1. Yes, this algorithm is correct. Since a main job of sorting algorithms is to swap elements in the sequence to make it in a sorted order.

2. We would expect the algorithm to take $\Omega(1)$ time in the best case. In this case, the sequence only have two out-of-place elements and the algorithm happened to choose the two elements in the first iteration of the loop; we only iterated the loop once and leave, since the sequence is now sorted.

3. In the worst case, this algorithm will run forever. In this case, the algorithm happened to always choose the wrong elements to swap, and we never arrived at the state that the sequence is sorted, the loop is an infinite loop.

4. On average, we will expect it to take around $O(n!)$ time.
Assume every element in the sequence is unique (for simplifying calculations, the overall complexity should not change if sequence has duplicate elements).
The number of possible permutations for the sequence will be $P(n, n) = n!$; and only one permutation is the correct order.
We would expect that, by swapping the values randomly, we will, on average, reach a multiple of all possible permutations, before arriving at the correct answer and stop the execution.
Which means that the loop will be iterated $c * n!$ times; for some constant c.
Therefore, the average-case time complexity for this algorithm is $O(n!)$.