Quicksort

Algorithm

Quicksort applies divide and conquer to sort an array A [p : r]

- Divide: Partition A [p : r] into two subarrays A [p : q - 1] and A [q+1 : r] such that
  - Elements in A [p : q - 1] <= Pivot A [q]
  - Pivot A [q] <= Elements in A [q + 1 : r]
- Conquer: Call quicksort recursively to sort each subarrays A [p : q - 1] and A [q+1 : r]
- Combine by doing nothing

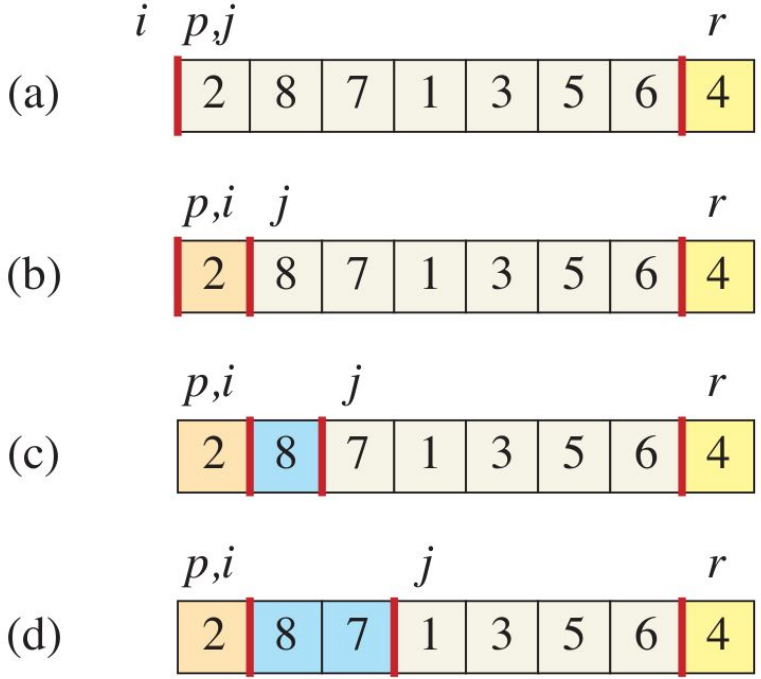
Algorithm

```
QUICKSORT(A, p, r)
1  if p < r
2        // Partition the subarray around the pivot, which ends up in A[q].
3        q = PARTITION(A, p, r)
4        QUICKSORT(A, p, q − 1)  // recursively sort the low side
5        QUICKSORT(A, q + 1, r)  // recursively sort the high side
```

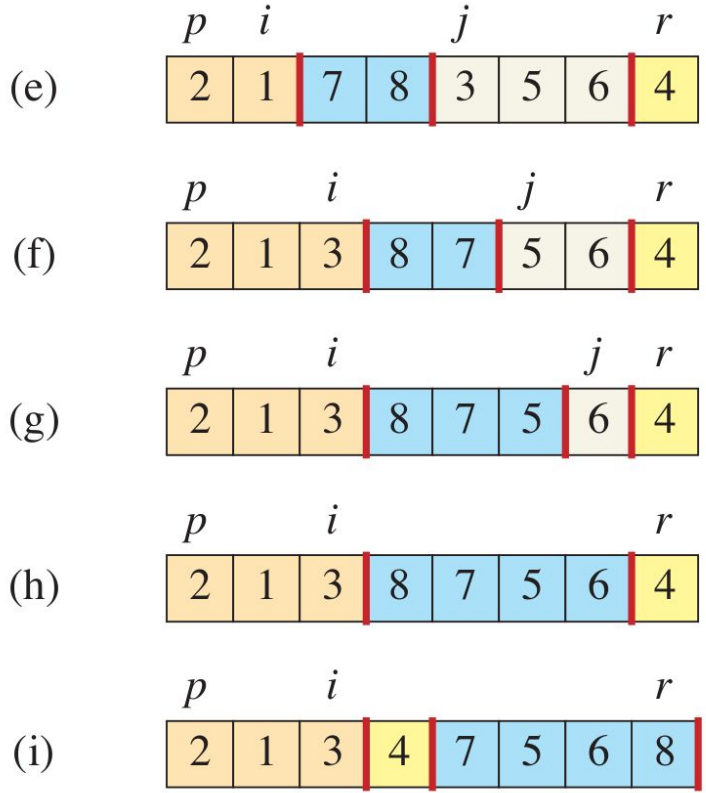
Partition

```
PARTITION(A, p, r)
1  x = A[r]                                  // the pivot
2  i = p − 1                                 // highest index into the low side
3  for j = p to r − 1                        // process each element other than the pivot
4      if A[j] ≤ x                           // does this element belong on the low side?
5          i = i + 1                         // index of a new slot in the low side
6          exchange A[i] with A[j]           // put this element there
7  exchange A[i + 1] with A[r]               // pivot goes just to the right of the low side
8  return i + 1                              // new index of the pivot
```

