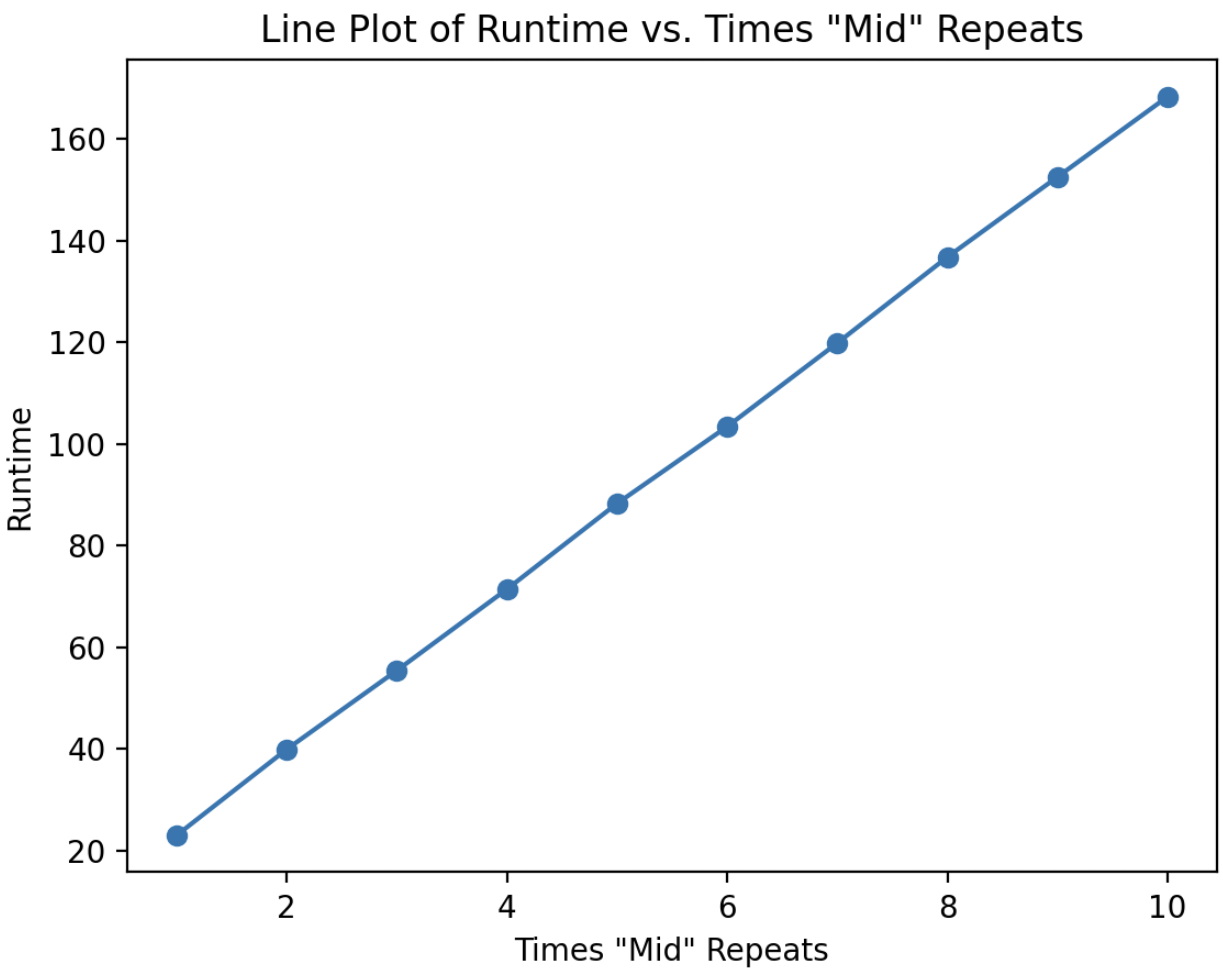
