[LeetCode](https://leetcode.com/problems/custom-sort-string/description/?envType=daily-question&envId=2024-03-11)

<https://github.com/AdityaKonda6/-50DaysOfCoding>

<https://leetcode.com/problems/custom-sort-string/description/?envType=daily-question&envId=2024-03-11>

<https://www.linkedin.com/in/aditya-adi-konda/>

Day 21 of [#50dayscodingchallenge](https://www.linkedin.com/feed/hashtag/?keywords=50dayscodingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633):  
[#leetcode](https://www.linkedin.com/feed/hashtag/?keywords=leetcode&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodechallenge](https://www.linkedin.com/feed/hashtag/?keywords=leetcodechallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodestreak](https://www.linkedin.com/feed/hashtag/?keywords=leetcodestreak&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode2024](https://www.linkedin.com/feed/hashtag/?keywords=leetcode2024&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode50day](https://www.linkedin.com/feed/hashtag/?keywords=leetcode50day&highlightedUpdateUrns=urn:li:activity:7166316239483461633)  
   
Just kicked off my coding journey with a fascinating problem - "Successfully solved LeetCode Problem “791. Custom Sorting String.” !”  
   
✨ Task: You are given two strings order and s. All the characters of order are unique and were sorted in some custom order previously.

Permute the characters of s so that they match the order that order was sorted. More specifically, if a character x occurs before a character y in order, then x should occur before y in the permuted string.

Return any permutation of s that satisfies this property.

Examples:

Example 1:

Input:  order = "cba", s = "abcd"

Output:  "cbad"

Explanation: "a", "b", "c" appear in order, so the order of "a", "b", "c" should be "c", "b", and "a".

Since "d" does not appear in order, it can be at any position in the returned

string. "dcba", "cdba", "cbda" are also valid outputs.

Example 2:

Input:  order = "bcafg", s = "abcd"

Output:  "bcad"

Explanation: The characters "b", "c", and "a" from order dictate the order for the characters in s. The character "d" in s does not appear in order, so its position is flexible.

Following the order of appearance in order, "b", "c", and "a" from s should be arranged as "b", "c", "a". "d" can be placed at any position since it's not in order. The output "bcad" correctly follows this rule. Other arrangements like "bacd" or "bcda" would also be valid, as long as "b", "c", "a" maintain their order.

Let's Connect:

If you find this problem intriguing or have insights to share, let's connect! I'm passionate about problem-solving, algorithmic thinking, and collaborative learning. Feel free to comment or reach out for engaging discussions and knowledge exchange.Unravel the mystery using your coding skills!

[#CodingChallenge](https://www.linkedin.com/feed/hashtag/?keywords=codingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#Algorithm](https://www.linkedin.com/feed/hashtag/?keywords=algorithm&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#LinkedInPost](https://www.linkedin.com/feed/hashtag/?keywords=linkedinpost&highlightedUpdateUrns=urn:li:activity:7166316239483461633) #Algorithm #Optimization #DataStructures #CodingChallenge  
  
Excited about the progress and challenges ahead!  
   
Make Sure You Follow My GitHub For Solutions: <https://github.com/AdityaKonda6/-50DaysOfCoding>  
  
  
Happy coding!

**Solution:-**

class Solution {

  public String customSortString(final String order, final String s) {

    StringBuilder sb = new StringBuilder();

    int[] count = new int[128];

    for (final char c : s.toCharArray())

      ++count[c];

    for (final char c : order.toCharArray())

      while (count[c]-- > 0)

        sb.append(c);

    for (char c = 'a'; c <= 'z'; ++c)

      while (count[c]-- > 0)

        sb.append(c);

    return sb.toString();

  }

}



