Author: Harley Phung Case ID: hkp15 Assignment 5: Programming Assignment

Benchmarking

1. *What system did you run your tests on? Describe the relevant specifications. What factors might have affected the timing results?*

- I run my tests on Windows 10 Pro, version 21H2. The Processor is Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz. System type is 64-bit operating system, x64-based processor. I think that the processor and my laptop's age significantly affected the performance, hence the timing results.

2. *Which method(s) is (are) faster for large n? How does each method scale with n for random input? Show your results either in a table or with a plot.*

- For large n (100, 200, 500, 1000, 2000, 5000) I find that Upgraded Quicksort is the fastest method.
- For random input of large n, the Upgraded Quicksort is the fastest, then Quick Sort and Merge Sort
- For random input of small n (10, 20, 50), Insertion Sort is the fastest algorithm, and the Bubble Sort

Table 1: Timing results of Insertion Sort, Bubble Sort, Shell Sort, Quick Sort, Merge Sort, Upgraded Quick Sort with size n being 10, 20, 50, 100, 200, 500, 1000, 2000, 5000.

N	Insertion Sort	Bubble Sort	Shell Sort	Quick Sort	Merge Sort	Upgraded Quick Sort
10	1100	1600	4100	1800	3800	2000
20	1100	2800	1800	1200	3700	500
50	6800	8500	5300	1300	6400	400
100	19500	18300	14050	2400	14500	400
200	10200	80600	19800	2300	31900	300
500	59000	598200	28100	3900	103200	200
1000	118900	1622900	400	45700	332000	200
2000	479800	6666300	400047	16800	311000	200
5000	3173100	42968100	295623	36200	822500	300

3. *Is your answer to the previous question consistent with the theory? Explain.*

- My answer is consistent with the theory. This is due to the fact that in theory, insertion sort's timing is quadratic when it comes to a small number of inputs. With large number of input, the time complexity of Upgraded Quick Sort algorithm is nlog(n), faster than insertion sort (n^2).

4. *Is insertion sort ever preferable? Your answer should include examples with benchmarks.*

- Insertion sort is only preferable when there are small number inputs. According to table 1, the time complexity for insertion sort of input N = 10 is the smallest (1100 nanoseconds). Or the time complexity for insertion sort of input N = 20 is the smallest comparing to the other algorithm (1100 nanoseconds).

5. *Are there input cases where one method performs poorly? Again, show examples with Benchmarks.*

- When it comes to a small number of inputs, Quick Sort, Merge Sort, and Upgraded Quick Sort is inefficient (1800, 3800, 2000 nanoseconds respectively at input N = 10).
- When it comes to a large number of inputs, Insertion Sort, Bubble Sort, and Shell Sort is inefficient (3173100, 42968100, 295623 nanoseconds respectively at input N =5000).

6. *Make a table to show for each of the input sizes (n = 10,20 ...) and input type (random, sorted, reverse-sorted) which sorting algorithm is optimal and how much time they take.*

Table 2: Timing results for random, sorted, reverse sorted array of Insertion Sort, Bubble Sort, Shell Sort, Quick Sort, Merge Sort,Upgraded Quick Sort with size n being 10, 20, 50, 100, 200, 500, 1000, 2000, 5000.

N	Array	Insertion Sort	Bubble Sort	Shell Sort	Quick Sort	Merge Sort	Upgraded Quick Sort
N = 10	Random	1100	1600	1000	1800	3800	2000
	Sorted	900	2000	2200	800	900	200
	Reversed Sorted	900	600	1300	500	1000	1000
N = 20	Random	1100	2800	1800	1200	3700	500
	Sorted	300	1500	1400	1400	3400	2000
	Reversed Sorted	1700	2400	1000	1100	3200	400
N = 50	Random	6800	8500	5300	1300	6400	400
	Sorted	800	10000	1400	1100	4800	200

	Reversed Sorted	8000	11800	3900	2900	9900	200
N = 100	Random	19500	18300	14050	2400	14500	400
	Sorted	1400	5100	7000	1700	18400	200
	Reversed Sorted	39400	8800	8300	4000	11400	200
N = 200	Random	10200	80600	19800	2300	31900	300
	Sorted	90000	31500	60000	150000	22400	300
	Reversed Sorted	50900	51600	52700	15300	24500	300
N = 500	Random	59000	598200	28100	3900	103200	200
	Sorted	1800	78700	45100	9800	153500	300
	Reversed Sorted	62200	82100	64700	21900	150000	800
N = 1000	Random	118900	1622900	40000	45700	332000	200
	Sorted	2900	527200	40100	28200	127000	300
	Reversed Sorted	243700	300600	234100	46200	50700	300
N = 2000	Random	479800	6666300	400047	16800	311000	200
	Sorted	6500	1548100	169800	12200	176800	300
	Reversed Sorted	2665600	1118500	2124000	106700	191500	300
N = 5000	Random	3173100	42968100	295623	36200	822500	300
	Sorted	10400	7994800	449300	34600	235300	200
	Reversed Sorted	6541300	8603100	6264000	100600	1210500	2200