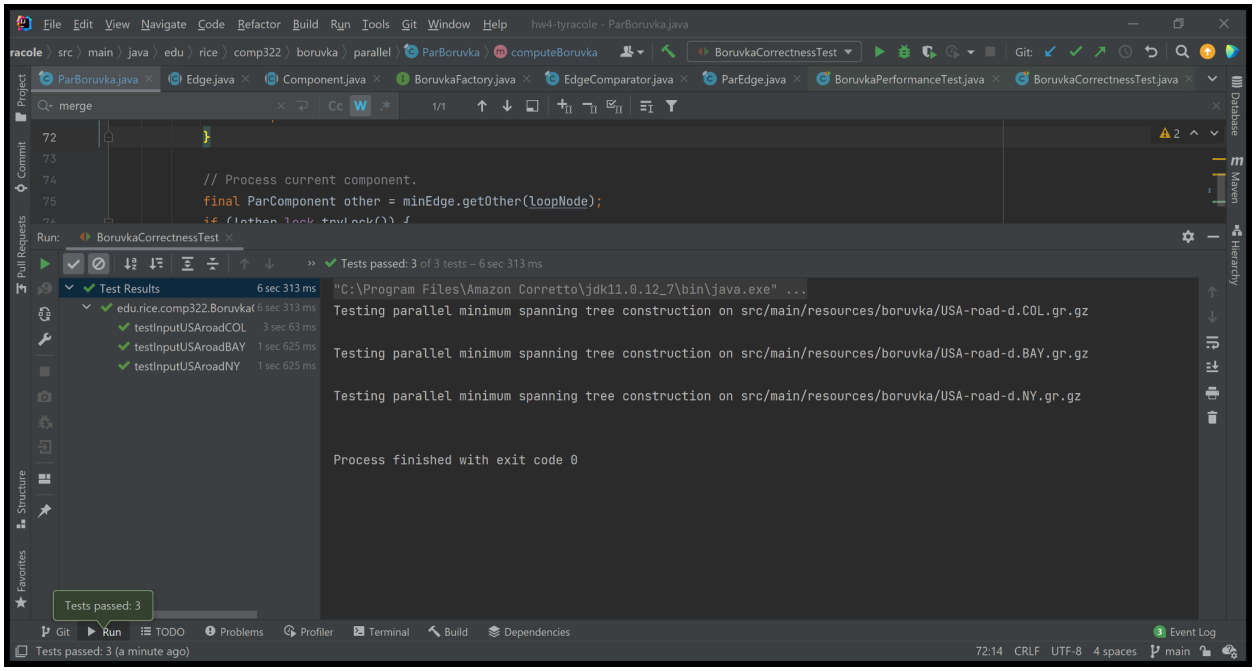
(a) My solution starts with me editing the ParComponent class so that it has as one of a parComponents global variable a lock that I am able to use in parBoruvka to ensure that any threads do not end up reading or writing the data of the same parallel component simultaneously. In computeBoruvka, as long as nodesLoaded is not empty, my algorithm polls a component and locks it until it is processed. Once it is finally processed, I unlock the component's lock. I then check if the graph is processed. If so, then I unlock the current component and break out of the while loop. If not, then I unlock the other component of the edge and add it to nodesLoaded. I merge the other component with the a component from the minimum edge, unlock both components, and add them to nodesLoaded. Once I exit the while loop, if the loopNode is not null, I compute the total edges and total weight of the graph.

(b) I believe that my solution avoids data races because my use of locks helps prevent two threads from either reading or writing the same data at the same time. Only live components are processed by my algorithm, and once the processing is done, they are unlocked and added

