Q1 $T(n) = \begin{cases} \Theta(1) & n=0 \\ T(n-1)+\Theta(n) & n>0 \end{cases}$

n: big String size

$T(n) = T(n-1)+n$

$T(n-1) = T(n-2)+n-1$

$T(n-2) = T(n-3)+n-2$

$\Rightarrow T(n) = T(n-2)+(n-1)+n$

$T(n) = T(n-3)+(n-2)+(n-1)+n$

$\vdots$

$T(n) = T(\underbrace{n-k}_{0})+(n-(k-1))+(n-(k-2))+\dots+(n-1)+n$

$n-k=0 \Rightarrow k=n$

$T(n) = T(0)+\overbrace{(\cancel{n-}(n-1))+(\cancel{n-}(n-2))+\dots+(n-1)+n}^{1+2+\dots+(n-1)+n}$

$T(n) = 1+\frac{n.(n+1)}{2} = \cancel{\frac{n^2+n+2}{2}} \Rightarrow \Theta(n^2)$

$\left.\begin{array}{l} T_B = \Theta(1) \\ T_w = \Theta(n^2) \end{array}\right\} T_g = \boxed{O(n^2)}$

INDUCTION

$T(n) = T(n-1)+n \Rightarrow \frac{n^2+n+2}{2}$

Base case:

$T(0) = \frac{0^2+0+2}{2} = \frac{2}{2} = 1, n=0$ ✓

Induction Step:

$T(n+1) = \frac{(n+1)^2+(n+1)+2}{2} = \frac{n^2+3n+4}{2}$ Prove

$T(n+1) = \underbrace{T(n)}_{\frac{n^2+n+2}{2}}+n+1 = \frac{n^2+3n+4}{2}$ ✓

Q2 FindItems

$T(n) = \begin{cases} \Theta(1) & n=0 \\ 2T(n-1)+\log(n) & n>0 \end{cases}$

$T(n) = 2T(n-1)+\log(n)$

$T(n-1) = 2T(n-2)+\log(n-1)$

$T(n-2) = 2T(n-3)+\log(n-2)$

$\Rightarrow T(n) = 2[2T(n-2)+\log(n-1)]+\log(n) = 4T(n-2)+2\log(n-1)+\log(n)$

$T(n) = 4[2T(n-3)+\log(n-2)]+2\log(n-1)+\log(n)$

$= 8T(n-3)+4\log(n-2)+2\log(n-1)+\log(n)$

$\vdots$

$T(n) = 2^k T(\underbrace{n-k}_{0})+2^{k-1}\log(\cancel{n-}(k-1))+2^{k-2}\log(\cancel{n-}(k-2))+\dots+\log(n)$

$n-k=0 \Rightarrow k=n$

$T(n) = 2^n+2^{n-1}\log(1)+2^{n-2}\log(2)+\dots+\log(n) \Big\} \sum_{i=2}^{n} 2^i \times \log n$

$T(n) = \Theta(2^n \log(n))$

$\left.\begin{array}{l} T_B = \Theta(1) \\ T_w = \Theta(2^n \log(n)) \end{array}\right\} T_g = \boxed{O(2^n \log(n))}$

wrapperFindItems

$T(n) = O(2^n \log(n))$

contain

$T(n) = \begin{cases} 1 & n=0 \\ T(n/2)+1 & n>0 \end{cases}$

$T(n) = T(n/2)+1$

$T(n/2) = T(n/2^2)+1$

$T(n/2^2) = T(n/2^3)+1$

$\vdots$

$T(n) = T(\underbrace{n/2^k}_{})+k$

$n=2^k \rightarrow k=\log n$

$T(n) = 1+\log n = \Theta(\log(n))$

$T_g = O(\log(n))$

Q3 $T(n) = \begin{cases} \theta(1) & n=0 \\ T(n-1)+\theta(n) & n>0 \end{cases}$

$T(n) = T(n-1) + n$

$T(n-1) = T(n-2) + n - 1$

$T(n-2) = T(n-3) + n - 2$

$\Rightarrow T(n) = T(n-2) + (n-1) + n$

$T(n) = T(n-3) + (n-2) + (n-1) + n$

$\vdots$

$T(n) = T(\underbrace{n-k}_{0}) + \overbrace{(n-(k-1)) + (n-(k-2)) + \dots + (n-1) + n}^{1+2+\dots+(n-1)+n}$

