

Indian Institute of Technology Madras
ZANZIBAR CAMPUS

Z5007 · PROGRAMMING AND DATA STRUCTURES

Arrays: Unordered, Ordered and Binary Search

Lecture 5 · Monday 12 October 2026 · Canning, Broder & Lafore, Ch 2 (Arrays)

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

Today's Plan

IIT Madras Zanzibar

1. (0–5 min) Recap, and a question about your phone's contacts
2. (5–12 min) The array: four operations, and how it sits in memory
3. (12–20 min) Unordered arrays: fast insert, slow search
4. (20–32 min) Ordered arrays and binary search, traced step by step
5. (32–40 min) The trade-off: when to keep data sorted
6. (40–48 min) Exercise and solution
7. (48–50 min) Summary and Thursday's first lab

Learning Outcomes

IIT Madras Zanzibar

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

Explain	why reading a[i] is O(1) but inserting at the front is O(n), using the memory picture
Implement	insert, search and delete for unordered and ordered arrays
Trace	binary search by hand, showing low, mid and high at every step
Compare	the Big-O of each operation in unordered versus ordered arrays
Prove	that binary search is correct, using a loop invariant

Your Phone Already Solved Today's Problem

IIT Madras Zanzibar



Phone contacts

kept SORTED by name: searching is fast, so typing 'Z' jumps straight there



Mobile-money transaction log

kept in ARRIVAL order: adding must be instant, millions of times an hour

? **Think:** both are arrays. Why does one stay sorted and the other not? What does each one do most often?

Arrays: Unordered, Ordered and Binary Search

IIT Madras Zanzibar

→ **The Array as a Data Structure**

Unordered Arrays

Ordered Arrays and Binary Search

The Trade-off

In-Class Exercise

An Array Supports Four Basic Operations

IIT Madras Zanzibar

1 Insert

add a new item, for example a new student's record

2 Search

find an item by its key, for example registration number Z5007-11

3 Delete

remove an item, and close the gap it leaves

4 Traverse

visit every item once, for example print the class list

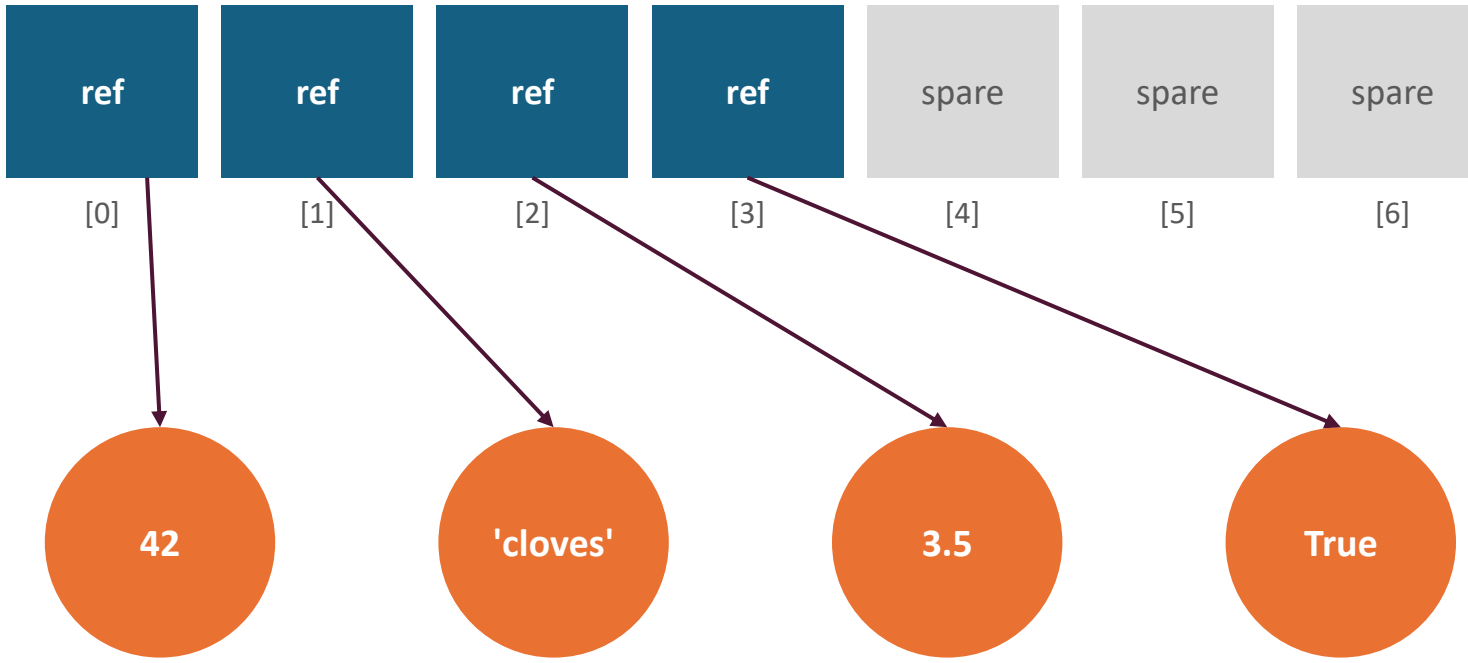
? **Think:** which of these four do YOU expect to cost O(n) no matter what we do? Why?

