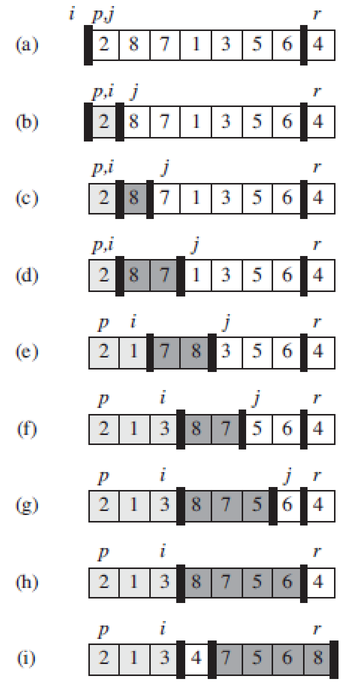
Chapter 7: Quicksort

Quicksort is an algorithm that solves the sorting problem. It uses the divide-and-conquer approach. Although the quicksort has a worst-case running time of Θ(n^2), it is commonly used in practice as it has Θ(nlgn) running time on average. Like Insertion sort and Heapsort it is an in-place algorithm. Not a *stable* algorithm.

```
QUICKSORT(A, p, r)
  if p < r
      q = PARTITION(A, p, r)
      QUICKSORT(A, p, q - 1)
      QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
  x = A[r]
  i = p - 1
  for j = p to r - 1
      if A[j] ≤ x
          i = i + 1
          exchange A[i] with A[j]
  exchange A[i + 1] with A[r]
  return i + 1
```

Example:



Performance of quicksort

The running time of quicksort depends on how the input is portioned: if the portioning is balance or not. For example, if it is well balanced (equally) then T(n)= 2T(n/2) + Θ(n).

Worst-case partitioning

The worst case occurs when the partition algorithms divides the input into two of which one has n-1 elements, and the other has 0 elements. T(n) = T(n-1) + T(0) + Θ(n).

T(n) = T(n-1) + cn =

= cn + c(n-1) + T(n-2) =

= cn + c(n-1) + c(n-2) + T(n-3)=

…

= cn + c(n-1) + c(n-2) +… +c*1 =

= c $*\sum_{i=1}^{n} i$ = c*n*(n+1)/2 = Θ(n^2)

Let's verify this by the substitution method:

T(n) = T(n-1) + cn

T(n) <= dn^2

T(n) <= d(n-1)^2 + cn = d(n^2-2n+1)+cn = dn^2-2dn+d+cn =

= dn^2 – (2dn –cn –d). We need 2dn –cn –d >=0 so that we can say T(n) <= dn^2.

2dn – cn – d >= 0 ➔ n*(2d-c) – d >= 0

For n=1, 2d - c >= d ➔ d >= c. So, we found constants d >= c > 0.

Best-case partitioning

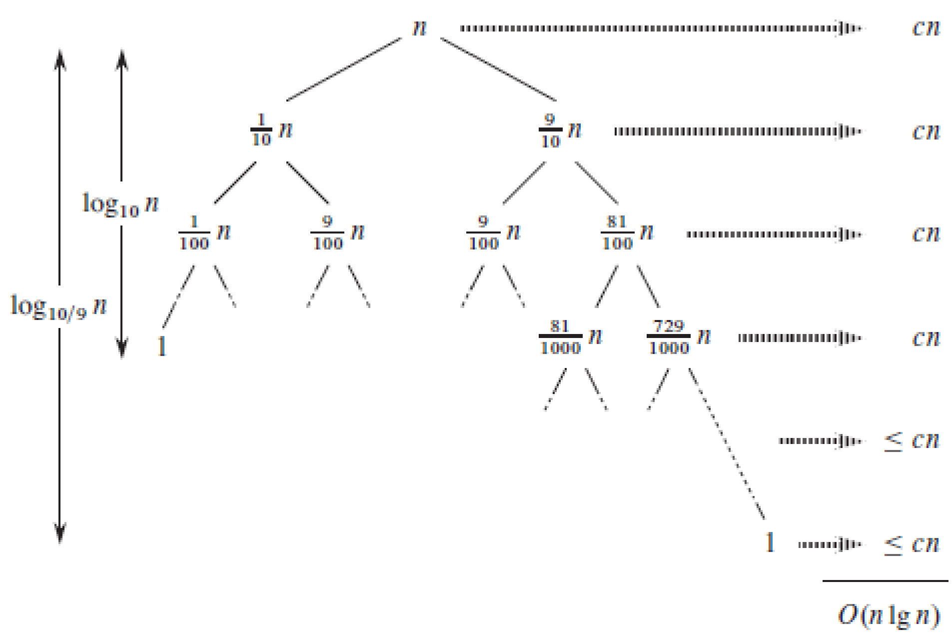
The best case occurs when the portioning is well balanced, i.e., T(n)= 2T(n/2) + Θ(n).

By the master method (case 2), T(n) = Θ(nlgn).

Average case

Closer to the best case = O(nlgn)

T(n)= T(9n/10) + T(n/10) + Θ(n).



A randomized version of quicksort

```
RANDOMIZED-PARTITION(A, p, r)
i = RANDOM(p, r)
exchange A[r] with A[i]
return PARTITION(A, p, r)
```

```
RANDOMIZED-QUICKSORT(A, p, r)
if p < r
    q = RANDOMIZED-PARTITION(A, p, r)
    RANDOMIZED-QUICKSORT(A, p, q - 1)
    RANDOMIZED-QUICKSORT(A, q + 1, r)
```