

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Quicksort and Quickselect: Partition, Then Conquer

Lecture 17 · Thursday 5 November 2026 · CLRS Ch 7 and 9.2; Canning, Broder & Lafore Ch 7

Dr. Innocent Nyalala | MTech Semester 1 | IIT Madras Zanzibar

Today's Plan

IIT Madras Zanzibar

1. (0–4 min) The idea: partition first, then sort each side
2. (4–16 min) Partition: every swap drawn; the loop invariant; the code
3. (16–23 min) Quicksort: the code and the full tree of calls
4. (23–33 min) Best and worst case; sorted input; the random pivot fix
5. (33–38 min) Quickselect: the k-th smallest in average O(n)
6. (38–50 min) Merge sort vs quicksort, demo, practice, your questions

Learning Outcomes

IIT Madras Zanzibar

By the end of this session, you will be able to:

Partition	a list around a pivot by hand (Lomuto), tracking i and j
Trace	quicksort as a tree of calls, and say where each pivot ends up
Explain	why quicksort is O(n log n) on average and $O(n^2)$ in the worst case
Fix	the worst case with a random pivot, and say what 'expected' time means
Use	partition to find the k-th smallest item in average O(n): quickselect

Key Terms for Today

IIT Madras Zanzibar

Term	Meaning	Example
Pivot	the item a partition compares everything with	the last item, a[hi]
Partition	rearrange so smaller items are left of the pivot and the rest right	[2 4 | 6 | 7 9]
In place	sorting inside the list, with O(1) extra memory per call	swaps only
Loop invariant	a statement true before and after every loop iteration	a[lo..i] < pivot
Balanced split	the pivot lands near the middle	4 | p | 3
Randomized algorithm	uses random choices; its time is an expectation	random pivot
k-th smallest	the item at index k after sorting (k = 0 is the minimum)	median: k = n // 2

Quicksort and Quickselect

IIT Madras Zanzibar

→ **The Idea**

Partition

Quicksort

Best, Worst and Random Pivots

Quickselect

Comparison, Demo and Practice

The Idea: Pick a Pivot, Split Around It

IIT Madras Zanzibar

1 Pick a pivot

e.g. the last item

2 Partition

smaller left, the rest right; the pivot lands in its final place

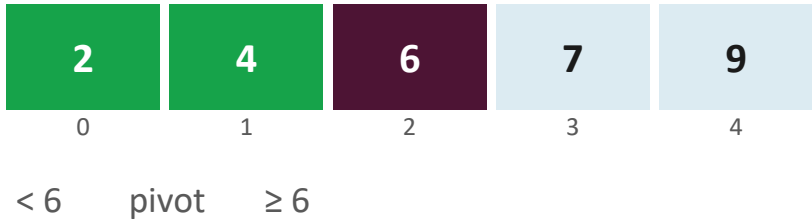
3 Recurse

quicksort the left part and the right part; nothing to combine

Before: pivot 6 is the last item



After partition: 6 is in its final place



? **Think:** after one partition, how many items are guaranteed to be in their final place?

Quicksort and Quickselect

IIT Madras Zanzibar

- The Idea
- → **Partition**
- Quicksort
- Best, Worst and Random Pivots
- Quickselect
- Comparison, Demo and Practice

Partition by Hand: Lomuto's Scheme

IIT Madras Zanzibar

start: pivot = a[4] = 6, i = −1



j = 0: 7 ≥ 6, leave it



j = 1: 2 < 6: i = 0, swap a[0], a[1]



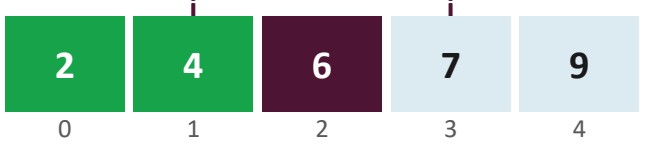
j = 2: 9 ≥ 6, leave it



j = 3: 4 < 6: i = 1, swap a[1], a[3]



end: swap pivot into a[i + 1] = a[2]



green: < pivot
light: ≥ pivot
orange: j, being checked
maroon: the pivot

? **Think:** how many comparisons did this partition make? How many for n items?

