Linear Search

Input: A list $A$ of length $n$, value $x$.
Output: Index $i$ such that $A[i] = x$, or 0 if $x \notin A$.

1. $i = 1$
2. while $i \leq n$ and $x \neq A[i]$ do
3.     $i = i + 1$
4. end
5. if $i \leq n$ then
6.     return $i$
7. end
8. return 0;

What is the worst case time complexity of this algorithm?

A. $n$
B. $n\log_2 x + (2n + 2)\log_2 n + 1$
C. $3n$
D. Can’t determine
Time Complexity Discussion

Why do we almost never calculate the exact time complexity?
Time Complexity Discussion

- Hard
- Different computers have different operations
- We only use computers for large amount of data. We don’t usually care if it is 10000000 operations or 10000001 operations.
Big-O

1. We only care about large input sizes
2. We only care about scaling, not the details.

What specific aspect of the definition of big-O notation captures each of these ideas.
Big-O

**Input**: \( n \in \mathbb{N} \)

1. **while** \( 0 \leq n \leq 100 \) **do**
2. \( n = n - 1; \)
3. **end**
4. **print** "All Done";

1. What is the smallest big-O bound on the time complexity of this algorithm? \( O(1) \) or \( O(n) \)? Find \( k \) and \( C \) to back up your claim.
2. Prove \( 2x^2 + 10 \neq O(x) \). (What proof technique?)