SHIFTKEY ACADEMY

Intro to Algorithmic Problem Solving

Week 1

SHIFTKEY ACADEMY

About ShiftKey Labs

ShiftKey is more than just a platform; it's a vibrant ecosystem dedicated to fostering innovation and collaboration across various industries. They actively seek partnerships and engagement with businesses, startups, and organizations, understanding that the most impactful solutions emerge from collective efforts.

SHIFTKEY SANDBOX

SHIFTKEY ACADEMY

SHIFTKEY BUILD

SHIFTKEY WORKS

SHIFTKEY ACADEMY

About Me

Yuhan Fu (She/Her)

Side A

- MCS @ Dalhousie
- Research - Optimization Algorithms
- TA for CSCI 3110 & CHIN 1031/1032
- Volunteering as a full-stack developer for a startup
- Board member of Nova Scotia Chinese Women+ Network (IT support)
- Joined 4 hackathons and won awards in 3 of them
- Interested in building AI Agents

Side B

- A dancer
- Running a K-pop dance cover team for more than 3 years
- Covered 20 songs and gave 14 public performances
- Teaching K-pop dance classes

SHIFTKEY ACADEMY

Course Schedule

- Lectures: June 5th, 12th, 19th, 26th 18:00-20:00
- Assignments: due June 11th, 18th, 25th, and July 2nd
- Soft Skill Submission: due July 2nd
- Exam: July 3rd 18:00-20:00

Join on Brightspace:
https://dal.brightspace.com/d2l/home/388990
Join on Discord:
https://discord.com/channels/1230565799048122388/1374468690649786 15

SHIFTKEY ACADEMY

Assessment

Pass 70%

- **Attendance**
  - Attend at least **three** of the four sessions.
- **Weekly Practice Assignments (2.5%*4 = 10%)**
  - **Screenshot** of Leetcode success submission (including your **code**, along with its **time and space complexity** analysis). (Please read the **late policy** in syllabus!!!)
  - Practice questions will be posted after each lecture.
- **Soft Skills Assessment (20%)**
  - Submit a **video** explaining your solution for an algorithm question and analyze its time and space complexity (maximum 2 min).
  - Question will be posted on June 27
- **In-person Exam (70%)**
  - Algorithm analysis questions (5'' x5) + Algorithm design questions (15''x3).

SHIFTKEY ACADEMY

Reward

- Answering Questions & Brainstorming

Or

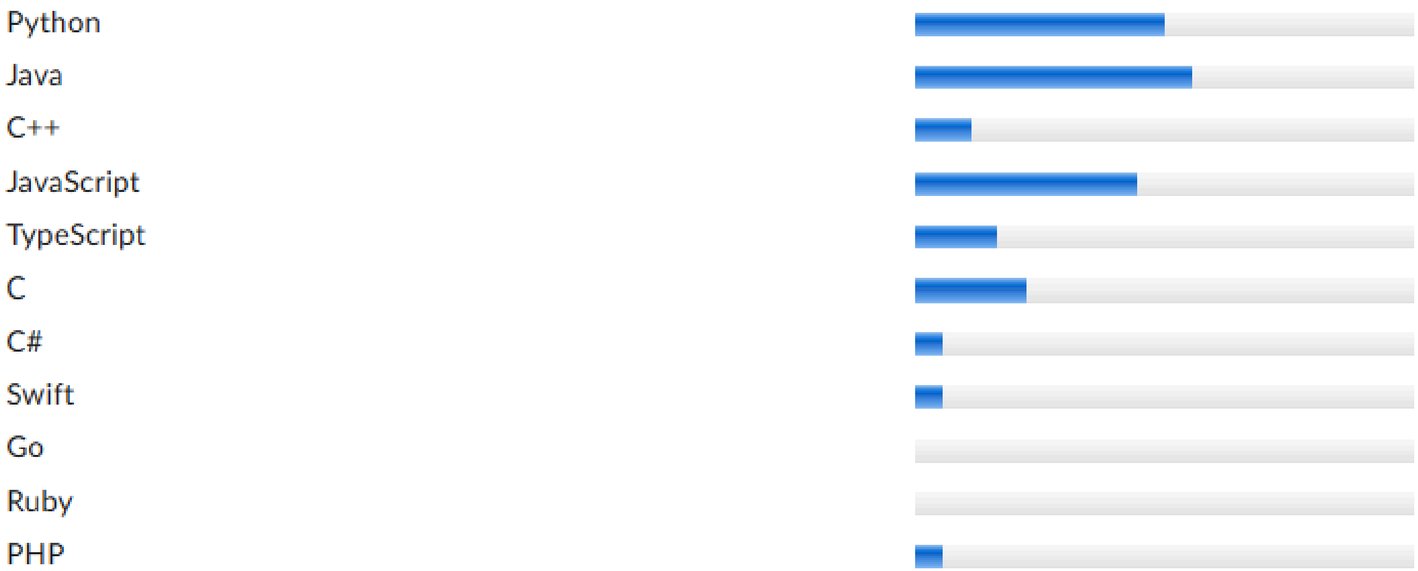
- Kahoot! Quiz (at the end of each lecture)

x 3

SHIFTKEY ACADEMY

About Survey

- **Programming Languages**



SHIFTKEY ACADEMY

About Survey

- **Questions**
  - Assignments?
  - Excercies related to technical interviews?
  - Learning path?
  - Advanced data structure?
  - Pace?
  - A post-course friendly DSA competition?

