教学内容

- 枚举法
- 递推法
- 递归法
- **高精度计算**
- 数论
- 模拟法
- 分治法
- 动态规划
- 贪心算法
- 深搜与回溯
- 广搜与分支定界
- 博弈论
- 组合数学
- 计算几何
- 数据结构
- 图论

内容提纲

- 什么是高精度计算？
  - 高精度概念
  - 生活中高精度计算的示例
- 高精度计算的求解方法
  - 高精度计算（加、减、乘、除）
  - 高精度计算（阶乘、取余、开平方）
- 算法示例
  - 练习系统
  - 蓝桥真题

You are here!

你在这儿！

基本数据类型的取值范围

- 整数类型

类型	数值范围	数值位数	占字节数
unsigned char	0…255		1
char	-128…127		1
int(long)	-21447483648…2147483647	10^9	4
unsigned int(long)	0..4294967295		4
long long	-9223372036854775808 … 9223372036854775807	10^{18}	8
unsigned long long	0…18446744073709551615		8
long long 使用时有限制，例如，不能作为数组下标等			

0 引例 水

- 【试题描述】
  - 已知两个整数a和b。求a、b的和
- 【样例输入】

  999

  99
- 【样例输出】

  1098

引例

- 要求1：a、b的范围为：0 ~ 10^9
  - scanf("%d%d",&a,&b);
  - printf("%d\n",a+b);
- 要求2：a、b的范围为：0 ~ 10^{18}
  - scanf("%I64d%I64d",&a,&b);　　// %lld
  - printf("%I64d\n",a+b);
- 超出范围——溢出！！
  - 溢出原因：超出标准数据类型的“整型”范围

大数计算基本概念

- 某些情况下，我们必须处理位数相当多的一个整数，例如 100 位数，系统内的数据类型无论是int、long、long long等，其位数显然都不够用。
- 要解决这个问题，必须自己用程序来处理，解决方案就是用字符串来处理，即字符数组。
  - 1、字符串形式输入，并将其转化为整数数组。
  - 2、用一个整型变量记录数据的实际长度（即数组的元素个数）

大数运算基本概念

- 实际编程时，不一定要费心思去把数组大小定得正好合适，稍微开大点也无所谓，以免不小心没有算准这个“正好合适”的数值，而导致数组小了，产生越界错误。
- 一般很少会出现单独考高精度计算的题目，但是往往一些题目的答案会很大，这个时候我们必须要使用高精度计算。

字符串到整数数组的转换

- 字符串存储时，数的高位存放在字符串的低位
- 字符串s：

‘7’	‘6’	‘3’	‘2’	‘1’	‘8’	\0	\0
0	1	2	3	4	5	6	7

- 整型a：

6	8	1	2	3	6	7	0
0	1	2	3	4	5	6	7

- 高精度数的位数可存放在a[0]中。也可以另用一个变量存放。后面的题都采用第二种方式

生活中的高精度计算示例

- 生活中的高精度计算的例子通常是浮点数的精确计算，而算法中的计算主要以大整数计算为主，浮点数的精确计算为辅。
- 由于超出数据类型的有效范围，故不能采用常规方法，通常采用数组的方法或者链表的方法。我们以数组的方法为例。

内容提纲

- 什么是高精度计算？
  - 高精度概念
  - 生活中高精度计算的示例
- 高精度计算的求解方法
  - 高精度计算（加、减、乘、除）
  - 高精度计算（阶乘、取余、开平方）
- 算法示例
  - 练习系统
  - 蓝桥真题

大数加法

- 例如：189+23
- 运算过程：从右到左　　实现过程：从左到右

```
   189                    981
 + 023                  320 +
----------             ----------
   212                    212
```

- 从左到右计算的好处：
  - 最高位加法进位时，只需要往后面加1位，假如依旧从右到左进行加法，那么最后需要进位时需要把后面的数字都往后移一位，比较难以复杂和掌握。

大数加法

- 大数加法基本思路：
  - 将字符数组逆置赋给整型数组
  - 从左到右依次相加进位
  - 判断最高位是否进位，若是进位则数组长度加1
  - 最后将整型数组逆置输出

1 大数加法 易

- **问题描述**
  - 求两个不超过**200**位的非负整数的和。
- **输入数据**
  - 有两行，每行是一个不超过**200**位的非负整数。
- **输出要求**
  - 一行，即相加后的结果。结果里不能有多余的前导0，即如果结果是342，那么就不能输出为0342。
- **输入样例**
  - 22222222222222222
  - 33333333333333333
- **输出样例**
  - 55555555555555555555

算法分析

- **大数的输入与存储（Input函数实现）**
  - 将输入的文字存到一个字符串。
  - 将字串的最后一个字符转换成数字后存到个位数、倒数第二个字符存到十位数…反复这个步骤，直到所有字符的值都存进去为止。
  - 将前面有数字归零。
- 技巧
  - #define MAX 200
  - int x[MAX];
  - 将字符转换为数值：s[i]-’0’

