# 字符串相加

给定两个字符串形式的非负整数 num1 和 num2 ，计算它们的和。

**提示：**

1. num1 和 num2 的长度都小于 5100
2. num1 和 num2 都只包含数字 0-9
3. num1 和 num2 都不包含任何前导零
4. **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**

```
class Solution {
public:
//每一位的和：当前位的值+上一步进位
/* 相加之后对应位的值：
和>9  和-10  进位：1
和<9  和     进位：0
*/
    /*string addStrings(string num1, string num2) {
        int s1=num1.size()-1;
        int s2=num2.size()-1;//记录下标 逆序遍历 逐位相加
        int add=0;//进位
        string res="";
        while(s1>=0||s2>=0||add!=0)
        {
            int x=s1>=0?num1[s1]-'0':0;
            int y=s2>=0?num2[s2]-'0':0;
            int result=x+y+add;
            res.push_back(result%10+'0');
            add=result/10;
            s1--;
            s2--;
        }
        reverse(res.begin(),res.end());//逆序翻转
        return res;
    }*/
//逆序只需尾插  O（1）的复杂度  如果不逆序  就得头插 O（n）的复杂度 头插导致算法复杂度略高
    string addStrings(string num1, string num2)
    {
        //从个位开始相加，最后一个字符相加
        int end1=num1.size()-1;
```

```
        int end2=num2.size()-1;
        string ret;//保存最终结果
        int step=0;//进位
        while(end1>=0||end2>=0)
        {
            //当前位的和 默认初始值为 step 0 或 1
            int cursum=step;
            if(end1>=0)
            {
                cursum+=num1[end1--]-'0';
            }
            if(end2>=0)
            {
                cursum+=num2[end2--]-'0';
            }
            if(cursum>9)
            {
                cursum=cursum-10;
                step=1;
            }
            else
                step=0;
            //保存当前位的值
            ret.insert(0,1,cursum+'0');
        }
        //出循环 step 仍为 1  判断最高位是否有进位
        if(step==1)
        {
            ret.insert(0,1,'1');
        }
        return ret;
    }
};
```