



Dash



All



Articles



Videos



Problems



Quiz

&lt;&lt; Prev

Next &gt;&gt;

# Quick Sort using Hoare Partition

## Hoare's Partition Scheme:

Hoare's Partition Scheme works by initializing two indexes that start at two ends, the two indexes move toward each other until an inversion is (A smaller value on the left side and greater value on the right side) found. When an inversion is found, two values are swapped and the process is repeated.

## Algorithm:

```
partition(arr[], lo, hi)
    pivot = arr[lo]
    i = lo - 1 // Initialize left index
    j = hi + 1 // Initialize right index

    // Find a value in left side greater
    // than pivot
    do
        i = i + 1
        while arr[i] < pivot

    // Find a value in right side smaller
    // than pivot
    do
        j--;
        while (arr[j] > pivot);
```



Dash



All



Articles



Videos



Problems



Quiz



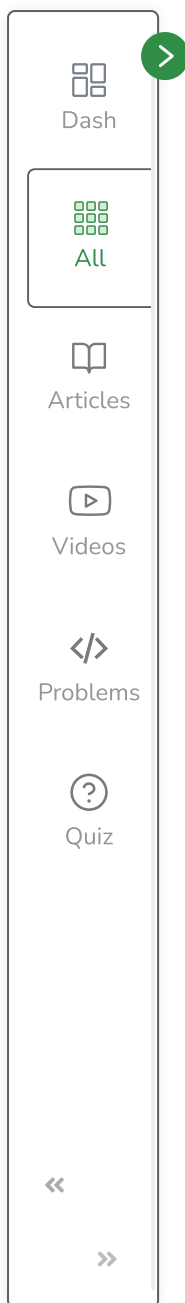
```
if i >= j then  
    return j
```

```
swap arr[i] with arr[j]
```

Below are implementations of this approach:-

**C++**

```
#include <bits/stdc++.h>  
using namespace std;  
  
/* This function takes first element as pivot, and places  
all the elements smaller than the pivot on the left side  
and all the elements greater than the pivot on  
the right side. It returns the index of the last element  
on the smaller side*/  
int partition(int arr[], int low, int high)  
{  
    int pivot = arr[low];  
    int i = low - 1, j = high + 1;  
  
    while (1) {  
        // Find leftmost element greater than  
        // or equal to pivot  
        do {  
            i++;  
        } while (arr[i] < pivot);
```



```
// Find rightmost element smaller than
// or equal to pivot
do {
    j--;
} while (arr[j] > pivot);

// If two pointers met.
if (i >= j)
    return j;

int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
}

/* The main function that implements QuickSort
arr[] --> Array to be sorted,
low --> Starting index,
high --> Ending index */
void quickSort(int arr[], int low, int high)
{
    if (low < high) {
        /* pi is partitioning index, arr[p] is now
        at right place */
        int pi = partition(arr, low, high);

        // Separately sort elements before
```





Dash



All



Articles



Videos



Problems



Quiz



```
// partition and after partition
quickSort(arr, low, pi);
quickSort(arr, pi + 1, high);
}
}

/* Function to print an array */
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout<<arr[i]<<" ";
    cout<<"\n";
}

// Driver Code
int main()
{
    int arr[] = { 10, 7, 8, 9, 1, 5 };
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout<<"Sorted array: "<<"\n";
    printArray(arr, n);
    return 0;
}
```

**Output**

Courses

Tutorials

Jobs

Practice

Contests



Dash



All



Articles



Videos



Problems



Quiz

**Time Complexity:  $O(N)$** **Auxiliary Space:  $O(1)$** 

**Note :** If we change Hoare's partition to pick the last element as pivot, then the Hoare's partition may cause QuickSort to go into an infinite recursion. For example, {10, 5, 6, 20} and pivot is arr[high], then returned index will always be high and call to same QuickSort will be made. To handle a random pivot, we can always swap that random element with the first element and simply follow the above algorithm.

**Comparison:**

1. Hoare's scheme is more efficient than Lomuto's partition scheme because it does three times fewer swaps on average, and it creates efficient partitions even when all values are equal.
2. Like Lomuto's partition scheme, Hoare partitioning also causes Quick sort to degrade to  $O(n^2)$  when the input array is already sorted, it also doesn't produce a stable sort.
3. Note that in this scheme, the pivot's final location is not necessarily at the index that was returned, and the next two segments that the main algorithm recurs on are (lo..p) and (p+1..hi) as opposed to (lo..p-1) and (p+1..hi) as in Lomuto's scheme.
4. Both Hoare's Partition, as well as Lomuto's partition, are unstable.

**Hoare partition algorithm**

Generally, the first item or the element is assumed to be the initial pivot element. Some choose the middle element and even the last element.

It is a linear algorithm.

**Lomuto partition algorithm**

Generally, a random element of the array is located and picked and then exchanged with the first or the last element to give initial pivot values. In the aforementioned algorithm, the last element of the list is considered as the initial pivot element.

It is also a linear algorithm.

**Hoare partition algorithm****Lomuto partition algorithm**


It is relatively faster.	It is slower.
It is slightly difficult to understand and to implement.	It is easy to understand and easy to implement.
It doesn't fix the pivot element in the correct position.	It fixes the pivot element in the correct position.





Marked as Read


 Report An Issue


If you are facing any issue on this page. Please let us know.


 Dash

 All

 Articles

 Videos

 Problems

 Quiz

