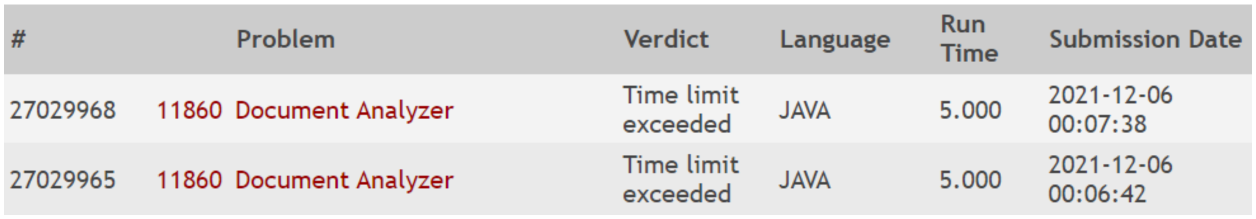
<u>Part 3: Report</u>

<u>Part 1:</u>

The code for my solution of document analyzer (11860) produced the correct results that were expected in the “testDocAnalyser1” and “testDocAnalyserBig”.txt files, as their outputs matched up correctly. My program is able to break down a document into only words, and then search for the smallest index that contains all words. There are currently no known bugs that would cause this program to not work. Because my solution is O(n^2), it did not pass the online judge test, demonstrating that my program is less efficient than what is ideal. Its run time was deemed to be >5 seconds. This is the screenshot of my results.



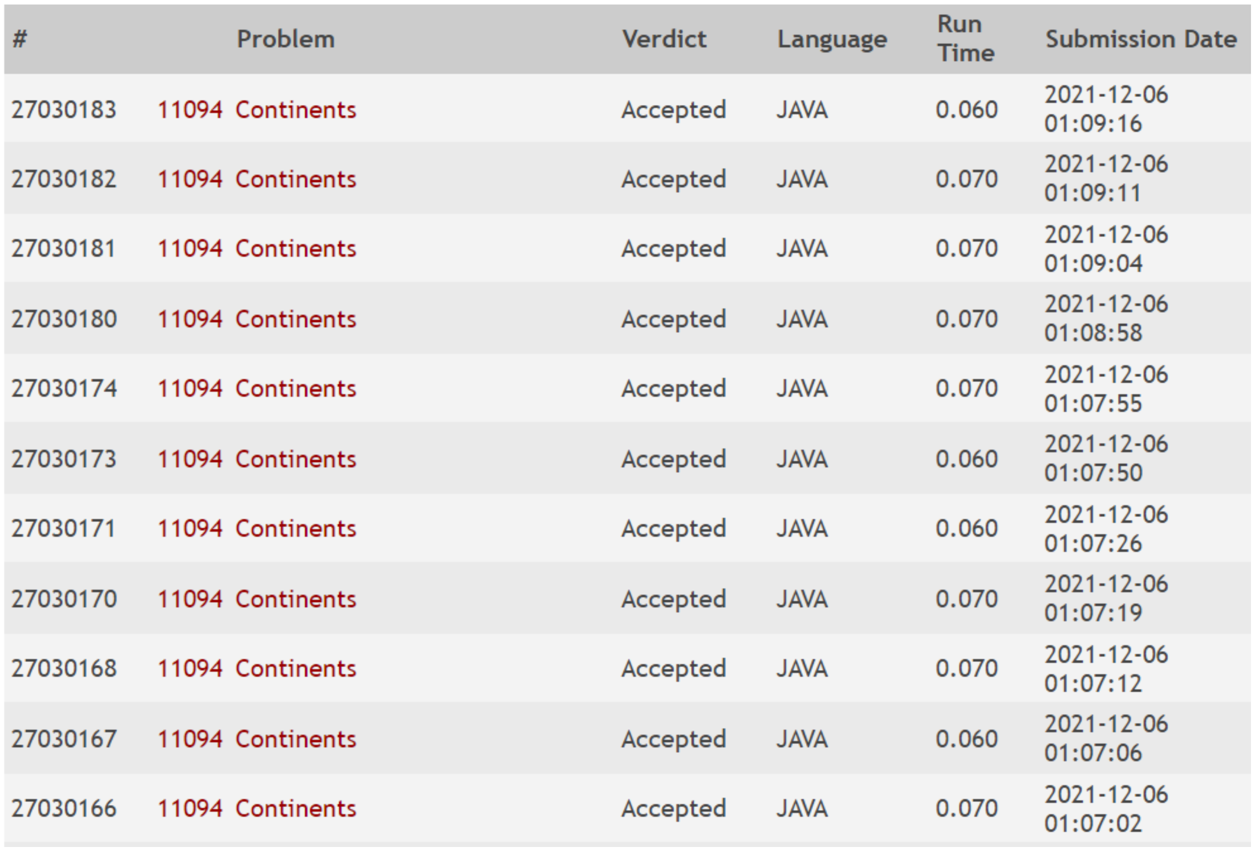
#	Problem	Verdict	Language	Run Time	Submission Date
27029968	11860 Document Analyzer	Time limit exceeded	JAVA	5.000	2021-12-06 00:07:38
27029965	11860 Document Analyzer	Time limit exceeded	JAVA	5.000	2021-12-06 00:06:42