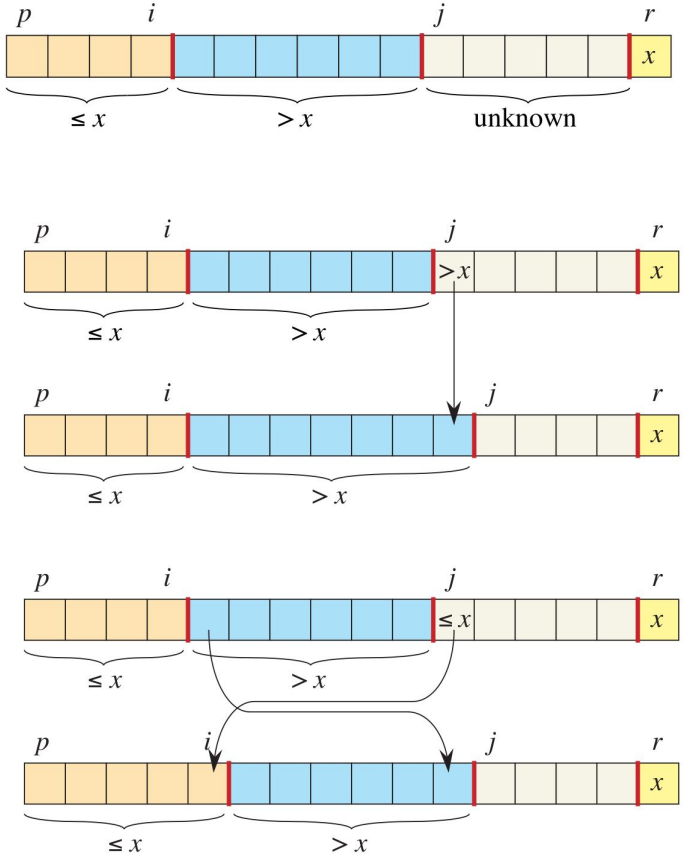
Illustration



Illustration



Illustration



Question

Illustrate the operation of partition on the array

A = [13, 19, 9, 5, 12, 8, 7, 4, 11]

Answer: [9 5 8 7 4 11 19 12 13]

Question

Q: What value of q does Partition return when all elements in the array A [p : r] have the same value?

A: r

Question

What is runtime of Partition on a subarray of size n?

A: $\Theta(n)$

Question

Modify Quicksort to sort array into monotonically decreasing order.

A: Modify Line 4 in Partition function to → “if A[j] > x”

Sample array [13 19 9 5 12 8 7 4 [11]]

Step 1 : [13 19 12] [11] [9 8 7 4 5]

Step 2 : [19 13] [12] [11] [9 8 7] [5] [4]

Performance of Quicksort: Worst case partitioning

Worst case behavior

Before partitioning [1 2 3 4 5 6 7 8]

n elements

Pivot = 8

After partitioning [1 2 3 4 5 6 7] [8] [ ]

n -1 elements 0 elements

T (n) = $\Theta(n)$ + T (n - 1) + T (0) = $\Theta(n)$ + T (n - 1) + $\Theta(1)$ = T (n-1) + $\Theta(n)$

$$T(n) = \Theta(n^2)$$

Best case partitioning

Best case:

