Searching algorithms in Java

Posted on October 18, 2014 by Arulkumaran Kumaraswamipillai — No Comments ↓

This assumes that you understand "**Swapping, partitioning, and sorting algorithms in Java**".

Q1. Can you write a code to search for number 5 in 7 3 6 8 2 9 5 4?

A1. The code below uses the **linear search algorithm**. The linear search algorithm's two advantages are simplicity and the ability to search either sorted or unsorted one-dimensional arrays. Its sole disadvantage is the time spent in examining the elements. The average number of elements to examine is half the array size, and the maximum is to examine the entire array. For example, a one-dimensional array with 1 million elements requires a linear search to examine an average of 0.5 million elements and a maximum of 1 million elements. The linear search has the complexity of O(n).

```java
public class LinearSearch {

    public static boolean found(int[ ] input, in
        if(input == null || input.length == 0){
            throw new IllegalArgumentException("
        }

        for (int i = 0; i < input.length; i++) {
            if(number == i){
                return true;
            }
        }
        return false;
    }
     //...
}
```

For very large values of n, the **binary search** gives a better complexity of O(log n). It uses the divide and conquer strategy to achieve better performance for very large arrays. For example, a one-dimensional array with 1,048,576 elements requires binary search to examine a maximum of 20 elements. Binary search algorithm's two disadvantages are increased complexity and the need to presort the one dimensional array prior to searching. To search for the value of 5:

Unsorted values.

7	3	6	8	2	9	5	4

Firstly sort it, using the quick sort algorithm discussed earlier.

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9

left idx mid idx right idx

Divide the array into two halves around the center (left idx + right idx)/2 = (0 + 7)/2 = 3. In this case, the idx 3 (which has the value of 5) is the middle index. Compare the search value of 5 with the mid idx. If the search value (i.e. 5) is greater than mid idx value, then the search value may or may not be on the RHS of the mid idx. The LHS can be safely ignored. If the search value is less than the mid idx, then the search value may or may not be on the LHS of the idx. The RHS can be safely ignored. If the search value is neither less than or greater, then the search value is the mid idx. In this example, the value of 5 is in the mid idx, and hence will be found immediately. To illustrate this, search for a value of 8 as shown below.

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9
left idx			mid idx				right idx

The search value (i.e. 8) is greater than the mid idx value of 5, so set the left idx to mid idx + 1 = 4, and then recompute the new mid idx to (left idx + right idx)/2 = (4 + 7)/2 = 5

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9
				left idx	mid idx		right idx

The search value of 8 is still greater than the mid idx, hence the left idx becomes mid idx + 1 = 5 + 1 = 6, and the new mid idx becomes (left idx + right idx) / 2 = (6 + 7) / 2 = 6.

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9
						left idx mid idx	right idx

The search value of 8 is equal to the value of the mid idx, hence the value is found. Also, note that the if the value were not be found, the terminating condition is the left idx > right idx. The code snippet below demonstrates the binary search.

```
public class BinarySearch {

    public static boolean found(int[ ] input, in
        if (input == null || input.length == 0)
            throw new IllegalArgumentException("
        }

        int leftIdx = 0;
        int rightIdx = input.length - 1;
        int midIdx;

        while (leftIdx <= rightIdx) {
            midIdx = (leftIdx + rightIdx) / 2;
            if (number > input[midIdx])
                leftIdx = midIdx + 1;
            else if (number < input[midIdx])
                rightIdx = midIdx - 1;
            else
               return true;
        }
       return false;
    }

    //...
```

80+ step by step Java Tutorials

100+ Java pre-interview coding tests

How good are your .....?

```
}
```

Job Hunting & Resum

Q2. Is there a way to improve the above code snippet? COQ

A2.The above code snippet only handles int values, and the code can be made more reusable by making it handle binary search of any Comparable object as shown below:

```
public class BinarySearch2<T>{

    @SuppressWarnings("unchecked")
    public static boolean found(Comparable[ ] in
        if (input == null || input.length == 0)
            throw new IllegalArgumentException("
        }

        int leftIdx = 0;
        int rightIdx = input.length - 1;
        int midIdx;
        while (leftIdx <= rightIdx) {
            midIdx = (leftIdx + rightIdx) / 2;
            if (compare.compareTo(input[midIdx])
                leftIdx = midIdx + 1;
            else if (compare.compareTo(input[mid
                rightIdx = midIdx - 1;
            else
               return true;
        }

       return false;

    }

    // ....

}
```

The above code can be used with an Integer, BigDecimal, String, and any custom class that implements the interface **Comparable**.

Note: There are other algorithms for sorting like merge sort, heap sort, insert sort, etc. Also do your research on the interpolation search, which is a slight variation from the binary search algorithm. This algorithm is a bit more complex and efficient than the binary search algorithm.

Popular Posts

- ♦ 11 Spring boot interview questions & answers

823 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

765 views

18 Java scenarios based interview Questions and Answers

399 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts interview questions & answers

201 views

Bio | **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ Swapping, partitioning, and sorting algorithms in Java

♦ Tree traversal algorithms in Java ›

Posted in Algorithms**,** member-paid

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer