实验报告

一、 小学整数乘法

1. 伪代码

gradeSchoolMult (x, y):

- Write x = $a_1*10^{n-1}+a_2*10^{n-2}+a_3*10^{n-3}......+a_{n-1}*10+a_n$
- compute xDigits*y

$b_1= a_1*y*10^{n-1}$

$b_2= a_2*y*10^{n-2}$

……

$b_n= a_n*y$

- Add them up to get the sum

sum = $b_1+b_2+......+b_n$

2. 具体的实现代码

```
# look at each pair of digits, and add them up with appropriate shifts.
def gradeSchoolMult( X, Y ): # X and Y are integers
    x = getDigits(X)
    y = getDigits(Y)
    summands = []
    for xDigit in range(len(x)):
        currentXDigit = x[len(x) - xDigit -1]
        z = [0 for i in range(xDigit)] # z is the digits of xDigit times y; start it out with some zeros
        carry = 0
        for yDigit in range(len(y)):
            newProd = getDigits( currentXDigit * y[len(y) - yDigit - 1] + carry )
            z.insert( 0, newProd[-1] ) # put the new digit at the front of our new summand
            if len(newProd) > 1:
                carry = newProd[0]
            else:
                carry = 0
        z.insert(0, carry)
        summands.append(makeInt(z))
    return sum(summands) # finally add them all together
```

3. 正确性验证

经过验证，得到的结果正确无误

```
X = 123456789876
Y = 6543212345
print(gradeSchoolMult(X,Y))
print(X*Y)
```

```
807803991590714219220
807803991590714219220
```

二、 基于分治的整数乘法

1. 伪代码

divideAndConquerMult1(X, Y):

- **If** n=1:

  Return xy

- Write $x = a\,10^{\frac{n}{2}} + b$

- Write $y = c\,10^{\frac{n}{2}} + d$

- Recursively compute $ac,\ ad, bc, bd$:

  ac = **divideAndConquerMult1** (a, c)

  ad = **divideAndConquerMult1** (a, d)

  bc = **divideAndConquerMult1** (b, c)

  bd = **divideAndConquerMult1** (b, d)

- Add them up to get xy:

  xy = ac 10^n + (ad + bc) $10^{n/2}$ + bd

2. 具体的实现代码

```
def divideAndConquerMult1( X, Y ):
    return divideAndConquerMult1_helper( getDigits(X), getDigits(Y) )

def divideAndConquerMult1_helper( x, y ):
    n = max( len(x), len(y) )
    # pad the shorter one with zeros until it's the same length
    while len(x) < n:
        x.insert(0,0)
    while len(y) < n:
        y.insert(0,0)
    if n == 1:
        return x[0]*y[0] # this is the base case, we are allowed to multiply one-digit integers :)
    mid = round(n/2)
    xhigh = x[:mid] # this is [ x[0], x[1], ..., x[mid-1] ]
    xlow = x[mid:] # this is [ x[mid], ..., x[n-1] ]
    yhigh = y[:mid]
    ylow = y[mid:]
    highhigh = divideAndConquerMult1_helper( xhigh , yhigh )
    highlow = divideAndConquerMult1_helper( xhigh , ylow )
    lowhigh = divideAndConquerMult1_helper( xlow , yhigh )
    lowlow = divideAndConquerMult1_helper( xlow , ylow )

    # now shift things appropriately (see slides for explanation) and add them together
    HH = getDigits(highhigh) + [ 0 for i in range(2*(n - mid))]
    MID = getDigits(lowhigh + highlow) + [0 for i in range(n-mid)]
    LL = getDigits(lowlow)
    result = makeInt(HH) + makeInt(MID) + makeInt(LL)
    return result
```

3. 正确性验证

经过验证，得到的结果正确无误

```
X = 123456789876
Y = 65432123456
print(divideAndConquerMult1(X,Y))
print(X*Y)
```

```
8078039916647882931456
8078039916647882931456
```

三、 karatsuba 整数乘法

1. 伪代码

karatsuba(X, Y):

- **If** n=1:
  Return xy
- Write $x = a\,10^{\frac{n}{2}} + b$ and $y = c\,10^{\frac{n}{2}} + d$
- ac = **karatsuba** (a, c)
- bd = **karatsuba** (b, d)
- z = **karatsuba** (a+b, c+d)
- xy = ac 10^n + (z – ac - bd) $10^{n/2}$ + bd
- Return xy

2. 具体的实现代码

```
def karatsuba( X, Y ):
    return karatsuba_helper( getDigits(X), getDigits(Y))

def karatsuba_helper( x, y ):
    n = max( len(x), len(y) )
    # pad the shorter one with zeros until it's the same length
    while len(x) < n:
        x.insert(0,0)
    while len(y) < n:
        y.insert(0,0)
    if n == 1:
        return x[0]*y[0] # this is the base case, we are allowed to multiply one-digit integers :)
    mid = round(n/2)
    xhigh = x[:mid] # this is [ x[0], x[1], ..., x[mid-1] ]
    xlow = x[mid:] # this is [ x[mid], ..., x[n-1] ]
    yhigh = y[:mid]
    ylow = y[mid:]
    highhigh = karatsuba_helper( xhigh , yhigh )
    lowlow = karatsuba_helper( xlow , ylow )
    tmpTerm = karatsuba_helper( getDigits( makeInt(xlow) + makeInt(xhigh) ) , getDigits( makeInt(ylow) + makeInt(yhigh) ) )
    middleTerm = tmpTerm - highhigh - lowlow # this is equal to highlow + lowhigh in divideAndConquerMult1
    HH = getDigits(highhigh) + [ 0 for i in range(2*(n - mid))]
    MID = getDigits(middleTerm) + [0 for i in range(n-mid)]
    LL = getDigits(lowlow)
    result = makeInt(HH) + makeInt(MID) + makeInt(LL)
    return result
```

