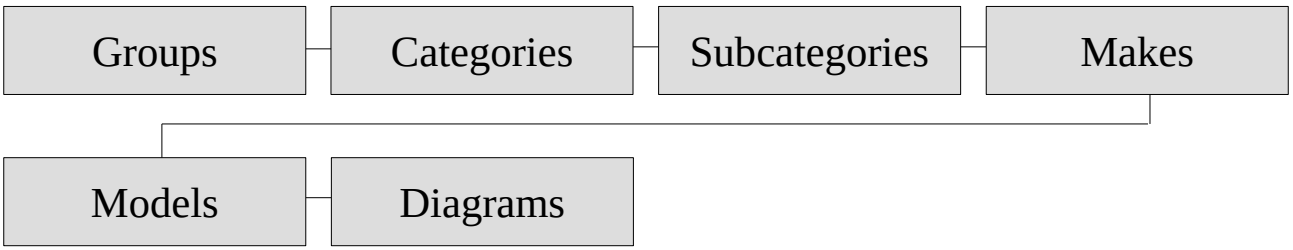
Test 1: Find all Combinations of Tags in a String

Test duration: 1 hour

AB

Outline

You are working on an online store with over 100K+ products. The store has the following hierarchy in order to keep the products organized:



(Fig.1)

This hierarchy was built using a tagging system. Each product is tagged with one or more of the prefixes above. For example, a product can have the following tags:

- **Group_**A
- **Category_**B
- **Subcategory_**C

Where (A) is the **group**, (B) is the **category**, and (C) is the **subcategory**.

Now, after adding tags to all the products, **we need to link them with the collection pages.** A collection is a page on the store that includes a **list of products filtered by their tags**.

Each collection has a **single-line text metafield** that can be **used to filter the products by their tags**. This single-line text metafield is a text box where one or more of the tags can be added to. This metafield is used internally to find all the products with the specified tags.

Example 1:

A collection with the following metafield value: **Group_Electric-Pallet-Jack-Parts-Make_BT-Prime-Mover**

Will include a list of products that have the tags: **Group_Electric-Pallet-Jack-Parts** and **Make_BT-Prime-Mover**

Example 2:

A collection with the following metafield value: **Group_Electric-Pallet-Jack-Parts-Category_Switches-Subcategory_Ignition-Switch**

Will include a list of products that have the tags: **Group_Electric-Pallet-Jack-Parts**, **Category_Switches**, and **Subcategory_Ignition-Switch**

Instructions

Write a function that returns an array of arrays of combinations from a single-line string.

Example 3:

Input:

findCombinationsFromText('Group_Electric-Pallet-Jack-Parts, Category_Switches, Subcategory_Ignition-Switch')

Output:

```
▼ (3) [Array(3), Array(2), Array(1)]
  ▶ 0: (3) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches', 'Subcategory_Ignition-Switch']
  ▶ 1: (2) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches']
  ▶ 2: ['Group_Electric-Pallet-Jack-Parts']
    length: 3
```

Here we passed the string 'Group_Electric-Pallet-Jack-Parts, Category_Switches, Subcategory_Ignition-Switch' to the function findCombinationsFromText and we got an array of 3 arrays, each array is a combination of tags.

Notice how the **first** combination of tags included the **Group, Category, and the Subcategory**, while the **second** combination of tags included only the **Group and the Category**, and the **third** included only the **Group**.

Example 4:

Input:

findCombinationsFromText('--Group_Electric-Pallet-Jack-Parts, Category_Switche@%s-!! Subcategory_Ignition-Switch))@!%')

Output:

```
▼ (3) [Array(3), Array(2), Array(1)]
  ▶ 0: (3) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches', 'Subcategory_Ignition-Switch']
  ▶ 1: (2) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches']
  ▶ 2: ['Group_Electric-Pallet-Jack-Parts']
    length: 3
```

Here we passed the string '--Group_Electric-Pallet-Jack-Parts, Category_Switche@%s-!! Subcategory_Ignition-Switch))@!%' to the function findCombinationsFromText and we got the same output as shown in Example 3.

Notice how the string contained **invalid characters** like the **two hyphens at the start**, the separator between the Group and the Category is a **comma not a hyphen**, and there are **invalid characters in the middle and the end of the string**. This did not affect the output.

Example 5:

Input:

findCombinationsFromText('Category_Switches-Group_Electric-Pallet-Jack-Parts-Subcategory_Ignition-Switch')

Output:

```
(3) [Array(3), Array(2), Array(1)]
  0: (3) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches', 'Subcategory_Ignition-Switch']
  1: (2) ['Group_Electric-Pallet-Jack-Parts', 'Category_Switches']
  2: ['Group_Electric-Pallet-Jack-Parts']
  length: 3
```

Here we passed the string Category_Switches-Group_Electric-Pallet-Jack-Parts-Subcategory_Ignition-Switch' to the function findCombinationsFromText and we didn't get a different output from Examples 3 and 4.

Notice how the string contains a Category that is followed by a Group. According to the hierarchy in (Fig.1), this is invalid. Group should always come before Category. This applies to all the other types. It should always be **Group → Category → Subcategory → Make → Model → Diagram**

Example 6:

Input:

findCombinationsFromText('Group_Tools-Hardware-Category_Roll-Pin-Make_Atlas')

Output:

```
(3) [Array(3), Array(2), Array(1)]
  0: (3) ['Group_Tools-Hardware', 'Category_Roll-Pin', 'Make_Atlas']
  1: (2) ['Group_Tools-Hardware', 'Category_Roll-Pin']
  2: ['Group_Tools-Hardware']
  length: 3
```

Here we have Make_Atlas without any Subcategory before it.

