415. Add Strings

June 23, 2020 | 7.1K views

Average Rating: 4.17 (18 votes)

Given two non-negative integers `num1` and `num2` represented as string, return the sum of `num1` and `num2`.

Note:

1. The length of both `num1` and `num2` is < 5100.
2. Both `num1` and `num2` contains only digits `0-9`.
3. Both `num1` and `num2` does not contain any leading zero.
4. You **must not use any built-in BigInteger library** or **convert the inputs to integer** directly.

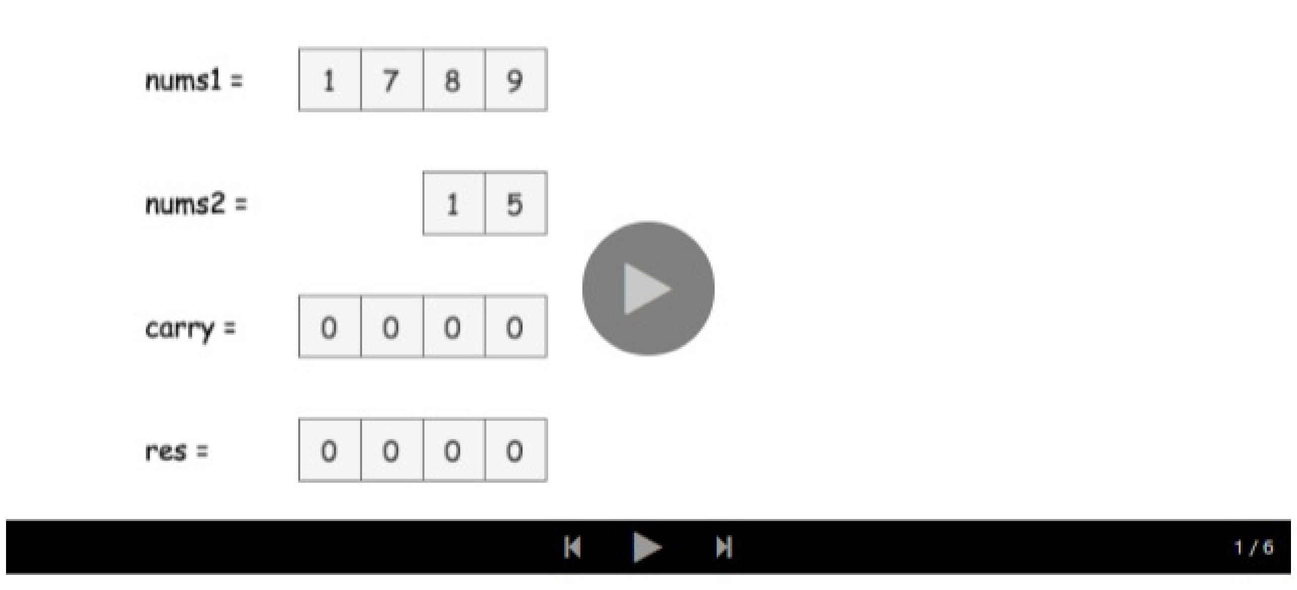
Overview

Facebook interviewers like this question and propose it in four main variations. The choice of algorithm should be based on the input format:

1. Strings (the current problem). Use schoolbook digit-by-digit addition. Note, that to fit into constant space is not possible for languages with immutable strings, for example, for Java and Python. Here are two examples:
   - Add Binary: sum two binary strings.
   - Add Strings: sum two non-negative numbers in a string representation without converting them to integers directly.
2. Integers. Usually, the interviewer would ask you to implement a sum without using `+` and `-` operators. Use bit manipulation approach. Here is an example:
   - Sum of Two Integers: Sum two integers without using `+` and `-` operators.
3. Arrays. The same textbook addition. Here is an example:
   - Add to Array Form of Integer.
4. Linked Lists. Sentinel Head + Textbook Addition. Here are some examples:
   - Plus One.
   - Add Two Numbers.
   - Add Two Numbers II.

Approach 1: Elementary Math

Here we have two strings as input and asked not to convert them to integers. Digit-by-digit addition is the only option here.



Algorithm

- Initialize an empty `res` structure. Once could use array in Python and StringBuilder in Java.
- Start from `carry = 0`.
- Set a pointer at the end of each string: `p1 = num1.length() - 1`, `p2 = num2.length() - 1`.
- Loop over the strings from the end to the beginning using `p1` and `p2`. Stop when both strings are used entirely.
  - Set `x1` to be equal to a digit from string `nums1` at index `p1`. If `p1` has reached the beginning of `nums1`, set `x1` to `0`.
  - Do the same for `x2`. Set `x2` to be equal to digit from string `nums2` at index `p2`. If `p2` has reached the beginning of `nums2`, set `x2` to `0`.
  - Compute the current value: `value = (x1 + x2 + carry) % 10`, and update the carry: `carry = (x1 + x2 + carry) / 10`.
  - Append the current value to the result: `res.append(value)`.
- Now both strings are done. If the carry is still non-zero, update the result: `res.append(carry)`.
- Reverse the result, convert it to a string, and return that string.

Implementation

Java | Python3 | Copy

```python
class Solution:
    def addStrings(self, num1: str, num2: str) -> str:
        res = []

        carry = 0
        p1 = len(num1) - 1
        p2 = len(num2) - 1
        while p1 >= 0 or p2 >= 0:
            x1 = ord(num1[p1]) - ord('0') if p1 >= 0 else 0
            x2 = ord(num2[p2]) - ord('0') if p2 >= 0 else 0
            value = (x1 + x2 + carry) % 10
            carry = (x1 + x2 + carry) // 10
            res.append(value)
            p1 -= 1
            p2 -= 1

        if carry:
            res.append(carry)

        return ''.join(str(x) for x in res[::-1])
```

Complexity Analysis

- Time Complexity: $\mathcal{O}(\max(N_1, N_2))$, where N_1 and N_2 are length of `nums1` and `nums2`. Here we do $\max(N_1, N_2)$ iterations at most.
- Space Complexity: $\mathcal{O}(\max(N_1, N_2))$, because the length of the new string is at most $\max(N_1, N_2) + 1$.

Rate this article: ★★★★★

Comments: 7

Sort By

Type comment here... (Markdown is supported)

Preview Post

TheGreatest ★ 10 June 24, 2020 11:59 PM
Great breakdown and thank you for the categorical reference of the similar problems in the overview section!
6 Share Reply

ausafahmed ★ 5 June 26, 2020 8:57 AM
Quite a wholesome article, with breakdown. Thank you!
1 Share Reply

cosR ★ 4 June 25, 2020 10:19 AM
When i used stringbuilder.insert(0,num) , the time complexity was more. However, when i used stringbuilder.append(num) and later stringbuilder.reverse() the time improved. Why? Isn't inserting at 0th position and appending takes same time?
1 Share Reply
SHOW 3 REPLIES

ryabkin ★ 8 July 1, 2020 4:51 AM
My initial thinking about the constraint **"4. You must not use any built-in BigInteger library or convert the inputs to integer directly."** was not to use `int(string)`. I guess converting every char to ordinal is more appropriate solution

```
class Solution(object):
```
Read More
1 Share Reply
SHOW 1 REPLY

mikram ★ 0 July 9, 2020 7:48 AM
Simple and more readable python solution which requires reversing numbers:

```
class Solution:
    def addStrings(self, num1: str, num2: str) -> str:
        # reverse both numbers
```
Read More
0 Share Reply

GaloisTheMLE ★ 0 July 2, 2020 9:43 AM

```
class Solution(object):
    def addStrings(self, num1, num2):
        """
        :type num1: str
```
Read More
0 Share Reply

samirdeeb ★ 0 June 30, 2020 3:46 AM
using deque will save reversing the result
0 Share Reply