

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Insertion Sort

Lecture 8 · Monday 19 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. (0–7 min) Sorting a hand of cards: the idea, with real cards
2. (7–12 min) From cards to an array: sorted part, unsorted part, key, i and j
3. (12–24 min) Step by step on [5, 2, 4, 6, 1, 3]: every comparison and shift
4. (24–32 min) The code line by line, and a live demo
5. (32–40 min) How fast? Best and worst case, and inversions
6. (40–50 min) Correctness, properties, exercise, summary

Learning Outcomes

IIT Madras Zanzibar

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

Explain	insertion sort with a hand of cards, and with an array split into sorted and unsorted parts
Trace	it by hand, showing key, i and j and every shift
Read	the code line by line, and run it
Count	its comparisons and shifts, using inversions; best case $O(n)$, worst $O(n^2)$
Prove	it correct with a loop invariant, and say why it is stable, in-place and online

Key Terms for Today

IIT Madras Zanzibar

Term	Meaning	Example
Sorted part	the left part a[0 … i−1], always in order	[2, 5]
Unsorted part	the right part a[i … n−1], not yet looked at	[4, 6, 1, 3]
Key	the item being inserted; the card in your hand	key = a[i] = 4
i	index of the next item to insert: 1, 2, …, n−1	i = 2
j	walks LEFT through the sorted part, comparing with the key	j = 1, then 0
Shift	move an item one place right to make room	a[j+1] = a[j]
Inversion	a pair of positions i < j with a[i] > a[j]: two items in the wrong order	(5, 2) in [5, 2]
Online	can sort items as they arrive, without seeing them all first	new exam marks arriving

Insertion Sort

IIT Madras Zanzibar

→ **Sorting a Hand of Cards**

From Cards to an Array

Step by Step: [5, 2, 4, 6, 1, 3]

The Code and a Live Demo

How Fast? Inversions

Correctness, Properties, Practice

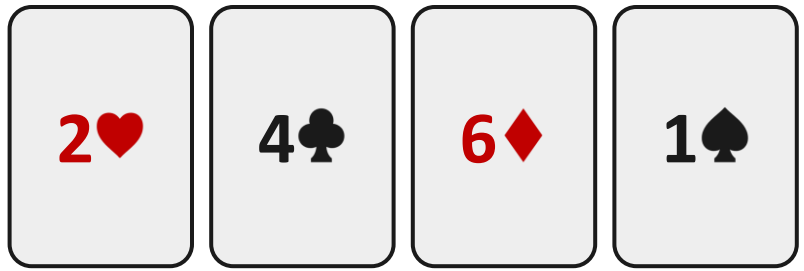
You Already Know This Algorithm

IIT Madras Zanzibar

Your hand (sorted)



Table (unsorted)



Start: the first card is a sorted hand of one. The other four are still on the table.

What you do with every new card

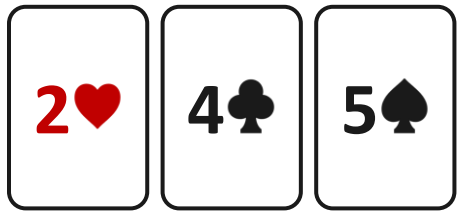
1. Your hand is always sorted.
2. Pick up the next card from the table.
3. Slide it left past every bigger card, and put it in the gap. Repeat until the table is empty.

? **Think:** with real cards: which card do you look at first when you slide a new card into your hand, the left end or the right end?

Card by Card: 5, 2, 4, 6, 1



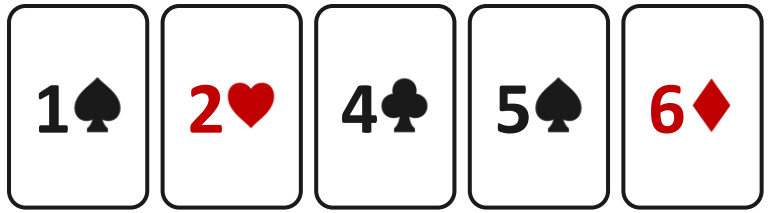
Pick 2: 2 < 5, so 2 goes before 5.



