

## TIME COMPLEXITY SOLUTIONS

Solution :

- a. Option A -> Time complexity =  $O(n \cdot \log n)$

In the loop, j keeps doubling till it is less than or equal to n. Several times, we can double a number till it is less than n would be  $\log(n)$ .

Let's take the examples here.

for  $n = 16$ ,  $j = 2, 4, 8, 16$

for  $n = 32$ ,  $j = 2, 4, 8, 16, 32$

So, j would run for  $O(\log n)$  steps.

i runs for  $n/2$  steps.

So, total steps =  $O(n/2 * \log(n)) = O(n \cdot \log n)$

- b. Option C -> Time complexity =  $O(\log kn)$

Because loops for the  $kn-1$  times, so after taking log it becomes  $\log kn$ .

- c. Option B. -> false

The Big-O notation provides an asymptotic comparison in the running time of algorithms. For  $n < n_0$ , algorithm A might run faster than algorithm B, for instance.

- d. Time complexity -  $O(\sqrt{n})$

Space complexity -  $O(1)$

- e. Time complexity -  $O(n^2)$

Space complexity -  $O(1)$

