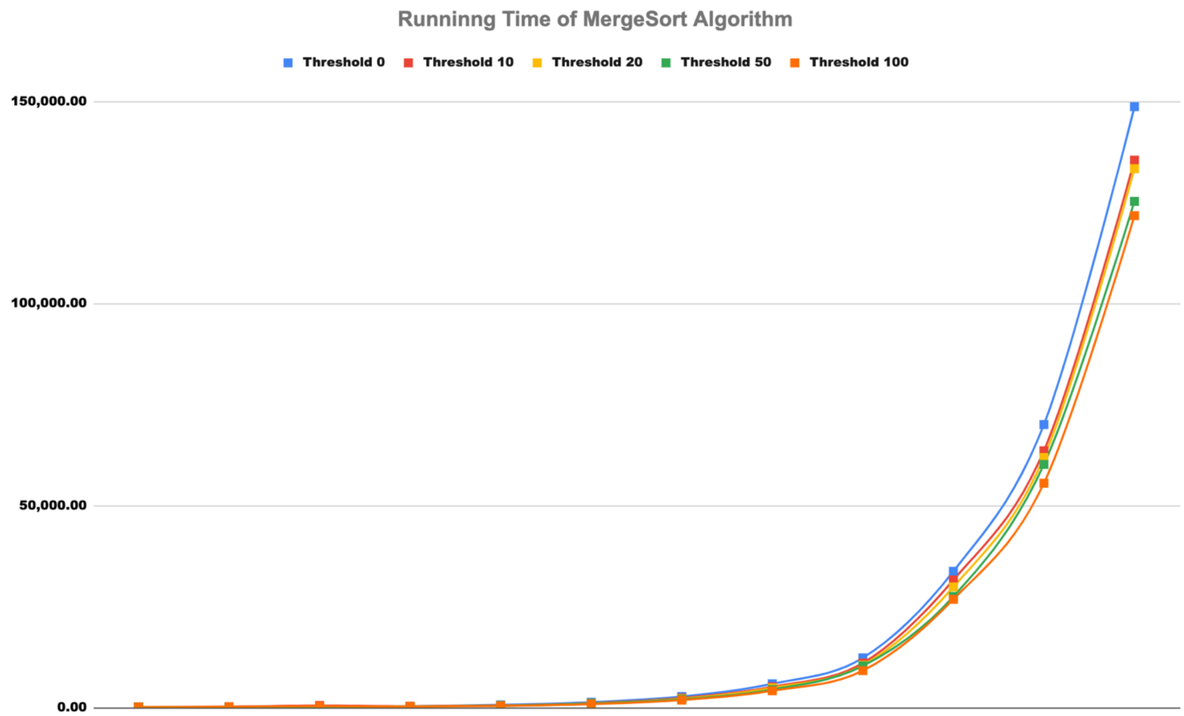
When you are satisfied that your program is correct, write a detailed analysis document. The analysis document is 40% of your assignment grade. Ensure that your analysis document addresses the following.

Note that if you use the same seed to a Java Random object, you will get the same sequence of random numbers (we will cover this more in a lab soon). Use this fact to generate the same permuted list every time, after switching threshold values or pivot selection techniques in the experiments below. (i.e.,re-seed the Random with the same seed)

1. Who are your team members?
   Junming Jin
   Jinyi Zhou
   Jaehyung Park

2. Mergesort Threshold Experiment: Determine the best threshold value for which mergesort switches over to insertion sort. Your list sizes should cover a range of input sizes to make meaningful plots, and should be large enough to capture accurate running times. To ensure a fair comparison, use the same set of permuted-order lists for each threshold value. Keep in mind that you can't resort the same ArrayList over and over, as the second time the order will have changed. Create an initial input and copy it to a temporary ArrayList for each test (but make sure you subtract the copy time from your timing results!). Use the timing techniques we already demonstrated, and be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Note that the best threshold value may be a constant value or a fraction of the list size. Plot the running times of your threshold mergesort for five different threshold values on permuted-order lists (one line for each threshold value). In the five different threshold values, be sure to include the threshold value that simulates a full mergesort, i.e., never switching to insertion sort (and identify that line as such in your plot).

