Solution: Find the Difference

Let's solve the Find the Difference problem using the Bitwise Manipulation pattern.

We'll cover the following

Statement

Given two strings, `str1` and `str2`, find the index of the extra character that is present in only one of the strings.

Constraints:

- $0 \leq$ `str1.length`, `str2.length` ≤ 1000
- Either `str2.length` $=$ `str1.length + 1`, or, `str1.length` $=$ `str2.length + 1`
- The strings consist of lowercase English letters.

Solution

So far, you've probably brainstormed some approaches and have an idea of how to solve this problem. Let's explore some of these approaches and figure out which one to follow based on considerations such as time complexity and any implementation constraints.

Naive approach

The naive approach is to first sort both the strings. Then loop over the longer string and compare both strings, character by character. Finally, when one extra character is found in the longer string which does not match with the character at the corresponding index in the other string, we break out of the loop and return the index where the comparison failed.

The time complexity of this solution would be $O(n \log n + n)$, that is $O(n \log n)$. Here, $O(n \log n)$ is the cost of sorting one of the strings. The space complexity would be $O(1)$.

Optimized approach using bitwise manipulation

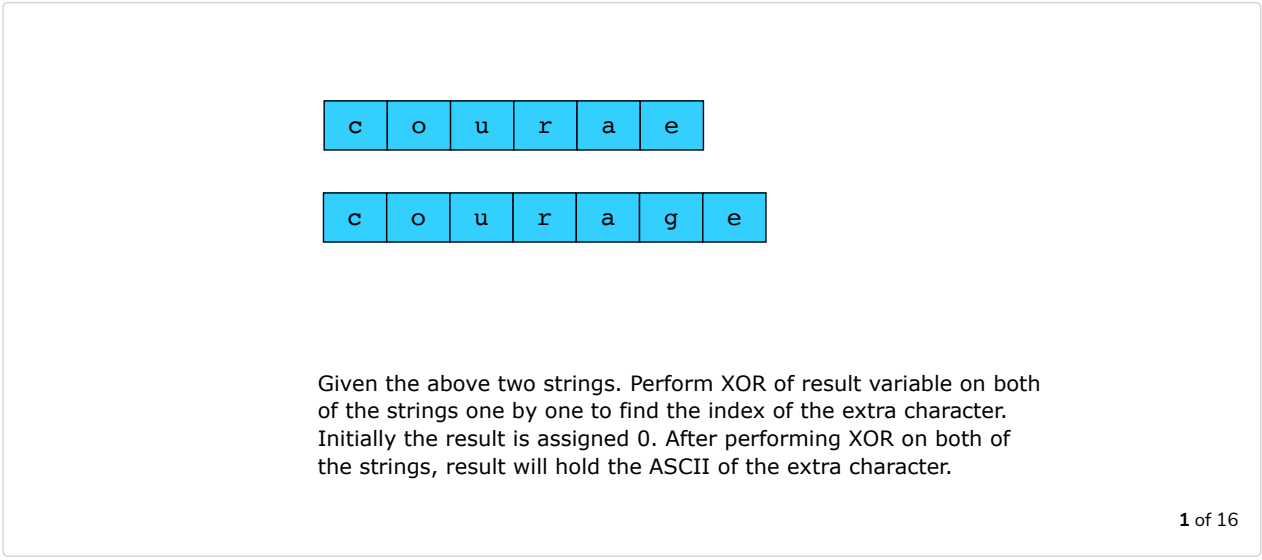
An optimized approach to solve this problem is using the bitwise manipulation technique. We use the bitwise XOR operation to identify the extra character from one of the strings and return the index of that character.

