1. Derive the formula for worst-case complexity

Deriving formula for worst-case complexity

$T(n) = T(n_1) + T(n_2) + Cn$
$n_1 + n_2 + 1 = n \longrightarrow n_1 \approx n_2$] general formula

Worst case ($O(n^2)$) → when you have an already sorted array.

$T(n) = T(0) + T(n-1) + Cn$

n
$0 \quad n-1 \longrightarrow cn$
$0 \quad n-2 \longrightarrow c(n-1)$
$0 \quad n-3 \longrightarrow c(n-2)$
$\ldots$
$1 \longrightarrow c(2)$

$C\{n + (n-1) + (n-2) + \ldots + 1\}$

$C\,\frac{n(n+1)}{2}$

$= O(n^2)$

2. Come up with a vector of 16 elements which incurs worst-case complexity. Manually show the workings of the algorithm until the vector is sorted.

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

no matter what I choose as the pivot it will land in the same spot and parition the left and right side.
Let's say the pivot is the last element; pivot = 16.
if the pivot is larger than the item at index j, then nothing happens and j increments by 1.

i j
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

here 16 (the pivot) is checked with every element and nothing happens since it's in the right spot in the array. This proves it is worst case since the pivot created a very uneven partition with almost the entire array on the left and an empty array on the right.

Plot from questions 3 and 4: