Ran Ju #1621899 HW5 Sort Explorer

CSE 373

1.

Alpha

DataType	InOrder	ReverseOrder	AlmostOrder	Random
Comparisons	15	120	34	103
Movements	30	150	51	125
Total time	0	0	0	0

Beta

DataType	InOrder	ReverseOrder	AlmostOrder	Random
Comparisons	120	120	120	120
Movements	0	24	3	45
Total time	1	0	0	0

Gamma

DataType	InOrder	ReverseOrder	AlmostOrder	Random
Comparisons	85	72	83	80
Movements	136	120	133	127
Total time	0	0	0	0

Delta

DataType	InOrder	ReverseOrder	AlmostOrder	Random

Comparisons	150	158	111	125
Movements	15	39	18	66
Total time	0	0	0	0

Epsilon

DataType	InOrder	ReverseOrder	AlmostOrder	Random
Comparisons	46	55	55	51
Movements	36	80	59	73
Total time	0	0	0	0

Zeta

DataType	InOrder	ReverseOrder	AlmostOrder	Random
Comparisons	32	32	41	47
Movements	128	128	128	128
Total time	0	0	0	0

2

ArraySize(Alpha)	5000	10000	30000	80000
InOrder	0	0	0	0
ReverseOrder	24	92	781	5497
AlmostOrder	2	7	55	342
Random	10	48	398	2693

ArraySize(Beta)	2000	5000	10000	30000

InOrder	4	26	88	783
ReverseOrder	4	25	91	768
AlmostOrder	4	22	90	761
Random	5	27	90	772

ArraySize(Gamma)	5000	20000	100000	1000000
InOrder	0	2	15	143
ReverseOrder	0	2	13	141
AlmostOrder	0	3	15	151
Random	1	3	20	212

ArraySize(Delta)	1000	3000	5000	7000
InOrder	1	9	23	47
ReverseOrder	1	10	26	46
AlmostOrder	0	0	1	0
Random	0	0	0	1

ArraySize(Epsilon)	10000	50000	500000	1000000
InOrder	0	1	19	40
ReverseOrder	1	3	33	63
AlmostOrder	0	2	25	50
Random	1	5	54	104

ArraySize(Zeta)	100000	500000	800000	1000000

InOrder	12	64	91	118
ReverseOrder	11	59	92	114
AlmostOrder	20	76	108	137
Random	21	107	166	213

3.

Sort	Alpha	Beta	Gamma	Delta	Epsilon	Zeta
Running Time	$O(n^2)$	$O(n^2)$	O(n*logn)	$O(n^2)$	O(n*logn)	O(n*logn)

4.

4.1 Alpha

Alpha is insertion sort for the following reasons:

Firstly, the runtime of alpha based on question 2 shows that Alpha runtime is O(n^2). Therefore, the alpha mush be the selection, or inserting, or quick sort(simple).

Moreover, we focus on the four different orders, the comparison of in-order is minimum compared with other orders (based on the question 1), and 15 means 16-1 = 15. Also, the comparison of Reverse Order is 150 which means that the reverse input of size 16 is 1+2+3+4…+15 = 120. Thus, as we can see, both reverse order and in order correspond with insertion sort. Also, the runtime between in-order and reverse-order have a great difference which means the worst case and best case have a great difference.

4.2 Beta

Beta is selection sort for the following reasons.

Firstly, the runtime of Beta based on question 2 is $O(n^2)$. Also, in the question 1, it shows that the comparison of those different input order is the largest one among those different sorting. To be more specific, the comparison of input order is 120 which means no matter the order is, each element should compare 1+2+3+4…+15 = 120 times. In the movement, in-order has 0 time move. Which mean it should not move anything. Also, the runtime among those input-orders have litter difference which means the worst case and the best case can be regard as the same.

4.3Gamma

Gamma is heap sort for the following reasons.

Firstly, the runtime of Gamma based on question 2 is O(n*logn), which means it only possible heap sort, quick sort(optimized) and merge sort. Also, in those four different input order, the reverse-order has smallest comparison and movement. It is because heap sort uses deleteMax to find the minimum which means reverse-order should have less movement and comparison than others. Moreover, the movement and comparison

has little vary among those four input orders, which also can identify that gamma is heap sort.

4.4 Delta

Delta is quick sort(simple) for the following reasons. The runtime of Delta is O(n^2). Also, in the simple quick sort, the first element as the pivot to compare with other elements. Therefore, in the in-order and reverse-order, the pivot is the smallest one or largest one which means it should do more comparison compared with almost-order and random. Moreover, in the movement, the simple quick sort don't need more extra movements in in-order. Based on those reason, it can identify that Delta is simple quick sort.

4.5Epsilon

Epsilon is quick sort(optimized) for the following reasons. First, the runtime of Epsilon is O(n*logn). And based on question 1, the comparison of different input is nearly same. It is because optimized quick sort uses the median of three for pivot selection. Also, the movement of reverse-order should be larger than other input order.

4.6Zeta

Zeta is merge sort for the following reasons.

Firstly, the runtime of Zeta based on question 2 is O(n*logn). Moreover, in the question 1, the movement of Zeta in four different types order is 128. As we all know, 16 means 2^4 so that we should use 4 steps to separate the whole array to each one element. Thus, in merge sort, it must move 16*4*2 = 128 steps to make the array sorted regardless of the input order.