算法分析

- 台语：阵列——数组

編號	0	1	2	3	4	5	6	7	8	9
字串	'1'	'2'	'3'	'4'	'5'	0	X	X	X	X
陣列	5	4	3	2	1	0	0	0	0	0

- 也就是说数组的数字顺序和原本的字符串是相反的，这样做的原因是当我们要做加法、乘法时，比较容易对齐每个位数。

算法分析

- **大数的输出**（**Output函数实现**）
  - 从数组的最后面找第一个不是 0 的数。
  - 从那个数开始往左边逐个把它们打印出来。
  - 打印出换行符号(非必要)。

add.c

- **大数的加法**（**Add函数实现**）

數字	編號	0	1	2	3	4	5	6	7	8
123456789	A	9	8	7	6	5	4	3	2	1
123456789	B	9	8	7	6	5	4	3	2	1
相加	相加	18	16	14	12	10	8	6	4	2
進位	進位	8	7	5	3	1	9	6	4	2

大数减法

- 例如：123-89
- 运算过程：从右到左　　实现过程：从左到右

```
   123                    321
 - 089                  980 -
----------             ----------
    34                     43
```

- 从左到右计算的好处：（类似加法）
  - 最高位退1位时，只需要往后面减去1位，假如依旧从右到左进行减法，那么最后需要减1位时需要把后面的数字都往前移一位，比较难以复杂和掌握。

大数减法

- 大数减法基本思路：
  - 将字符数组逆置赋给整型数组，并设状态量决定输出的字符串中是否需要加‘-’号
  - 从左到右依次相减和退位
  - 判断最高位是否减1位，若是则数组长度减1
  - 由状态量判断是否要在子串中加‘-’
  - 最后将整型数组逆置输出

2 大数减法 易

- 2个大的正整数相减的差。
- 输入：多组测试数据，每组分别是被减数a，和减数b(a > b)，数据不超过100个字符。
- 输出：相应的整数差。

样例输入

```
2
99999999999999999999999999999999999999
9999999999999

540965677509785089568705679806897093454654657567676867843543534
51
```

样例输出

```
99999999999999999999999990000000000000
54096567750978508956870567980689709345465465756767686784354353 44
```

算法分析

- 与大整数相加类似
- 其他函数不变，新增大数减法函数
- **大数的减法（用Substract函数实现 2736）**
  - 注意：借位问题。
  - 由于a>b，所以，无需考虑负数问题
- **如果a<b，如何处理呢？ 3115**

substract1.c

substract2.c

大数乘法

- 高精度乘以低精度
- 运算过程：

方法1：

```
    123
  *  89
----------
   1107
   984
----------
  10947
```

方法2：

```
    123
  *  89
----------
   267
  178
   89
----------
  10947
```

- 方法1或方法2，都是一个数的元素乘以另一个数

算法分析

- 大数乘法基本思路：(高精度乘以低精度)
  - 将高精度大数的字符数组逆置赋给整型数组
  - 整型数组的每个元素分别乘以低精度
  - 对整型数组进行进位运算（统一或逐一进位）
  - 最后将整型数组逆置输出
- 参见练习系统大数阶乘题目

大数乘法

- 大数乘法基本思路：（高精度乘以高精度）
  - 将字符数组逆存给整型数组
  - 然后对应位相乘，求余取模赋值给相应位
  - 新数长度为两者之和
  - 逆序输出整型数组

3 大数乘法 中

- 求两个不超过200位的非负整数的积。
- 输入：有两行，每行是一个不超过200位的非负整数，没有多余的前导0。
- 输出：相乘后的结果。结果里不能有多余的前导0，即如果结果是342，那么就不能输出为0342。

样例输入
12345678900 98765432100

样例输出
1219326311126352690000

算法分析

- 假设用int a[200]和int b[200] 分别存放两个乘数，则积**c[XXX]**长度至少应该**是多少**？
- 一个数的第i位和另一个数的第j位相乘所得的数，一定是要累加到结果的**第几位上**？
- 一个数的第i位和另一个数的第j位相乘后**是否立即处理**进位？

算法分析

- 用int a[200]和int b[200]分别存放两个乘数，用c[400]来存放积。计算的中间结果也存在c数组中。c数组长度取400是因为两个200位的数相乘，积最多有**400位（如：两位数最大是99，两个最大两位数相乘值为：9801，三位数最大是999，两个最大三位数相乘结果是：998001，六位数）**a[0], b[0], c[0]都表示个位。
- 一个数的第i位和另一个数的第j位相乘所得的数，一定是要累加到结果的第**i+j**位上。i, j都从右往左，**并从0**开始。看下面的例子，**为什么加在i+j位上**？

算法分析

- 现以835×49为例来说明程序的计算过程。
  - 先算**835×9**。5×9得到45个1（加在第**0+0**的位置），3×9得到27个10（加在第**1+0**的位置），8×9得到72个100（加在第**2+0**的位置）。由于不急于处理进位，所以835×9算完后，数组c：

陣列	0	1	2	3	4	5	
C	45	27	72	0	0	0	...

