小马加编信息学教案(二十九)

高精度加法，乘法

一. 课程内容

1. 什么是高精度运算
2. 高精度加法
3. 高精度乘法

二. 知识讲解

1. 什么是高精度运算

> 之前讲过在C++中，有很多数据类型用于存储变量，但是变量由于所占的空间有限，若存储的数很大则只能降低数据存储的精度。比如`double`类型，当存储的数过大时，会采取科学计数法的方式，只保留高位数字，达到又能存储数据，又能尽可能保证数据可信度的效果。

> 但是在编程进行数值运算时，有时会遇到运算的精度要求特别高，远远超过各种数据类型的精度范围。有时数据又特别大，远远超过各种数据类型的极限值。当遇到这些情况时就只能使用**高精度运算**。

> 高精度运算简单来说就是用数组准确存储下每一位值，然后进行模拟竖式加法或乘法的过程，通过增加运算次数，来获得高精度，大数据的运算结果。

2. 高精度加法

2. 1 算法解析

> 假如有题目要求对两个含100位的整数进行加法操作。

> 那么首先，我们要做的就是能读入这两个数，并对他们进行存储，而我们已知的存储类型没有能够准确存储100位的整数，所以我们需要用字符串来进行读入，然后将读到的整数按**从低位到高位**存在数组a中，并用$lena$表示当前数的位数。

```
char s[105];
int a[105];
scanf("%s", s);
int lena = strlen(s);
for (int i = 0; i < lena; i ++) {
    a[lena - i - 1] = s[i] - '0';
}
```

> 同理我们将另一个数的数值和位数分别存在数组b和变量$lenb$中。

> 假设读入的两个数分别位7512996019和123456，由于我们存储的是每一位的值，并不是完整的数，所以不能直接进行加法运算，只能按位处理。那么可以模拟竖式乘法的运算过程。

	7	5	1	2	9	9	6	0	1	9
+					1	2	3	4	5	6
进位	0	0	0	1	1	0	0	0	1	0
	7	5	1	3	1	1	9	4	7	5

> 类似竖式加法，将加法结果表示成数组c和位数$lenc$，我们从低位往高位模拟，每次将对应位相加，并考虑上进位。而$lenc$很容易确定，只需考虑$max(lena, lenb) + 1$位有没有进位可以了。

> 核心代码

```
lenc = max(lena, lenb);
for (int i = 0; i <= lenc; i ++) c[i] = 0;
//清空数组
for (int i = 0; i < lenc; i ++) {
    c[i] = c[i] + a[i] + b[i];
    //c[i]之前存储的是上一位有没有进位
    if (c[i] > 10) {
        c[i + 1] = 1;
        c[i] -= 10
    }
    //如果当前位大于10，则向下一位进位
}
if (c[lenc] > 0) lenc ++:
//若第lenc + 1位有进位，则c的位数要加一
```

> 那么最后将c数组按位输出$lenc$位即可。

2. 2 时间复杂度

> 要按位枚举数组a和数组b的每一位进行加法操作，所以时间复杂度是$O(max(lena, lenb))$。

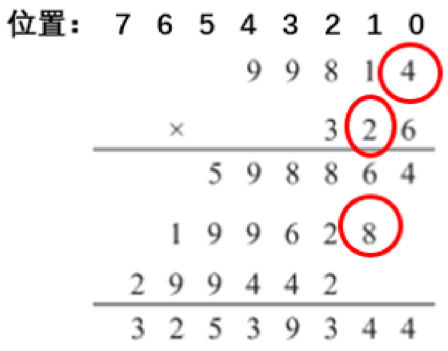
3. 高精度乘法

3. 1 算法解析

> 假如有题目要求对两个含100位的整数进行乘法操作。

> 和高精度加法同样，对于数的存储，我们用数组a，变量$lena$和数组b变量$lenb$分别存储两个整数，将结果存在数组c和变量$lenc$中。

> 由于也需要进行按位乘法，也考虑竖式乘法的运算过程。假设输入的两个数分别是99814和326。



> 我们将数从0开始编号，不难发现a的第i位和b的第j位相乘，会贡献到c的第$i+j$位。比如a的第0位4和b的第1位2相乘，就会贡献到c的第1位上。

> 那么我们只需枚举a和b的任意两位然后相乘，贡献到相应的位置上，记得考虑上进位。若最高位有进位，那么$lenc$最大就为$lena+lenb$，否则为$lena+lenb-1$

