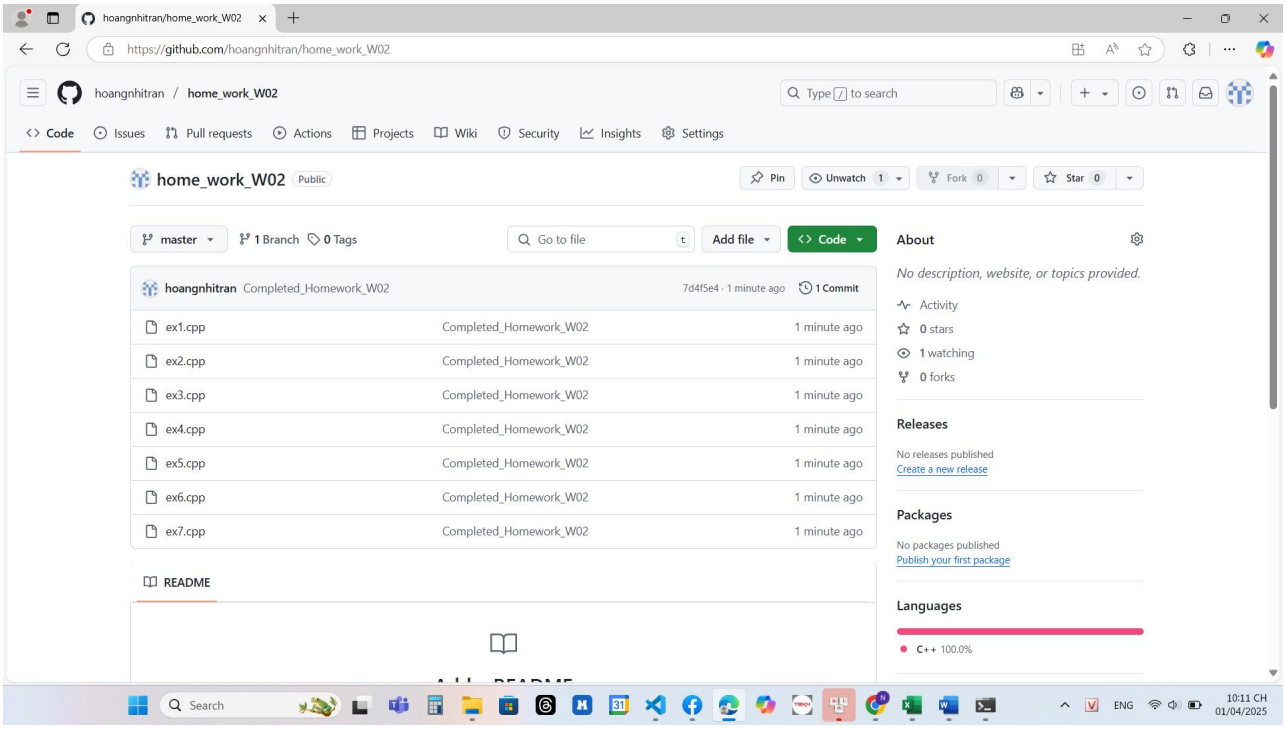
# Weekly Homework 2 Report

## 1 Github




## 2 Exercises

### Exercise 1

Given an array of $N$ integers and a target integer $K$, implement a program to find the first occurrence of $K$ in the array using the **Linear Search** algorithm. If $K$ is found, return its index (0-based). Otherwise, return -1.

**Example**

**Input**

5

1 3 5 7 9

5

**Output**

2

**Giải thích thuật toán**

Sử dụng **Linear Search** để tìm vị trí xuất hiện đầu tiên của số K trong mảng N phần tử.

- **Duyệt mảng**: Dùng vòng lặp for kiểm tra từng phần tử.
- **So sánh**: Nếu phần tử bằng K, trả về vị trí.

- **Không tìm thấy**: Trả về -1.

**Test Case**

| Input | Output |
|---|---|
| 5<br>1 3 5 7 9<br>5 | 2 |
| 4<br>10 20 30 40<br>25 | -1 |

## Exercise 2

Use **Linear Search with Sentinel** to solve Exercise 1.

**Giải thích thuật toán**

Sử dụng **Linear Search with Sentinel** để tìm vị trí xuất hiện đầu tiên của số K trong mảng.

- **Đặt Sentinel**: Gán K vào vị trí cuối cùng của mảng.
- **Duyệt mảng**: Sử dụng vòng lặp while để tìm K.
- **Kiểm tra kết quả**:
    - Nếu tìm thấy trước vị trí n, trả về chỉ số.
    - Nếu đến vị trí n, trả về -1 (không tồn tại trong mảng).

**Test Case**

| Input | Output |
|---|---|
| 5<br>1 3 5 7 9<br>5 | 2 |
| 4<br>10 20 30 40<br>25 | -1 |

## Exercise 3

Suppose an array of length *n* sorted in ascending order is **rotated** between 1 and *n* times. For example, the array

nums = [0,1,2,4,5,6,7]

might become:

- [4,5,6,7,0,1,2] if it was rotated 4 times.
- [0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that rotating an array

$$[a_0,a_1,a_2,\ldots,a_{n-1}]$$

one time results in the array

$$[a_{n-1},a_0,a_1,a_2,\ldots,a_{n-2}].$$

Given the sorted rotated array nums of **unique** elements, return *the minimum element of this array*.

Your algorithm must run in $O(\log n)$ time.

**Example**

**Input**

5

3 4 5 1 2

**Output**

1

**Explanation:** The original array was [1,2,3,4,5] rotated 3 times.

**Giải thích thuật toán**

Vì mảng đã được sắp xếp và được xoay k lần nên ta sử dụng **Binary Search** để tìm số bé hơn số ngay bên trái nó, số đó sẽ là số nhỏ nhất trong mảng.

**Các bước thực hiện**

1. **Khởi tạo biến** low = 0, high = n - 1.
2. **Lặp**:
    - Tính mid = low + (high - low) / 2.
    - Nếu a[mid] > a[low], phần nhỏ nhất nằm bên phải → low = mid + 1.
    - Ngược lại, phần nhỏ nhất nằm bên trái → high = mid - 1.
3. **Kết quả**: a[low] là phần tử nhỏ nhất.

**Test Case**

| Input | Output |
|---|---|
| 5<br>3 4 5 1 2 | 1 |
| 7<br>4 5 6 7 0 1 2 | 0 |

## Exercise 4

A conveyor belt has packages that must be shipped from one port to another within days days.

The $i^{th}$ package on the conveyor belt has a weight of weights[i]. Each day, we load the ship with packages on the conveyor belt (in the order given by weights). We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within days days.

**Example**

**Input**

10

1 2 3 4 5 6 7 8 9 10

5 # days = 5

**Output**

15

**Explanation:**

A ship capacity of 15 is the minimum to ship all the packages in 5 days like

1st day : 1 , 2 , 3 , 4 , 5

2nd day : 6 , 7

3rd day : 8

4th day : 9

5th day : 10

**Giải thích thuật toán**

Chương trình tìm tải trọng nhỏ nhất của tàu sao cho tất cả các gói hàng được vận chuyển trong số ngày quy định bằng cách sử dụng **Tìm kiếm nhị phân (Binary Search)**.

**Các bước thực hiện**

1. **Xác định phạm vi tìm kiếm**
    - Giá trị nhỏ nhất của tải trọng tàu là gói hàng nặng nhất (getMax(a, n)).
    - Giá trị lớn nhất là tổng trọng lượng tất cả các gói (sumWeight(a, n)).
2. **Dùng Binary Search With Condition để tìm tải trọng tối ưu**
    - Tính mid = (low + high) / 2 làm tải trọng thử nghiệm.

- Kiểm tra số ngày cần để vận chuyển tất cả hàng với tải trọng mid (getDayAtMaxWeight(a, n, mid)).
- Nếu số ngày đúng bằng days, lưu kết quả và thử tìm tải trọng nhỏ hơn (high = mid - 1).
- Nếu số ngày ít hơn days, tức là tải trọng quá lớn, thử tải trọng nhỏ hơn (high = mid - 1).
- Nếu số ngày nhiều hơn days, tức là tải trọng quá nhỏ, tăng tải trọng (low = mid + 1).

3. **Kết quả**: Giá trị nhỏ nhất của mid thỏa mãn điều kiện sẽ là tải trọng tối thiểu cần tìm.