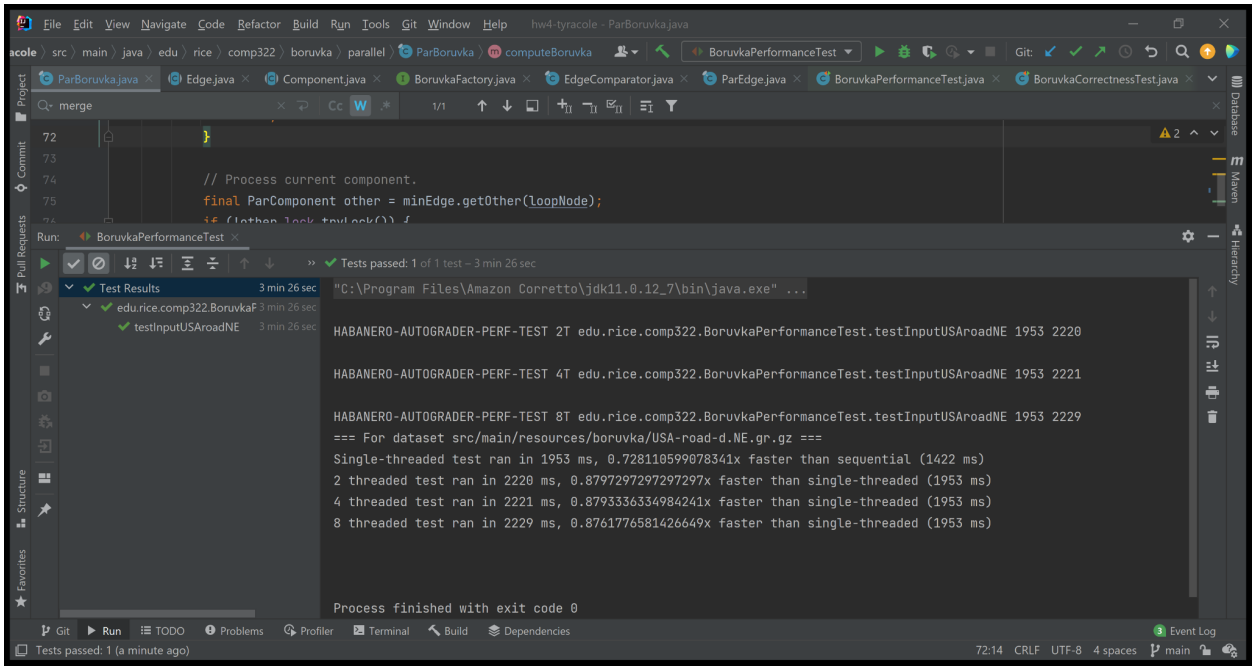
(c) My solution avoids deadlock by setting a component to either dead (or alive), and it is only unlocked if made dead, because it would have already been processed. So, it is not likely that two or more threads will reach a deadlock with my solution.

(d) My solution avoids livelock by setting a component to either dead (or alive), and it is only unlocked if made dead, because it would have already been processed. So, it is not expected that two or more threads will encounter a livelock

Correctness



Performance


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
hw4-tyracole - ParBoruvka.java
BoruvkaPerformanceTest
ParBoruvka.java
Edge.java
Component.java
BoruvkaFactory.java
EdgeComparator.java
ParEdge.java
BoruvkaPerformanceTest.java
BoruvkaCorrectnessTest.java
merge
72 }
74 // Process current component.
75 final ParComponent other = minEdge.getOther(loopNode);
Run: BoruvkaPerformanceTest
Tests passed: 1 of 1 test – 3 min 26 sec
Test Results 3 min 26 sec
edu.rice.comp322.BoruvkaF 3 min 26 sec
testInputUSAroadNE 3 min 26 sec
"C:\Program Files\Amazon Corretto\jdk11.0.12_7\bin\java.exe" ...
HABANERO-AUTOGRADER-PERF-TEST 2T edu.rice.comp322.BoruvkaPerformanceTest.testInputUSAroadNE 1953 2220
HABANERO-AUTOGRADER-PERF-TEST 4T edu.rice.comp322.BoruvkaPerformanceTest.testInputUSAroadNE 1953 2221
HABANERO-AUTOGRADER-PERF-TEST 8T edu.rice.comp322.BoruvkaPerformanceTest.testInputUSAroadNE 1953 2229
=== For dataset src/main/resources/boruvka/USA-road-d.NE.gr.gz ===
Single-threaded test ran in 1953 ms, 0.728110599078341x faster than sequential (1422 ms)
2 threaded test ran in 2220 ms, 0.8797297297297297x faster than single-threaded (1953 ms)
4 threaded test ran in 2221 ms, 0.8793336334984241x faster than single-threaded (1953 ms)
8 threaded test ran in 2229 ms, 0.8761776581426649x faster than single-threaded (1953 ms)
Process finished with exit code 0
Git Run TODO Problems Profiler Terminal Build Dependencies Event Log
Tests passed: 1 (a minute ago)
72:14 CRLF UTF-8 4 spaces main