Pick 4: 4 < 5 but 4 > 2: between them.



Pick 6: 6 > 5: stays at the end. No moves.



Pick 1: smaller than all: slides to the front.

Insertion Sort

IIT Madras Zanzibar

Sorting a Hand of Cards

→ **From Cards to an Array**

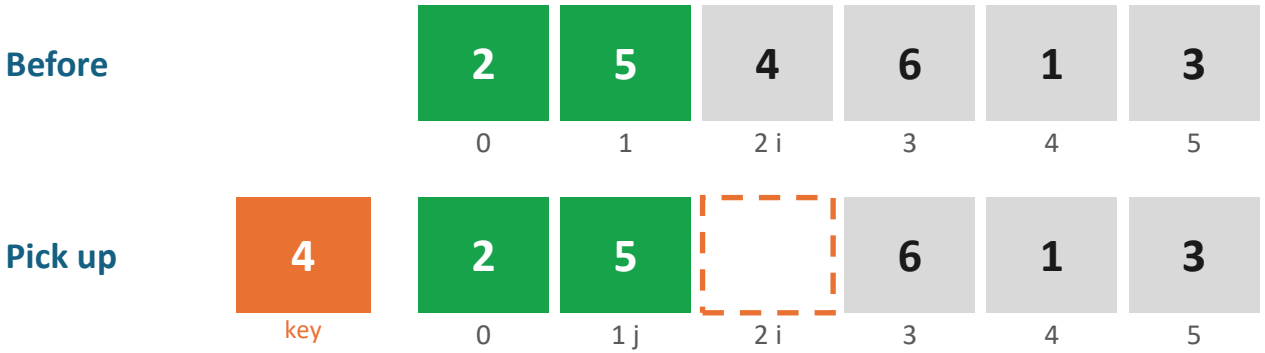
Step by Step: [5, 2, 4, 6, 1, 3]

The Code and a Live Demo

How Fast? Inversions

Correctness, Properties, Practice

From Cards to an Array: Two Parts in One Row



Sorted part a[0…1] = [2, 5]. Unsorted part starts at i = 2.

key = a[i] = 4; its box is a gap. j = i − 1 = 1.

The four things to track

1. Sorted part (green) = the hand. Unsorted part (grey) = the cards on the table.
2. i = index of the next card to insert. It moves right: 1, 2, …, n − 1.
3. key = a[i]: the card you are holding. Saving it lets us overwrite its box.
4. j starts at i − 1 and moves LEFT, comparing a[j] with key.

sorted part unsorted part key (the card in your hand) gap

The Rule for j: Shift or Stop

IIT Madras Zanzibar

1 **a[j] > key**

shift: a[j+1] = a[j]; then j = j − 1 (the gap moves left)

2 **a[j] ≤ key**

stop: the key belongs at j + 1

3 **j = −1**

ran off the left end: the key goes at index 0

```
key = a[i]
j = i - 1
while j >= 0 and a[j] > key:
    a[j + 1] = a[j]     # shift right
    j -= 1              # move left
a[j + 1] = key          # drop the key
```

Two details

1. Why ≤ stops and not <: equal items stay in their original order (stable).
2. Why save key first: the first shift overwrites box i.

Definition: **Shift:** copy an item one box to the right (a[j+1] = a[j]). Unlike a swap, it moves one item; the key is written only once at the end.