A [1 7 3 9] [11] [13 15 14 20]

n/2 elements Pivot n/2 elements

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

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

Solution: $T(n) = \Theta(n \log n)$ as we have found in merge sort

Average case: Balanced partitioning

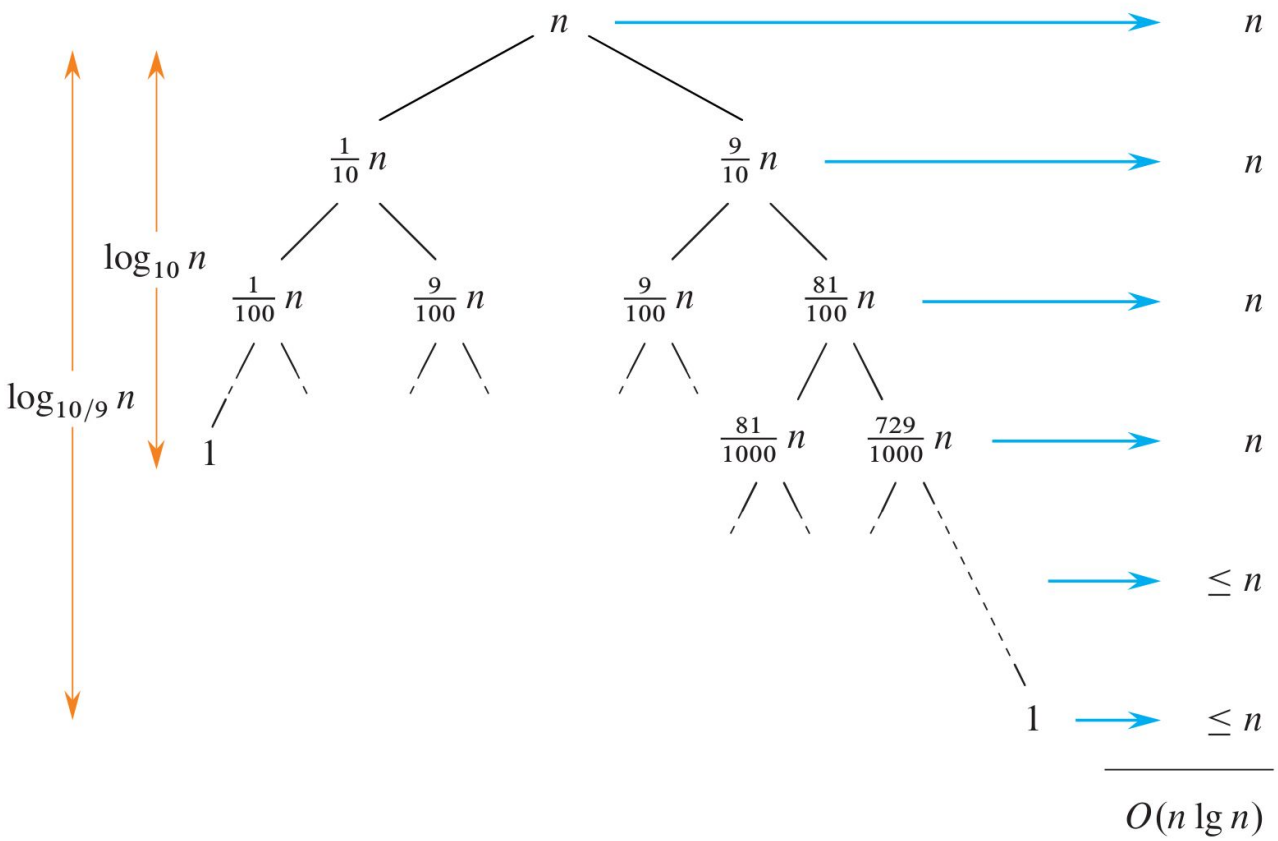
A = [13 6 7 8 21 32 77 24 10] [80] [100]

9 elements ` Pivot 1 elements

Partitioning splits the array in 9:1 ratio at each step

$$T(n) = T(9n/10) + T(n/10) + \Theta(n)$$

$$T(n) = T(9n/10) + T(n/10) + \Theta(n)$$



Each level costs at most n = $O(n)$

Recursion terminates at level $\log_{10/9} n$

Quicksort runs in O(n log n) in this case

Any split of constant proportionality yields recursion tree of depth $\Theta(\log n)$ Where each level cost $O(n)$

Run-time O(n log n)

Questions

What is the running-time of quicksort if all elements of an array A have same value?

Show that running time of quick sort is $\Theta(n^2)$ if elements are distinct and sorted in decreasing order.

Suppose that the splits at every level of quicksort are in the constant proportion α to β, where $\alpha + \beta = 1$ and $0 < \alpha \leq \beta < 1$. Show that the minimum depth of a leaf in the recursion tree is approximately $\log_{1/\alpha} n$ and that the maximum depth is approximately $\log_{1/\beta} n$. (Don't worry about integer round-off.)