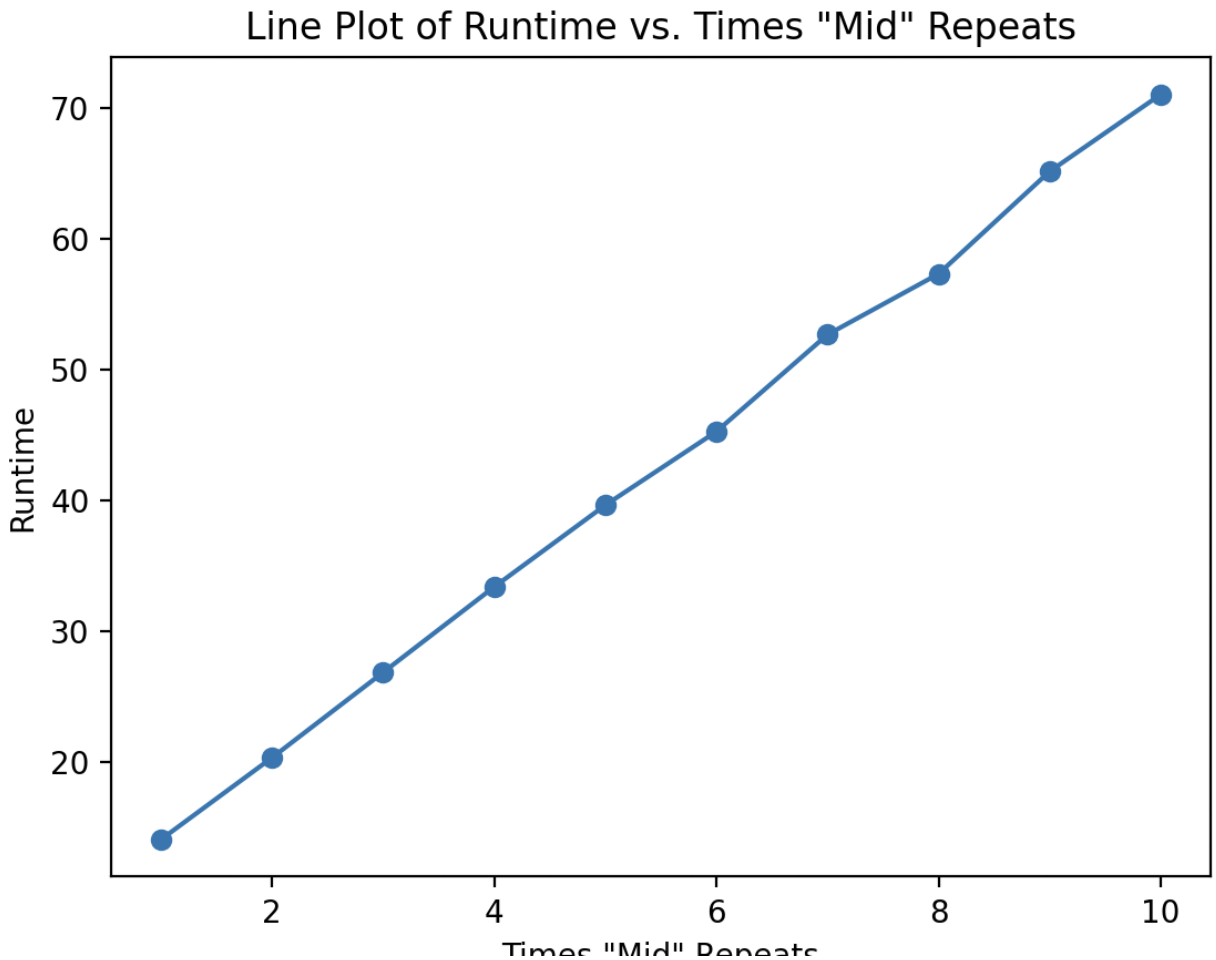
a.