Threshold Number of Elements	0	10	20	50	100
512	141,975.00	163,266.00	115,953.00	96,955.00	92,941.00
1024	86,089.00	221,230.00	57,401.00	88,502.00	174,215.00
2048	126,866.00	499,435.00	102,753.00	92,980.00	329,772.00
4096	286,625.00	329,345.00	241,967.00	196,527.00	190,057.00
8192	663,044.00	510,667.00	459,589.00	451,922.00	398,368.00
16384	1,316,752.00	1,108,588.00	1,019,477.00	1,022,870.00	857,005.00
32768	2,738,919.00	2,368,287.00	2,206,759.00	2,070,279.00	1,861,170.00
65536	5,880,041.00	5,134,925.00	4,868,886.00	4,464,745.00	4,171,578.00
131072	12,308,930.00	11,019,416.00	10,492,795.00	10,310,860.00	9,146,387.00
262144	33,734,284.00	31,589,654.00	29,867,237.00	27,491,825.00	26,803,849.00
524288	70,071,105.00	63,571,975.00	61,910,008.00	60,194,912.00	55,541,682.00
1048576	148,763,295.00	135,520,771.00	133,411,442.00	125,318,478.00	121,777,905.00

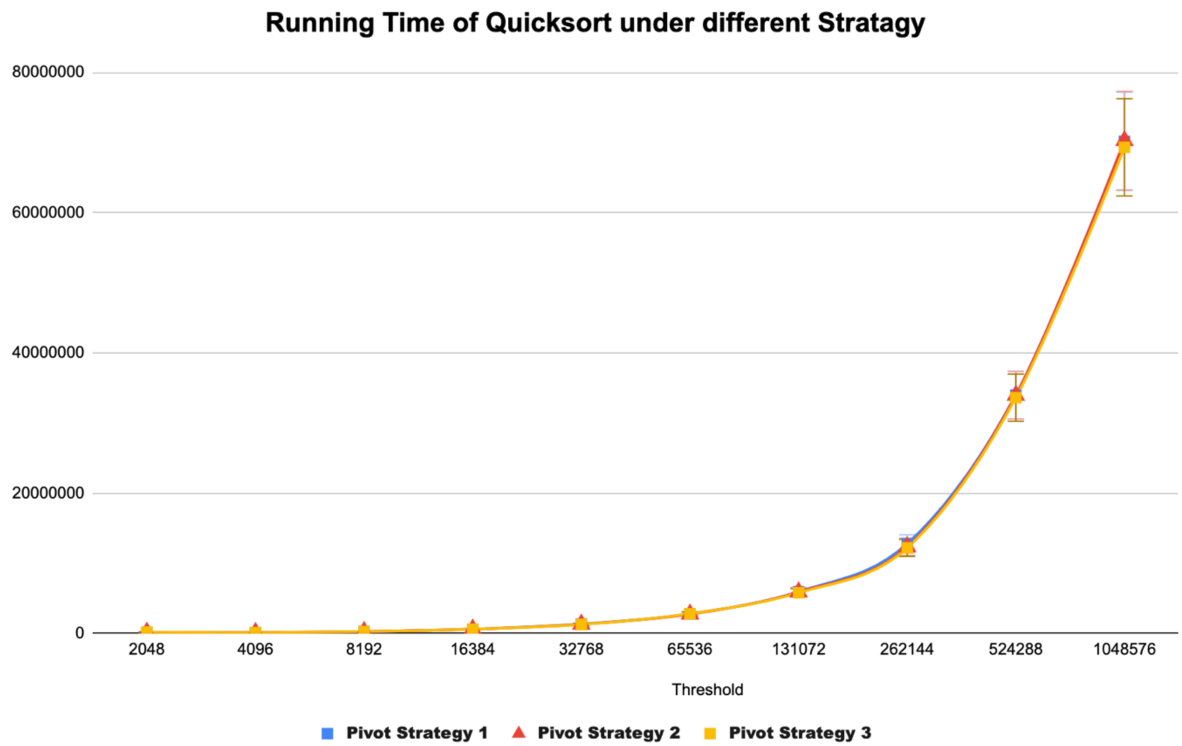


The chart and graph show that the running time decreases when the threshold increases. When the threshold was 0, it took the longest running time with the same number of elements. When the threshold was 100(the largest amount threshold), the running time was the shortest.

