516 字符串相加

```
Lable：字符串

给定两个字符串形式的非负整数 num1 和num2 ，计算它们的和。

num1 和num2 的长度都小于 5100
num1 和num2 都只包含数字 0-9
num1 和num2 都不包含任何前导零
你不能使用任何内建 BigInteger 库， 也不能直接将输入的字符串转换为整数形式
```

- 内存最少 （思路同倒序遍历相同）

```java
class Solution {
    public String addStrings(String num1, String num2) {
        int l1 = num1.length() - 1;
        int l2 = num2.length() - 1;
        StringBuilder sb = new StringBuilder("");
        int temp = 0;
        while (l1 >= 0 || l2 >= 0) {
            // 三元运算符，超出部分使用 0 进行填充
            int n1 = l1>=0 ? num1.charAt(l1) - '0' : 0;
            int n2 = l2>=0 ? num2.charAt(l2) - '0' : 0;
            temp = temp + n1 + n2;
            sb.append(temp % 10);
            temp = temp / 10;
            l1--;
            l2--;
        }
        if (temp > 0) {
            sb.append(temp);  // 最后可能还有进位
        }
        return sb.reverse().toString();
    }
}
```

- 用时最少（ASCII int 与 char 进转换）

```
class Solution {
    public String addStrings(String num1, String num2) {

        if (num1.length() > num2.length()) { // nums1作为较短的串
            return addStrings(num2, num1);
        }


        char[] buf = new char[Math.max(num1.length(), num2.length()) + 1]; // 加
1 是为了进位

        int i = num1.length() - 1, j = num2.length() - 1;
        int k = buf.length - 1, remain = 0, num;

        // 短部分
        while (i >= 0) {
            num = num1.charAt(i--) + num2.charAt(j--) - '0' * 2 + remain;  //
ASCII 数字 避免使用API转换
            if (num >= 10) {
                num -= 10;
                remain = 1;
            }else {
                remain = 0;
            }
            buf[k--] = (char)(num + '0');
        }

        // 对准较长的部分
        while (j >= 0) {
            num = num2.charAt(j--) - '0' + remain;

            if (num >= 10) {
                num -= 10;
                remain = 1;
            }else {
                remain = 0;
            }
            buf[k--] = (char)(num + '0');
        }

        if (remain == 1) {
            buf[k] = '1';
            return new String(buf);
        }else {
            return new String(buf, 1, buf.length - 1);
        }
    }
}
```

- 倒序遍历（mine）

```
class Solution {
    public String addStrings(String num1, String num2) {

        int curr1 = num1.length() - 1;
        int curr2 =  num2.length() - 1;
        int carry = 0;
        int add = 0;
        int reminder = 0;
        StringBuilder stringBuilder = new StringBuilder();

        while (curr1 >=  0 || curr2 >= 0) {

            if (curr1 < 0) {

                add =  Integer.parseInt(num2.substring(curr2, curr2 + 1)) +
carry;
                reminder = add % 10;
                carry = add / 10;

            } else if (curr2 < 0) {

                add = Integer.parseInt(num1.substring(curr1, curr1 + 1)) +
carry;
                reminder = add % 10;
                carry = add / 10;

            } else {

                add = Integer.parseInt(num1.substring(curr1, curr1 + 1)) +
Integer.parseInt(num2.substring(curr2, curr2 + 1)) + carry;
                reminder = add % 10;
                carry = add / 10;
            }

            stringBuilder.append(reminder);

            curr1--;
            curr2--;
        }

        if (carry != 0) {
            stringBuilder.append(carry);
        }

        return stringBuilder.reverse().toString();  // 倒序
    }
}
```