

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Sorting: The Big Picture, Bubble Sort and Selection Sort

Lecture 7 · Friday 16 October 2026 · Canning, Broder & Lafore Ch 3 (Simple Sorting)

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

Today's Plan

IIT Madras Zanzibar

1. (0–6 min) Why sorting matters, and what 'sorted' means exactly
2. (6–14 min) The family of sorting algorithms: a map of the next three weeks
3. (14–19 min) How we judge a sort: comparisons, swaps, memory, stability
4. (19–31 min) Bubble sort: idea, trace, code, cost, invariant
5. (31–41 min) Selection sort: idea, trace, code, cost, invariant
6. (41–50 min) Comparison, exercise, solution, summary

Learning Outcomes

IIT Madras Zanzibar

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

Define	what a sorted array is, and the terms comparison, swap, pass, in-place and stable
Name	the main sorting algorithms and their costs, and say when each is studied
Trace	bubble sort and selection sort by hand, pass by pass
Count	their comparisons and swaps, and explain why both are $O(n^2)$
Prove	each is correct using a loop invariant, and say which one is stable

Key Terms for Today

IIT Madras Zanzibar

Term	Meaning	Example
Sorted (ascending)	every item is ≤ the next one: a[0] ≤ a[1] ≤ … ≤ a[n−1]	[3, 5, 5, 9]
Key	the value we sort by	a student's mark
Comparison	one test of two items: is a[i] > a[j]?	5 > 3? yes
Swap	exchange two items' positions	[5, 3] → [3, 5]
Pass	one full sweep of the algorithm over the unsorted part	pass 1 of bubble sort
In-place	needs only a few extra variables, not a second array	bubble, selection
Stable	items with EQUAL keys keep their original order	two 70s stay in name order
Loop invariant	a statement true before and after every pass of a loop	'the last k items are final'

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

→ **Why Sorting Matters**

The Family of Sorting Algorithms

How We Judge a Sort

Bubble Sort

Selection Sort

Comparing the Two, and Practice

Why Sorting Matters

IIT Madras Zanzibar

1 Statements

mobile-money transactions by date

2 Rankings

exam results by mark; search results by relevance

3 Fast search

binary search needs sorted data: 20 steps instead of 1,000,000

4 Other problems

duplicates, median, merging: easy once sorted

Definition: Sorting: rearranging items so that a[0] ≤ a[1] ≤ … ≤ a[n−1] (ascending order). The items stay the same; only their order changes.

? **Think:** how would you find whether a list of 1,000,000 phone numbers contains a duplicate? How does sorting help?

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

- Why Sorting Matters
- → **The Family of Sorting Algorithms**
- How We Judge a Sort
- Bubble Sort
- Selection Sort
- Comparing the Two, and Practice

The Family of Sorting Algorithms

IIT Madras Zanzibar

Algorithm	Idea in one line	Time (worst)	When we study it
Bubble sort	swap neighbours that are out of order; the largest 'bubbles' to the end	$O(n^2)$	today
Selection sort	find the smallest remaining item and put it next	$O(n^2)$	today
Insertion sort	take the next item and insert it into the sorted part on its left	$O(n^2)$	Mon 19 Oct
Merge sort	split in half, sort each half, merge the two sorted halves	O(n log n)	Mon 2 Nov
Quicksort	pick a pivot; put smaller items left, larger right; repeat on each side	$O(n^2)$, average O(n log n)	Wed 4 Nov
Heap sort	build a heap, then remove the largest item repeatedly	O(n log n)	Week 9
Counting / radix sort	count how often each key occurs; no comparisons at all	O(n + k)	Fri 6 Nov
Timsort	Python's sorted(): merge sort + insertion sort on natural runs	O(n log n)	Fri 6 Nov

Today: the first two rows, in depth. The rest follow over the next three weeks.

Three Groups: Simple, Divide-and-Conquer, No Comparisons

IIT Madras Zanzibar

1

Simple sorts

$O(n^2)$: bubble, selection, insertion. Easy to write; fine for small n; teach us invariants

2

Divide and conquer

O(n log n): merge sort, quicksort (average), heap sort. Used on large data

3

No comparisons

O(n + k): counting, radix. Very fast when keys are small whole numbers

Why study slow algorithms at all?

1. Why start with the slow ones? They are the simplest place to learn the tools: tracing, counting steps, and proving correctness with an invariant.
2. Insertion sort, a simple sort, is still used inside Timsort for short runs, because on small or nearly sorted data it is very fast.