3. Quicksort Pivot Experiment: Determine the best pivot-choosing strategy for quicksot. (As in #2, use large list sizes, the same set of permuted-order lists for each strategy, and the timing techniques demonstrated before.) Plot the running times of your quicksort for three different pivot-choosing strategies on permuted-order lists (one line for each strategy).

Number of Elements	Pivot Strategy 1	Pivot Strategy 2	Pivot Strategy 3
1024	162652	178306	205104
2048	133840	135004	136668
4096	282514	283454	280176
8192	619512	621026	610106
16384	1351009	1323011	1291976
32768	2799937	2755242	2820228
65536	5966974	5917607	5816520
131072	12795807	12369793	12215712

262144	33997614	33960524	33630827
524288	70179559	70283281	69332849
1048576	150150655	150440846	149143132



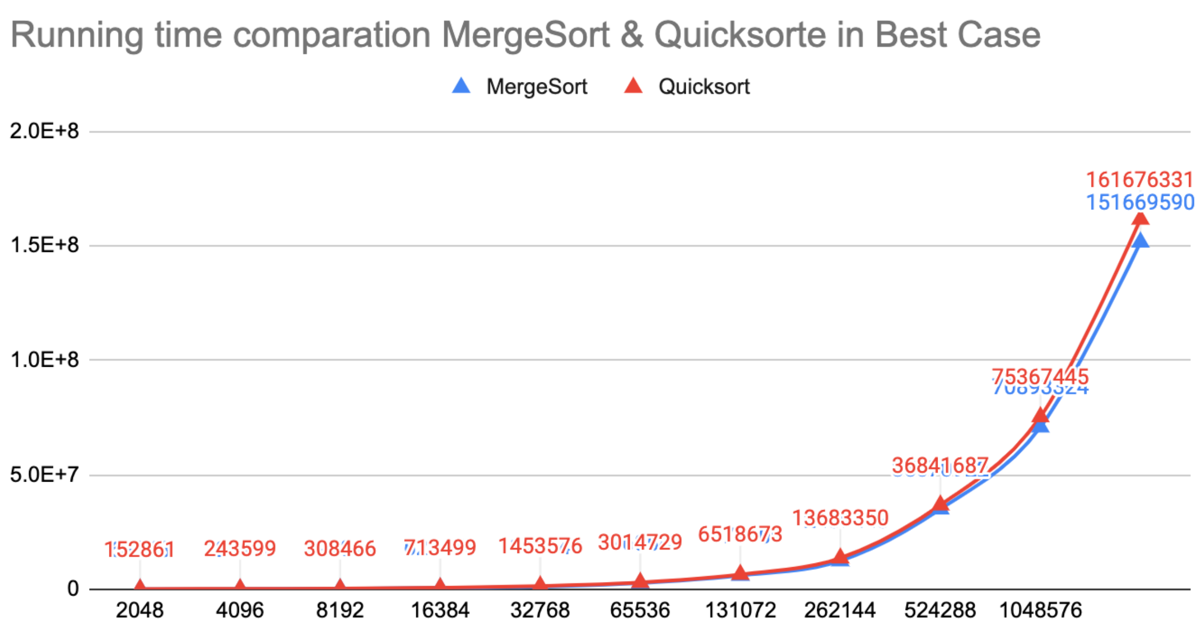
Pivot Strategy 1: the middle element of the arraylist.
Pivot Strategy 2: random select number of elements of the array list.
Pivot Strategy 3: the smallest between the start element, the middle element and the ending element

