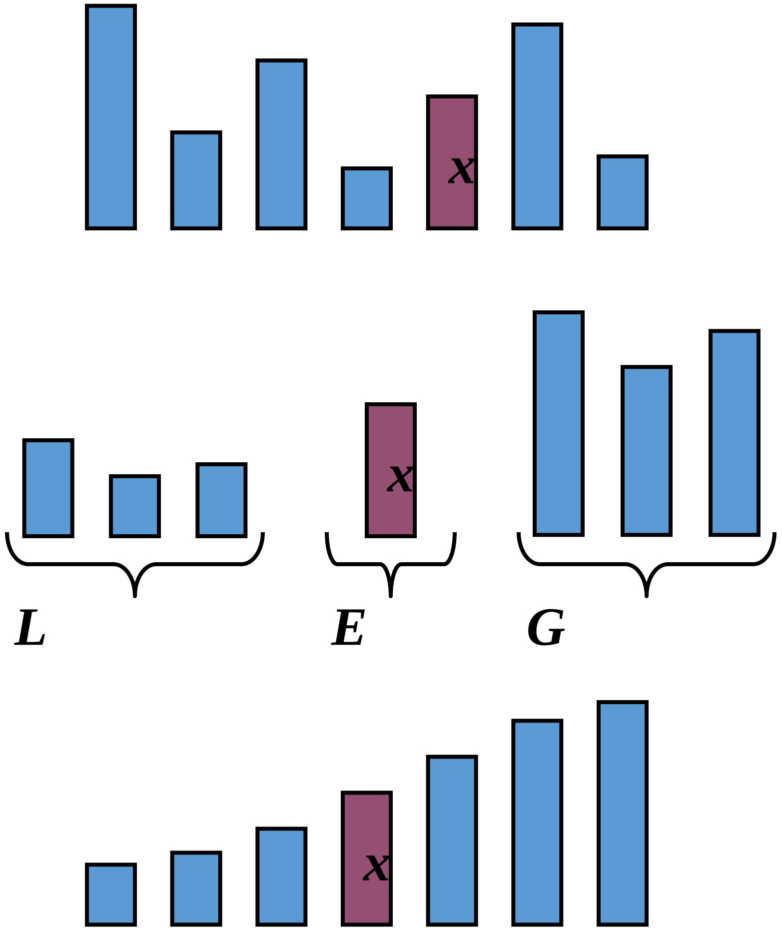
Quick Sort

Algorithm Problem Solving (APS)
17ECSE309

By,
Abhay K
01FE15BEC001

Quick-Sort

- Quick-sort is a randomized sorting algorithm based on the divide-and-conquer paradigm:
  - Divide: pick a random element x (called pivot) and partition S into
    - L elements less than x
    - E elements equal x
    - G elements greater than x
  - Recur: sort L and G
  - Conquer: join L, E and G



Partition

- We partition an input sequence as follows:
  - We remove, in turn, each element y from S and
  - We insert y into L, E or G, depending on the result of the comparison with the pivot x
- Each insertion and removal is at the beginning or at the end of a sequence, and hence takes $O(1)$ time
- Thus, the partition step of quick-sort takes $O(n)$ time

Algorithm $partition(S, p)$
    Input sequence S, position p of pivot
    Output subsequences L, E, G of the elements of S less than, equal to, or greater than the pivot, resp.
    $L, E, G \leftarrow$ empty sequences
    $x \leftarrow S.remove(p)$
    while $\neg S.isEmpty()$
        $y \leftarrow S.remove(S.first())$
        if $y < x$
            $L.insertLast(y)$
        else if $y = x$
            $E.insertLast(y)$
        else $\{ y > x \}$
            $G.insertLast(y)$
    return L, E, G

Pseudo code

Input: an array A[p, r]

```
Quicksort (A, p, r) {
  if (p < r) {
        q = Partition (A, p, r)   //q is the position of the pivot element
        Quicksort (A, p, q-1)
        Quicksort (A, q+1, r)
  }
}
```

Picking the Pivot

- Use the first element as pivot
  - if the input is random, ok
  - if the input is presorted (or in reverse order)
    - all the elements go into S2 (or S1)
    - this happens consistently throughout the recursive calls
    - Results in $O(n^2)$ behavior (Analyze this case later)
- Choose the pivot randomly
  - generally safe
  - random number generation can be expensive

Picking the Pivot

- Use the median of the array
  - Partitioning always cuts the array into roughly half
  - An optimal quicksort (O(N log N))
  - However, hard to find the exact median
    - e.g., sort an array to pick the value in the middle

Time Complexity : The complexity of quicksort in the average case is O(n*log(n)) – same as Merge sort. The problem is that in the worst case it is $O(n^2)$ – same as bubble sort.

Advantages:

- Recursive implementation is easy
- In general its speed is same as merge sort – O(n*log(n))
- Elegant solution with no tricky merging as merge sort

Disadvantages:

- As slow as bubble sort in the worst case!
- Iterative implementation isn't easy
- There are faster algorithms for some sets of data types

Applications:

- Commercial application use quicksort
- Life critical such as medical monitoring and life support in aircraft & spacecraft
- Mission critical :-
    1. Monitoring & control in industrial & research plants handling dangerous material
    2. Control for aircraft.

References :

www.cs.bu.edu/fac/gkollios/cs113/Slides/quicksort.ppt

https://www.slideshare.net/priyankanaidu6/quick-sort-2017249

homepages.math.uic.edu/~leon/cs-mcs401-s08/handouts/quicksort.pdf

https://en.wikipedia.org/wiki/Quicksort

https://www.hackerearth.com/practice/algorithms/sorting/quick-sort/tutorial/

http://www.stoimen.com/blog/2012/03/13/computer-algorithms-quicksort/