Binary Search Algorithm

What is a search algorithm;

In computer science, a **search algorithm** is an algorithm for finding an item with specified properties among a collection of items. The items may be stored individually as records in a database; or may be elements of a search space defined by a mathematical formula or procedure, such as the roots of an equation with integer variables or a combination of the two, such as the Hamiltonian circuits of a graph.

Περιεχόμενα:

Binary Search Algorithm

Binary search relies on a divide and conquer strategy to find a value within an already-sorted array.

The algorithm is deceptively simple.

This project has been funded with support from the European Commission.
This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

How it operates?

Algorithm is quite simple. It can be done either recursively or iteratively:

1. get the middle element;
2. if the middle element equals to the searched value, the algorithm stops;
3. otherwise, two cases are possible:
   - ***searched value is less, than the middle element***. In this case, go to the step 1 for the part of the array, before middle element.
   - ***searched value is greater, than the middle element***. In this case, go to the step 1 for the part of the array, after middle element.

Now we should define, when iterations should stop. First case is when searched element is found. Second one is when subarray has no elements. In this case, we can conclude, that searched value doesn't present in the array.

Example

Noticing Figure 1, there is a sorted array that contains 15 elements. The searched value is the number 38. Initially, the algorithm gets the “middle” element that contains the number 27. We notice that the searched value (number 38) is greater than the middle element (number 27). So, we exclude the part of the array that is before the middle element (number 27). Continuing, we examine the part of the array after the middle element. Now, we get the new middle element that contains the number 40. We notice that the searched value (number 38) is less than the middle element (number 40). So in this step, we exclude the part of the array that is after the middle element (number 40). Now, we have the part of the array that contains only the 3 elements (numbers 35,37,38). Now, we get the new middle element that contains the number 37.

This project has been funded with support from the European Commission.
This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The middle element is (number 37) is less than the searched value (number 38). So we exclude the part of the array that is before the middle element (number 37). Finally, the only element that remains contains the number 38, that is the searched value.

Now suppose that the searched value is the number 39 (see Figure 2). If we apply the same algorithm and follow the same process, the algorithm will fail to find the searched value (number 39).

How this happens?

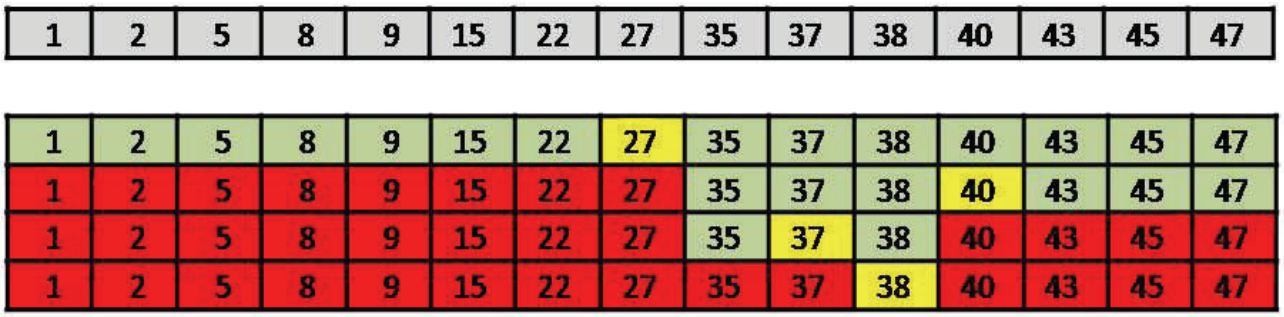

Figure 1

Examining element («middle element»)

The part of the array that will be examined

The part of the array that has been excluded

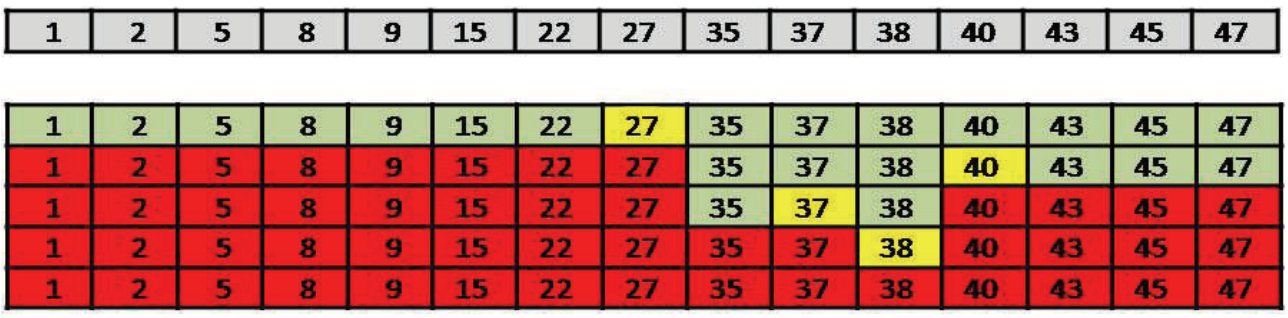

Figure 2

This project has been funded with support from the European Commission.
This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.