Michael Lederer

Glodon Take-Home Exam Writeup

Going into this assignment, my mind began to race while studying the problem and thinking through an algorithm. It was very tricky given the minimal problem statement, but I think there’s something interesting about ambiguous problems. On one hand, it gives you a lot of freedom to interpret the problem and formulate your solution, however in a vacuum like a job interview, I definitely felt a little lost as to how my solution was supposed to fit together in the grand scheme and the exact output expected.

In my experience, (mostly through my CS education) when I’ve been faced with programming problems, there is usually clearly defined input and output. In this problem, there was a clearly defined input, but not output, and there was some ambiguity in exactly how to solve the problem. Each question I asked brought about different angles to the problem I hadn’t considered prior. Once I had a complete understanding of what was expected, though, I had a ton of fun building this project from start to finish, and I believe this was a really great exercise in utilizing a lot of the modern C++ concepts and showcasing my programming style.

**Suggestions:**

* I think it would be beneficial to give a couple of examples of input and the desired output to give a better understanding of the problem statement, OR expand on the problem statement further
* I realized through analyzing the provided cpp/h files that it would likely only be feasible to solve this problem in a timely manner if I made edits to the provided source and header files. Perhaps including a sentence in the README specifying that we may edit the provided files would save a significant amount of time for future candidates

**Important notes:**

In order to produce a deliverable in a reasonable time frame, there were a couple of aspects of the problem I did not address in my solution

* I do not have any handling for improper JSON input
* I do not have any handling for lines with undefined slopes, though that may not be too difficult to implement with further research
* I’m not entirely sure how to continuously stream in JSON input, so I simply split my code into source and header files that can be included in a file that will be able to pipe in input

**Function Runtime Analysis**

* collinearMapInsertion – O(n) where n is the number of total line segments
* mergeHelper – O(m\*log(m)) for sorting where m is the number of collinear line segments + O(m) for the merge operations. Combined runtime = O(m\*log(m))
* mergeLines – O(n\*log(n)) where n is the total number of line segments + O(n) for std::vector::insert. Combined runtime = O(n\*log(n))