Top techniques

1. Use stack to reverse string here s

- can be applied to reverse string

2. Use stack

3. Two pointers

4. Sliding window

**Stack with an inner while loop for removal**

Have a stack

1. You add everythigng stack.add(b)

2. Use a while loop inside the stack and then if the condition is true, keep removing from that stack here

The problem here is remove duplicate letters here

**Stack by keeping the last occurrence of sth here:**

**How to pop and push without using a data structure?**

Use below:

Temp = rev \* 10 + pop

Can also be done with reverse integer here in leetcode

Know that when you remove string

**Trick 4: Turn string number into actual integer number**

And then here we have the code below here:

while index < len(s) and s[index].isdigit():

number = number \* 10 + int(s[index])

index += 1

Another problem that we can see from before here

So we can

A safe strategy is to iterate over the string and insert each character we want to keep into a **list** (Python) or **StringBuilder** (Java). Then once we have all the characters, it is a single *O*(*n*) step to convert them into a string.

**Stickers to spell word**

We are given n different types of stickers. Each sticker has a lowercase English word on it.

You would like to spell out the given string target by cutting individual letters from your collection of stickers and rearranging them. You can use each sticker more than once if you want, and you have infinite quantities of each sticker.

Return *the minimum number of stickers that you need to spell out*target. If the task is impossible, return -1.

**Note:** In all test cases, all words were chosen randomly from the 1000 most common US English words, and target was chosen as a concatenation of two random words.

**Input:** stickers = ["with","example","science"], target = "thehat"

Interleave string

[**97. Interleaving String**](https://leetcode.com/problems/interleaving-string/)

Medium

Topics

Companies

Given strings s1, s2, and s3, find whether s3 is formed by an **interleaving** of s1 and s2.

An **interleaving** of two strings s and t is a configuration where s and t are divided into n and m

substrings

 respectively, such that:

* s = s1 + s2 + ... + sn
* t = t1 + t2 + ... + tm
* |n - m| <= 1
* The **interleaving** is s1 + t1 + s2 + t2 + s3 + t3 + ... or t1 + s1 + t2 + s2 + t3 + s3 + ...

**Note:** a + b is the concatenation of strings a and b.

**Example 1:**

A diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of

Description automatically generated

**Input:** s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac"

**Output:** true

**Explanation:** One way to obtain s3 is:

Split s1 into s1 = "aa" + "bc" + "c", and s2 into s2 = "dbbc" + "a".

Interleaving the two splits, we get "aa" + "dbbc" + "bc" + "a" + "c" = "aadbbcbcac".

Since s3 can be obtained by interleaving s1 and s2, we return true.

[**127. Word Ladder**](https://leetcode.com/problems/word-ladder/)

Hard

Topics

Companies

A **transformation sequence** from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

Given two words, beginWord and endWord, and a dictionary wordList, return *the****number of words****in the****shortest transformation sequence****from* beginWord *to* endWord*, or*0*if no such sequence exists.*

**Example 1:**

**Input:** beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]

**Output:** 5

**Explanation:** One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> cog", which is 5 words long.

Regular expression matching