Why It Works: the Loop Invariant

IIT Madras Zanzibar

a[lo..i]	a[i+1..j−1]	a[j..hi−1]	a[hi]
< pivot	≥ pivot	not yet seen	pivot

1. Start: green and light zones are empty: true.
2. Big item at j: the light zone grows by one: still true.
3. Small item at j: i += 1, swap a[i] and a[j]: the green zone grows, the light zone shifts right: still true.
4. End (j = hi): nothing unseen; swap the pivot to i + 1, between the zones.

Definition: **Loop invariant:** a statement true before every iteration; if each iteration keeps it true, it is true when the loop ends.

? **Think:** why does the pivot go to index i + 1, not i?

The partition Code, Line by Line

IIT Madras Zanzibar

```
1  def partition(a, lo, hi):
2      pivot = a[hi]
3      i = lo - 1
4      for j in range(lo, hi):
5          if a[j] < pivot:
6              i += 1
7              a[i], a[j] = a[j], a[i]
8      a[i + 1], a[hi] = a[hi], a[i + 1]
9      return i + 1
```

Line by line

Line 2 pivot = the last item

Line 3 i = end of the 'smaller' zone; empty at first

Line 4 j scans every item except the pivot

Line 5–7 smaller: grow the zone and swap the item into it

Line 8 pivot goes between the zones

Line 9 return the pivot's index p

? **Think:** partition([1, 2, 3, 4, 5], 0, 4) returns what?

Quicksort and Quickselect

IIT Madras Zanzibar

- The Idea
- Partition
- → **Quicksort**
- Best, Worst and Random Pivots
- Quickselect
- Comparison, Demo and Practice

The quicksort Code

IIT Madras Zanzibar

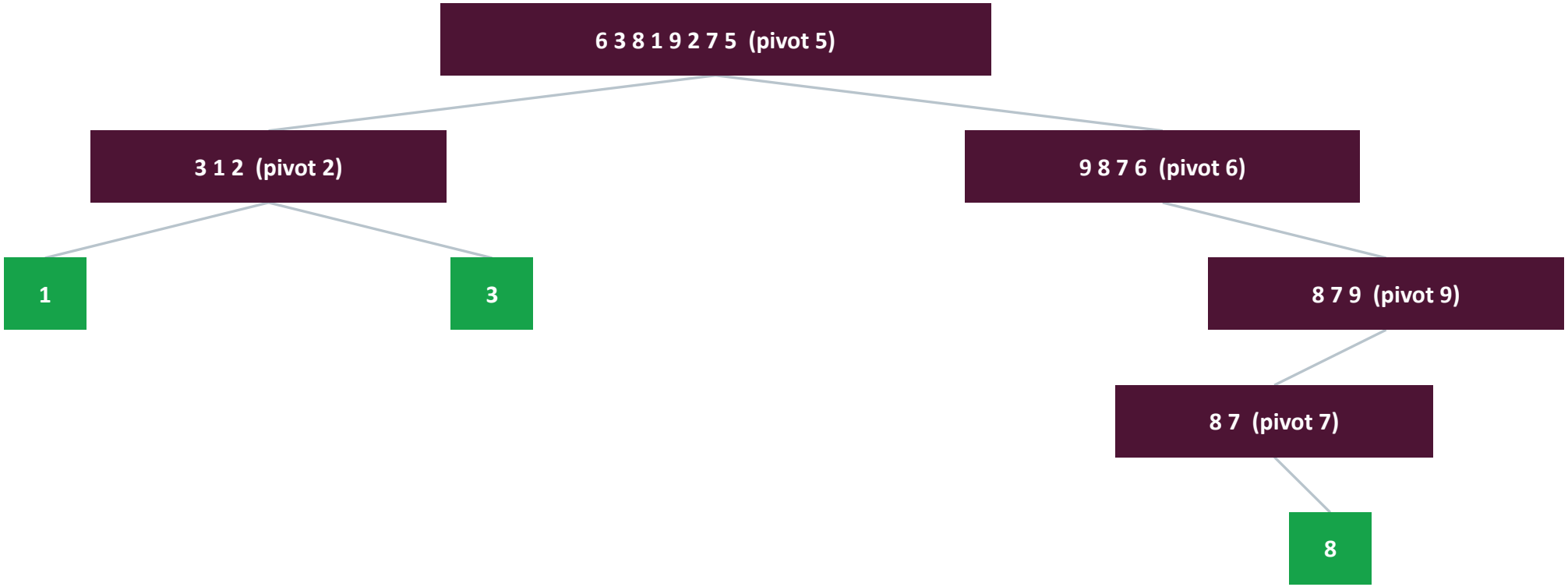
```
1  def quicksort(a, lo=0, hi=None):
2      if hi is None: hi = len(a) - 1
3      if lo < hi:
4          p = partition(a, lo, hi)
5          quicksort(a, lo, p - 1)
6          quicksort(a, p + 1, hi)
```

Line by line

Line 3	base case: parts of 0 or 1 item are sorted
Line 4	partition; p = the pivot's final index
Line 5	sort the left part a[lo..p−1]
Line 6	sort the right part a[p+1..hi]
—	no merge: the work was done by partition

? **Think:** why p − 1 and p + 1, and not p?

Quicksort on 8 Items: the Tree of Calls



Result: 1 2 3 5 6 7 8 9. Maroon: a partition with its pivot. Green: a part of one item (base case).

? **Think:** which pivots split their part well, and which were the worst possible?

Quicksort and Quickselect

IIT Madras Zanzibar

The Idea

Partition

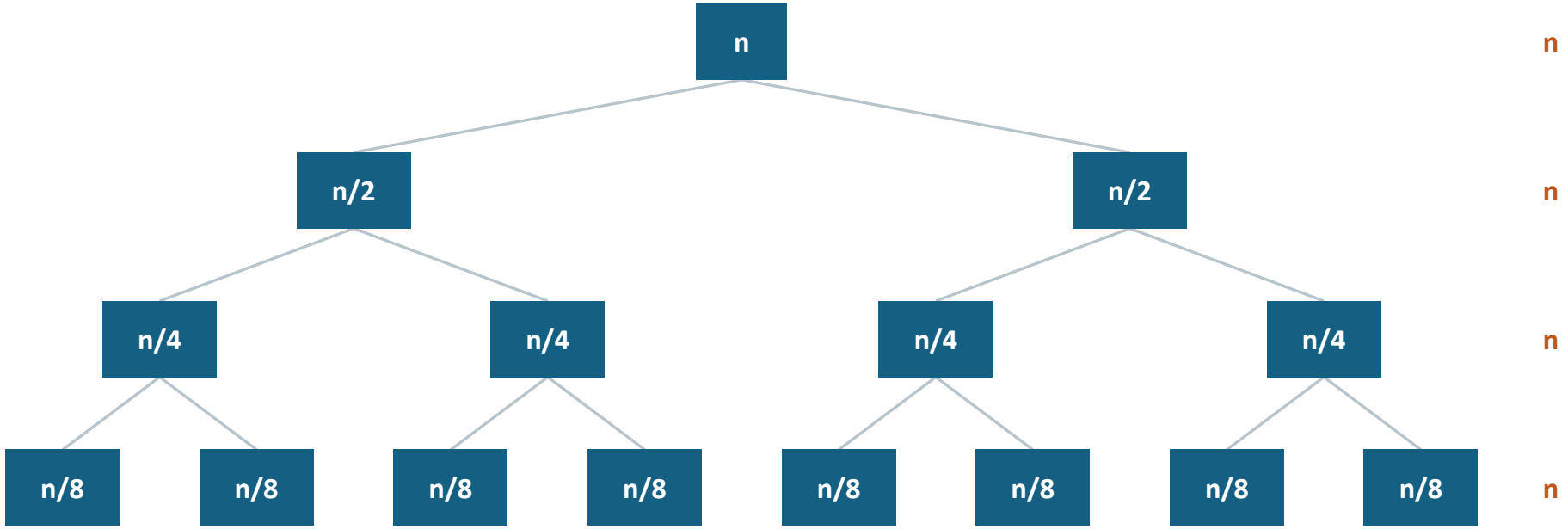
Quicksort

→ **Best, Worst and Random Pivots**

Quickselect

Comparison, Demo and Practice

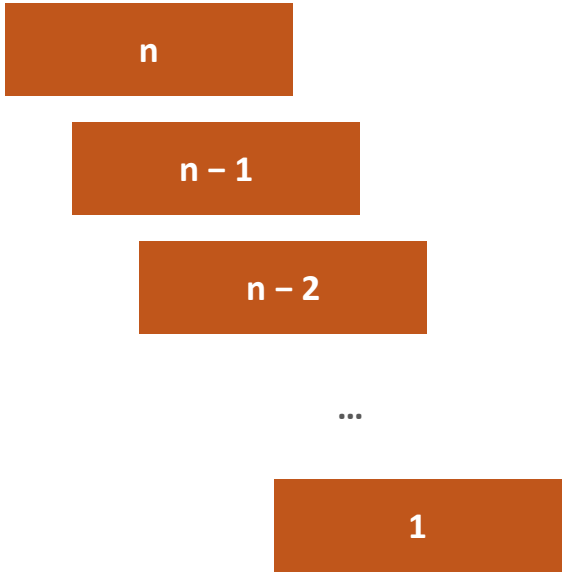
Best Case: Every Pivot Lands in the Middle



Best case: $T(n) = 2T(n/2) + n \rightarrow$ n per level × $\log_2$ n levels $\rightarrow O(n \log n)$

Think: best case, n = 1,000,000: how deep is the call stack?

Worst Case: Already Sorted Input



$T(n) = T(n-1) + n \rightarrow O(n^2)$

n (sorted input)	fixed last pivot	random pivot	n(n – 1)/2
500	124,750	about 4,900	124,750
1,000	499,500	about 11,700	499,500
2,000	1,999,000	about 23,600	1,999,000

1. Sorted input: the pivot is always the largest: one item peeled off per call.
2. Depth n: large sorted lists can raise RecursionError.

? **Think:** which other input is just as bad with the last item as pivot?

The Fix: Choose the Pivot at Random

IIT Madras Zanzibar

```
r = random.randint(lo, hi)        # a random position
a[r], a[hi] = a[hi], a[r]         # make it the pivot
p = partition(a, lo, hi)          # partition as before
```

1. No input is bad any more.
2. Expected ≈ 1.39 n $\log_2$ n comparisons.
3. Same code otherwise.

Definition: **Expected running time:** the average over the algorithm's own random choices, for the worst possible input.

1. Even a split of 1 : 9 at every level gives only about $\log_{10/9}$ n ≈ 6.6 $\log_2$ n levels: still O(n log n).

? **Think:** with a random pivot, can the O(n^2) case still happen? How likely is it?

Quicksort and Quickselect

IIT Madras Zanzibar

The Idea

Partition

Quicksort

Best, Worst and Random Pivots

→ **Quickselect**

Comparison, Demo and Practice

Quickselect: Partition, Then Recurse into ONE Side

IIT Madras Zanzibar

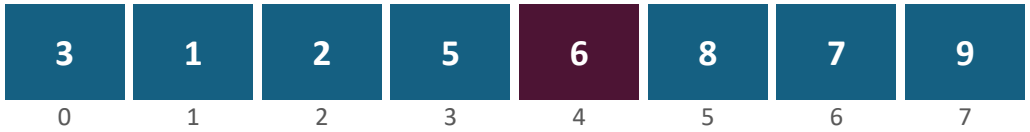
k = 5; partition a[0..7], pivot 5 lands at p = 3

k > p: go RIGHT only (a[4..7])



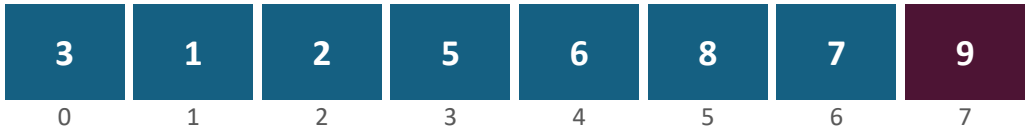
partition a[4..7], pivot 6 lands at p = 4

k > p: go RIGHT only (a[5..7])



