es

Complexity Analysis of Binary Search

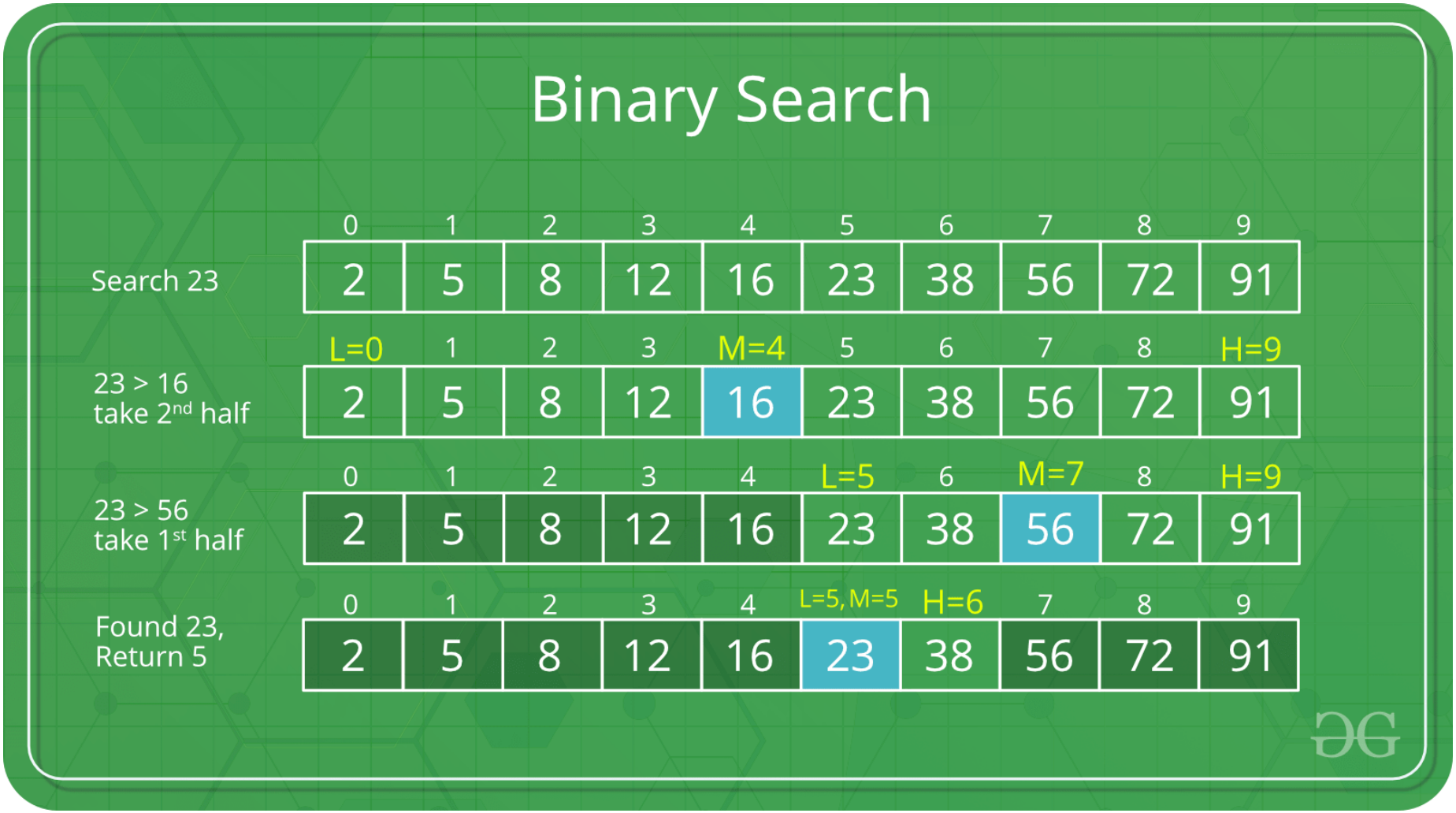
Complexities like **O(1)** and **O(n)** are simple t nstant time to perform operations like to reach an element in constant time as in case of dictio value of n to perform operations such as searching an element in an array of n elements.

in

But for **O(Log n)**, it is not that simple. Let us discuss this with the help of Binary Search Algorithm whose complexity is **O(log n)**.

Binary Search: Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Example:




```
Sorted Array of 10 elements: 2, 5, 8, 12, 16, 23, 38, 56, 72, 91

Let us say we want to search for 23.
```

Finding the given element:

Now to find 23, there will be many iterations with each having steps as mentioned in the figure above:

- **Iteration 1:**

```
Array: 2, 5, 8, 12, 16, 23, 38, 56, 72, 91
```

- Select the middle element. (**here 16**)
- Since 23 is greater than 16, so we divide the array into two halves and consider the sub-array after element 16.
- Now this subarray with the elements after 16 will be taken into next iteration.

- **Iteration 2:**

```
Array: 23, 38, 56, 72, 91
```

- Select the middle element. (**now 56**)
- Since 23 is smaller than 56, so we divide the array into two halves and consider the sub-array before element 56.
- Now this subarray with the elements before 56 will be taken into next iteration.

- **Iteration 3:**

```
Array: 23, 38
```

- Select the middle element. (**now 23**)
- Since 23 is the middle element. So the iterations will now stop.

Calculating Time complexity:

- Let say the iteration in Binary Search terminates after **k** iterations. In the above example, it terminates after 3 iterations, so **here k = 3**
- At each iteration, the array is divided by half. So let's say the length of array at any iteration is **n**
- At **Iteration 1**,

```
Length of array = n
```

- At **Iteration 2**,

```
Length of array = n/2
```

- At **Iteration 3**,

```
Length of array = (n/2)/2 = n/2^2
```

- Therefore, after **Iteration k**,

Length of array = $n/2^k$

- Also, we know that after

  After k divisions, the **length of array becomes 1**

- Therefore

  Length of array = $n/2^k = 1$
  => $n = 2^k$

- Applying log function on both sides:

  => $\log_2 (n) = \log_2 (2^k)$
  => $\log_2 (n) = k \log_2 (2)$

- As **($\log_a (a) = 1$)**
  Therefore,

  => $k = \log_2 (n)$

Hence, the time complexity of Binary Search is

$\log_2 (n)$

Recommended Posts:

Complexity of different operations in Binary tree, Binary Search Tree and AVL tree

Practice Questions on Time Complexity Analysis

Time Complexity Analysis | Tower Of Hanoi (Recursion)

Meta Binary Search | One-Sided Binary Search

Why is Binary Search preferred over Ternary Search?

Interpolation search vs Binary search

Linear Search vs Binary Search

Analysis of Algorithm | Set 5 (Amortized Analysis Introduction)

Binary Search

Variants of Binary Search

Binary Search using pthread

Binary Search In JavaScript

Uniform Binary Search

The Ubiquitous Binary Search | Set 1

Analysis of Algorithms | Set 1 (Asymptotic Analysis)

ravikishor

Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Improved By : ddg2112

Article Tags : Algorithms Analysis Divide and Conquer Mathematical Searching

Practice Tags : Searching Mathematical Divide and Conquer Algorithms

7

1

To-do Done

Based on **1** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos