### OOP第二周作业文档

2019010175 孔瑞阳 土木92

### 功能说明

输入两个正整数a,b。

当整数a,b不是正整数的时候，提示输出的不是正整数，并要求重新输入。

先输出a,b的最大公约数。

然后输出计算最大公约数的时间。

然后输出a,b的最小公倍数。

最后输出计算最小公倍数的时间。

### 模型

##### 欧几里得算法

首先，记(a,b)为a,b的最大公约数，[a,b]为a,b的最小公倍数。

那么可能显然地得到若干个性质。

1. (a,a) = (a,0) = a
2. (a,b) = (a,a+b) = (a,ka+b)
3. (a,b) = (a, b mod a) （结论 (2) 的推论）

多次运用上述性质，即可求得两个数的最大公约数。

##### 算法实现

设置一个函数int gcd(int a, int b)，用来计算a和b的最大公约数。

根据性质(3)，我们就可以得到gcd(a,b) = gcd(b, a mod b)。

利用递归算法不断进行递归，直到b=0，那么根据性质(1)，就可以得到gcd(a,b)=a。

再利用 [a,b] = a \* b / (a,b)，可以求出a和b的最小公倍数。

（为了防止结果不溢出但是a\*b溢出的情况，实际计算时可以表示为a / (a,b) \* b）。

关于计时，采用clock() 函数，算法运行前后的clock差就是算法所花费的时钟周期数。

那么再除以CLOCKS\_PER\_SEC，即可得到算法运行的时间（秒数）。

##### 算法时间复杂度的证明

引理：若a <= b，则b mod a < b/2。

引理的证明：

1. 若 a <= b / 2，则 b mod a < a <= b / 2，结论成立。
2. 若 b / 2 < a <= b，则 b mod a = b - a < b / 2，结论成立。

也就是说，我们每进行一次递归，a和b中的某一个数就会减少为原来的1/2，也就是说，总共只会进行O(log a + log b)次递归，那么算法的时间复杂度就是O(log a + log b)。

### 验证

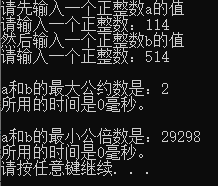
##### 关于输入内容的验证

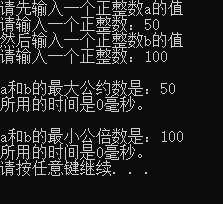
由于程序的输入内容直接使用的上周作业已经编写过的代码，并且在上周文档中已经叙述过输入验证的内容，所以在此不再赘述。

##### 关于计时功能的验证

第一步：对于比较小的数据进行手算。

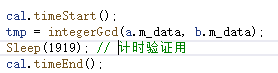
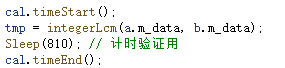
首先，根据 二、3 的内容，计算最大公约数是非常非常快的，以至于我们如果直接在源程序上输出验证，只会得到以下这样子的内容：





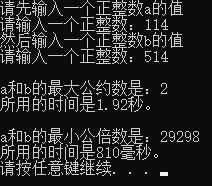
也就是说会直接显示0毫秒。

那么为了验证这个部分的正确性，我们可以利用Sleep函数，在计时周期内不仅考虑计算gcd/lcm的时间，再加上一段人为规定的时间。

不妨设置为1919ms和810ms。如果计时功能正确，那么最终的输出也应该是这个时间。

输出结果如下：



符合预期，计时功能没有出现问题。

##### 关于计算gcd, lcm的正确性验证

首先，对于规模较小的数据，可以直接通过枚举的方法判断递归算法的正确性。

具体地来说，不妨先验证a,b <= 100的所有数，我们可以先通过性质3的递推用一个二维数组把所有情况的gcd先生成出来，再一个个通过编写的gcd程序判断两种方法的答案是否相同。



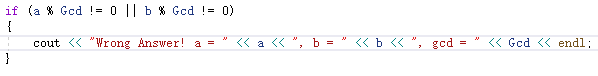
如果没有问题，那么程序就会输出Accept!

QQ截图20200303103926 没有问题。

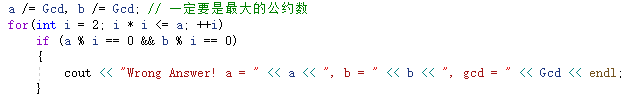
事实上，由于gcd算法是一个一般性的算法，并且较小的数据已经包括了若干特殊情况（比如a=b, a=0），所以在较小的数据下没有问题的话基本上可以证明算法和程序正确。

而当数据规模比较大的时候，不管是对于时间还是空间，枚举不再适用。那么我们可以考虑多次随机生成一组数来验证程序的正确性。（自动验证？）

首先gcd一定要是a和b的公约数。



其次，为了要是最大的公约数，所以a/gcd和b/gcd是没有公约数的，可以枚举a/gcd的公约数进行验证。



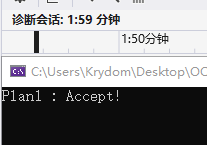
如果随机若干次都没有问题，那么就基本可以保证在大数情况下的正确性。

不妨先随机个5000次。 QQ截图20200303105624

QQ截图20200303105703 依然没有问题。

QQ截图20200303105810

接下来，我们试着：直到出现错误的时候验证停止。如果没有出现错误，那么程序将一直进行下去。



经过简单计算，1s至少可以验证1w次，那么通过2分钟之后，已经进行了上百万次随机验证，都没有出现问题。

以上，基本可以验证算法以及程序的正确性。

以及Lcm，由于公式简单，肉眼观察法以及上述过程中已经进行过的简单验证即可保证其的正确性。

QQ截图20200303111607

##### 关于以上验证的充分性和有效性

首先，对于计时功能，由于功能十分单一，进行简单的测试即可。

对于gcd验证的有效性：

在验证方案1中，采用了两种不同方法计算。具有有效性。

在验证方案2中，分别验证了一组数据的最大公约数，一定是公约数，并且是最大的一个公约数，保证了验证的有效性。

对于gcd验证的充分性：

对于小部分数据进行枚举，验证了某些特殊情况的正确性，并且可以基本保证算法和程序的正确性没有问题。同时对于大数据进行上百万次随机验证，并且算法本身比较简单，基本上可以保证验证的充分性。