LE1: Bubble Sort vs Merge Sort Time Complexity

STEP 7

A. SMALL DATA SET

Number of Data Elements **100**

Sort Algorithm					**Executi**
	1	**2**	**3**	**4**	**5**
Bubble Sort	0.0000390000	0.0000395000	0.0000411000	0.0000396000	0.0000395000
Merge Sort	0.0000315000	0.0000314000	0.0000316000	0.0000340000	0.0000320000

B. MEDIUM DATA SET

Number of Data Elements **10000**

Sort Algorithm					**Executi**
	1	**2**	**3**	**4**	**5**
Bubble Sort	0.4496475000	0.4600797000	0.4510218000	0.4578332000	0.4447989000
Merge Sort	0.0029667000	0.0028824000	0.0030028000	0.0029983000	0.0029168000

C. LARGE DATA SET

Number of Data Elements **100000**

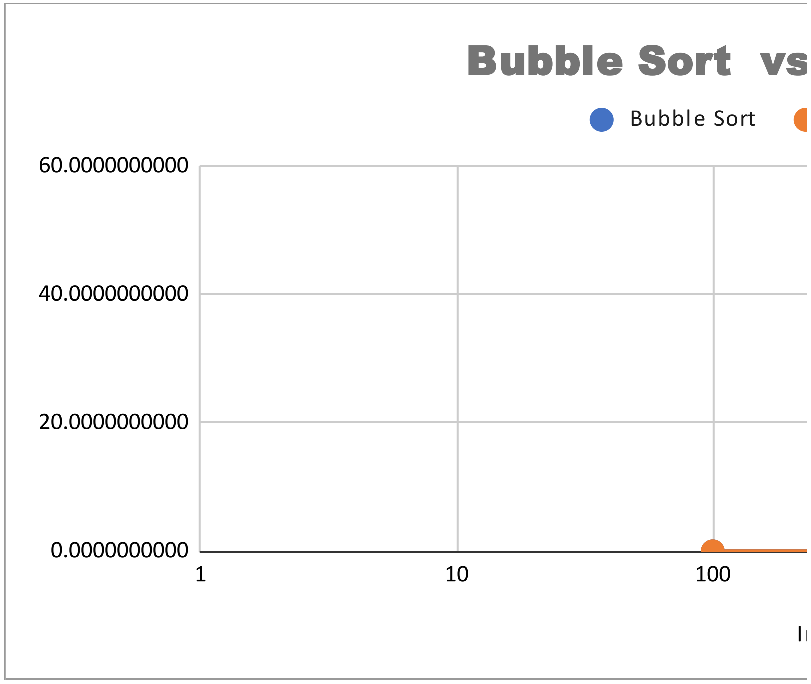
Sort Algorithm	Executi				
	1	2	3	4	5
Bubble Sort	54.0000469000	44.6067122000	48.9109467000	44.5278063000	50.5320228000
Merge Sort	0.0347734000	0.0347836000	0.0340212000	0.0332029000	0.0341671000

STEP 8

Insert the chart here.

Bubble Sort

Input Size	Bubble Sort	Merge Sort
100	0.0000404400	0.0000456500
10000	0.4502552000	0.0030634300
100000	50.1831432300	0.0345779800



STEP 9

Provide your analysis here.

Based on the chart above, the merge sort consistently ran below 1 second regardless what dataset I used. M

I noticed that when I used the medium dataset (10000 input size) and large dataset (100000 input size), the

With that observation, we might assume that Merge Sort will always run the fastest regardless of input size
dataset with input size of 100. Of course, we may still use the Merge Sort Algorithm since it is still fast.

I, therefore, conclude that Bubble Sort is great to use with smaller datasets while Merge Sort is more efficie

STEP 10

Question 1: What is your observation in the running times of bubblesort()?

It was only fast when I used the small data set (100 input size). When I used the medium data set, it got a bi
I had to wait for about a minute or less for the code to finish running.

Question 2: What is your observation in the running times of mergesort()?

