

期中复习之高精度—乘法篇

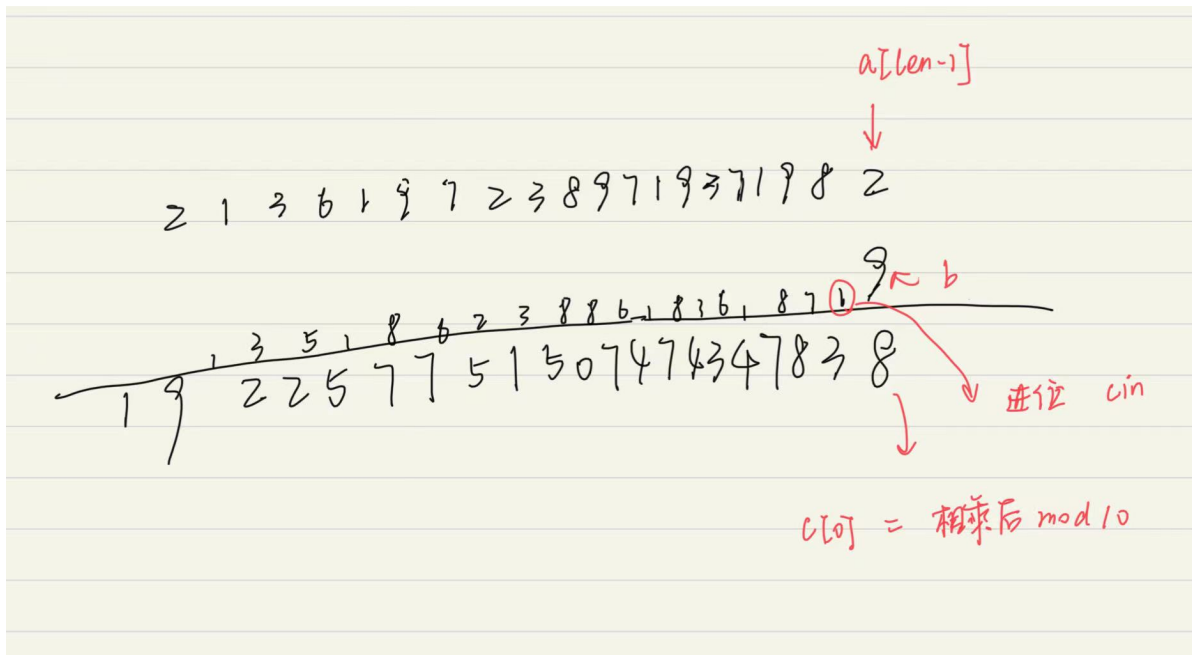
为什么只有乘法呢？因为我debug de了我好长时间（救命）

高精度乘法(A * B problem)

Part1: 高精度的数据 乘 一位的数

例如：a = 21361972389719371982, b=9 这样的乘法

(和高精度加法的处理方法类似)



代码如下：

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  char * mul_0(char* a, char b);
6
7  int main()
8  {
9      char a[20000];
10     char b;
11     char* c = (char*)malloc(sizeof(char)*12);
12
13     scanf("%s", a); //以字符串的形式读入 a
14     getchar(); //去除掉缓冲区的'\n'
15     scanf("%c", &b); //以字符的形式读入 b
16
17     c = mul_0(a,b); // 计算乘法的函数
18
19     printf("%s", c);
```

```

20
21     return 0;
22 }
23
24 char * mul_0(char* a, char b)
25 {
26     int i,j=0;
27     int cin=0;
28     int len;
29     char* c = (char*)malloc(sizeof(char)*200000);
30     char* d = (char*)malloc(sizeof(char)*200000);
31     //创建了两个超长的字符串数组来存结果。
32     len = strlen(a);
33
34     for(i=len-1;i>=0;i--){
35         c[j] = (cin + (a[i]-'0')*(b-'0')) % 10 + '0';
36         //乘法
37         cin = (cin + (a[i]-'0')*(b-'0')) /10;
38         //处理进位
39         j++;
40     }
41
42     if(cin!=0){
43         //判断最高位是否进位(是不是和加法的代码很像)
44         c[j]=cin+'0';
45         for(i=0;i<=j;i++)
46             d[i]=c[j-i];
47         //算完之后c[0]是最低位，要把他倒过来输出
48         d[i]='\0';
49     }
50     else{
51         for(i=0;i<j;i++)
52             d[i]=c[j-1-i];
53         d[i]='\0';
54     }
55     return d;//结果存在d的字符串数组中
56 }

```

所以答案就是：192257751507474347838~

注意适用范围：正整数乘法（高精度 * 1位数）并且不包含前导零且那个1位数不是0；

Part 2: 高精度 * 高精度

就是part1 的plus版本，但思路差不多

A*B Problem

题目描述

给出两个非负整数，求它们的乘积。

输入格式

输入共两行，每行一个非负整数。

输出格式

输出一个非负整数表示乘积。

样例 #1

样例输入 #1