Definition: **Comparison sort:** a sort that learns about the order ONLY by comparing two items (a[i] > a[j]?). Bubble, selection, insertion, merge, quick and heap sort are all comparison sorts.

Why $O(n^2)$ Versus $O(n \log n)$ Matters

IIT Madras Zanzibar

n	n^2 (simple sorts)	$n \log_2 n$ (merge sort)	Ratio
1,000	1,000,000	about 10,000	100×
100,000	10,000,000,000	about 1,700,000	about 6,000×
1,000,000	10^{12}	about 20,000,000	about 50,000×

What this means in practice

1. Python does roughly 10^7 simple steps per second: 10^{12} steps take more than a day; 2×10^7 steps take about 2 seconds.
2. Simple sorts: fine for small n (tens or hundreds of items). Large data needs O(n log n).

Definition: $O(n^2)$, quadratic time: if n doubles, the work multiplies by 4. Typical of a loop inside a loop, each running up to n times.

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

- Why Sorting Matters
- The Family of Sorting Algorithms
- → **How We Judge a Sort**
- Bubble Sort
- Selection Sort
- Comparing the Two, and Practice

How We Judge a Sorting Algorithm: Four Measures

IIT Madras Zanzibar

1. Comparisons

how many times we test two items

2. Swaps / moves

how many times items are written to a new place

3. Extra memory

in-place (a few variables) or a whole second array?

4. Stability

do equal keys keep their original order?

Before (alphabetical)	Stable sort by mark	Unstable sort by mark
Asha 70	Baraka 55	Baraka 55
Baraka 55	Asha 70	Juma 70
Juma 70	Juma 70	Asha 70

1. Asha and Juma have the same mark.
2. Stable: Asha stays before Juma, as in the input.

Definition: **Stable sort:** a sort in which items with equal keys appear in the output in the same order as in the input.

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

Why Sorting Matters

The Family of Sorting Algorithms

How We Judge a Sort

→ **Bubble Sort**

Selection Sort

Comparing the Two, and Practice

Bubble Sort: The Idea

IIT Madras Zanzibar

The rule

1. Compare a[0] and a[1]; if a[0] > a[1], swap them.
2. Move one step right: compare a[1] and a[2]; swap if needed. Continue to the end of the unsorted part.
3. After this pass, the largest item is at the end, in its final place.
4. Repeat on the remaining items. Stop early if a whole pass makes no swap.

1 **After pass 1**

the largest item is last

2 **After pass k**

the k largest items are at the end, in order

3 **Stop**

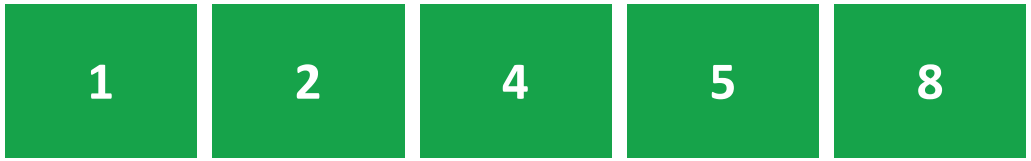
when a pass makes no swaps

Definition: Pass: one sweep from the start of the array to the end of the unsorted part, comparing every pair of neighbours.

Bubble Sort Traced: [5, 1, 4, 2, 8]

IIT Madras Zanzibar

Pass	Comparisons made (and swaps)	Array after the pass	Fixed at the end
start		[5, 1, 4, 2, 8]	
1	5>1 swap · 5>4 swap · 5>2 swap · 5<8 no	[1, 4, 2, 5, 8]	8
2	1<4 no · 4>2 swap · 4<5 no	[1, 2, 4, 5, 8]	5, 8
3	1<2 no · 2<4 no (no swaps: stop)	[1, 2, 4, 5, 8]	sorted



Totals: 9 comparisons, 4 swaps, 3 passes (the third pass had no swaps).

? **Think:** how many passes would an array that is ALREADY sorted need? How many comparisons?

