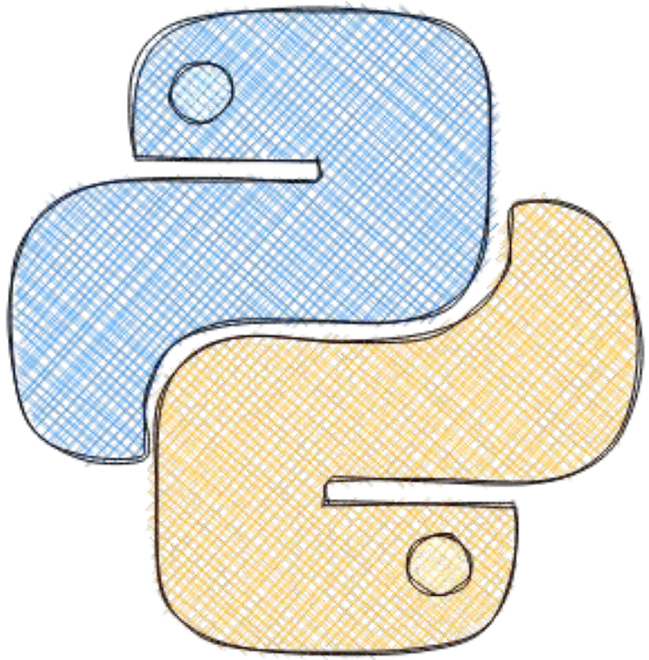


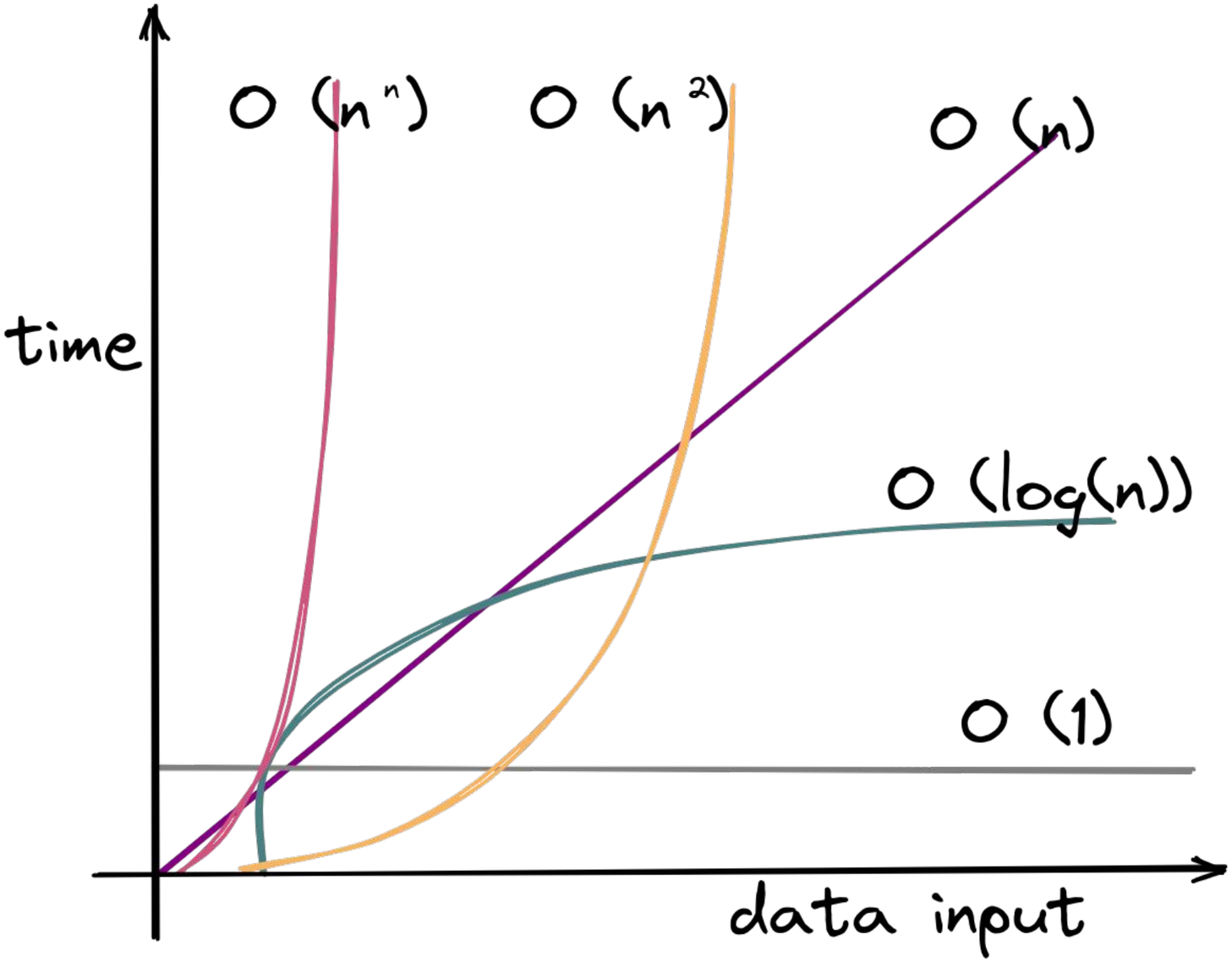
Data Structures and Algorithms

Week 3

Introduction to Algorithms

Middlesex University Dubai; Winter22
CST4050; Instructor: Ivan Reznikov

Algorithm time complexity



- O (1) – array access
- O (log(n)) – binary search
- O (n) – linear search
- O (n²) – bubble sort
- O (nⁿ) – Hanoi tower problem

Algorithm time complexity

- $O(100) \rightarrow O(1)$
- $O(2*\log(n)) \rightarrow O(\log(n))$
- $O(5*n+6) \rightarrow O(n)$
- $O(n^2 + 2*n) \rightarrow O(n^2)$
- $O(n^n + 2*n^3 + 5\log(n)) \rightarrow O(n^n)$

Algorithm time complexity

```
def return_sum(N):
    result_sum = 0                          O(1)
    for i in range(N+1):                    O(n)
        result_sum = result_sum + i
    return result_sum                       O(1)
