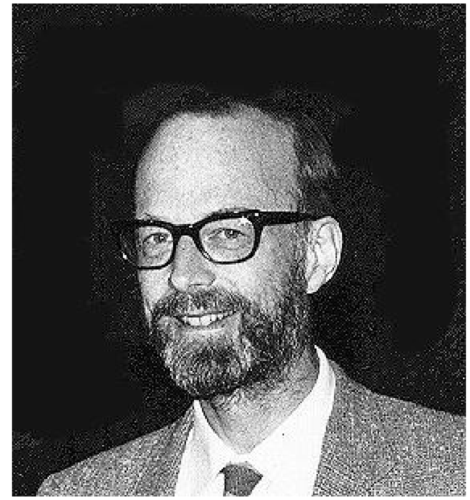
Quicksort



- Proposed by C.A.R. Hoare in 1962.
- Divide-and-conquer algorithm.
- Sorts *in situ* (in place).

  Like insertion sort, but not like merge sort.
- Very efficient in practice.
  - ▷ Few instructions in innner loop.
  - ▷ Difficult to code—best not to do it yourself.

Divide and conquer

Quicksort an n-element array:

1. **Divide:** Partition the array into two subarrays around a *pivot* x.

$\leq x$	x	$\geq x$

2. **Conquer:** Recursively sort the two subarrays.

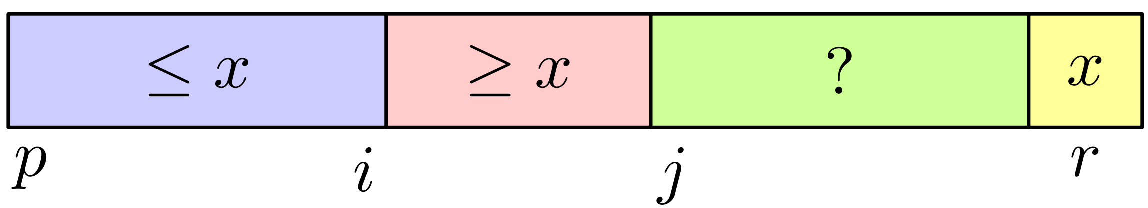
3. **Combine:** Not needed.

Key: Linear-time partitioning subroutine.

Partitioning subroutine

```
PARTITION(A, p, r)                        ▷ O(n) for n elements
x ← A[r]                                  ▷ pivot=A[r]
i ← p − 1
for j ← p to r − 1
        do if A[j] ≤ x
              then i ← i + 1
                   exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```

Invariant:



Example of partitioning

$\boxed{2}$	8	7	1	3	5	6	4
2	$\boxed{8}$	7	1	3	5	6	4
2	8	$\boxed{7}$	1	3	5	6	4
2	8	7	$\boxed{1}$	3	5	6	4
2	1	7	8	$\boxed{3}$	5	6	4
2	1	3	8	7	$\boxed{5}$	6	4
2	1	3	8	7	5	$\boxed{6}$	4
2	1	3	8	7	5	6	$\boxed{4}$
2	1	3	4	7	5	6	8

Pseudocode for Quicksort

$\textsc{Quicksort}(A, p, r)$

1. **if** $p < r$
2. **then** $q \leftarrow \textsc{Partition}(A, p, r)$
3. $\textsc{Quicksort}(A, p, q - 1)$
4. $\textsc{Quicksort}(A, q + 1, r)$

Initial call: $\textsc{Quicksort}(A, 1, n)$

Analysis of Quicksort

- Assume all input elements are distinct.
- There are better ways to partition when input has duplicates.
- Let $T(n)$ be worst-case running time on an array of n elements.

Worst-case of Quicksort

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

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

Best-case of Quicksort

- If we're lucky, PARTITION splits the array evenly.

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

- What if the split is $\frac{1}{10} : \frac{9}{10}$?

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

- The answer to this recurrence is $T(n) = O(n \lg_{\frac{10}{9}} n) = O(n \lg n)$.

- How can we make sure we are usually lucky?

Randomized Quicksort

Idea: 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 triggers the worst-case behavior.
- The worst case is determined only by the random-number generator.

Quicksort in practice

- Quicksort is a great general-purpose sorting algorithm.
- Quicksort is typically over twice as fast as merge sort.
- Quicksort can benefit substantially from *code tuning*.
- Quicksort behaves well even with caching and virtual memory.