# Performance: sortOfSort Method

While performing my theoretical analysis, I created three tables to count the number of static and repeating instructions for each of my for-loops.

***First for-loop:***

|  |  |
| --- | --- |
| once | repeating |
| int i = 0  i < repeat | i < repeat  int tempHoldR =  int tempHoldL =  **for(right side)** …  if()  break  **for(left side)** …  if()  break  i++ |
| 2 instructions + | 64 \* *repeat* |

**= 2 + 64\**repeat***

***Nested for-loop: right side***

|  |  |
| --- | --- |
| once | repeating |
| int r = 0  r < 2 | r < 2  if()  break  max =  maxInd =  arr[] =  if()  arr[] =  tempRight - -  if()  if()  tempHoldR =  r++ |
| 2 instructions + | 13\*2 instructions |

**= 28 instructions**

***Nested for-loop: left side***

|  |  |
| --- | --- |
| once | repeating |
| int l = 0  l < 2 | l < 2  if()  break  max =  maxInd =  arr[] =  if()  arr[] =  tempLeft - -  if()  if()  tempHoldL =  l++ |
| 2 instructions + | 13\*2 instructions |

**= 28 instructions**

If **n** is defined as the length of the array, the value ***repeat*** is defined as **n**/4 or **n**/4 + 1 depending on whether the length is even or odd, respectively. In the case of **n** being an even number, the time complexity of sortOfSort is: **T(n) = 2 + 16n**.If **n** is odd, **T(n) = 66 + 16n**.