

Introduction à la science informatique

Semaine 6 (1 séance de 2h)

Récursivité

La factorielle

$$n! = \prod_{i=1}^n i = 1 \times 2 \times \cdots \times (n - 1) \times n$$

$n!$ = nb. permutations de n objets

$n!$ = nb. permutations de n objets



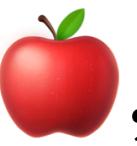
$n!$ = nb. permutations de n objets

{ ,  }

[, ]

[, ]

$n!$ = nb. permutations de n objets

{ ,  }

[, ]

[, ]

$2! = 2$

$n!$ = nb. permutations de n objets

{ ,  }

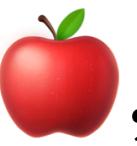
{ , ,  }

[, ]

[, ]

$$2! = 2$$

$n!$ = nb. permutations de n objets

{ ,  }

{ , ,  }

[, ]

[, , ]

[, , ]

[, ]

[, , ]

[, ]

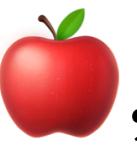
[, , ]

[, , ]

[, , ]

$2! = 2$

$n!$ = nb. permutations de n objets

{  ,  }

{  ,  ,  }

[ , ]

[ ,  , ]

[ ,  , ]

[ , ]

[ ,  , ]

[ , ]

[ ,  , ]

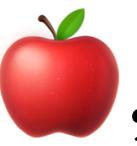
[ ,  , ]

[ ,  , ]

$$2! = 2$$

$$3! = 6$$

$n!$ = nb. permutations de n objets

{ ,  }

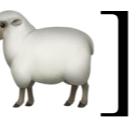
{ , ,  }

{  }

[, ]

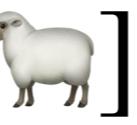
[, ]

[, , ]

[, , ]

[, , ]

[, , ]

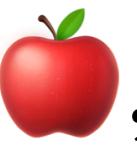
[, , ]

[, , ]

$$2! = 2$$

$$3! = 6$$

$n!$ = nb. permutations de n objets

{  ,  }

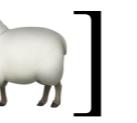
{  ,  ,  }

{  }

[ , ]

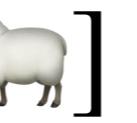
[ , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

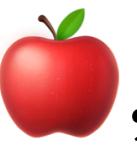
[ ,  , ]

[]

$$2! = 2$$

$$3! = 6$$

$n!$ = nb. permutations de n objets

{  ,  }

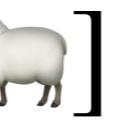
{  ,  ,  }

{  }

[ , ]

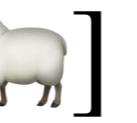
[ , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

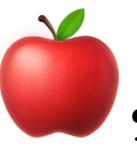
[]

$$2! = 2$$

$$3! = 6$$

$$1! = 1$$

$n!$ = nb. permutations de n objets

{  ,  }

{  ,  ,  }

{  }

\emptyset

[ , ]

[ ,  , ]

[]

[ , ]

[ ,  , ]

[ ,  , ]

[ ,  , ]

$$2! = 2$$

$$3! = 6$$

$$1! = 1$$

$n!$ = nb. permutations de n objets

{ , }

{ , , }

{ }

\emptyset

[,]

[,]

[, ,]

[, ,]

[, ,]

[, ,]

[, ,]

[, ,]

[]

[]

$$2! = 2$$

$$3! = 6$$

$$1! = 1$$

$n!$ = nb. permutations de n objets

{ , }

{ , , }

{ }

\emptyset

[,]

[, ,]

[]

[]

[,]

[, ,]

[, ,]

[, ,]

$2! = 2$

$3! = 6$

$1! = 1$

$0! = 1$

Algorithme itératif pour $n!$

$$n! = 1 \times 2 \times \cdots \times (n - 1) \times n$$

Algorithme itératif pour $n!$

$$n! = 1 \times 2 \times \cdots \times (n - 1) \times n$$

```
def fact_iter(n):
    r = 1
    for i in range(1, n + 1):
        r = r * i
    return r
```

La factorielle

$$n! = \prod_{i=1}^n i = 1 \times 2 \times \cdots \times (n - 1) \times n$$

La factorielle

$$n! = \prod_{i=1}^n i = \underbrace{1 \times 2 \times \cdots \times (n-1) \times n}_{\text{un produit de } n \text{ termes}}$$

La factorielle

$$n! = \prod_{i=1}^n i = \underbrace{1 \times 2 \times \cdots \times (n-1) \times n}_{(n-1)!}$$

La factorielle

$$n! = \prod_{i=1}^n i = \underbrace{1 \times 2 \times \cdots \times (n-1) \times n}_{(n-1)!}$$

La factorielle

$$n! = \prod_{i=1}^n i = \underbrace{1 \times 2 \times \cdots \times (n-1) \times n}_{(n-1)!} + \underbrace{n \times (n-1)!}$$

La factorielle

$$n! = \prod_{i=1}^n i = \underbrace{1 \times 2 \times \cdots \times (n-1) \times n}_{(n-1)!} + \underbrace{n \times (n-1)!}$$

remarque : $0! = \prod_{i=1}^0 i = 1$

Algorithme récursif pour $n!$

$n! = n \times (n - 1)!$ avec $0! = 1$

Algorithme récursif pour $n!$

$$n! = n \times (n - 1)! \text{ avec } 0! = 1$$

```
def fact(n):
```

Algorithme récursif pour $n!$

$$n! = n \times (n - 1)! \text{ avec } 0! = 1$$

```
def fact(n):  
    if n == 0:
```

Algorithme récursif pour $n!$

$$n! = n \times (n - 1)! \text{ avec } 0! = 1$$

```
def fact(n):  
    if n == 0:  
        return 1
```

Algorithme récursif pour $n!$

$$n! = n \times (n - 1)! \text{ avec } 0! = 1$$

```
def fact(n):
    if n == 0:
        return 1
    else:
```

Algorithme récursif pour $n!$

$$n! = n \times (n - 1)! \text{ avec } 0! = 1$$

```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)
```

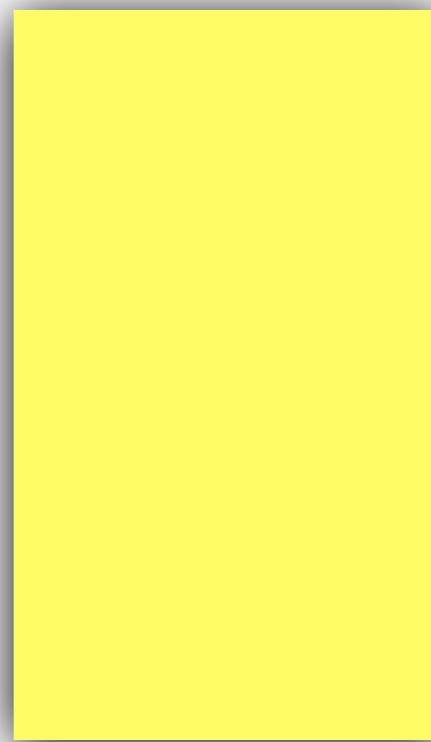
Exécution d'un algorithme récursif

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

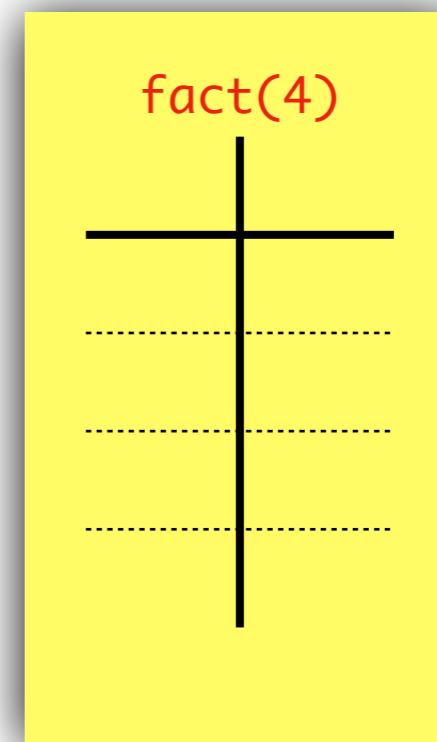
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



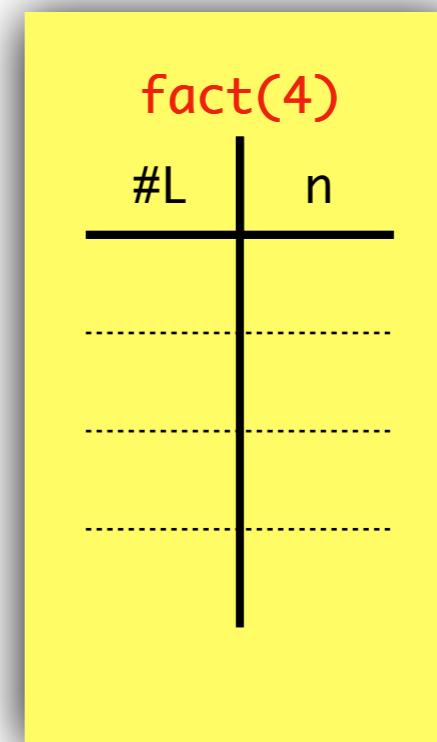
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$

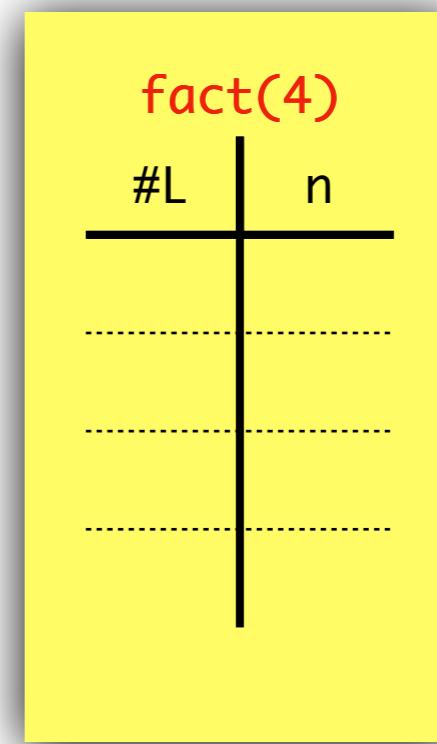
```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$



```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$



```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

fact(4)	
#L	n
1	4

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



fact(4)	
#L	n
1	4

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



fact(4)	
#L	n
1	4
2	

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

fact(4)

#L	n
1	4
2	

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



fact(4)	
#L	n
1	4
2	
4	

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



fact(4)	
#L	n
1	4
2	
4	
5	

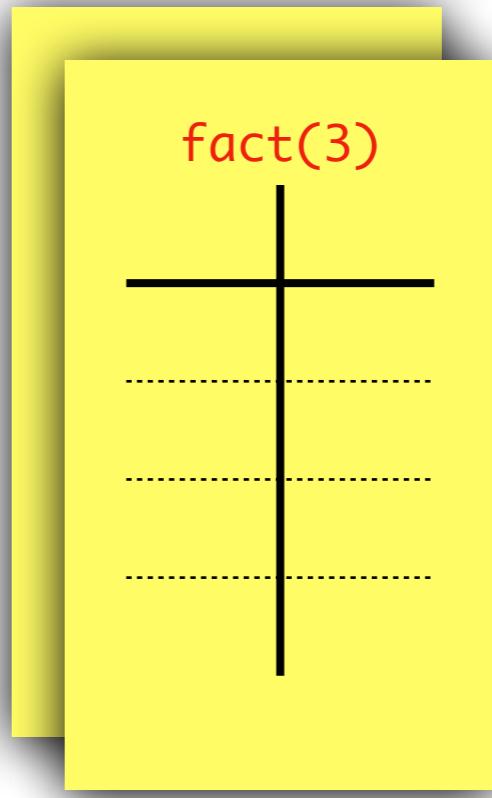
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

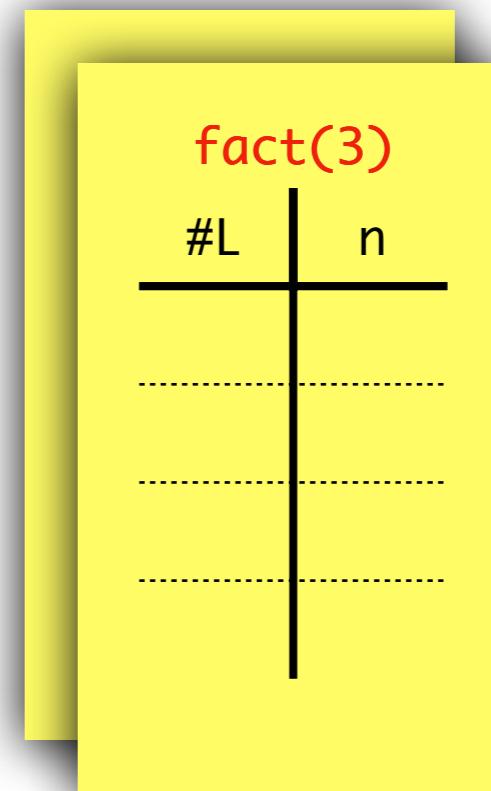


A yellow sticky note with a black border contains a calligraphic representation of the factorial function `fact(3)`. The note has a small shadow and is positioned to the right of the code. The calligraphic representation consists of a vertical line with horizontal dashed lines extending from it to the left and right.



Calcul de $4! = 24$

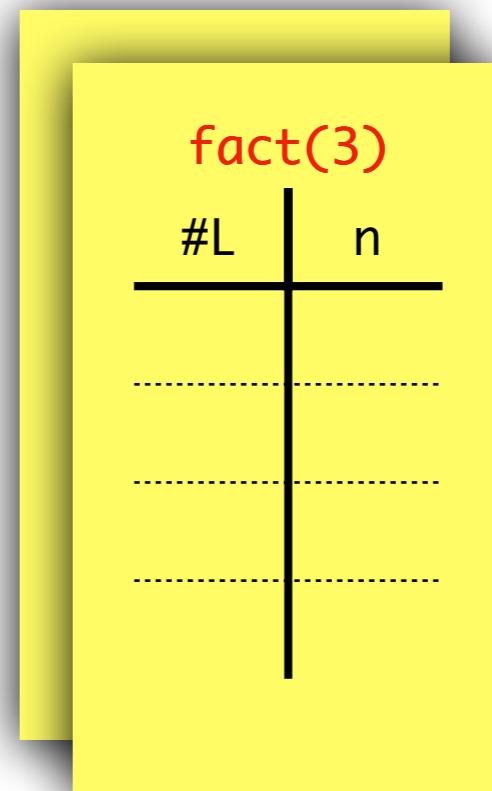
```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$



