Bitwise Manipulation: Introduction

Let's go over the bitwise manipulation pattern, its real-world applications, and some problems we can solve with it.

We'll cover the following

Overview

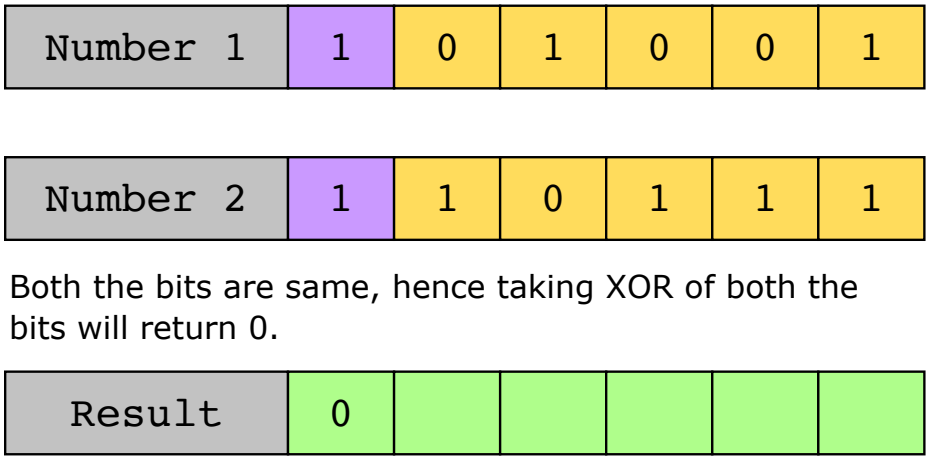
Bitwise Manipulation is the process of modifying bits algorithmically using bitwise operations. Logical bitwise operations are the fastest computations, because processors natively support them. This approach generally leads to efficient solutions in problems where we can efficiently transform the input into its binary form or manipulate it directly at the bit level to produce the required output.

A bitwise operation works on a bit string, bit array, or a binary numeral. Bitwise operators take bits as their operands and calculate the corresponding bit value in the result.

The table below describes the functioning of the four widely used bitwise operators:

Operator	Functioning
NOT (~)	This is a unary operator. If the argument is a 1-bit, flip it to change a 1 to a 0, and vice versa. If the argument is a string of bits, all bits in the string are reversed, turning 1's into 0's and vice versa.
AND (&)	If both the bits are 1, then 1 is returned. Otherwise, 0 is returned.
OR (|)	If either of the bits is 1, then 1 is returned. Otherwise, 0 is returned.
XOR (^)	If both bits are equal, then 0 is returned. Otherwise, 1 is returned.

The following illustration demonstrates how the XOR operator works.



Number 1	1	0	1	0	0	1

Number 2	1	1	0	1	1	1

Now both the bits are different, hence taking XOR of both the bits will return 1.

Result	0	1				

The rest of the problem will be solved by the same approach.

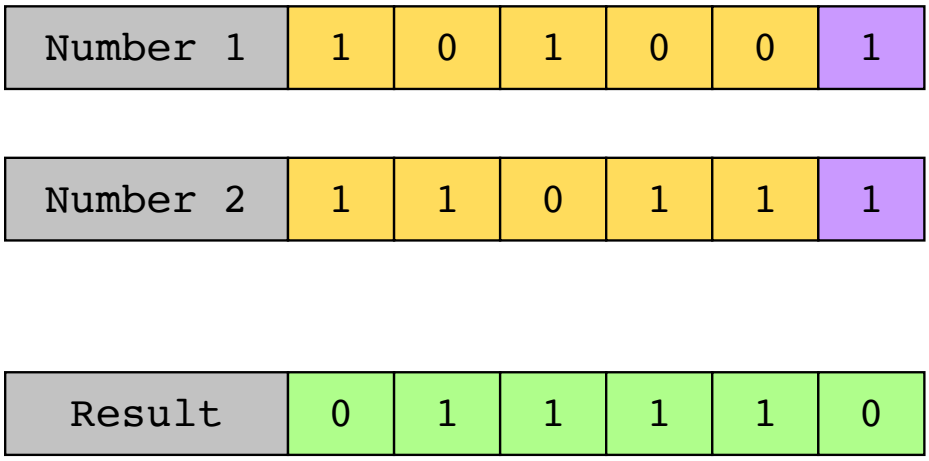
Number 1	1	0	1	0	0	1

Number 2	1	1	0	1	1	1

Result	0	1	1			

Number 1	1	0	1	0	0	1
Number 2	1	1	0	1	1	1
Result	0	1	1	1		

Number 1	1	0	1	0	0	1
Number 2	1	1	0	1	1	1
Result	0	1	1	1	1	



Examples

The following examples illustrate some problems that can be solved with this approach:

Swap two numbers without using a temporary variable

x	10

y	5

Convert input numbers to their binary form:

x	1	0	1	0

y	0	1	0	1

Take XOR of x and y and storing the result in x:

x	1	1	1	1

Take XOR of x and y and now storing the result in y:

y	1	0	1	0

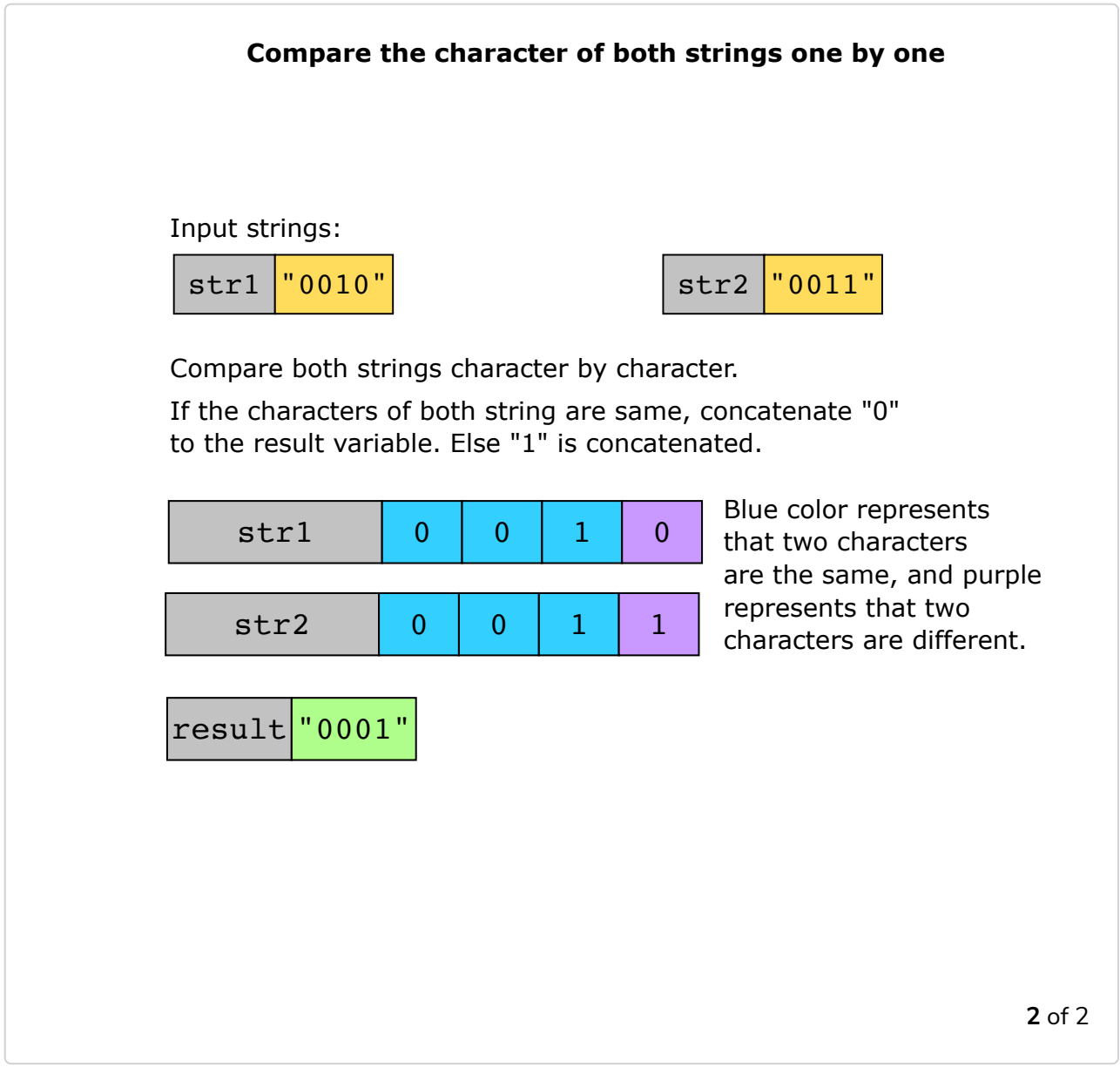
Lastly, take XOR of x and y again and now store the result in x:

x	0	1	0	1

Note that if we now convert both the values back into the decimal numbers both the variables have their values swapped.

x	5

y	10



Does my problem match this pattern?

- Yes, if either of these conditions is fulfilled:
  - The input data can be usefully manipulated at the level of the primitive bitwise logical operations, in order to compute some portion or all of the solution.
  - The input data is unsorted, and the answer seems to require sorting, but we want to do better than $O(n \log n)$.
- No, if the input data type is not in numeric form or cannot be converted to numeric form.

Real-world problems

Many problems in the real world share the bitwise manipulation pattern. Let's look at some examples.

- **Bit fields (flags):** They can be used to implement a way of representing things whose state is defined by a boolean expression.