Problem1

Problem 1: Simplest Divide and Conquer on Sparse Vector

a, Binary division (m=2):

```
function divideAndConquerAndReturnNonZeroValueRecursive(v)
    n = length(v)
    if n=1:
        return v[1]
    mid = int(round(n/2))
    max_left = divideAndConquerAndReturnNonZeroValueRecursive(v[1:mid])
    max_right = divideAndConquerAndReturnNonZeroValueRecursive(v[mid+1:end])
    return max(max_left, max_right)
end.
```

```
def divide(v,2):
    n=len(v)
    if n=1:
        return v[0]
    mid = round(n/2)
    max_left = divide(v[0:mid],2)
    max_right = divide(v[mid:len(v)],2)
    return max(max_left, max_right)

# test.
v= [0,0,1,0,0,0,0,0]
print(divide_binary(v))  # : output : 1
```

$T(n) = 2T(\frac{n}{2}) + O(1)$

function divide (v, ..., m) (for any 2 << m)

```
def divide_m(v,m):
    n = len(v)
    if n=1:
        return v[0]
    block_size = round(n/m)
    result .= []
    for i in range(m):
        start_idx = i*block_size
        end_idx = min((i+1)*block_size, n)
        results . append(divide(v[start_idx:end_idx], m))
    return max(results)
```

(b). Complexity Analysis.

For $m=2$ (Binary) : $T(n)=2T(\frac{n}{2})+O(1)$

For $m=3$ (Ternary) : $T(n)=3T(\frac{n}{3})+O(1)$

For general m : $T(n)=mT(\frac{n}{m})+O(1)$

Master Theorem : For $T(n)=aT(\frac{n}{b})+O(n^d)$

$a=m$, $b=m$, $d=0$

compare $\log_b a=\log_m m=1$ with $d=0$

science $\log_b a=1>d=0$, $\Rightarrow T(n)=O(n^{\log_b a})=O(n^1)=O(n)$

So. for all m, the complexity is $O(n)$.

(c). Collect the Unique 1 and its position.

cost $f(n)$: $f(n)=O(1)$, $T(n)=T(\frac{n}{2})+O(1)$.

Master Theorem. $a=1$, $b=2$, $d=0$

$\log_b a=\log_2 1=0=d$.

$\Rightarrow T(n)=O(n^d \log n)=O(\log n)$. so the complexity is $O(\log n)$

Problem2

Problem 2 : Multiplication in Basis 10.

$$x = \sum_{i=0}^{n_x} X[i] \cdot 10^i \quad , \quad y = \sum_{j=0}^{n_y} Y[j] \cdot 10^j$$

1. Multiplication.

```
def multiply (X, Y):
    n = len (X)
    m = len (Y)
    result = [0] * (n+m)
    for i in range (n):
        for j in range (m):
            result [i+j] = result [i+j] + X[i] * Y[j]
    carry = 0
    for i in range (len (result)):
        temp = result [i] + carry
        result [i] = temp % 10
        carry = int (temp // 10)
    while len (result) > 1 and result [-1] == 0:
        result . pop ()
    return result.
```

2. def multiplyLarge (X, Y):

```
n=len(X)
m=len(Y)
result =[0]*(n+m)
for i in range(n):
    carry = 0
    for j in range(m):
        temp =result [i+j] + X[i] * Y[j] + carry
        result [i+j] = temp % 10
        carry = int ( temp/10 )
    k =i+m
    while carry >0:
        temp =result [k] + carry
        result [k] = temp %10
        carry = int (temp/10 )
        k = k+1
while len (result) >1 and result[-1] =0:
    result.pop( )
return result
```

4. Computing $\sum_{i=1}^{n} i$ Using One Multiplication

$$\sum_{i=1}^{n} i = n + (n-1) + \cdots + 2 + 1 = \frac{n(n+1)}{2}$$

$$\Rightarrow \sum_{i=1}^{n} i = \frac{1}{2} \cdot \text{mult}(V, W)$$

$\therefore V = n, \quad W = n+1$

Verification: $\frac{1}{2} \cdot \text{mult}(n, n+1) = \frac{n(n+1)}{2} = \sum_{i=1}^{n} i$

3、Divide.

$n \rightarrow$ number of digits.

$x = x_1 \cdot 10^{\frac{n}{2}} + x_0$

$y = y_1 \cdot 10^{\frac{n}{2}} + y_0$

$x_1, y_1 \rightarrow$ high bits.

$x_0, y_0 \rightarrow$ low bits.

Standard Recursive Approach:

$$x \cdot y = (x_1 \cdot 10^{\frac{n}{2}} + x_0)(y_1 \cdot 10^{\frac{n}{2}} + y_0).$$

$$= x_1 y_1 10^n + (x_1 y_0 + x_0 y_1) 10^{\frac{n}{2}} + x_0 y_0.$$

requires 4 multiplications of size $\frac{n}{2}$.

$T(n) = 4T(\frac{n}{2}) + O(n)$

Master Theorem : $a=4$, $b=2$, $d=1$. $\Rightarrow \log_2 4 = 2 > 1$.

$\therefore$ Complexity $= O(n^2)$

Karatsuba Algorithm :

Hint 2. formula.

$x \cdot y = z_2 \cdot 10^n + z_1 \cdot 10^{\frac{n}{2}} + z_0$

the middle term $x_1 y_0 + x_0 y_1$.

$(x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0$.

This requires only 3 multiplications: $z_2 = x_1 y_1$, $z_0 = x_0 y_0$,

and $z_1 = (x_1 + x_0)(y_1 + y_0)$.

$T(n) = 3T(\frac{n}{2}) + O(n)$.

Master Theorem:

$a=3$, $b=2$, $d=1$.

$\log_2 3 \approx 1.585 > 1$.

$\therefore$ Complexity : $O(n^{1.585})$