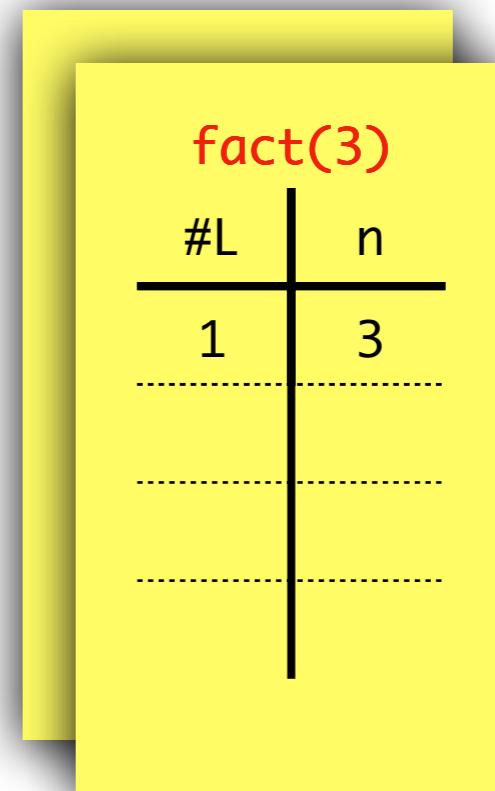
```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$

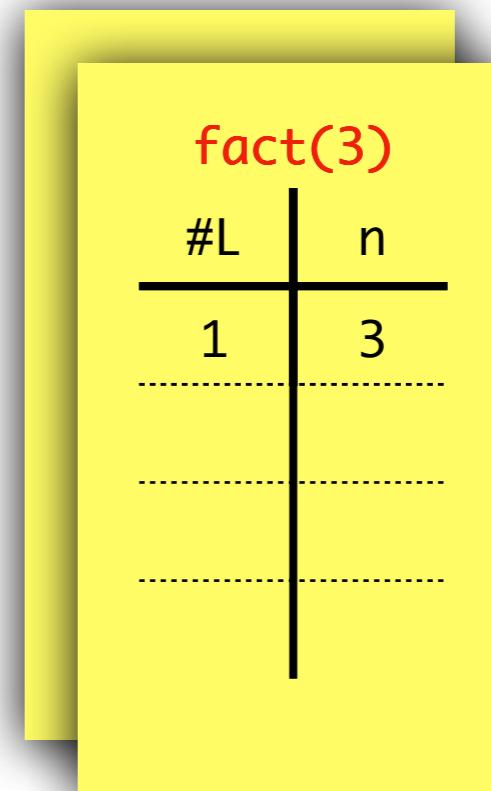
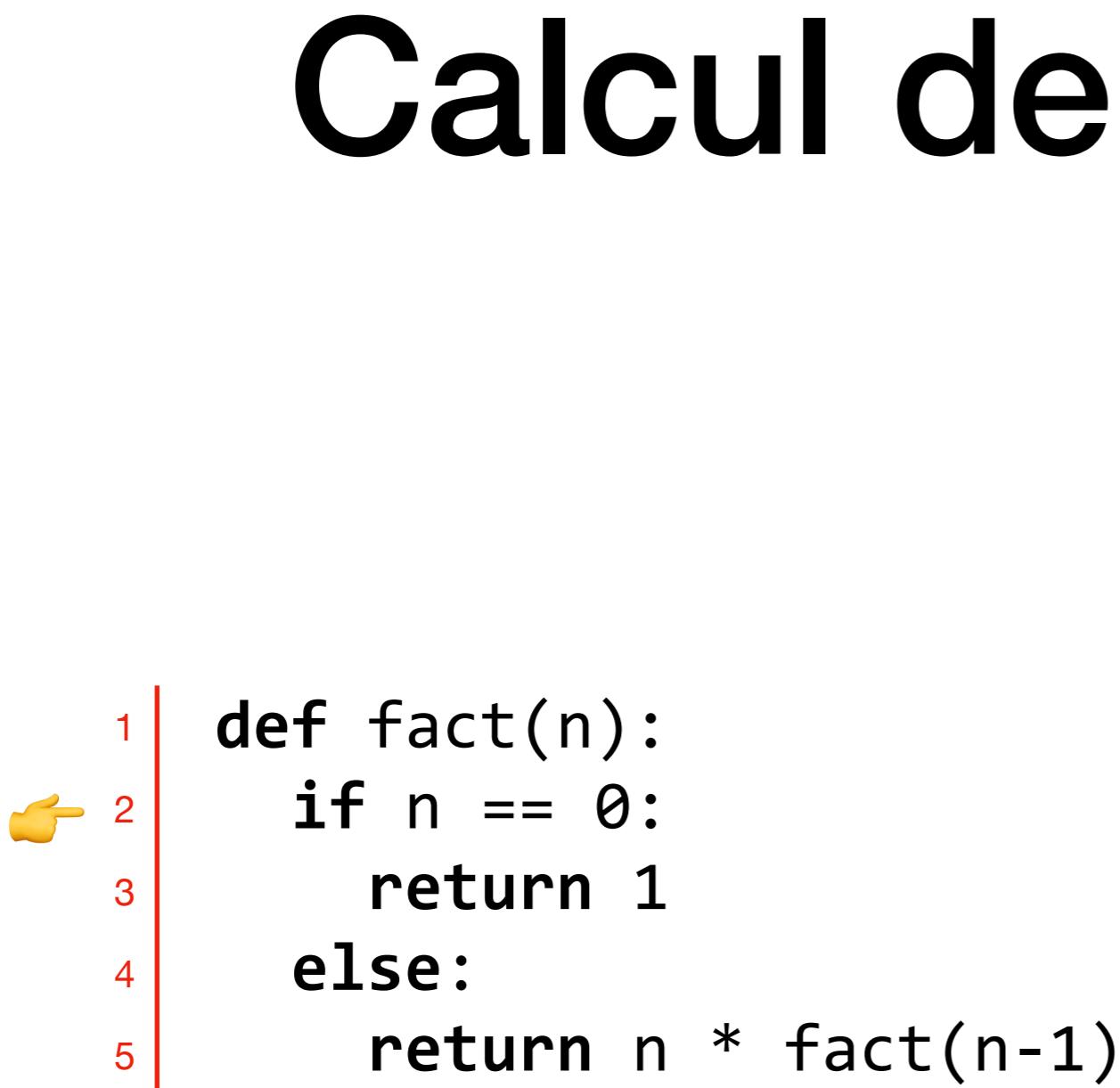


```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



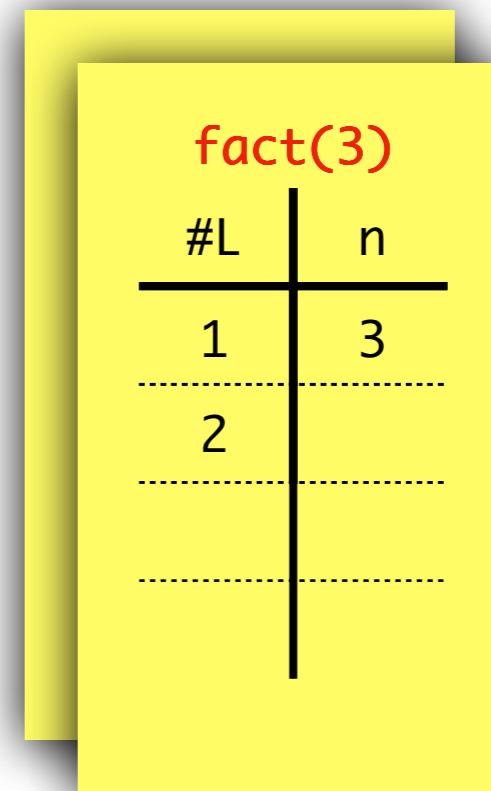
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



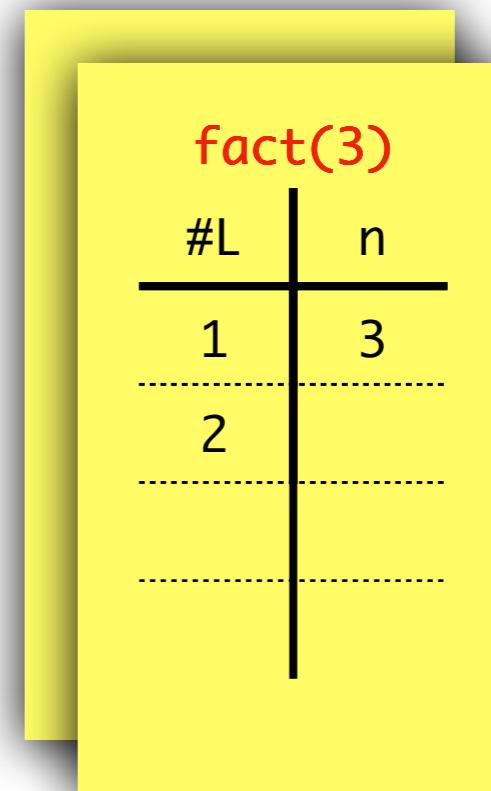
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



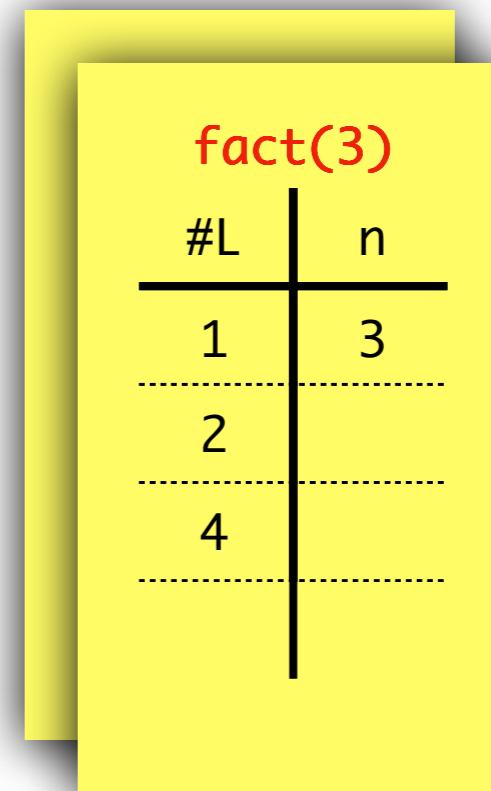
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



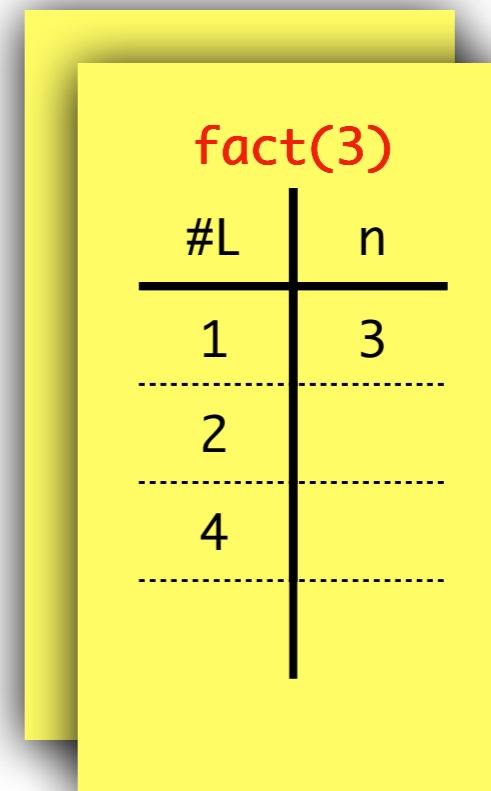
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



fact(3)	
#L	n
1	3
2	
4	
5	

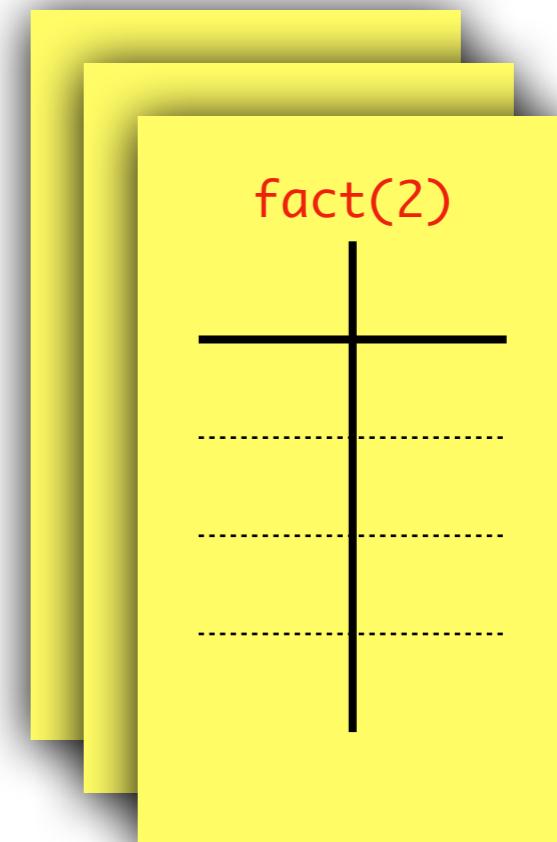
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



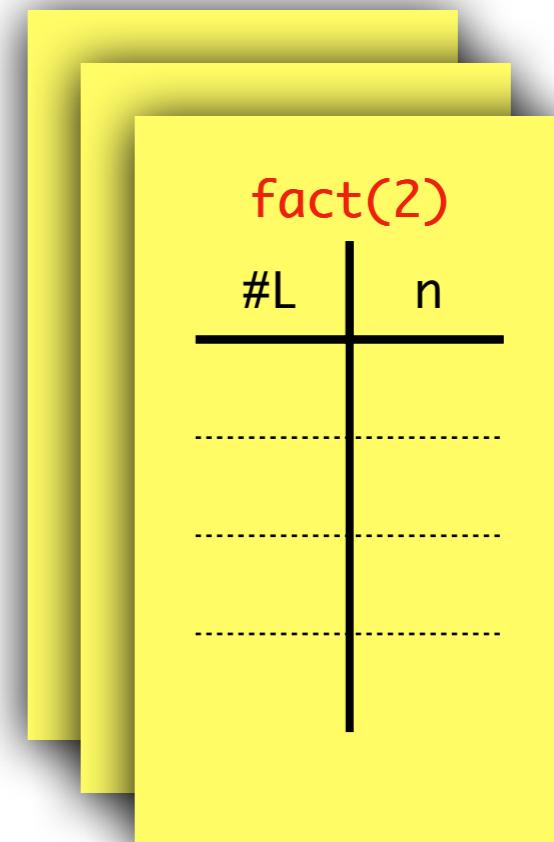
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$

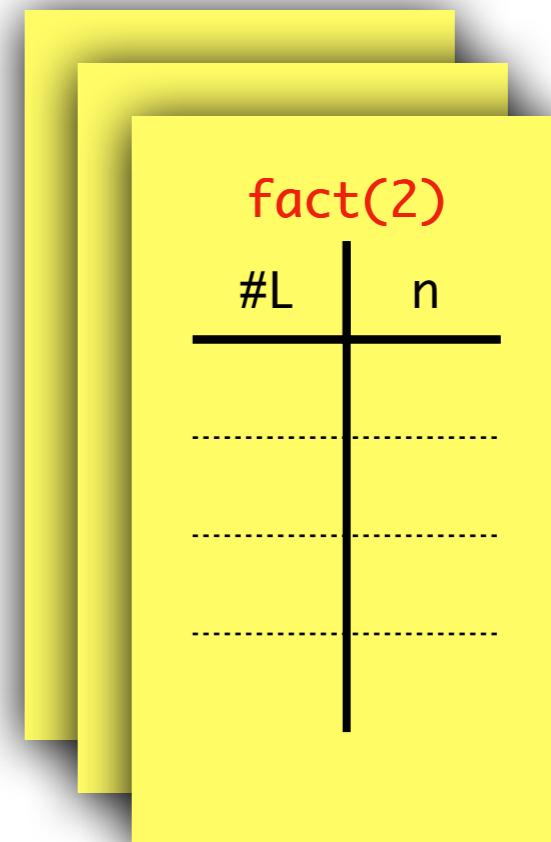
```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$



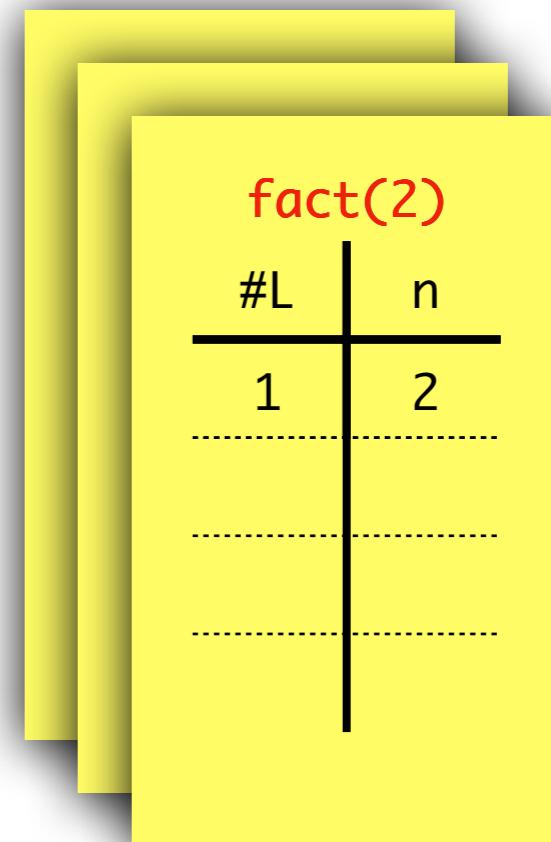
```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