```

return_sum(3) → 3+2+1 → 6

return_sum(10) → 55

O(1 + n + 1) = O(n)

Algorithm time complexity

Given an array of integers *nums* and an integer *target*, return *indices* of the two numbers such that they add up to *target*. You may assume that each input would have exactly one solution, and you may not use the same element twice.

Example 1:

Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:
Input: nums = [3,2,4], target = 6
Output: [1,2]

https://leetcode.com/problems/two-sum

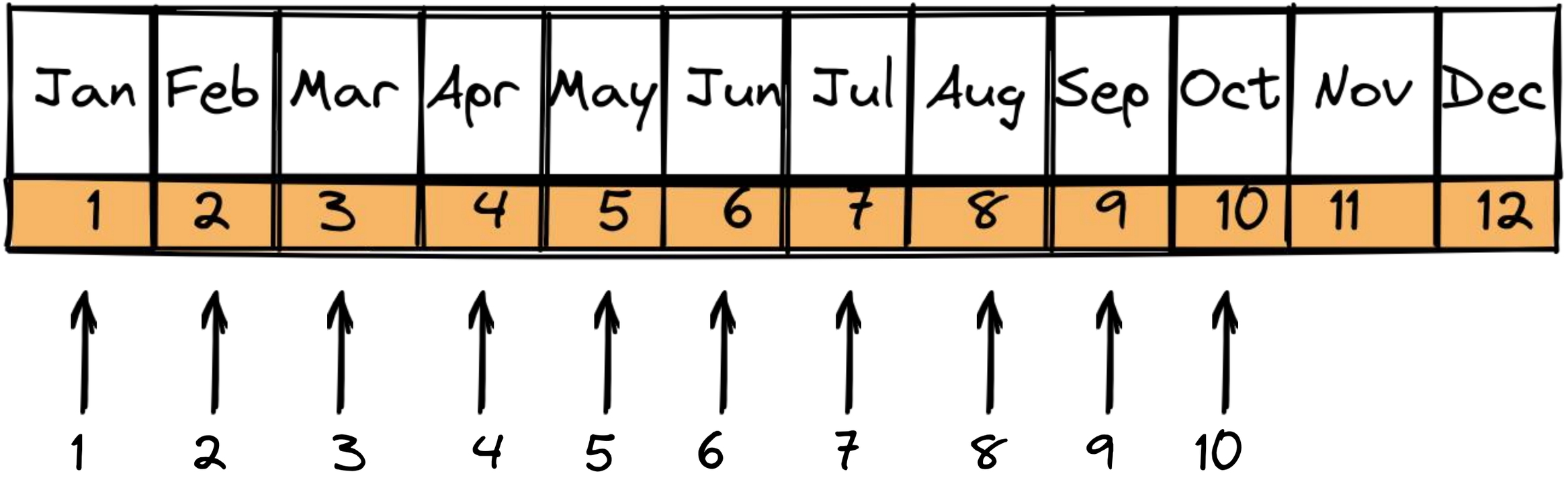
Algorithm time complexity

```
def sum_of_two(arr, target):
    for i, x in enumerate(arr):                          O(N)
        for i2, y in enumerate(arr[i+1:]):               O(N)
            if x+y == target:
                return i, i2+i+1                         O(1)
```