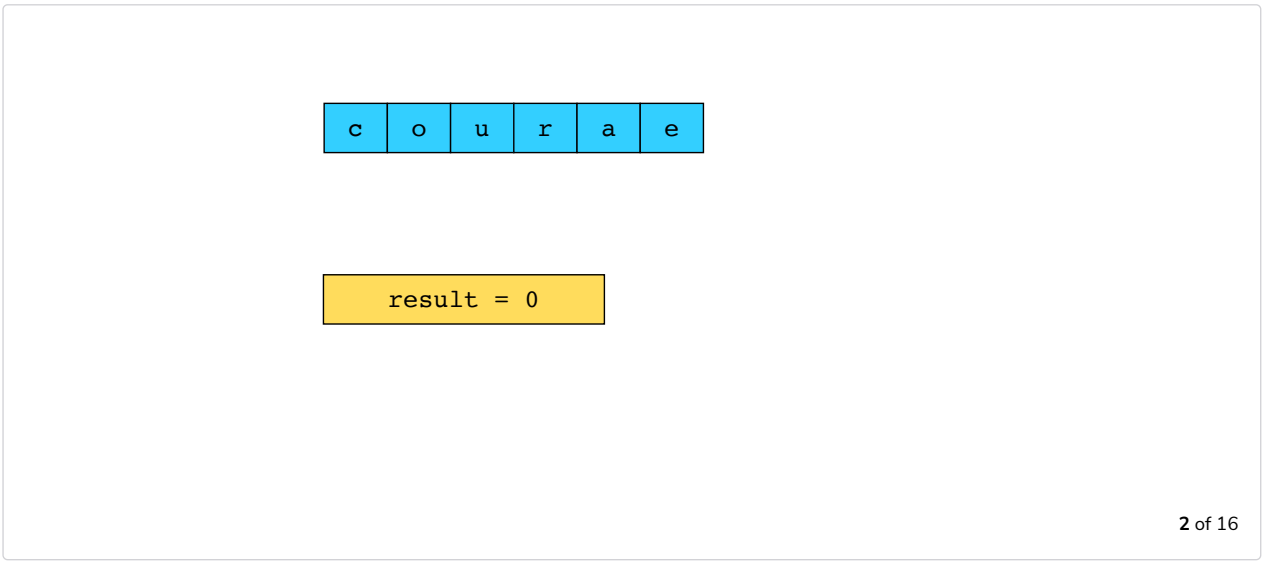
The algorithm proceeds through the following steps:

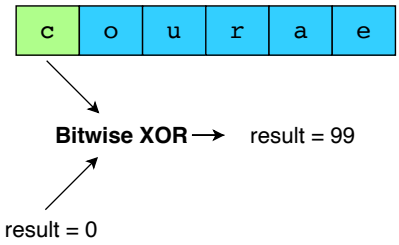
1. Initialize a variable, `result`, with 0 to store the XOR result.

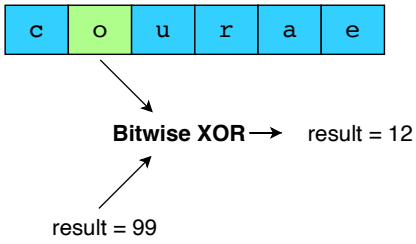
2. Find the lengths of both strings.
3. Iterate over the characters of `str1`, and for each character, do the following operations:
    - Compute the ASCII value of the character.
    - Perform a bitwise XOR operation between the current value of `result` and the ASCII value.
    - Store the result of the bitwise XOR operation back in `result`.
4. Now, iterate over the characters of `str2` and repeat the operations performed in step 3 for each character in `str2`.
5. After iterating over both strings, the `result` variable will correspond to the ASCII value of the extra character.
6. Find and return the index of the extra character from the string with a greater length.

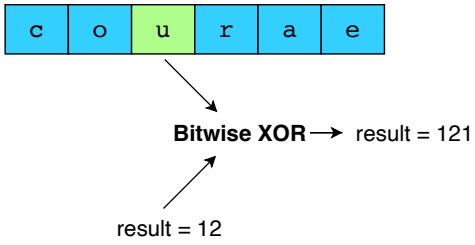
The slides below illustrate how we would like the algorithm to run:

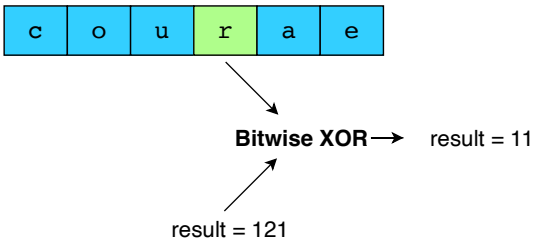


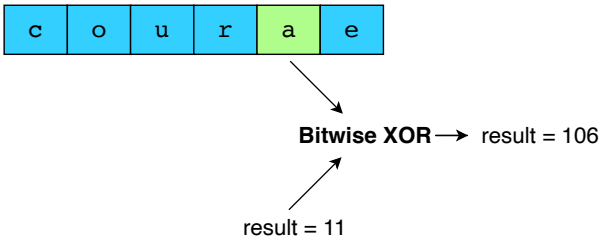


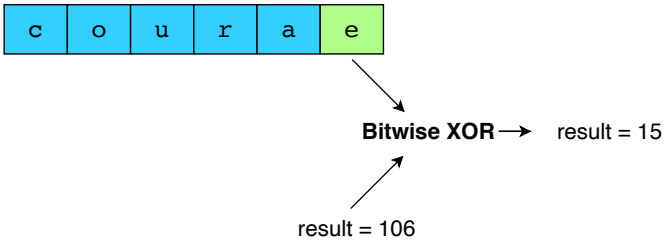

c o u r a e
Bitwise XOR
result = 99
result = 0

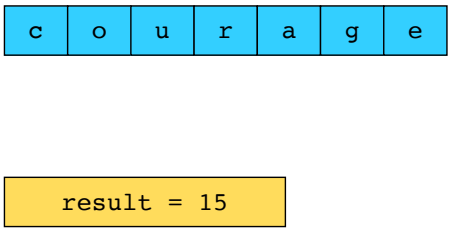

c o u r a e
Bitwise XOR
result = 12
result = 99

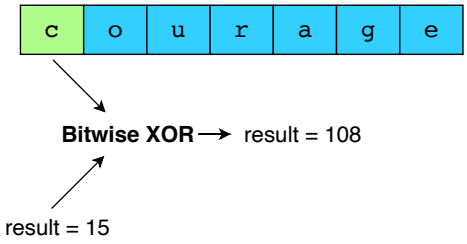

c o u r a e
Bitwise XOR
result = 121
result = 12

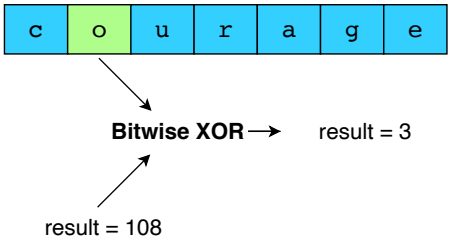

c o u r a e
Bitwise XOR
result = 11
result = 121

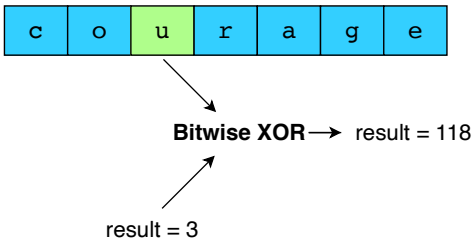

c o u r a e
Bitwise XOR
result = 106
result = 11


c o u r a e
Bitwise XOR
result = 15
result = 106

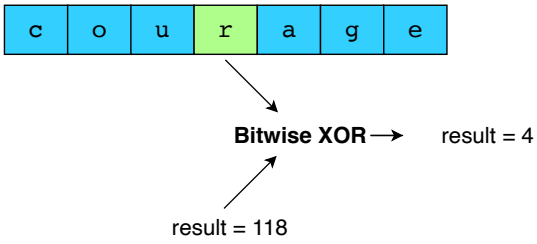

c o u r a g e
result = 15


c o u r a g e
Bitwise XOR
result = 108
result = 15

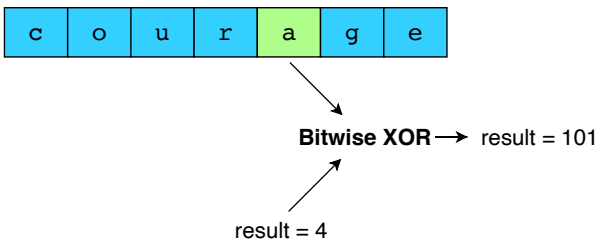

c o u r a g e
Bitwise XOR
result = 3
result = 108



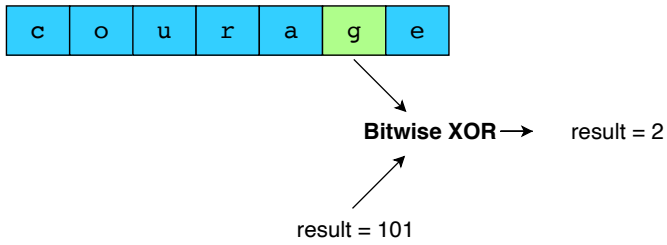
12 of 16



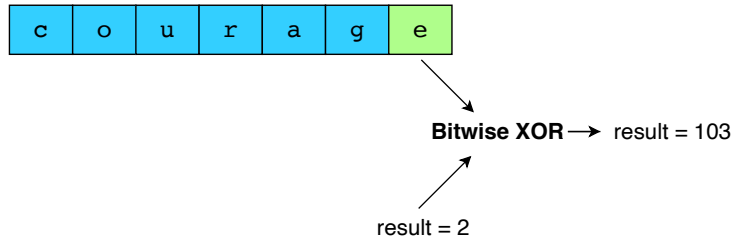
13 of 16



14 of 16



15 of 16



The result = 103, which is the ASCII of "g".
The index of "g" is **5**.

16 of 16

> **Note**: In the following section, we will gradually build the solution. Alternatively, you can skip straight to just the code.