Calcul de $4! = 24$

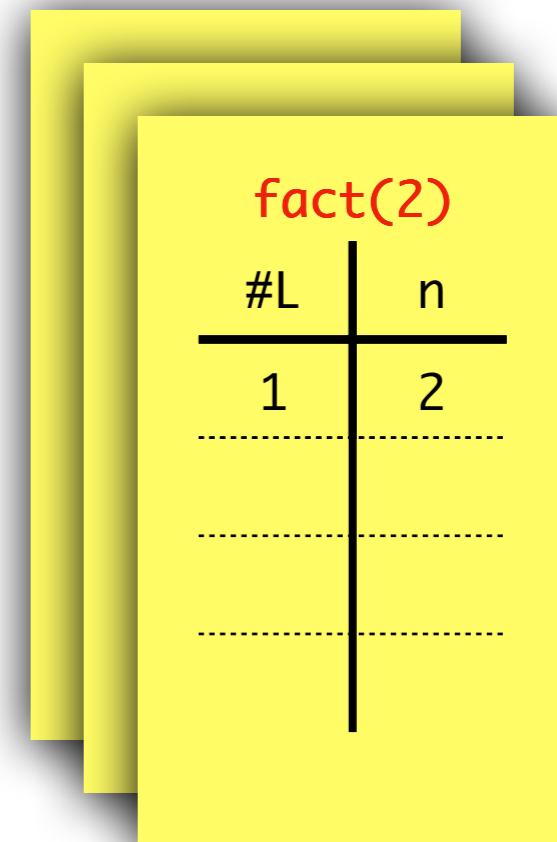


```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



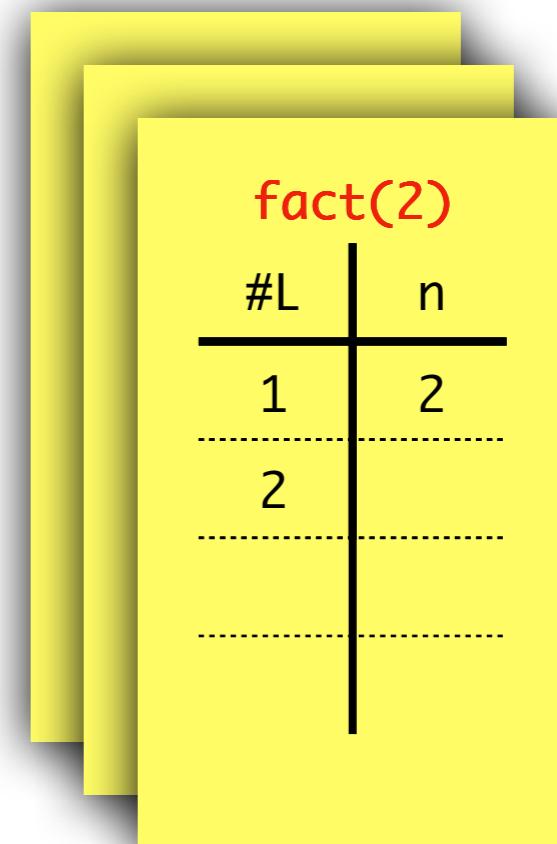
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



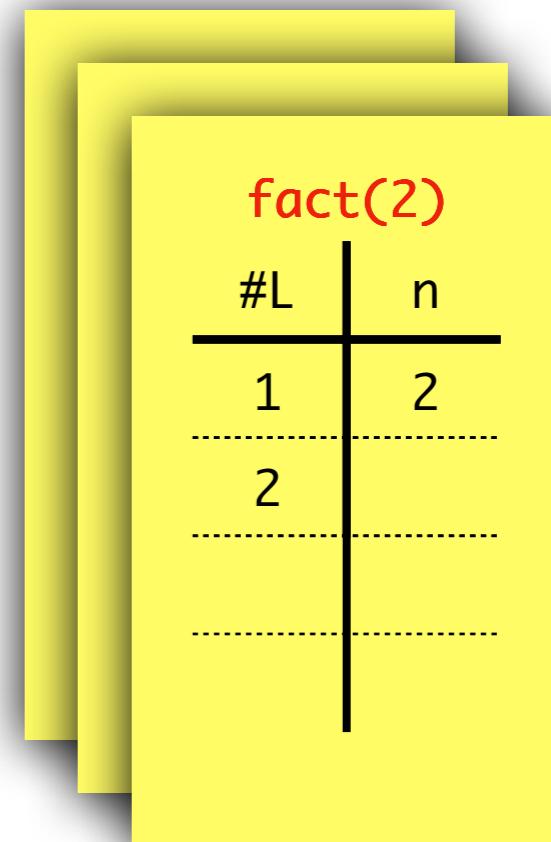
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



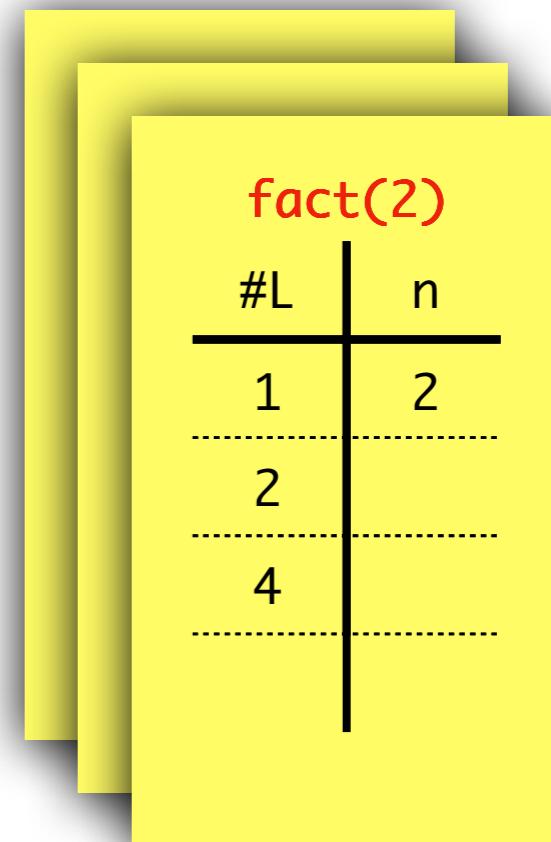
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



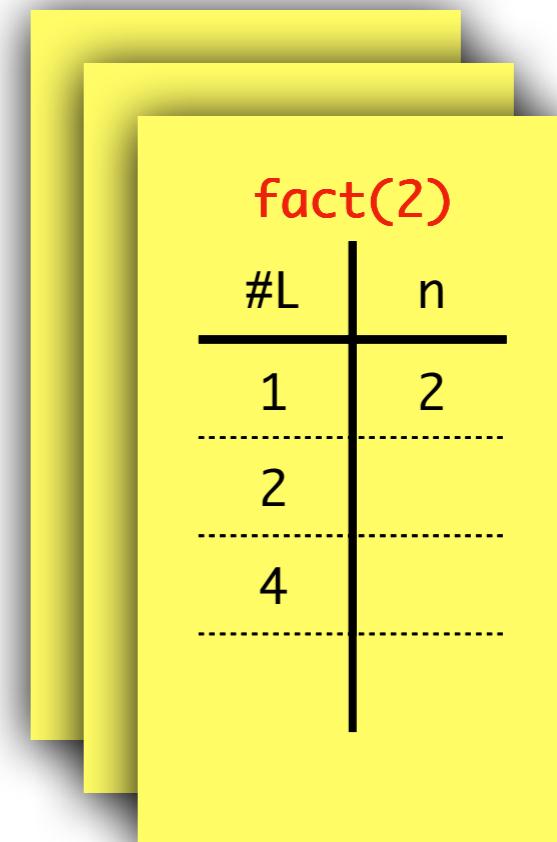
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



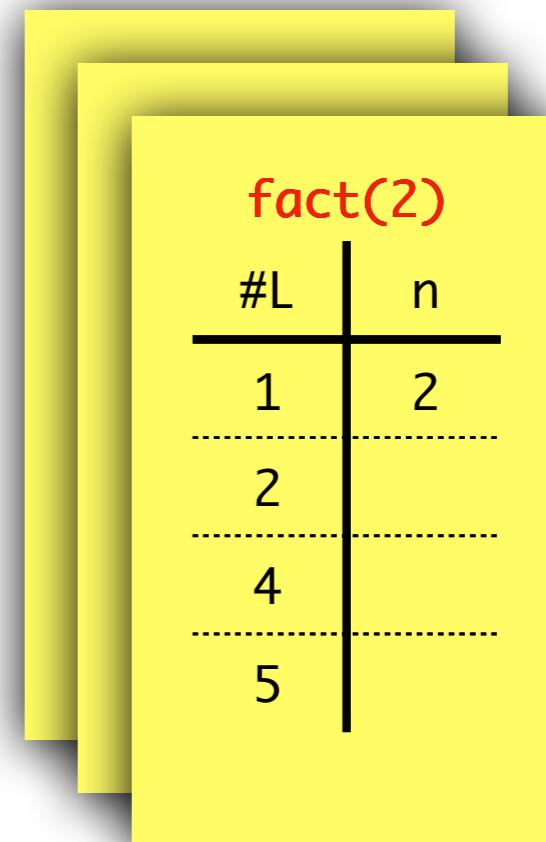
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



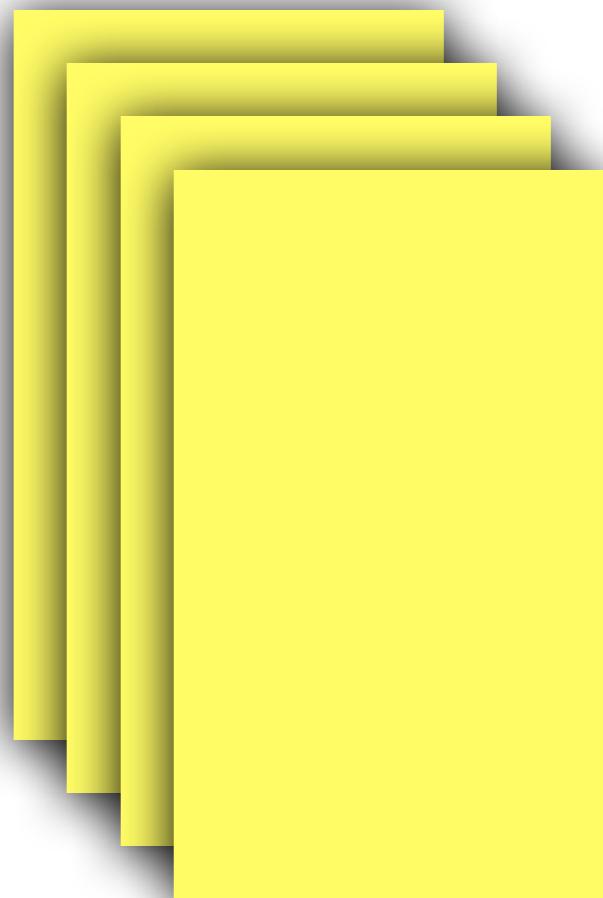
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



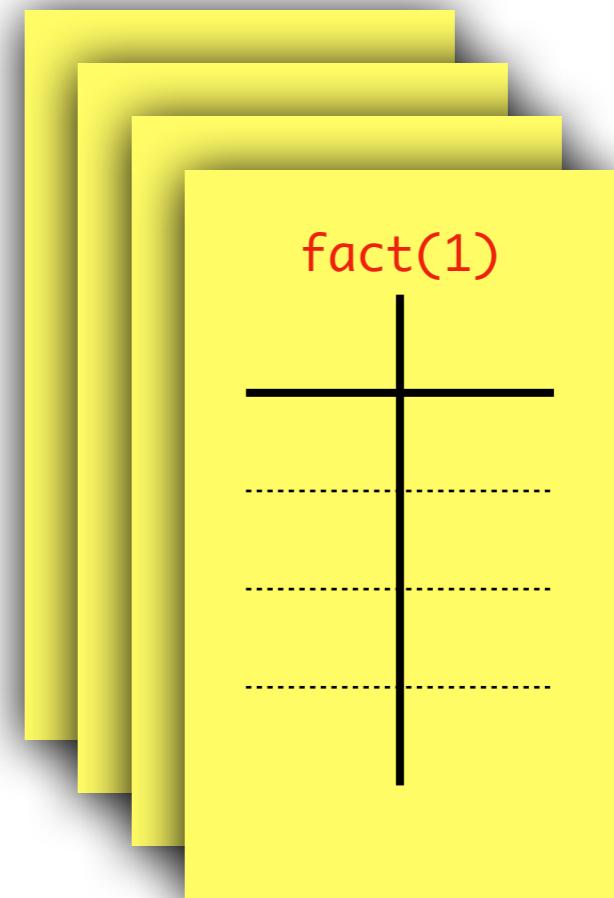
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



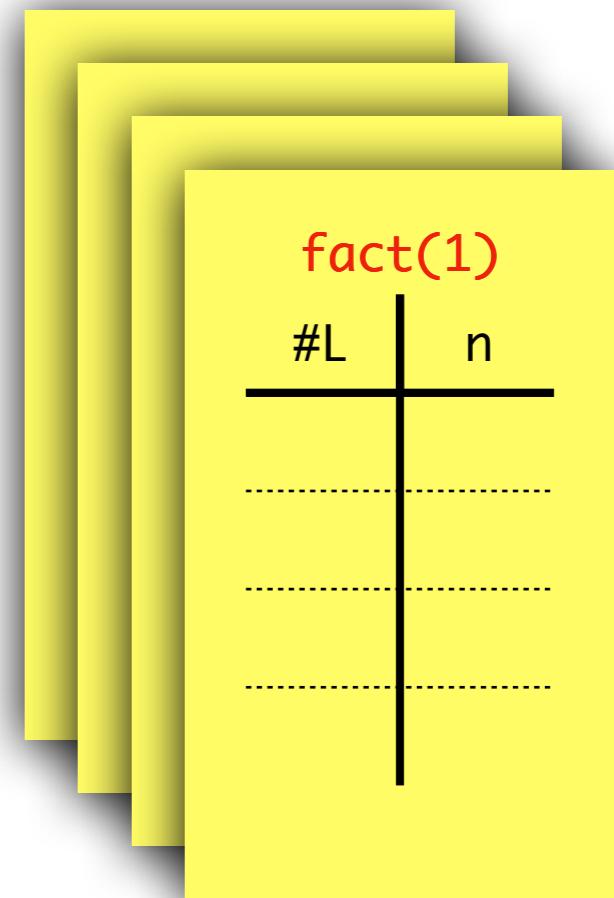
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



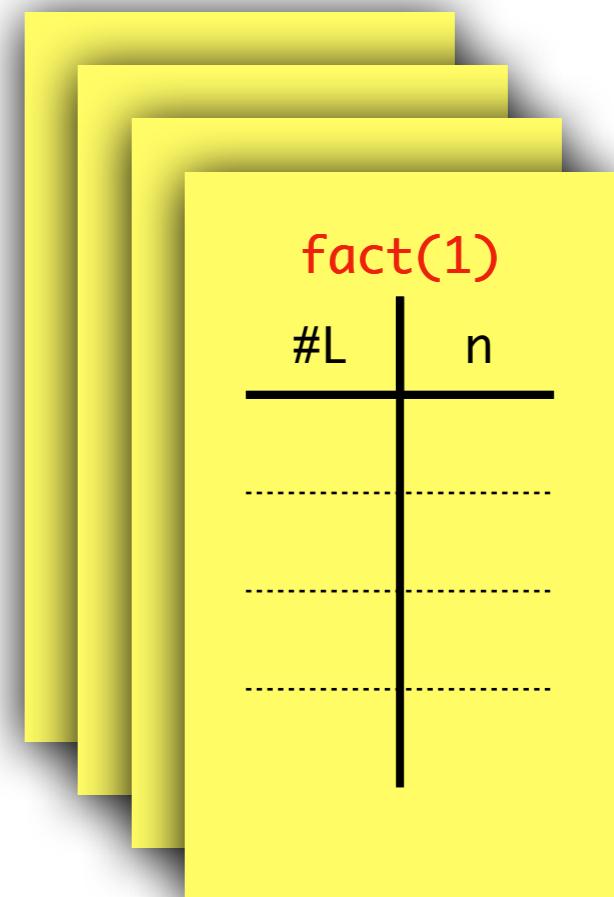
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



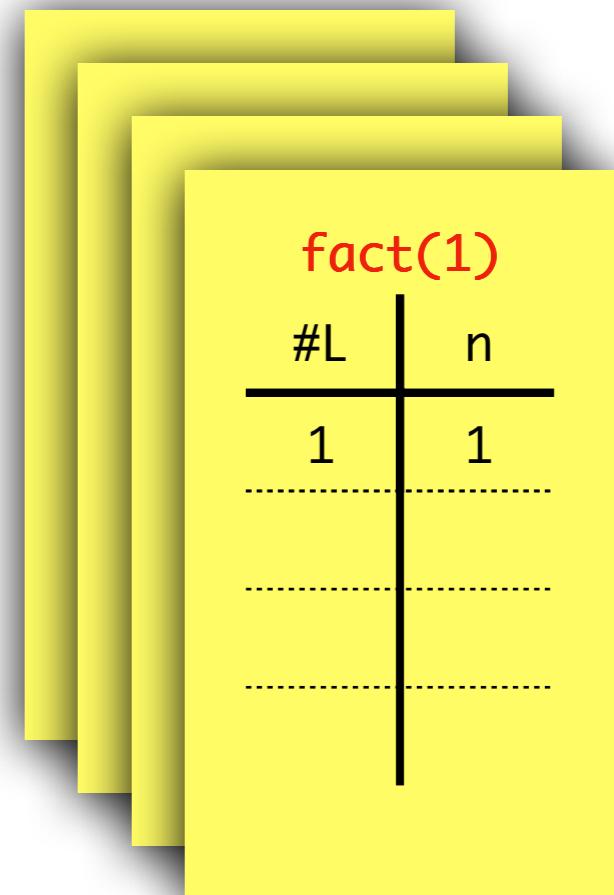
Calcul de $4! = 24$

👉 1 | **def** fact(n):
2 | **if** n == 0:
3 | **return** 1
4 | **else**:
5 | **return** n * fact(n-1)



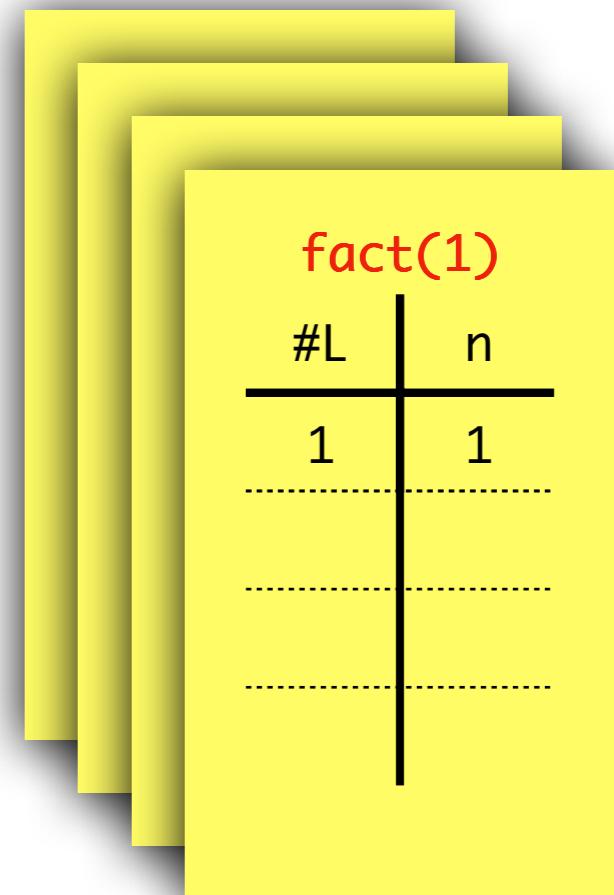
Calcul de $4! = 24$

👉 1 | **def** fact(n):
2 | **if** n == 0:
3 | **return** 1
4 | **else**:
5 | **return** n * fact(n-1)



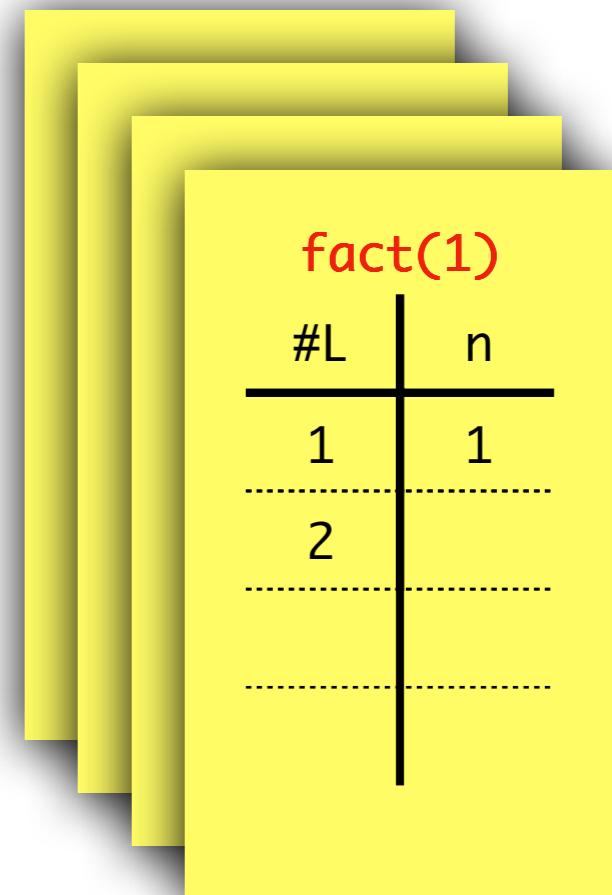
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



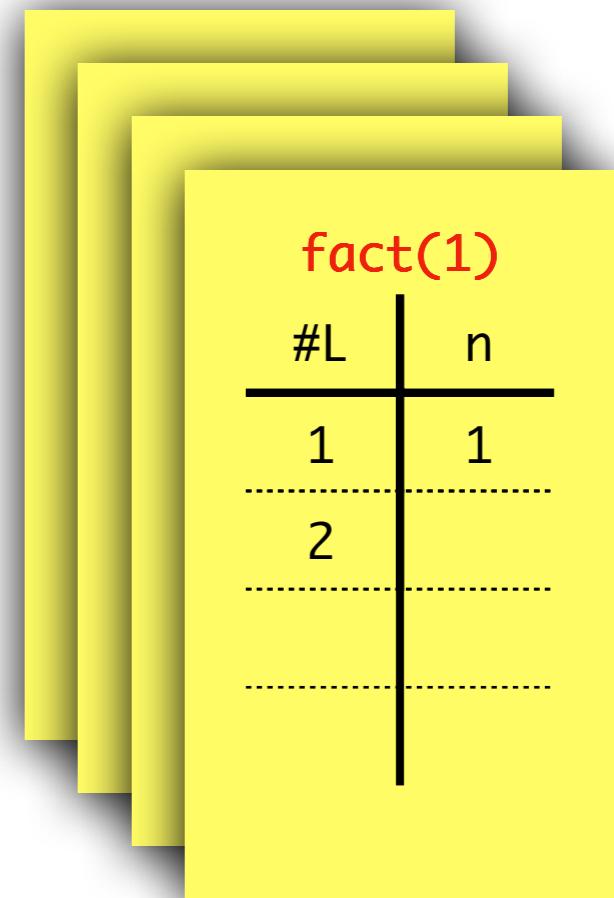
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



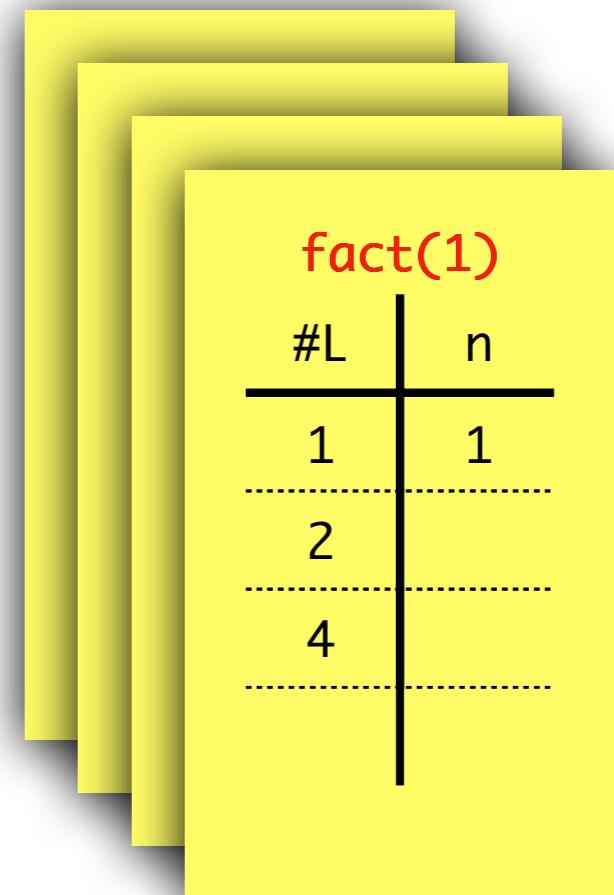
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



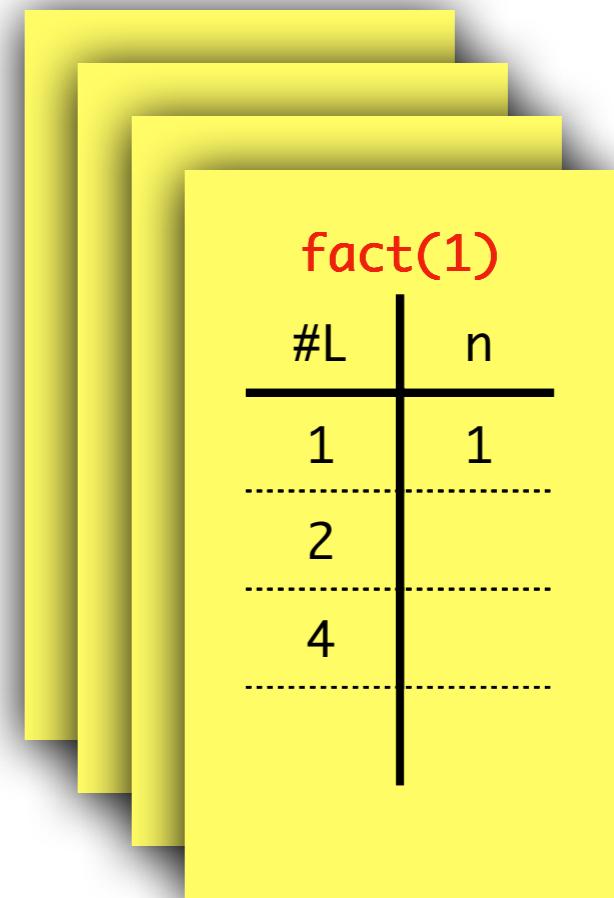
