Assignment 7

Due time: 04/10/2022, 11:59pm

Total credits: 100, 4 questions

Submission guide:

1. Create a folder and name it with the format FirstName_LastName_Aassigment7 for example Chunyu_Yuan_Assignment7

2. Inside the folder, you should have 4 java files

3. compress your file to .zip format and submit it to the blackboard,

4. if you have any question, please send email to cyuan1@gradcenter.cuny.edu

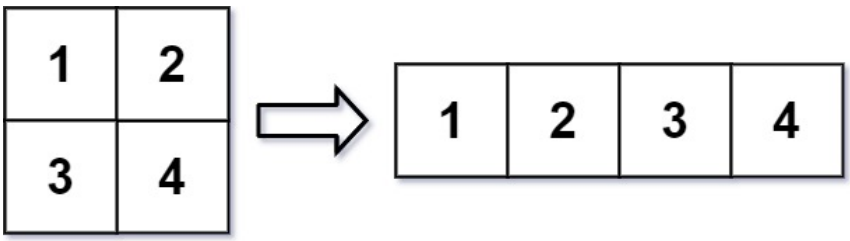
1. Matrix reshape (25 credits)

You are given an m x n matrix mat and two integers r and c representing the number of rows and the number of columns of the wanted reshaped matrix.

The reshaped matrix should be filled with all the elements of the original matrix in the same row-traversing order as they were.

If the reshape operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

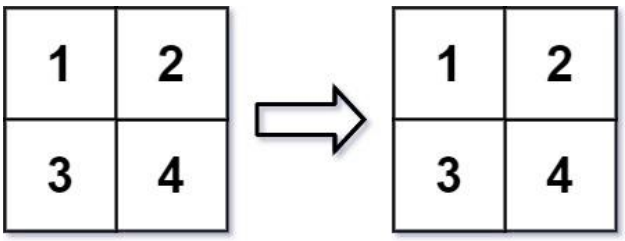
Example 1:



Input: mat = [[1,2],[3,4]], r = 1, c = 4

Output: [[1,2,3,4]]

Example 2:



Input: mat = [[1,2],[3,4]], r = 2, c = 4

Output: [[1,2],[3,4]]

Constraints:

- m == mat.length
- n == mat[i].length
- 1 <= m, n <= 100
- -1000 <= mat[i][j] <= 1000
- 1 <= r, c <= 300

```
class Solution {
   public int[][] matrixReshape(int[][] mat, int r, int c) {
    // your methods
   }
}
```

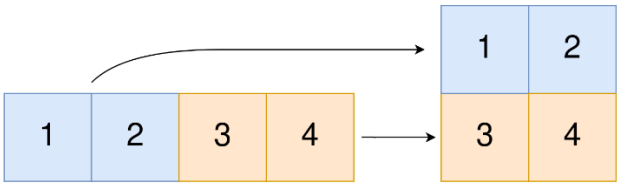
2. one dimension array into two dimensions array (25 credits)

You are given a 0-indexed 1-dimensional (1D) integer array original, and two integers, m and n. You are tasked with creating a 2-dimensional (2D) array with m rows and n columns using all the elements from original.

The elements from indices 0 to n - 1 (inclusive) of original should form the first row of the constructed 2D array, the elements from indices n to 2 * n - 1 (inclusive) should form the second row of the constructed 2D array, and so on.

Return an m x n 2D array constructed according to the above procedure, or an empty 2D array if it is impossible.

Example 1:



Input: original = [1,2,3,4], m = 2, n = 2

Output: [[1,2],[3,4]]

Explanation: The constructed 2D array should contain 2 rows and 2 columns.

The first group of n=2 elements in original, [1,2], becomes the first row in the constructed 2D array.

The second group of n=2 elements in original, [3,4], becomes the second row in the constructed 2D array.

Example 2:

Input: original = [1,2,3], m = 1, n = 3

Output: [[1,2,3]]

Explanation: The constructed 2D array should contain 1 row and 3 columns. Put all three elements in original into the first row of the constructed 2D array.

Example 3:

Input: original = [1,2], m = 1, n = 1

Output: []

Explanation: There are 2 elements in original. It is impossible to fit 2 elements in a 1x1 2D array, so return an empty 2D array.

Constraints:

- $1 <= original.length <= 5 * 10^4$
- $1 <= original[i] <= 10^5$
- $1 <= m, n <= 4 * 10^4$

```
class Solution {
  public int[][] construct2DArray(int[] original, int m, int n) {
      // your methods
  }
}
```

3. Top K Frequent Elements (25 credits)

Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.

Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2

Output: [1,2]
```

Example 2:

```
Input: nums = [1], k = 1

Output: [1]
```

Constraints:

- 1 <= nums.length <= 10^5
- k is in the range [1, the number of unique elements in the array].
- It is guaranteed that the answer is unique.

```
class Solution {

    public int[] topKFrequent(int[] nums, int k) {

        // your methods

    }

}
```

4. Top K Frequent Words (25 credits)

Given an array of strings words and an integer k, return *the* k *most frequent strings*.

Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by their lexicographical order.

Example 1:

Input: words = ["i","love","leetcode","i","love","coding"], k = 2

Output: ["i","love"]

Explanation: "i" and "love" are the two most frequent words.

Note that "i" comes before "love" due to a lower alphabetical order.

Example 2:

Input: words = ["the","day","is","sunny","the","the","the","sunny","is","is"], k = 4

Output: ["the","is","sunny","day"]

Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrence being 4, 3, 2 and 1 respectively.

Constraints:

- 1 <= words.length <= 500
- 1 <= words[i] <= 10
- words[i] consists of lowercase English letters.
- k is in the range [1, The number of **unique** words[i]]

```
class Solution {
    public ArrayList<String> topKFrequent(String[] words, int k) {
            // your methods
    }
}
```