1. Compute asymptotic worst case time complexity of the following algorithm (see pseudocode conventions in [Cormen, Section 2.1]). You must use Θ-notation. For justification, provide execution cost and frequency count for each line in the body of the secret procedure. Optionally, you may provide details for the computation of the running time T (n) for worst case scenario. Proof for the asymptotic bound is not required for this exercise.

```
1    /* A is a 0-indexed array,
2    * n is the number of items in A */
3    secret(A, n):
4        k := 0
5        for i = 1 to n-1:
6            k := k + 1
7            j := i
8            while j < n and A[j-1] ≥ A[j]:
9                j := j + k
10           exchange A[i] with A[j]
```

Solution:

cost	time
c3	1
c4	1
c5	n
c6	n-1
c7	n-1
c8	$\Sigma_1^{n-1} (t_i/i)$
c9	$\Sigma_1^{n-1} (t_i - 1/i)$
c10	n-1

$f(n) = \Theta(g(n)) \qquad \Omega(g(n)) \le \Theta(g(n)) \le O(g(n))$

$\exists\ (c1, c2, n_0)\ \forall\ (n \ge n_0)\ |\ c1 * g(n) \le f(n) \le c2 * g(n)$

Array is already sorted - the best case - Θ(n)

the array is reverse sorted and the while loop is never ignored until i=n - the worst case – Θ(n*log(n))

If the array is reverse sorted then the algorithm has the same time complexity as the following code's algorithm:

```
| k=0
| for i in range(0, n):
|    k+=1
|    j=i
|       while(j<n):
|          j = j+k
```

$\Sigma_1^{n-1} ((t_i-1)/i) = n+n/2+n/3+n/4...+n/(n-2)+n/(n-1) = \Theta(n*\log(n-1))$

$\Theta(n*\log(n-1)) = \Theta(n*\log(n))$

2. Indicate, for each pair of expressions (A, B) in the table below whether A = O(B), A = o(B), A = Ω(B), A = ω(B), or A = Θ(B). Write your answer in the form of the table with yes or no written in each box:

A	B	A=O(B)	A=o(B)	A=Ω(B)	A=ω(B)	A=Θ(B)
$\log^5 n$	$\sqrt[4]{n}$					
n^{1000}	1.0001^n					
$n^{\cos n}$	$\log n$					
3^n	$3^{0.5n}$					

Solution:

A	B	A=O(B)	A=o(B)	A=Ω(B)	A=ω(B)	A=Θ(B)
$\log^5 n$	$\sqrt[4]{n}$	**no**	**no**	**yes**	**yes**	**no**
n^{1000}	1.0001^n	**no**	**no**	**yes**	**yes**	**no**
$n^{\cos n}$	$\log n$	**no**	**no**	**no**	**no**	**yes**
3^n	$3^{0.5n}$	**yes**	**no**	**yes**	**no**	**yes**

3. Let f and g be functions from positive integers to positive reals. Assume f (n) > n for n > 0. Using definition of asymptotic notation, prove formally that

$$\min(f(n) - n,\ g(n) + n) = O(f(n) + g(n))$$

Solution:

$\min(f(n) – n, g(n) + n) = O(f(n) + g(n))$
$\exists\ (c, n_0)\ \forall\ (n \geq n_0)\ |\ f(n) \leq c * g(n)$
$\min(f(n) – n, g(n)+n) \leq c * (f(n) + g(n))$

1st case: if (n + g(n)) is minimum then **n + g(n) < f(n) + g(n)** because f(n) > n > 0,
2nd case: if (f(n) – n) is minimum **f(n) – n < f(n) + g(n)** because g(n) > 0 and n>0, (x - positive) is always less than (x + positive)
when **c ≥ 1**, return value of **min** function is always less than **c * (f(n) + g(n)) = O(f(n)+g(n))**