Input Size	Bubble Sort running time (s)	Selection Sort running time (s)	c Sort running time (s)
100	0.000074	0.000022	0.000009
1000	0.005183	0.001853	0.000092
10000	0.250502	0.085859	0.001231
100000	27.413128	5.774565	0.018153
1000000	N/A	N/A	0.105246
10000000	N/A	N/A	1.028767

1 - Explain what you think the worst-case big O complexity and the best-case big O complexity of bu

Worst case Big O is n*(n -1)/2. Best case big. Best case Big O is the O(n^2). Fro the w

2 - Is there a more efficient way to write bubble sort that changes the performance in the best case?

for the inner loop. We only need to increment the j pointer to size - i -1. Since the last l

3 - Explain what you think the worst-case big O complexity and the best-case big O complexity of sel

Worst case Big O is n*(n -1)/2. Best case big. Best case Big O is the O(n^2). Fro the w

4 - Does selection sort require any additional storage (i.e. did you have to allocate any extra memory

No it is an in-place sorting algorithm. It does not need additional space

Explain what you think big O complexity of sorting algorithm that is built into the C libraries is. Why d

The Big O of standard sort algorithm of C library is nlogn. Becase it is a quick sort.

bble sort is as implemented in our code. Give reasons for why you think that is the big O complexity f
orst case, the given array is in descending order, we need to swap every single pairs. But for the best

? If so, describe (in general terms, we don't need the exact C code) how that implementation of bubbl

orst case, the given array is in descending order, we need to swap every single pairs. But for the best

case, we only need to traverse all the pair without any swap.