> 核心代码

```
lenc = lena + lenb;
for (int i = 0; i < lenc; i ++) c[i] = 0;
for (int i = 0; i < lena; i ++)
    for (int j = 0; j < lenb; j ++) {
        c[i + j] += a[i] * b[j];
        //由于存在进位以及c数组一个位置会被多对i,j更新，所以要进行累加
        c[i + j + 1] += c[i + j] / 10;
        c[i + j] %= 10;
        //向下一位进位
    }
if (c[lenc - 1] == 0) lenc --;
//判断lena + lenb位有没有进位
```

> 那么最后将c数组按位输出$lenc$位即可。

3. 2 时间复杂度

> 要将a数组每一位与b数组每一位分别相乘，所以高精度乘法的复杂度是$O(lena*lenb)$

三. 经典例题

1. 高精度加法

> 输入两个1000位以内的正整数，输出它们的和。

> 输入格式：
> 第一行一个1000位以内的正整数a
> 第二行一个1000位以内的正整数b

> 输出格式：
> 一行一个正整数，表示a,b的和。

样例输入	样例输出
123456789 987654321	1111111110

2. 高精度乘法

> 输入两个 100 位以内的正整数，输出它们的乘积。

> 输入格式：
> 第一行一个100位以内的正整数a
> 第二行一个100位以内的正整数b

> 输出格式：
> 一行一个正整数，表示a, b它们的积。

样例输入	样例输出
123456789 987654321	121932631112635269

3. Fibonacci数列

> Fibonacci数列满足
>
> $$f(1) = f(2) = 1$$
> $$f(n) = f(n-1) + f(n-2), n \geq 3$$
>
> 问Fibonacci第n项时多少。

> 输入格式：
> 第一行一个正整数n。
> $n \leq 800$

> 输出格式：
> 一行一个整数表示Fibonacci第n项的值。

样例输入	样例输出
20	6765

四. 提高巩固

1. n进制加法

> 输入n和两个1000位以内的n进制正整数，输出它们的和。

> 输入格式：
> 第一行一个正整数n
> 第一行一个1000位以内的n进制正整数a
> 第一行一个1000位以内的n进制正整数b

> 输出格式：
> 一行一个n正整数，表示a, b的和。

样例输入	样例输出
8 1234567 7654321	11111110

2. $n!$的精确值

> 输入n，输出$n!$的精确值，$n! = 1 \times 2 \times 3 \times \ldots \times n$。

> 输入格式：
> 一行一个正整数n。
> $1 \le n \le 1000$

> 输出格式：
> 一行一个正整数表示$n!$。

样例输入	样例输出
12	479001600

3. 回文数

> 若一个数（首位不为零）从左向右读与从右向左读都是一样，我们就将其称之为回文数。例如：给定一个十进制数56，将56加65（即把56从右向左读），得到121是一个回文数。又如，对于十进制数87：
> step1: 87 + 78 = 165
> step2: 165 + 561 = 726
> step3: 726 + 627 = 1353
> step4: 1353 + 3531 = 4884
> 在这里的一步是指进行了一次n进制的加法，上列最少用了4步得到回文数4884。写一个程序，给定一个n进制数m。求最少经过几步可以得到回文数。如果在30步以内（包含30步）不可能得到回文数，则输出$Impossible$。

> 输入格式：
> 第一行一个正整数n表示进制。
> 第二行一个n进制数m。
> $2 < n \le 10$
> m小于1000位。

> 输出格式：
> 若能在30步内完成，则输出最小步数，否则输出$Impossible$。

样例输入	样例输出
10 78	4