Insertion Sort

IIT Madras Zanzibar

Sorting a Hand of Cards

From Cards to an Array

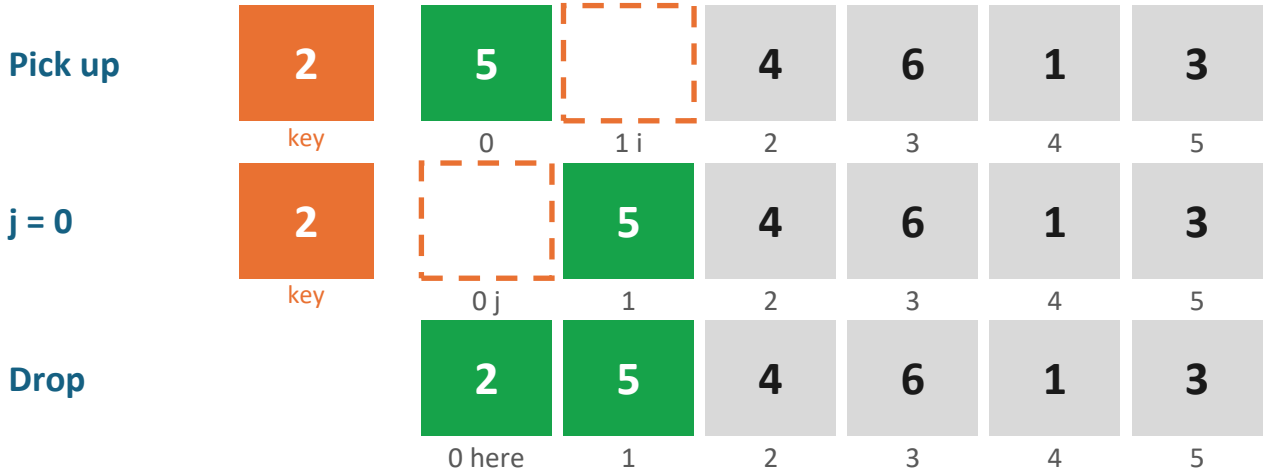
→ **Step by Step: [5, 2, 4, 6, 1, 3]**

The Code and a Live Demo

How Fast? Inversions

Correctness, Properties, Practice

Round i = 1: Insert key = 2



key = a[1] = 2. Its box is now free (the gap).

a[0] = 5 > 2: shift 5 one place right; the gap moves left.

j = −1: reached the left end, stop. Put 2 in the gap at index 0.

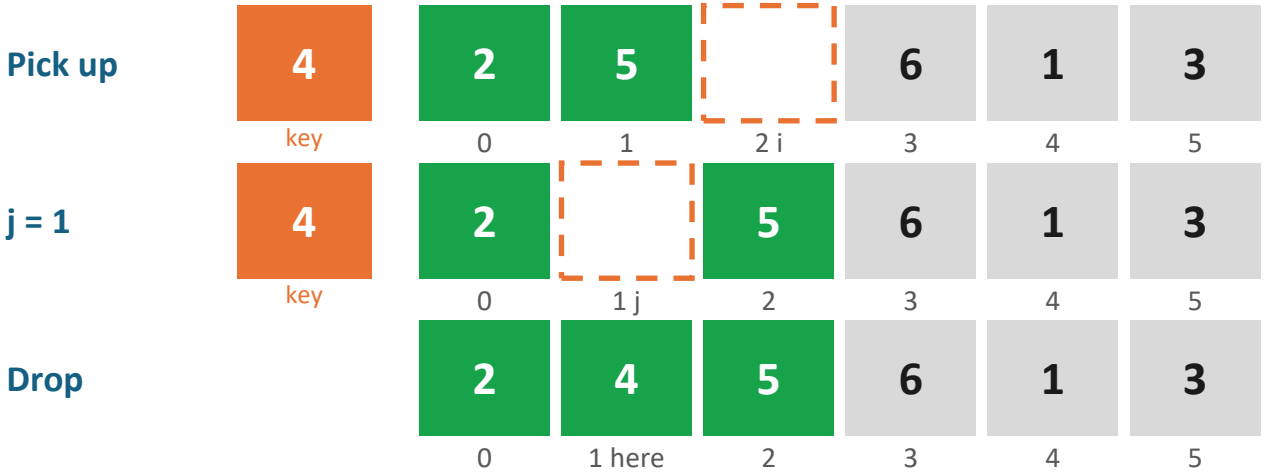
sorted part

unsorted part

key (the card in your hand)

gap

Round i = 2: Insert key = 4



key = a[2] = 4. Its box is now free (the gap).

a[1] = 5 > 4: shift 5 one place right; the gap moves left.

a[0] = 2 ≤ 4: stop. Put 4 in the gap at index 1.

sorted part

unsorted part

key (the card in your hand)

gap

Round i = 3: Insert key = 6



key = a[3] = 6. Its box is now free (the gap).

a[2] = 5 ≤ 6: stop. Put 6 in the gap at index 3.