Measure: comparisons and moves, as a function of n. Assume no duplicates for now.

An Array in Memory: One Row of Equal-Sized Boxes

IIT Madras Zanzibar

```
a = [42, 'cloves', 3.5, True]
```



length = 4
capacity = 8

the values live elsewhere; each slot stores only an address

Key idea: one continuous block of equal-sized slots, with spare capacity at the end.

Why a[i] Is Instant: Address Arithmetic

IIT Madras Zanzibar

```
address of a[i]  =  start  +  i × 8 bytes
```

Index i	Calculation	Steps
0	start + 0 × 8	2
500	start + 500 × 8	2
9,999,999	start + 9,999,999 × 8	2

1 **Array**

jump straight to slot i: O(1)

2 **Train / linked list**

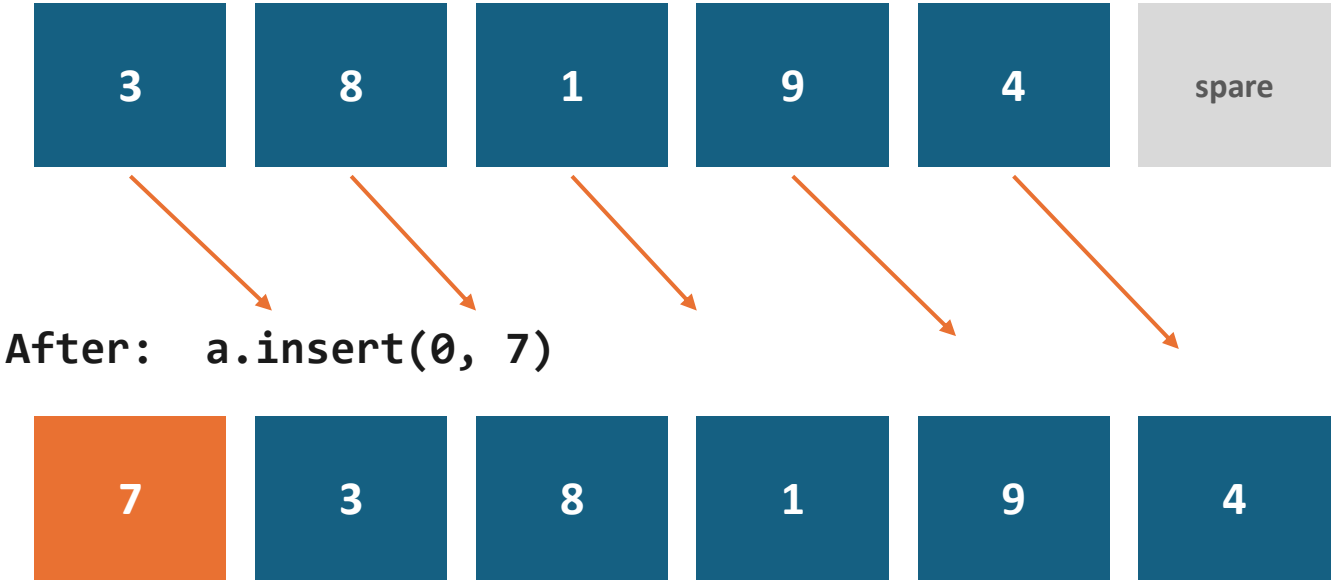
walk through i items: O(i)

? **Think:** an array of 1,000,000 phone numbers. How many steps to read the 700,000th one?

Inserting at the Front: Every Item Must Move

IIT Madras Zanzibar

```
Before:  a = [3, 8, 1, 9, 4]
```




```
After:  a.insert(0, 7)
```


80,000 items added	Time (measured)
insert(0, x)	1.13 s
append(x)	0.006 s

Cost: insert or delete at the front: O(n). At the end: O(1).

? **Think:** where in an array is inserting cheapest? Where is it most expensive? Why?

Arrays: Unordered, Ordered and Binary Search

IIT Madras Zanzibar

The Array as a Data Structure

→ **Unordered Arrays**

Ordered Arrays and Binary Search

The Trade-off

In-Class Exercise

Unordered Array: Insert Is O(1), Search Is O(n)

IIT Madras Zanzibar

Unordered array: search for 61



checked 27, 84, 15, then found 61: 4 comparisons

```
def search(a, key):
    for i in range(len(a)):
        if a[i] == key:
            return i
    return -1
```

Operation	Unordered array	Why
Insert	O(1)	append at the end
Search	O(n)	check items one by one
Delete	O(n)	search, then shift the rest left
Traverse	O(n)	visit every item

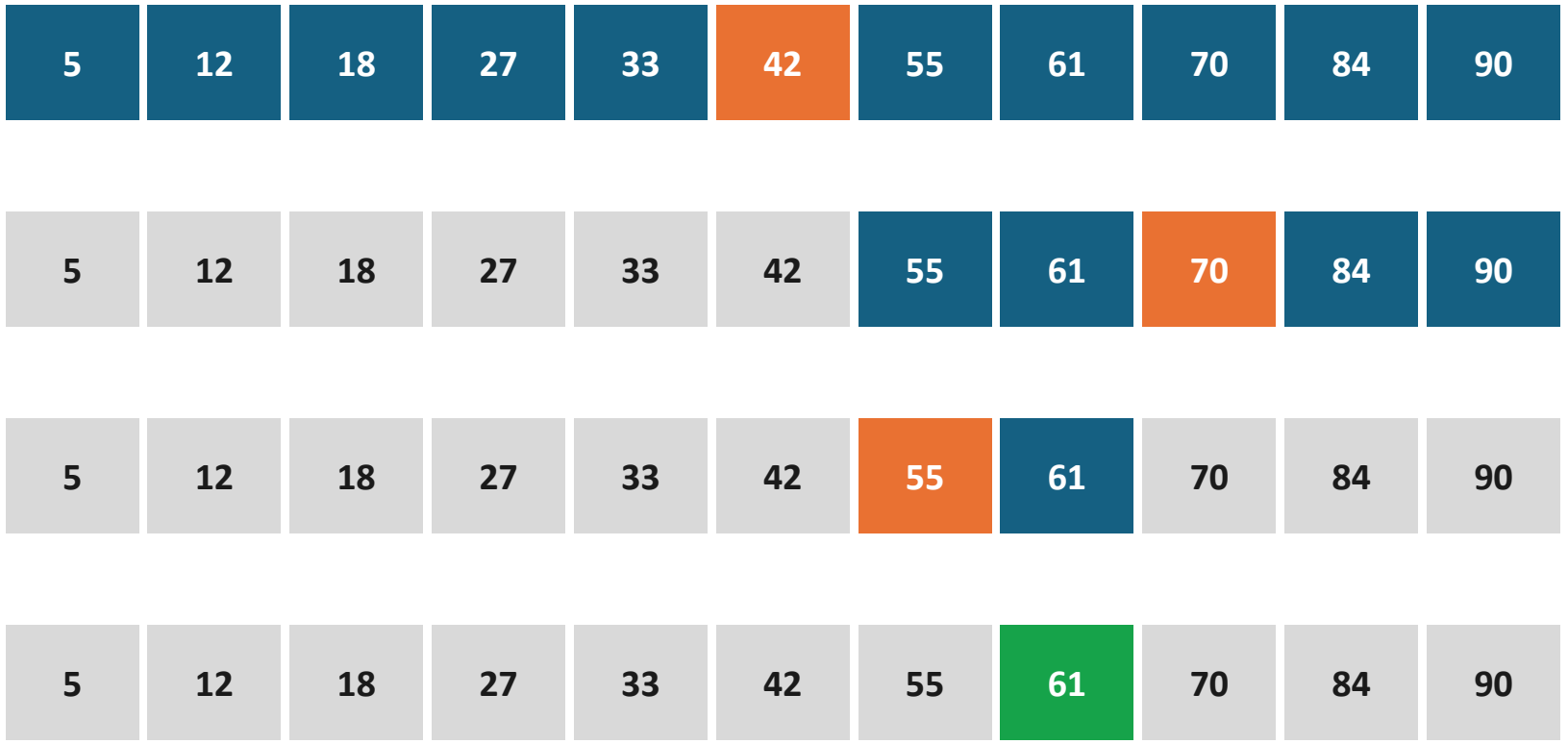
? **Think:** if half your searches are for items that are NOT there, what is the cost of each of those searches?

