Glodon Software

Hello,

Thank you for the opportunity to complete the take home coding challenge, I greatly appreciate it. Below is a write up on my thoughts about the challenge, the process I took to create my solution, and general comments I have.

The goal or delivery of this coding challenge was to create a solution that could analyze an array of 2 dimensional line segments and determine if any of the line segments could be merged. The instructions state that some of the line segments can be merged, they are parallel and share an endpoint. To be quite honest, upon reading this I was quite confused on what exactly it meant by stating that certain line segments could be merged. I understood the concept of merging line segments but line segments being parallel and sharing endpoints seemed contradictory to me. The solution that has been designed may be incorrect due to a misunderstanding of the problem but still I tried to formulate an answer and design something that is complete.

My idea was that I should look at the slopes of all the line segments and see if there are any similarities that I could piece together. The reason for this is because if two line segments have the same slope they are either parallel or the same line; thus in either case satisfying one of the conditions for a mergeable line segment. An analysis of the slopes of the line segments showed me that there were some that were quite similar but not exact. I determined that for most of the line segments with similar slopes they were only different by less than 0.02 and usually within the range of 0.01. I reevaluated the problem statement and upon not being able to really think of some other way to look at the problem, I decided that my solution would be designed around this. My idea was to use the slopes to find line segments with very close slopes, and then merge them into one line. After having a list of line segments with similar gradients, I could determine a line that encompasses both line segments. Because none of the slopes were exactly the same, this meant the lines would most likely intersect or have some overlap, this to me meaning one of the line's endpoint would be shared on the other line. My solution was created on this premise.

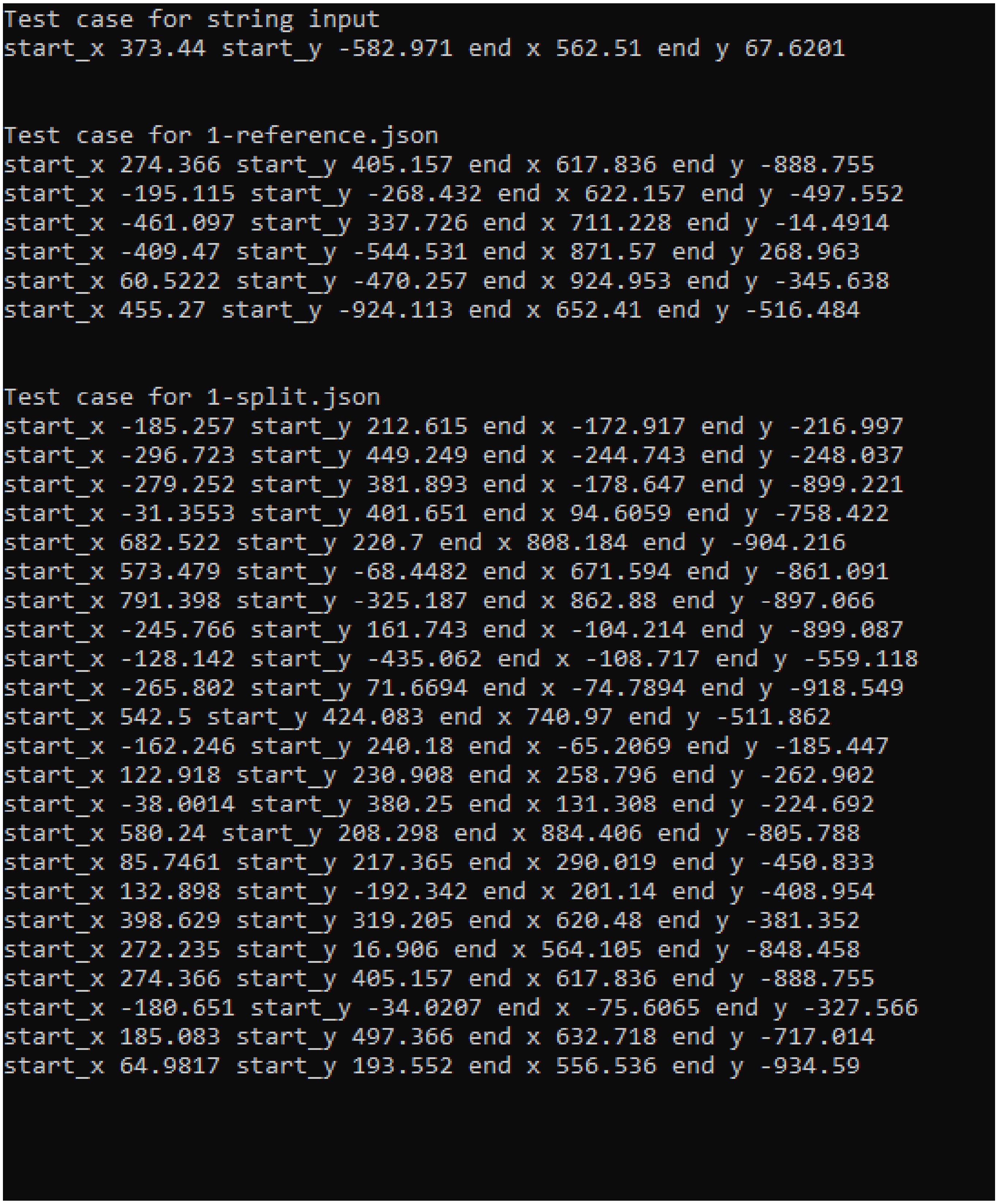
When it came to testing the code, I made a separate function for each of the test files that were included. I did not know exactly what I could test my test cases against, just simply that they ran without errors. I did my own analysis of the "1-reference.json" file and made sure the solution that was presented matched my theoretical understanding of how the problem should be solved. I did not explicitly check "1-split.json". I would have liked to generate my own random array of points and create test cases for those. Furthermore I think it would be quite interesting to test different sizes of arrays to see the run time and how fast the algorithm I developed was. My code is definitely not the most efficient code out there and maybe there would be faster ways to store and analyze the coordinates. Another point I think can be improved on is I could have made certain parts of the function into separate functions. Each function should be responsible for something specific and the deserialize function could have been broken down more.

