Data Structures and Algorithms

Lecture 08

Aniket Basu Roy

BITS Pilani Goa Campus

2026-01-30 Fri

Agenda

Quicksort

- Continued from last lecture
- Proof of Correctness
- Time Complexity

Comparison-Based Sorting

- Lower Bounds

Quicksort

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

Quicksort

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

Quicksort

Proof of Correctness

Quicksort

Proof of Correctness

- Correctness of the `PARTITION` function
  - Loop Invariant
- Proof by Induction on the number of array elements

Quicksort

Proof of Correctness

- Correctness of the `PARTITION` function
  - Loop Invariant

Quicksort

Proof of Correctness

- Correctness of the `PARTITION` function
  - Loop Invariant
    - Initialization
    - Maintenance
    - Termination

Quicksort

Define

- Let $x = A[r]$
- $LEFT := A[p : i]$
- $RIGHT := A[i + 1 : j - 1]$
- $UNKNOWN := A[j : r - 1]$

Observe

- i is th last index of $LEFT$
- j is the first index of $UNKNOWN$

Quicksort

Define

- Let $x = A[r]$
- $LEFT := A[p : i]$
- $RIGHT := A[i + 1 : j - 1]$
- $UNKNOWN := A[j : r - 1]$

Loop Invariant

- $\forall a \in LEFT,\ a \leq x$
- $\forall b \in RIGHT,\ b > x$

Loop Invariant

Initialization

- $LEFT : A[p : p-1] = \emptyset$
- $RIGHT : A[p : p-1] = \emptyset$

Loop Invariant

Initialization

- $LEFT : A[p : p-1] = \emptyset$
- $RIGHT : A[p : p-1] = \emptyset$

Maintenance

- $A[j] \leq x$
  - $i++$
  - $A[i] \leftrightarrow A[j]$
  - $j++$
- $A[j] > x$
  - $j++$

Loop Initialization

Termination

- $j = r$
- $A[j : r - 1] = \emptyset$

Quicksort

Time Complexity

Quicksort

Time Complexity

- (Un)balanced Partitions
  - Worst-case
  - Best-case
  - Balanced

Quicksort

Worst-case Time Complexity

$$T(n) = \max_{0 \leq q \leq n-1} \big(T(q) + T(n - q - 1) \big) + \Theta(n)$$

Quicksort

Worst-case Time Complexity

$$T(n) = \max_{0 \leq q \leq n-1} \big(T(q) + T(n - q - 1)\big) + \Theta(n)$$

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

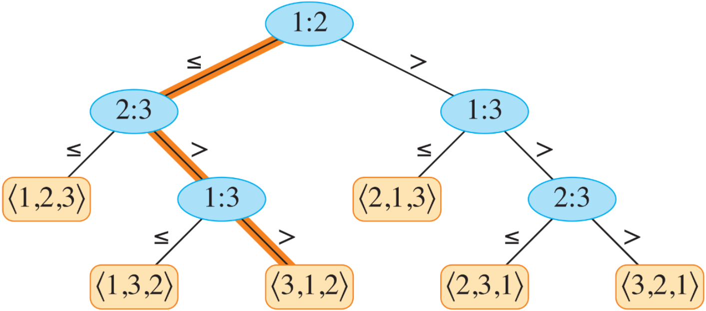
Comparison-Based Sorting

Insertion Sort

Merge Sort

Quicksort

Decision Trees



Lower Bound

Any comparison sort algorithm requires $\Omega(n \log n)$ comparisons in the worst case.