Table 1: Results of online judge for Document Analyser

As we can see, these 2 attempts are not accepted by the online judge as they go over 5 seconds. The exact running time of the tests are unknown.

<u>Part 2A:</u>

The code for my solution of Continents (11094) produced the correct results that were expected in the “testContinents1” and “testContinentsBig”.txt files, as their outputs were also able to match up correctly. My program is able to traverse any given map and calculate the sizes of all continents Majid is not located in, and then prints out the size of the largest continent. There are currently no known bugs that would cause this program to not work. My solution was able to pass the online judge test with an efficient time. I tested my run time multiple times by resubmitting my code into the online judge. Below is my results from running my tests 11 times.



#	Problem	Verdict	Language	Run Time	Submission Date
27030183	11094 Continents	Accepted	JAVA	0.060	2021-12-06 01:09:16
27030182	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:09:11
27030181	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:09:04
27030180	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:08:58
27030174	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:07:55
27030173	11094 Continents	Accepted	JAVA	0.060	2021-12-06 01:07:50
27030171	11094 Continents	Accepted	JAVA	0.060	2021-12-06 01:07:26
27030170	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:07:19
27030168	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:07:12
27030167	11094 Continents	Accepted	JAVA	0.060	2021-12-06 01:07:06
27030166	11094 Continents	Accepted	JAVA	0.070	2021-12-06 01:07:02

Table 2: Results of online judge for Continents

As we can see, these 11 attempts were accepted by the online judge as they all ran in under 3 seconds. The average running time out of the 11 attempts was approximately 0.0663 seconds.

Part 2B:

The code for Part 2B produced partially correct results for each .txt file. It was able to produce the correct output for the test case “testContinents1” but was not able to fully produce all of the correct outputs for “testContinentsBig”.txt files, as it was only able to produce about half of the correct results. For instance, for the first 13 test cases in “testContinentsBig”.txt, 11 out of 13 of my outputs were correct.

My Output	testContinentsBigOutputPartB.txt

29 15
82 26
1 0
18 7
5 4
5 4
0 -1
11 5
33 11
4 3
1 0
37 12
0 -1
	29 15
	82 27
	1 0
	18 7
	5 4
	5 4
	0 -1
	11 5
	33 12
	4 3
	1 0
	37 12
	0 -1

Table 2: Comparison of my outputs and the correct outputs (Note: Values in red indicate incorrect result)

Currently the code works as intended in certain cases, however it has bugs that make it occasionally unable to get the correct result if the largest region has edges in common. This is the only known bug that prevents it from being fully functional.

<u>Program Development</u>

For Part 1, some programing improvements I was able to make include using no helper methods. I was originally using a method to process a given document, however I decided to make my program only run in the main method, which should theoretically reduce my overall run time. I also used a buffered reader to avoid excessive flushing and also reduce running time.

For Part 2 A, I used 2D array instance variables to store the values of the map in a coordinate format, and I also had a boolean array to store if a coordinate has been visited or not. I made both instance variables private to prevent any other classes from affecting them. Just like Part 1, I originally had this program all stored in a method that would be called in the main, however I noticed after removing the method and just moving all of its code into the main I was getting faster results from the online judge (0.100s to 0.060s). For Part 2 B I simply reused my code from part A and added more private instance variables that keep track of the coordinates of the biggest continent (with the biggest distance to the northwest point).