Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 3 REPORT

Yusuf Can Kan
16104007

Course Assistant: Özgü Göksu

1 INTRODUCTION

1.1 Problem Definition

In first part the problem is, seperating white and black areas and finding number of white areas inside of an image which representing with integer binary matrix. Matrix includes 1 and 0. 1 represens white areas and 0 represents black areas. When we are counting white areas, if area has another white area throught its left,top,bottom or right we have to count both areas are one. All neighbor areas are counts one.

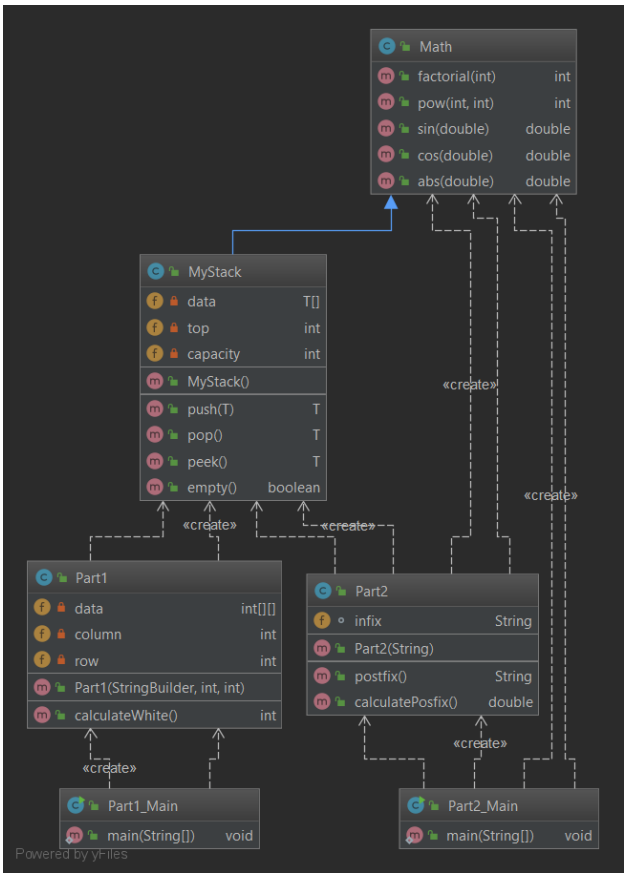
In second part the problem is writing a calculator which takes infix expression as an input and converts it into a postfix and evaluates the value. In expression besides the binary operators there may be mathematical functions such as sin,cos and abs.

1.2 System Requirements

My solution does not requires spesific piece of hardware. It requires just a little memory. You can use it with the so simple computer like calculator. It requires Java Virtual Machine (JVM). It works every machine which has JVM. It does not requires spesific operating system. You can use it on Linux. My solution just so simple, you can use it almost eveywhere.

2 METHOD

2.1 Class Diagrams



2.1.1 Use Case Diagrams

In part1 this software can be use as dividing two type of feature in a simple image with looking for its bit patterns. For example in this homework we looked image files which contains bit batterns such as 0 and 1, and we seperate each of them and count it. The expectation from user is giving a right input. For example if user gives different type of image (besides 0 and 1) program won't work. The only thing which user do is giving the path of image.

In part 2, this software can be use as simple calculator or infix to prefix converter. For simple calculator user must give the infix expression. Besides binary operation, user can apply some mathematical functions such as sin,cos and abs. In addition while software calculating the infix expression, it converts infix to postfix so user can reach to postfix expression to. The expectation from user is giving a right input.If user gives an input which doesn't contain

parantesis, the program will collapse. The only thing which user do is giving the path of proper file.

2.2 Problem Solution Approach

> Explain your method and justify your decisions. How did you achieve the homework goal? Why did you make the design decisions that you've made? Which data structure/algorithm your are using and why? What is your program's time and space complexity? If you didn't solve the problem where did you get stuck? Why?

In first part if program finds one white element inside the matrix, it can reach the neighbor white elements. So if i hold the white matrix with that i can reach and hold neighbor matrix and with that i can reach other neighbor and it continues like that. So for that the most proper data structure is stack. First i read the text file and take all file into an integer array. After that with a loop i find every white element and when i find that element, i find every neighbor of that element and make that black element so the loop doesn't reach again the same white area. For that i created one class that takes StringBuilder parameter, row size and column size. So the only thing user must do is taking file inside a StringBuilder. An in the same class it has calculation method. I choose that design because it is easy to use. In that design and algorithm i can use the stack best efficient way. My whole program time complexity is consist of that O(n) for taking file,creating and filling the array(which n is character size) and O(n) for calculation which n is the character size and reached first white bits neighbor size. So the whole time complexity is O(n). For part one the space complexity is also O(n) because besides the variables it uses n unit of space for line, n unit of space for reaching every element of the matrix and n unit for neghboor of first element of white areas. So the space complexity is O(n).

In second part for calculating the expression first we can change the variables with its values. So first i read the variable lines and stored it into a string and after that when i read the expression line i switch the variables. After that for calculation i converted the infix expression to postfix. For that i use stack. After this conversion i calculate the expressions value and also i used stack. I choose the stack because for that kind of operations that is the best kind of data structure. When converting expression to postfix i looked the parantesis and i stored it different stack. When i find the closing parantesis i put the variables which i holded another stack into the end of statement. I create a class which

takes the infix expression and it has conversiton to prefix method and calculation method. I choose this design because it is easy to reach,use and change methods. The whole program time complexity is O(n) for reading data from file and preparing infix expression,O(n) for converting to postfix which n is the expression character long, and O(n) for the calculation. So the time complexity is O(n) and the space complexity is the same. In addition to all i implement the nessesary mathematical method inside of an another class.(sin,cos,abs,pow,factorial)

I created 2 main class for prevent the confusion.

3 RESULT

3.1 Test Cases

Part 1

0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
Result : 9

0 0 0 1 0 0 1

1 0 1 0 1 0 1
1 0 0 0 1 0 0
Result : 5

1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Result:1

0 0 0 0 0 0
Result:0

0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 1 0 0 0 0
0 1 1 1 1 0 0 1 0 1 0
0 1 1 0 1 0 0 0 0 1 1
0 0 0 1 1 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0
Result:4

Part 2

Input:
w = 5
x = 6

(w + 4) * (cos(x) - 77.9)

Output:
Infix Expression: (5 + 4) * (cos(6) - 77.9)
Postfix Expression: 5 4 + 6 cos 77.9 - *
Result: -692.5549652782204

Input

y = 3

z = 16

(y + sin(y * z)) + (z * abs(-10.3))

Output:

Infix Expression: (3 + sin(3 * 16)) + (16 * abs(-10.3))

Postfix Expression: 3 3 16 * sin + 16 -10.3 abs * +

Result: 168.54283980242

Input:

x = 10

y = 5

z = 15

t = 20

(cos(z + t) + t) * 4 + (abs(x * y) + 3.2) + (x)

Output:

Infix Expression: (cos(15 + 20) + 20) * 4 + (abs(10 * 5) + 3.2) + 10

Postfix Expression: 15 20 + cos 20 + 4 * 10 5 * abs 3.2 + + 10 +

Result: 146.4609672096799

3.2 Running Results

Part 1

Part 2

```
Infix Expression: ( 5 + 4 ) * ( cos( 6 ) - 77.9 )
Postfix Expression: 5 4 + 6 cos 77.9 - *
Result: -692.5549652782204
```

```
Infix Expression: ( 3 + sin( 3 * 16 ) ) + ( 16 * abs( -10.3 ) )
Postfix Expression: 3 3 16 * sin + 16 -10.3 abs * +
Result: 168.54283980242
```

```
Infix Expression: ( cos( 15 + 20 ) + 20 ) * 4 + ( abs( 10 * 5 ) + 3.2 ) + ( 10 )
Postfix Expression: 15 20 + cos 20 + 4 * 10 5 * abs 3.2 + + 10 +
Result: 146.4609672096799

Process finished with exit code 0
```