Arrays: Unordered, Ordered and Binary Search

IIT Madras Zanzibar

- The Array as a Data Structure
- Unordered Arrays
- → **Ordered Arrays and Binary Search**
- The Trade-off
- In-Class Exercise

Binary Search: Halve the Search Space at Every Step



teal = still possible orange = mid checked grey = ruled out

4 comparisons, not 11. For 1,000,000 items: at most 20.

Binary Search in Python, and the Classic Bug

IIT Madras Zanzibar

```
def binary_search(a, key):
    lo, hi = 0, len(a) - 1
    while lo <= hi:
        mid = (lo + hi) // 2
        if a[mid] == key:
            return mid
        elif a[mid] < key:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1
```

1 **lo <= hi, not lo < hi**

with <, a one-item range is never checked

2 **mid + 1 and mid - 1**

with lo = mid, two items can loop forever

3 **Cost**

halving each step: O(log n), like the guessing game

? **Think:** trace binary_search([3, 8], 8) with 'lo < hi' instead of 'lo <= hi'. What goes wrong?

Why Binary Search Is Correct

IIT Madras Zanzibar

Invariant: if key is in the array at all, then it is in a[lo..hi].

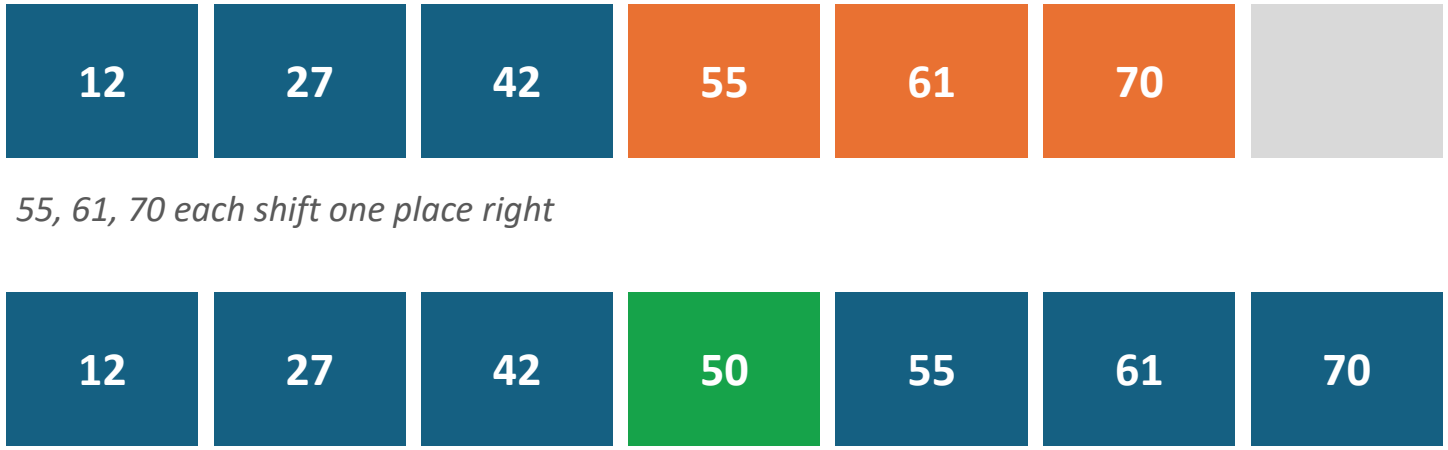
1	**Before the loop**	lo = 0, hi = n − 1: the range is the whole array. True.
2	**Each iteration**	if a[mid] < key, sorted order means a[0..mid] < key too; so lo = mid + 1 keeps it true. Same on the other side.
3	**The loop ends**	hi − lo shrinks every time, so it stops after at most about $\log_2 n$ rounds.
4	**At the end**	lo > hi: the range is empty, so the key is not in the array. Returning −1 is correct. ■

? **Think:** where in the proof did we use the fact that the array is SORTED?

The Price of Order: Insert and Delete Become O(n)

IIT Madras Zanzibar

Insert 50 into a sorted array



1 Find the spot

binary search: O(log n)

2 Make room

shift larger items: O(n)

Insert = O(log n) + O(n) = O(n). The shifting dominates. Delete is the same, shifting left.

? **Think:** for an ordered array, which is cheaper: inserting 1 (the smallest) or 99 (the largest)? Why?

