2590 users solved this topic. Latest completion was about 3 hours ago.

Theory: Algorithms in Java

12 minutes 0 / 4 problems solved

Skip this topic Start practicing

As you know, algorithms are language-independent, i.e. the same algorithm can be implemented in different languages including Kotlin, Python, C++, and so on. Since we have been learning Java, we will implement algorithms using this language and its features.

§1. Implementing algorithms in Java

The Java standard library includes many classic algorithms to solve typical problems that arise when writing programs. These algorithms are written by professionals, optimized, well-tested, and used by millions of programmers around the world. However, it is fine to understand how they work internally. That helps to analyze similar algorithms from different libraries or develop a new algorithm by customizing a well-known one to solve the specific problem. Finally, it is a good point to practice Java!

Algorithms are often represented as static methods. But this is not the only way to implement them in Java. Here are some examples:

- the class `Arrays` has algorithms for manipulating arrays such as sorting and searching;
- the class `Math` has a mathematical algorithm for calculating the square root of a number.

It is very convenient to take algorithms into separate methods and then call the methods from external code, passing the input values (arguments) and obtaining an output (result).

Note: an algorithm written in Java can take input values not only from arguments, but also from a file, a database, the Internet, as well as being able to output the result to a console, file and so on.

§2. Example

Let's consider a simple algorithm written in Java and then a typical problem arising when implementing algorithms.

The following algorithm finds an index of the maximum number in an array of ints.

```
public static int findIndexOfMax(int[] numbers) {
    int index = 0;
    for (int i = 1; i < numbers.length; i++) {
        if (numbers[i] > numbers[index]) {
            index = i;
        }
    }
    return index;
}
```

This algorithm is implemented as the static method named `findIndexOfMax`. It takes an array of ints and returns the index of max. The input array can be either ordered or unordered.

Let's describe the algorithm with words to understand it better:

1. Suppose the number with the index 0 (the first element) is a candidate to be the maximum.
2. In the loop, we take the following number of the array (from the second to the last one), while the end of the array is not reached and do (3), otherwise, go to (5).
3. If the taken number is greater than the candidate, save its index (found a new candidate).
4. Go to the following iteration (2).

Current topic:

Algorithms in Java

Topic depends on:

✓ Computer algorithms

✓ Functional decomposition Stage 3

✓ Iterating over arrays

Topic is required for:

Finding max and min in arrays

Linear search in Java

Binary search in Java

Jump search in Java

Selection sort in Java

Bubble sort in Java

Insertion sort in Java

Counting sort in Java

Merge sort in Java

Quicksort in Java

Edit distance in Java

Hamming distance in Java

Searching a substring in Java

Trees in Java

Hash table in Java

Table of contents:

5. Return the index of a found candidate.

Let's call the method passing various arrays:

```
findIndexOfMax(new int[] {1, 5, 3, 2, 3}); // 1
findIndexOfMax(new int[] {6, 8, 7, 9}); // 3
findIndexOfMax(new int[] {99}); // 0, the algorithm processes single-element arrays as well
```

§3. Considering corner cases

Corner case is a situation that occurs when an algorithm's input values may lead to incorrect or ambiguous behavior. Considering corner cases is an important part of designing algorithms.

Let's return to our example algorithm. It may seem like it works well. But what if the array contains several numbers that are equal to the maximum? Also, what if the array is empty?

Actually, the presented algorithm finds the first maximum in the array.

```
findIndexOfMax(new int[] {3, 2, 8, 8, 1}); // 2
```

Sometimes we may need the first one, and sometimes the last one. It depends on the problem you are solving.

If a passed array is empty, the method returns `0` as the index of the maximum number.

```
findIndexOfMax(new int[] {}); // 0
```

It may be better to return the special value `-1` to mean that no maximum was found in the array. Let's add the special condition in the method:

```
if (numbers.length == 0) {
    return -1;
}
```

Similar problems can arise when developing *any* algorithm. Therefore, when developing algorithms, it's very important to clearly understand the expected result and what the algorithm should do in different cases and boundary conditions. And then we need to explain it to the computer.

§4. Conclusion

Being a language-independent thing, algorithms can be realized in Java. In the following topics, we will focus on implementing basic algorithms in Java. The standard library already comprises some popular algorithms written by experts and proven by the time. In order to implement a reliable algorithm, it is not enough to solve the main problem, but also define the behavior for all possible datasets and situations, called corner cases.

Report a typo

220 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (4) Hints (0) Useful links (0) Show discussion