performing n append operations on an initially empty dynamic array with fixed increment size c

for n appends, we would have to resize n/c times

the resizing would take c operations for the first resize, 2c for the second resize, 3c for the third etc... up to mc operations for the last resize, where m = n/c

for example, if we wanted to do 1000 append operations, and we increase array size by 4 each time, we would have array sizes of:

$$\underbrace{4, 8, 12, 16 \ldots\ldots 992, 996, 1000,}_{250 \text{ of these}}$$

Generally:

$$c, 2c, 3c, 4c \ldots\ldots\ldots mc$$

where $m = \frac{n}{c}$

and for each array size, we have to copy over all the elements, so the number of steps taken for the n append operations is equal to the sum of array sizes:

$$\sum_{i=1}^{m} c\cdot i = c\cdot\sum_{i=1}^{m} i = c\cdot\frac{m(m+1)}{2} = c\cdot\frac{\frac{n}{c}\left(\frac{n}{c}+1\right)}{2} \geq \frac{n^2}{2c}$$

Oh the other hand, if we for example doubled the array size each time, then for 1000 appends we would have array sizes of:

1, 2, 4, 8, 16 ..... 512, 1024

Or, more generally:

$$\underbrace{2^1, 2^2, 2^3, 2^4 \ldots\ldots \qquad 2^{\log n}}_{\log n \text{ of these}}$$

and for each array size, we have to copy all the elements, so the number of steps taken for the n appends is equal to the sum of the series

$$\sum_{i=1}^{\log n} 2^i = \frac{1-2^{\log_2 n}}{1-2} = \frac{1-n}{-1} = n-1$$

Therefore, the time complexity for doing n appends is O(n)