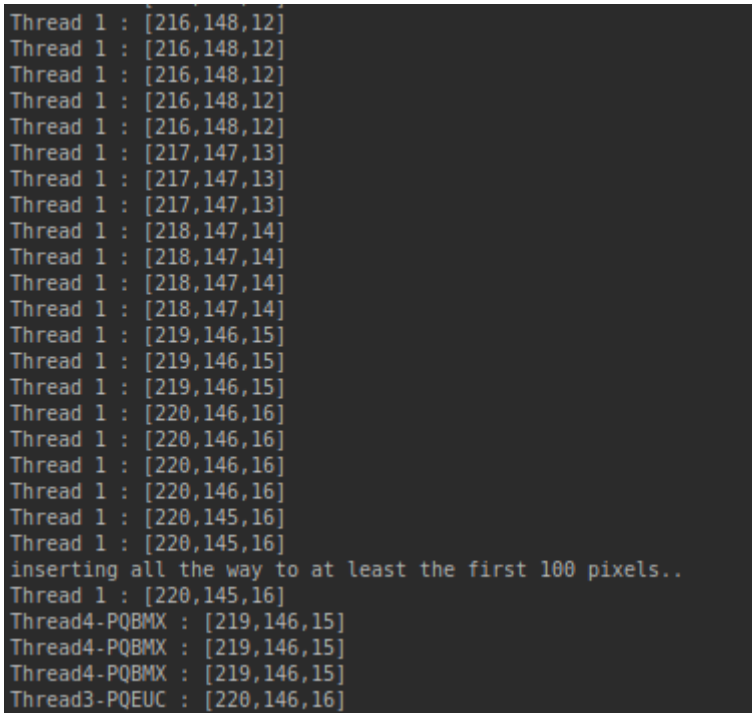
3.1 Test Cases

All thread are started. It must path name of picture on command line argument. If wrong path is entered . Then path will be taken in user input.

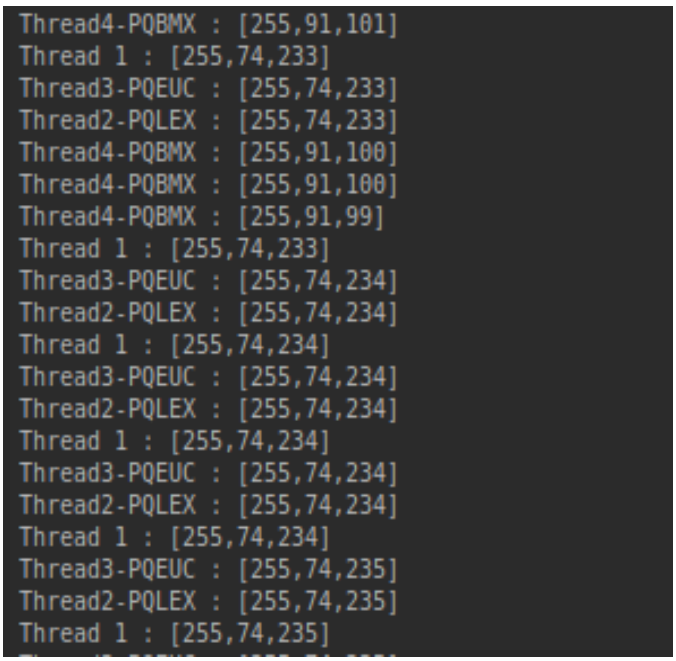
```
BufferedImage bi;
try {
     bi = ImageIO.read(new File(args[0]));
}catch (Exception e){
    try {

        Scanner myObj = new Scanner(System.in);  // Create a Scanner
        System.out.println("\nNo such a command line argument \n " +
                           "Enter picture path(Example: example.png) : "
                           );

        String pathnmame = myObj.nextLine();  // Read user input
        bi = ImageIO.read(new File(pathnmame));
    }catch (Exception e1){
        System.out.println("No such a picture");
        return;
    }
}
Pixel [][] PixelArray = new Pixel[bi.getWidth()][bi.getHeight()];

for (int x = 0; x < bi.getWidth(); x++) {
    for (int y = 0; y < bi.getHeight(); y++) {
        int pixel = bi.getRGB(x, y);

        PixelArray[x][y] = new Pixel();
        PixelArray[x][y].setRed( (pixel >> 16) & 0xff);
        PixelArray[x][y].setGreen((pixel >> 8) & 0xff);
        PixelArray[x][y].setBlue((pixel) & 0xff);
    }
}

    threadClass T1 = new threadClass( name: "Thread 1");

    T1.pixelArray = PixelArray;

    T1.start();

    threadClass T2 = new threadClass( name: "Thread2-PQLEX");
    T2.start();

    threadClass T3 = new threadClass( name: "Thread3-PQEUC");
    T3.start();

    threadClass T4 = new threadClass( name: "Thread4-PQBMX");
    T4.start();
```

3.2 Running Results

Thread-2 , Thread- 3 , Thread-4 will be waited until inserting 100 Pixels.



you can basicly see according to comparator , max pixel will been removing.



Which Thread will run , this situation is decided by Operating System.

```
Thread3-PQEUC : [104,148,255]
Thread2-PQLEX : [104,148,255]
Thread4-PQBMX : [104,148,255]
Thread 1 : [104,148,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [104,148,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread3-PQEUC : [105,147,255]
Thread2-PQLEX : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread 1 : [105,147,255]
Thread4-PQBMX : [105,147,255]
Thread Thread3-PQEUC is exiting.
Thread Thread 1 is exiting.
Thread Thread4-PQBMX is exiting.
Thread Thread2-PQLEX is exiting.

Process finished with exit code 0
```

All threads are finished then exiting in thread one by one.