algorithm

何书文
1201220707@pku.edu.cn

2020 年 1 月 7 日

目录

Chapter
list

list

addTwoNumbers

给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储一位数字。如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例：输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

输出：7 -> 0 -> 8

原因：342 + 465 = 807

```
1
2       package main
3
4       import (
5           "fmt"
6       )
7
8       type ListNode struct {
9           Val  int
10          Next *ListNode
11      }
12      type List struct {
13          headNode *ListNode // head node
14      }
15
16      // 1.Insert
17      func Insert(value int, list *ListNode, position *
            ListNode) {
18          tempCell := new(ListNode)
19          if tempCell == nil {
20              fmt.Println("out of space")
```

```
21          }
22          tempCell.Val = value
23          tempCell.Next = position.Next
24          position.Next = tempCell
25      }
26
27      // 2.Print
28      func PrintList(list *ListNode) {
29          if list.Next != nil {
30              fmt.Println(list.Val)
31              PrintList(list.Next)
32          } else {
33              fmt.Println(list.Val)
34          }
35      }
36
37      func main() {
38          l1 := new(ListNode)
39          listDate := l1
40          // insert data to l1
41          Insert(2, listDate, l1)
42          Insert(4, listDate, l1)
43          Insert(3, listDate, l1)
44          l2 := new(ListNode)
45          //
46          listDate2 := l2
47          // insert data to l1
48          Insert(5, listDate2, l2)
49          Insert(6, listDate2, l2)
50          Insert(4, listDate2, l2)
51          l3 := addTwoNumbers(l1, l2)
52          PrintList(l3)
53      }
54
55      func addTwoNumbers(l1 *ListNode, l2 *ListNode) *ListNode
            {
56          promotion := 0      // 进位值, 只可能为0或1
57          var head *ListNode // 结果表的头结点
```

```
58          var rear *ListNode // 保存结果表的尾结点
59          for nil != l1 || nil != l2 {
60              sum := 0
61              if nil != l1 {
62                  sum += l1.Val
63                  l1 = l1.Next
64              }
65              if nil != l2 {
66                  sum += l2.Val
67                  l2 = l2.Next
68              }
69
70              sum += promotion
71              promotion = 0
72
73              if sum >= 10 {
74                  promotion = 1
75                  sum = sum % 10
76              }
77
78              node := &ListNode{
79                  sum,
80                  nil,
81              }
82
83              if nil == head {
84                  head = node
85                  rear = node
86              } else {
87                  rear.Next = node
88                  rear = node
89              }
90          }
91
92          if promotion > 0 {
93              rear.Next = &ListNode{
94                  promotion,
95                  nil,
```

```
96                  }
97              }
98              return head
99          }
```