The order of the tags should match with their positions on the hierarchy. So, **Make should always come after Subcategory**, but **if the Subcategory does not exist**, **Make comes after the Category** and so on.

Note: If the string did not include anything but Make, Model, and Diagram, the order becomes **Make → Model → Diagram**.

Example 7:

Input:

findCombinationsFromText('Group_Tools-Hardware-Category_Roll-Pin-Make_Atlas-Group_Test')

Output: []

Here we passed the string 'Group_Tools-Hardware-Category_Roll-Pin-Make_Atlas-Group_Test' to the function findCombinationsFromText and we got an empty array.

The reason is that **there are two occurrences of the Group tag**. This is considered as **invalid** and should not be processed. This applies to all the other types.

Example 8:

Input:

findCombinationsFromText('Group_Tools-Hardware-Category_Roll-Pin-Make_Atlas-WrongPrefix_Test')

Output: []

Here we passed the string 'Group_Tools-Hardware-Category_Roll-Pin-Make_Atlas-WrongPrefix_Test' to the function findCombinationsFromText and we got an empty array.

The reason is that 'WrongPrefix_Test' is an **invalid tag**. It's not a Group, Category, Subcategory, Make, Model, nor a Diagram.

Note: Misspelled prefixes are also considered invalid.

Other Notes

- Ensure the validity of the tags returned by the function. For example, they should not include invalid characters.
- Tags can be separated by a comma, a hyphen or any other separator. The function should work with all of them.
- If the text string does not contain any valid prefixes or contains duplicate prefixes or invalid tags, the function should return an empty array.
- Removing invalid characters from the string should not result in '--' (string containing double hyphens).
- Result should not include spaces.

Good Luck!

More Examples

Input:

findCombinationsFromText('Group_Tools-Hardware-Category_Roll-Pin-Make_U-Line-Model_H-1193')

Output:

```
(4) [Array(4), Array(3), Array(2), Array(1)]
  0: (4) ['Group_Tools-Hardware', 'Category_Roll-Pin', 'Make_U-Line', 'Model_H-1193']
  1: (3) ['Group_Tools-Hardware', 'Category_Roll-Pin', 'Make_U-Line']
  2: (2) ['Group_Tools-Hardware', 'Category_Roll-Pin']
  3: ['Group_Tools-Hardware']
  length: 4
```

Input:

findCombinationsFromText('Group_Tools-Hardware-Category_Roll-Pin-Make_Multiton-Model_J')

Output:

```
(4) [Array(4), Array(3), Array(2), Array(1)]
  0: (4) ['Group_Tools-Hardware', 'Category_Roll-Pin', 'Make_Multiton', 'Model_J']
  1: (3) ['Group_Tools-Hardware', 'Category_Roll-Pin', 'Make_Multiton']
  2: (2) ['Group_Tools-Hardware', 'Category_Roll-Pin']
  3: ['Group_Tools-Hardware']
  length: 4
```

Input:

findCombinationsFromText('Make_Atlas-Model_Zenith-Type9-Diagram_Frame')

Output:

```
(3) [Array(3), Array(2), Array(1)]
  0: (3) ['Make_Atlas', 'Model_Zenith-Type9', 'Diagram_Frame']
  1: (2) ['Make_Atlas', 'Model_Zenith-Type9']
  2: ['Make_Atlas']
  length: 3
```

Input:

findCombinationsFromText('Group_Tools-&-Hardware')

Output:

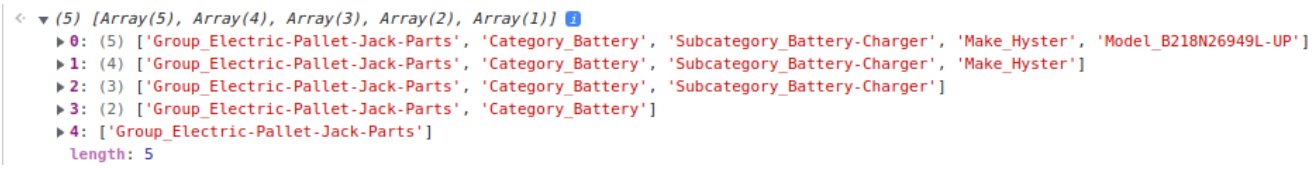
```
[Array(1)]
  0: ['Group_Tools-Hardware']
  length: 1
```

(Notice how the ampersand was removed without resulting in double hyphens)

Input:

findCombinationsFromText('Group_Electric-Pallet-Jack-Parts-Category_Battery-Subcategory_Battery-Charger-Make_Hyster-Model_B218N26949L-UP')

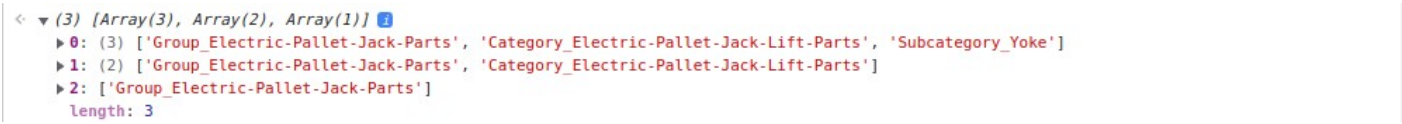
Output:



Input:

findCombinationsFromText('Group_Electric-Pallet-Jack-Parts-Category_Electric-Pallet-Jack-Lift-Parts-Subcategory_Yoke')

Output:



Input:

findCombinationsFromText('Group_Industrial-Seal-Kits-Make_Yutani')

Output:

Input:

findCombinationsFromText('Make_Atlas-Model_Zenith-Type9')

Output:

Input:

findCombinationsFromText('Group_Wheel-Bearings-Category_Bearing-Wiper')

Output: