# Sequential Search Algorithm:

Given *list*, a list of values, and *target*, a target value that we are looking for

Set *found* to **false**

Make the first item in *list* the current item

While there are items left in *list* and *found* is **false**

If the current item is equal to *target*

Set *found* to **true**

Else

Move to the next item in *list* as the current item

If *found* is **true**

Print “Target was found”

Else

Print “Target was not found in the list”

* Note that additional processing could also be done when the item was found, but this is not important to our analysis.
* Note that the version of the algorithm above is a general algorithm. The version of the algorithm below assumes that the list is implemented with a zero-based array. The analysis of the efficiency of the algorithm is the same in either case.

Given *list*, a list of values, and *target*, a target value that we are looking for

Set *found* to **false**

Set *currentItem* to 0

While *currentItem* < length of *list* and *found* is **false**

If *list*[*currentItem*] is equal to *target*

Set *found* to **true**

Else

Add 1 to *currentItem*

If *found* is **true**

Print “Target was found”

Else

Print “Target was not found in the list”

# Binary Search Algorithm:

Given *list*, an array of values of size *n* that is sorted in ascending order

and *target*, a target value that we are looking for

Set *lowerLimit* to 0

Set *upperLimit* to *n* – 1

Set *found* to **false**

While *lowerLimit* ≤ *upperLimit* and *found* is **false**

Set *test* to (*lowerLimit* + *upperLimit*) / 2 [using integer division]

If *list*[*test*] is equal to *target*

Set *found* to **true**

Else if *list*[*test*] > *target*

Set *upperLimit* to *test* – 1

Else

Set *lowerLimit* to *test* + 1

If *found* is **true**

Print “Target was found”

Else

Print “Target was not found in the list”

* Note that the version of the algorithm above assumes that the list is implemented with a zero-based array.