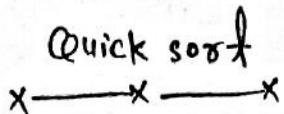


(1)



- ↳ 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 Ideq.

(2)

I/P: 

8	10	6	12	8	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 16 | 7 8 10 | 12 16 11 element 7, 8 and 10 sorted

Pass 3: 6 7 8 10 | 12 | 16 - sorted

(3)

Algorithm:

Quick sort

$\text{Quicksort}(A, P, \tau)$ ,  $P$  is a initial index of Array and  $\tau$  is a last index of Array.

{ if ( $P < \tau$ )

{      $q = \text{Partition}(A, P, \tau);$

$\text{quicksort}(A, P, q-1);$

$\text{quicksort}(A, q+1, \tau);$

}

 $\text{Partition}(A, P, \tau)$ 

{

$x = A[\tau]$  (Pivot element).

$i = P - 1$ ,

for ( $j = P$  to  $\tau - 1$ )

{

    if ( $A[j] \leq x$ )

$i = i + 1$

} } Swap  $A[i]$  with  $A[j]$

Swap  $A[i+1]$  with  $A[\tau]$ return  $i + 1$ 

{