  - 接下来算**835×4**。5×4。此处4×5的结果代表20个10（加在**0+1**的位置），因此 c[1]+=20，为

陣列	0	1	2	3	4	5	
C	45	47	72	0	0	0	...

算法分析

- 再下来算3×4。此处3×4的结果代表12个100（加在**1+1**的位置），因此要 c[2]+=12，变为：

陣列	0	1	2	3	4	5	
C	45	47	84	0	0	0	...

- 最后算 8×4。此处8×4的结果代表 32个1000（加在**2+1**的位置），因此要c[3]+= 32，变为：

陣列	0	1	2	3	4	5	
C	45	47	84	32	0	0	...

- 乘法完毕，处理进位，变为：

陣列	0	1	2	3	4	5	
C	5	1	9	0	4	0	...

算法分析

- 与大整数相加类似
- 其他函数不变，新增大数乘法函数
- **大数的乘法（用Product函数实现）**
  - 统一进位：c[i+j]+=b[i] * a[j]; c[i+1]+=c[i]/10; c[i]%=10;
  - 逐一进位：c[i+j]+=b[i] * a[j] + w; w=c[i+j]/10; c[i+j]%=10;

multi1.c

multi2.c

大数除法

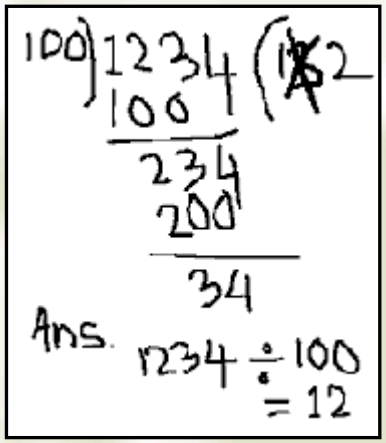
- 大数除法基本思路（高精度除以低精度）
  - 将字符数组转换为整型数组，不需要逆置
  - 从高到低依次进行除法，并记录结果
  - 从0开始找第一个不是0的位置开始赋值

4 A/B Problem 中

- 题目很简单，给你两个数a和b，求出它们的商和余数。（大数除小数）
- 输入：多组测试数据，A/B或A%B，A可能会很长，B是一个int范围的数。
- 输出：商或余

样例输入
7 2 123 5 9 3

样例输出
3 1 24 3 3 0



算法分析

- 该题目是简单**模拟题**。如：1234/100。
- 模拟步骤：
  - 读入1234中的1，除数为1，除100，不够除，商为0，余为1
  - 读入1234中的2，除数为12，除100，不够除，商为0，余为12
  - 读入1234中的3，除数为123，除100，够除，商为1，余为23，并记录商的开始位置，
  - 读入1234中的4，除数为234，除100，够除，商为2，余为34
  - 从商的起始位置，循环输出0012中的12，以及余数34。

算法分析

- 定义一个字符数组a存储大数，小数直接用简单变量b搞定，运算的结果商和余分别用整型数组c和简单变量t存放。
- 首先将字符数组中的大数a转换为整型数组，结果存放到c中，注意不要逆序存放。并用c数组的数值和b进行除和取余。
- 断点调试1234/100，查看a,t,c变量的值，其中c分别查看c[0],c[1]…

ab.c

5 大数除法 难

- 求2个大的正整数相除的商。
- 输入：多组测试数据，每组占2行，分别是被除数和除数，每行数据不超过100个字符。
- 输出：相应的整数商。

样例输入

```
3
240533731296337335900926045774205743923049649393035559579766079108273964 6
298719258531870175258442993116087037290707924897109501250979055088379319789 4

1000000000000000000000000000000000000000
10000000000

5409656775097850895687056798068970934546546575676768678435435345
1
```

样例输出

```
0
100000000000000000000000000000000
5409656775097850895687056798068970934546546575676768678435435345
```

算法分析

- 基本的思想是**反复做减法**，看看从被除数里最多能减去多少个除数，商就是多少。
- 一个一个减显然太慢，如何减得更快一些呢？
  - 以7546除以23为例来看一下：开始商为0。
  - 先减去23的100倍，就是2300，发现够减3次，余下646。于是商的值就增加300。
  - 然后用646减去230，发现够减2次，余下186，于是商的值增加20。
  - 最后用186减去23，够减8次，因此最终商就是328。
- 所以本题的核心是要写一个大整数的减法函数，然后反复调用该函数进行减法操作。

算法分析

- 大数的减法函数
  - int Substract(int p1[], int p2[],int nlen1,int nlen2)

  函数参数说明：nlen1，数组p1的长度；nlen2，数组p2的长度，函数返回值为减后的数组长度或-1，减后的商存放到p1中，-1表示不够减
  - 相减判断：如果nlen1>nlen2，肯定够减，减吧，如果nlen1==nlen2，逆序判断，如果p1[i]>p2[i]，够减，减吧。否则，不够减，返回-1
  - 相减操作，同前，减法借位

算法分析

- 模拟除法：7546/23
- 步骤：
  - 求nlen1=4和nlen2=2
  - 两个不一样长度，求差值，得到2，23左移2位变成2300，表明，7546可以减n个23的100倍。
  - 循环减吧，减3个级别，百位，十位，个位。
  - n个100位是多少？还是循环减吧，直到不够减即可。

divide.c

6 比大小 水

- 给两个大数，能否判断出他们的大小呢？比如123456789123456789要大于-123456。
- 输入：每组两个不超过1000位的10进制整数a,b，数据保证输入的a,b没有前缀的0。
- 输出：如果a>b则“a>b”，如果a<b则“a<b”，如果相等则“a==b”。

样例输入

```
11111111111111111111111111 88888888888888888
-111111111111111111111111  22222222
0 0
```

样例输出

```
a>b
a<b
```

算法分析

- 本题不难，用两个字符数组存储字符串，然后，判断两个字符串是否同为正数或负数，或一正一负。
  - 如果一正一负，则正数大
  - 如果两个都为正或者两个都为负，则比较两个字符串长度。
    - 正数时，长度长的大，长度短的小；
    - 长度相同时，逐个比较数组元素。

cmp.c

7 浮点数加法 中

- 输入输出中出现浮点数都有如下的形式：$P_1P_2...P_i.Q_1Q_2...Q_j$，
  对于整数部分，$P_1P_2...P_i$是一个非负整数
  对于小数部分，Q_j不等于0
- 输入：多组测试数据，每组两个加数。
- 输出：相应的和。

样例输入

```
2
0.11111111111111111111111111111
0.11111111111111111111111111111

10000000.65555555555555555555555555555
1.44444444444444444444444444445
```

样例输出

```
0.22222222222222222222222222222
10000002.1
```

算法分析

- 这道题真“坑爹”，首先，原题中说每组测试数据间有空行，胡扯。其次，原题中的“保证一定是一个小数部分不为0的浮点数”这句话，无视。测试数据中有不是浮点数情况
- 整数部分的处理同大数加法
- 小数部分，顺序存储，则从后向前模拟相加，便于进位，另外小数部分范围应选取最长的小数长度。

flaotadd.c

8 火星上的加法 中

- 在22世纪，科学家发现在火星上居住着智能程度很高的居民。火星人很喜欢做算术题。现在任务是计算两个100位数的和，谁用的时间最少，谁将是胜利者。需要告诉你的是，火星人用的是20进制。
- 输入：20进制采用的数码是0~9，以及小写字母a~j，小写字母分别代表十进制中的10~19。
- 输出：这两个数的和。

Sample Input:

```
1234567890
abcdefghij
99999jjjjj
9999900001
```

Sample Output:

```
bdfi02467j
iiiij00000
```

算法分析

- 输入字符串，并保存到一个字符数组中。转换并逆置到整形数组中。
- 火星人是20进制，所以，数字字符转换的时候需要 a[i]-’0’，不是字母，需要a[i]-’a’+10。
- 按照加法的法则进行计算。
- 20进制的输出，<10，直接输出，否则，+’a’-10，另外，注意必须注意一些比较特殊的情况。例如如果输入是两个0，必须能打印出 “0”。

martian.c

9 大数开根 难

- 输入一个正整数N(1≤N≤10^100)，试用二分法计算它的平方根的整数部分。
- 输入：一个大整数N。
- 输出：一个数，N的平方根的整数部分。
- 本题为填空题。

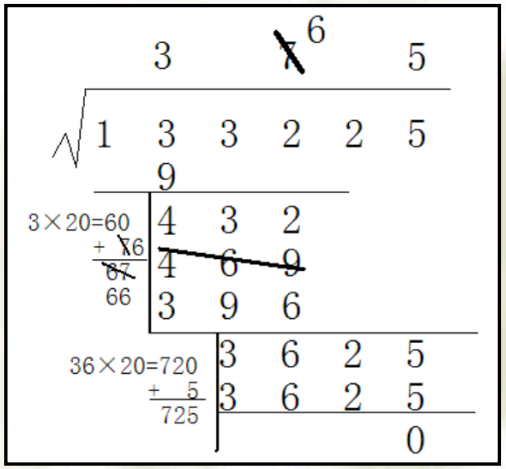
算法分析

- 方法1：二分法，求√121= ?
  - 初始化：left=0，right=121
  - 并计算：middle=(left+right) / 2 = 60
  - 如果middle²>n 则更新right，否则更新left，显然60*60>121，更新right=60
  - 计算middle=(0+60)/2=30，显然30*30>121，继续更新right=30
  - 计算middle=(0+30)/2=15，显然15*15>121，继续更新right=15
  - 计算middle=(0+15)/2=7，显然7*7<121，更新left=7
  - .....
  - 最后left=[√n]

