quick sort

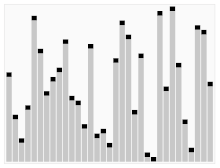
# library link

x

# basic description

Quicksort is an alignment algorithm developed by Charles Anthony Richard Horr. It belongs to a comparative alignment that performs alignment only by comparison with other elements. Quick sorting performs O(n2) comparisons in the worst case and O(n) comparisons on average when sorting n data.

<https://en.wikipedia.org/wiki/Quicksort>



1. If the range has less than two elements, return immediately as there is nothing to do. Possibly for other very short lengths a special-purpose sorting method is applied and the remainder of these steps skipped.
2. Otherwise pick a value, called a *pivot*, that occurs in the range (the precise manner of choosing depends on the partition routine, and can involve randomness).
3. *Partition* the range: reorder its elements, while determining a point of division, so that all elements with values less than the pivot come before the division, while all elements with values greater than the pivot come after it; elements that are equal to the pivot can go either way. Since at least one instance of the pivot is present, most partition routines ensure that the value that ends up at the point of division is equal to the pivot, and is now in its final position (but termination of quicksort does not depend on this, as long as sub-ranges strictly smaller than the original are produced).
4. [Recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)) apply the quicksort to the sub-range up to the point of division and to the sub-range after it, possibly excluding from both ranges the element equal to the pivot at the point of division. (If the partition produces a possibly larger sub-range near the boundary where all elements are known to be equal to the pivot, these can be excluded as well.)

# dataset

* Random data generation.

# code description

* We sort random numbers from 0 to 99 with quick\_sort1 implementing the quick sort algorithm and quick\_sort2 utilizing Python's strengths.