Name: Jinhao Chen

Class: COT4400

Project 1

Results

	SC	SS	SR	IC	IS	IR	MC	MS	MR	QC	QS	QR
1000	0	0	0	0	0	0	0	0	0	0	0	0
1500	0	0	10	0	0	0	0	0	0	0	0	0
3000	20	20	20	0	0	20	0	0	0	20	0	0
5500	70	80	80	0	0	70	0	0	0	60	0	0
10000	250	260	270	0	0	220	0	0	0	190	0	0
15000	550	590	600	0	0	500	0	0	0	430	0	0
30000	2220	2380	2380	0	0	1990	0	0	10	1720	0	0
55000	7450	8000	8010	0	0	6700	10	10	20	5790	0	10
100000	24680	26440	26480	0	0	22110	20	20	40	19150	10	30

Analysis

Selection Sort				
	T(max)	T(min)	Ratio	Behavior
Constant	100000	3000	1234	Quadratic
Sorted	100000	3000	1322	Quadratic
Random	100000	1500	2548	Quadratic

Selection Sort:

Compare this result with the theoretical analysis table. I see that what I get for those three cases behavior is the same with result in theoretical analysis table. My result make sense because in this algorithm, we will have two for loop, inner for loop is try to find the min after sort min value to the first data, so it should be the same with theoretical analysis table.

Insertion Sort				
	T(max)	T(min)	Ratio	Behavior
Constant	0	0	0/0	Undefined
Sorted	0	0	0/0	Undefined
Random	100000	3000	1105.5	Quadratic

Insertion Sort:

Compare this result with the theoretical analysis table. For the constant and sorted, I did not get any max running time. So my ratio is all 0 for these two cases. Therefore my behavior is undefined in this case. In my opinion, the insertion sort checks the value there against the largest value in the sorted list, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position. However in constant and sorted cases, we get the same values all the time, therefore it does not need take more time to find the largest value. The random case is different, because the random list will be different. Each time program will need to find the large data and insertion the data; therefore in random this case gets quadratic behavior.

Merge Sort				
	T(max)	T(min)	Ratio	Behavior
Constant	100000	55000	2	n lgn
Sorted	100000	55000	2	n lgn
Random	100000	30000	4	n lgn

Merge sort:

Compare this result with the theoretical analysis table. I get the same behavior with the theoretical analysis table. My result makes sense because in this algorithm is based on the divide-and-conquer paradigm. First the array divide to two sub array, and then do conquer step, finally, combine the sub array to array to get sorted list. Therefore it just depends on the size of array to divide into sub array, therefore, it takes more time when it goes to high, so all three case will get same behavior, the random take more time because it will swap during the sub array.

Quick Sort				
	T(max)	T(min)	Ratio	Behavior
Constant	100000	3000	957.5	Quadratic
Sorted	100000	55000	10/0	Undefined
Random	100000	55000	3	Quadratic

Quick sort:

Compare this result with the theoretical analysis table, the behavior of my constant is quadratic, sorted is undefined, and random is quadratic. In this algorithm, Quick Sort chooses as pivot one of the items in the array to be sorted. Then array is then partitioned on either side of the pivot. Elements that are less than or equal to pivot will move toward the left and elements that are greater than or equal to pivot will move toward the right. The best case for quick sort, it should take the pivot from the middle, and in our algorithm, our pivot takes from median with first value and middle value and last value, so I guess this is why I got constant and random case quadratic.