Gebze Technical University
Computer Engineering

CSE 222 – 2019 Spring

HOMEWORK 5 REPORT

CANER KARAKAŞ
131044061

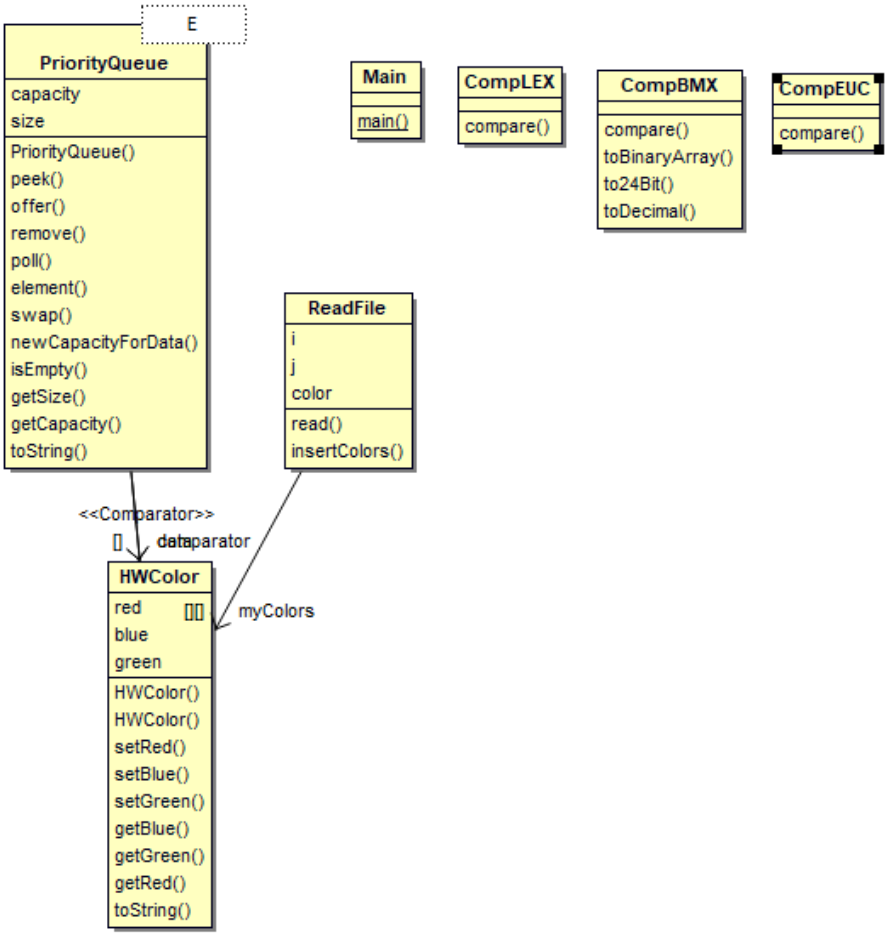
INTRODUCTION

1.1 Problem Definition

In this homework, all pixels of aphoto are saved in a sequence.while reading this file, the pixels to be read are kept in a data with 3 different sequences. This data will be subject to some processing at the same time. Help is obtained from the threads here. Desired sorting criteria LEX, EUC and BMX.

2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

We make the desired data structure with the binary heap and priorty queue. We want to construct max priority queues for color pixels.

-PRİORİTY QUEUE CLASS

This class stores the array of colors, the capacity and size of the array. In addition, it contains a comporator object. The comparator object determines which we must first create the data structure we are trying to create.

When adding elements to our data structure, we used binary heap. Accordingly, our method of offer puts the in relation to the parent uses the relationship (Parents = p , left child => 2*p+1, right child => left child + 1). The remove method moves with the same logic. According to the sorting type, the first one will continue to rank first and then the next. when there is not enough capacity in adding, new place is reserved.

The comparator used for the desired priority when performing addition and subtraction operations, three different classes were created since the comparator used for the desired priority can be in three different types. These are CompLEX and CompEUC and CompBMX. When I create the object of this class, I set the order with these helper classes.

