Data Files	*N*	Trial 1	Trial 2	Trial 3	**Average**
Random Order	*2000*	0.021	0.025	0.022	**0.0226**
	4000	0.056	0.054	0.057	**0.0556**
	8000	0.213	0.205	0.211	**0.2097**
	1600	1.106	1.09	1.198	**1.1313**
Sorted Order	*2000*	0.022	0.034	0.024	**0.0267**
	4000	0.049	0.052	0.053	**0.0513**
	8000	0.170	0.175	0.168	**0.1710**
	1600	0.907	0.877	0.946	**0.9100**
Reversed Order	*2000*	0.023	0.028	0.024	**0.0250**
	4000	0.064	0.064	0.065	**0.0643**
	8000	0.237	0.241	0.236	**0.2380**
	1600	1.03	1.026	1.025	**1.0270**

3. According to the doubling hypothesis, if the input doubles and the runtimes double, the runtime for this function is linear. SInce this is true for this function, this function is linear. For example, the runtime approximately doubles for Random Order form 0.0226 to 0.0556 when we double the input from 2000 to 4000.

4. Assuming that calling less() does not add to runtime, we see there are two array accesses in two nested for loops, therefore the rate of growth for selection sort is $\sim N^2$.

5. Looking at the selection sort source code, **less** and **exch** are always called. This is because the if condition in the nested for loop is the only condition, so it always calls within the nested for loop and we can see that the exch() function is within the first for loop which runs from 0 to N. Therefore, there should exist no significant time difference between sorted order, reversed order, and random order.