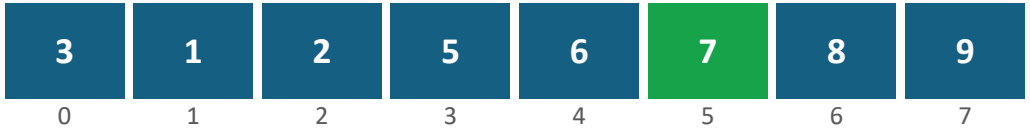
partition a[5..7], pivot 9 lands at p = 7

k < p: go LEFT only (a[5..6])



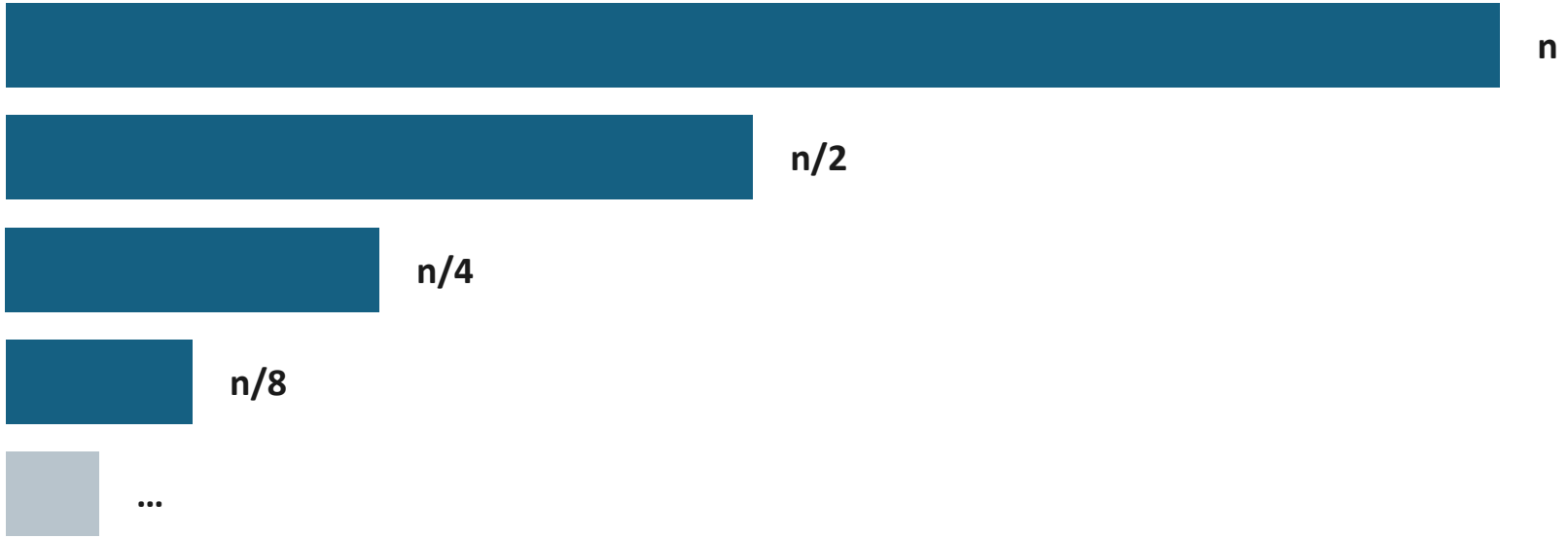
partition a[5..6], pivot 7 lands at p = 5

k == p: the answer is 7



? **Think:** why can the other side be ignored completely?

Quickselect's Cost: n + n/2 + n/4 + … ≈ 2n



n + n/2 + n/4 + n/8 + … < 2n → O(n) on average

	Quicksort	Quickselect	Sort, then index
calls per partition	2 (both sides)	1 (one side)	—
average cost	O(n log n)	O(n)	O(n log n)

? **Think:** why 2n and not n log n, when both start with a partition of n?

Quicksort and Quickselect

IIT Madras Zanzibar

The Idea

Partition

Quicksort

Best, Worst and Random Pivots

Quickselect

→ **Comparison, Demo and Practice**

Merge Sort or Quicksort?