**Test Case**

| Input | Output |
|---|---|
| 10<br>1 2 3 4 5 6 7 8 9 10<br>5 | 15 |
| 5<br>3 2 2 4 1<br>3 | 6 |

## Excercise 5

Given an array of positive integers nums and a positive integer target, return the *minimal length* of a **subarray** whose sum is greater than or equal to target. If there is no such subarray, return 0 instead.

**Example**

**Input:**

target = 7, nums = [2,3,1,2,4,3]

**Output:**

2

**Explanation:** The subarray [4,3] has the minimal length under the problem constraint.

**Giải thích thuật toán**

Chương trình sử dụng **Sliding Window** để tìm dãy con có tổng lớn hơn hoặc bằng target với độ dài nhỏ nhất.

1. Duyệt mảng bằng hai con trỏ left và right:
   - Tăng right để mở rộng cửa sổ và cập nhật sum.
   - Nếu sum ≥ target, thu hẹp cửa sổ bằng cách tăng left và cập nhật độ dài nhỏ nhất (min_length).

2. Kết quả: Trả về min_length nếu tìm thấy, ngược lại trả về 0.

**Test Case**

| Input | Output |
|---|---|
| 6<br>2 3 1 2 4 3<br>7 | 2 |
| 5<br>1 1 1 1 1<br>5 | 5 |

## Exercise 6

Given a sorted array of integers and a target sum, determine if there exist two numbers in the array whose sum equals the target.

**Example**

**Input:**

5

1 2 3 4 6

5

**Output:**

YES

**Giải thích thuật toán**

Sử dụng **Two Pointers**

1. Khởi tạo hai con trỏ:
    - left = 0
    - right = n - 1
2. Lặp đến khi hai con trỏ gặp nhau:
    - Nếu a[left] + a[right] == target, trả về "YES".
    - Nếu a[left] + a[right] < target, tăng left để tăng tổng.
    - Nếu a[left] + a[right] > target, giảm right để giảm tổng.
3. Nếu hết vòng lặp mà không tìm thấy cặp số, trả về "NO".

**Test Case**

| Input | Output |
|---|---|

| Input | Output |
|---|---|
| 5<br>1 2 3 4 6<br>5 | YES |
| 4<br>1 3 5 7<br>10 | YES |

## Exercise 7

Given an integer array nums, return all the triplets [nums[*i*],nums[*j*],nums[*k*]] where:

$$nums[i] + nums[j] + nums[k] = 0$$

and the indices *i,j,* and *k* are all distinct.

The output should **not** contain any duplicate triplets. You may return the output and the triplets in **any order**.

**Example**

**Input:**

nums = [-1,0,1,2,-1,-4] **Output:**

[[-1,-1,2], [-1,0,1]]

**Explanation:** The triplets [−1,−1,2] and [−1,0,1] satisfy the condition where their sum equals zero. No duplicate triplets are allowed.

**Giải thích thuật toán**:

Sử dụng **Two Pointers** kết hợp với việc sắp xếp mảng trước. Cụ thể, các bước thực hiện như sau:

1. Sắp xếp mảng theo thứ tự tăng dần bằng Insertion Sort.
2. Duyệt mảng với chỉ số chính giữa trong bộ ba, bỏ qua các phần tử trùng lặp nếu có 3 số bằng nhau liên tiếp trở lên.
3. Sử dụng hai con trỏ:
    - left bắt đầu từ vị trí i + 1 (vị trí ngay sau i).
    - right bắt đầu từ cuối mảng (n - 1).
    - Tính sum = a[i] + a[left] + a[right].
    - Nếu sum = 0, bộ ba hợp lệ. Sau đó, di chuyển con trỏ left và right để tiếp tục kiểm tra các cặp số tiếp theo, đồng thời đảm bảo không bị trùng lặp.
    - Nếu sum > 0, giảm con trỏ right để giảm sum.
    - Nếu sum < 0, tăng con trỏ left để tăng sum.

4. Kết thúc: Khi duyệt hết mảng mà không còn bộ ba nào thỏa mãn điều kiện, thuật toán kết thúc.

**Test Case**

| Input | Output |
|---|---|
| 6<br>-1 0 1 2<br>-1 -4 | [[-1, -1, 2][-1, 0, 1]] |
| 5<br>1 2 3 4<br>6 | |