Linear Search

**ALGORITHM 2  The Linear Search Algorithm.**

```plaintext
procedure linear search(x: integer, a₁, a₂, …, aₙ:  distinct integers)
i := 1
while (i ≤ n and x ≠ aᵢ)
    i := i + 1
if i ≤ n then location := i
else location := 0
return location{location is the subscript of the term that equals x, or is 0 if x is not found}
```

Searching Algorithms

The problem of locating an element in an ordered list occurs in many contexts. For instance, a program that checks the spelling of words searches for them in a dictionary, which is just an ordered list of words. Problems of this kind are called searching problems. We will discuss several algorithms for searching in this section. We will study the number of steps used by each of these algorithms in Section 3.3.

The general searching problem can be described as follows: Locate an element $x$ in a list of distinct elements $a₁, a₂, …, aₙ$, or determine that it is not in the list. The solution to this search problem is the location of the term in the list that equals $x$ (that is, $i$ is the solution if $x = aᵢ$) and is 0 if $x$ is not in the list.

**THE LINEAR SEARCH**

The first algorithm that we will present is called the linear search, or sequential search, algorithm. The linear search algorithm begins by comparing $x$ and $a₁$. When $x = a₁$, the solution is the location of $a₁$, namely, 1. When $x \neq a₁$, compare $x$ with $a₂$. If $x = a₂$, the solution is the location of $a₂$, namely, 2. When $x \neq a₂$, compare $x$ with $a₃$. Continue this process, comparing $x$ successively with each term of the list until a match is found, where the solution is the location of that term, unless no match occurs. If the entire list has been searched without locating $x$, the solution is 0. The pseudocode for the linear search algorithm is displayed as Algorithm 2.

**THE BINARY SEARCH**

We will now consider another searching algorithm. This algorithm can be used when the list has terms occurring in order of increasing size (for instance: if the terms are numbers, they are listed from smallest to largest; if they are words, they are listed in lexicographic, or alphabetic, order). This second searching algorithm is called the binary search algorithm. It proceeds by comparing the element to be located to the middle term of the list. The list is then split into two smaller sublists of the same size, or where one of these smaller lists has one fewer term than the other. The search continues by restricting the search to the appropriate sublist based on the comparison of the element to be located and the middle term. In Section 3.3, it will be shown that the binary search algorithm is much more efficient than the linear search algorithm. Example 3 demonstrates how a binary search works.
Insertion Sort

procedure insertion sort\(a_1, a_2, \ldots, a_n:\) real numbers with \(n \geq 2\)

for \(j := 2\) to \(n\)

\(i := 1\)

while \(a_j > a_i\)

\(i := i + 1\)

\(m := a_j\)

for \(k := 0\) to \(j - i - 1\)

\(a_{j-k} := a_{j-k-1}\)

\(a_i := m\)

\{\(a_1, \ldots, a_n\) is in increasing order\}
What is the runtime?

• For i=1 to n
  – For j=1 to i
    • For k=1 to j
      – Print(“Hello!”)
    • For r=1 to i
      – Print(“Good Bye!”)