Contents

1. Backspace String Compare 2
2. Shortest Distance to a Character 3
3. Most Common Word 3
4. Number of Lines to Write String 4
5. Rotated Digits 5
6. Sentence Similarity 6
7. Detect Capital 7
8. Number of Segments in a String 7
9. First Unique Character in a String 8
9. Remove Linked List Elements 8
10. Capacity to Ship Packages Within D Days 9
11. Most Profit Assigning Work 11
12. Number of Lines To Write String 11

1. Backspace String Compare

Given two strings S and T, return if they are equal when both are typed into empty text editors. # means a backspace character.

Example 1:

```
Input: S = "ab#c", T = "ad#c"

Output: true

Explanation: Both S and T become "ac".
```

Example 2:

```
Input: S = "ab##", T = "c#d#"

Output: true

Explanation: Both S and T become "".
```

Example 3:

```
Input: S = "a##c", T = "#a#c"

Output: true

Explanation: Both S and T become "c".
```

Example 4:

```
Input: S = "a#c", T = "b"

Output: false
```

```
Explanation: S becomes "c" while T becomes "b".
```

Note:

1. `1 <= S.length <= 200`
2. `1 <= T.length <= 200`
3. `S` and `T` only contain lowercase letters and `'#'` characters.

2. Shortest Distance to a Character

Given a string `S` and a character `C`, return an array of integers representing the shortest distance from the character `C` in the string.

Example 1:

```
Input: S = "loveleetcode", C = 'e'

Output: [3, 2, 1, 0, 1, 0, 0, 1, 2, 2, 1, 0]
```

Note:

1. `S` string length is in `[1, 10000].`
2. `C` is a single character, and guaranteed to be in string `S`.
3. All letters in `S` and `C` are lowercase.

3. Most Common Word

Given a paragraph and a list of banned words, return the most frequent word that is not in the list of banned words. It is guaranteed there is at least one word that isn't banned, and that the answer is unique.

Words in the list of banned words are given in lowercase, and free of punctuation. Words in the paragraph are not case sensitive. The answer is in lowercase.

```
Example:
Input:
paragraph = "Bob hit a ball, the hit BALL flew far after it was hit."
banned = ["hit"]
Output: "ball"
Explanation:
"hit" occurs 3 times, but it is a banned word.
"ball" occurs twice (and no other word does), so it is the most frequent non-banned w
ord in the paragraph.
Note that words in the paragraph are not case sensitive,
that punctuation is ignored (even if adjacent to words, such as "ball,"),
and that "hit" isn't the answer even though it occurs more because it is banned.
```

Note:

- `1 <= paragraph.length <= 1000`.
- `1 <= banned.length <= 100`.
- `1 <= banned[i].length <= 10`.
- The answer is unique, and written in lowercase (even if its occurrences in `paragraph` may have uppercase symbols, and even if it is a proper noun.)
- `paragraph` only consists of letters, spaces, or the punctuation symbols `!?',;.`
- Different words in `paragraph` are always separated by a space.
- There are no hyphens or hyphenated words.
- Words only consist of letters, never apostrophes or other punctuation symbols.

4. Number of Lines to Write String

We are to write the letters of a given string `S`, from left to right into lines. Each line has maximum width 100 units, and if writing a letter would cause the width of the line to exceed 100 units, it is written on the next line. We are given an array `widths`, an array where widths[0] is the width of 'a', widths[1] is the width of 'b', ..., and widths[25] is the width of 'z'.

Now answer two questions: how many lines have at least one character from `S`, and what is the width used by the last such line? Return your answer as an integer list of length 2.

```
Example :
```

```
Input:
widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
10]
S = "abcdefghijklmnopqrstuvwxyz"
Output: [3, 60]
Explanation:
All letters have the same length of 10. To write all 26 letters,
we need two full lines and one line with 60 units.
```

```
Example :
Input:
widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
10]
S = "abcdefghijklmnopqrstuvwxyz"
Output: [3, 60]
Explanation:
All letters have the same length of 10. To write all 26 letters,
we need two full lines and one line with 60 units.
```

Note:

- The length of `S` will be in the range [1, 1000].
- `S` will only contain lowercase letters.
- `widths` is an array of length `26`.
- `widths[i]` will be in the range of `[2, 10]`.

5. Rotated Digits

X is a good number if after rotating each digit individually by 180 degrees, we get a valid number that is different from X. A number is valid if each digit remains a digit after rotation. 0, 1, and 8 rotate to themselves; 2 and 5 rotate to each other; 6 and 9 rotate to each other, and the rest of the numbers do not rotate to any other number.

Now given a positive number `N`, how many numbers X from `1` to `N` are good?

```
Example:
Input: 10
Output: 4
Explanation:
There are four good numbers in the range [1, 10] : 2, 5, 6, 9.
Note that 1 and 10 are not good numbers, since they remain unchanged after rotating.
```

Note:

- `N` will be in range `[1, 10000]`.

6. Sentence Similarity

Given two sentences `words1, words2` (each represented as an array of strings), and a list of similar word pairs `pairs`, determine if two sentences are similar.

For example, "great acting skills" and "fine drama talent" are similar, if the similar word pairs are `pairs = [["great", "fine"], ["acting","drama"], ["skills","talent"]]`.

Note that the similarity relation is not transitive. For example, if "great" and "fine" are similar, and "fine" and "good" are similar, "great" and "good" are **not** necessarily similar.

