Analysis document: ArrayListSorter

● Graded

Student

HARRY KIM

Total Points

43.5 / 50 pts

Question 1

Question 1

1 / 1 pt

✓ **– 0 pts** Correct

Question 2

Question 2

3 / 3 pts

✓ **– 0 pts** Correct

Question 3

Question 3

4 / 4 pts

✓ **– 0 pts** Correct

Question 4

Question 4

3 / 3 pts

✓ **– 0 pts** Correct

Question 5

Question 5

💬 **7.5** / 9 pts

✓ **– 1 pt** X- or Y-axis is unlabeled, missing units, or otherwise incorrect

✓ **– 0.5 pts** Insufficient explanation on how the timing experiment was conducted

💬 X label not specific enough

Question 6

Question 6

4 / 4 pts

✓ **– 0 pts** Correct

Question 7

Question 7

4 / 4 pts

✓ **– 0 pts** Correct

Question 8

Question 8

6.5 / 9 pts

- ✓ **– 1 pt** X- or Y-axis is unlabeled, missing units, or otherwise incorrect
- ✓ **– 1 pt** The plot doesn't "speak for itself"; it is unclear, or hard to read or interpret it some way.
- ✓ **– 0.5 pts** Insufficient explanation on how the timing experiment was conducted

Question 9

Question 9

6.5 / 9 pts

- ✓ **– 1 pt** X- or Y-axis is unlabeled, missing units, or otherwise incorrect
- ✓ **– 1 pt** The plot doesn't "speak for itself"; it is unclear, or hard to read or interpret it some way.
- ✓ **– 0.5 pts** Insufficient explanation on how timing experiments were conducted

Question 10

Question 10

4 / 4 pts

- ✓ **– 0 pts** Correct

Question 11

Late Penalty

0 / 0 pts

- ✓ **– 0 pts** submitted before deadline

Harry Kim
U1226472
9/30/2020
CS 2420

Analysis Document — Assignment 5

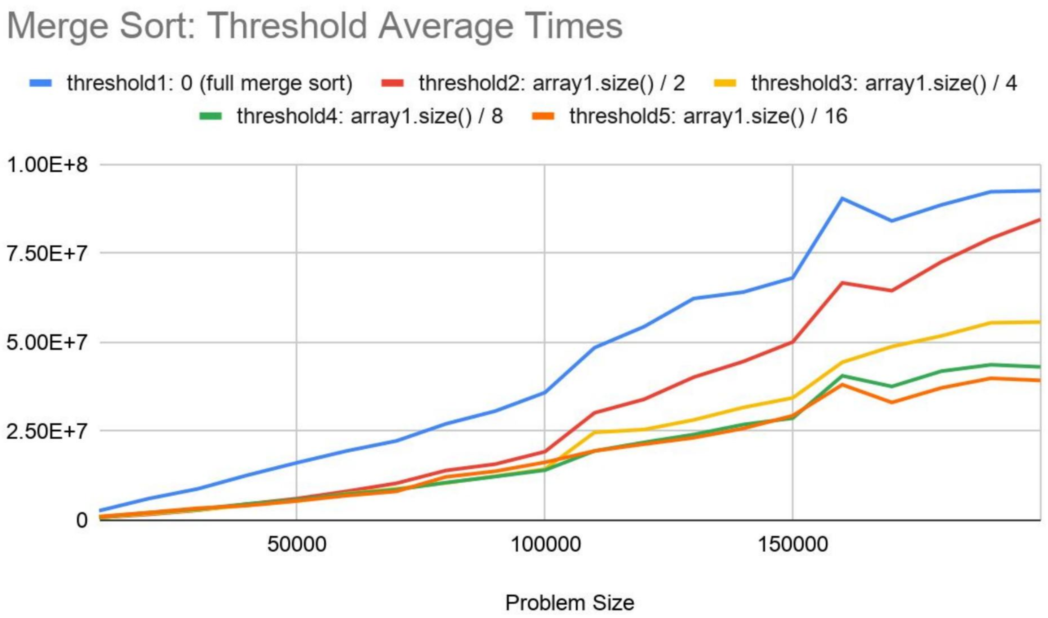
1. I am **not** invoking one of my three exemptions for pair programming due to an extenuating circumstance. My programming partner is Braden Morfin, and I submitted the program to Gradescope.

2. Pair programming experience:
    - My partner and I spent about two hours each day for 4 days to complete the assignment and create tests for the assignment for a total of 8 hours.
    - Braden’s very smart and is a nice person, he’s good at communication, and he codes very well. I would plan to work with Braden again.

3. The expected growth rates of the merge sort in all three cases, that of which being the best, average, and worst cases, is O(NlogN). The expected growth rate of merge sort is important because no matter how the list is ordered, the running time will always be O(NlogN). Because of this property, the ordering of the list to be sorted does not affect the running time of merge sort.

4. We invoked insertion sort when the threshold for small sublists in merge sort was reached via the use of an if statement which stated that checks that there's still subarrays to be sorted **and** if the size of the threshold surpasses the size of the subarray. In this way, we ensured that insertion sort is called only when we want it to and not over the entire array.

Questions assigned to the following page: 5, 6, and 7

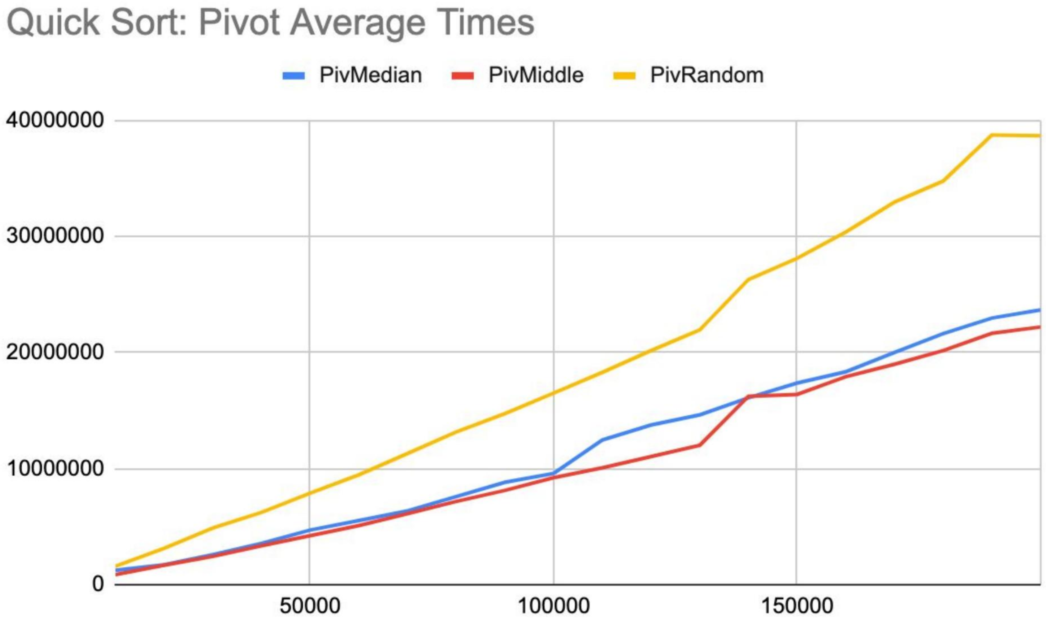


5. The threshold value that simulates a full merge sort is threshold1, as seen in the chart above. According to our data, the best threshold value for which merge sort switches over to insertion sort is somewhere in between the size of the array divided by 8, and the size of the array divided by 16 (size/16 being the better of the two).

6. The expected growth rate of quick sort in the best case is O(NlogN), in the average case it is O(NlogN), and in the worst case, it is O(N^2). The ordering doesn't affect the run time as much while the choice of pivot affects the running time of quick sort significantly as the best case of quick sort happens when the pivot partitions the array into two equally-sized subarrays at each stage while the worst case is when the partition generates an empty subarray at each stage.

