Diana A Garcia
CS2401 Lab
Week 7

SortOfSort Performance Analysis

SortOfSort

Instructions that run once.	Instructions that will repeat.
int runTimes = 0;	while(start!= end)
int start = 0;	for(int i = 0; i < nums.length; i++)
int end = nums.length -1;	

3 + 8n + $4n^2$

While Loop

Instructions that run once.	Instructions that will repeat.
	int maxIndex= maxIndex(nums,start, end);
	if (runTimes % 4 < 2) **OR** else

0 + 5n + 3n +(4n)n

maxIndex

Instructions that run once.	Instructions that will repeat.
int i = start	int maxIndex = start
i = end	i <= end;
return maxIndex	i++
	maxIndex = i

3 + 4(n)

If Statement or Else Statement (Since only one will run a time)

Instructions that run once.	Instructions that will repeat.
int temp = nums[end];	
nums[end] = nums[maxIndex];	
nums[maxIndex] = temp;	
end--;	
runTimes++;	
*All these also run with the else but with other value)	

5 +

Diana A Garcia
CS2401 Lab
Week 7

Print For Loop (Since this for loop is just for me to see the array, not to sort it, I did not include it in my theoretical analysis)

Instructions that run once.	Instructions that will repeat.
Int i = 0;	i < nums.length;
	System.*out*.print(nums[i] + " ");
	i++;

1 + 3(n)

T(n) = $4n^2 + 8n + 3$
O(n) = O(n^2)

Best Case, Worst Case, Average Case

The best case for the performance of the method is when the array has less than one integer. The reason for this is that the method will not have to sort anything and can just return what is in the array. Now the worst and average will be n^2, since this will happen no matter how long the array is.