2025-01-11 - Handout – Binary Search

Q1. Binary Search

Link: https://leetcode.com/problems/binary-search/description/

Given an array of integers *nums* which is sorted in ascending order, and an integer *target*, write a function to search *target* in *nums*. If *target* exists, then return its index. Otherwise, return -1.

You must write an algorithm with O(log n) runtime complexity.

Example 1	**Example 2**
Input: nums = [-1,0,3,5,9,12], target = 9 Output: 4	Input: nums = [-1,0,3,5,9,12], target = 2 Output: -1

Constraints:

- All the integers in nums are **unique**.
- nums is sorted in ascending order.

Q2. Find Peak Element

Link: https://leetcode.com/problems/find-peak-element/description/

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array *nums*, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

Example 1	**Example 2**
Input: nums = [1,2,3,1] **Output:** 2	**Input:** nums = [1,2,1,3,5,6,4] **Output:** 5

Constraints:

- 1 <= nums.length <= 1000
- -2^{31} <= nums[i] <= 2^{31} - 1
- nums[i] != nums[i + 1] for all valid i.

Q3. Capacity to Ship Packages within D Days

Link: https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/description/

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 1	**Example 2**	**Example 3**
Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 5 **Output:** 15	**Input:** weights = [3,2,2,4,1,4], days = 3 **Output:** 6	**Input:** weights = [1,2,3,1,1], days = 4 **Output:** 3

Constraints:

- 1 <= days <= weights.length <= 5 * 10^4
- 1 <= weights[i] <= 500

Q4. Split Array Largest Sum

Link: https://leetcode.com/problems/split-array-largest-sum/description/

Given an integer array *nums* and an integer *k*, split *nums* into *k* non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return *the minimized largest sum of the split*.

A **subarray** is a contiguous part of the array.

Example 1	**Example 2**
Input: nums = [7,2,5,10,8], k = 2 **Output:** 18	**Input:** nums = [1,2,3,4,5], k = 2 **Output:** 9

Constraints:

- 1 <= nums.length <= 1000
- 0 <= nums[i] <= 10^6
- 1 <= k <= min(50, nums.length)