2024-01-06 - Handout – Heaps

Q1. Kth Largest Element in an Array

Link: https://leetcode.com/problems/kth-largest-element-in-an-array/

Given an integer array `nums` and an integer `k`, return *the* `kth` *largest element in the array*.

```
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
```

```
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4
```

Q2. Sort Characters By Frequency

Link: https://leetcode.com/problems/sort-characters-by-frequency/description/

Given a string `s`, sort it in **decreasing order** based on the **frequency** of the characters. The **frequency** of a character is the number of times it appears in the string.

Return *the sorted string*. If there are multiple answers, return *any of them*.

```
Input: s = "tree"
Output: "eert"
```

```
Input: s = "Aabb"
Output: "bbAa"
Explanation: "bbaA" is also a valid answer, but "Aabb" is incorrect.
Note that 'A' and 'a' are treated as two different characters.
```

Q3. Distant Barcodes

Link: https://leetcode.com/problems/distant-barcodes/description/

In a warehouse, there is a row of barcodes, where the `ith` barcode is `barcodes[i]`.

Rearrange the barcodes so that no two adjacent barcodes are equal. You may return any answer, and it is guaranteed an answer exists.

```
Input: barcodes = [1,1,1,1,2,2,3,3]
Output: [1,3,1,3,1,2,1,2]
```

```
Input: barcodes = [1,1,1,2,2,2]
Output: [2,1,2,1,2,1]
```

Q4. Maximum Number of Eaten Apples

Link: https://leetcode.com/problems/maximum-number-of-eaten-apples/description/

There is a special kind of apple tree that grows apples every day for `n` days. On the i^{th} day, the tree grows `apples[i]` apples that will rot after `days[i]` days, that is on day `i + days[i]` the apples will be rotten and cannot be eaten. On some days, the apple tree does not grow any apples, which are denoted by `apples[i] == 0` and `days[i] == 0`.

You decided to eat **at most** one apple a day (to keep the doctors away). Note that you can keep eating after the first `n` days.

Given two integer arrays `days` and `apples` of length `n`, return *the maximum number of apples you can eat*.

```
Input: apples = [1,2,3,5,2], days = [3,2,1,4,2]
Output: 7
Explanation: You can eat 7 apples:
- On the first day, you eat an apple that grew on the first
day.
- On the second day, you eat an apple that grew on the second
day.
- On the third day, you eat an apple that grew on the second
day. After this day, the apples that grew on the third day
rot.
- On the fourth to the seventh days, you eat apples that grew
on the fourth day.
```

```
Input: apples = [3,0,0,0,0,2], days = [3,0,0,0,0,2]
Output: 5
```

Constraints:

- `n == apples.length == days.length`
- $1 <= n <= 2 * 10^4$
- $0 <= apples[i], days[i] <= 2 * 10^4$
- `days[i] = 0` if and only if `apples[i] = 0`.