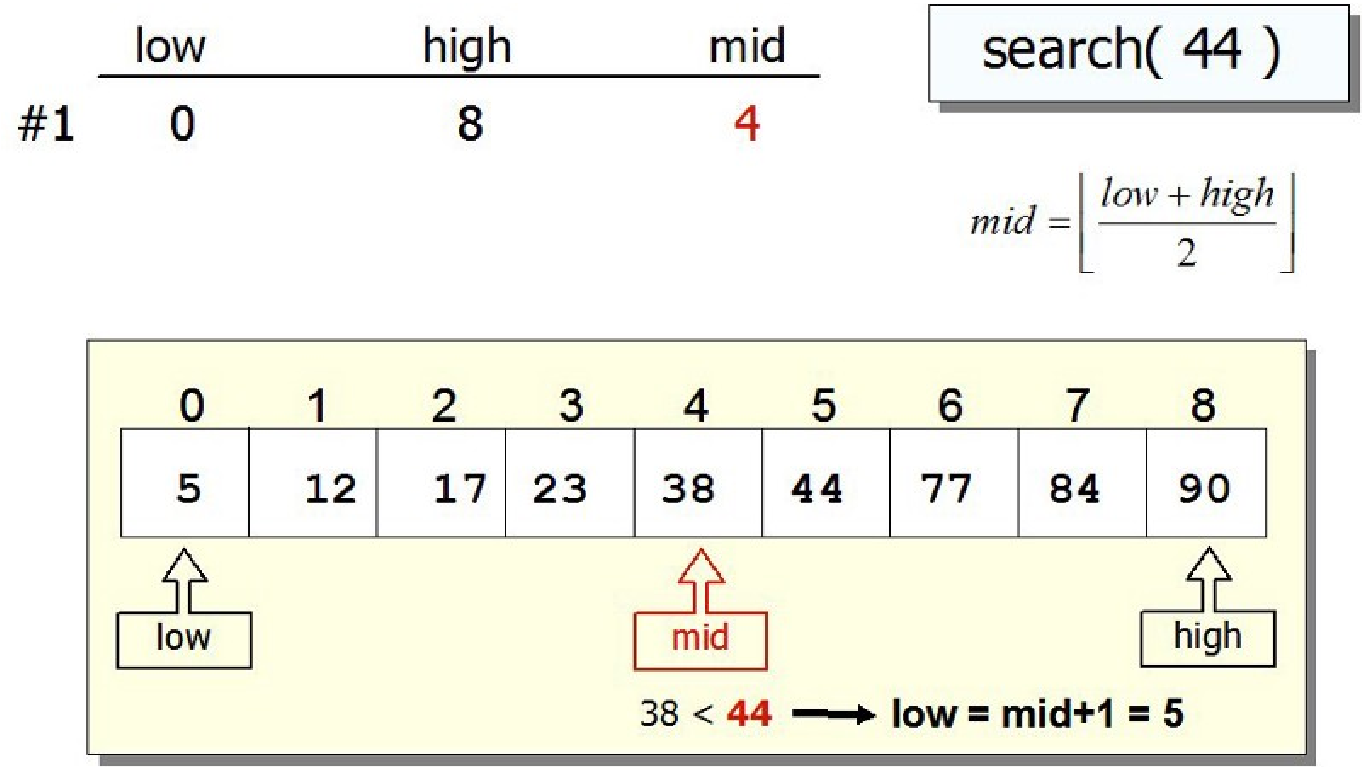
21 JANUARY 2019 / #PROGRAMMING

How to implement a Binary Search Algorithm in Java without recursion



by javinpaul

An Iterative implementation of the popular binary search algorithm to find an element in a sorted array.

structure articles on my blog, but this one is the first one here. In this article, we'll examine popular fundamental algorithms for interviews.

Yes, you guessed it right: you need to implement a **binary search** in Java, and you need to write both iterative and recursive binary search algorithms.

In computer science, a binary search, or half-interval search, is a **divide and conquer algorithm** that locates the position of an item in a sorted array. Binary searching works by comparing an input value to the middle element of the array.

The comparison determines whether the element equals the input, is less than the input, or is greater than the input.

When the element being compared equals the input, the search stops and typically returns the position of the element.