Bubble Sort: The Code, Line by Line

IIT Madras Zanzibar

```
 1  def bubble_sort(a):
 2      n = len(a)
 3      for end in range(n - 1, 0, -1):
 4          swapped = False
 5          for i in range(end):
 6              if a[i] > a[i + 1]:
 7                  a[i], a[i + 1] = a[i + 1], a[i]
 8                  swapped = True
 9          if not swapped:
10              break
```

Line	What it does
1	sort the list a in place
2	n = number of items
3	end = last unsorted index: n−1, n−2, …, 1
4	no swap yet in this pass
5	i = 0 … end−1: neighbours a[i], a[i+1]
6	out of order?
7	swap the two neighbours
8	remember that we swapped
9–10	no swap in a whole pass: sorted, stop

Bubble Sort: Counting the Cost

IIT Madras Zanzibar

Worst and best case

1. Worst case (reverse order): passes make n−1, n−2, …, 1 comparisons.
2. Sum: 1 + 2 + … + (n−1) = n(n−1)/2. Pair the ends: each pair adds to n, and there are (n−1)/2 pairs.
3. So about $n^2/2$ comparisons and swaps: $O(n^2)$.
4. Best case (already sorted): one pass, n−1 comparisons, no swaps, stop: O(n).

n	Worst-case comparisons n(n−1)/2
5	10
10	45
1,000	499,500
1,000,000	about 5×10^{11}

Definition: **Worst case / best case:** the largest / smallest number of steps the algorithm takes over all inputs of size n.

? **Think:** check the formula: for our trace, n = 5. Worst case 10 comparisons; we made 9. Why fewer?

Bubble Sort: Why It Is Correct (Loop Invariant) and Stable

IIT Madras Zanzibar

Correctness in four steps

1. Invariant: after k passes, the last k items are the k largest, in sorted order, in their final positions.
2. Start (k = 0): nothing is claimed, so it is true.
3. One pass: the largest item of the unsorted part is carried to its end, so the statement now holds for k + 1.
4. End: all items are fixed, or a pass made no swaps (every neighbor pair is in order). Either way, the array is sorted.

Stable: the swap happens only when a[i] > a[i+1] (strictly). Equal items are never swapped, so they keep their order.

Definition: **Loop invariant:** a statement that is true before the loop starts and stays true after every pass. At the end it gives us what we wanted.

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

Why Sorting Matters

The Family of Sorting Algorithms

How We Judge a Sort

Bubble Sort

→ **Selection Sort**

Comparing the Two, and Practice

Selection Sort: The Idea

IIT Madras Zanzibar

The rule

1. Position 0: scan the whole array, find the smallest item, swap it into position 0.
2. Position 1: scan from index 1 to the end, find the smallest, swap it into position 1.
3. Continue: position i gets the smallest item of a[i … n−1].
4. After n−1 positions, the last item is automatically the largest.

1 Select

find the smallest remaining item

2 Place

swap it into the next position

3 At most one swap

per pass: at most n − 1 swaps in total

Definition: **Minimum:** the smallest item in a range. Finding it in a[i … n−1] needs n − 1 − i comparisons: you must look at every item.

Selection Sort Traced: [64, 25, 12, 22, 11]

IIT Madras Zanzibar

Pass i	Smallest in a[i…]	Swap	Array after the pass
start			[64, 25, 12, 22, 11]
0	11 (index 4)	a[0] ↔ a[4]	[11, 25, 12, 22, 64]
1	12 (index 2)	a[1] ↔ a[2]	[11, 12, 25, 22, 64]
2	22 (index 3)	a[2] ↔ a[3]	[11, 12, 22, 25, 64]
3	25 (index 3)	none needed	[11, 12, 22, 25, 64]

Totals: 10 comparisons (4 + 3 + 2 + 1), only 3 swaps. Bubble sort on the same array would make 9 swaps.

? **Think:** selection sort on an ALREADY sorted array: how many comparisons? Does it stop early like bubble sort?

Selection Sort: The Code, Line by Line

IIT Madras Zanzibar

```
1  def selection_sort(a):
2      n = len(a)
3      for i in range(n - 1):
4          m = i
5          for j in range(i + 1, n):
6              if a[j] < a[m]:
7                  m = j
8          if m != i:
9              a[i], a[m] = a[m], a[i]
```

Line	What it does
1–2	sort a in place; n = number of items
3	i = position being filled: 0 … n−2
4	m = index of the smallest seen so far
5	scan the rest: j = i+1 … n−1
6–7	found a smaller item? remember its index
8	smallest not already at position i?
9	swap it into position i (one swap per pass)

Selection Sort: Cost, Correctness and Stability

IIT Madras Zanzibar

Cost and correctness

1. Comparisons: $(n-1) + (n-2) + \ldots + 1 = n(n-1)/2$ in EVERY case: $O(n^2)$.
2. Swaps: at most one per pass, so at most $n - 1$: $O(n)$.
3. Invariant: after pass i, a[0 … i] holds the i + 1 smallest items, in order.