This class stores the array of colors. For this, We created our own colors class. It is HWColor class.

-READFILE CLASS

This class stores the array of HWColor, and Color object. What we do here is to read pixels from a given file and store them in a two-dimensional array, read operation with thread1. At the same time, Priority queue objects create in three different sort order. After reading the 100th pixel, the other threads are created and they are started. The second thread starts to erase the data structure that is sorted by the size of red, green and blue starting from the size of the reds. The third thread starts deleting the data structure sorted by its distance from the origin. The last thread starts deleting the data structure sorted by BMX. They give information to screen output about each process.

3 RESULT

Running Results

```
Thread4-PQBMX: [255, 117, 56] size => 125
Thread3-PQEUC: [206, 155, 2] size => 127
Thread4-PQBMX: [254, 121, 50] size => 126
Thread3-PQEUC: [206, 155, 2] size => 128
Thread3-PQEUC: [206, 155, 2] size => 129
Thread3-PQEUC: [206, 155, 2] size => 130
Thread2-PQLEX: [228, 140, 24] size => 118
Thread4-PQBMX: [255, 121, 50] size => 127
Thread3-PQEUC: [206, 155, 2] size => 131
Thread2-PQLEX: [205, 156, 1] size => 119
Thread2-PQLEX: [255, 78, 201] size => 120
Thread4-PQBMX: [255, 121, 50] size => 128
Thread4-PQBMX: [246, 127, 42] size => 128
Thread3-PQEUC: [206, 155, 2] size => 132
Thread4-PQBMX: [255, 120, 53] size => 130
Thread3-PQEUC: [207, 155, 3] size => 133
Thread4-PQBMX: [255, 75, 221] size => 131
Thread4-PQBMX: [255, 121, 51] size => 131
Thread3-PQEUC: [208, 154, 4] size => 134
Thread3-PQEUC: [208, 154, 4] size => 135
Thread2-PQLEX: [255, 114, 60] size => 121
Thread3-PQEUC: [206, 155, 2] size => 136
Thread2-PQLEX: [255, 115, 60] size => 122
Thread2-PQLEX: [255, 115, 60] size => 123
Thread3-PQEUC: [208, 153, 4] size => 137
Thread3-PQEUC: [222, 144, 18] size => 138
Thread4-PQBMX: [255, 121, 51] size => 133
Thread4-PQBMX: [255, 114, 60] size => 134
Thread3-PQEUC: [255, 114, 60] size => 139
Thread4-PQBMX: [255, 115, 60] size => 135
Thread3-PQEUC: [255, 115, 60] size => 140
Thread4-PQBMX: [255, 115, 60] size => 136
Thread 1 : [255, 114, 60]
Thread3-PQEUC: [255, 115, 60] size => 140
Thread4-PQBMX: [255, 121, 51] size => 134
Thread 1 : [255, 114, 61]
Thread 1 : [255, 114, 61]
Thread 1 : [255, 113, 61]
Thread 1 : [255, 113, 61]
Thread 1 : [255, 113, 62]
Thread 1 : [255, 113, 62]
Thread 1 : [255, 113, 63]
Thread 1 : [255, 113, 63]
Thread 1 : [255, 112, 64]
Thread 1 : [255, 112, 64]
Thread2-PQLEX: [255, 114, 61] size => 111
Thread 1 : [255, 112, 65]
Thread 1 : [255, 111, 65]
Thread2-PQLEX: [255, 114, 61] size => 113
Thread 1 : [255, 111, 65]
```

TIME COMPLEXITY

Class	Method	Complexty
PriorityQueue	Peek()	O(1)
PriorityQueue	Offer()	O(logn)
PriorityQueue	Remove()	O(logn)
PriorityQueue	Poll()	O(logn)
PriorityQueue	Element()	O(1)
PriorityQueue	Swap()	O(1)
ReadFile	Read()	O(n*logn)