SHIFTKEY ACADEMY

Course Overview

Week 1

Algorithm Intro
Time/Space Complexity
Data Structures
Two pointers
Prefix Sum

Week 2

Recursion
Divide and Conquer
Sorting
Binary Search

Week 3

Math Problems
Bit Manipulation
Greedy
Dynamic Programing

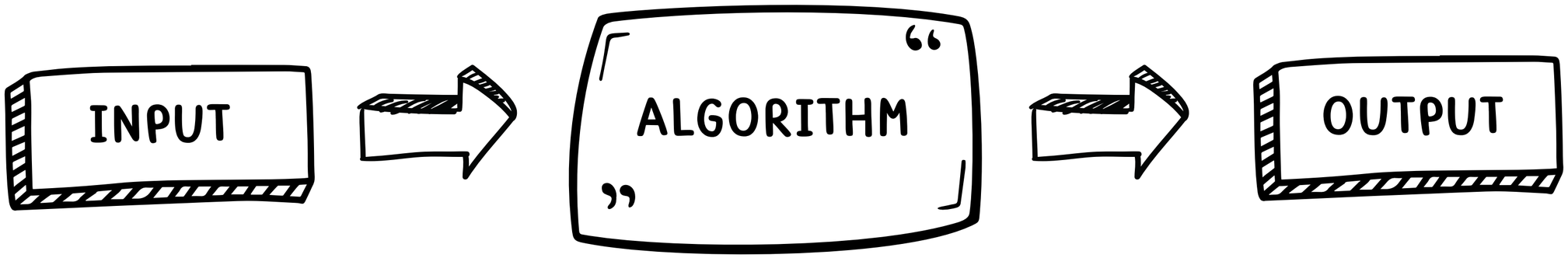
Week 4

Tree & Graph
BFS/DFS
Backtracking
Union-Find

SHIFTKEY ACADEMY

What is Algorithm?

A set of instructions for solving a problem or accomplishing a task.



SHIFTKEY ACADEMY

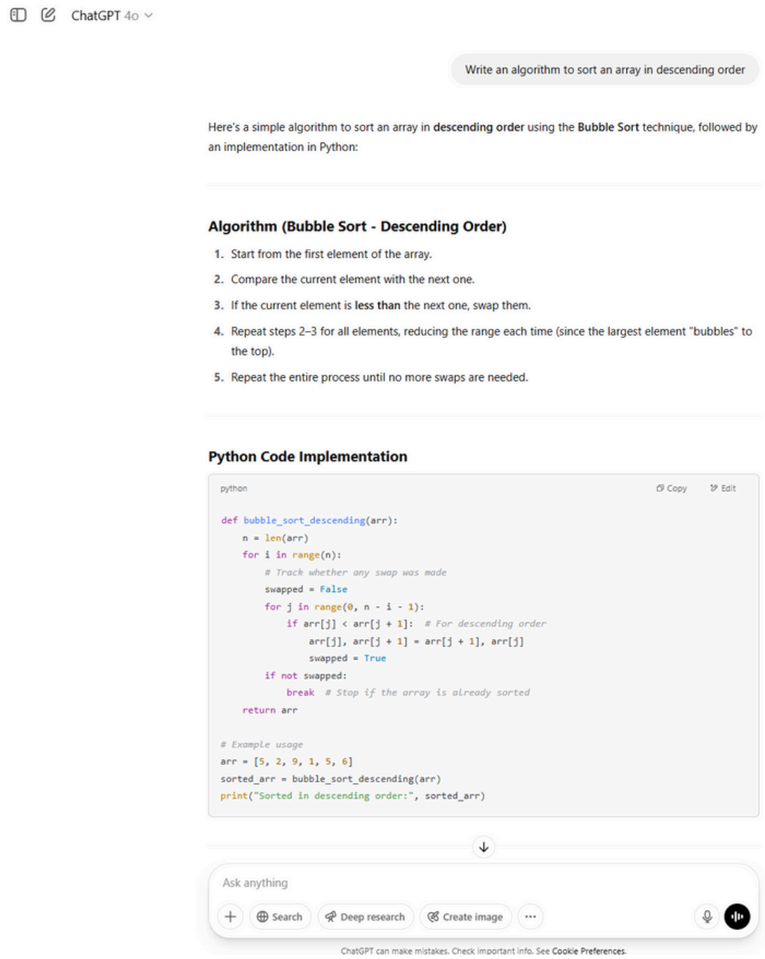
What is Leetcode?

