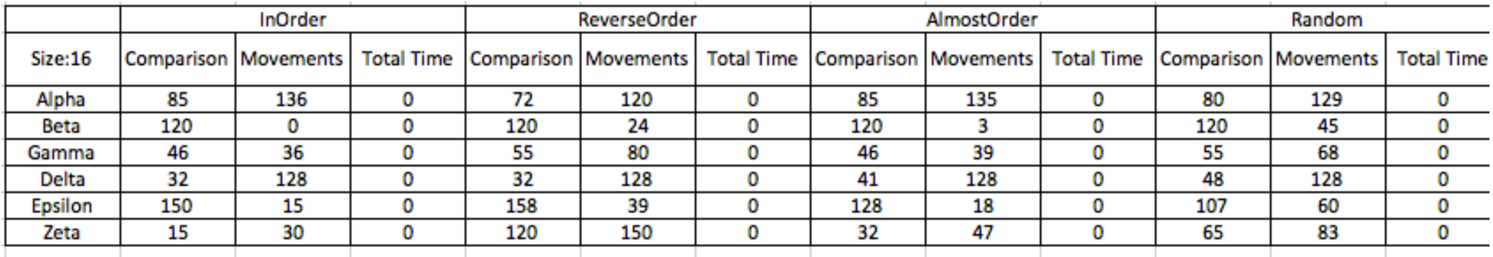
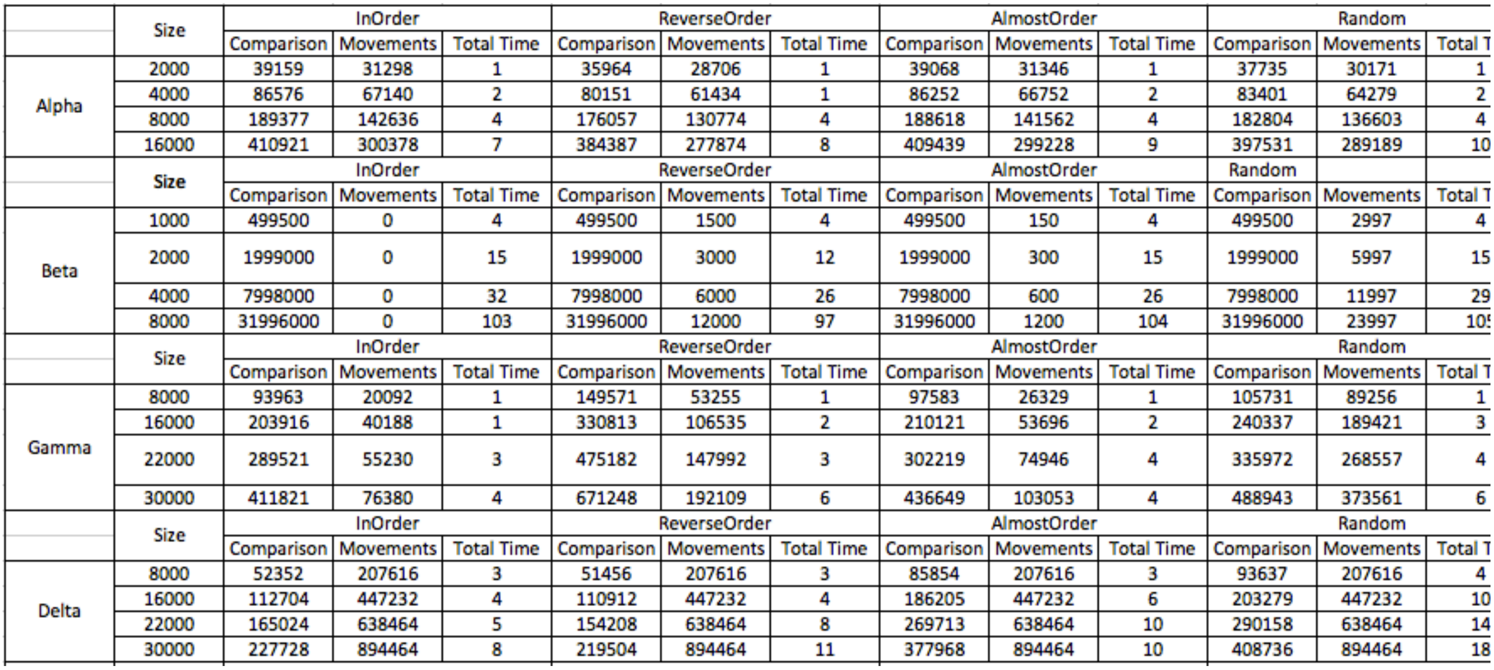
Daniel W. Chen
Aaron Bauer
CSE 373

(1)