sqrt1.c

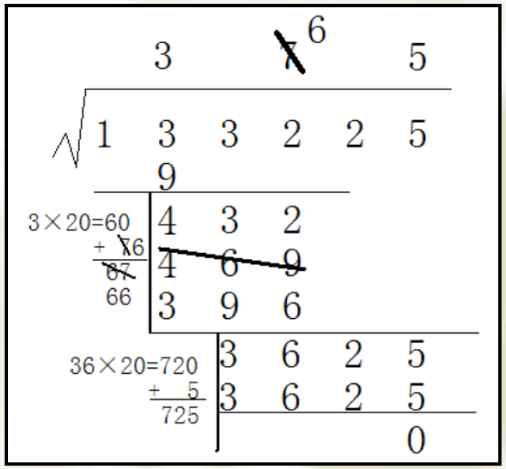
算法分析

- 方法2：模拟法：手动开根号
  - 1、将被开方数分节。从后向前分，2位一节
  - 2、取小于等于第一节数字且与它最接近的平方数，写在第一节的下面（右对齐）
  - 3、将这个数开方，写在第一节上面（右对齐）
  - 4、将第一节与平方数做差
  - 5、落下下一节



算法分析

- 基本方法：
  - 6、用根号上面的**所有数乘20**，写在算式左面（划分隔竖杠），用竖杠左面的数**加上这个数**，再乘这个数，得数写在被减数下面，进行运算。注，如果次数大于被减数，则将第第二节上面的数减1，重复第6步
  - 重复5、6步即可算出。



sqrt2.c

10 麦森数 难

- 形如2^p-1 的素数称为麦森数，这时p一定也是个素数。但反过来不一定，即如果P是个素数。2^p-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是p=3021377，它有909526位。你的任务：输入p，计算2^p-1 的位数和最后500 位数字（用十进制高精度数表示）。
- 输入：整数p(1000<p<310
- 输出：十进制高精度数2^p
精度数2^p-1的最后500位数
共输出10 行，不足500 位

样例输出

```
386
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000001040793219466439908192524032736408553
86152622472667048053191123504036080596733602980123
23944173232418484242161395428100779138356624832346
49081399066056773207629241295093892203457731833496
61583550472959420547689811211693677147548478866962
50138443826029173234888531116082853841658502825560
46662248318909188018470682222031405210266984354887
32958028878050869736186900714720710555703168729087
```

算法分析

- 第一问是很简单的，只需要求一个对数而已，数学原理：十进制正整数n的位数为(int)log10(n)+1。所以2^P-1的位数(int)log10(2)*p+1 。
- 第二问求2^p-1的最后500位数字。
  - 在前面的高精度计算中，我们用数组来存放大整数，数组的一个元素对应于十进制大数的**一位**。如果本题也这样做，就会超时。怎么做呢？
  - **2分求幂法（反复平方法）、分段表示法（压位法）**

分段表示法

- **分段表示**法，也称作高精度计算的**压位**法。
  - 用一个数组元素对应于大整数的4位，即将大整数表示为10000进制，也就是说逢10000进1。
  - 如，用int 型数组a来存放整数6373384，那么只需两个数组元素即可，a[0]=3384，a[1]=637。俗称压位，压了4位。
  - 求最后500位数字。因为每个数组元素存放十进制大整数的4位，所以本题中的数组最多只需要125个元素。

算法分析

- 如，求2^1279，通常做法是循环求解，一个2一个2乘，由于p是大数，p<3100000，转p圈，什么都不用做，就直接就超时了。
- 既然不是循环一个一个相乘，那怎么乘呢？答案：***2分求幂法，反复平方法***。
  - **2**^1279=2***4**^639=2*(4***16**^319)=2*(4*(16***256**^159))=……

算法分析

- 所以算法伪代码循环如下

```
while(p)     // 2^p     p=1279，  初始化：x=2
{
    if（p是奇数）          第一圈              第二圈
        ans*=x          // ans初始化为1       ans=4
    p=p/2               // p→639              p→319
    x=x*x               // x=4                x=16
}
```

mason.c

内容提纲

- 什么是高精度计算？
  - 高精度概念
  - 生活中高精度计算的示例
- 高精度计算的求解方法
  - 高精度计算（加、减、乘、除）
  - 高精度计算（阶乘、取余、开平方）
- 算法示例
  - **练习系统**
  - 蓝桥真题

阶乘的求解方法

- 12以内的阶乘
- 20以内的阶乘
- 大数阶乘

1 12以内的阶乘

水

- 求12以内n的阶乘
- 输入：n（n<=12）。
- 输出：数值n！。

样例输入
3

样例输出
6

```
int i,n,f=1;
scanf("%d",&n);
for(i=1;i<=n;i++)
        f*=i;
printf("%d\n",f);
```