According to the graphs above, strategy 1 has the worst performance compared to the other 2. Strategies 2 and 3 have similar performance for the method's running time. However, Strategy 3 is slightly better than Strategy 2 compared to the array's first, middle, and last elements and select the smallest amount in the pivot.

4. Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case). For the mergesort, use the threshold value that you determined to be the best. For the quicksort, use the pivot-choosing strategy that you determined to be the best. Note that the best pivot strategy on permuted lists may lead to O(N^2) performance on best/worst case lists. If this is the case, use a different pivot for this part. As in #2, use large list sizes, the same list sizes for each category and sort, and the timing techniques

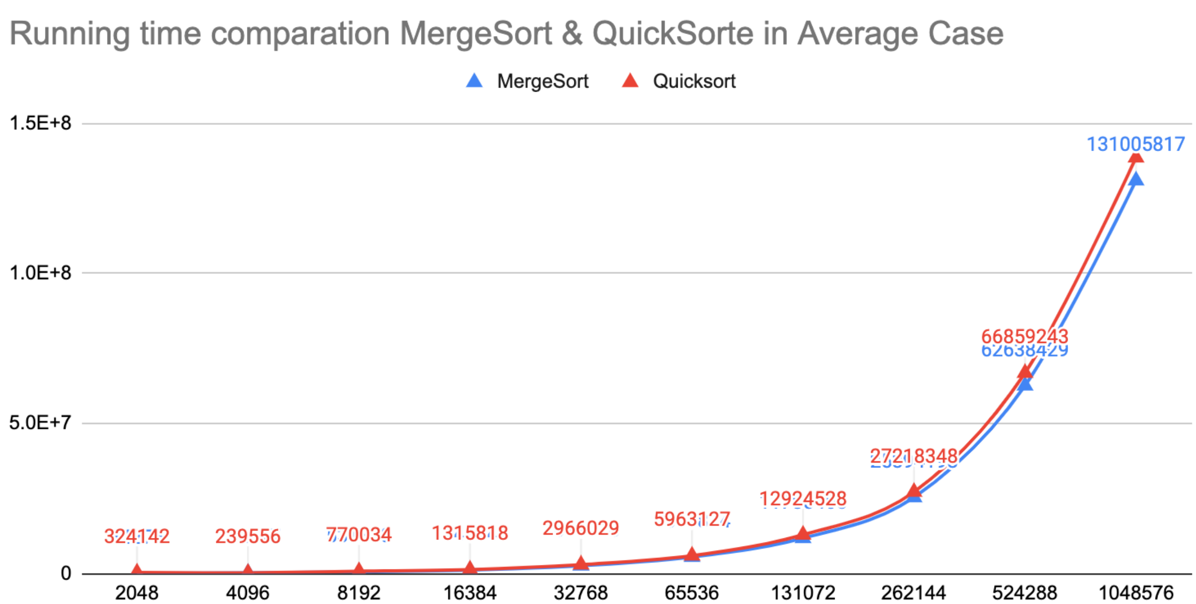
demonstrated before. Plot the running times of your sorts for the three categories of lists. You may plot all six lines at once or create three plots (one for each category of lists).

