部分乘法算法的实现及效率比较

由于时间关系，以及不清楚具体要实现哪些算法。所以本次作业仅实现此次了整数乘法的两个算法，分别是运算数扫描方式的整数乘算法（**记为算法 1**）和积扫描的整数乘算法（**记为算法 2**）。

实现这些算法的感受

这些算法都是比特级别的算法，不太适合软实现，因为用编程语言实现的时候，往往需要将整数转化成比特级别的数据进行操作，这样的转化反而消耗了更多的时间，使得效率的比较也更加困难，因为我们无法知道这个过程是将数据转成比特级别的操作耗时，还是算法中哪些为了进行乘法运算所进行的操作更加耗时。

具体的代码见最后。

代码计算 24*32 所消耗的时间比较

如下为代码的运行结果。

768
0.00013208389282226562
768
9.107589721679688e-05

第一、三行分别为算法 1、算法 2 的计算结果。

第二、四行分别为算法 1、算法 2 所消耗的时间。

可以看出算法 1 所消耗的时间比算法 2 更多。

多次运行该算法，则大多数情况下算法二的速度也要更快，偶尔会出现算法二更慢的情况。但事实上这两者的算法效率应该是相当的，都是 O（n^2）级别。

源代码：

```
import time
T = 10
def int_multi_op(a,b):
        C = [0 for i in range(2*T)]
        A = bin(a).replace('0b','')[::-1].ljust(T,'0')
        A = [int(x) for x in A]
        B = bin(b).replace('0b','')[::-1].ljust(T,'0')
        B = [int(x) for x in B]
        for i in range(T):
                u = 0
                for j in range(T):
                        uv = C[i+j] + A[i]*B[j] + u
                        v = uv % 2
                        u = uv >> 1
                        C[i+j] = v
```

```
                C[i+T] = u
        c = ''.join(map(str, C))[::-1]
        c = int(c, 2)
        return int(c)

def int_multi_prod(a, b):
        C = [0 for i in range(2*T)]
        A = bin(a).replace('0b','')[::-1].ljust(T,'0')
        A = [int(x) for x in A]
        B = bin(b).replace('0b','')[::-1].ljust(T,'0')
        B = [int(x) for x in B]

        r0 = 0
        r1 = 0
        r2 = 0

        for k in range(2*T - 2):
                for i in range (T - 1):
                        j = k - i
                        if not j in range(T): break
                        uv = A[i]*B[j]
                        v = uv % 2
                        u = uv >> 1

                        er0 = r0 + v
                        r0 = er0 % 2
                        epsilon = er0 >> 1

                        er1 = r1 + u + epsilon
                        r1 = er1 % 2
                        epsilon = er1 >> 1

                        r2 = r2 + epsilon
                        # print(r2)
                C[k] = r0
                r0 = r1
                r1 = r2
                r2 = 0
        C[2*T - 1] = r0

        c = ''.join(map(str, C))[::-1]
        c = int(c, 2)
        return int(c)
```

```
start_time = time.time()
print(int_multi_op(24,32))
print(time.time()-start_time)

start_time = time.time()
print(int_multi_prod(24,32))
print(time.time()-start_time)
```