Line Plot of Runtime vs. Times "Mid" Repeats
Runtime
Times "Mid" Repeats
20
40
60
80
100
120
140
160
2
4
6
8
10

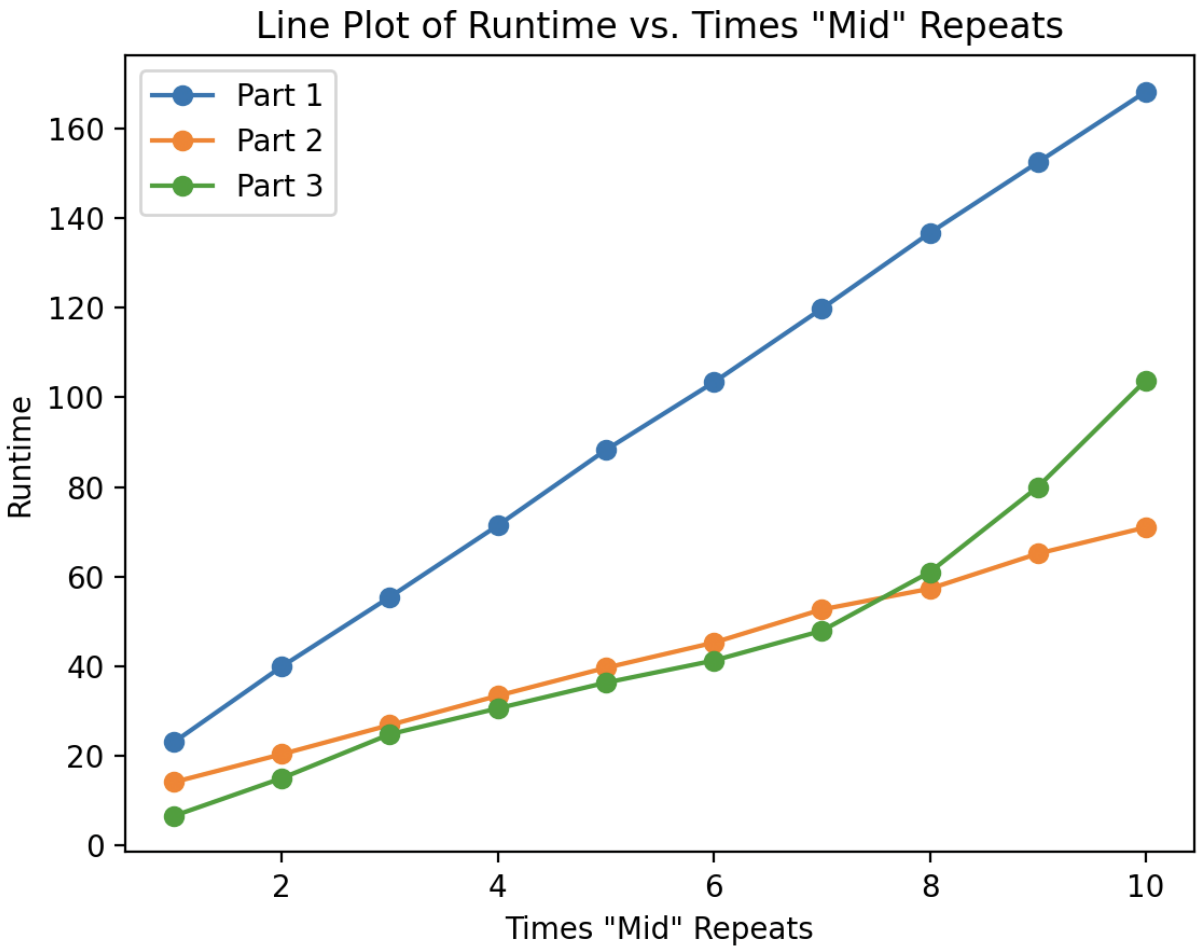
b.



Comment on the speedup that you observe thanks to the multi-threading that was introduced. Is it more or less than what you expected?

Speed up is not exactly what I expected as the performance doesn't increase proportional to the number of work threads. But the runtime increase is still linear, which is reasonable.

c.



Compare now the runtime trend of the system in Part 2 vs. Part 3. What is the additional speedup? Where is the additional improvement coming from if we are performing the same exact operations over the same images?

The additional speedup:
[-7.549, -5.36, -2.06, -2.82, -3.32, -4.0, -4.76, 3.7, 14.88, 32.8]
which is around 1/10 on average (excluding the last two runs where part 3 behaves unnormal)

This is because the server is processing requests according to FIFO policy. Requests in part 2 come at the order that operations on same image are coming together, whereas in part 3, the target image is changed every time.

This causes the server unable to simultaneously process several requests in part 2, as the order has to be kept and mutual exclusion should hold.
In part 3, there can be more parallelism going on, so faster.

d.
FR-FCFS.
We can think of requests targeting the same image as not ready, and requests targeting different images from the one operation is going on as ready. This way, the server does not have to be restricted by the arrival order of requests, and can jump to ready requests right away to improve performance in Part2’s case.