Design and Analysis of Algorithms

L12: Quicksort

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

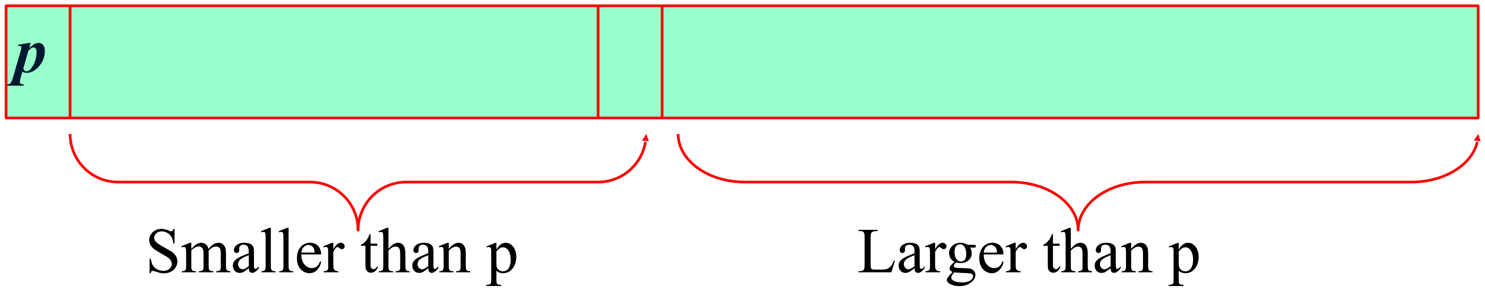
- Text book 1: Levitin (QuickSort)
-

Sort Algorithms

- Bubble sort
- Selection sort
- Insertion sort
- Mergesort
- **Quicksort**
- Shell sort
- Heap sort
- Radix sort

QuickSort

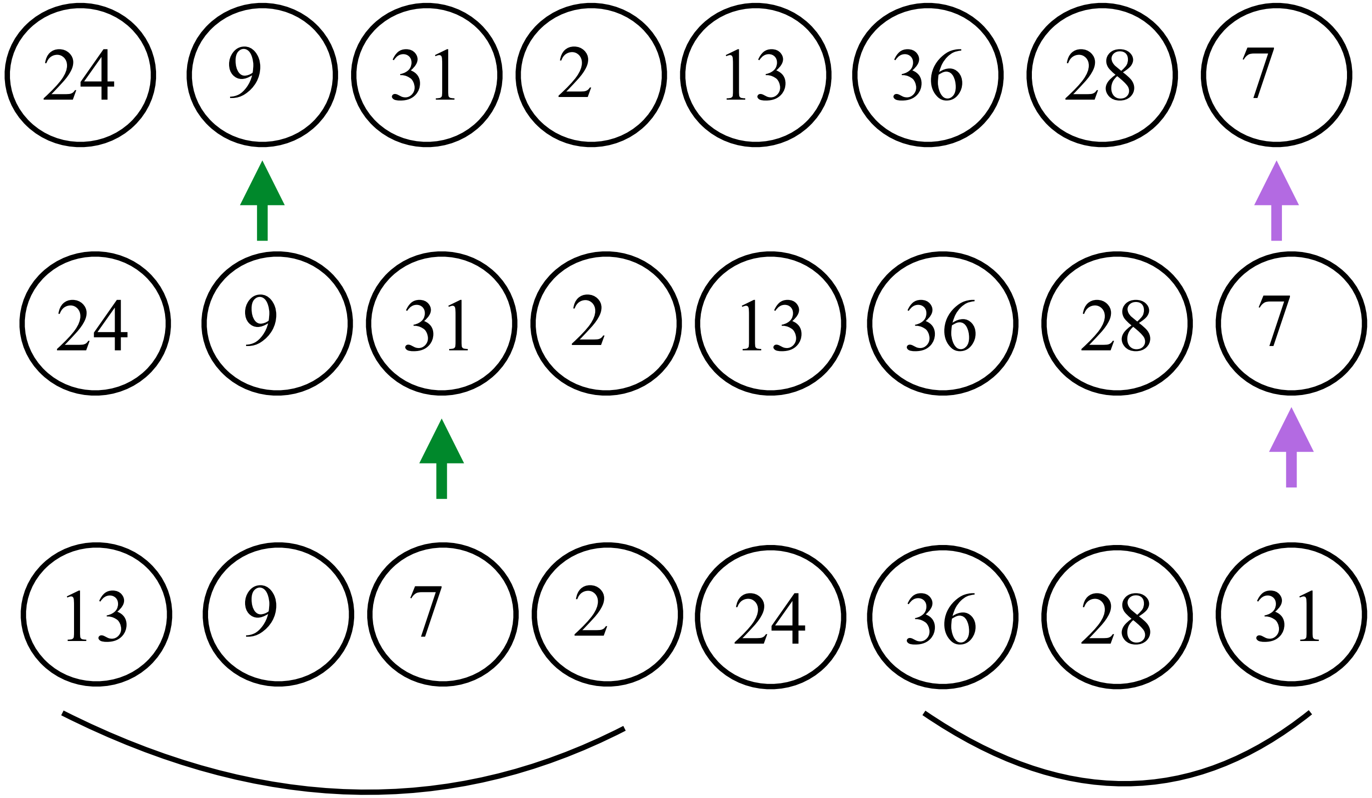
- A highly efficient algorithm
- Divides input array in smaller arrays using a specified value (pivot)
  - One array contains smaller values
  - Other array contains larger values
- Exchange the pivot with last element in first array
  - pivot is in its final position
- Sort the sub arrays recursively



QuickSort Algorithm: Steps

- Select a *pivot* (partitioning element)
  - e.g. 1st element or last element.
  - You can choose any element and swap it with last element
- Rearrange the array as follows
  - All elements in first `s-1` positions are ≤ pivot
  - All elements in remaining `n-s` positions ≥ pivot
- Repeat the process

Quicksort



24 9 31 2 13 36 28 7

24 9 31 2 13 36 28 7

13 9 7 2 24 36 28 31

Quicksort

- **Algo** `quicksort(left, right,A[])`

```
#i/p: left - array index to start from
     right - array index up to which to consider
     array[] defined by left and right indices
#o/p: array[]] sorted in ascending order

if left < right
  s ← partition(left, right, A[])
  quicksort(left, s-1, A[])
  quicksort(s+1, right, A[])
return
```

Quicksort

- Algo partition(L,R,A[])

```
p←A[R]; i←1; j←R-1
while (i <=j)
    while (A[i] <= pivot && i < R)
        i←i+1
    if (i == R) #all elements smaller than pivot
      return R
    while (j >= L) && (A[j] > pivot)
        j←j-1
    if (j < L) #all elem greater than pivot
      swap(A[L], A[r])
      return L
    if (i<j) #swap low and high elements
      swap(A[i], A[j])
swap(A[i], A[r]) #put pivot in its place
return i
```

Analysis: QuickSort

- Best case: split is approximately in the middle

  $T(n) = 2T(n/2) + \Theta(n)$

  $= \Theta(n \log_2 n)$
- Worst case: split is at the end (or beginning)
  - e.g. sorted array

  $T(n) = T(n-1) + \Theta(n)$

  $= \Theta(n^2)$
- Average case:

  $T(n) = \Theta(n \log_2 n)$
- Improvements (20-25%)
  - Better pivot selection : take median
  - Use insertion sort on smallar array size
  - Eliminate recursion and use iteration

Summary

- Mergesort
  - Not in place sort
- Quicksort
  - In place sort
  - Practically used on large data