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



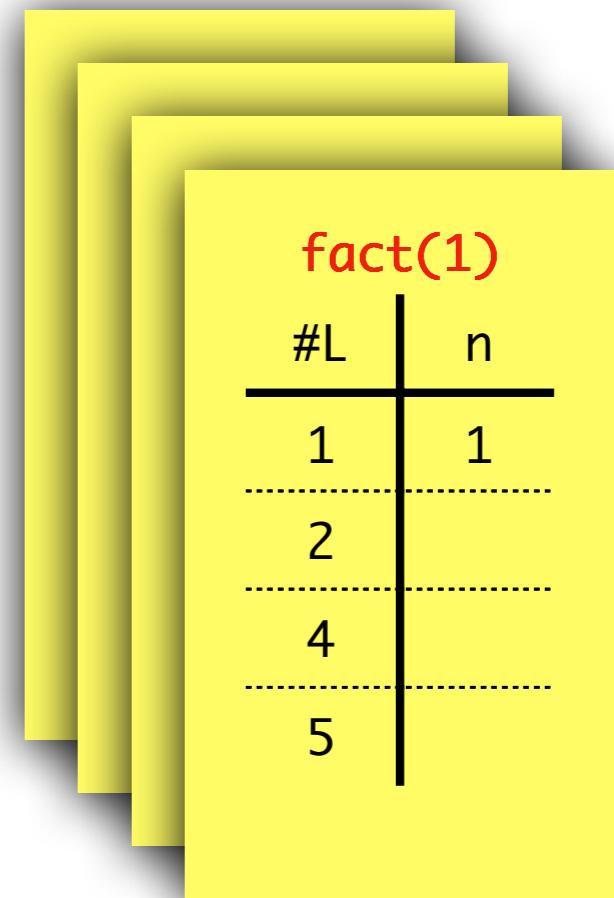
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



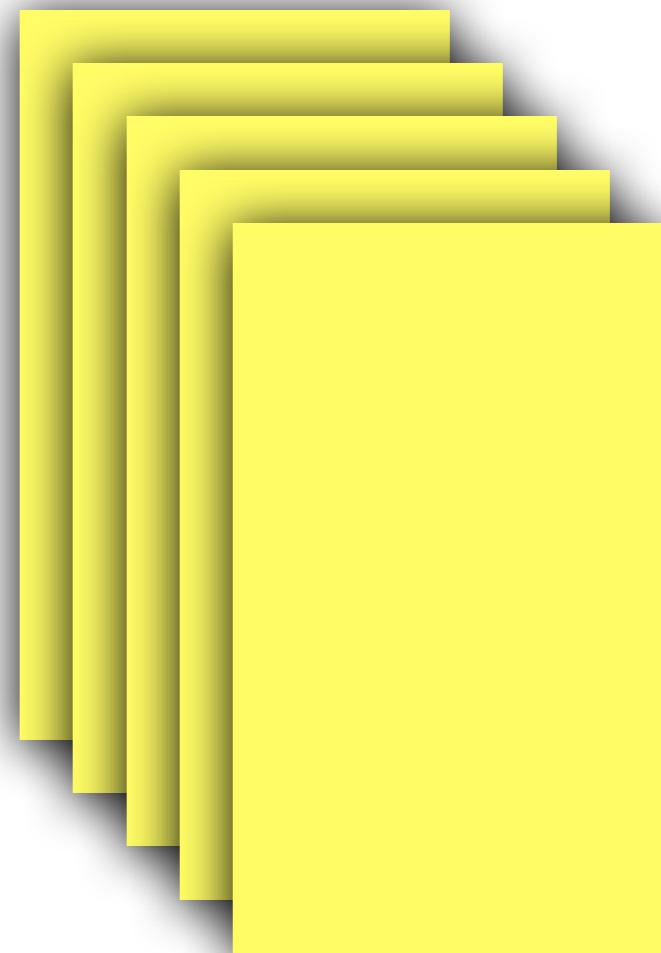
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



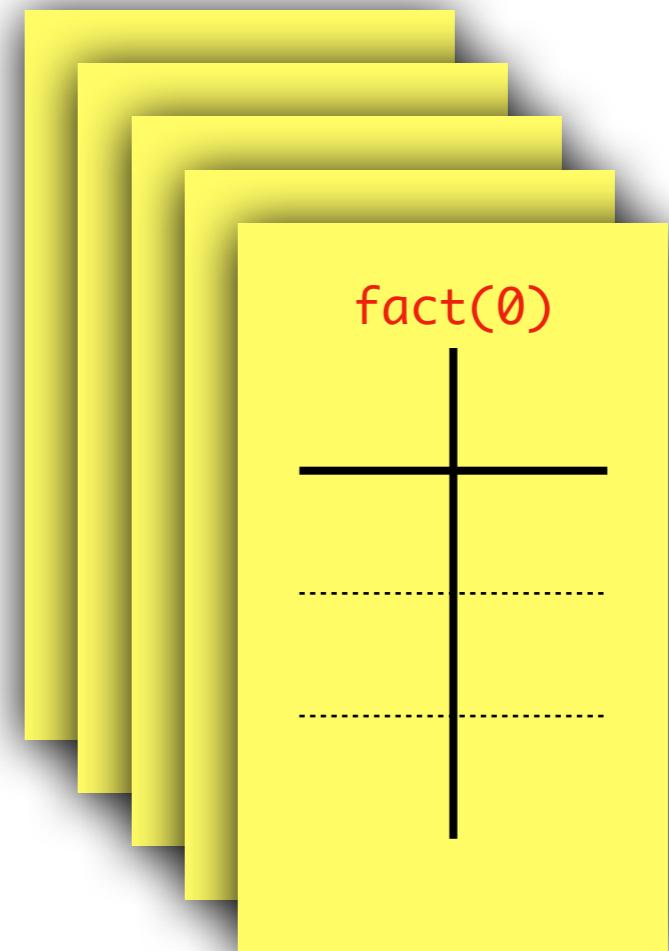
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



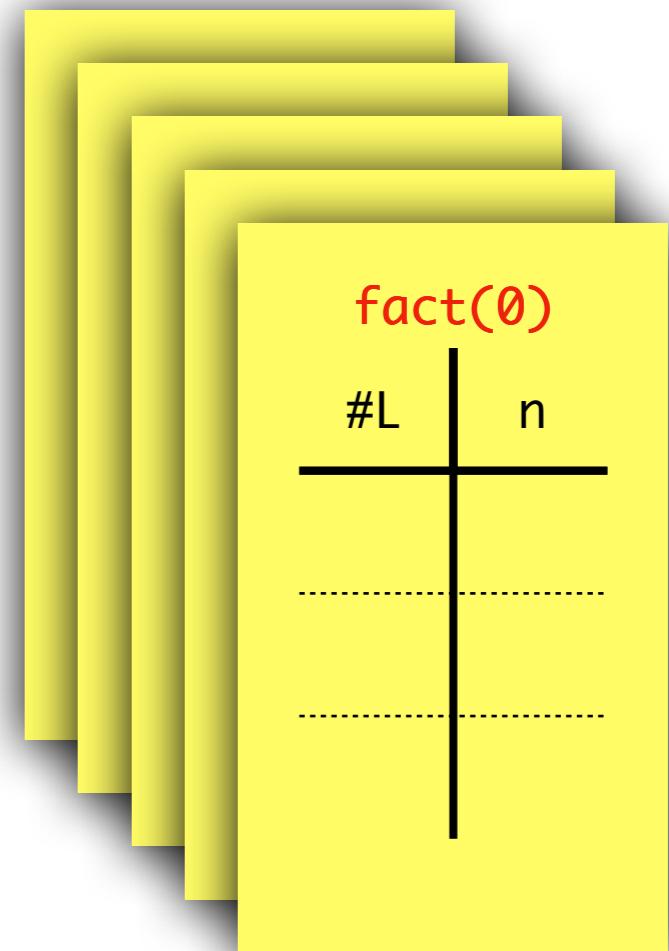
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



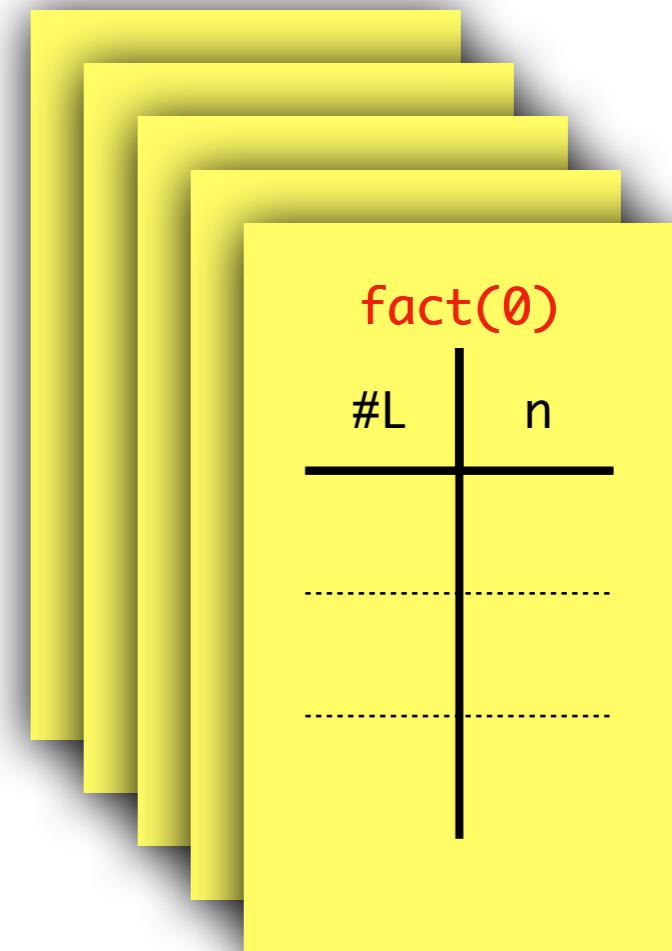
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



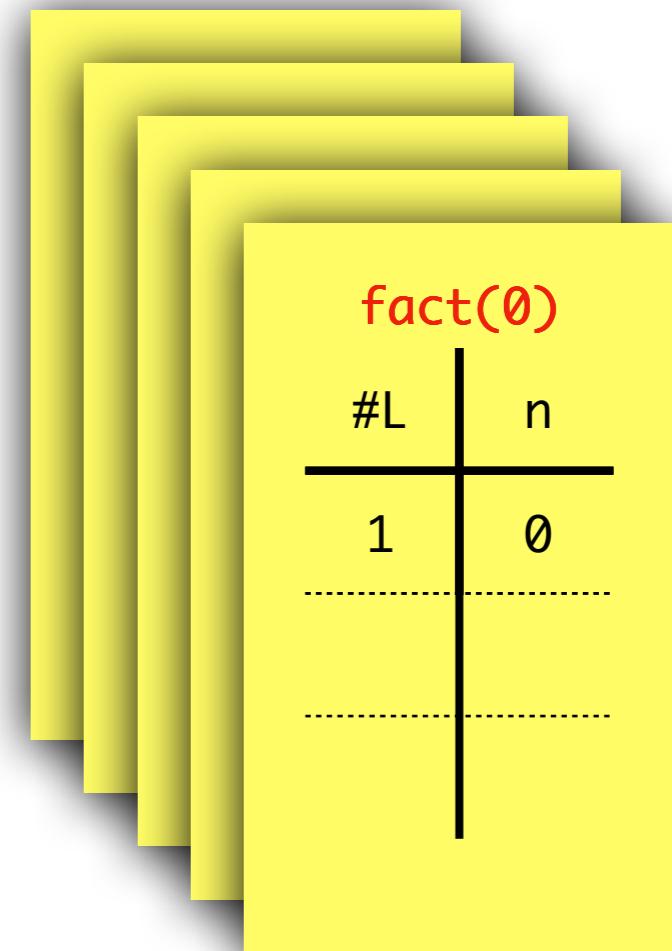
Calcul de $4! = 24$

👉 1 | **def fact(n):**
2 | **if n == 0:**
3 | **return 1**
4 | **else:**
5 | **return n * fact(n-1)**



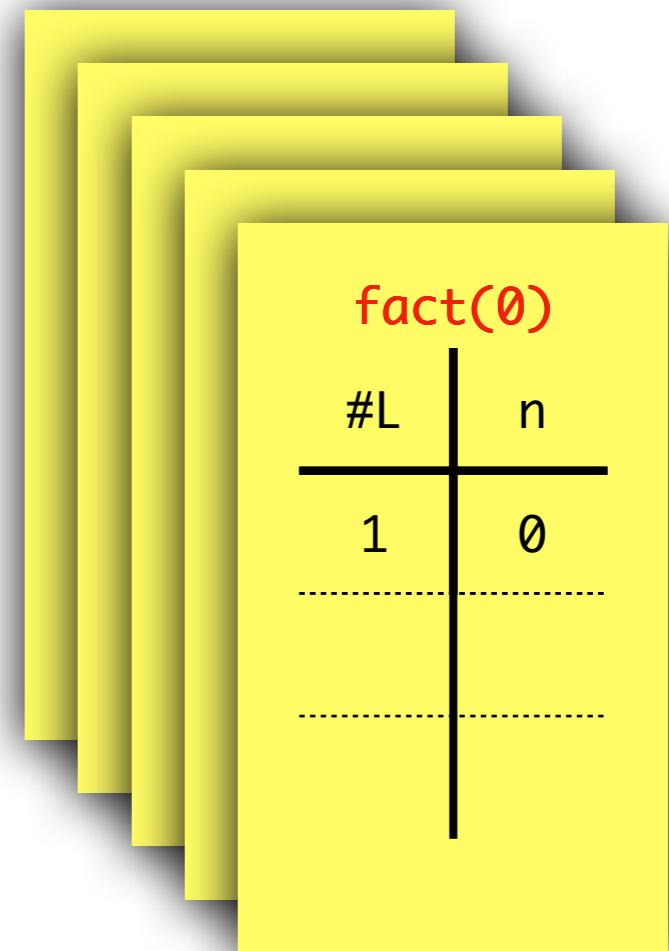
Calcul de $4! = 24$

👉 1 | **def fact(n):**
2 | **if n == 0:**
3 | **return 1**
4 | **else:**
5 | **return n * fact(n-1)**



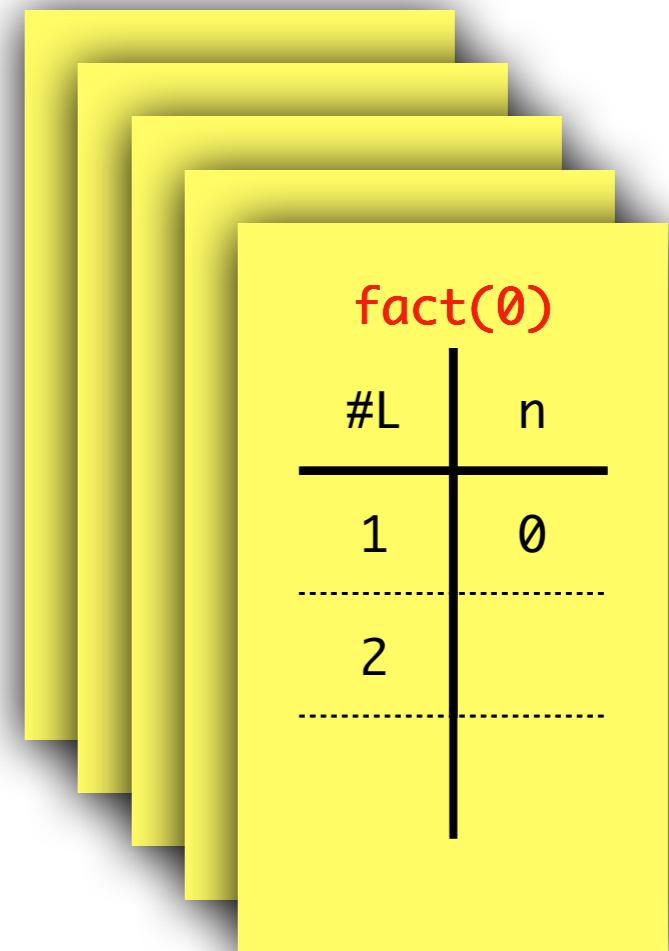
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



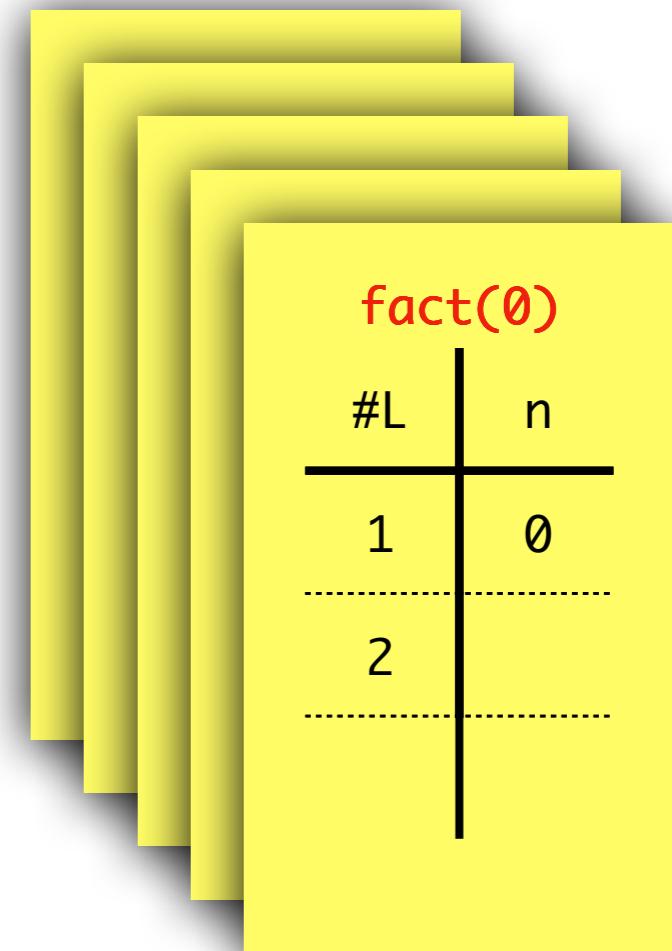
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



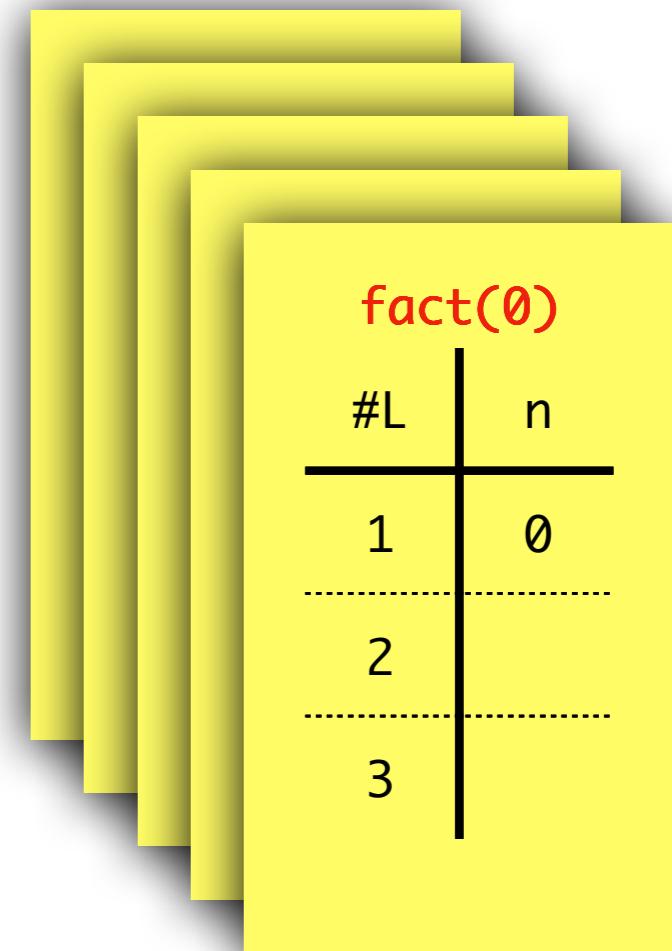
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



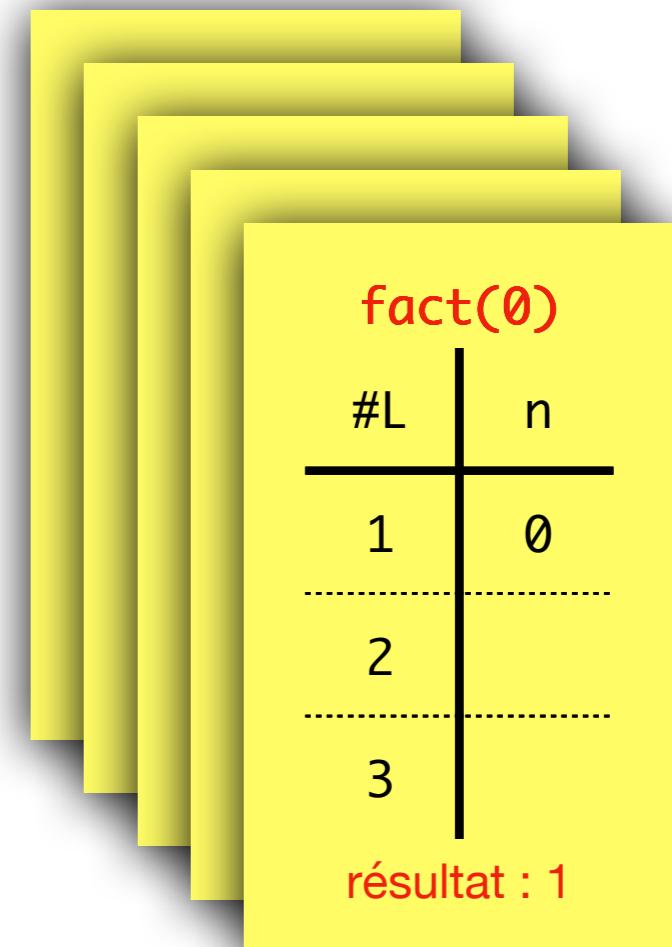
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



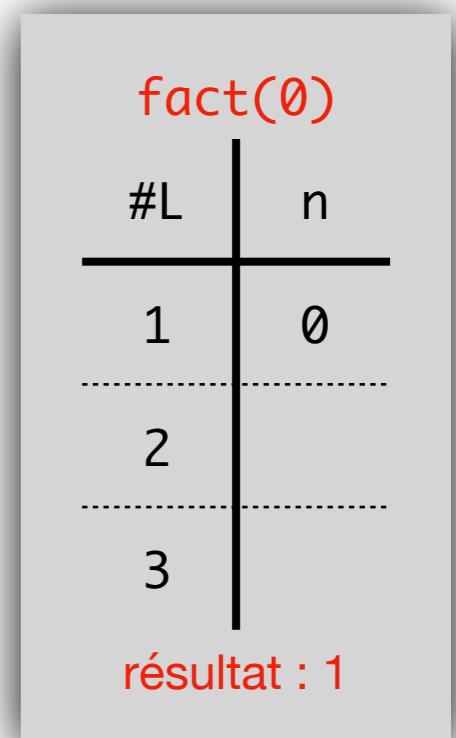
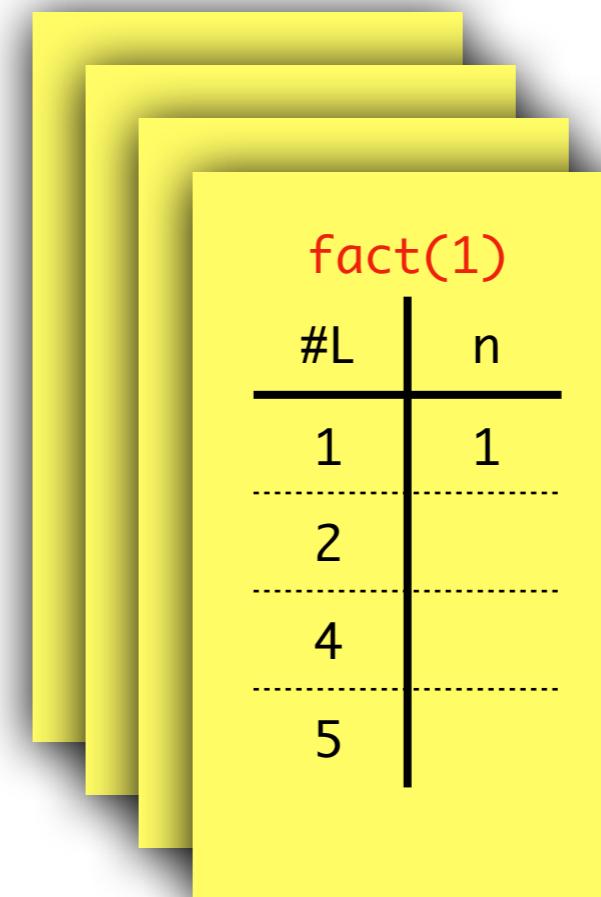
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



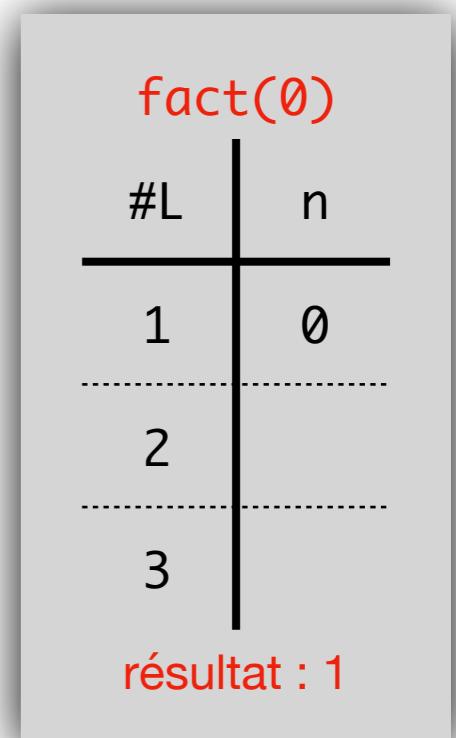
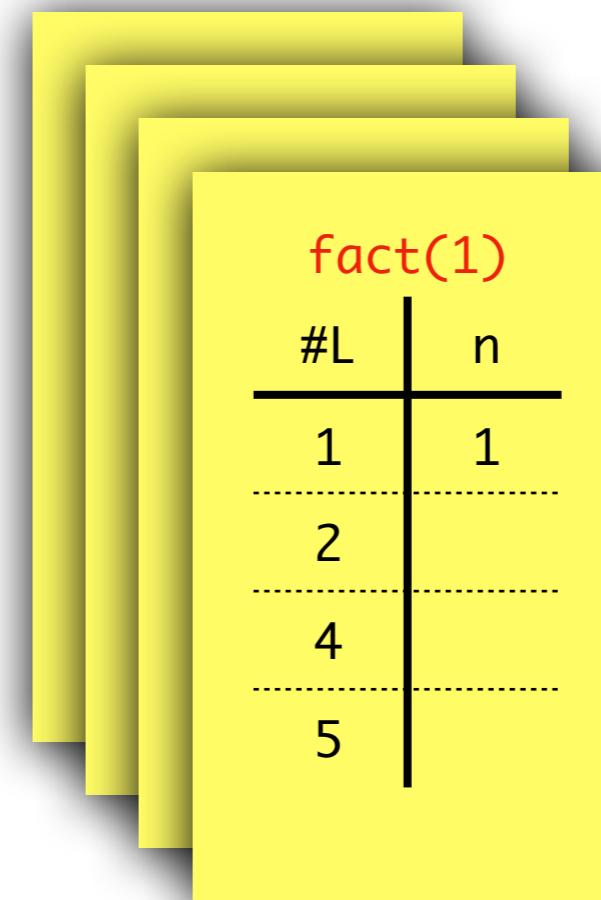
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```



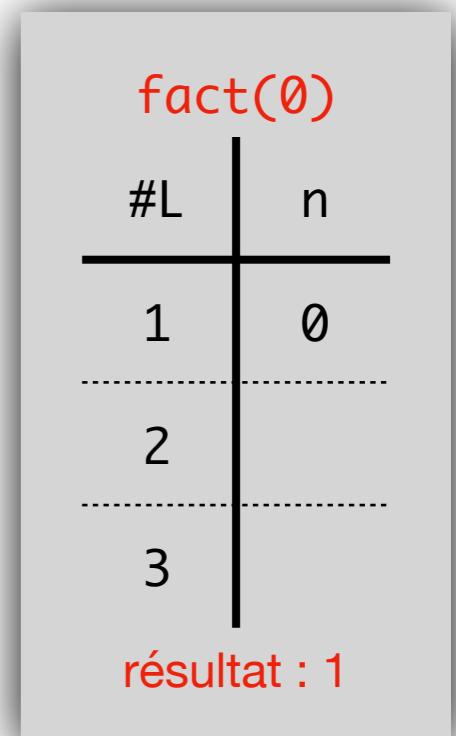
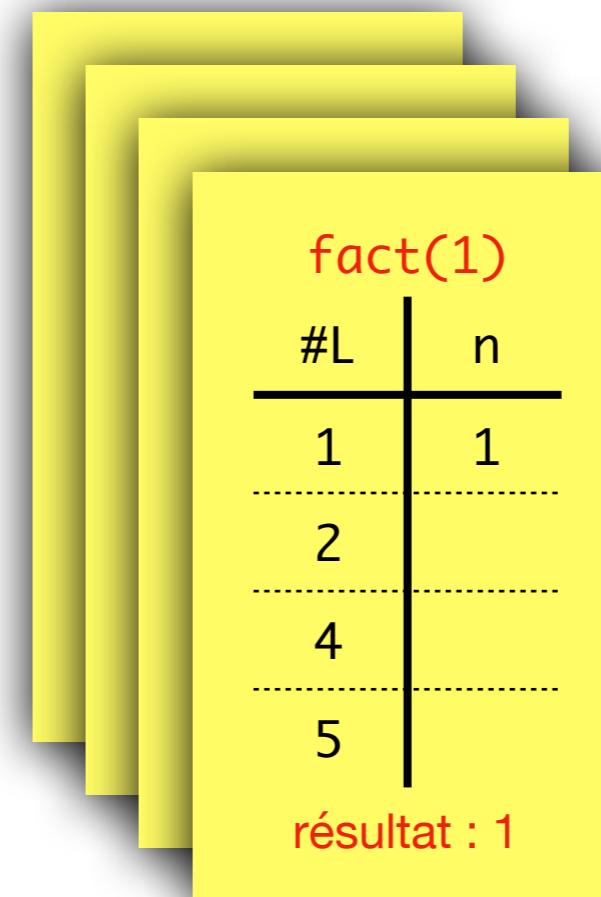
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         ➡️ return n * fact(n-1)
```



