

Sorting

Sorting is rearranging the given data in a specific format- ascending or descending. The purpose of sorting elements is to greatly improve the efficiency of searching while doing computations on the data. In Java, we can sort primitive data (using sorting algorithms) , or user-defined data (using Comparable or Comparator interfaces)

We have 2 main kinds of sorting:

1. Internal Sorting - done in main memory
2. External Sorting - algorithms that use external memory like tape, or a disk for sorting.

External sorting is used when data is too large to fit into main memory.

Bubble Sort

Technique Brute Force

How? Each element is compared with every other element, and when they are not in correct order, they are swapped

Time n^2
Complexity

Space $O(1)$ - because it is done in place
Complexity

Merge Sort

Technique Divide and Conquer

Best Used when merging 2 or more already sorted input lists

How? Dividing the input data into half recursively till each input is of size=1. Then merging the sorted data into one

Merge Sort (cont)

Time $O(n \log n)$ - $\log n$ to sort half the data in each recursion, n to merge all the input data to give the final sorted output

Space $O(n)$ extra space required
Complexity when merging back the sorted data

Merge Sort does not preserve ordering of elements of the same value

Insertion Sort

Technique

Best used for small data

Advantages Preserves insertion order of elements of same values

How? Removes an element from the input list and insert into the correct position of the already sorted list.

Time $O(n^2)$
Complexity

Space $O(1)$ - because it is done in place
Complexity

Quick Sort

Technique Divide and Conquer

How? Select a pivot element. Split the array into 2 parts - elements less than pivot and elements $>$ pivot. Recursively repeat this process till all the elements are sorted.

Time $O(n \log n)$
Complexity

Space $O(1)$ - it is done in place
Complexity

Selection Sort

Technique

Best Used only for small set of data, it doesn't scale well for larger data sets

How? Find min value in the list, swap it with element at current index. Repeat the process till the list is sorted.

Time $O(n^2)$
Complexity

Space $O(1)$ - because it is done in place
Complexity

Heap Sort

Technique Divide and Conquer

Best Used Priority Queues

How? Insert all elements into a heap (minHeap or MaxHeap). Then remove elements from the heap till the heap is empty.

Heap The main property of a heap is that it always contains the min/max element at the root. As elements are inserted and deleted, the heap makes use of the "heapify" property to ensure that the min/max element is always at the root. So, always the min/max element is returned when the heap is dequeued

Time $O(n \log n)$
Complexity

Space $O(n)$



By evanescen09

Published 17th August, 2019.

Last updated 17th August, 2019.

Page 1 of 1.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>