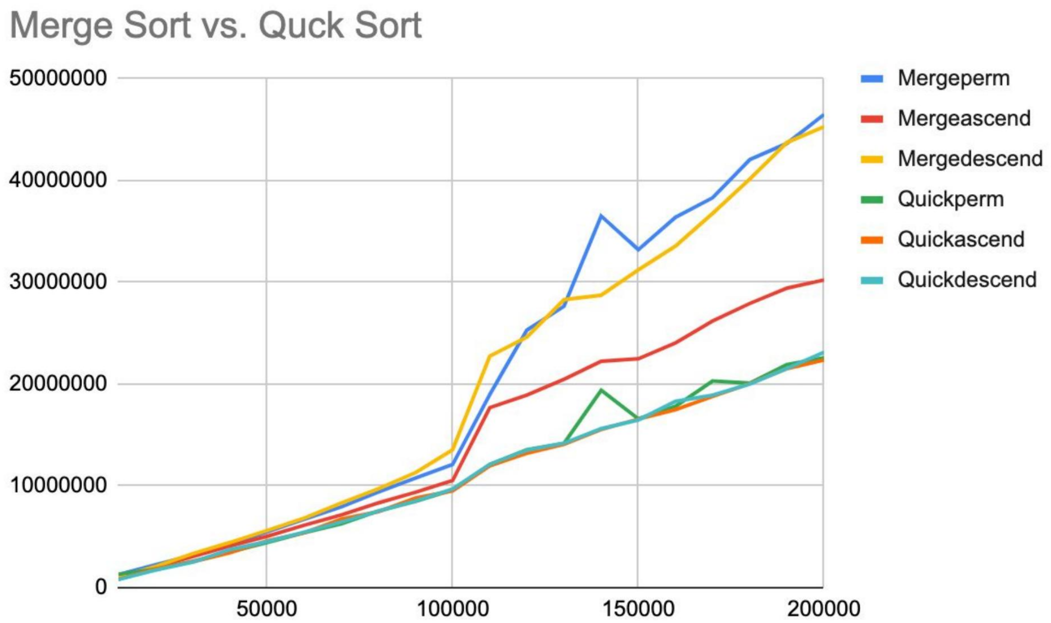
7. The three strategies that we used to choose the pivot in our quick sort implementation was finding the median from the first, middle, and last elements of the array, a random pivot point from the array, and the middle element of the array. The Big-O behavior of each strategy is O(1) as the most complex statements in choosing the pivot point were if statements. Each strategy required no loops or nested loops of any kind. Each pivot-selection strategy should not affect the overall Big-O behavior of quick sort.