Overall I enjoyed this coding challenge very much. Working with geometry and coordinates is quite fun and interesting. I think that If I understood the problem more I could have devised a solution that may be more correct but I think understanding the problem statement and using my own judgement is also part of the challenge. I have also attached an appendix with my solution below. Please let me know if there any questions or concerns about my work.

Thanking You,

Asish Mehta

Appendix



```
Test case for string input
start_x 373.44 start_y -582.971 end x 562.51 end y 67.6201


Test case for 1-reference.json
start_x 274.366 start_y 405.157 end x 617.836 end y -888.755
start_x -195.115 start_y -268.432 end x 622.157 end y -497.552
start_x -461.097 start_y 337.726 end x 711.228 end y -14.4914
start_x -409.47 start_y -544.531 end x 871.57 end y 268.963
start_x 60.5222 start_y -470.257 end x 924.953 end y -345.638
start_x 455.27 start_y -924.113 end x 652.41 end y -516.484


Test case for 1-split.json
start_x -185.257 start_y 212.615 end x -172.917 end y -216.997
start_x -296.723 start_y 449.249 end x -244.743 end y -248.037
start_x -279.252 start_y 381.893 end x -178.647 end y -899.221
start_x -31.3553 start_y 401.651 end x 94.6059 end y -758.422
start_x 682.522 start_y 220.7 end x 808.184 end y -904.216
start_x 573.479 start_y -68.4482 end x 671.594 end y -861.091
start_x 791.398 start_y -325.187 end x 862.88 end y -897.066
start_x -245.766 start_y 161.743 end x -104.214 end y -899.087
start_x -128.142 start_y -435.062 end x -108.717 end y -559.118
start_x -265.802 start_y 71.6694 end x -74.7894 end y -918.549
start_x 542.5 start_y 424.083 end x 740.97 end y -511.862
start_x -162.246 start_y 240.18 end x -65.2069 end y -185.447
start_x 122.918 start_y 230.908 end x 258.796 end y -262.902
start_x -38.0014 start_y 380.25 end x 131.308 end y -224.692
start_x 580.24 start_y 208.298 end x 884.406 end y -805.788
start_x 85.7461 start_y 217.365 end x 290.019 end y -450.833
start_x 132.898 start_y -192.342 end x 201.14 end y -408.954
start_x 398.629 start_y 319.205 end x 620.48 end y -381.352
start_x 272.235 start_y 16.906 end x 564.105 end y -848.458
start_x 274.366 start_y 405.157 end x 617.836 end y -888.755
start_x -180.651 start_y -34.0207 end x -75.6065 end y -327.566
start_x 185.083 start_y 497.366 end x 632.718 end y -717.014
start_x 64.9817 start_y 193.552 end x 556.536 end y -934.59
```