

CST2016 1-1 BigInt

描述

邓俊辉老师的作业常常过于简单，数据类型只需使用int。助教们一致认为，向同学们介绍Python中自带的长整型是十分有必要的。例如，它可以计算几百位的整数乘法。但是，在介绍长整型之前，助教决定让你自己实现一遍长整型乘法，以加深对它的理解。

输入

输入共包含 $n+1$ 行，第1行包含一个整数 n ，表示你需要计算 n 组乘法。

接下来 n 行，每行包含两个非负整数 a 和 b 。

输出

输出共包含 n 行，请对于每一组输入的 a 、 b ，输出他们的乘积。

输入样例

```
3
1 1
2 2
123123 789789
```

*此样例是第1个测试点。

*本课堂的编程作业中，对于非全int输入的题目，有的会把一个样例作为第1个测试点方便调试。

输出样例

```
1
4
97241191047
```

*注：参考答案的每一行都有换行符，最后一行也有换行符。但OJ比对文本时一般会忽略行末空格和文末空行。

数据范围

$n \leq 500$

$a < 10^{5000}$

$b < 10^{5000}$

资源限制

时间限制: 1 sec

内存限制: 256MB

提示

对于所有问题, 请尽力优化你的算法并确保其运行正确。用于黑盒测评的19,20两个测例会是整个问题中规模最大 (或者对边界条件最高) 的两个测例。

即便是64位整型, 由于存储字节有限, 所以不能完整表示一个很大整数的精确值 (它只能区分 2^{64} 个不同的数值), 无法表示两个乘数, 遑论计算。而对于更大整数的运算, 这时候就得用到其他的方法, 我们称之为高精度算法。例如我们考查高精度加法, 一种最简单的思路如下

例:

```
12345678910111213 + 11111111111111111111
```

使用两个数组存储:

```
a[]={3,1,2,1,1,1,0,1,9,8,7,6,5,4,3,2,1};  
b[]={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
```

两个数组分别把数值倒存, 逐位相加, 每位加后判断是否大于10, 再进行进位即可。

对于高精度的乘法, 问题相应来说会复杂一些, 类似于刚才的加法思路, 这一过程中有这样一些问题值得考虑:

1. 对于这两个整数应当如何存储? 2进制? 10进制? (或者其它) 低位在前? 高位在前? (或者其它)
2. 大整数乘法本身有什么算法 (可以自行查阅相应资料)? 什么样的算法能够较好地适应这个问题的数据规模?