Swipe for more →

# Binary Search Algorithm

🔗 https://en.wikipedia.org/wiki/Binary_search_algorithm

This algorithm is used to search through iterables
 * Significantly faster than linear search.
 * Has a time complexity of O(log n).

Swipe for more →

## Advantage:

Imagine an array of numbers as given below.

| | | | | |
|---|---|---|---|---|
| 3 | 7 | 8 | 13 | 16 |

To search for an element in this list, say 13.

We would have to make 4 comparisons
searching through whole array from start.
This is called linear search.

While it is possible with only 2 comparisons
with binary search!
and it gets better with more number of elements.

## Disadvantage:

Binary search algorithm expects Sorted input.

Swipe for more →

## Basic Working Principle.

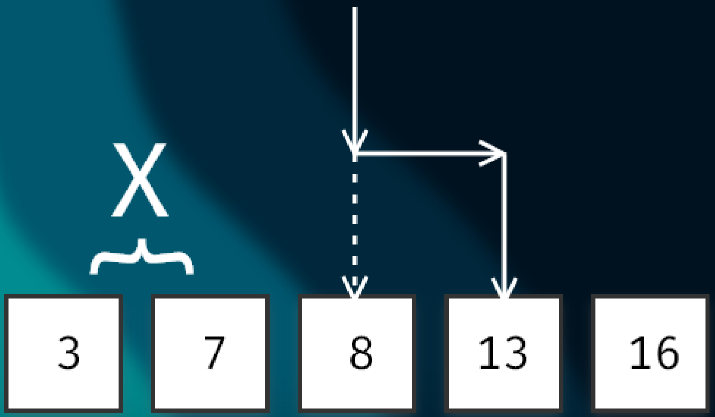
Taking the same array as before

| | | | | |
|---|---|---|---|---|
| 3 | 7 | 8 | 13 | 16 |

To Find 13 in this array, using binary search:

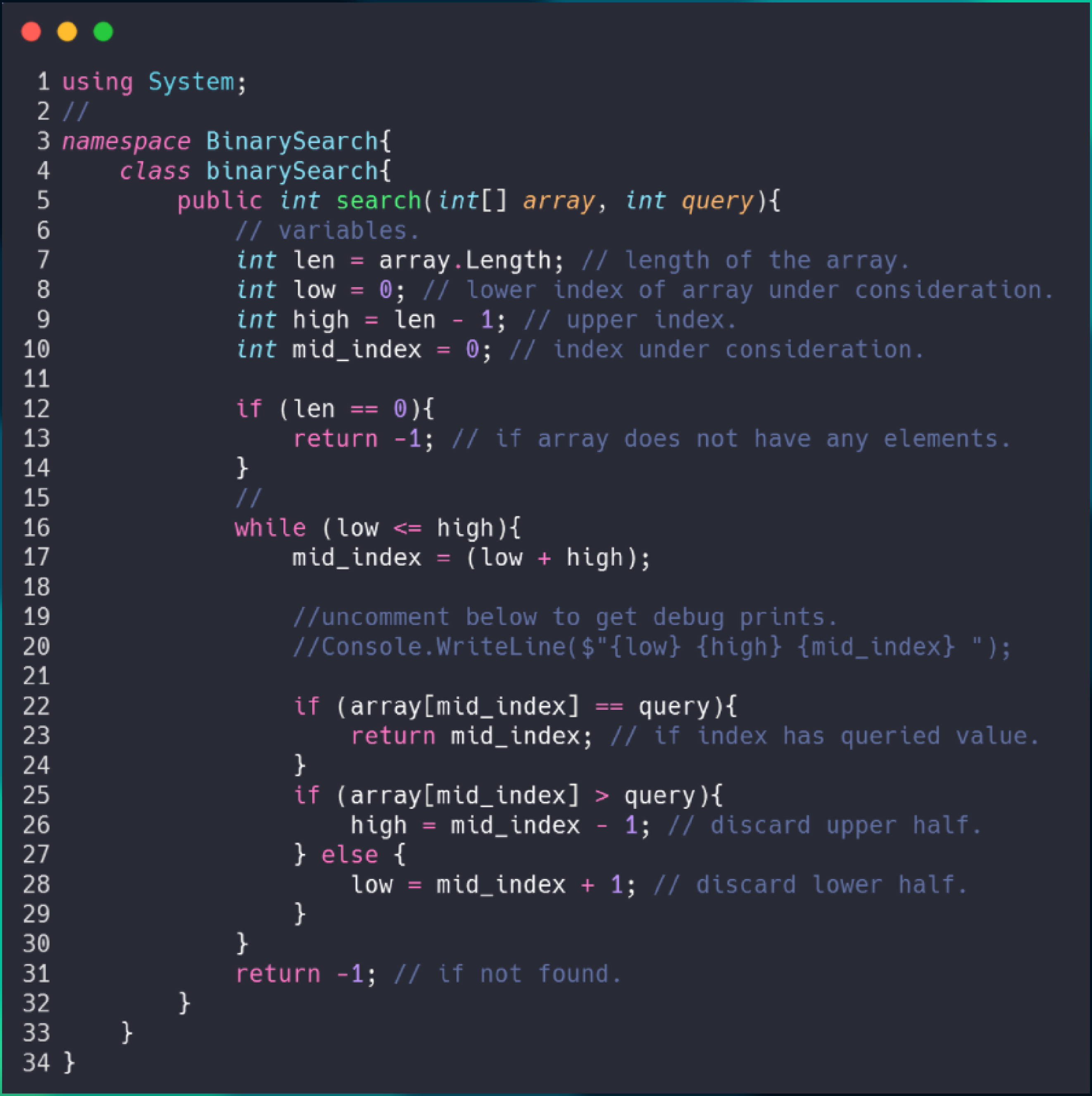
We first look at the middle element
* if it is the element we return the index of it.
* if it is lesser, we discard lower half of array.
  as there is no possibility of finding it there.
* if it is greater, we discard upper half of array.

And we repeat this procedure untill we find it.



Swipe for more →

# Implementation in C#



```csharp
using System;
//
namespace BinarySearch{
    class binarySearch{
        public int search(int[] array, int query){
            // variables.
            int len = array.Length; // length of the array.
            int low = 0; // lower index of array under consideration.
            int high = len - 1; // upper index.
            int mid_index = 0; // index under consideration.

            if (len == 0){
                return -1; // if array does not have any elements.
            }
            //
            while (low <= high){
                mid_index = (low + high);

                //uncomment below to get debug prints.
                //Console.WriteLine($"{low} {high} {mid_index} ");

                if (array[mid_index] == query){
                    return mid_index; // if index has queried value.
                }
                if (array[mid_index] > query){
                    high = mid_index - 1; // discard upper half.
                } else {
                    low = mid_index + 1; // discard lower half.
                }
            }
            return -1; // if not found.
        }
    }
}
```

Thank you for reading.

Resources and C# Implementation
On Github.

https://github.com/CoderTatva-2006/Algorithms

Follow on Instagram and Github to stay updated.

@codertatva

https://github.com/CoderTatva-2006