- A leading online platform for coding practice and technical interview preparation.
- Over 3,500+ coding problems across data structures and algorithms
- Problems categorized by topic and difficulty (Easy / Medium / Hard)
- Supports multiple languages: Python, Java, C++, JavaScript, etc.
- Weekly/Biweekly contests and global leaderboards
- Detailed discussion forums and solution walkthroughs
- Paid premium tier for company-specific questions and mock interviews

LeetCode

SHIFTKEY ACADEMY

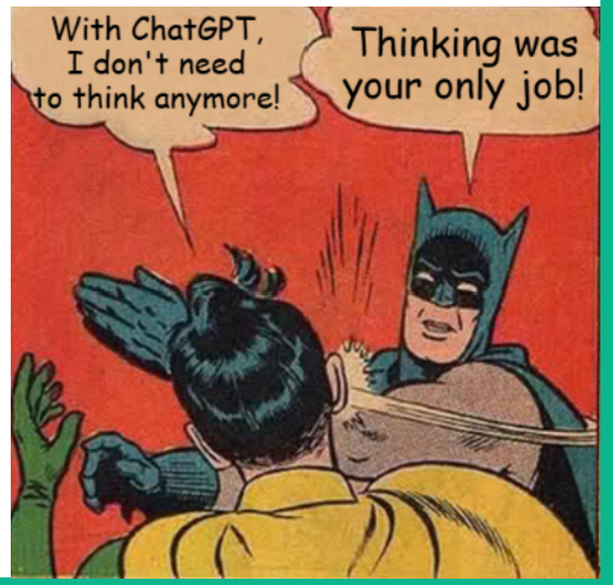
AI Killed Leetcode?



SHIFTKEY ACADEMY

AI Changed the Game

- LLMs solve problems because they were trained by millions of LeetCode solutions
- For untrained questions, AI might produce code that is incorrect or incomplete, has poor performance, or misses edge cases.
- You need to understand what AI generated.
- Practice problem solving, not just coding
- Learn to read, review, and critique AI’s code



SHIFTKEY ACADEMY

How AI Can Help

- Let AI craft a personalized roadmap/study plan for you.
- Brainstorm with AI using Socratic questioning to deepen understanding.
- Transform abstract concepts into clear visuals or animations via AI tools. (https://videotutor.io/ & https://thetawave.ai/)
- Explain your solutions with AI and get instant, targeted feedback on your solutions.

SHIFTKEY ACADEMY

Let’s Start!

SHIFTKEY ACADEMY

Two Key Goals in Algorithm Design

- **Correctness**
  - An algorithm must produce the correct outputs for all valid inputs (including edge cases).
- **Efficiency**
  - Time Efficiency
    → How fast does the algorithm run?
  - Space Efficiency
    → How much memory does it use?

SHIFTKEY ACADEMY

Edge Cases

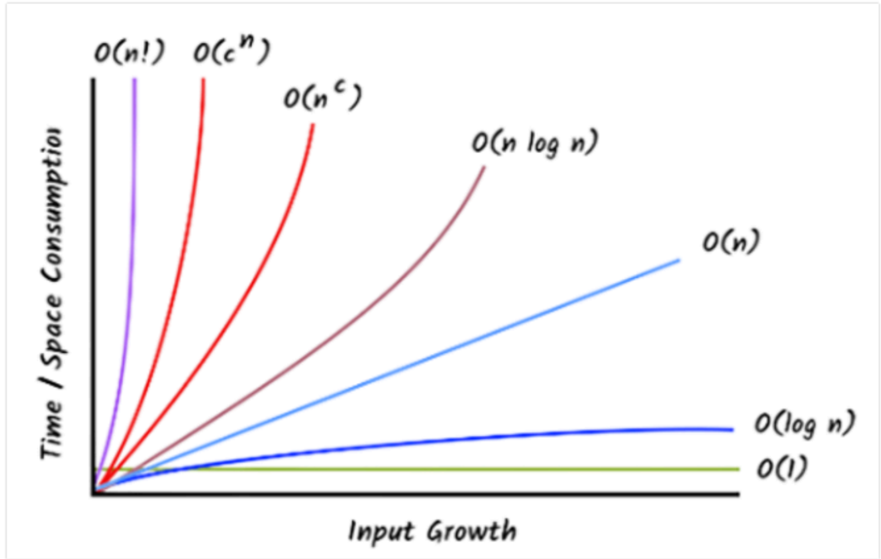
Type	Example
Empty Input	[], "", null
Single Element	[1], "a"
Max Input Size	n = 10^5 or more
All Elements Same	[3, 3, 3, 3]
Negative / Zero Values	[-1, 0, 2]
Sorted / Reverse Sorted	[1, 2, 3], [3, 2, 1]
No Solution Exists	No pair adds to target
Duplicates Present	[2, 2, 4] with target 4