$n = k$

$T(n) = T(0) + \frac{n \cdot (n+1)}{2} = \cancel{\frac{n^2+n+2}{2}} = \theta(n^2)$

$\left.\begin{array}{l} T_B = \theta(1) \\ T_W = \theta(n^2) \end{array}\right\} \boxed{T_g = O(n^2)}$

INDUCTION

$T(n) = T(n-1) + n \Rightarrow \frac{n^2+n+2}{2}$

Base case:

$T(0) = \frac{0^2+0+2}{2} = 1, \ n = 0$ ✓

Induction Step:

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

$T(n+1) = \frac{n^2+3n+4}{2}$ (Prove this)

$T(n+1) = \underbrace{T(n)}_{\frac{n^2+n+2}{2}} + n + 1 = \frac{n^2+3n+4}{2}$ ✓

Q4 Output: Returns the result of multiplying 2 integer value.

How it works: It performs 3 multiplications for sub0, sub1 and sub2.

Base case = There is at least one integer that is less than 10.

Example: integer1 = 14, integer2 = 20

$n=2$

$half=1$

$int1=1, int2=4$

$int3=2, int4=0$

$sub0 = foo(4,0) \Rightarrow 0$

$sub1 = foo(5,2) \Rightarrow 10$

$sub2 = foo(1,2) \Rightarrow 2$

return $(\underbrace{(2\times 10^2)}_{200} + (\underbrace{8\times 10}_{80}) + 0)$; $\Rightarrow 280$

Time Complexity:

$$T(n) = \begin{cases} \Theta(1) & n\le 1 \\ 3T(n/2)+n & n>1 \end{cases}$$

$T(n) = 3T(n/2)+n$

$T(n/2) = 3T(n/2^2)+(n/2)$

$T(n/2^2) = 3T(n/2^3)+(n/4)$

$\Rightarrow T(n) = 3[3T(n/2^2)+(n/2)]+n$

$= 3^2T(n/2^2) + 3\frac{n}{2}+n$

$T(n) = 3^2[3T(n/2^3)+(n/2^2)] + \frac{3}{2}n+n$

$= 3^3T(n/2^3) + 3^2\frac{n}{2^2} + \frac{3}{2}n+n$

$\vdots$

$T(n) = 3^k \underbrace{T(n/2^k)}_{1} + 3^{k-1}\frac{n}{2^{k-1}} + 3^{k-2}\frac{n}{2^{k-2}} + \dots + \frac{3}{2}n+n$

$n=2^k$

$k=\log(n)$

$T(n) = 3^k + \underset{2^k}{n}\left(\underbrace{\left(\frac{3}{2}\right)^{k-1} + \left(\frac{3}{2}\right)^{k-2} + \dots + \frac{3}{2}+1}_{\left(\frac{3}{2}\right)^k-1}\right)$

$T(n) = 3^k + 2^k\left(\frac{3^k}{2^k}-1\right) = 3^k+3^k-2^k = 2\cdot 3^k - 2^k$

$T(n) = 2\cdot 3^{\overset{\log(n)}{k}} - \overset{n}{2^k}$

$T(n) = \cancel{2}\cdot 3^{\log(n)} \cancel{-n}$

$T(n) = 3^{\log n} = n^{\log 3} \Rightarrow \Theta(n^{\log 3})$

$T_g = \boxed{O(n^{\log 3})}$

```
# foo (integer1, integer2)

    if (integer1 < 10) or (integer2 < 10)
      return integer1 * integer2

    //number_of_digit returns the number of digits in an integer
    n = max(number_of_digits(integer1), number_of_digits(integer2))
    half = int(n/2)

    // split_integer splits the integer into returns two integers
    //               from the digit at position half. i.e.,
```

```
    // first integer = integer / 2^half
    // second integer = integer % 2^half
    int1, int2 = split_integer (integer1, half)
    int3, int4 = split_integer (integer2, half)

    sub0 = foo (int2, int4) -T(n/2)
    sub1 = foo ((int2 + int1), (int4 + int3)) -T(n/2)   } 3T(n/2)
    sub2 = foo (int1, int3) -T(n/2)

    return (sub2*10^(2*half))+((sub1-sub2-sub0)*10^(half))+(sub0) -O(n)
```