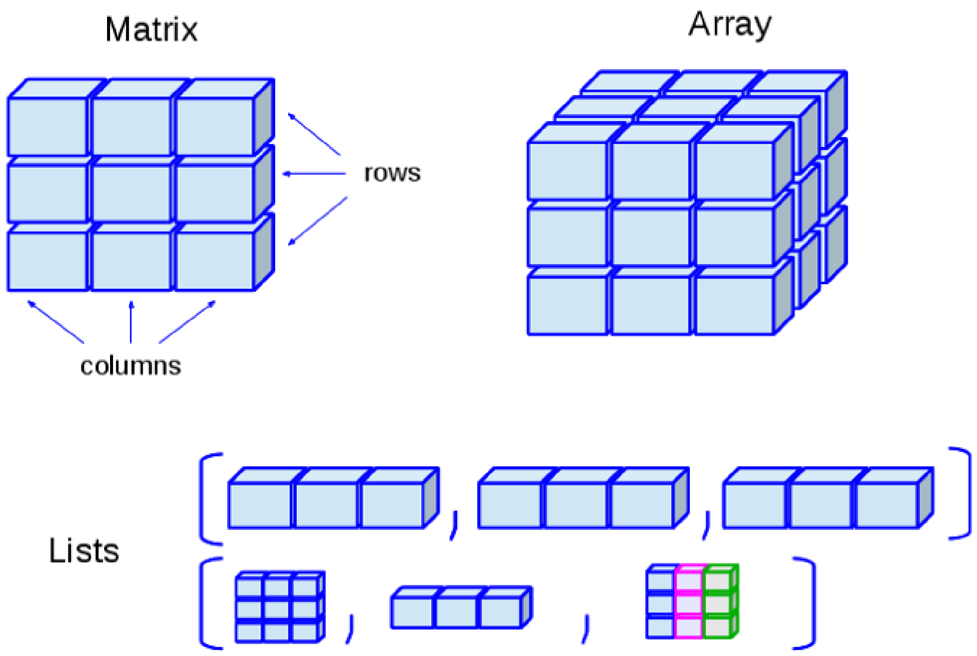
If the element is not equal to the input, then a comparison is made to determine whether the input is less than or greater than the element.

Depending on the result, the algorithm then starts over again, but only searching the top or a bottom subset of the array's elements.

If the input is not located within the array, the algorithm will usually output a unique value indicating this like -1 or just throw a RuntimeException in Java like NoValueFoundException.

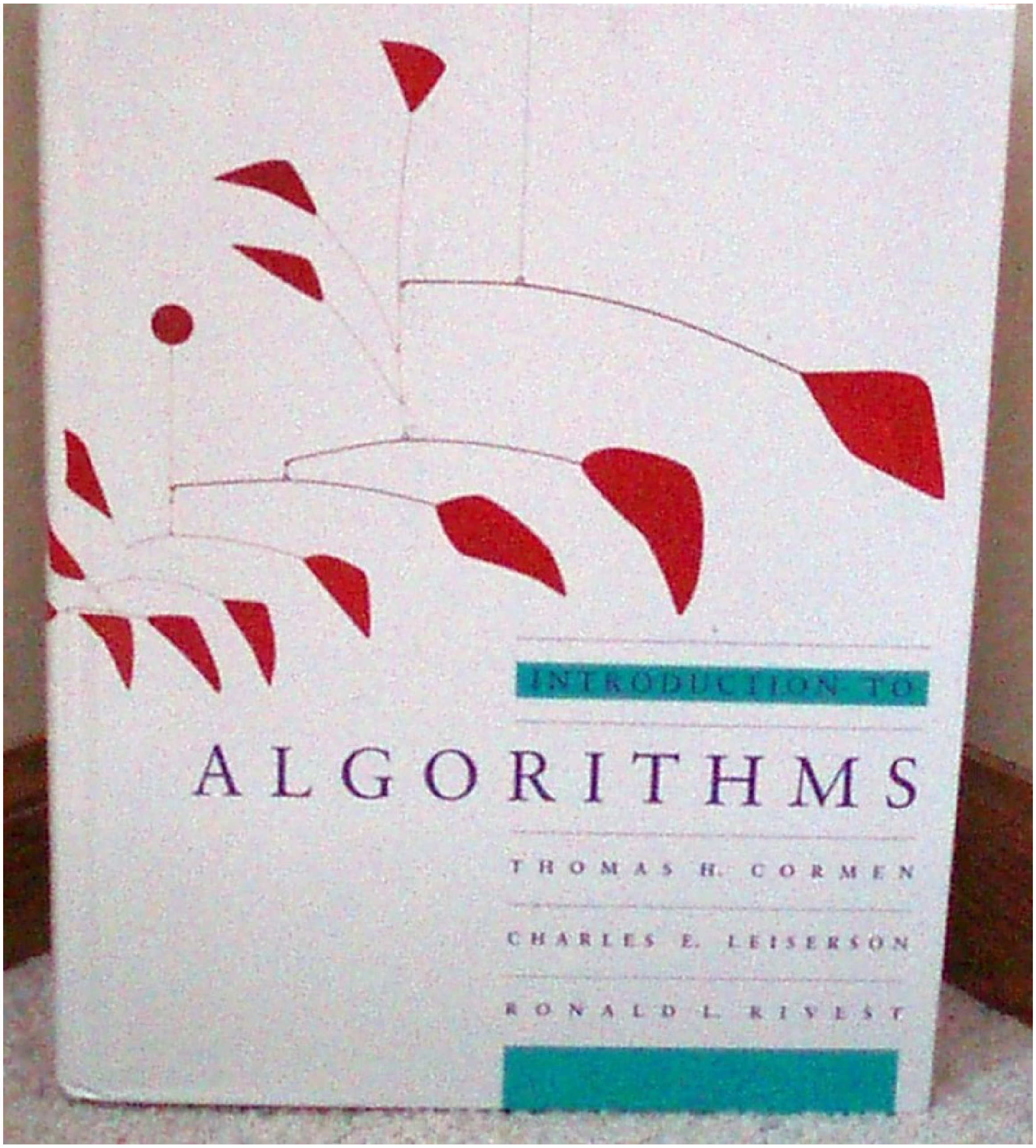
Binary search algorithms typically halve the number of items to check with each successive iteration, thus locating the given item (or determining its absence) in logarithmic time.

Btw, if you are not familiar with fundamental search and sort algorithms, then you can also join a course like **Data Structures and Algorithms: Deep Dive Using Java** to learn fundamental algorithms.

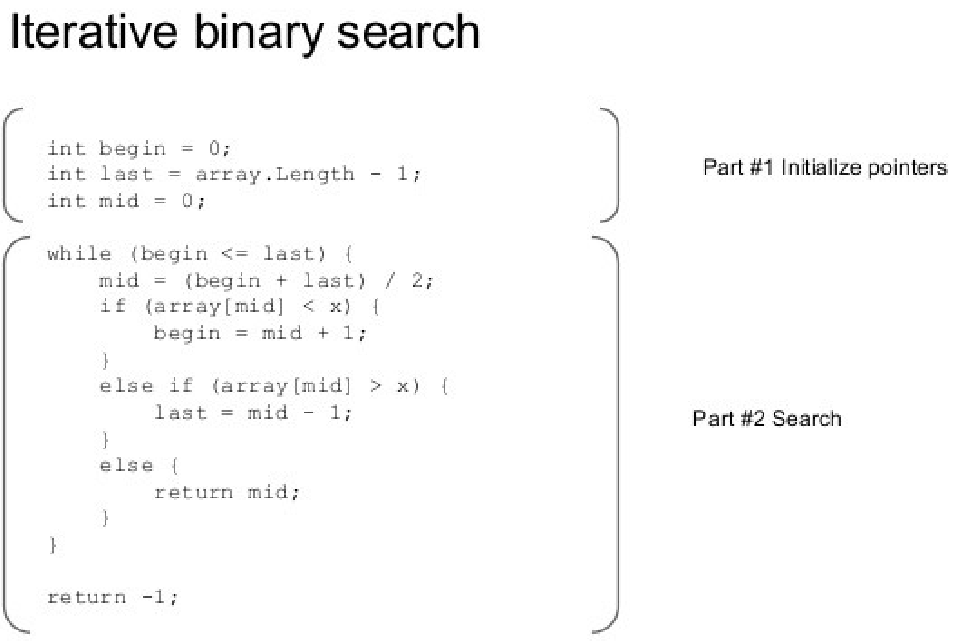


If Java is not your choice of language, you can find more recommendations for JavaScript and Python in this list of algorithms courses.

Btw, if you prefer books, I suggest you read a comprehensive algorithm book like **Introduction to Algorithms** by Thomas H. Cormen.


