# Design and Analysis of Algorithms
# Quick Sort

Dr. Lê Nguyên Khôi

The VNU University of Engineering and Technology

# Contents

# Quick Sort

- Proposed by C.A.R. Hoare in 1962
- Divide-Conquer technique
- Sorts “in place” (like insertion sort, but not like merge sort)
- Very practical (with tuning)

# Divide Conquer

## Quick sort an $n$-element array:

- Divide: Partition the array into 2 sub-arrays around a pivot element $x$ such that elements in left sub-array $\leq x$ and elements in right sub-array $\geq x$
- Conquer: Recursively sort the 2 sub-arrays
- Combine: trivial

Key: Linear-time partitioning sub-routine

# Partitioning Subroutine - Pseudocode

**Partition** $(A, p, q)$ $\Rightarrow$ $A[p, q]$

  $x \leftarrow A[p]$ $\Rightarrow$ pivot $A[p]$

  $i \leftarrow p$

  **for** $j \leftarrow p + 1$ **to** $q$ **do**

    **if** $A[j] \leq x$ **then**

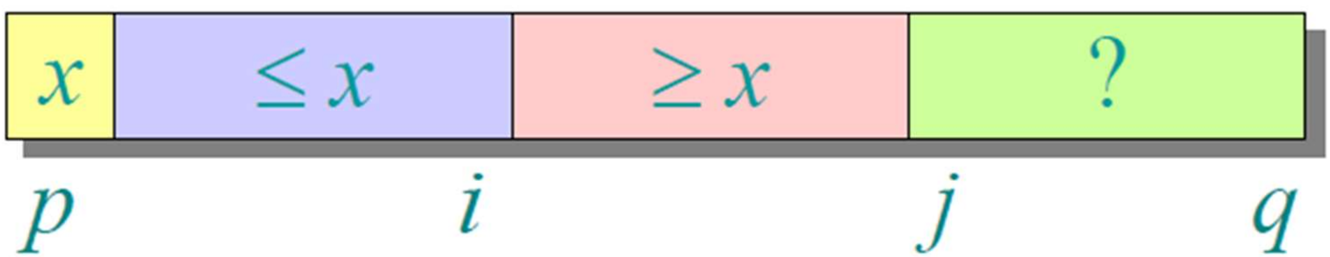
      $i \leftarrow i + 1$

      exchange $A[i] \leftrightarrow A[j]$

  exchange $A[p] \leftrightarrow A[i]$

  **return** $i$

Maintain



# Partition – Example

```
6 10 13  5  8  3  2 11    i=1
i        j                j=2->4
6  5 13 10  8  3  2 11    i=2
   i              j       j=5->6
6  5  3 10  8 13  2 11    i=3
      i              j    j=7
6  5  3  2  8 13 10 11    i=4
         i              j j=8
2  5  3  6  8 13 10 11    i=4
```

# Partition – Other Method

**`Partition`** $(A, p, q)$ $\Rightarrow$ $A[p, q]$

$x \leftarrow$ $\Rightarrow$ pivot

# Partition – Ex 7.1-1 p.173

$$A = \langle 13 , 19 , 9 , 5 , 12 , 8 , 7 , 4 , 21 , 2 , 6 , 11 \rangle$$

# Quick Sort - Pseudocode

```
QuickSort (A, p, q)
    if p < q then
        r ← Partition (A, p, q)
        QuickSort (A, p, r − 1)
        QuickSort (A, r + 1, q)
```

1st call: **QuickSort** $(A, 1, n)$

# Quick Sort - Analysis

- Assume all input elements are distinct
  - In practice, there are better partitioning algorithms for when duplicate input elements may exist
- Let $T(n) =$ worst-case running time on an array of $n$ elements

# Worst-case Analysis

- Input sorted or reverse sorted
- Partition around min or max element
- One side of partition always has no elements

$$
\begin{aligned}
T(n) &= T(0) + T(n-1) + \Theta(n) \\
&= \Theta(1) + T(n-1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&\in \Theta(n^2)
\end{aligned}
$$

# Worst-case Analysis

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

- Mathematical method
- Substitution method
- Recurrence-tree method
- Master theorem
  - In recurrence form $T(n) = aT(n/b) + f(n)$

# Best-case Analysis

**`Partition`** splits the array evenly:

$$
\begin{aligned}
T(n) &= 2T(n/2) + \Theta(n) \\
&= \Theta(n \log n) \qquad \text{(like } \mathbf{\texttt{MergeSort}}\text{)}
\end{aligned}
$$

# Other-case Analysis

**`Partition`** splits the array by the ratio of $\frac{1}{10}:\frac{9}{10}$?

$$T(n) \qquad = T\left(\frac{1}{10}n\right) + T\left(\frac{9}{10}n\right) + \Theta(n)$$

Solution to this currence $T(n) = ?$

$$cn\log_{10} n \le T(n) \le cn\log_{10/9} n + O(n)$$

# Other-case Analysis

**`Partition`** splits the array as worst-case and best-case in turn

$$L(n) = 2U(n/2) + \Theta(n) \qquad \text{best-case}$$
$$U(n) = L(n-1) + \Theta(n) \qquad \text{worst-case}$$

$$L(n) = 2\left(L\left(\frac{n}{2} - 1\right) + \Theta\left(\frac{n}{2}\right)\right) + \Theta(n)$$
$$= 2L\left(\frac{n}{2} - 1\right) + \Theta(n)$$
$$L(n) \in \Theta(n \log n)$$

# Randomized Quick Sort

Partition around a random element:

- Running time is independent of the input order
- No assumptions need to be made about the input distribution
- No specific input elicits the worst-case behavior
- The worst case is determined only by the output of a random-number generator

# Partition – More Discussion

- Assumption for analysis
  - All input elements are distinct
- If there are non-distinct elements?
- Special case: all input elements are the same
  - Worse-case running time

# Partition – More Discussion

- Partition into 3 sub-arrays
  - Left sub-array $< x$
  - Right sub-array $> x$
  - Central sub-array $= x$
- A bit faster
- Special case, all input elements are the same
  - Linear time
- Pseudocode?

# Quick Sort In Practice

- Quick sort is a great general-purpose sorting algorithm
- Quick sort is typically over twice as fast as merge sort
  - Constant $c$ in $\Theta(n)$ is quite small
- Quicksort can benefit substantially from code tuning
- Quicksort behaves well even with caching and virtual memory