2 20以内的阶乘

水

- 求20以内n的阶乘。
- 输入：n（n<=20）。
- 输出：数值n!。

样例输入
16

样例输出
20922789888000

```
int i,n;
__int64 f=1;
scanf("%d",&n);
for(i=1;i<=n;i++)
    f*=i;
printf("%I64d\n",f);
```

递归与数学求解

递归求解

```
__int64 f(int n)
{
    if(n<=1)
        return 1;
    else
        return n*f(n-1);
}
```

数学公式求解

斯特林公式

```
double pi=acos(-1);
scanf("%d",&n);
printf("%.0lf\n",(sqrt(2*pi*n)*
pow(n/exp(1),n)*10+0.5)/10);
```

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

3 大数阶乘 易

- 我们都知道如何计算一个数的阶乘，可是，如果这个数很大呢，我们该如何去计算它并输出它？
- 输入：整数m(0<m<=5000)。
- 输出：m的阶乘。

样例输入

```
50
```

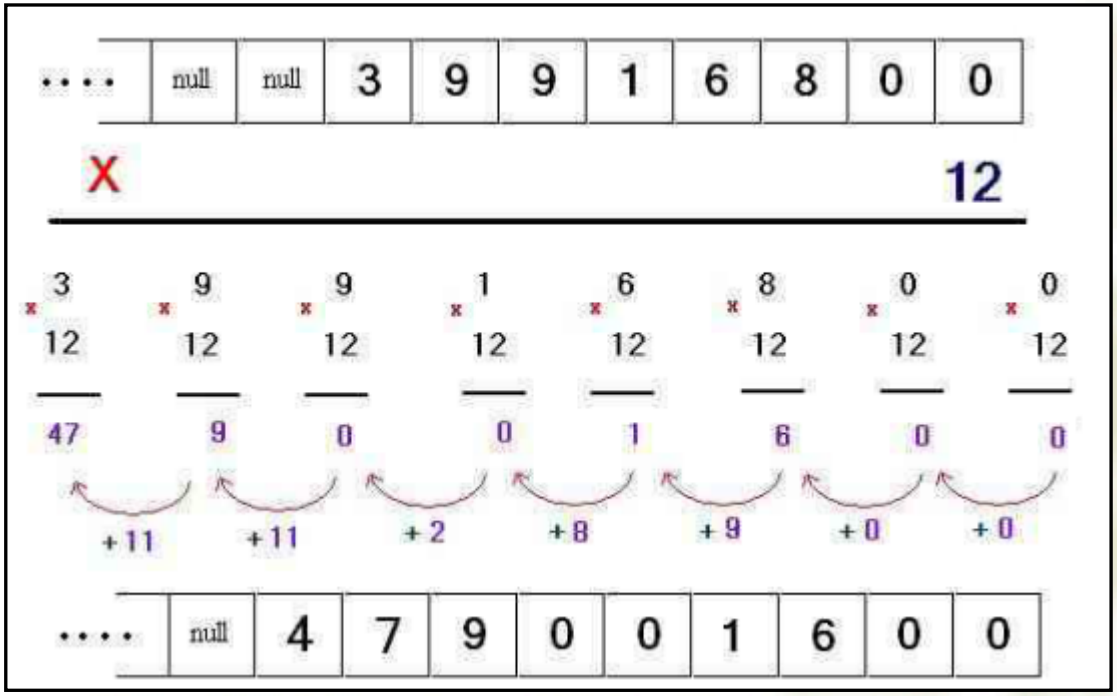
样例输出

```
30414093201713378043612608166064768844377641568960512000000000000
```

算法分析

- 数组模拟
- 例如：求12！

```
12!  = 11! * 12
11!  = 39916800
12!  = 479001600
```



算法分析

- 分段表示法
  - 表示一个大数1234567890…1234567890，则可表示为a=1234567890,…,n=1234567890，最后大数则为abc…lmn。以十位数为分割。
  - 假设数组元素最大9999，以四位数分割，超过9999，则进位到下一个元素，最后从后向前输出，不足4位的补0。

名称	值
a	0x0042ca30 _a
[0]	0
[1]	0
[2]	0
[3]	512
[4]	6896
[5]	6415
[6]	4377
[7]	6884
[8]	647
[9]	8166
[10]	1260
[11]	436
[12]	3378
[13]	171
[14]	932
[15]	414
[16]	3

4 回文数 易

- 若一个数从左向右读与从右向左读都一样，则称为回文数。例如：给定一个10进制数56，56加65得到121，是一个回文数。又如：10进制数87：STEP1：87+78 =165；STEP2：165+561=726；STEP3：726+627=1353；STEP4：1353+3531=4884，这里一步是指进行了一次N进制的加法，上例最少用4步得到回文数4884。给定一个N进制数M，写程序，求最少经过几步可以得到回文数。
- 输入：N与M
- 输出：30步以内得到回文数，输出STEP=xx，xx为步数；否则输出Impossible!

