E)

-Case A (Worst case) :

I used a function to calculate the general time it took to run the algorithm as a function of the number of elements in the array. For increasing values of n or number of elements , the time it took to run the algorithm increased drastically as n increased at greater rates. The resulting graph was an increasing parabola. The function I came up with in calculating this time is

Y = 7x – 14 + 3(x - 1)(x – 2) + x + 3.

I assumed a unit time of 1 for a single operation to generate this function for the selection sort algorithm.

* Asymptotic behavior : The graph or plot approaches infinity as the number of elements approaches infinity so it belongs to both the small and big Omega relationships as the limit as number of elements approaches infinity is both greater than 0 and equals infinity so it is bounded tightly below by Big Omega and loosely below by small Omega.
* In regards to constants , for a constant X and n number of elements, the time it takes to run is X multiplied by Ω(n2).

OR RUNTIME = X\* Ω(n2) for a constant X.

-Case B (Best case) :

In the best case, there are no swaps done as the best case is when the array is already ordered.

So the function is going to be just a constant or number just for the first initializations for the variables and the first and one and only comparisons in the loops and if statements. The resulting graph is a horizontal line.

Like Y = k. for constant k

* Asymptotic behavior : The graph or plot belongs to Big O asymptotic behavior and it is tightly upper bounded.
* Runtime for best case scenario does not depend on number of elements. So the runtime is always equal to the original function for increasing number of elements.

SO RUNTIME = CONSTANT = k. (Original time k) always.

-Average Case :

The average case plot is a slightly increasing parabola with increasing number of elements.

-Asymptotic behavior : The plot belongs to Theta(Θ) relationship and is tightly bounded both above and below.

-Runtime : Runtime for a constant X and n number of elements, is X multiplied by Θ(n2).

SO RUNTIME = X\*Θ(n2).