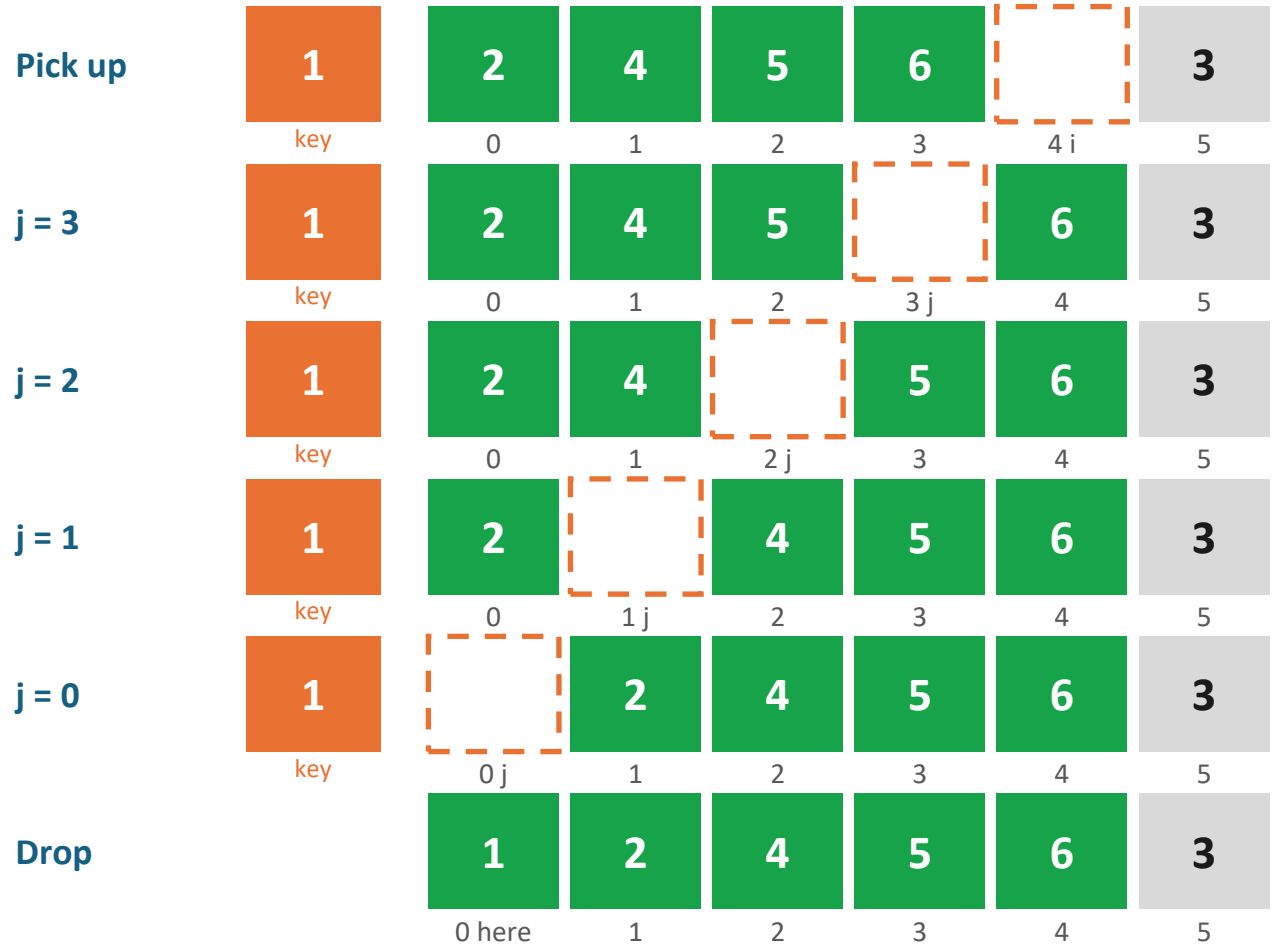
sorted part

unsorted part

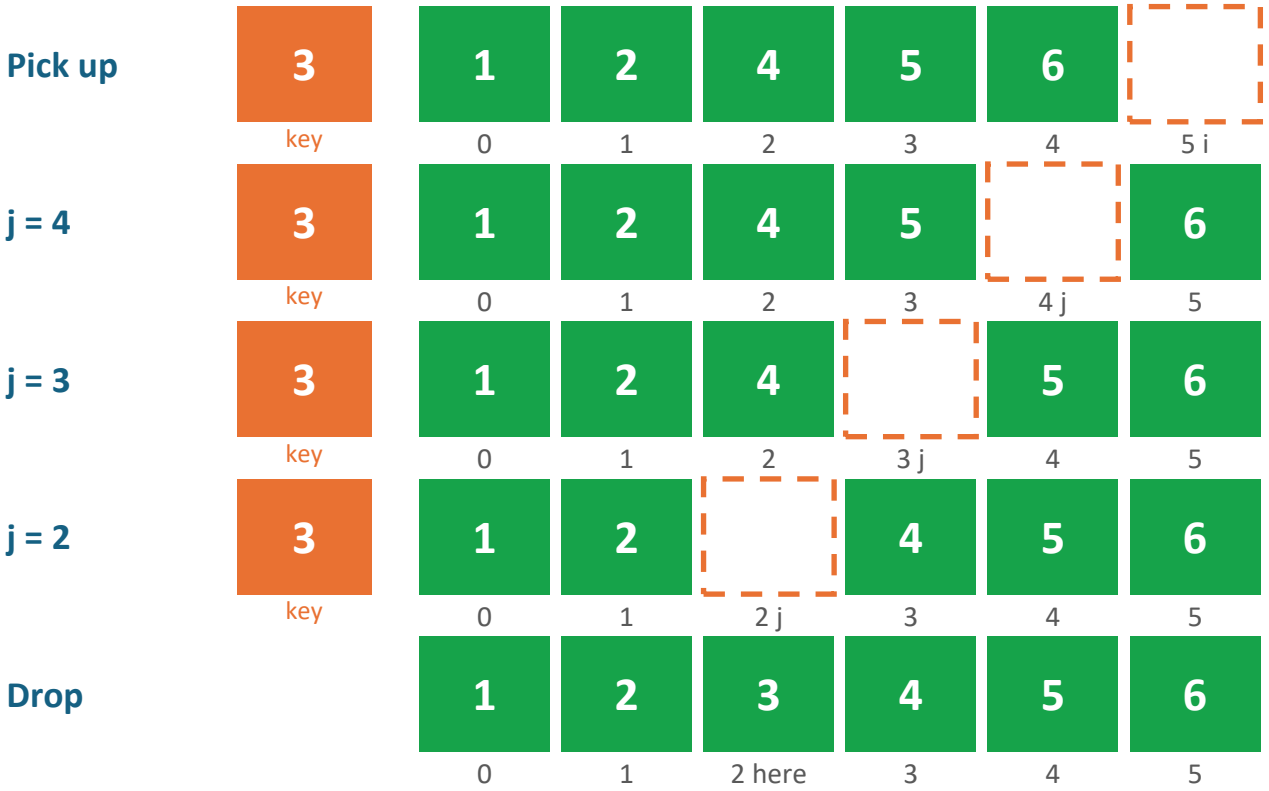
key (the card in your hand)

gap

Round i = 4: Insert key = 1



Round i = 5: Insert key = 3



sorted part unsorted part key (the card in your hand) gap

The Whole Run in One Table

IIT Madras Zanzibar

i	key	Comparisons	Shifts	Array after the round
1	2	1	1	[2, 5, 4, 6, 1, 3]
2	4	2	1	[2, 4, 5, 6, 1, 3]
3	6	1	0	[2, 4, 5, 6, 1, 3]
4	1	4	4	[1, 2, 4, 5, 6, 3]
5	3	4	3	[1, 2, 3, 4, 5, 6]
Total		12	9	sorted

Notice: items already in place cost 1 comparison and 0 shifts. Items far from home cost many shifts.

