运行说明

本程序只额外导入了random包，无其它依赖文件，可在IDE中直接运行。

实验步骤

1. 素数p的生成

```python
def is_Prime(num):
    # 小素数列表
    smallPrime=[
        2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,103,107,109,
        241,251,257,263,269,271,277,281,283,293,307,311,313,317,331,337,347,349,353,359,367,373
        421,431,433,439,443,449,457,461,463,467,479,487,491,499,503,509,521,523,541,547,557,563
        607,613,617,619,631,641,643,647,653,659,661,673,677,683,691,701,709,719,727,733,739,74
        809,811,821,823,827,829,839,853,857,859,863,877,881,883,887,907,911,919,929,937,941,947
        ]
    if(num < 2):
        return False

    if num in smallPrime: # 在表中
        return True

    for prime in smallPrime:
        if(num % prime == 0):
            return False

    return(Miller_Rabin(num))
```

使用Miller_Rabin函数来进行素性测试。

```python
def Miller_Rabin(num): # 素性测试
    safeTime = 128
    s = num - 1
    t = 0
    while(s % 2 == 0):
        s >>= 1
        t = t + 1

    for trials in range(safeTime):
        random_a = random.randrange(2, num - 1)
        v = pow(random_a, s, num) # random_a ** s % num
        if v != 1:
            i = 0
            while v != (num - 1):
                if i == t - 1:
                    return False
                else:
                    i = i + 1
                    v = (v ** 2) % num
    return True
```

2. 求解素数p的本原根

```python
def Generate_BenYuanGen(): # 生成本原根
    global p, q
    p = Output_Safeprime_p()
    q = p * 2 + 1
    while True: # 从最小值开始寻找本原根
        for g in range (2, q - 2): # 前闭后闭
            if(FastExpMod(g, 2, q) != 1 and FastExpMod(g, p, q) != 1):
                return g
```

其中注意从最小值开始遍历，当有多个本原根时，默认取最小值。

3. 基于蒙哥马利快速模幂运算计算通信双方的公钥 Ya 和 Yb，验证是否符合条件

```python
def FastExpMod(YB, XA, q):
    power = 1
    while XA:
        if XA & 1:
            power = (power * YB) % q
        XA >>= 1
        YB = (YB * YB) % q
    return power
```

运行效果