SHIFTKEY ACADEMY

Time Complexity

- **Big-O Notation**
  - Mathematical notation to describe the upper bound of growth rate

Big-O Notation	Name	Example Algorithms
O(1)	Constant Time	Accessing an array element
O(log n)	Logarithmic Time	Binary search
O(n)	Linear Time	Single loop over an array
O(n log n)	Linearithmic Time	Merge sort
O(n^2)	Quadratic Time	Nested loops, Bubble sort
O(n^3)	Cubic Time	Triple nested loops
O(2^n)	Exponential Time	Brute-force subsets
O(n!)	Factorial Time	Permutation generation



SHIFTKEY ACADEMY

Time Complexity

- **For Example**

```
def example(nums):
    print("Start")                  # Line A
    for num in nums:                # Line B
        print(num)                  # Line C
    print("End")                    # Line D
```

Line	What it does	Number of operations
A	Prints once	1 (constant)
B	Loops through the array	n
C	Prints each item in the array	n
D	Prints once	1 (constant)

$$T(n) = 1+n+n+1$$
$$= 2+2n$$
$$= O(n)$$

SHIFTKEY ACADEMY

Space Complexity

Big-O Notation	Name	Example Scenarios
O(1)	Constant Space	Variable swap, pointer iteration
O(log n)	Logarithmic Space	Recursive binary search (call stack space)
O(n)	Linear Space	Copying an array, storing visited nodes in DFS
$O(n^2)$	Quadratic Space	2D matrices, all pairwise comparisons
$O(2^n)$	Exponential Space	Memoization of all subsets / binary combinations
O(n!)	Factorial Space	Storing all permutations

SHIFTKEY ACADEMY

Time VS Space Tradeoffs

- In algorithm design, improving time efficiency often requires more memory, and reducing space usage can slow down execution.
- **Prioritize time efficiency when:**
  - System has abundant memory
  - Performance is critical
- **Prioritize space efficiency when:**
  - Memory is limited
  - Working with large datasets

SHIFTKEY ACADEMY

Data Structures

SHIFTKEY ACADEMY

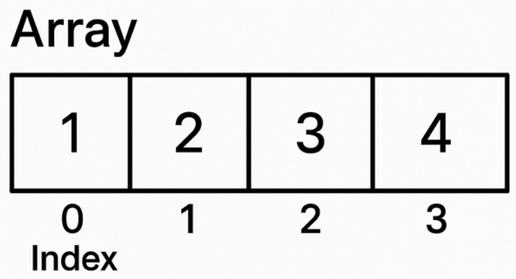
Array

A data structure that stores a collection of elements of the same data type, stored in contiguous memory locations, where each element can be accessed directly by using an index or position.

Access: O(1)

Search: O(n)

Insert/Delete: O(n)




```
nums = [1, 2, 3, 4]
for i in range(len(nums)):
    print(nums[i])
```

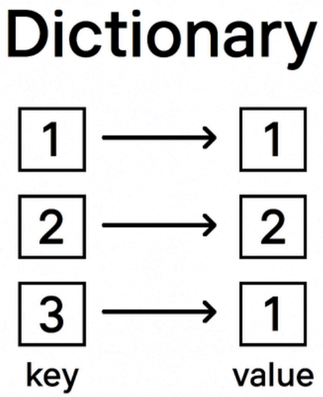
- In traditional languages (like JAVA and C++), arrays have a fixed size, but in many high-level languages (like Python), we use dynamic arrays that can automatically resize as needed.
- We can also use ArrayList in Java or Vector in C++ for dynamic arrays.

SHIFTKEY ACADEMY

HashMap (Dictionary)

A data structure that stores key-value pairs, allowing efficient retrieval of values based on their associated keys.

- Access: O(1)
- Insert/Delete: O(1)
- Worst-case (hash collisions): O(n)




```python
freq = {}
for num in [1, 2, 2, 3]:
    freq[num] = freq.get(num, 0) + 1
print(freq)
```

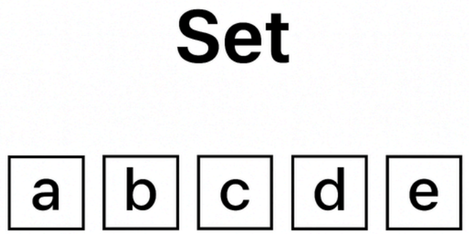
Java: HashMap Python: Dictionary JavaScript: Object or Map C#: Dictionary C++: unordered_map Ruby: Hash

SHIFTKEY ACADEMY

HashSet (Set)

A data structure that stores a collection of distinct (unique) elements without any particular order.

