counting costs

want a coarse comparison of algorithms “speed” that ignores hardware, programmer virtuosity

which speed do we care about: best, worst, average? why?

define idealized “step” that doesn’t depend on particular hardware and idealized “time” that counts the number of steps for a given input.
def LS(A,x) :
    """ Return index i such that x == L[i]. Otherwise, return -1 """
1. i = 0
2. while i < len(A) :
3.     if A[i] == x :
4.         return i
5.     i = i + 1
6. return -1

Trace LS([2,4,6,8],4), and count the time complexity $t_{LS}([2,4,6,8],4)$

What is $t_{LS}(A,x)$, if the first index where $x$ is found is $j$?

What is $t_{LS}(A,x)$ is $x$ isn’t in $A$ at all?
worst case

denote the worst-case complexity for program $P$ with input $x \in I$,
where the input size of $x$ is $n$ as
$W_P(n) = \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\}$

The upper bound $W_P \in \mathcal{O}(U)$ means

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\} \leq cU(n)$$

That is:
$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall x \in I, \text{size}(x) \geq B \Rightarrow t_P(x) \leq cU(\text{size}(x))$$

The lower bound $W_P \in \Omega(L)$ means

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\} \geq cU(n)$$

That is:
$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \exists x \in I, \text{size}(x) = n \land t_P(x) \geq cL(n)$$
def IS(A) :
    """ IS(A) sorts the elements of A in non-decreasing order """
    i = 1
    while i < len(A) :
        t = A[i]
        j = i
        while j > 0 and A[j-1] > t :
            j = j-1
        A[j] = t
    i = i+1

I want to prove that $W_{IS} \in O(n^2)$. 
scratch