Theoretical Analysis:

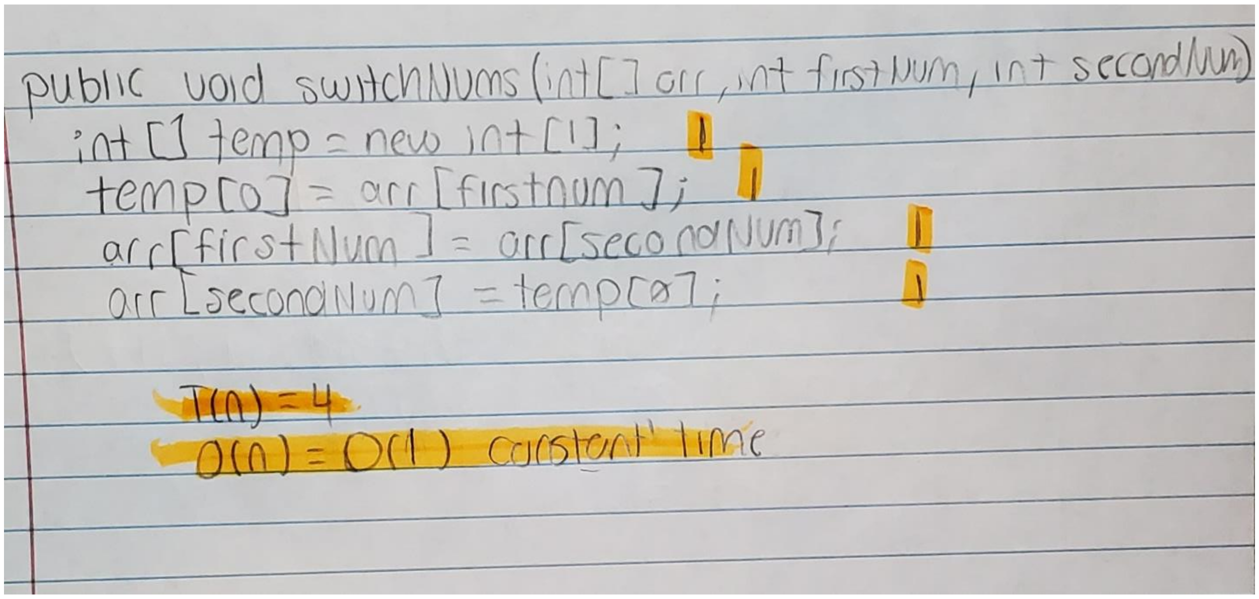
My sortOfSort algorithm begins with 5 variable initializations and an if statement. Next there is a main for loop and within it resides two nested if statements and four calls to my two helper methods.

Best case:

The best case scenario for my program would be an array length of 1. Since it would technically be sorted, it would only execute the first 5 variable initialization and the outside if statement. I calculated the **t(n) = 6** or the **O(n) = O(1**) which is constant time since the program will always execute these 6 steps.

Worst case/Average Case:

Both of my helper methods: findMax and switchNums have a time complexity of O(n) and O(1) respectively. Hence the time for main for loop would be 2 once commands and 7 repeating, which is 2+ (7*n)*(O(n)). On average my program would have a running time **t(n) = 6 + 7*(n^2)** and an **O(n) = (O)(n^2).** The best and worst case are the same because the both depend on the n size of the array.



```
public void switchNums(int[] arr, int firstNum, int secondNum)
  int[] temp = new int[1];            1
  temp[0] = arr[firstnum];            1
  arr[firstNum] = arr[secondNum];         1
  arr[secondNum] = temp[0];               1

    T(n) = 4
    O(n) = O(1) constant time
```

```
public int findMax (int [] arr, int beginIndex, int endInd)

int maxNum = arr [begin Index ]; 1
int index  = 0;                  1

For(int i = 0; i< arr.length - end Ind; i++){
        if (arr[i] >= maxNum){
            maxNum= arr[i];
            index = i;
        }
 }
return index   1
}
```

once	repeating
int i = 0	if (arr[i] >= maxNum)
= arr.length - endI	maxnum = arr[i];
	index = i;
	i++
2	4 * n

$1 + 1 + 1 + \cancel{2} + 4 * n$

$3 + 2 + 4(n)$

$T(n) = 5 + 4n$

$O(n) = O(n)$ linear time

```
~~int startIndex = 0;~~          ~~1~~
int maxRightIndex = 0;          1 ⎫
int maxLeftIndex = 0;           1 ⎬ 4
int switchSides = 0;            1 ⎪
int maxIndex;                   1 ⎭
```

```
for(int i = 0; i <= (arr.length/2)+2; i++){
```

once	repeating
int i = 0	i < (arr.length/2)+2
i = (arr.length/2)+2	max = findmax (arr, start, max
	if (go right)
	switchNums()
	maxRightIndex ++;
	switchSides ++
	if (switchsides == 2)
	go Right = false;
	Switchsides = ∅
	else
	switchNums()
	maxleftIndex ++
	switch sides ++
	if (switchsides = 2)
	goRight = true
	switchsides = ∅
2	

$2 \quad + \quad 7n \cdot O(n) \cdot O(1)$

$4 + 2 + 7n + O(n) + O(n)$

$6 + 7n + O(n^2)$

$T(n) = 6 + 7n + O(n^2)$

$O(n) = O(n^2)$