- Access: O(1)
- Insert/Delete: O(1)
- Worst-case (hash collisions): O(n)




```python
seen = set()
for char in "abcade":
    if char in seen:
        print("Duplicate found:", char)
    seen.add(char)
```

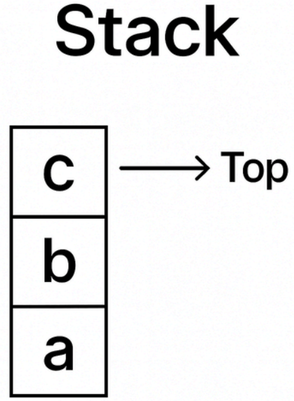
Java: HashSet Python: Set C#: HashSet C++: unordered_set Ruby: Set JavaScript: Set

SHIFTKEY ACADEMY

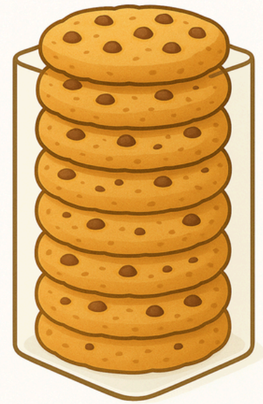
Stack

A linear data structure that follows the Last-In-First-Out (LIFO) principle, where elements are added and removed from the same end, called the "top" of the stack.

- Push: O(1)
- Pop: O(1)
- Peek: O(1)




```
stack = []
for char in "abc":
    stack.append(char)
while stack:
    print(stack.pop())
```

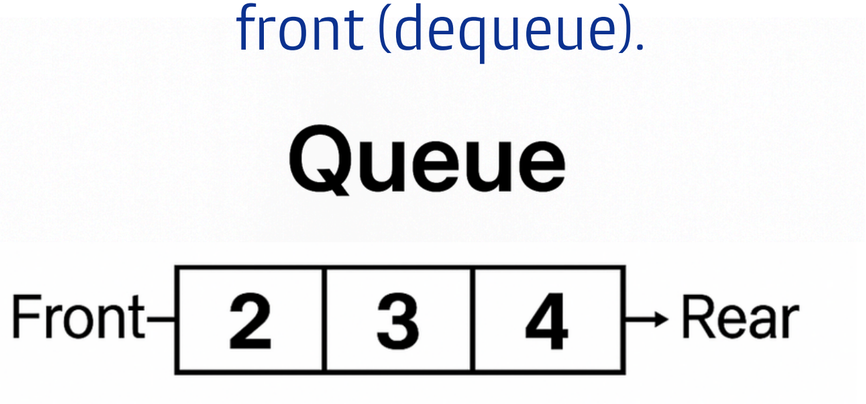


SHIFTKEY ACADEMY

Queue (Deque)

A linear data structure that follows the First-In-First-Out (FIFO) principle, where elements are added at the rear (enqueue) and removed from the front (dequeue).

- Enqueue: O(1)
- Dequeue: O(1)
- Insert Front: O(1)
- Delete Back: O(1)



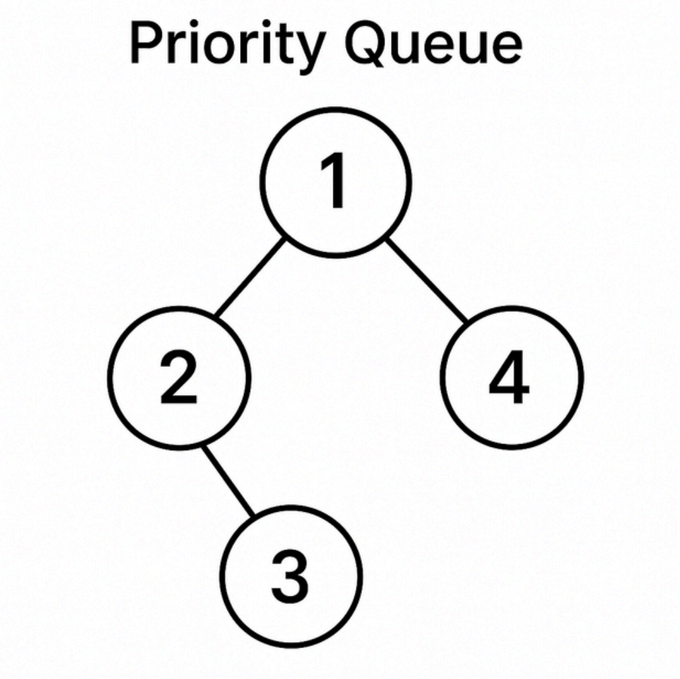

```
from collections import deque
queue = deque([1, 2, 3])
queue.append(4)
print(queue.popleft())
```

SHIFTKEY ACADEMY

Heap (Priority Queue)

A heap queue or priority queue is a data structure that allows us to quickly access the smallest (min-heap) or largest (max-heap) element.

