```
public static int[] insertionSort(int[] A, int n) {
	for (int i = 1; i < N; ++i) {
		int key = A[i];
		int j = i - 1;'

		while (j >= 0 && A[j] > key) {
			A[j + 1] = A[j];
			j = j - 1;
		}
		A[j + 1] = key;
	}
	return A;
}
```

1. for(int i = 1; i < N; ++i): This is a loop that runs (N-1) times.

2. int key = A[i]; int j = i - 1; These are simple assignments which take constant time, O(1).

3. while(j >= 0 && A[j] > key): The worst-case scenario for this inner loop is when the input array is reverse sorted, and it has to traverse through all the sorted elements. In the worst-case scenario, this takes O(n) time. In the best case, when the array is already sorted, it takes O(1) time. On average, it takes O(n/2) time.

4. A[j + 1] = A[j]; j = j - 1; : These are simple assignments inside the while loop and take constant time per iteration of the loop. They are part of the time complexity calculated for the inner loop.

5. A[j + 1] = key; : This is a simple assignment which takes constant time, O(1).

Applying the sum and multiplication rules, the time complexity in the worst case is: O(1) * N + O(1) * N + O(N*(N-1)/2) + O(1) * N = O(N^2).

So, the time complexity of insertion sort in the worst case is O(N^2).