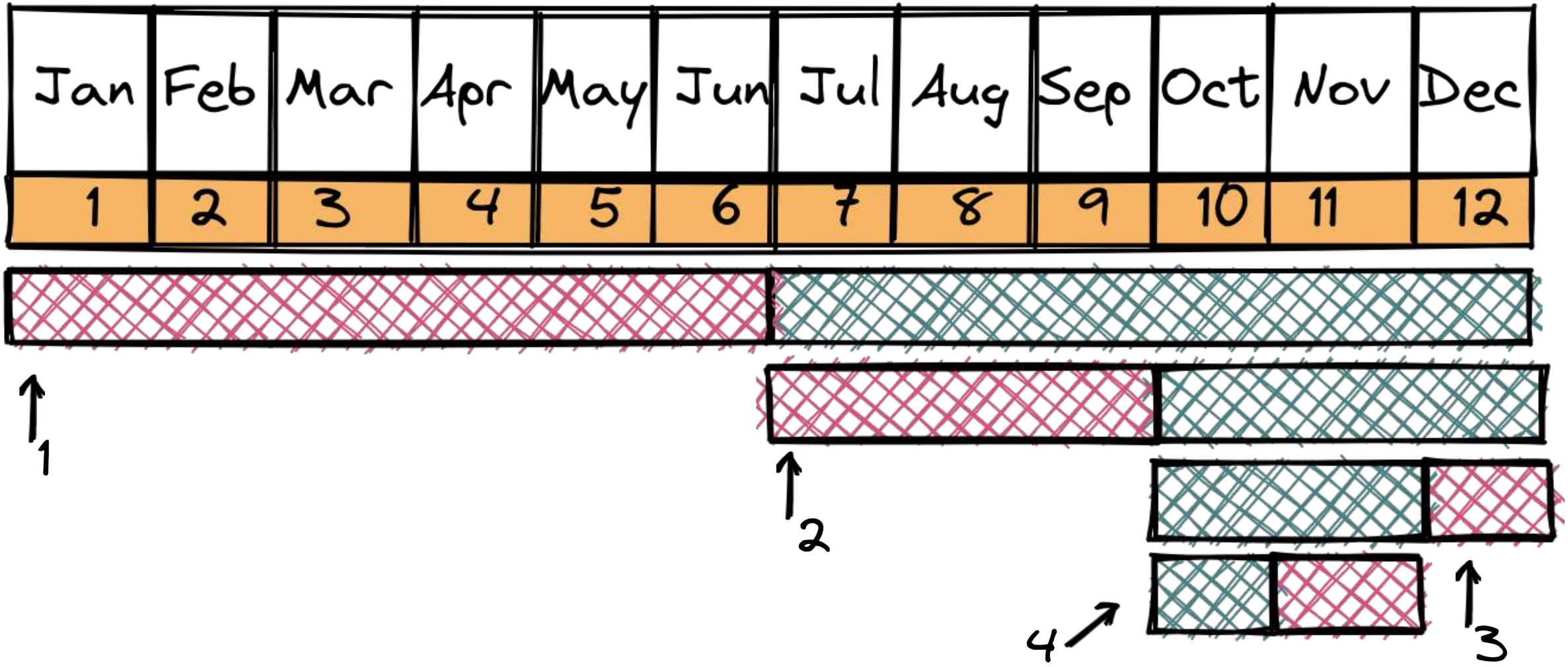
Algorithm time complexity

```
def sum_of_two(arr, target):
    passed_values ={}                                   O(1)
    for i in range(len(arr)):                           O(N)
        if target-arr[i] in passed_values:
            return [passed_values[target-arr[i]], i]
        passed_values[arr[i]] = i                       O(1)
```