样例输入	样例输出
9 87	STEP=6

算法分析

- 字符数组判断回文，不难，两头向中间走
- 大数相加，采用数组相加，且进位
- 进制转换，判断字符是否是数字，如果是数字则数组元素s[i]-’0’，否则为字母A~F，则数组元素s[i]-’A’+10

内容提纲

- 什么是高精度计算？
  - 高精度概念
  - 生活中高精度计算的示例
- 高精度计算的求解方法
  - 高精度计算（加、减、乘、除）
  - 高精度计算（阶乘、取余、开平方）
- 算法示例
  - 练习系统
  - **蓝桥真题**

本栏目木有啦

作业

- 整数研究
- 初等算术
- 反转数的大数相加
- A-B Problem
- 总和
- Product
- A/B Problem
- 大数阶乘
- 天使的起誓
- 循环数

1 整数研究 易

- 十进制大整数的加法运算。
- 输入：多组测试数据，每组数据每行为一个非常长的十进制整数组成，输入0表示结束。
- 输出：输出它们的和。

样例输入

```
123456789012345678901234567890
123456789012345678901234567890
123456789012345678901234567890
0
```

样例输出

```
370370367037037036703703703670
```

算法分析

- 字符数组读入大整数
- 将字符数组的数字逆序存储到整型数组中
- 进行相加运算，并进位，进位可以采用统一进位，也可以采用逐一进位

2 初等算术 易

- 小学生在学多位数加法时，是将两个加数右对齐，然后从右往左一位一位地相加。多位数的加法经常有进位。如果对齐的两位相加结果大于等于十就给左边一位进一。对小学生来说，进位的判断是比较难的。你的任务是：给定两个加数，统计进位的次数，从而帮助老师评估加法的难度。
- 输入：每一行是两个无符号整数，少于10位。
- 输出：计算两个加数进行加法运算时进位的次数。

样例输入

```
123 456
555 555
123 594
0 0
```

样例输出

```
No carry operation.
3 carry operations.
1 carry operation.
```

算法分析

- 与大数加法类似，进位后需要记录大整数的长度，另外，注意情况考虑全面，且输出的英文有单双数之分。
- 参考测试数据：
  - 0　5
  - 999999　8
  - 901　99

样例输出

```
No carry operation.
3 carry operations.
1 carry operation.
```

3 反转数的大数相加 易

- 给任意的两个数，要你求出其两个数颠倒过来的数相加的结果，然后在颠倒输出。
- 输入：多组测试数据。
- 输出：大数反转后相加的结果，逆序输出。

样例输入

```
3
24 1
4358 754
305 794
```

样例输出

```
34
1998
1
```

算法分析

- 与大数相加类似，写一个反转大数的函数即可。
- 本题后台测试数据较弱，不用大数依然可以实现。

4 A-B problem 易

- 有两个实数A和B，能不能判断出A-B的值是否等于0呢？
- 输入：多组测试数据，每组A和B。数字前可能包含+,- 号。数字前后都可能有多余的0。
- 输出：如果A-B=0,输出YES,否则输出NO。

样例输入

```
1
1

1.0
2.0
```

样例输出

```
YES
NO
```

算法分析

- 找到小数点
- 去除小数点后的0
- 分别处理 正号 +、负号 - 与无符号的情况

5 总和 易

- 给定一组标准格式的货币金额，计算其总和。标准格式为：
  - （1）每个金额以符号“$”开头；
  - （2）仅当金额小于1时，金额有前导0；
  - （3）每个金额小数点后有两位数；
  - （4）金额小数点前的各位，以3位一组进行分组，且以逗号隔开，最前面的一组可能只有1位或2位。
- 输入：多组测试数据。N个金额。
- 输出：总和。

Sample Input

```
2
$1,234,567.89
$9,876,543.21
3
$0.01
$0.10
$1.00
0
```

Sample Output

```
$11,111,111.10
$1.11
```

算法分析

- 将读入的字符串，去掉美元符号、逗号、点号逆序存放到整型数组中
- 大数求和
- 输出时，请考虑周全，如：需考虑0、0.01和0.22 这些情况

6 Product 中

- 这个问题是请你做2个整数 X, Y相乘（ 0 <= X,Y < 10250 ）
- 输入： 多组测试数据，X 和 Y
- 输出： 输出 X*Y 的结果

- 算法分析
  - 类似大数乘法，注意一个数为0时的处理。

7 A/B Problem 中

- 做了A+B Problem，A/B Problem不是什么问题了吧！
- 输入：多组测试数据，每组一个号码A，然后一个符号（ / 或者 % ），再是一个号码B，A可能会很长，B是一个int范围的数。
- 输出：输出结果

