

def f_1(m: int):

for i in range(10): 0..9

for j in range(m): 0..m-1

print("Hello!") 1 step $\Rightarrow \Theta(1)$ (constant complexity)

$$T(m) = 10 * m * 1 = 10m \in O(m)$$
$$\in \Theta(m)$$

def f_2(m: int):

for i in range(m, m+10): m, m+1, ..., m+9 $\Rightarrow 10$ (regardless of m)

for j in range(m): 0..m-1

print("Hello!") 1

$$T(m) = 10 * m * 1 = 10m \in \Theta(m)$$

def f_3(m: int):

for i in range(1, m):

for j in range(i, m):

print("Hello!")

I $i=1, \dots, m-2, m-1$
 $j=i, i+1, \dots, m-1$

i=1

i=2

i=3

i=m-2

i=m-1

1 2 ... m-1
2 ... m-2
3 ... m-1

STEPS

m-2 m-1 2

m-1 1

+

$= T(m)$

$$T(m) = 1 + 2 + \dots + m-1 = \frac{(m-1)m}{2}$$

$$= \frac{m^2 - m}{2} = \underline{\frac{1}{2}m^2} - \underline{\frac{1}{2}m}$$

dominant term,
discard constant

$$T(m) \in O(m^2), \Theta(m^2)$$

$$\text{II } T(n) = \sum_{i=1}^{n-1} \sum_{j=i}^{n-1} 1$$

$$\sum_{j=i}^{n-1} 1 = \underbrace{1+1+1+1+\dots+1}_{m-1-i+1 \text{ times}} \Rightarrow i \ i+1 \dots n-1$$

$$\Rightarrow T(n) = \sum_{i=1}^{n-1} m-i = \sum_{i=1}^{n-1} m - \sum_{i=1}^{n-1} i \\ = m \cdot \sum_{i=1}^{n-1} 1 - \sum_{i=1}^{n-1} i$$

$$= m \cdot (n-1) - (1+2+\dots+m-1) \\ = m(m-1) - \frac{(m-1)m}{2} \\ = \frac{m(m-1)}{2} \in \Theta(m^2)$$

new unary terms?

e.g. $i=5 \quad \underline{\quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad}^{m-1}$
 $m=10 \quad 5 = 9-5+1$

def f_4(n: int):

for i in range(n):
 for j in range(2*i+1):
 print("Hello!")

	j	STEPS
I	0	1
i=0	0	1
i=1	0, 1, 2	3
i=2	0, 1, 2, 3, 4	5
i=3	0, 1, 2, 3, 4, 5, 6	7

$$i=m-1 \quad 0, 1, \dots, 2m-2, 2m-1$$

+

T(n)

$$T(n) = 1+3+5+\dots+2m-1 = 1+2+3+\dots+2m-2+2m-1-2(1+2+\dots+m-1) \\ = \frac{(2m-1) \cdot 2m}{2} - \cancel{2} \cdot \frac{(m-1)m}{2} = m(2m-1)-m(m-1) \\ = m(2m-1-m+1)$$

$$\text{II } T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{2i} 1$$

$$\sum_{j=0}^{2i} 1 = 1 = \underbrace{1+1+1+\dots+1}_{2i+1} = 2i+1$$

$$T(n) = \sum_{i=0}^{n-1} 2i+1 = 2 \sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1$$

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

$$= \cancel{2} \cdot \frac{(n-1) \cdot n}{\cancel{2}} + n$$

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

$$= n^2 \in \Theta(n^2)$$

def f_5(n: int):

for i in range(n^{**2}): 0.. n^2-1

 for j in range(i): 0.. $i-1$

 print("Hello!") 1

$$\text{I } \sum_{i=0}^{n^2-1} \sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n^2-1} i = 0+1+\dots+n^2-1$$

$$= \frac{(n^2-1) \cdot n^2}{2}$$

$$= \frac{(m^4-m^2)}{4} \in O(m^4)$$

$$\Theta(m^4)$$

$$1^2 + 2^2 + \dots + m^2 = \frac{n(n+1)(2n+1)}{6}$$



	<u>j</u>	STEPS
i=0	-	0
i=1	0	1
i=2	0,1	2
i=3	0,1,2	3

$$T(n) = 0+1+\dots+n^2-1$$

$$= \frac{(n^2-1) \cdot n^2}{2}$$

= ...

$$i=n^2-1 \quad | \quad 0, \dots, n^2-2 \quad | \quad n^2-1 \quad +$$

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

```

def f_9(m: int):
    for i in range(m):
        for j in range(m):
            print("Hello World!")
    for k in range(m):
        print("Hello world!")

```

$$T(m) = \sum_{i=0}^{m-1} \left(\sum_{j=0}^{m-1} 1 + \sum_{k=0}^{m-1} 1 \right) = \sum_{i=0}^{m-1} 2m = 2m \sum_{i=0}^{m-1} 1 = 2m \cdot m = 2m^2 \in \Theta(m^2)$$

|| ||
 m m

does not
 depend on i

```

def f_13(data: list):
    if len(data) == 0:
        return 0
    if len(data) == 1:
        return data[0]
    m = len(data)//2
    return f_13(data[:m]) + f_13(data[m:])

```

$$T(m) = \begin{cases} 1 & \text{if } m = \text{len}(data) \leq 1 \\ 2 \cdot T(m/2) + 1 & \end{cases}$$

TIME COMPLEXITY

$$T(m) = 2 \cdot T(m/2) + 1$$

$$T(m/2) = 2 \cdot T(m/4) + 1$$

$$T(m/4) = 2 \cdot T(m/8) + 1$$

:

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

$$T(1) = 1$$

$$\begin{aligned}
 T(m) &= 2 \cdot [2 \cdot T(m/4) + 1] + 1 \\
 &= 2^2 T(m/4) + 2 + 1 \\
 &= 2^2 [2T(m/8) + 1] + 2 + 1 \\
 &= 2^3 T(m/8) + 4 + 2 + 1 \\
 &= \dots
 \end{aligned}$$

Let $k = \text{number of times list is split in halves} \Rightarrow m = 2^k$

$$k = \log_2 m$$

$$\begin{aligned}
 \Rightarrow T(n) &= 2^k \cdot T(1) + 2^{k-1} + 2^{k-2} + \dots + 2^2 + 2 + 1 \\
 &= 2^k + 2^{k-1} + \dots + 2^2 + 2 + 1 \\
 &= 2^{k+1} - 1 \\
 &= 2^{\log_2 n} \cdot 2 - 1 \\
 &= 2n - 1 \in \Theta(n)
 \end{aligned}$$

EXTRA SPACE COMPLEXITY

$$T(n) = \begin{cases} 1, & n \leq 1 \\ 2T(n/2) + m, & \text{otherwise} \end{cases}$$

! SLICING

→ repeat the same process as for time complexity

$$T(n) = 2 \cdot T(n/2) + m$$

$$T(n/2) = 2T(n/4) + m/2$$

$$T(n/4) = 2T(n/8) + m/4$$

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

$$T(1) = 1$$

$$\begin{aligned}
 T(n) &= 2 \cdot T(n/2) + m \\
 &= 2 [2T(n/4) + m/2] + m \\
 &= 2^2 T(n/4) + m + m \\
 &= 2^2 [2T(n/8) + m/4] + m + m \\
 &= 2^3 T(n/8) + m + m + m \\
 &= \dots \\
 &= 2^k T(1) + k \cdot m \\
 &= m + m \cdot \log n \\
 &= (m+1) \log n \in \Theta(m \log n)
 \end{aligned}$$

```
def recursive_f_1(n: int):
```

```
    if n <= 0:  
        return 1
```

```
    else:
```

```
        return 1 + recursive_f_1(n-1)
```

$$T(n) = \begin{cases} 1 & \text{if } n \leq 0 \\ T(n-1) + 1 & \text{otherwise} \end{cases}$$

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

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

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

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

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

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

⋮

= ...

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

$$= \underbrace{1 + 1 + 1 + \dots + 1}_{n+1}$$

$$T(1) = T(0) + 1$$

$$= n+1 \in \Theta(n)$$

$$T(0) = 1$$

```
def recursive_f_2(n: int)
```

```
    if n <= 1:  
        return 1
```

```
    else:
```

```
        return 1 + recursive_f_2(n-5)
```

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n-5) + 1 & \text{otherwise} \end{cases}$$

$$T(n) = T(n-5) + 1$$

$$T(n-5) = T(n-10) + 1$$

⋮

$$T(k) = 1, k \leq 1$$

} $n/5$ (how many
times can I
subtract 5 from
my number?)

$$T(n) = 1 + T(n-5)$$

$$= 1 + 1 + T(n-10)$$

$$= n/5 \in \Theta(n)$$

def recursive_f3(n: int)

if $n \leq 1$:

return 0

else

return 1 + recursive_f3(n/2)

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ T(n/2) + 1 & \text{otherwise} \end{cases}$$

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

$$T(n/2) = T(n/4) + 1$$

$$\begin{aligned} T(n) &= T(n/2) + 1 \\ &= T(n/4) + 1 + 1 \\ &= T(n/8) + 1 + 1 + 1 \\ &= \dots \end{aligned}$$

$$T(k) = 1, k \leq 1$$

RESULT: $T(n) \in \Theta(\log n)$

def recursive_f4(n: int, m: int, o: int):

if $n \leq 0$:

print(n, m, o)

else:

recursive_f4(n-1, m+1, o)

recursive_f4(n-1, m, o+1)

$$T(n) = \begin{cases} 1 & \text{if } n \leq 0 \\ 2T(n-1) + 1 & \text{otherwise} \end{cases}$$

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

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

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

:

$$T(0) = 2T(-1) + 1$$

$$T(1) = 2 \cdot T(0) + 1$$

$$T(0) = 1$$

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

$$\begin{aligned} &= 2 \cdot [2T(n-2) + 1] + 1 \\ &= 2^2 \cdot T(n-2) + 2 + 1 \\ &= 2^2 [2T(n-3) + 1] + 2 + 1 \\ &= 2^3 T(n-3) + 2^2 + 2 + 1 \\ &= \dots \\ &= 2^{n-1} T(1) + 2^{n-2} + 2^{n-3} + \dots + 1 \\ &= 2^n T(0) + 2^{n-1} + \dots + 2 + 1 \\ &= 1 + 2 + \dots + 2^n \\ &= 2^{n+1} - 1 \in \Theta(2^n) \end{aligned}$$

```
def recursive_f5(n):
    print(n)
    for i in range(n):
        print("Hello!")
    if n < 1:
        return 1
    else:
        return 1 + recursive_f5(n-1)
```

$$T(n) = \begin{cases} 1 & \text{if } n < 1 \\ T(n-1) + n & \text{otherwise} \end{cases}$$

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

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

:

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

$$T(1) = T(0) + 1$$

$$T(0) = 1$$

$$\text{then } T(n) = T(n-1) + n$$

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

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

= ...

$$= T(2) + 3 + 4 + \dots + n$$

$$= T(1) + 2 + 3 + 4 + \dots + n$$

$$= T(0) + 1 + 2 + 3 + \dots + n$$

$$= 1 + 1 + 2 + 3 + \dots + n$$

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

$$= \frac{2 + n^2 + n}{2} \in \Theta(n^2)$$