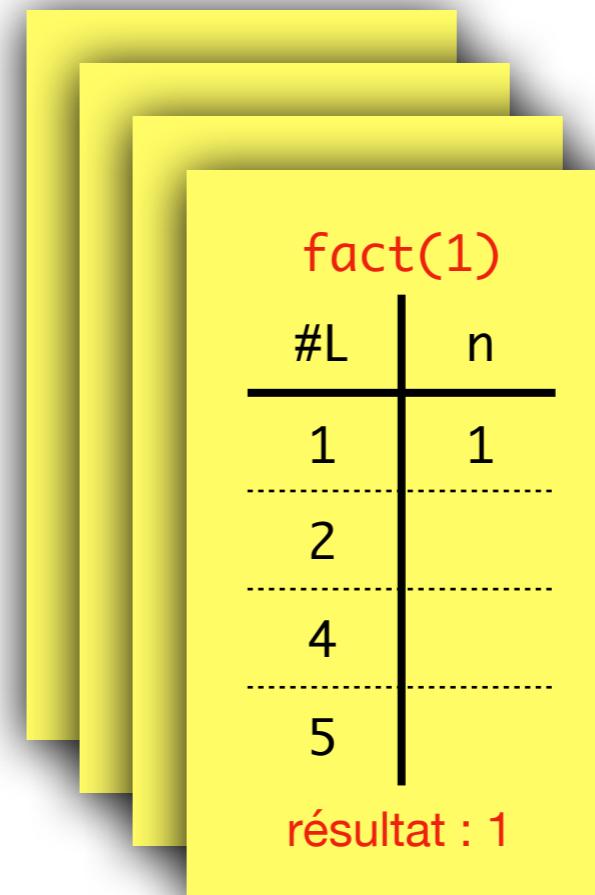
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         ➡️ return n * fact(n-1)
```



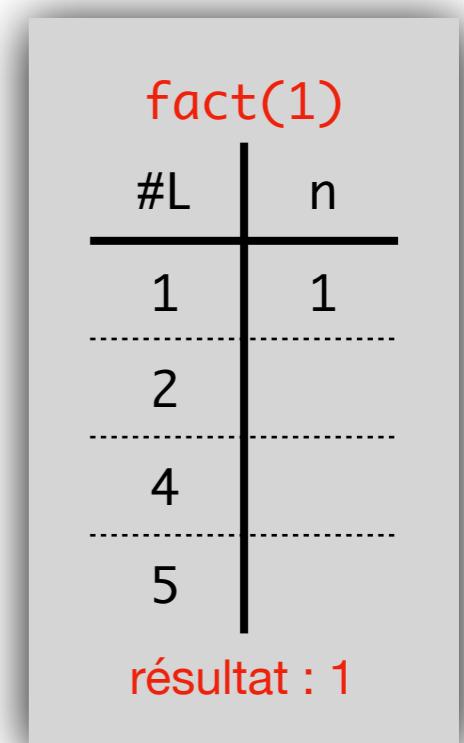
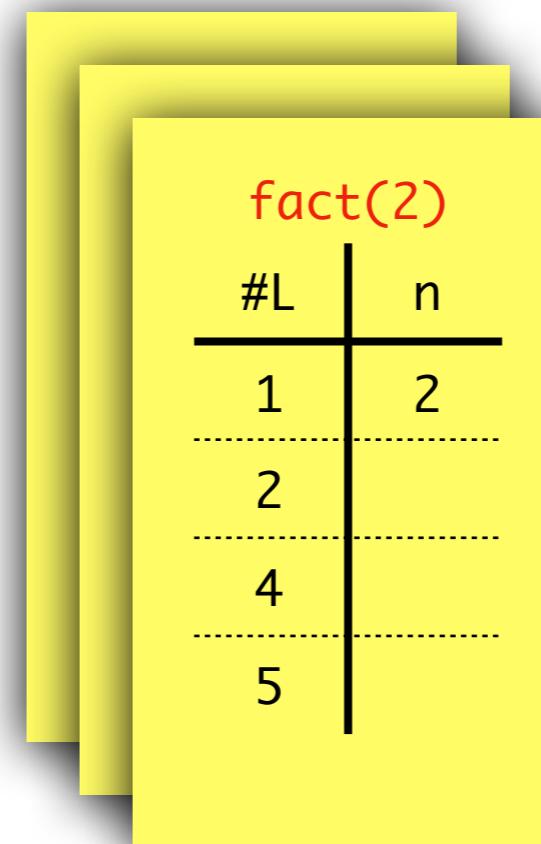
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         ➡️ return n * fact(n-1)
```