Arrays: Unordered, Ordered and Binary Search

IIT Madras Zanzibar

The Array as a Data Structure

Unordered Arrays

Ordered Arrays and Binary Search

→ **The Trade-off**

In-Class Exercise

The Trade-off, in One Table

IIT Madras Zanzibar

Operation	Unordered array	Ordered array
Insert	O(1)	O(n)
Search	O(n)	O(log n)
Delete	O(n)	O(n)
Traverse in sorted order	O(n log n): sort first	O(n)

1 **Mostly searching?**

keep it sorted: dictionaries, results lookup, contacts

2 **Mostly inserting?**

keep arrival order: logs, sensor readings, transactions

3 **Both, heavily?**

neither array: binary search trees and hash tables, Weeks 7–8

? **Think:** a ride-hailing app adds 10,000 trips a minute and is searched by trip ID 10 times a minute. Ordered or unordered?

A Quick Calculation: When Does Sorting Pay Off?

IIT Madras Zanzibar

Daily workload (n = 1,000,000)	Unordered: I·1 + S·n	Ordered: I·n + S·20	Winner
1,000 inserts, 1,000 searches	about 10^9	about 10^9	a draw
1,000 inserts, 10,000 searches	about 10^{10}	about 10^9	ordered, ~10×
10,000 inserts, 1,000 searches	about 10^9	about 10^{10}	unordered, ~10×

Rule of thumb: compare searches × n against inserts × n. Whichever operation dominates decides.

? **Think:** what if you could sort ONCE, then only search (no more inserts)? Which wins, and by how much?

Common Mistakes to Avoid

IIT Madras Zanzibar

1. **Counting from 1**
   the first index is 0; the last is len(a) − 1

2. **Binary search on unsorted data**
   it gives wrong answers silently

3. **lo < hi instead of lo <= hi**
   misses a one-item range

4. **lo = mid instead of mid + 1**
   can loop forever on two items

5. **Forgetting the shift**
   insert or delete in the middle is O(n), not O(1)

6. **Mixing up cases**
   Big-O here means the worst case

Lab 1 on Thursday: you will write these functions yourself, and the tests will catch every mistake above.

In-Class Exercise (6 minutes, in pairs)

IIT Madras Zanzibar

IN-CLASS TASK: answer with Big-O and a one-line reason



1. Trace binary search for 47 in the array above. List lo, hi and mid at each step.

2. What is the MAXIMUM number of comparisons binary search needs for 1,000 items? For 1,000,000?

3. A university stores 20,000 student records and looks one up 50,000 times a day; 300 new students join each year. Ordered or unordered?

4. Delete 21 from the array above. How many items move? What is the cost in Big-O?

Solution

IIT Madras Zanzibar

1 **1: two steps**

lo=0, hi=8, mid=4: 30 < 47, so lo=5. Then lo=5, hi=8, mid=6: 47, found.

2 **2: 10 and 20**

$\lceil \log_2 1{,}000 \rceil = 10$; $\lceil \log_2 1{,}000{,}000 \rceil = 20$. Each comparison halves the range.

3 **3: ordered**

50,000 searches a day vs 300 inserts a year: searching dominates by far; keep the records sorted.

4 **4: five items move, O(n)**

30, 38, 47, 52, 66 each shift one place left. In the worst case every item shifts.

Exam tip: for tracing questions, always show lo, hi and mid at every step.

Summary and Exam Checkpoints

IIT Madras Zanzibar

Array operations

insert, search, delete, traverse, measured as O(f(n))

Unordered

insert O(1), search O(n): best when you mostly add

Ordered + binary search

search O(log n), insert O(n): best when you mostly look up

Decide with numbers

compare inserts × cost against searches × cost

Next

Thu 15 Oct, 10:00: Dynamic arrays: how Python lists grow, and amortized cost

Thu 15 Oct, 2:00 pm: Lab 1: build an Array class and binary search; measure them

Read: Canning, Broder & Lafore, Ch 2 (ordered arrays, binary search)

References and Further Reading

1. Canning, J., Broder, A. & Lafore, R. (2022). Data Structures & Algorithms in Python. Addison-Wesley. Chapter 2, Arrays.
2. Bentley, J. (2000). Programming Pearls, 2nd ed. Addison-Wesley. Column 4, Writing Correct Programs (binary search).
3. Skiena, S. S. (2020). The Algorithm Design Manual, 3rd ed. Springer. Section 5.1, Binary Search and Related Algorithms.
4. VisuAlgo, sorting and searching animations: visualgo.net