For Merge Sort, the runtime was consistently below 1 second.

Question 3: Based on the average running time, which one is the fastest? Explain further.

The fastest was the Bubble Sort using the small dataset because it had an average of 0.0000404400 seconds

Question 4: Based on the average running time, which one is the slowest? Explain further.

The slowest was the Bubble Sort using the large dataset because it had an average of 50.1831432300

Question 5: What can you conclude about the time complexity of Bubble Sort and Merge Sort based on you

I conclude that Bubble Sort Algorithm is best to use for small dataset input while Merge Sort Algorithm is be

Question 6: Which algorithm is more efficient for larger input sizes?

Merge Sort Algorithm

Question 7: How do your empirical findings compare to the theoretical time complexities of these algorithm

The time complexity of the Bubble Sort Algorithm is O(n^2) while the Merge Sort Algorithm is O(n log n). Th
and its runtime because O(n log n) is more efficient that O(n^2) and during our observation, Merge Sort Alg

on Time per Run (in secs)

6	7	8	9	10	Average
0.0000475000	0.0000393000	0.0000396000	0.0000402000	0.0000391000	0.0000404400
0.0000339000	0.0001651000	0.0000334000	0.0000319000	0.0000317000	0.0000456500

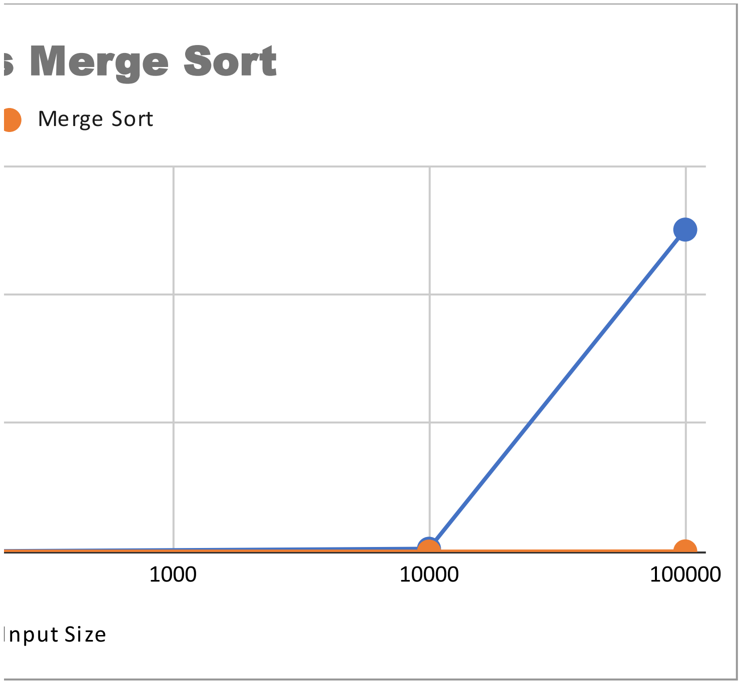
on Time per Run (in secs)

6	7	8	9	10	Average
0.4561440000	0.4375074000	0.4541714000	0.4494432000	0.4419049000	0.4502552000
0.0030731000	0.0031991000	0.0036576000	0.0031074000	0.0028301000	0.0030634300

on Time per Run (in secs)

6	7	8	9	10	Average
46.1414857000	49.9194851000	49.5336677000	56.4267579000	57.2325010000	50.1831432300
0.0340038000	0.0359561000	0.0362812000	0.0339730000	0.0346175000	0.0345779800

V



Meanwhile, the Bubble Sort was only fast when I used the small dataset (100 input size).

runtime of the function call became considerably slower.

however that is not the case because Bubble Sort runs faster than Merge Sort if we use the small

nt to use if we will have larger datasets.

t slower and when I used the large dataset,

.

r empirical analysis?

est used for larger dataset input.

ns?

is proves to be accurate based on our observation with our code
orithm was more effecient to use especially if we have big datasets.