Searching & Sorting in Java – Sequential Search

A sequential search is a systematic technique where you begin your search at the beginning (of the array), and stop your search once you reach the desired value.

For example, the following code performs a sequential search on an array of `String` values, looking for the value matching `item`. If the search is successful, it returns the location of `item` in the array `list`. If the search fails, the method returns `-1`.

```
public static int seqSearch ( String[] list, String item)
{
  int location = -1;
  for (int i = 0; i < list.length; i++)
  {
    if (list[i].equals(item))
      location = i;
  }
  return location;
}
```

Note that the variable `location` is initialized to reflect an unsuccessful search. If item is not found, the method will return this default value to indicate a failed search.

Although this method works, it is not as efficient as we could make it. Even if it finds the search value, it continues through the entire array. It would be more efficient to stop the search once we had found the search value. We can accomplish this improvement through the use of a `boolean` variable `found`.

```
public static int seqSearch ( String[] list, String item)
{
  int location = -1;
  boolean found = false;
  for (int i = 0; i < list.length && !found; i++)
  {
    if (list[i].equals(item))
    {
      location = i;
      found = true;
    }
  }
  return location;
}
```

Exercises

1. Suppose that the first example of seqSearch was used to search an array. What would the method return if item appeared in the list more than once?

2. What changed should be made to the sequential search in the second example so that it searches an array of values starting at the end and moving to the beginning?

3. A modification of the basic sequential search operates in the following way: If the item being sought is found, it is interchanged with the item that preceded it. If, for example, we were searching for 7 in the list

   3 9 5 7 2 8 4

   then, after finding 7, the list would be rearranged in the order

   3 9 **7** **5** 2 8 4

   (a) Write a method that implements this technique to search an array of int values.

   (b) Test your method in a program that first asks for the length of the list to be searched and then reads that many integers into an array. The program should then repeatedly prompt the user for values until the user supplies a sentinel of zero. The program should print the initial list and then, for each non-zero value read, it should use your modified sequential search to try to locate the item and print the resulting array.

   (c) Why might this modification sometimes improve the efficiency of a sequential search?