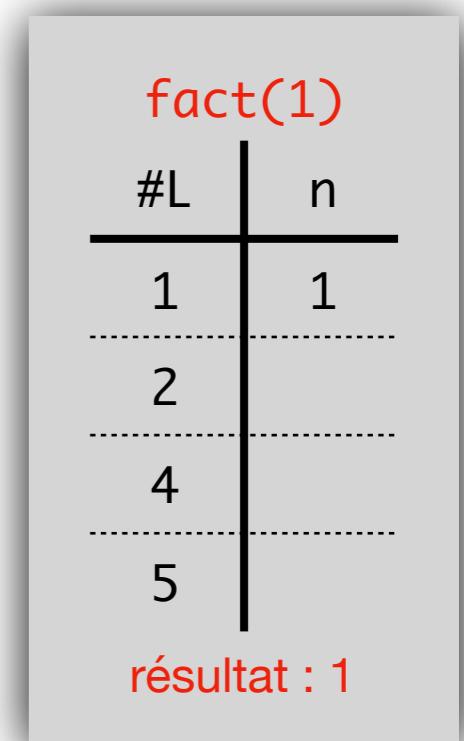
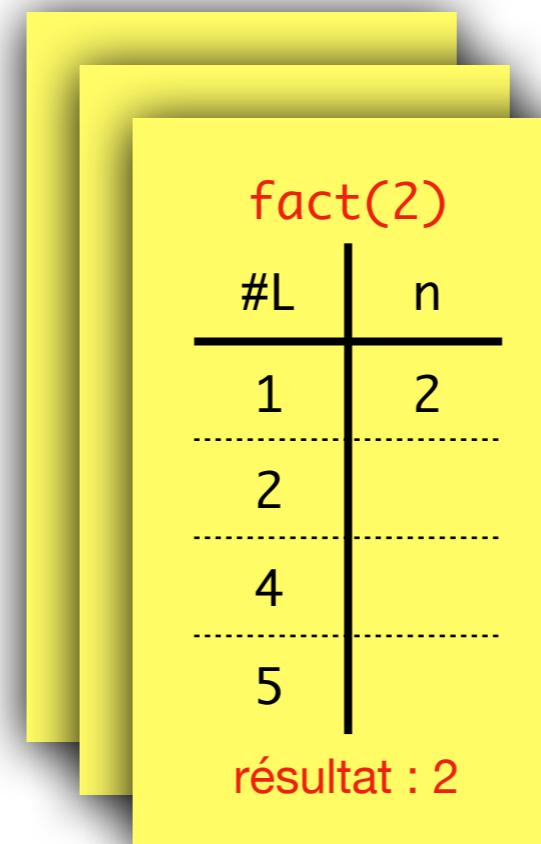
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



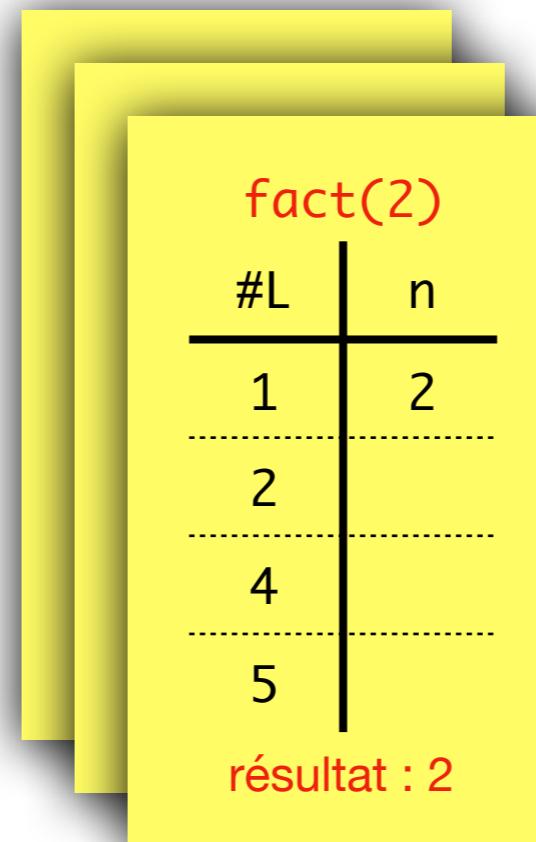
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



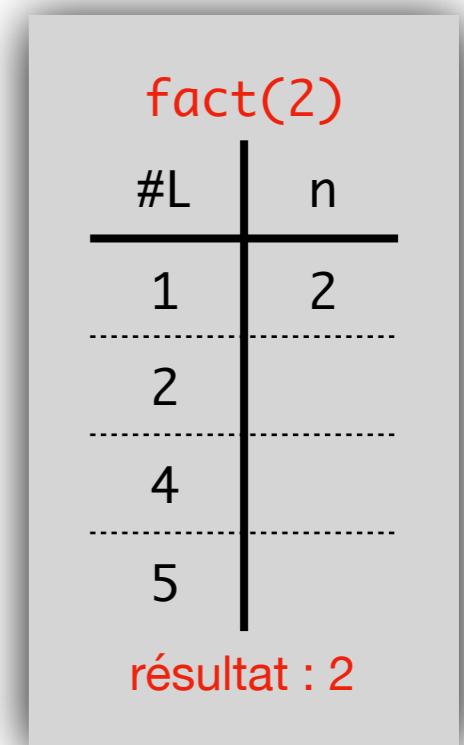
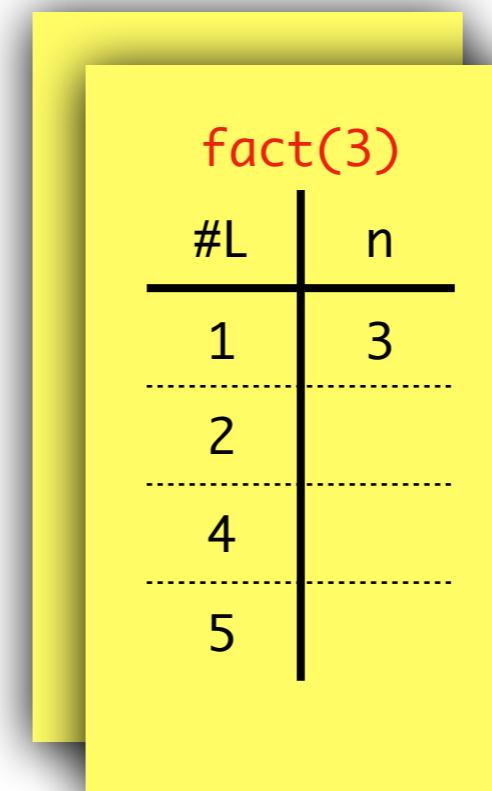
Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         ➡️ return n * fact(n-1)
```



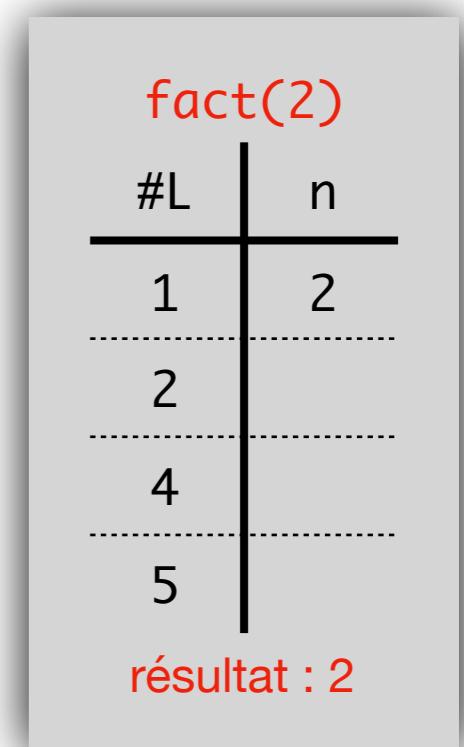
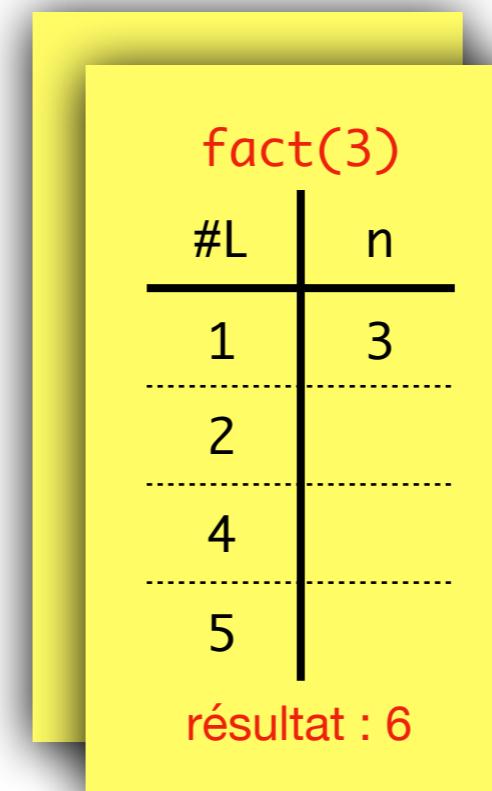
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



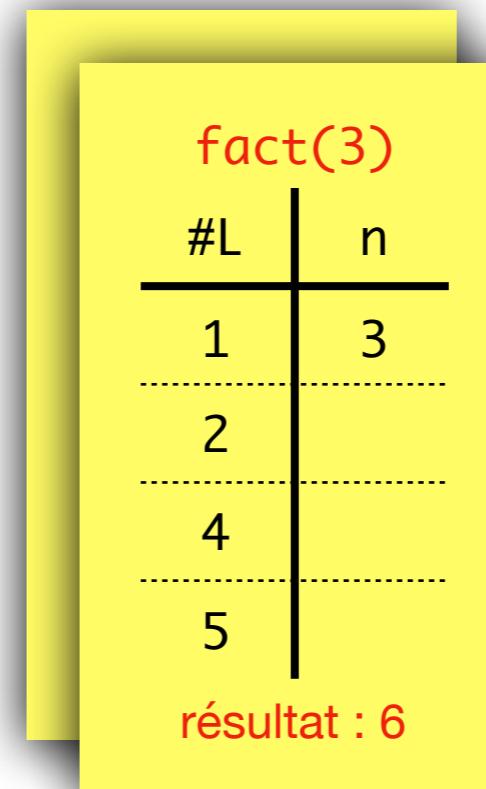
Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```



Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	
5	

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

Calcul de $4! = 24$

