**SortOfSort Performance**

**public static void** sortOfSort(**int**[] array) { **Steps**

**int** n = array.**length**; 1  
 **int** count = 0; 1  
  
 **while**(n >= count + 2) { 18n / 2 +2  
  
 **for** (**int** i = n - 1; i >= n - 3; i--) { 7n2  
 **int** maxIndex = i; 1  
 **for** (**int** j = i - 1; j >= count; j--) { 2n3  
 **if** (array[j] > array[maxIndex]) { 1  
 maxIndex = j; 1  
 }  
 }  
 **int** temp = array[maxIndex]; 1  
 array[maxIndex] = array[i]; 1  
 array[i] = temp; 1  
 }  
  
 n -= 2; 1  
  
 **for** (**int** i = count; i < 2 + count; i++) { 7n2  
 **int** maxIndex = i; 1  
 **for** (**int** j = i + 1; j < n; j++) { 2n3  
 **if** (array[j] > array[maxIndex]) { 1  
 maxIndex = j; 1  
 }  
 }  
 **int** temp = array[maxIndex]; 1  
 array[maxIndex] = array[i]; 1  
 array[i] = temp; 1  
 }  
  
 count += 2; 1  
 }  
}

**Results: 4n3 + 14n2 + 9n + 16 = 0(n3)**

The performance of sortOfSort() method is 0(n3) because in order to sort the array, the method utilices 2 nested for loops inside a while loop. The for loops check for the largest number in a certain part of the array while the while loop reiterates the whole process if the array is not completely sorted.

**Worst case: 0(n3)**

As the array is sorted completely and the while loop repeats.

**Best case: O(n2)**

Best case is O(n2) if the while loop is not considered and the two nested for loops only run once each. If the array is almost sorted or the the array is ≤ 4 elements long.