Also, a word is always similar with itself. For example, the sentences `words1 = ["great"], words2 = ["great"], pairs = []` are similar, even though there are no specified similar word pairs.

Finally, sentences can only be similar if they have the same number of words. So a sentence like `words1 = ["great"]` can never be similar to `words2 = ["doubleplus","good"]`.

Note:

- The length of `words1` and `words2` will not exceed `1000`.
- The length of `pairs` will not exceed `2000`.
- The length of each `pairs[i]` will be `2`.
- The length of each `words[i]` and `pairs[i][j]` will be in the range `[1, 20]`

7. Detect Capital

Given a word, you need to judge whether the usage of capitals in it is right or not.

We define the usage of capitals in a word to be right when one of the following cases holds:

1. All letters in this word are capitals, like "USA".
2. All letters in this word are not capitals, like "leetcode".
3. Only the first letter in this word is capital if it has more than one letter, like "Google".

Otherwise, we define that this word doesn't use capitals in a right way.

Example 1:

```
Input: "USA"

Output: True
```

Example 2:

```
Input: "FlaG"

Output: False
```

Note: The input will be a non-empty word consisting of uppercase and lowercase Latin letters.

8. Number of Segments in a String

Count the number of segments in a string, where a segment is defined to be a contiguous sequence of non-space characters.

For example,

Input: "Hello, my name is John"

Output: 5

9. First Unique Character in a String

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

Examples:

```
s = "leetcode"

return 0.



s = "loveleetcode",

return 2.
```

Note: You may assume the string contain only lowercase letters.

9. Remove Linked List Elements

Remove all elements from a linked list of integers that have value val.

Example
Given: 1 --> 2 --> 6 --> 3 --> 4 --> 5 --> 6, val = 6
Return: 1 --> 2 --> 3 --> 4 --> 5

Python:

```
# Definition for singly-linked list.
# class ListNode:
```

```
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    # @param {ListNode} head
    # @param {integer} val
    # @return {ListNode}
    def removeElements(self, head, val):
        dummy = ListNode(0)
        dummy.next = head
        pre, cur = dummy, head
        while cur:
            if cur.val == val:
                pre.next = cur.next
            else:
                pre = cur
            cur = cur.next
        return dummy.next
```

10. Capacity to Ship Packages Within D Days

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

Example 1:

```
Input: weights = [1,2,3,4,5,6,7,8,9,10], D = 5
Output: 15
Explanation:
A ship capacity of 15 is the minimum to ship all the packages in 5 days like this:
```

```
1st day: 1, 2, 3, 4, 5
2nd day: 6, 7
3rd day: 8
4th day: 9
5th day: 10


Note that the cargo must be shipped in the order given, so using a ship of capacity 1
4 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10)
is not allowed.
```

Example 2:

```
Input: weights = [3,2,2,4,1,4], D = 3
Output: 6
Explanation:
A ship capacity of 6 is the minimum to ship all the packages in 3 days like this:
1st day: 3, 2
2nd day: 2, 4
3rd day: 1, 4
```

Example 3:

```
Input: weights = [1,2,3,1,1], D = 4
Output: 3
Explanation:
1st day: 1
2nd day: 2
3rd day: 3
4th day: 1, 1
```

Note:

1. `1 <= D <= weights.length <= 50000`
2. `1 <= weights[i] <= 500`

11. Most Profit Assigning Work

We have jobs: `difficulty[i]` is the difficulty of the `i`th job, and `profit[i]` is the profit of the `i`th job.

Now we have some workers. `worker[i]` is the ability of the `i`th worker, which means that this worker can only complete a job with difficulty at most `worker[i]`.

Every worker can be assigned at most one job, but one job can be completed multiple times.

For example, if 3 people attempt the same job that pays $1, then the total profit will be $3. If a worker cannot complete any job, his profit is $0.

What is the most profit we can make?

Example 1:

```
Input: difficulty = [2,4,6,8,10], profit = [10,20,30,40,50], worker = [4,5,6,7]
Output: 100
Explanation: Workers are assigned jobs of difficulty [4,4,6,6] and they get profit of
 [20,20,30,30] seperately.
```

Notes:

- `1 <= difficulty.length = profit.length <= 10000`
- `1 <= worker.length <= 10000`
- `difficulty[i], profit[i], worker[i]` are in range `[1, 10^5]`

12. Number of Lines To Write String

We are to write the letters of a given string `S`, from left to right into lines. Each line has maximum width 100 units, and if writing a letter would cause the width of the line to exceed 100 units, it is written on the next line. We are given an array `widths`, an array where widths[0] is the width of 'a', widths[1] is the width of 'b', ..., and widths[25] is the width of 'z'.

Now answer two questions: how many lines have at least one character from `S`, and what is the width used by the last such line? Return your answer as an integer list of length 2.

```
Example :
Input:
widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
10]
S = "abcdefghijklmnopqrstuvwxyz"
Output: [3, 60]
Explanation:
All letters have the same length of 10. To write all 26 letters,
we need two full lines and one line with 60 units.
```

```
Example :
Input:
widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,
10]
S = "abcdefghijklmnopqrstuvwxyz"
Output: [3, 60]
Explanation:
All letters have the same length of 10. To write all 26 letters,
we need two full lines and one line with 60 units.
```

Note:

- The length of `S` will be in the range [1, 1000].
- `S` will only contain lowercase letters.
- `widths` is an array of length `26`.
- `widths[i]` will be in the range of `[2, 10]`.