```
1 def fact(n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         ➡️ return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	
5	
résultat : 24	

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

Calcul de $4! = 24$

```
1 | def fact(n):  
2 |     if n == 0:  
3 |         return 1  
4 |     else:  
5 |         return n * fact(n-1)
```

fact(4)	
#L	n
1	4
2	
4	
5	
résultat : 24	

Nombre d'operations : 19

fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)	
#L	n
1	0
2	
3	

résultat : 1

Nombre d'operations : 19

fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)	
#L	n
1	0
2	
3	

résultat : 1

4 + 4 + 4 + 4 + 3

Terminaison d'un algorithme récursif

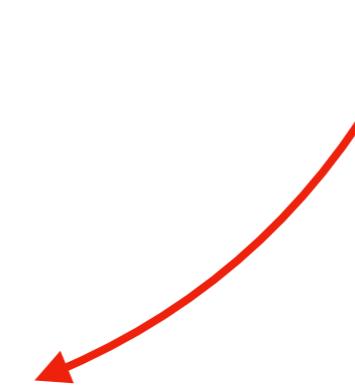
Terminaison de fact

```
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)
```

Terminaison de fact

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n - 1)
```

*si $n = 0$ alors
on s'arrête*



Terminaison de fact

```
def fact(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact(n - 1)
```

si $n = 0$ alors
on s'arrête

sinon on appelle
 $fact(n-1)$ et
puis on s'arrête

Terminaison de fact

```
def fact(n):  
    if n == 0:  
        return 1  
    else:
```

si $n = 0$ alors
on s'arrête

```
        return n * fact(n - 1)
```

sinon on appelle
 $fact(n-1)$ et
puis on s'arrête

et $fact(n-1)$ a un
paramètre plus petit,
qui se rapproche de 0

Code récursif (erroné !)

```
def fact2(n):  
    return n * fact2(n - 1)
```

Code récursif (erroné !) sans cas de base

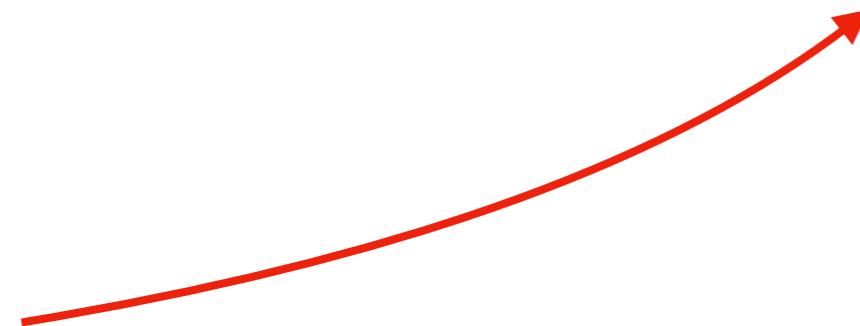
```
def fact2(n):  
    return n * fact2(n - 1)
```

Code récursif (erroné !)

sans cas de base

```
def fact2(n):  
    return n * fact2(n - 1)
```

on appelle
`fact2(n-1)`



Code récursif (erroné !)

sans cas de base

```
def fact2(n):  
    return n * fact2(n - 1)
```

on appelle
fact2(n-1)

...qui appelle fact2(n-2),
qui appelle fact2(n-3),
qui appelle fact2(n-4)...

Code récursif (erroné !)

```
def fact3(n):
    if n == 0:
        return 1
    else:
        return n * fact3(n)
```

Code récursif (erroné !) sans réduction de l'entrée

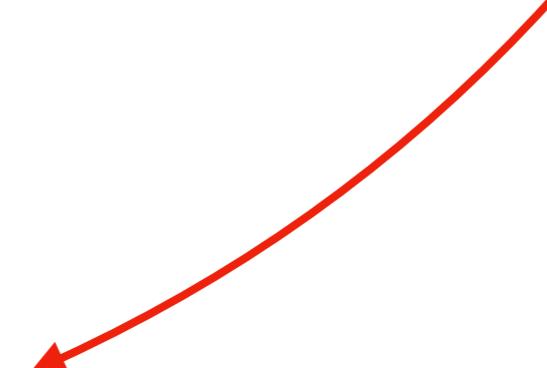
```
def fact3(n):
    if n == 0:
        return 1
    else:
        return n * fact3(n)
```

Code récursif (erroné !)

sans réduction de l'entrée

```
def fact3(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact3(n)
```

si $n = 0$ alors
on s'arrête



Code récursif (erroné !)

sans réduction de l'entrée

```
def fact3(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact3(n)
```

si $n = 0$ alors
on s'arrête

mais si $n > 0$
alors on appelle
`fact3(n)`

Code récursif (erroné !)

sans réduction de l'entrée