- Insert: O(log n)
- Delete min/max: O(log n)
- Peek min/max: O(1)



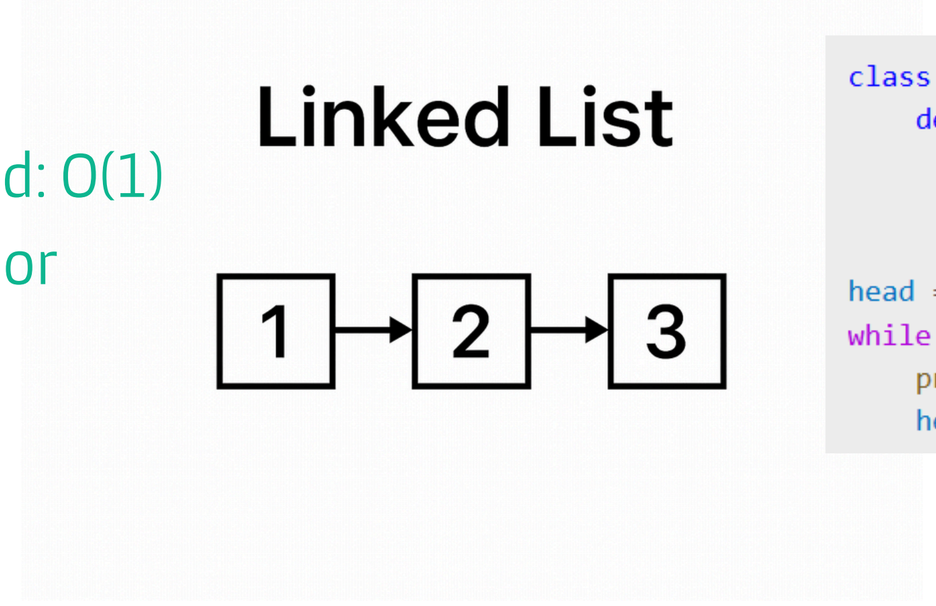

```python
import heapq
nums = [3, 1, 4, 2]
heapq.heapify(nums)
print(heapq.heappop(nums))
```

SHIFTKEY ACADEMY

Linked List

A linear data structure in which elements, called nodes, are connected using pointers, with each node containing a value and a reference to the next node.

- Access: O(n)
- Insert/Delete at head: O(1)
- Insert/Delete at tail or middle: O(n)




```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

head = ListNode(1, ListNode(2, ListNode(3)))
while head:
    print(head.val)
    head = head.next
```

SHIFTKEY ACADEMY

Practice Questions

SHIFTKEY ACADEMY

1. Two Sum

Easy | Topics | Companies | Hint

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.

You can return the answer in any order.

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]
```

Example 3:

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

SHIFTKEY ACADEMY

Brute Force

Check every pair of numbers to find the one that adds up to the target.

```python
#brute force
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        n = len(nums)
        for i in range(n-1):
            for j in range(i+1,n):
                if nums[i] + nums[j] == target:
                    return [i,j]
        return []
```

Time complexity: $O(n^2)$ Space complexity: $O(n)$

SHIFTKEY ACADEMY

Two-pass Hash Table

First store all values in a hash table, then scan again to find the complement.

```python
#two-pass hash table
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        n = len(nums)
        numMap = {}

        #build the hash table
        for i in range(n):
            numMap[nums[i]] = i

        #find the complement
        for i in range(n):
            complement = target - nums[i]
            if complement in numMap and numMap[complement] != i:
                return [numMap[complement], i]

        return []
```

Time complexity: O(n) Space complexity: O(n)

SHIFTKEY ACADEMY

One-pass Hash Table

Build the hash table while checking for complements in one pass.

```python
#one-pass hash table
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        n = len(nums)
        numMap = {}

        for i in range(n):
            complement = target - nums[i]
            if complement in numMap:
                return [numMap[complement], i]
            numMap[nums[i]] = i

        return []
```

Time complexity: O(n) Space complexity: O(n)

SHIFTKEY ACADEMY

167. Two Sum II - Input Array Is Sorted

Medium | Topics | Companies

Given a **1-indexed** array of integers `numbers` that is already ***sorted in non-decreasing order***, find two numbers such that they add up to a specific `target` number. Let these two numbers be `numbers[index1]` and `numbers[index2]` where `1 <= index1 < index2 <= numbers.length`.

Return *the indices of the two numbers,* `index1` *and* `index2`*,* ***added by one*** *as an integer array* `[index1, index2]` *of length 2.*

The tests are generated such that there is **exactly one solution**. You **may not** use the same element twice.

Your solution must use only constant extra space.

Example 1:

> **Input:** numbers = [2,7,11,15], target = 9
> **Output:** [1,2]
> **Explanation:** The sum of 2 and 7 is 9. Therefore, $index_1 = 1$, $index_2 = 2$. We return [1, 2].

Example 2:

> **Input:** numbers = [2,3,4], target = 6
> **Output:** [1,3]
> **Explanation:** The sum of 2 and 4 is 6. Therefore $index_1 = 1$, $index_2 = 3$. We return [1, 3].

Example 3:

> **Input:** numbers = [-1,0], target = -1
> **Output:** [1,2]
> **Explanation:** The sum of -1 and 0 is -1. Therefore $index_1 = 1$, $index_2 = 2$. We return [1, 2].

SHIFTKEY ACADEMY

Two Pointers

Use the two-pointer technique by initializing one pointer at the start and one at the end.

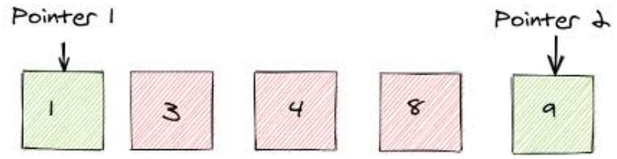
```python
# two-pointers
class Solution:
    def twoSum(self, numbers: List[int], target: int) -> List[int]:
        l, r = 0, len(numbers)-1
        while l < r:
            s = numbers[l] + numbers[r]
            if s == target:
                return [l+1, r+1]
            elif s < target:
                l += 1
            else:
                r -= 1
```

Time complexity: O(n) Space complexity: O(1)

SHIFTKEY ACADEMY

Two Pointers

- A technique where two indices traverse the data structure (typically an array or linked list) from different directions to solve a problem efficiently.



Pattern	Description	Example Use
Opposite Ends	One pointer starts from the beginning, the other from the end	Two Sum II (sorted array)
Forward Together	Both pointers move in the same direction, but with conditions	Remove Duplicates, Merge Sorted Arrays
Fast and Slow	One pointer moves faster to detect cycles	Linked List Cycle, Find Middle
Sliding Window	Two pointers define a dynamic range	Longest Substring Without Repeating Characters

SHIFTKEY ACADEMY

26. Remove Duplicates from Sorted Array

Solved

Easy | Topics | Companies | Hint

Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Example 1:

```
Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1
and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).
```

Example 2:

```
Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0,
1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).
```

SHIFTKEY ACADEMY

Two Pointers (forward together)

Use two pointers to overwrite duplicates: one scans the array, the other tracks the position to place the next unique element.

```python
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        #nums[:] = sorted(list(set(nums)))
        #return len(nums)
        index = 1
        for i in range(1,len(nums)):
            if nums[i] != nums[i-1]:
                nums[index] = nums[i]
                index += 1
        return index
```

Time complexity: O(n) Space complexity: O(1)

SHIFTKEY ACADEMY

141. Linked List Cycle

Solved

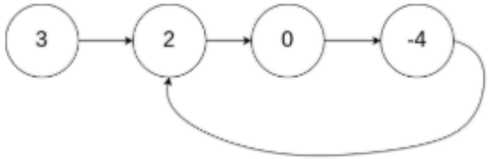
Easy | Topics | Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter.**

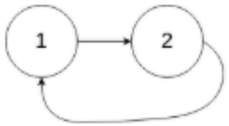
Return `true` *if there is a cycle in the linked list.* Otherwise, return `false`.

Example 1:




```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
```

Example 2:




```
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.
```

SHIFTKEY ACADEMY

Two Pointers (fast and slow)

Use the fast and slow pointer technique (Floyd's cycle detection) to check if a linked list contains a cycle.

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def hasCycle(self, head: Optional[ListNode]) -> bool:
        fast = head
        slow = head
        while fast and slow and fast.next:
            fast = fast.next.next
            slow = slow.next
            if fast == slow:
                return True
        return False
```

Time complexity: O(n)
Space complexity: O(1)

SHIFTKEY ACADEMY

3. Longest Substring Without Repeating Characters

Solved

Medium | Topics | Companies | Hint

Given a string `s`, find the length of the **longest** substring without duplicate characters.

Example 1:

```
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Example 3:

```
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.
```

SHIFTKEY ACADEMY

Two Pointers (sliding window)

Use the sliding window technique with a hash map to find the length of the longest substring without repeating characters.

