Suleyman Hanyyev
CS201 HW 2
02.04

The result table as well as the graphs are provided on the next page. Here I would like to answer the questions given by the assignment:

5.1 Specifications of my computer:

Proc: I7 2nd gen. dual-core 2.8Ghz
16GB of Ram DDR2
MacOS Catalina

5.2 What are the theoretical worst, average, and best cases for each algorithm? What are the expected running times for each scenario for each algorithm?

In algorithm 1, (theoretical) the best case scenario is when the key element is the first element in the array. The average case is when the key element is in the middle of the array. The worst case scenario, is when the key element is not in the array.

In algorithm 2, the best case scenario is when the key element is the last element in the array. The average case scenario, is similar to the average case of the first algorithm. Furthermore, the worst case scenario is when the element is not in the array.

In algorithm 3, the best case scenario is when the key element is in the middle of the array O(1). The average case scenario and the worst case scenarios are somewhat similar in time complexity and equal to O(logn). The worst case scenario is when the element is not found in the array.

Theoretically, algorithm 4 should be better than linear search but worse than binary. In algorithm 4, the best case scenario is when the key is the first element of the array. The average case is a bit difficult to calculate, it is time complexity is similar to the worst case scenario when the element is not in the array, which is O(√n).

5.3. What are the worst, average, and best cases for each algorithm based on your table and the plots? Do theoretical and observed results agree? If not, give your best guess as to why that might have happened.

In algorithm 1, the worst case is when the key is not in the array. The average case is b) which is when the element is in the middle. And the best case occurs when a) the element is in the beginning of the array. Here, the theoretical and observed results agree.

In algorithm 2, the worst case is when the key is not in the array. The average case is b) which is when the element is in the middle. And the best case occurs when c) the element is in the end of the array. Here, the theoretical and observed results agree.

In algorithm 3, the worst case is when the key is not in the array. The average case is similar to the worst case in time complexity. And the best case occurs when c) the element is in the middle of the array. Here, I could not give a fully correct answer since the input N is still small for the algorithm, and the time elapsed is still very little, even when N= 250000.

In algorithm 4, the worst case is when the key is not in the array. The average case is b) which is when the element is in the middle. And the best case occurs when a) the element is in the beginning of the array. Same as in algorithm 3, here the theoretical and observed results do not fully agree since the N= 250000 is still small for this algorithm to make a conclusion of its time complexity. It is more like the functions do find the index momentarily.

The time provided down below is in microseconds:

N	Linear (Iterative)				Linear (Recursive)				Binary Search				Jump Search			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
10	2	2	1	1	2	2	1	1	1	1	2	1	16	2	2	2
100	1	1	2	2	1	2	1	1	1	1	1	1	2	2	2	2
500	2	3	3	4	6	4	1	6	2	1	1	2	1	2	3	4
1000	1	2	4	5	22	5	1	9	1	1	1	1	2	2	2	2
5000	2	9	16	17	86	20	5	41	1	1	1	2	2	2	2	1
10000	3	22	27	28	131	30	8	73	1	1	1	1	1	2	2	2
25000	6	42	68	72	342	108	19	176	8	1	1	2	2	1	3	1
50000	11	69	131	272	766	150	33	334	1	2	2	1	2	2	5	1
75000	12	98	185	196	712	223	39	513	8	1	1	1	2	4	4	1
100000	17	138	251	281	1518	551	63	954	2	1	1	1	3	4	7	1
150000	23	219	363	416	1502	484	92	1119	1	1	3	1	3	8	10	1
200000	37	327	491	546	1972	735	119	1563	3	1	1	1	3	15	19	2
250000	37	461	791	1482	2430	1292	423	3442	3	2	1	1	3	11	15	1