```
1 | 1
2 | 2
```

样例输出 #1

```
1 | 2
```

提示

每个非负整数不超过 10^{2000} 。

PS：让大家体会一下用函数(结构化编程)的好处~。

比如要算：a = 12837192378, b = 12736183

The image shows a handwritten multiplication process on a digital notepad. The numbers 12837192378 and 12736183 are written, with the first number circled and labeled "简称'上面一坨'" (short name 'the top part'). The multiplication is shown with partial products and a final result of 385115771340. Red annotations explain the steps: "先用3去乘上面一坨" (First use 3 to multiply the top part), "得出这个东西" (Get this thing), "再把上面一坨后面加个'0' (乘10) 用8去乘" (Then add a '0' to the back of the top part (multiply by 10) and use 8 to multiply), "最后令(加)在一起就好了" (Finally let (add) together is good), and "这个要用高精度" (This needs high precision).

代码：

```
1 | #include <stdio.h>
```

```

2  #include <stdlib.h>
3  #include <string.h>
4
5  char* mul(char* a, char* b);
6  // 高精度 * 高精度
7  char* add(char* a, char* b);
8  // 高精度加法
9  char * mul_0(char* a, char b);
10 // 高精度 * 1位数
11
12 int main()
13 {
14     char a[5000];
15     char b[5000];
16     char* c =(char*)malloc(sizeof(char) * 5000);
17
18     scanf("%s",a);
19     scanf("%s",b);
20
21     c = mul(a,b);
22
23     printf("%s", c);
24
25     return 0;
26 }
27
28 char* mul(char*a, char*b)
29 {
30     int i;
31     int len1 = strlen(a);
32     char* c = (char*)malloc(sizeof(char) * 5000);
33     char* d = (char*)malloc(sizeof(char) * 5000);
34
35     if((strcmp(a,"0")==0)||(strcmp(b,"0")==0)){
36         c[0] = '0';
37         c[1] = '\0';
38         return c;
39     }
40     //特判a和b中有一个为0的情况
41
42     for(i=len1-1;i>=0;i--){
43         if(i==len1-1){
44             d = mul_0(b,a[i]);
45         }
46         else{
47             c = mul_0(b,a[i]);
48             d = add(c,d);
49         }
50         strcat(b,"0");
51     }
52
53     return d;
54 }
55
56 char * mul_0(char* a, char b)

```

```

57 //高精度 * 一位 乘法
58 {
59     int i,j=0;
60     int cin=0;
61     int len;
62     char* c = (char*)malloc(sizeof(char)*5000);
63     char* d = (char*)malloc(sizeof(char)*5000);
64     len = strlen(a);
65
66     for(i=len-1;i>=0;i--){
67         c[j] = (cin + (a[i]-'0')*(b-'0')) % 10 + '0';
68         cin = (cin + (a[i]-'0')*(b-'0')) /10;
69         j++;
70     }
71
72     if(cin!=0){
73         c[j]=cin+'0';
74         for(i=0;i<=j;i++)
75             d[i]=c[j-i];
76         d[i]='\0';
77     }
78     else{
79         for(i=0;i<j;i++)
80             d[i]=c[j-1-i];
81         d[i]='\0';
82     }
83     return d;
84 }
85
86 char* add(char*a, char*b)
87 //高精度加法
88 {
89     int i,j,k=0;
90     int cin=0;
91     char* c=(char*)malloc(sizeof(char)*5000);
92     char* d=(char*)malloc(sizeof(char)*5000);
93     i = strlen(a)-1;
94     j = strlen(b)-1;
95
96     while(i>=0||j>=0){
97         a[i]=(i<0)?'0':a[i];
98         b[j]=(j<0)?'0':b[j];
99         c[k]=(a[i]-'0'+b[j]-'0'+cin)%10 + '0';
100        cin = (a[i]-'0'+b[j]-'0'+cin)/10;
101        if(i>=0)
102            i--;
103        if(j>=0)
104            j--;
105        k++;
106    }
107
108    if(cin==1){
109        c[k]='1';
110        for(i=0;i<=k;i++)
111            d[i]=c[k-i];

```

```
112         d[i]='\0';
113     }
114     else{
115         for(i=0;i<=k-1;i++)
116             d[i]=c[k-1-i];
117         d[i]='\0';
118     }
119     return d;
120 }
```

结果是这个东西: 163496831332413174

注意适用范围: 正整数乘法 (高精度 * 高精度) , 且没有前导0