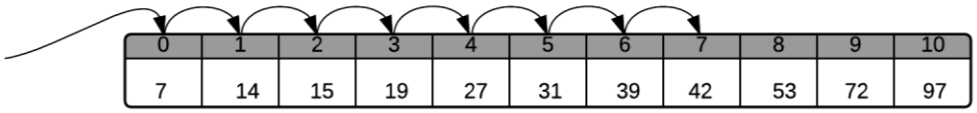
Linear Searches

We often need to search through data stored in computers. In order to do this, the easiest algorithm to understand is a **linear search**. In this search we simply **compare each item one by one** to see if it matches what we are searching for. For example, if we had an array storing numbers and we wanted to search to find out if the number 42 was in the array called *arrayOfNumbers* we could use the following algorithm:




```
FOR i ← 0 TO 10
   IF arrayOfNumbers[i] = 42 THEN
   OUTPUT "Item found"
   STOP
   ENDIF
ENDFOR
```

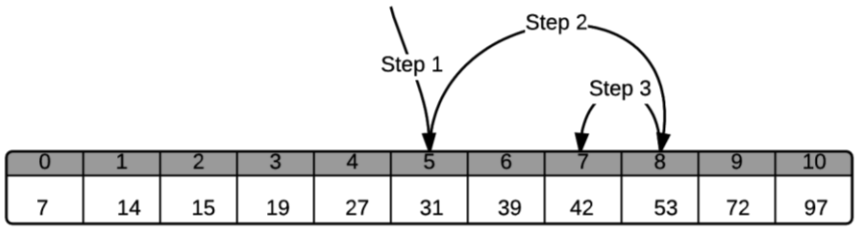
The algorithm will go through each of the items in the array one by one until it reaches listOfNumbers[7] at which point it finds the value 42 and can print "Item found".

This algorithm is easy to understand and simple to program. It also has the advantage that it works even if the array isn't sorted like the one shown above. However, it has one big problem – **it isn't very efficient**. On average we will need to search half the array to find an item. If we wanted to search an English dictionary to check the spelling of a word, we might need to make tens of thousands of comparisons to check if the word is in the dictionary using a linear search.

Binary Searches

A **binary search** is called "binary" because it splits the array into two halves as part of the algorithm. When using a binary search the array must be sorted.

In a binary search we first find the **middle item** and compare it to the value we are searching for. If it doesn't match, then if the item we are looking for is less than the one found we perform a binary search on the left of the list; otherwise we perform a binary search on the right of the list. This **function** is **recursive** as it calls itself.




```
SUBROUTINE binarySearch(Item, List)
   midpoint = List.length / 2
   IF List[midpoint] = Item THEN
      OUTPUT "Item found"
      STOP
      ELSE
         IF Item < List[midpoint] THEN
            binarySearch(Item, RightHandList)
         ELSE
            binarySearch(Item, LeftHandList)
         ENDIF
   ENDIF
ENDSUBROUTINE
```

If we wanted to search for the number 42 the algorithm would work like this:

Step 1: Search halfway through the list and find 31
Step 2: As 42 is greater than 31 do a binary search on the right hand side of the list and find 53
Step 3: As 42 is less than 53 do a binary search on the left hand side of this sublist and find 42

Although a binary search is a little harder to program, it is far more efficient. A binary search through 11 numbers would need a maximum of 4 comparisons, whereas with a linear search it would need up to 11 comparisons.

We can search a dictionary of over 100,000 words with a maximum of 17 comparisons by using a binary search.

Licensed to knole Academy, #10193259