	InOrder			ReverseOrder			AlmostOrder			Random		
Size:16	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time
Alpha	85	136	0	72	120	0	85	135	0	80	129	0
Beta	120	0	0	120	24	0	120	3	0	120	45	0
Gamma	46	36	0	55	80	0	46	39	0	55	68	0
Delta	32	128	0	32	128	0	41	128	0	48	128	0
Epsilon	150	15	0	158	39	0	128	18	0	107	60	0
Zeta	15	30	0	120	150	0	32	47	0	65	83	0

(2)



	Size	InOrder			ReverseOrder			AlmostOrder			Random		
		Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total T
Alpha	2000	39159	31298	1	35964	28706	1	39068	31346	1	37735	30171	1
	4000	86576	67140	2	80151	61434	1	86252	66752	2	83401	64279	2
	8000	189377	142636	4	176057	130774	4	188618	141562	4	182804	136603	4
	16000	410921	300378	7	384387	277874	8	409439	299228	9	397531	289189	10
	Size	InOrder			ReverseOrder			AlmostOrder			Random		
		Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total T
Beta	1000	499500	0	4	499500	1500	4	499500	150	4	499500	2997	4
	2000	1999000	0	15	1999000	3000	12	1999000	300	15	1999000	5997	15
	4000	7998000	0	32	7998000	6000	26	7998000	600	26	7998000	11997	29
	8000	31996000	0	103	31996000	12000	97	31996000	1200	104	31996000	23997	10
	Size	InOrder			ReverseOrder			AlmostOrder			Random		
		Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total T
Gamma	8000	93963	20092	1	149571	53255	1	97583	26329	1	105731	89256	1
	16000	203916	40188	1	330813	106535	2	210121	53696	2	240337	189421	3
	22000	289521	55230	3	475182	147992	3	302219	74946	4	335972	268557	4
	30000	411821	76380	4	671248	192109	6	436649	103053	4	488943	373561	6
	Size	InOrder			ReverseOrder			AlmostOrder			Random		
		Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total Time	Comparison	Movements	Total T
Delta	8000	52352	207616	3	51456	207616	3	85854	207616	3	93637	207616	4
	16000	112704	447232	4	110912	447232	4	186205	447232	6	203279	447232	10
	22000	165024	638464	5	154208	638464	8	269713	638464	10	290158	638464	14
	30000	227728	894464	8	219504	894464	11	377968	894464	10	408736	894464	18

(3)
Alpha: O(NLOGN)
Beta: O(N^2)
Gamma: O(N^2)
Delta: O(NLOGN)
Epsilon: O(N^2)
Zeta: O(N^2)

(4)
Alpha: Heap Sort
According to the data that I have collected, it showed that for each different order list, they all have similar run time, comparison and movement. This result could be reflected to the characteristic of Heap Sort. In the sorting operation of heap sort, it will always build up a heap for start and use deletemin() to sort. So, no matter what kind of order the list was given, the run time, comparison and movement will always shown to be similar.

Gamma: Quick Sort Optimized
The data I have collected showed that the run time in each order list sort happened to be similar. The result could be reflected to the characteristic of Quick Sort Optimized. The algorithm of Quick Sort Optimized will take the median of three as the pivot and uses a cutoff value. As a result, the run time for each order should be close to O(NLOGN).

Delta: Merge Sort
The data I have collected showed that the number of movements happened to be similar in each different kind of ordered list. The result could be reflected to the characteristic of Merge Sort. Merge Sort will always have the same number of times splitting the elements and merging them back.

Epsilon: Quick Sort
According to the data that I have collected, it showed that the number of runtime is very large and with an almost quadratic growth rate with the list order inorder and reverse order. However, the list order of random appeared to have linear run time. These results could be reflected to the characteristic of Quick Sort Simple. In the algorithm of Quick Sort Simple, it will always take the first element in the list as its pivot. As a result, in inorder and reverse order, it will always pick the largest or the smallest element and cause to have the worst case run time O(N^2). On the other hand, the random order will tend to have the run time of O(NlogN).

Zeta: Insertion Sort
The data I have collected showed that the runtime of inorder appeared to have linear result and the runtime of reverse order appeared to have almost quadratic. These results could be reflected to the characteristic of Insertion Sort. Insertion sort will have it best run time for inorder in O(n) runtime while the worst runtime O(N^2) in reverse order list.

Beta: Selection Sort
The data I have collected showed that the movements of inorder list sort always have the result of zero. Also, the run time appeared to be having an almost quadratic growth rate. In each different order list, they all have exactly the same number of comparisons. These results could be reflected to the characteristic of Selection Sort. No movements are needed in inorder selection sort. Also, the worst, best and average case of run time are all O(N^2).