

踏浪前行

记录学习，分享心得

≤	2012年4月							≥
	日	一	二	三	四	五	六	
	25	26	27	28	29	30	31	
	1	2	3	4	5	6	7	
	8	9	10	11	12	13	14	
	15	16	17	18	19	20	21	
	22	23	24	25	26	27	28	
	29	30	1	2	3	4	5	

昵称: [Jason Yang](#)
园龄: 2年10个月
粉丝: 5
关注: 0
[+加关注](#)

搜索

找我看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[算法\(2\)](#)
[c/c++\(1\)](#)
[Linux\(1\)](#)
[Python\(1\)](#)
[Ubuntu\(1\)](#)

随笔分类

[.Net](#)
[C#](#)
[C/C++\(3\)](#)
[Linux](#)
[Python\(1\)](#)
[Qt](#)
[Ubuntu](#)
[算法\(3\)](#)

随笔档案

[2012年4月 \(5\)](#)

[博客园](#) [首页](#) [新随笔](#) [联系](#) [订阅](#) [XML](#) [管理](#)

posts - 5, comments - 6, trackbacks - 0

算法题-大数相乘问题

今天在网上看到一个大数相乘的问题，题目是这样的：输入两个整数，要求输出这两个数的乘积。输入的数字可能超过计算机内整形数据的存储范围。

分析：

由于数字无法用一个整型变量存储，很自然的想到用字符串来表示一串数字。然后按照乘法的运算规则，用一个乘数的每一位乘以另一个乘数，然后将所有中间结果按正确位置相加得到最终结果。可以分析得出如果乘数为A和B，A的位数为m，B的位数为n，则乘积结果为m+n-1位（最高位无进位）或m+n位（最高位有进位）。因此可以分配一个m+n的辅存来存储最终结果。为了节约空间，所有的中间结果直接在m+n的辅存上进行累加。最后为了更符合我们的乘法运算逻辑，可以讲数字逆序存储，这样数字的低位就在数组的低下标位置，进行累加时确定下标位置容易些。

下面是我的解法。

首先是对数组逆序的函数：

```
1 void reverseOrder(char* str, int p, int q)
2 {
3     char temp;
4     while(p < q)
5     {
6         temp = str[p];
7         str[p] = str[q];
8         str[q] = temp;
9         p ++;
10        q --;
11    }
12 }
```

然后是完成大数相乘的函数：

```
1 char* multiLargeNum(char* A, char* B)
2 {
3     int m = strlen(A);
4     int n = strlen(B);
5     char* result = new char[m+n+1];
6     memset(result, '0', m+n);
7     result[m+n] = '\0';
8     reverseOrder(A, 0, m-1);
9     reverseOrder(B, 0, n-1);
10 }
```

文章分类

[.NET](#)
[C#](#)
[C/C++](#)
[Linux\(1\)](#)
[Python](#)
[Qt](#)
[Ubuntu\(1\)](#)
[算法](#)

阅读排行榜

1. 算法题-大数相乘问题(2245)
2. 算法题-字符串循环移位问题(1613)
3. Python小应用1 - 抓取网页中的链接地址(1106)
4. const修饰指针的情况分析(299)
5. 微软面试记录(165)

评论排行榜

1. const修饰指针的情况分析(4)
2. 算法题-字符串循环移位问题(2)

推荐排行榜

1. 算法题-字符串循环移位问题(4)

```

11  int multiFlag; // 乘积进位
12  int addFlag;   // 加法进位
13  for(int i=0; i <= n-1; i++) // B的每一位
14  {
15      multiFlag = 0;
16      addFlag = 0;
17      for(int j=0; j <= m-1; j++) // A的每一位
18      {
19          // '0' - 48 = 0
20          int temp1 = (A[j] - 48) * (B[i] - 48) + multiFlag;
21          multiFlag = temp1 / 10;
22          temp1 = temp1 % 10;
23          int temp2 = (result[i+j] - 48) + temp1 + addFlag;
24          addFlag = temp2 / 10;
25          result[i+j] = temp2 % 10 + 48;
26      }
27      result[i + m] += multiFlag + addFlag;
28  }
29  reverseOrder(result, 0, m+n-1); // 逆序回来
30
31  return result;
32 }

```

最后是测试程序:

```

1  int main()
2  {
3      char A[] = "962346239843253528686293234124";
4      char B[] = "93459382645998213649236498";
5      char *res = multiLargeNum(A, B);
6      if(res[0] != 48)
7          printf("%c", res[0]);
8      printf("%s", res+1);
9      delete [] res;
10     return 0;
11 }

```

时间复杂度分析:

3个逆序操作的时间分别为 $O(n)$ 、 $O(m)$ 、 $O(m+n)$ ，双重循环的时间复杂度为 $O(mn)$ ，则总的时间复杂度为 $O(mn + (m+n))$ ，通常 $m+n \ll mn$ ，因此可近似认为 $O(mn)$ 。而且，逆序操作只是为了思考更容易，完全可以去掉。

空间复杂度为 $O(m+n)$

分类: [C/C++](#), [算法](#)

绿色通道:

[好文要顶](#)

[关注我](#)

[收藏该文](#)

[与我联系](#)



Jason Yang

[关注 - 0](#)

[粉丝 - 5](#)

[+加关注](#)

0

0

(请您对文章做出评价)

« 上一篇: [算法题-字符串循环移位问题](#)

posted on 2012-04-26 22:56 [Jason Yang](#) 阅读(2246) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

【免费课程】案例: 版本管理工具介绍---SVN篇

[【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库](#)
[融云，免费为你的App加入IM功能——让你的App“聊”起来！！](#)



最新IT新闻：

- [腾讯涉足无人机 好一个美丽的烟雾弹](#)
 - [网购退货的烦恼，这个App帮你搞定](#)
 - [窝窝团遭最后通牒：必须本月底前完成上市发行](#)
 - [马化腾：创业者不需要忧虑和恐惧腾讯](#)
 - [谷歌研究人员取得量子计算领域重要突破](#)
- » [更多新闻...](#)



最新知识库文章：

- [给公司部门设计的SOA架构](#)
 - [好代码不值钱](#)
 - [关于响应式布局](#)
 - [软件专家的对话模式（第一部分）](#)
 - [从商业角度探讨API设计](#)
- » [更多知识库文章...](#)

Copyright ©2015 Jason Yang Powered By: [博客园](#) 模板提供: [沪江博客](#)