Why it is not stable

1. Input: [4a, 4b, 1] (4a and 4b have equal keys).
2. Pass 0: smallest is 1; swap a[0] ↔ a[2] → [1, 4b, 4a].
3. 4a now comes after 4b: the order of equal keys changed. NOT stable.



Definition: **Long-distance swap:** a swap of two items far apart; it can jump an item over an equal one, which breaks stability.

Sorting: The Big Picture, Bubble Sort and Selection Sort

IIT Madras Zanzibar

Why Sorting Matters

The Family of Sorting Algorithms

How We Judge a Sort

Bubble Sort

Selection Sort

→ **Comparing the Two, and Practice**

Bubble Versus Selection, Side by Side

IIT Madras Zanzibar

	Bubble sort	Selection sort
Comparisons (worst)	$n(n-1)/2$: $O(n^2)$	$n(n-1)/2$: $O(n^2)$
Comparisons (already sorted)	$n - 1$: $O(n)$, stops early	$n(n-1)/2$: $O(n^2)$, never stops early
Swaps (worst)	$n(n-1)/2$: $O(n^2)$	$n - 1$: $O(n)$
Extra memory	$O(1)$: in-place	$O(1)$: in-place
Stable?	yes (swaps only neighbours, only if >)	no (long-distance swaps)
Best use	teaching; detecting 'already sorted'	when writes are expensive (few swaps)

Next: Monday, insertion sort: also $O(n^2)$, but fast on nearly sorted data and stable. It is used inside Python's own sort.

Common Mistakes

IIT Madras Zanzibar

1 **Inner loop runs too far**

bubble: i goes to end − 1; a[i + 1] must exist

2 **Swapping inside selection's scan**

find the minimum first; swap ONCE after the scan

3 **Using >= in bubble sort**

swaps equal items and breaks stability

4 **'Selection sort stops early'**

no: it scans everything even on sorted data

In-Class Exercise (6 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: show the array after every pass

1. Bubble sort [3, 1, 2]: the array after each pass; total comparisons and swaps.

2. Selection sort [29, 10, 14, 37, 13]: the array after each pass; total swaps.

3. Records (Asha, 70), (Juma, 70), (Baraka, 55), in this order. Sort by mark with bubble sort, then with selection sort. Is Asha still before Juma in each?

Solution

IIT Madras Zanzibar

1 **1: two passes; 3 comparisons, 2 swaps**

pass 1: 3>1 swap → [1, 3, 2]; 3>2 swap → [1, 2, 3]. Pass 2: 1<2, no swap → stop.

2 **2: 3 swaps**

[10, 29, 14, 37, 13] → [10, 13, 14, 37, 29] → no swap (14 in place) → [10, 13, 14, 29, 37].

3 **3: bubble yes, selection no**

bubble: Baraka moves left one step at a time; Asha and Juma (equal) are never swapped → (Baraka, Asha, Juma). Selection: pass 0 swaps Asha with Baraka → (Baraka, Juma, Asha): Juma is now before Asha.

Quiz 1: expect a trace, a count of comparisons or swaps, and a stability question.

Summary and Exam Checkpoints

IIT Madras Zanzibar

Sorting matters

fast search, rankings, duplicates; three weeks of this course

The family

simple $O(n^2)$; divide and conquer $O(n \log n)$; counting $O(n + k)$

Bubble sort

swap neighbours; stops early; stable; $O(n^2)$, best $O(n)$

Selection sort

select the minimum, place it; $\leq n - 1$ swaps; not stable; always $O(n^2)$

Next

Mon 19 Oct: Insertion sort and inversions

Thu 22 Oct, 2:00 pm: ICA 1, then Lab 2: sorts and stacks

Read: Canning, Broder & Lafore, Chapter 3

References and Further Reading

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

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). Introduction to Algorithms, 4th ed. MIT Press. Chapter 2 (loop invariants), Problem 2-2 (bubble sort).

[3] Knuth, D. E. (1998). The Art of Computer Programming, Vol. 3: Sorting and Searching, 2nd ed. Addison-Wesley. Section 5.2.