IIT Madras Zanzibar

	Merge sort	Quicksort (random pivot)
average time	O(n log n)	O(n log n), expected; small constant
worst time	O(n log n)	$O(n^2)$ (very unlikely)
extra memory	O(n)	O(log n) expected, for the call stack
stable?	yes	no
work done	after the calls (merge)	before the calls (partition)
in practice	Python's sorted (Timsort), Java objects	C's qsort, C++ std::sort (introsort)

? **Think:** 100 million records, little memory, stability not needed: which sort, and why?

Live Demo

IIT Madras Zanzibar

```
cd ~/PDS/Class_Demos/Week_05
python3 W05_Thu_Nov05_quicksort_quickselect.py partition 7 2 9 4 6
python3 W05_Thu_Nov05_quicksort_quickselect.py 6 3 8 1 9 2 7 5
python3 W05_Thu_Nov05_quicksort_quickselect.py worst
python3 W05_Thu_Nov05_quicksort_quickselect.py select 6 6 3 8 1 9 2 7 5
```

What to watch

1. Compare each swap with the partition slide.
2. worst: fixed pivot = $n(n-1)/2$ exactly; random pivot $\approx 1.39\, n \log_2 n$.

In-Class Exercise (7 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: partition by hand

1. Partition [4, 8, 1, 6, 3, 5] with Lomuto (pivot 5). Give the array after each swap, the final array and p.

2. Quicksort with the last item as pivot on the sorted list [1, 2, …, 8]. How deep is the recursion? How many comparisons?

3. Quickselect for k = 1 (the 2nd smallest) in [7, 2, 9, 4, 6], last-item pivot. Which parts are partitioned? Answer?

4. Is quicksort stable? Partition [2a, 2b, 1] (pivot 1) and look at the order of 2a and 2b.

5. With a random pivot, which input makes quicksort slow?

Solution

IIT Madras Zanzibar

1

1: [4, 1, 3, 5, 8, 6], p = 3

swaps: a0↔a0; a1↔a2 → 4 1 8 6 3 5; a2↔a4 → 4 1 3 6 8 5; pivot a3↔a5.

2

2: depth 7, 28 comparisons

each partition peels off one item: 7 + 6 + … + 1 = n(n − 1)/2.

3

3: answer 4, two partitions

[2, 4, 6, 7, 9], p = 2 > k = 1 → left part [2, 4]; pivot 4 lands at 1 = k.

4

4: not stable

final swap: [2a, 2b, 1] → [1, 2b, 2a]: equal items changed order.

5

5: none in particular

the time depends on the random choices; every input has expected O(n log n).

Common Mistakes

IIT Madras Zanzibar

1 **i = lo instead of lo − 1**

the 'smaller' zone must start empty

2 **j up to hi**

j stops at hi − 1: the pivot is not compared with itself

3 **Recursing on p**

use p − 1 and p + 1: the pivot is already in place

4 **Quickselect into both sides**

only the side that contains index k

Summary and Exam Checkpoints

IIT Madras Zanzibar

Partition

pivot = a[hi]; i marks the smaller zone; pivot to i + 1; O(n), in place

Quicksort

partition, then sort both sides; nothing to combine

Cost

balanced: O(n log n); sorted input + fixed pivot: $O(n^2)$; random pivot fixes it

Quickselect

partition, recurse into ONE side: O(n) on average

Next

Today, 2:00–2:45: ICA 2 (laptop offline). Then Lab 4: recursion, merge sort, quicksort

Fri 6 Nov: Recurrences and the Master Theorem: solving T(n) = aT(n/b) + f(n)

Demo code: Class_Demos/Week_05/W05_Thu_Nov05_quicksort_quickselect.py

References and Further Reading

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). Introduction to Algorithms, 4th ed. MIT Press. Chapter 7 (Quicksort); Section 9.2 (selection in expected linear time).

[2] Hoare, C. A. R. (1962). Quicksort. The Computer Journal, 5(1), 10–16.

[3] Canning, J., Broder, A. & Lafore, R. (2022). Data Structures & Algorithms in Python. Addison-Wesley. Chapter 7 (Advanced Sorting: quicksort).

[4] Musser, D. R. (1997). Introspective sorting and selection algorithms. Software: Practice and Experience, 27(8), 983–993.