Step-by-step solution construction

Let's start with the simplest step:

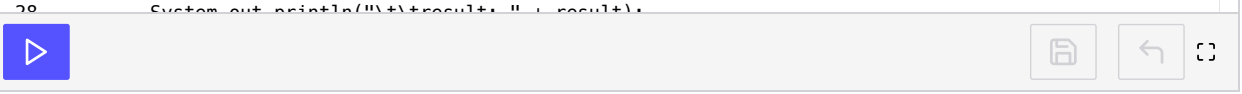
Initialize a variable `result` with `0`. Find the lengths of both strings. Iterate over each character of `str1`, and for each character, perform a bitwise XOR operation between the current value of `result` and the ASCII value of the character. Update the `result` with the computed XOR value every time.

Java

```
class FindDifference {

    public static String printStringWithMarkers(String strn, int pValue) {
      String out = "";
      for (int i = 0; i < pValue; i++) {
        out += String.valueOf(strn.charAt(i));
      }
      out += "«";
      out += String.valueOf(strn.charAt(pValue)) + "»";
      for (int i = pValue + 1; i < strn.length(); i++) {
        out += String.valueOf(strn.charAt(i));
      }
```

```
13        return out;
14      }
15
16      public static int extraCharcterIndex(String str1, String str2) {
17        // Initialize the result variable to store the result
18        int result = 0;
19        // Store the length of str1 in str1Length variable
20        int str1Length = str1.length();
21        // Store length of str2 in str2Length variable
22        int str2Length = str2.length();
23        // Traverse the string 1 till the end and perform xor with the result
24        System.out.println("\n\tTraversing first string");
25        for (int i = 0; i < str1Length; i++) {
26          System.out.println("\t\t" + printStringWithMarkers(str1, i));
27          System.out.println("\t\tCurrent character: " + str1.charAt(i));
28          System.out.println("\t\tresult: " + result);
```



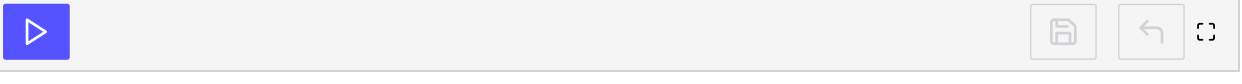
Find the Difference

Similar to the first step, now iterate over each character of the `str2`, and for each character, perform the bitwise XOR operation between the current value of the `result` and the ASCII value of the character. Again, update the `result` variable with the computed XOR value every time. After the traversals of both strings, `result` will contain the ASCII value of the extra character.

This technique works because XOR returns 0 if the bits are the same (both 0 or both 1) and 1 if the bits differ (one is 0, and the other is 1). For example, 111 XOR 111 returns 000, and 101 XOR 010 returns 111. Keeping this property in mind, when we traverse the `str2`, the common characters between the `str1` and `str2` cancel out, leaving only the extra character in the `result` variable.

Java

```
48        return 0;
49      }
50
51      public static void main(String[] args) {
52        // Driver code
53        // Example - 1
54        String[] string1 = {
55          "wxyz",
56          "cbda",
57          "jlkmn",
58          "courae",
59          "hello"
60        };
61        String[] string2 = {
62          "zwxgy",
63          "abc",
64          "klmn",
65          "couearg",
66          "helo"
67        };
68        for (int i = 0; i < string1.length; i++) {
69          System.out.println(i + 1 + ".\tString1 = " + string1[i] + " \n\tString2 = " + string2[i]);
70          extraCharcterIndex(string1[i], string2[i]);
71          System.out.println(PrintHyphens.repeat("-", 100));
72        }
73      }
74  }
```



Find the Difference

Now, we have the ASCII value of the extra character in the `result` variable. To find the index of this extra character, we check the length of both strings. Since the extra character is present in the longest string, we

return the index of that extra character from the longest string.

Java

```
64          return str2.indexOf((char)(result));
65        }
66      }
```

```
70        String[] string1 = {
71          "wxyz",
72          "cbda",
73          "jlkmn",
74          "courae",
75          "hello"
76        };
77        String[] string2 = {
78          "zwxgy",
79          "abc",
80          "klmn",
81          "couearg",
82          "helo"
83        };
84        for (int i = 0; i < string1.length; i++) {
85          System.out.println(i + 1 + ".\tString1 = " + string1[i] + " \n\tString2 = " + string2[i]);
86          System.out.println("at index "  + extraCharcterIndex(string1[i], string2[i]));
87          System.out.println(PrintHyphens.repeat("-", 100));
88        }
89      }
90  }
```

Find the Difference

Just the code

Here's the complete solution to this problem:

Java

```
25        }
26      }
27
28      public static void main(String[] args) {
29        // Driver code
30        // Example - 1
31        String[] string1 = {
```