Theory: Insertion sort

12 minutes 7 / 7 problems solved

Start practicing

Insertion sort is a simple sorting algorithm that performs an in-place sorting. It divides the array into sorted and unsorted parts. Each iteration, the algorithm moves an element from the unsorted part to the sorted one until all array elements are sorted. The current element is inserted in a suitable position in the sorted part.

The algorithm works as follows:

1. Assume the first element belongs to the sorted part of the array, and all remaining elements are in the unsorted part;
2. Choose the first element from the unsorted part and insert that element into the sorted list in a suitable position;
3. Repeat steps 1-2 until all elements are sorted.

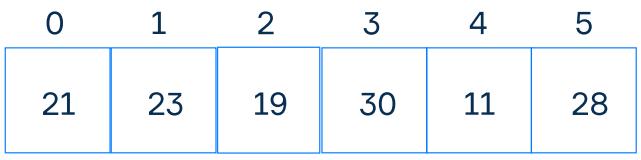
The main operation is inserting a value into the sorted part. The operation should not break the sorted part, that is, it always stays sorted. Suppose we sort an array in ascending order. The suitable position for a current element is the last index in the sorted part where the previous element is less than or equal to the current one.

The algorithm is not suitable for large arrays since its average and worst case time complexity is of $O(n^2)$, where **n** is the array length.

The algorithm is **stable**: it doesn't change the relative order of identical elements.

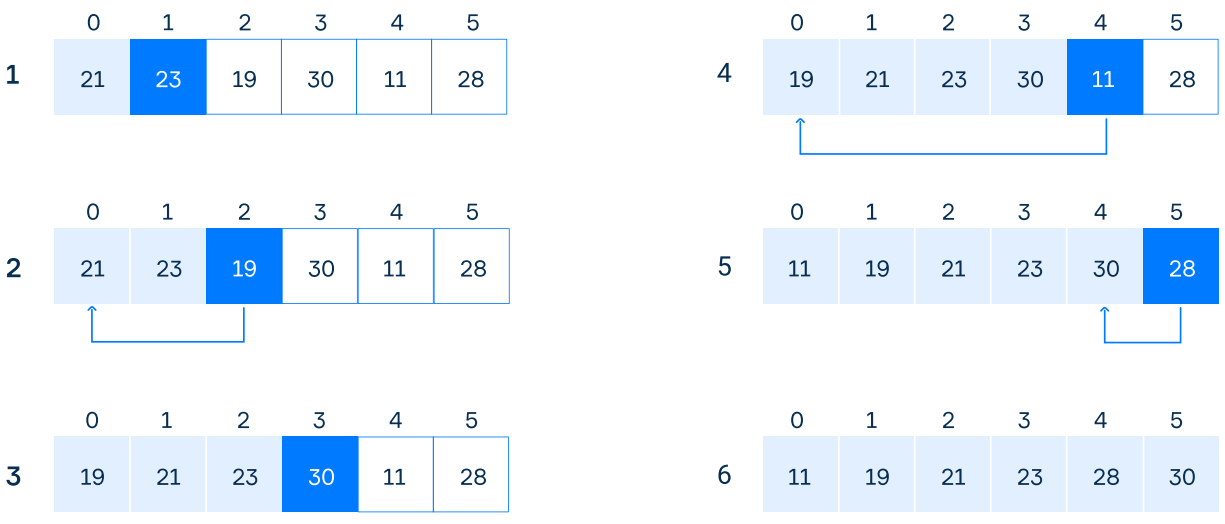
§1. Example

Suppose we have an unsorted array of integers. Our task is to sort it in ascending order.



The array has six elements, the first element has the index 0, the last one has the index 5.

The following image illustrates how the insertion sort algorithm works:



1) The sorted part includes only a single element with the index 0. It's the first element in the array. Let's consider the second one (23). It's greater than the last element in the sorted part, so we do not move it.

2) The sorted part includes elements with the indexes 0-1. We consider the element with the index 2 (19). It's less than the last element in the sorted part, so we move it to the left until it is less than the previous element. After moving it has the index 0.

3) The sorted part includes elements with the indexes 0-2. We consider the element with the index 3 (30). It's greater than the last element in the sorted part, hence we do not move it.

4) The sorted part includes elements with the indexes 0-3. We consider the element with the index 4 (11). It's less than the last element in the sorted part, so we move it to the left until it's less than the previous element. After it's moved, it has the index 0.

5) The sorted part includes elements with the indexes 0-4. We consider the element with the index 5 (28). It's less than the last element in the sorted part, and we move it to the left until it's less than the previous element. After moving it has the index 4.

6) The whole array is sorted.

Seeing a visualization of the Insertion sort algorithm may help you get the gist of it.

Report a typo

93 users liked this theory. **0** didn't like it. **What about you?**

Start practicing

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