样例输入

```
110 / 100
99 % 10
2147483647 / 2147483647
2147483646 % 2147483647
```

样例输出

```
1
9
1
2147483646
```

算法分析

- 同大数除法，高精度除以低精度。参看前面例题分析。

8 大数阶乘 中

- 求10000以内n的阶乘。
- 输入：整数n（n<=10000）。
- 输出：输出n!，每50位输出一行，位数不足50时，前面补零。

样例输入

```
100
```

样例输出

```
00000000000000000000000000000000000000000093326215
44394415268169923885626670049071596826438162146859
29638952175999932299156089414639761565182862536979
20827223758251185210916864000000000000000000000000
```

9 天使的起誓 难

- TENSHI被选为掌管智慧钥匙的天使。正式任职前要宣誓，宣誓的发言稿被放在N个呈圆形排列的宝盒中。按顺时针方向被编上号码1,2,…,N。开始天使们站在编号为N的宝盒旁。她们各自手上都有一个数字，代表她们自己的发言稿所在的盒子是从1号盒子开始按顺时针方向的第几个。例如：有7个盒子，如果天使手上的数字为9，那么她的发言稿就在第2个盒子中。请帮助TENSHI找到她想找的宝盒的编号。
- 输入：正整数N和M，其中2≤N≤10^8，2≤M≤10^1000
- 输出：宝盒的编号

算法分析

- 从题目中可以明显地看出来，此题就是要求二个数的余数，而且从题目中也可以看出数据的范围应该是高精度除以整型的数，而余数的求解也可以用高精度数除以低精度数的x的值，就是余数（见代码）。当然，此题中当余数为0时，编号应该为盒子的数目。

10 循环数 难

- n位的一个整数是循环数的条件是：当用一个1到n之间的整数去乘它时，会得到一个将原来的数首尾相接循环移动若干数字再在某处断开而得到的数字。也就是说，如果把原来的数字和新的数字都首尾相接，他们得到的环是相同的。只是两个数的起始数字不一定相同。例如，数字142857是循环数，因为：142857 *1 = 142857、142857 *2 = 285714、142857 *3 = 428571、142857 *4 = 571428、142857 *5 = 714285、142857 *6 = 857142
- 输入：包括多个长度为 2 位到 60 位的整数
- 输出：判断它是否是循环数

算法分析

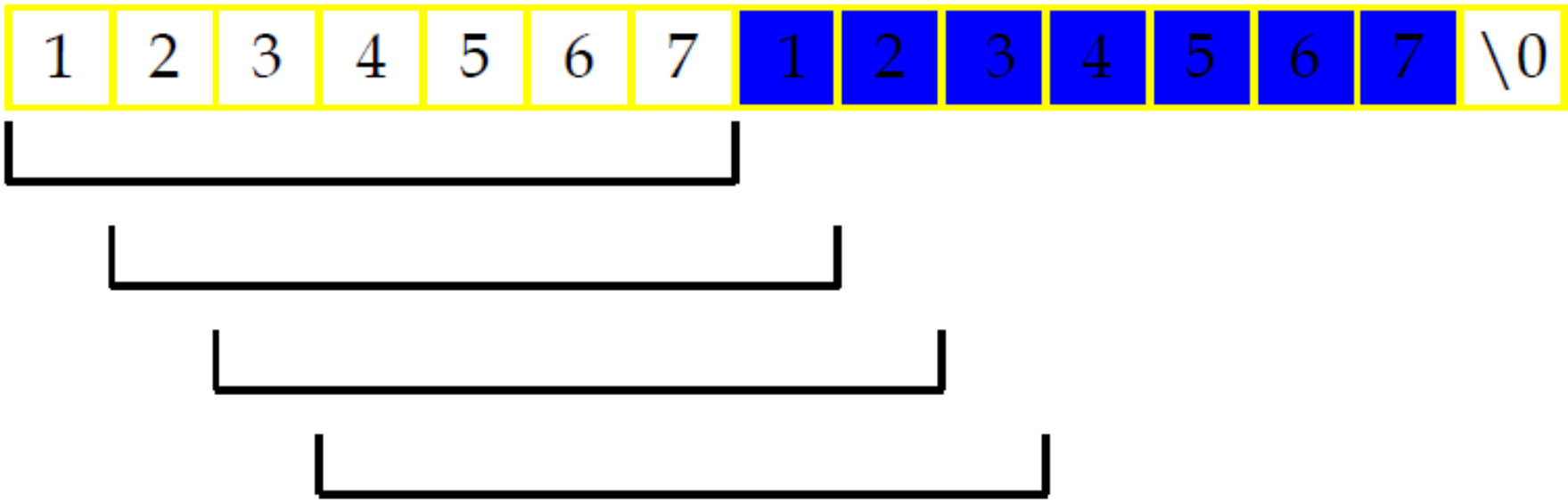
- 整数可能达60位
  - X*1: $Y_0 = X;$
  - X*2: $Y_1 = Y_0 + X$
  - X*3: $Y_2 = Y_1 + X$
  - ……
- Y_i是否是X的“循环”？

算法分析

穷举：N位整数，循环移位可以有N种可能

循环移位方法：

Y_i是否是“XX” 的子串？



下课了