Question assigned to the following page: 8

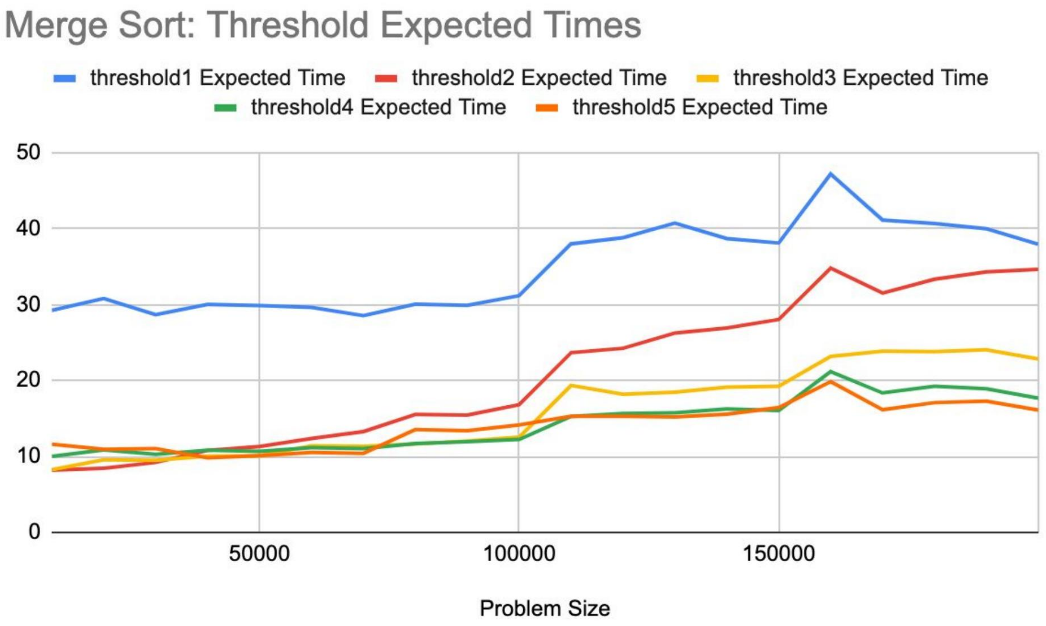


8. According to our data, the best pivot-selection strategy for quick sort seems to be choosing the pivot in the middle of the array regardless of what the value is in the middle index.

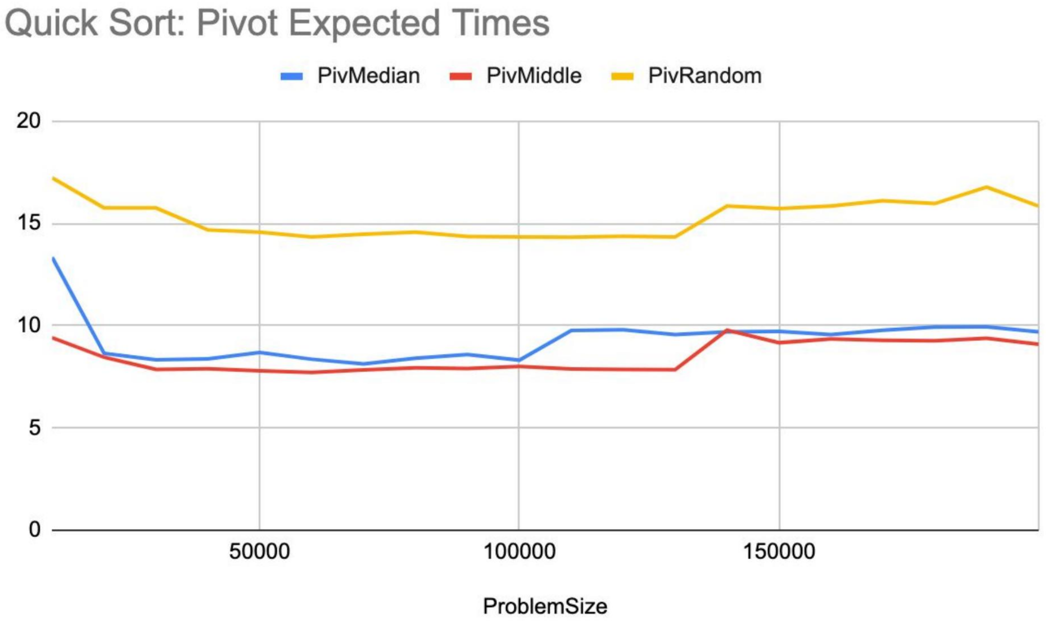
Question assigned to the following page: 9



9. For each of the three categories of lists (ascending, permuted, and descending), our data tells us that quick sort outperformed merge sort in every category even when choosing the best threshold for merge sort.



10.



Using the “Check Analysis” technique within our run times, The graphs of the expected times of merge sort and quick sort show that our actual run times exhibit expected growth rates because the “Check Analysis” technique states that if the graph converges

Question assigned to the following page: 10

to a positive number, it means our sorting methods are working as expected (convergence of T(N)(actual time)/F(N)(expected time)).