Quick sort

↳ Quick Sort is also an important sorting technique used to sort the given data.

↳ Quick sort technique based on divide and conquer strategy

Divide: Partition the array A[i--j] into subarrays A[i--q-1] and A[q+1--j] such that each element of A[i--q-1] < A[q] and each element of A[q+1--j] > A[q], where A[q] is the pivot element. [Generally we take first or last element of the array as a pivot element].

Conquer: Sort the two subarrays A[i--q-1] and A[q+1--j] by recursive call to quicksort.

Combine: Since the subarrays are sorted in place, no work is needed to combine them: the entire array A[i--j] is now sorted.

↳ $(\log_2 n)$ Pass are required to sort n elements

↳ Best case Time complexity : $O(n \log_2 n)$

↳ worst case Time complexity : $O(n^2)$

↳ Average case Time complexity : $O(n \log_2 n)$.

Rough Idea.

I/P: | 8 | 10 | 6 | 12 | 7 | 16 | 11 |

Let's take first element of the array as the Pivot element

Pass 1: [7 6] (8) [10 12 16 11] ~~one element sorted.~~

Pass 2 [6] (7) (8) (10) [12 16 11] element 7, 8 and 10 sorted

Pass 3: 6 7 8 10 [11] (12) [16] – sorted

Algorithm: Quick sort

```
Quicksort (A, P, r) ,   P is a initial index of Array and
{                        r is a last index of Array.
  if (P<r)
  {
    q = Partition (A, P, r);
    quicksort (A, P, q-1);
    quick sort (A, q+1, r);
  }
}.

Partition (A, P, r)
{
  x = A[r]  (Pivot element).
  i = P-1
    for (j = p to r-1)
    {
      if (A[j] ≤ x)
      {
          i = i+1
      }   }  Swap A[i] with A[j]
    Swap A[i+1] with A[r]
    return i+1
}
```