Bast case Number of Elements	MergeSort	Quicksort
1024	64343	152861
2048	128188	243599
4096	294814	308466
8192	623456	713499
16384	1143040	1453576
32768	2760545	3014729
65536	6079509	6518673
131072	12439443	13683350
262144	35070922	36841687
524288	70893324	75367445
1048576	151669590	161676331



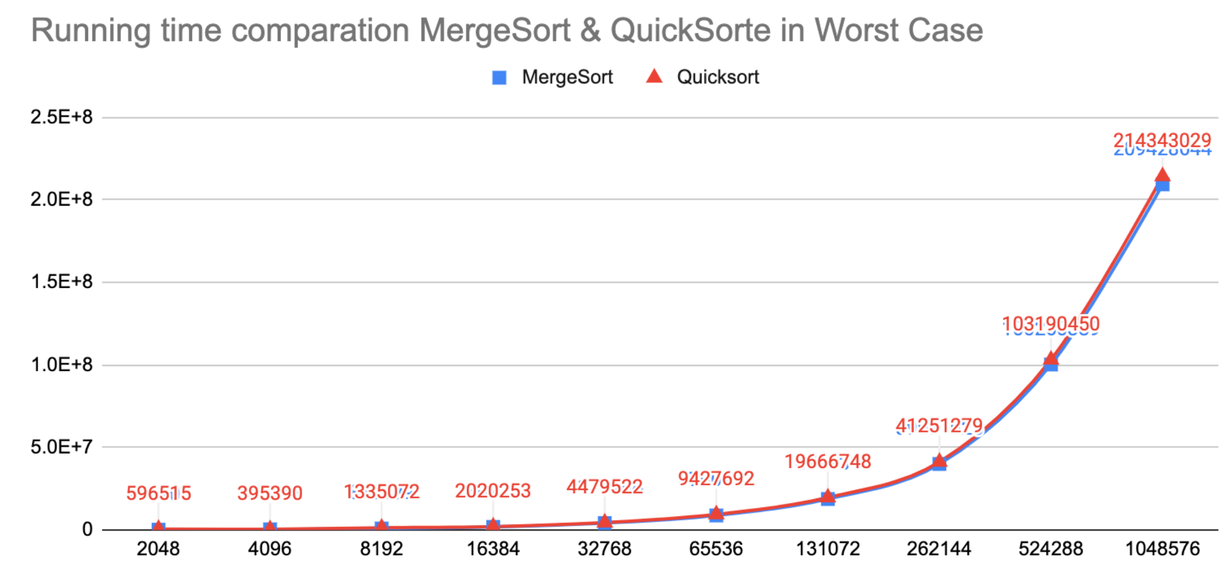
Average case Number of Elements	MergeSort	Quicksort
1024	84276	324142
2048	222017	239556
4096	531383	770034

8192	1143040	1315818
16384	2611016	2966029
32768	5503874	5963127
65536	11785466	12924528
131072	25394198	27218348
262144	62638429	66859243
524288	131005817	138655820
1048576	289887892	300448492



Worst case Number of Elements	MergeSort	Quicksort
1024	156103	596515
2048	378518	395390
4096	870239	1335072
8192	1849232	2020253
16384	4176494	4479522
32768	8790732	9427692
65536	18757310	19666748
131072	39991925	41251279
262144	100260389	103190450

524288	209428644	214343029
1048576	457430497	462112400



According to the graphs above, QuickSort performs better at running time than MergeSort in all the cases . Especially in the best-case and worst-case, the running time is significantly shorter.

5. Do the actual running times of your sorting methods exhibit the growth rates you expected to see? Why or why not? Please be thorough in this explanation.

The actual running times of sorting methods matched my expectation.
In the MergeSort method, O (N*log(N)) is this method's Big-O notation. The shape of the curve matches up with my graph above for all three cases.
In the QuickSort method, O (N*log(N)) is this method's Big-O notation for the best and average case. However, in the worse case, O(N^2) is this case's Big-O notation. Even though the graph is not ideally matched. Since the pivot selection strategy is based on the best case, therefore the graph of time complexity is smaller

Team members are encouraged to collaborate on the answers to these questions and generate graphs together. However, each member must write and submit his/her own solutions.

Upload your solution (.pdf only) through Canvas.