INTRODUCTION TO
ALGORITHMS
THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST

Here is some sample code which shows the logic of **iterative binary search in Java**:



Iterative binary search

```
int begin = 0;
int last = array.Length - 1;
int mid = 0;

while (begin <= last) {
    mid = (begin + last) / 2;
    if (array[mid] < x) {
        begin = mid + 1;
    }
    else if (array[mid] > x) {
        last = mid - 1;
    }
    else {
        return mid;
    }
}

return -1;
```

Part #1 Initialize pointers
Part #2 Search

Here is a sample program to implement binary search in Java. The algorithm is implemented recursively. Also, an interesting fact to know about binary search implementation in Java is that Joshua Bloch, author of the famous Effective Java book, wrote the binary search in “java.util.Arrays”.

```
import java.util.Arrays;import java.util.Scanner;
```

```
/** * Java program to implement Binary Search. We have implemented
```

```
public class IterativeBinarySearch {
```

```
  public static void main(String args[]) {    int[] list = new int[
```

```
    binarySearch(list, 12);    System.out.printf("Binary Search %d
```

```
    binarySearch(list, 43);    list = new int[]{123, 243, 331, 129
    binarySearch(list, 331);    System.out.printf("Binary Search %
```

```
   // Using Core Java API and Collection framework   // Preconditi
```

```
   // Search an element   int index = Arrays.binarySearch(list, 3)
```

```
  }
```

```
  /** * Perform a binary Search in Sorted Array in Java * @param inpu
```

```
  public static void binarySearch(int[] input, int number) {int firs
```

```
  while (first <= last) {  if (input[middle] < number) {  first = mi
```

```
System.out.printf(number + " found at location %d %n", middle);brea
```

```
middle = (first + last) / 2;
```

```
}
```

```
if (first > last) {  System.out.println(number + " is not present
```

```
}
```

```
}
```

```
OutputBinary Search 12 in integer array [12, 23, 31, 43]12 found a
```

That’s all about **how to implement binary search using recursion in**

algorithms you learn in your computer science class.

The binary search tree data structure takes advantage of this algorithm and arranges data in a hierarchical structure so that you can search any node in O(logN) time.

Though, you must remember that in order to use binary search, you need a sorted list or array, so you also need to consider the cost of sorting when you consider using binary search algorithm in the real world.

Further Learning
Data Structures and Algorithms: Deep Dive Using Java
Algorithms and Data Structures — Part 1 and 2
Data Structures in Java 9 by Heinz Kabutz
10 Algorithms books for Interviews
10 Data Structure and Algorithm Courses for Interviews
5 Free Courses to Learn Data Structure and Algorithms

Other **Data Structure and Algorithms tutorials** you may like

- How to implement Quicksort algorithm in place in Java? (tutorial)
- How to implement Binary Search Tree in Java? (solution)
- How to implement Quicksort algorithm without recursion? (tutorial)
- How to implement Insertion sort algorithm in Java? (tutorial)
- How to implement Bubble sort algorithm in Java? (tutorial)

Comparison based sorting algorithm? (answer)

- How to implement Bucket Sort in Java? (tutorial)
- How to implement a Binary Search Algorithm without recursion in Java? (tutorial)
- 50+ Data Structure and Algorithms Courses for Programmers (questions)

Thanks for reading this article. If you like this article then please share with your friends and colleagues. If you have any suggestion or feedback then please drop a comment.

P.S. — If you are serious about improving your Algorithms skills, I think the Data Structures and Algorithms: Deep Dive Using Java is the best one to start with.

If this article was helpful, tweet it.

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos,

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Our Nonprofit

About

Alumni Network

Open Source

Shop

Support

Sponsors

Academic Honesty

Code of Conduct

Privacy Policy

Terms of Service

Copyright Policy

Trending Guides

2019 Web Developer Roadmap

Python Tutorial

CSS Flexbox Guide

JavaScript Tutorial

Python Example

HTML Tutorial

Linux Command Line Guide

JavaScript Example

Git Tutorial

React Tutorial

Java Tutorial

Linux Tutorial

CSS Tutorial

jQuery Example

SQL Tutorial

CSS Example

React Example

Angular Tutorial

Bootstrap Example

How to Set Up SSH Keys

WordPress Tutorial

PHP Example

Donate