Theory: The sorting problem

12 minutes 9 / 9 problems solved

Start practicing

2210 users solved this topic. Latest completion was 23 minutes ago.

§1. Understanding the problem

The **sorting problem** often arises in programming practice when we have to order a sequence of elements. The required order can be ascending or descending. Often, the ascending order is considered a default.

To represent sequences of elements many languages support arrays or/and lists.

Here is an array of six elements:

0	1	2	3	4	5
3	2	4	7	3	5

As a sorting result, we get another array of the same size:

0	1	2	3	4	5
2	3	3	4	5	7

Many programming languages provide built-in algorithms for sorting lists and arrays. There are many different sorting algorithms in computer science, and in this topic, we will learn some of them.

§2. What can be sorted?

It is possible to sort data of different types:

- numbers in accordance with the arithmetic order;
- Unicode characters in accordance with their order in the Unicode character table;
- strings (lexicographically or by size);
- dates and times in accordance with the chronological order.

Also, it's often possible to sort data of more complex types if we know how to compare items. As a rule, such data has one or more fields called the **sorting keys**, by which sorting is performed.

§3. Key features of sorting algorithms

- **Time efficiency.** The size of an array to sort is very important for efficiency. If we want to sort an array consisting of a few dozen elements, we can use any sorting algorithm. But what if the array contains a lot of data? In that case, we should use only the effective sorting algorithms, otherwise, the results might take too long.

- **Stability.** An array to sort may contain several identical elements. Stable sort algorithms always sort identical elements in the same order as they appear in the input. Otherwise, the sorting algorithm is unstable. The stability is important when we sort complex structures such as objects, tuples, or something else.

- **In-place/out-of-place sorting.** An algorithm performs an **in-place** sorting if it requires only a constant amount of additional space, otherwise, the algorithm performs an **out-of-place** sorting. The larger the size of the array, the more additional memory is required by the **out-of-place** algorithms.

- **Internal or external sorting.** An algorithm performs an **internal** sorting if sorting data are kept entirely within the main memory of the computer. **External** sorting is required when the data does not fit into the main

Current topic:

✓ The sorting problem

Topic depends on:

✓ The big O notation

Topic is required for:

✓ Insertion sort
✓ Merge sort
✓ Bubble sort
✓ Counting sort
✓ Quick sort
✓ Selection sort
✓ Kruskal's algorithm

Table of contents:

memory of the computing device and instead, they must be kept in the slower external memory (usually a hard drive).

We will consider sorting algorithms with different properties.

Many sorting algorithms compare array items during sorting, but some algorithms use other techniques to sort. Such algorithms are also known as **non-comparison sorting algorithms**.

§4. Conclusion

The sorting problem often arises in programming practice and is one of the most studied problems in computer science. There are a lot of different algorithms for solving the problem. Often, sorting is the basic building block for other algorithms; hence, understanding sorting is integral to solving many other problems.

Report a typo

202 users liked this theory. **2** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (4) Hints (0) Useful links (1) Show discussion