Dennis Kuzminer
CSCI-UA 310-001 PS1

5.

a. The first for-loop creates a runtime of n. The second for-loop creates a runtime of n again, meaning that the first for-loop's run time will not matter as $n \to \infty$. The while-loop inside of the for-loop will consistently reduce the number of operations that need to be done in comparison O(n). For instance, if n = 10, then the first iteration of the for-loop will perform 10 (* 2) operations within the while-loop. In the second iteration, 5 (* 2). Then 3 (* 2), 2 (* 2), 2 (* 2), 1 (* 2), and so on. We can see that the number of iterations/operations of the while-loop is decreasing by some proportion of n as n grows. Therefore, we can simplify the runtime of this algorithm to n+nlogn. We can simplify, we get **O(nlogn)**.

b. If we run the program for an array of integers size 12, we will be adding one to indices A[i] every for-loop iteration. However, we do not add it to every index. We add one to every ith index. For instance, if i = 4, then we would only add one to A[4], A[8], and A[12]. At the end of the 12th iteration of the algorithm, we can see that A = [1,2,2,3,2,4,2,4,3,4,2,6]. Each value of A[i] actually **represents the number of factors of a given number i**.