Insertion Sort

IIT Madras Zanzibar

Sorting a Hand of Cards

From Cards to an Array

Step by Step: [5, 2, 4, 6, 1, 3]

→ **The Code and a Live Demo**

How Fast? Inversions

Correctness, Properties, Practice

Insertion Sort: The Code, Line by Line

IIT Madras Zanzibar

```
1  def insertion_sort(a):
2      n = len(a)
3      for i in range(1, n):
4          key = a[i]
5          j = i - 1
6          while j >= 0 and a[j] > key:
7              a[j + 1] = a[j]
8              j -= 1
9          a[j + 1] = key
```

Line	What it does
1–2	sort a in place; n = number of items
3	i = next item to insert: 1 … n−1
4	key = the card in your hand
5	j starts just left of i
6	inside the array AND bigger than key?
7	shift a[j] one place right
8	j moves one step left
9	drop key into the gap at j + 1

Live Demo: Watch the Code Run

IIT Madras Zanzibar

```
cd Class_Demos/Week_03
python3 W03_Mon_Oct19_insertion_sort.py                # lecture example, every step
python3 W03_Mon_Oct19_insertion_sort.py 9 4 7 1 3      # students' own numbers
python3 W03_Mon_Oct19_insertion_sort.py race           # random vs nearly sorted vs reversed
```

```
i=4: key = 1, sorted part = [2, 4, 5, 6]
   j=3: 6 > 1, shift right    2    4    5   [6]  [6]   3
   j=2: 5 > 1, shift right    2    4   [5]  [5]   6    3
   ...
   place key 1 at index 0   [1]   2    4    5    6    3
```

Ask the class: point at one printed line and ask which line of the code produced it, and what i, j and key are at that moment.

Insertion Sort

IIT Madras Zanzibar

Sorting a Hand of Cards

From Cards to an Array

Step by Step: [5, 2, 4, 6, 1, 3]

The Code and a Live Demo

→ **How Fast? Inversions**

Correctness, Properties, Practice

Predict First: Four Inputs

IIT Madras Zanzibar

Input (n = 6)	Shifts	Comparisons	Why
[1, 2, 3, 4, 5, 6] sorted	0	5	every key stops after 1 comparison
[1, 2, 4, 3, 5, 6] nearly sorted	1	6	only 4 and 3 are out of order
[5, 2, 4, 6, 1, 3] our example	9	12	9 pairs out of order
[6, 5, 4, 3, 2, 1] reversed	15	15	every pair out of order

? **Think:** what single number about the input decides how many shifts insertion sort makes?

Inversions: The Number That Decides the Cost

IIT Madras Zanzibar

Item	Smaller items after it	Inversions
5	2, 4, 1, 3	4
2	1	1
4	1, 3	2
6	1, 3	2
1	none	0
Total		9 = shifts

Why shifts = inversions

1. Each shift moves one bigger item past the key: it removes exactly ONE inversion.
2. Sorting ends with 0 inversions.
3. So: number of shifts = number of inversions in the input.

Definition: **Inversion:** a pair of positions $p < q$ with $a[p] > a[q]$. A sorted array has 0; a reversed one has $n(n-1)/2$.

Best, Average and Worst Case

IIT Madras Zanzibar

Case	Inversions	Running time	Example
Best: already sorted	0	O(n): one comparison per item	a leaderboard that is already in order
Nearly sorted (k out of place)	small	O(n + inversions): close to O(n)	yesterday's sorted list plus a few new entries
Average: random order	about $n^2/4$	$O(n^2)$	shuffled data
Worst: reversed	$n(n-1)/2$	$O(n^2)$	data sorted the wrong way round

Definition: **Adaptive sort:** a sort that runs faster when the input is already partly sorted. Insertion sort and bubble sort (with early stop) are adaptive; selection sort is not.

Real use: Python's sorted() (Timsort) uses insertion sort on short runs of up to 64 items.

