

Data Structures

" Data Structures are widely used to organize data into unique structures to enhance programs performance. "

8+1

[Suggest \(http://www.studytonight.com/suggest\)](http://www.studytonight.com/suggest)

(http://www.addthis.com/bookmark.php?v=300&winname=addthis&publish=US&s=linkedin&url=http%3A%2F%2Fwww.studytonight.com%2Fdata-sort&title=Quick%20Sort%20in%20Data%20Structures&ate=AT-ra-4fcb1/53eefbeb107d7373/2&frommenu=1&uid=53eefbeb532a43e0&ct=1&prestructures%2Fselection-sorting&tt=0&captcha_provider=nucaptcha)

[Home \(http://www.studytonight.com/\)](http://www.studytonight.com/)

[Core Java \(http://www.studytonight.com/java\)](http://www.studytonight.com/java)

[C++ \(http://www.studytonight.com/cpp\)](http://www.studytonight.com/cpp)

[C Language \(http://www.studytonight.com/c\)](http://www.studytonight.com/c)

[DBMS \(http://www.studytonight.com/dbms\)](http://www.studytonight.com/dbms)

[More... \(http://www.studytonight.com/library\)](http://www.studytonight.com/library)

Like

2

16

Quick Sort Algorithm

Quick Sort, as the name suggests, sorts any list very quickly. Quick sort is not stable search, but it is very fast and requires very less additional space. It is based on the rule of **Divide and Conquer**(also called *partition-exchange sort*). This algorithm divides the list into three main parts :

1. Elements less than the Pivot element
2. Pivot element
3. Elements greater than the pivot element

In the list of elements, mentioned in below example, we have taken **25** as pivot. So after the first pass, the list will be changed like this.

6 8 17 14 25 63 37 52

Hence after the first pass, pivot will be set at its position, with all the elements smaller to it on its left and all the elements larger than it on the right. Now 6 8 17 14 and 63 37 52 are considered as two separate lists, and same logic is applied on them, and we keep doing this until the complete list is sorted.

How Quick Sorting Works

✓ Basics and Sorting

- ➔ Introduction to Data Structures (introduction-to-data-structures)
- ➔ Time Complexity of Algorithms (time-complexity-of-algorithms)
- ➔ Introduction to Sorting (introduction-to-sorting)
- ➔ Bubble Sort (bubble-sort)
- ➔ Insertion Sort (insertion-sorting)
- ➔ Selection Sort (selection-sorting)
- ➔ Quick Sort (quick-sort)
- ➔ Merge Sort (merge-sort)
- ➔ Heap Sort (heap-sort)

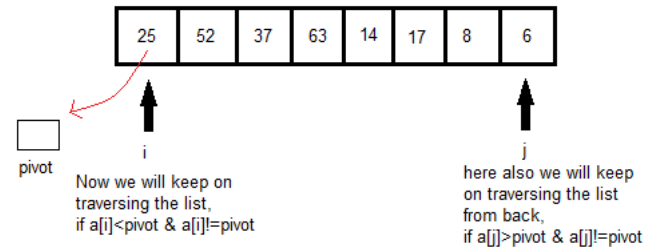
✓ Data Structures

- ➔ Stack Data Structure (stack-data-structure)
- ➔ Queue Data Structure (queue-data-structure)

Test Yourself !

If you have studied all the lessons of Data Structure, then evaluate yourself by taking these tests.

Coming Soon



if both sides we find the element
not satisfying their respective
conditions, we swap them. And
keep repeating this.

DIVIDE AND CONQUER - QUICK SORT

Sorting using Quick Sort Algorithm

```
/* a[] is the array, p is starting index, that is 0,
and r is the last index of array. */

void quicksort(int a[], int p, int r)
{
    if(p < r)
    {
        int q;
        q = partition(a, p, r);
        quicksort(a, p, q);
        quicksort(a, q+1, r);
    }
}

int partition(int a[], int p, int r)
{
    int i, j, pivot, temp;
    pivot = a[p];
    i = p;
    j = r;
    while(1)
    {
        while(a[i] < pivot && a[i] != pivot)
            i++;
        while(a[j] > pivot && a[j] != pivot)
            j--;
        if(i < j)
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
        else
        {
            return j;
        }
    }
}
```

Complexity Analysis of Quick Sort

Worst Case Time Complexity : $O(n^2)$

Best Case Time Complexity : $O(n \log n)$

Average Time Complexity : $O(n \log n)$

Space Complexity : $O(n \log n)$

- Space required by quick sort is very less, only $O(n \log n)$ additional space is required.
- Quick sort is not a stable sorting technique, so it might change the occurrence of two similar elements in the list while sorting.

Subjects : **Core Java** (<http://www.studytonight.com/java>) **C++** (<http://www.studytonight.com/cpp>) **C Language** (<http://www.studytonight.com/c>) **DBMS** (<http://www.studytonight.com/dbms>) **Servlet** (<http://www.studytonight.com/servlet>)

© Studytonight 2013 · Handcrafted with Love

About Us (<http://www.studytonight.com/about>) · Suggest (<http://www.studytonight.com/suggest>) · Terms (<http://www.studytonight.com/terms>) · Contact Us (<http://www.studytonight.com/contact>) · Collaborate (<http://www.studytonight.com/collaborate/>) · Blog (<http://studytonight.tumblr.com/>)