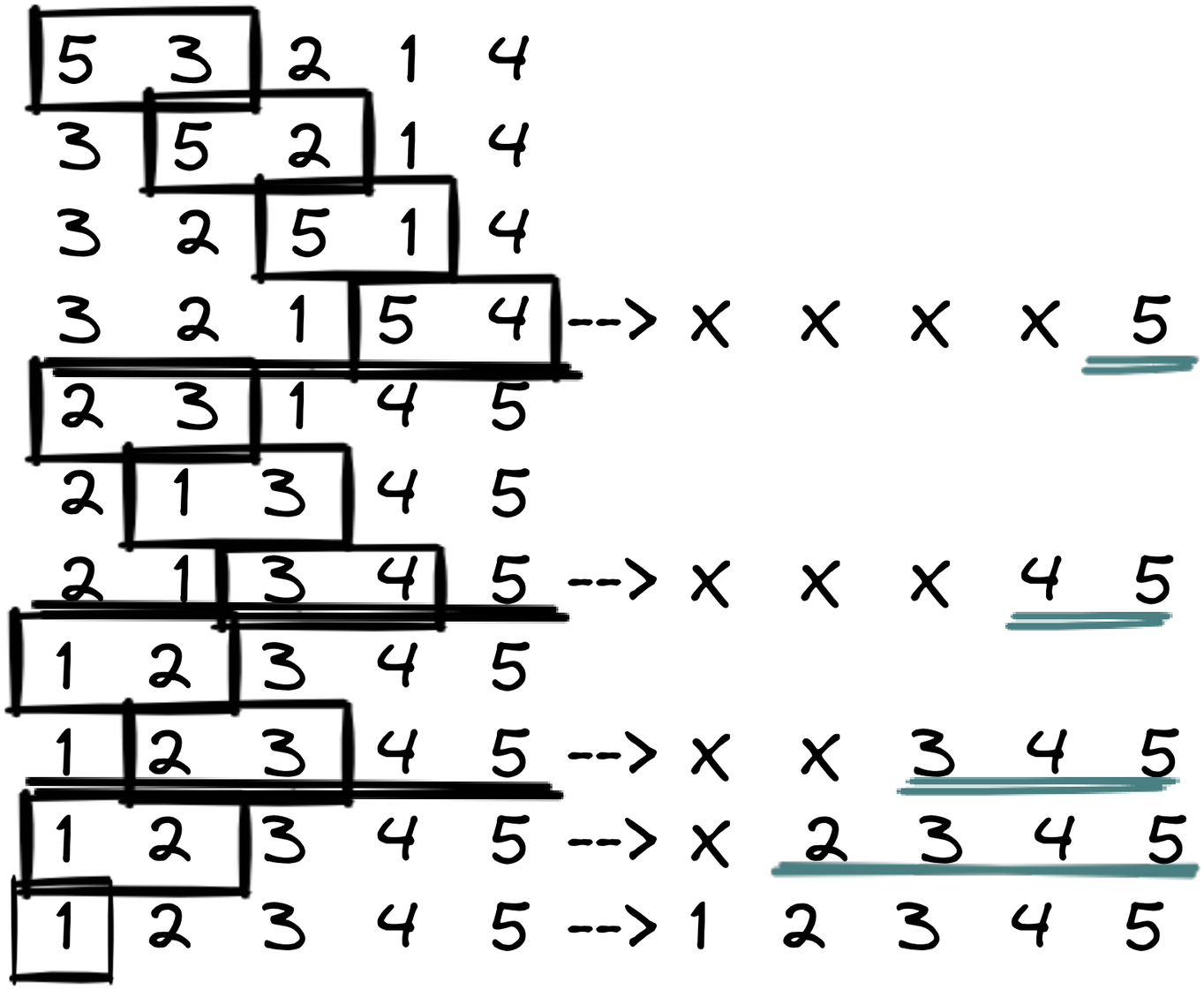
Basic search: O (n)



