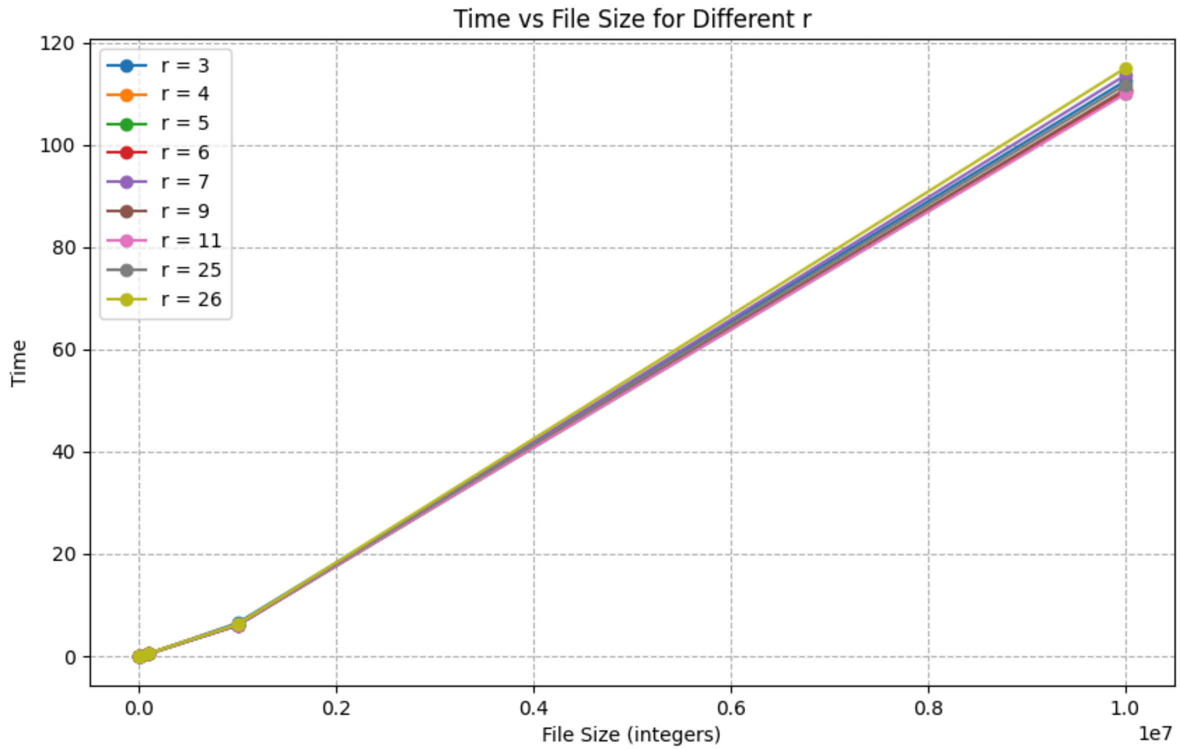
Programming Assignment 1 Report
COSC 581
Zack Malkmus

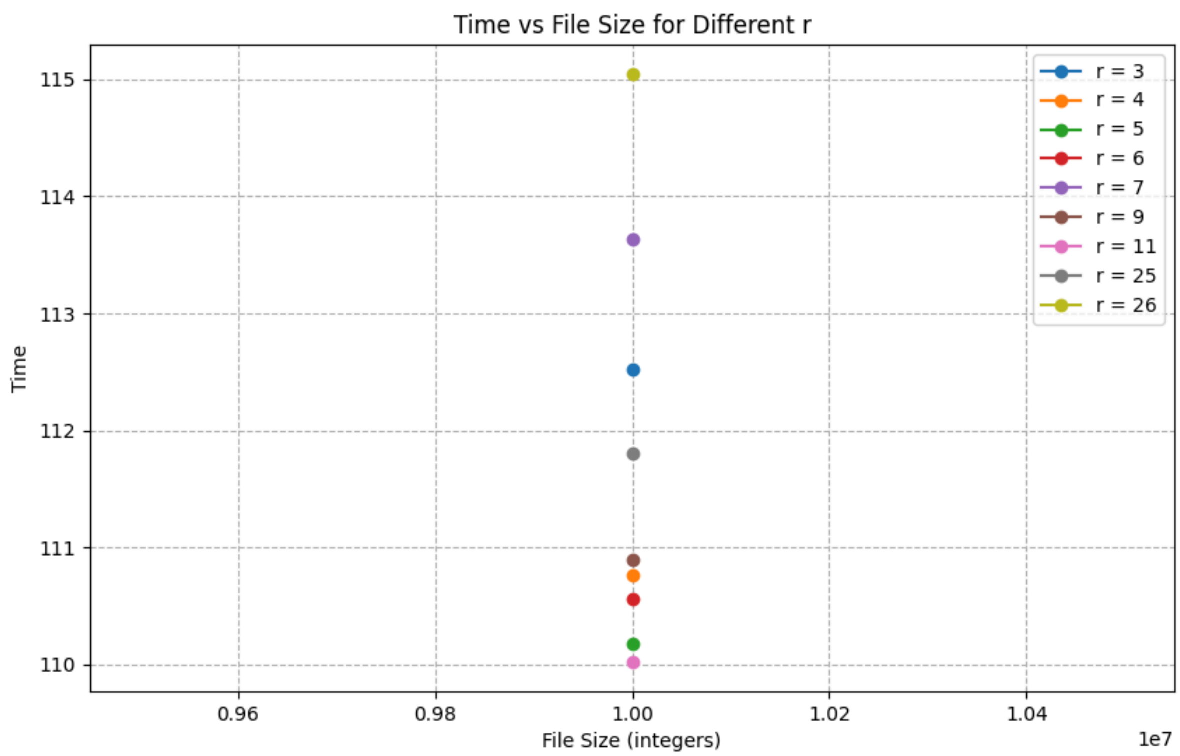
Implementation:
My quicksort is a basic implementation using 3 arrays to construct the sorted list recursively [quicksort(less) + pivot + quicksort(greater)] using my implementation of median of medians pivot selection method. After being divided into subfiles of size r, I select my pivot by looping through the subfile and removing the largest and smallest element, until one is remaining (the median). I call this function the gulag_median().

R Values:
I chose r values of 3, 4, 5, 6, 7, 9, 11, 25, and 26. As discussed in class, the odd values in r > 3 provide median of medians a linear time, while all even values provide a polynomial runtime. I also included values of 25 and 26 to test the impact of a larger subfile size (which should result in a slightly slower runtime with a poor pivot selection method like the gulag_median).

Methodology:
To test my quicksort, I created a program to generate pseudo-random values of a user-specified size in the same format as specified in the writeup. I compared the quicksort against the built in sorted command to ensure correctness. I measured runtime with the python time module for all r values for all file sizes to provide the following data.

Results:
Running on my local machine, I was able to only reasonably perform tests on up to file sizes of 100 million integers. The results mostly line up with the theoretical expectations, with a few outliers probably due to system variance. Most importantly, r=5 appears near the bottom and r=26 appears at the top as expected. In order to see polynomial runtime begin to really take over, the test cases will need to be much larger.


Time vs File Size for Different r
Time
File Size (integers)
1e7
r = 3
r = 4
r = 5
r = 6
r = 7
r = 9
r = 11
r = 25
r = 26
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0


Time vs File Size for Different r
Time
File Size (integers)
1e7
r = 3
r = 4
r = 5
r = 6
r = 7
r = 9
r = 11
r = 25
r = 26
110
111
112
113
114
115
0.96
0.98
1.00
1.02
1.04