Insertion Sort

IIT Madras Zanzibar

Sorting a Hand of Cards

From Cards to an Array

Step by Step: [5, 2, 4, 6, 1, 3]

The Code and a Live Demo

How Fast? Inversions

→ **Correctness, Properties, Practice**

Why It Is Correct: The Loop Invariant

IIT Madras Zanzibar

Correctness in four steps

1. Invariant: before round i, a[0 … i−1] holds the original first i items, in sorted order.
2. Start (i = 1): one item is always sorted.
3. Each round: bigger items shift right and the key fills the gap, so a[0 … i] is sorted.
4. End (i = n): a[0 … n−1], the whole array, is sorted.

Compare with selection sort: there the left part holds the SMALLEST items, already final. Here it holds the FIRST items, and later keys can still slot in between them.

Four Properties, and Where It Is Used

IIT Madras Zanzibar

Property	Insertion sort	Why
Stable	yes	it stops at a[j] ≤ key, so a key never passes an equal item
In-place	yes	only key, i and j are extra: O(1) memory
Adaptive	yes	time is O(n + inversions)
Online	yes	it can insert each new item as it arrives

1 **Inside Python's sort**

Timsort uses it for short runs

2 **Live rankings**

insert each new score into its place as it arrives

3 **Small arrays**

faster than O(n log n) sorts for tiny n

The Three Simple Sorts, Side by Side

IIT Madras Zanzibar

	Bubble	Selection	Insertion
Worst-case comparisons	$n(n-1)/2$	$n(n-1)/2$	$n(n-1)/2$
Already sorted	$O(n)$, stops early	$O(n^2)$	$O(n)$
Moves	swaps = inversions	$\leq n - 1$ swaps	shifts = inversions
Stable	yes	no	yes
Online	no	no	yes

Takeaway: of the simple sorts, insertion sort is the one that is actually used.

Common Mistakes

IIT Madras Zanzibar

1 **Not saving the key**

the first shift overwrites a[i]: save key = a[i] first

2 **Wrong order in the while test**

check j >= 0 BEFORE a[j]; a[−1] is the last item in Python

3 **Using >= instead of >**

shifts equal items past each other: no longer stable

4 **Dropping the key at j**

the gap is at j + 1 when the loop stops

In-Class Exercise (7 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: show the array after every round

1. Insertion sort [3, 7, 4, 9, 5, 2]: the array after each round, and the number of shifts in each round.

2. Count the inversions in [3, 7, 4, 9, 5, 2] without sorting. Does it match your total shifts?

3. A list of 1,000,000 sorted marks gets 10 new marks appended at the end. Insertion sort or selection sort to re-sort it? Estimate the work for each.

Solution

IIT Madras Zanzibar

1 **1: shifts per round 0, 1, 0, 2, 5 (total 8)**

i=1: [3,7,4,9,5,2] (7 stays) · i=2: [3,4,7,9,5,2] · i=3: [3,4,7,9,5,2] (9 stays) · i=4: [3,4,5,7,9,2] · i=5: [2,3,4,5,7,9].

2 **2: 8 inversions = 8 shifts**

(3,2) (7,4) (7,5) (7,2) (4,2) (9,5) (9,2) (5,2): 8 pairs, the same as the total shifts.

3 **3: insertion sort**

each new mark shifts past at most about 1,000,000 items: roughly 10^7 steps at worst, far less if the marks are high. Selection sort always does $n^2/2 \approx 5 \times 10^{11}$ comparisons.

Quiz 1: expect a trace with key, i and j; an inversion count; and best or worst case with a reason.

Summary and Exam Checkpoints

IIT Madras Zanzibar

Cards in your hand

keep the left part sorted; insert each new item into it

key, i, j

i = next item; key = its value; j walks left, shifting bigger items

Shifts = inversions

cost O(n + inversions): O(n) sorted, $O(n^2)$ reversed

Properties

stable, in-place, adaptive, online; used inside Python's sort

Next

Wed 21 Oct: Stacks

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

Demo code: Class_Demos/Week_03/W03_Mon_Oct19_insertion_sort.py

References and Further Reading

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

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2022). Introduction to Algorithms, 4th ed. MIT Press. Section 2.1 (insertion sort, loop invariants); Problem 2-4 (inversions).

[3] Peters, T. (2002). Timsort description, listsort.txt, CPython source distribution.