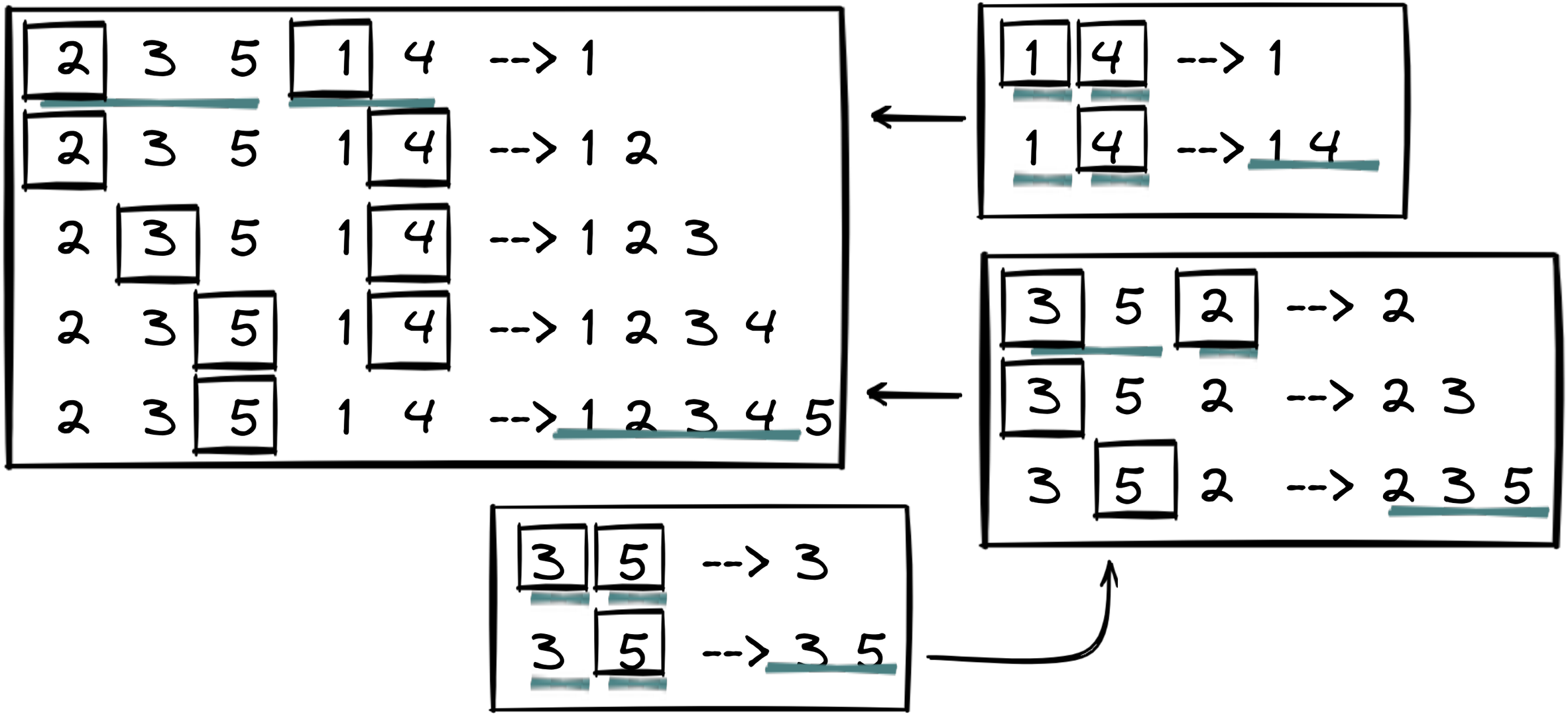
Binary search: O (log(n))



Bubble sort: O (n^2)



Merge sort: O (n×log(n))



Time Space Complexity

O (1) – constant time
O (N) – time proportional to N

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)

P/ NP Problems

P is the complexity class of decision problems that can be solved in polynomial time. (*multiplication*)

NP (nondeterministic polynomial) is a complexity class of all decision problems for which the problem solution can be **verified** in polynomial time (*Sudoku solver*).

NP Hard