Name: Samuel Yeboah

ID: 10665039

Course: CSCD 304 (Algorithms)

Assignment NO. III

Like [Merge Sort](http://quiz.geeksforgeeks.org/merge-sort/), QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

[**Best complexity**](https://www.google.com/search?sxsrf=ALeKk01XGPb-XOwdvYSLuM64nIenvh4ZvA:1582881442255&q=quicksort+best+complexity&stick=H4sIAAAAAAAAAOPgE-LUz9U3sDRKLirR0s5OttLPLojPKdcvzi8qycxLj0_MSc8vyizJyLVKSi0uiU_Ozy3ISa3ILKlcxCpZWJqZnA1SqACSU0DIAQBVSYP_VgAAAA&sa=X&ved=2ahUKEwjM89jw9PPnAhXoShUIHcHdDSgQ6BMoADACegQICRAC&sxsrf=ALeKk01XGPb-XOwdvYSLuM64nIenvh4ZvA:1582881442255)**:**n\*log(n)

In the most balanced case, each time we perform a partition we divide the list into two nearly equal pieces. This means each recursive call processes a list of half the size. Consequently, we can make only log2 *n* nested calls before we reach a list of size 1. This means that the depth of the [call tree](https://en.wikipedia.org/wiki/Call_stack) is log2 *n*. But no two calls at the same level of the call tree process the same part of the original list; thus, each level of calls needs only *O*(*n*) time all together (each call has some constant overhead, but since there are only *O*(*n*) calls at each level, this is subsumed in the *O*(*n*) factor). The result is that the algorithm uses only *O*(*n* log *n*) time.

[**Worst complexity**](https://www.google.com/search?sxsrf=ALeKk01XGPb-XOwdvYSLuM64nIenvh4ZvA:1582881442255&q=quicksort+worst+complexity&stick=H4sIAAAAAAAAAOPgE-LUz9U3sDRKLirR0slOttLPLojPKdcvzi8qycxLj0_MSc8vyizJyLUqzy8qLolPzs8tyEmtyCypXMQqVViamZwNUqkAllRASAIAPIwfbVgAAAA&sa=X&ved=2ahUKEwjM89jw9PPnAhXoShUIHcHdDSgQ6BMoADADegQIChAC&sxsrf=ALeKk01XGPb-XOwdvYSLuM64nIenvh4ZvA:1582881442255)**:**n^2

The most unbalanced partition occurs when one of the sublists returned by the partitioning routine is of size *n* − 1 This may occur if the pivot happens to be the smallest or largest element in the list, or in some implementations (e.g., the Lomuto partition scheme as described above) when all the elements are equal.

If this happens repeatedly in every partition, then each recursive call processes a list of size one less than the previous list. Consequently, we can make *n* − 1 nested calls before we reach a list of size 1. This means that the [call tree](https://en.wikipedia.org/wiki/Call_stack) is a linear chain of *n* − 1 nested calls. The *i*th call does *O*(*n* − *i*) work to do the partition, and {\displaystyle \textstyle \sum \_{i=0}^{n}(n-i)=O(n^{2})}, so in that case Quicksort takes *O*(*n*²) time.