```python
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        s1 = {}
        maxCount, left = 0, 0
        for right in range(len(s)):
            if s[right] not in s1 or s1[s[right]] < left:
                s1[s[right]] = right
                maxCount = max(maxCount,right-left+1)
            else:
                left = s1[s[right]]+1
                s1[s[right]] = right
        return maxCount
```

Time complexity: O(n)

Space complexity: O(k)

Where **k** is the size of the character set (e.g., 26 for lowercase letters, 128 or 256 for ASCII/Unicode). A hash map stores at most **k** characters.

SHIFTKEY ACADEMY

1732. Find the Highest Altitude

Solved

Easy | Topics | Companies | Hint

There is a biker going on a road trip. The road trip consists of `n + 1` points at different altitudes. The biker starts his trip on point `0` with altitude equal `0`.

You are given an integer array `gain` of length `n` where `gain[i]` is the **net gain in altitude** between points `i` and `i + 1` for all (`0 <= i < n`). Return *the* ***highest altitude*** *of a point.*

Example 1:

```
Input: gain = [-5,1,5,0,-7]
Output: 1
Explanation: The altitudes are [0,-5,-4,1,1,-6]. The highest is 1.
```

Example 2:

```
Input: gain = [-4,-3,-2,-1,4,3,2]
Output: 0
Explanation: The altitudes are [0,-4,-7,-9,-10,-6,-3,-1]. The highest is 0.
```

SHIFTKEY ACADEMY

Prefix Sum

It iteratively updates the current altitude using each gain value and keeps track of the maximum altitude reached during the journey.

```python
class Solution:
    def largestAltitude(self, gain: List[int]) -> int:
        highest, altitude = 0,0
        for i in gain:
            altitude += i
            highest = max(highest,altitude)
        return highest
```

Time complexity: O(n) Space complexity: O(1)

SHIFTKEY ACADEMY

Prefix Sum

- A technique to efficiently compute the sum of elements in a subarray. It transforms an array so that each position stores the cumulative total from the beginning.
- Given an array nums, the prefix sum array preSum is defined as:
  - preSum[0] = 0
  - preSum[i] = nums[0] + nums[1] + … + nums[i-1] (for $i \geq 1$)
- Quickly answer range sum queries in O(1) time
- Reduce redundant computation in nested loops

SHIFTKEY ACADEMY

724. Find Pivot Index

Solved

Easy | Topics | Companies | Hint

Given an array of integers `nums`, calculate the **pivot index** of this array.

The **pivot index** is the index where the sum of all the numbers **strictly** to the left of the index is equal to the sum of all the numbers **strictly** to the index's right.

If the index is on the left edge of the array, then the left sum is `0` because there are no elements to the left. This also applies to the right edge of the array.

Return *the **leftmost pivot index***. If no such index exists, return `-1`.

Example 1:

```
Input: nums = [1,7,3,6,5,6]
Output: 3
Explanation:
The pivot index is 3.
Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11
Right sum = nums[4] + nums[5] = 5 + 6 = 11
```

Example 2:

```
Input: nums = [1,2,3]
Output: -1
Explanation:
There is no index that satisfies the conditions in the problem statement.
```

SHIFTKEY ACADEMY

Prefix Sum

It uses a single pass with running left and right sums to find the index where both sides are equal.

```python
class Solution:
    def pivotIndex(self, nums: List[int]) -> int:
        left = 0
        right = sum(nums)
        for i in range(len(nums)):
            right -= nums[i]
            if right == left:
                return i
            left += nums[i]
        return -1
```

Time complexity: O(n) Space complexity: O(1)

SHIFTKEY ACADEMY

9. Palindrome Number

Easy | Topics | Companies | Hint

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

> **Input:** x = 121
> **Output:** true
> **Explanation:** 121 reads as 121 from left to right and from right to left.

Example 2:

> **Input:** x = -121
> **Output:** false
> **Explanation:** From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

Example 3:

> **Input:** x = 10
> **Output:** false
> **Explanation:** Reads 01 from right to left. Therefore it is not a palindrome.

Palindrome

An integer is a **palindrome** when it reads the same forward and backward.

For example, `121` is a palindrome while `123` is not.

SHIFTKEY ACADEMY

String Reverse

Convert the number to a string and check if it reads the same forward and backward.

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x<0:
            return False
        return str(x) == str(x)[::-1]
```

Time complexity: O(n) Space complexity: O(n)

SHIFTKEY ACADEMY

Math Reverse

Reverse the number mathematically and compare it with the original value.

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x<0:
            return False
        rev = 0
        org = x
        while x > 0:
            rev = rev*10 + x%10
            x //= 10
        return rev == org
```

Time complexity: $O(\log_{10}x)$ Space complexity: $O(1)$

x is an integer with d digits, then the loop runs d times, and $d \approx \log_{10}(x)$

SHIFTKEY ACADEMY

Quiz

SHIFTKEY ACADEMY

Reference

https://github.com/changgyhub/leetcode_101

https://github.com/halfrost/LeetCode-Go

https://github.com/krahets/hello-algo

SHIFTKEY ACADEMY

THANK YOU TO OUR PARTNERS

cognizant®

NOVA SCOTIA

DALHOUSIE UNIVERSITY | COMPUTER SCIENCE

SHIFTKEY ACADEMY

We want to hear from you!

Scan the QR Code to provide your feedback