3. 正确性验证

经过验证，得到的结果正确无误

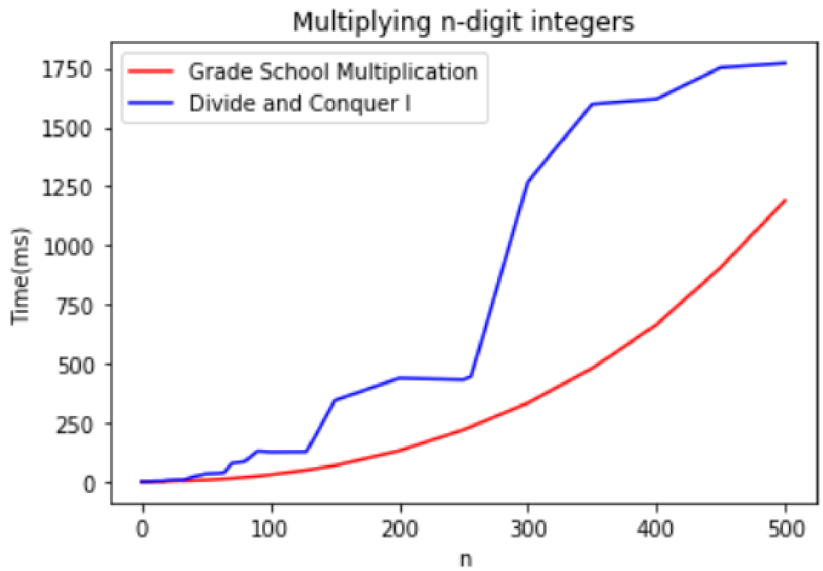
```
X = 123456765432
Y = 6543212345
print(karatsuba(X,Y))
print(X*Y)
```

```
807803831648431658040
807803831648431658040
```

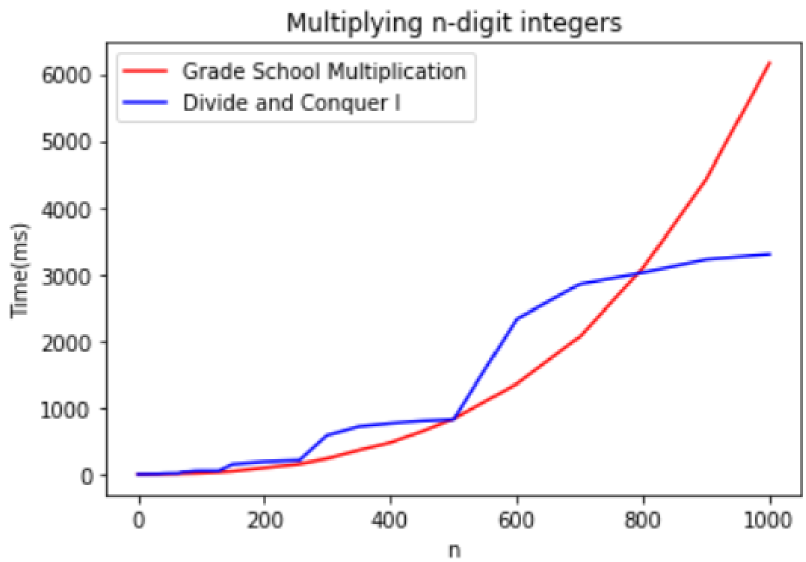
四、 运行时间比较

- 基于分治的整数乘法与小学整数乘法：

率比基于分治的整数乘法更好，效率最高。

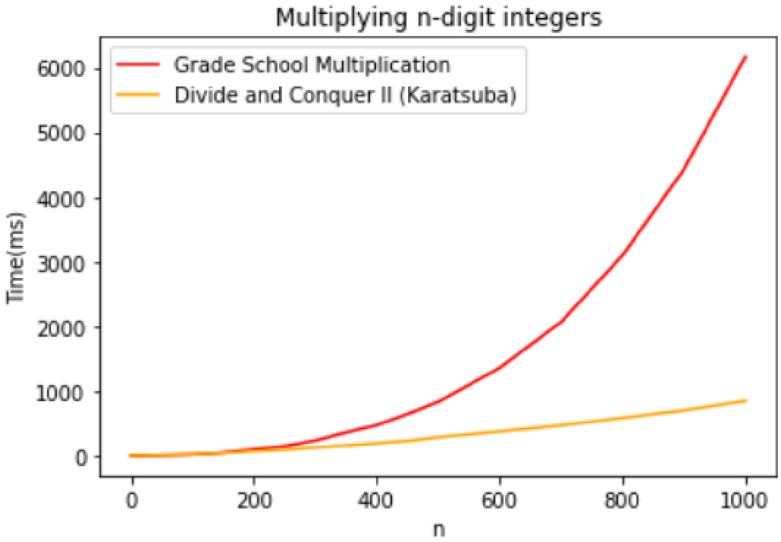


根据上图，当 n 小于 500 时，分治乘法的效果似乎不明显，甚至还略差于小学整数乘法。但是当我们将 n 扩大至 1000 时，如下图，可以看到在 n 较大时，分治乘法用时明显少于小学数学乘法，是有明显效果的。



- karatsuba 整数乘法与小学整数乘法

根据下图，可以看到 karatsuba 整数乘法的用时明显少于小学整数乘法，并且效率比基于分治的整数乘法更好，效率最高。


Multiplying n-digit integers
Grade School Multiplication
Divide and Conquer II (Karatsuba)
Time(ms)
6000
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000
n