```
def fact3(n):  
    if n == 0:  
        return 1  
    else:  
        return n * fact3(n)
```

si $n = 0$ alors
on s'arrête

mais si $n > 0$
alors on appelle
`fact3(n)`

...qui appelle `fact3(n)`,
qui appelle `fact3(n)`...

Exercice 1 du TD6

Somme d'un tableau

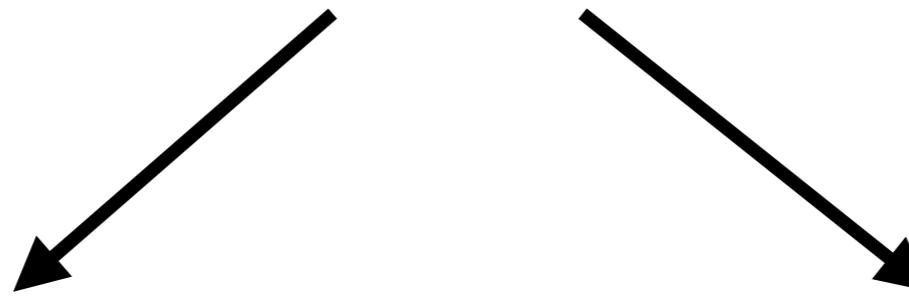
« diviser pour régner »

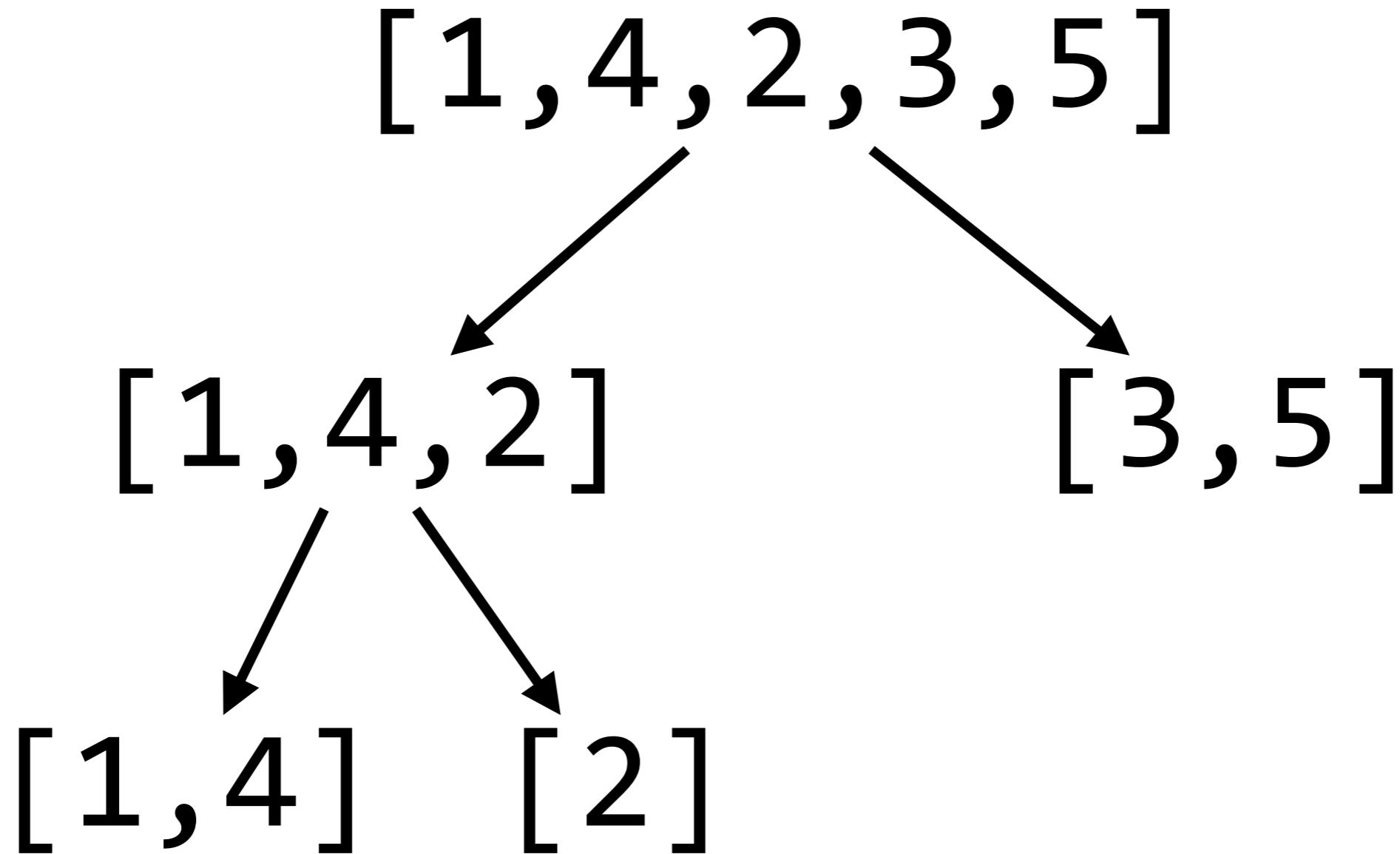
[1,4,2,3,5]

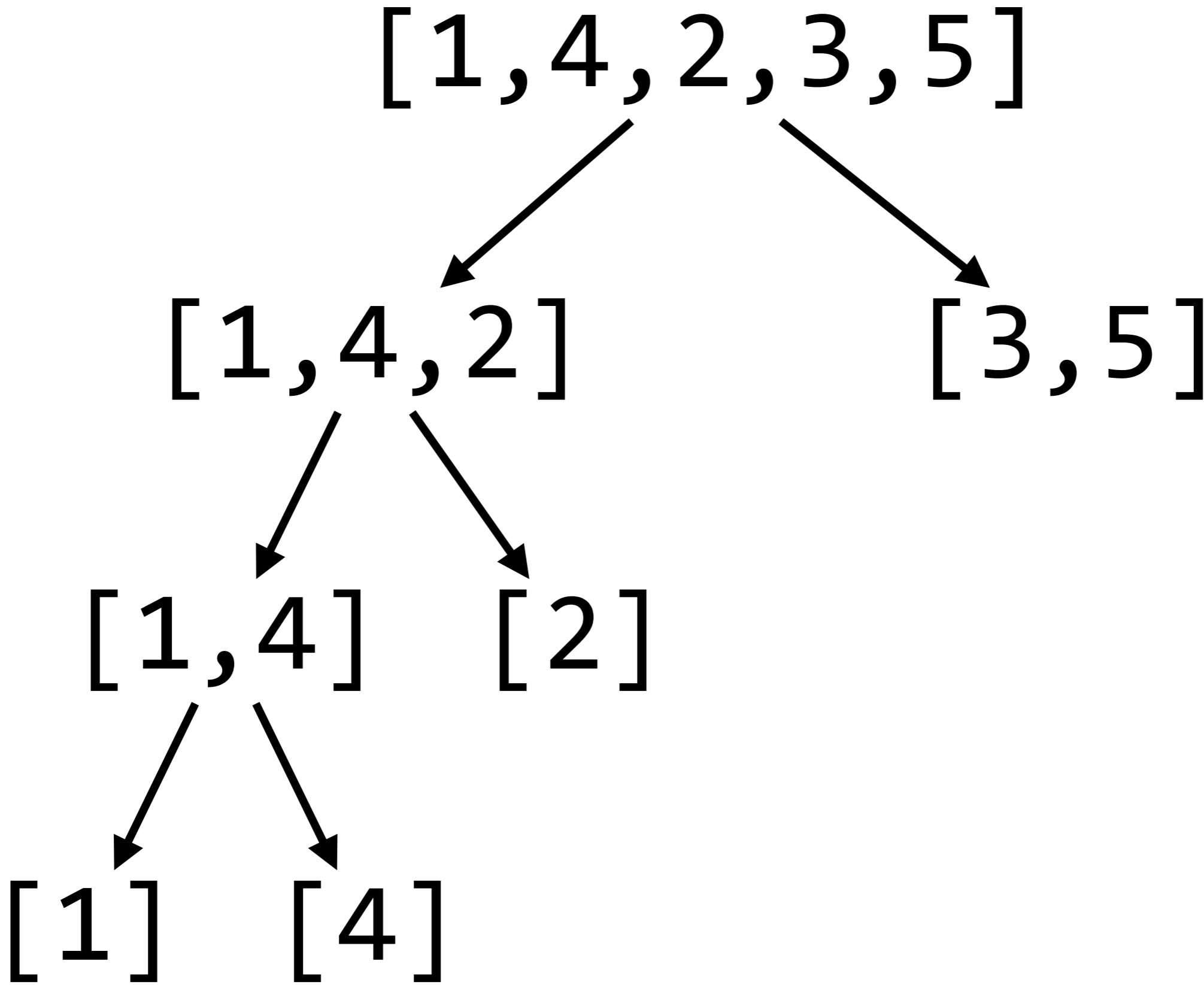
[1,4,2,3,5]

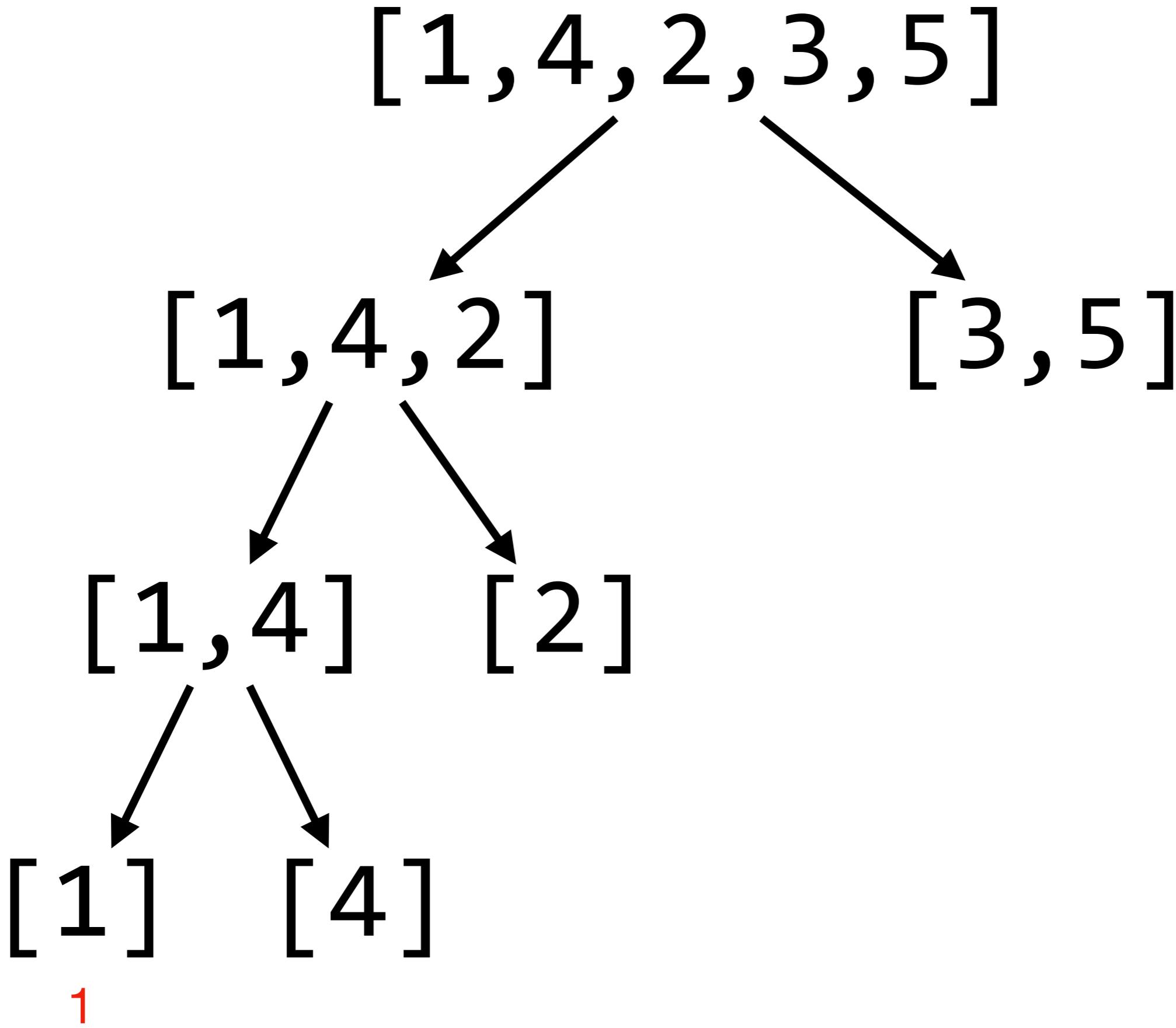
[1,4,2]

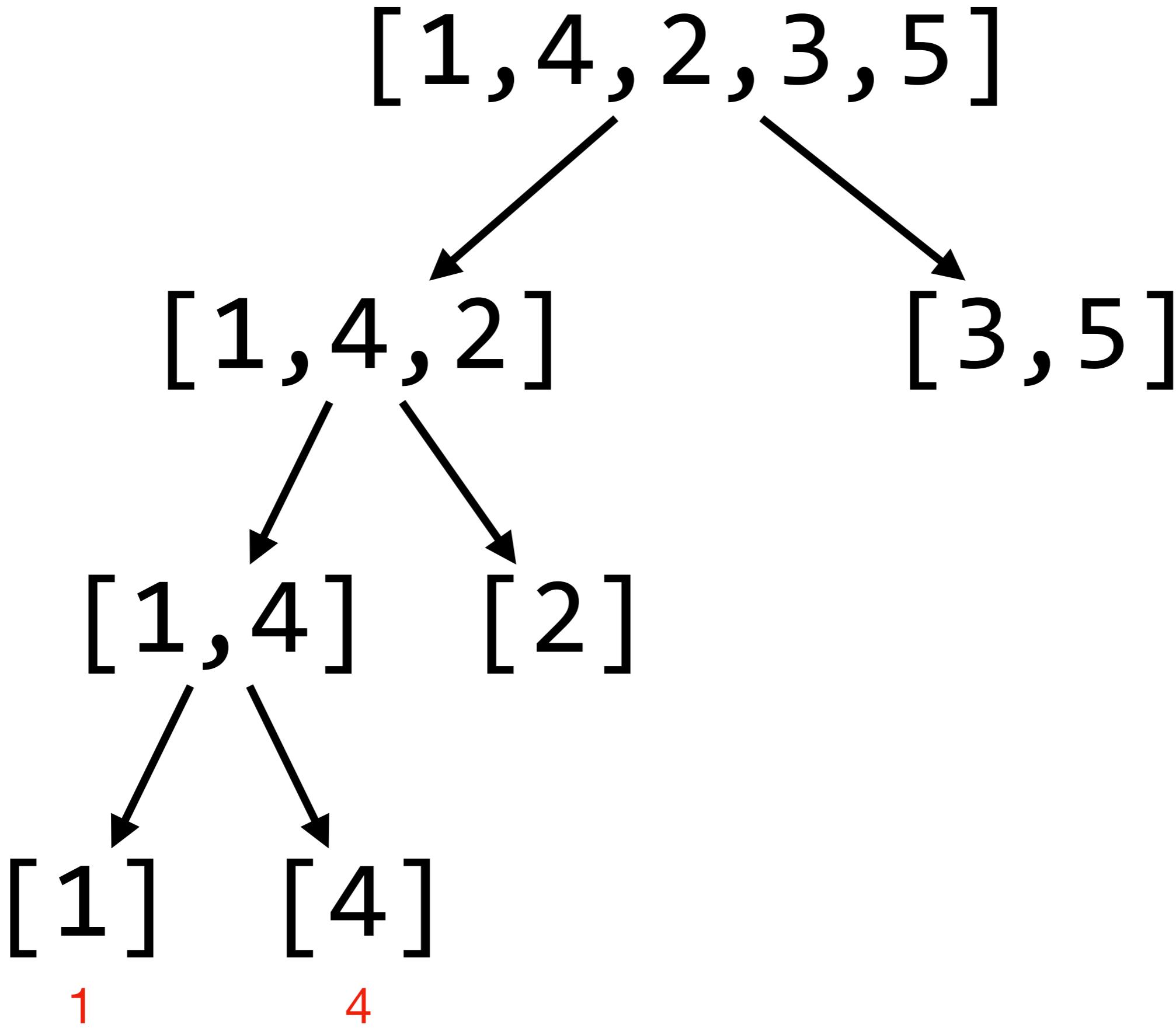
[3,5]

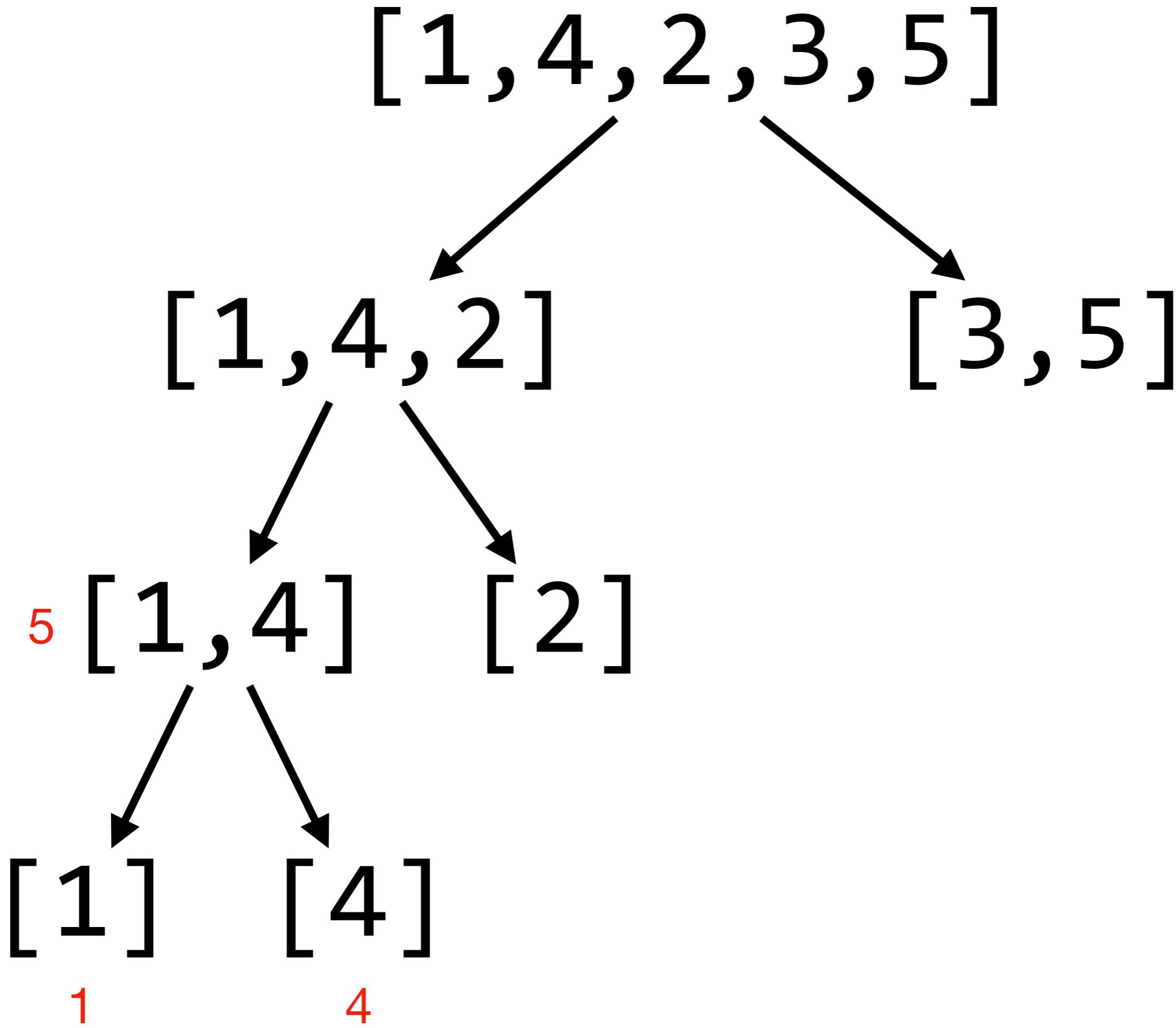


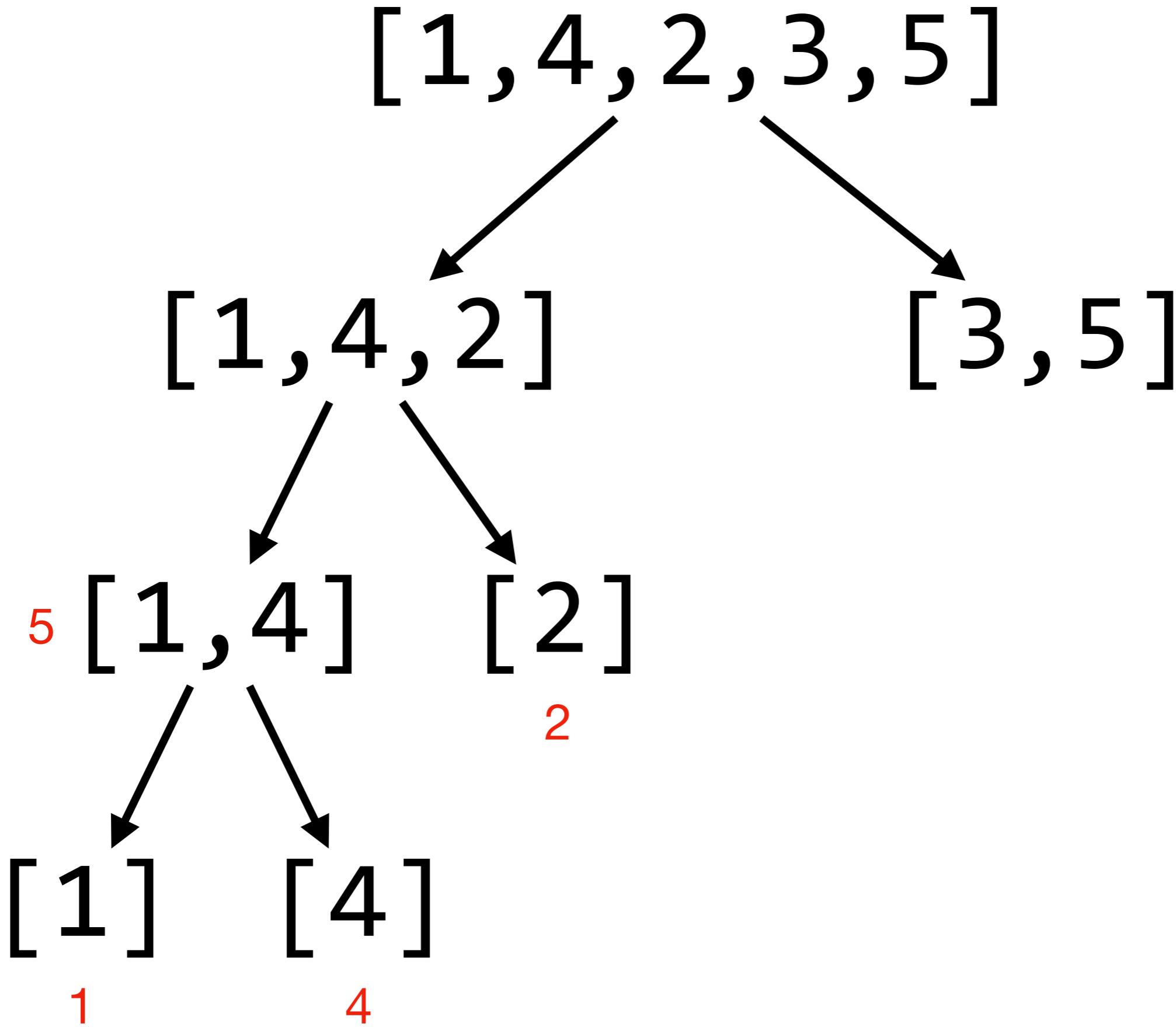


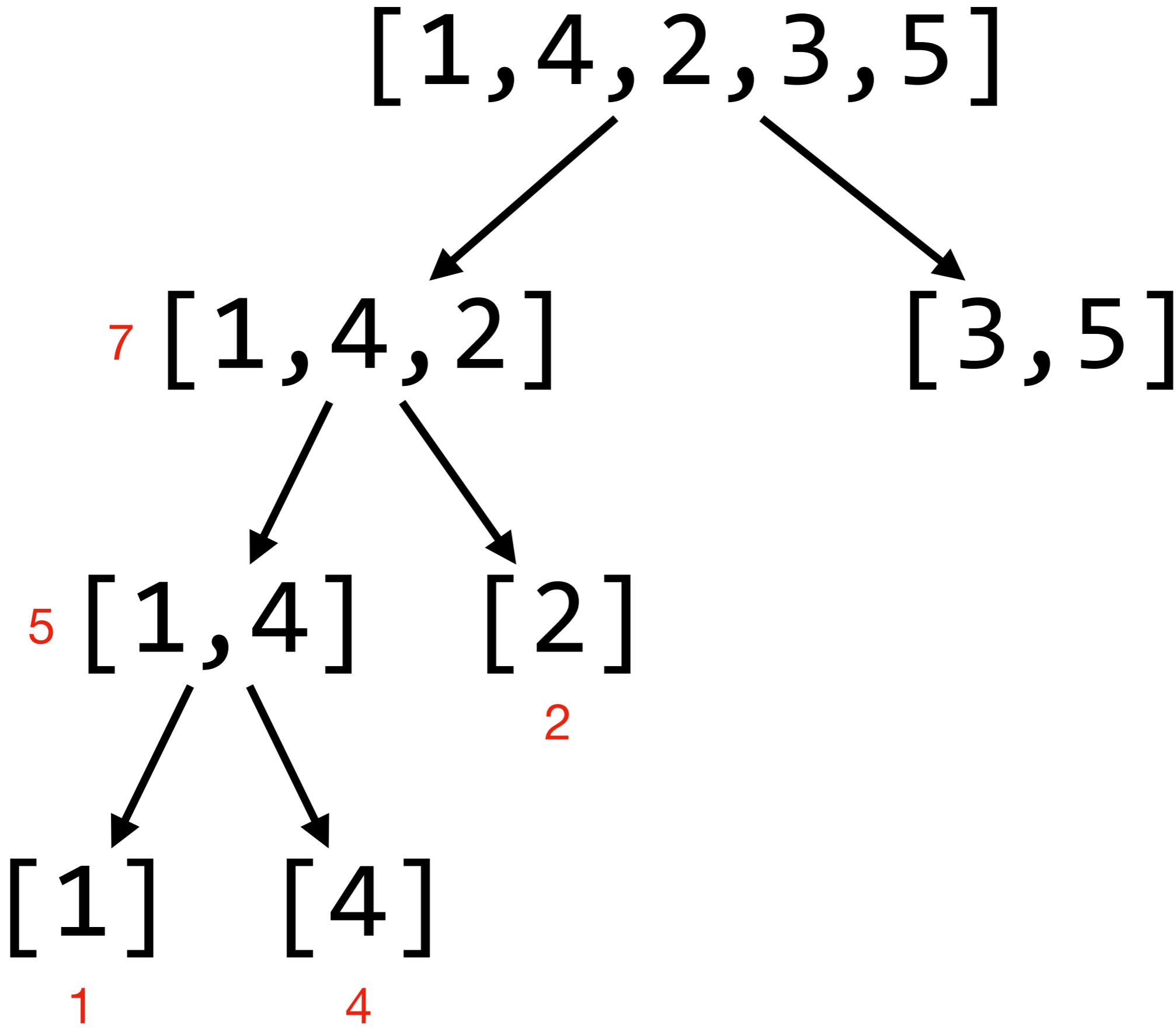


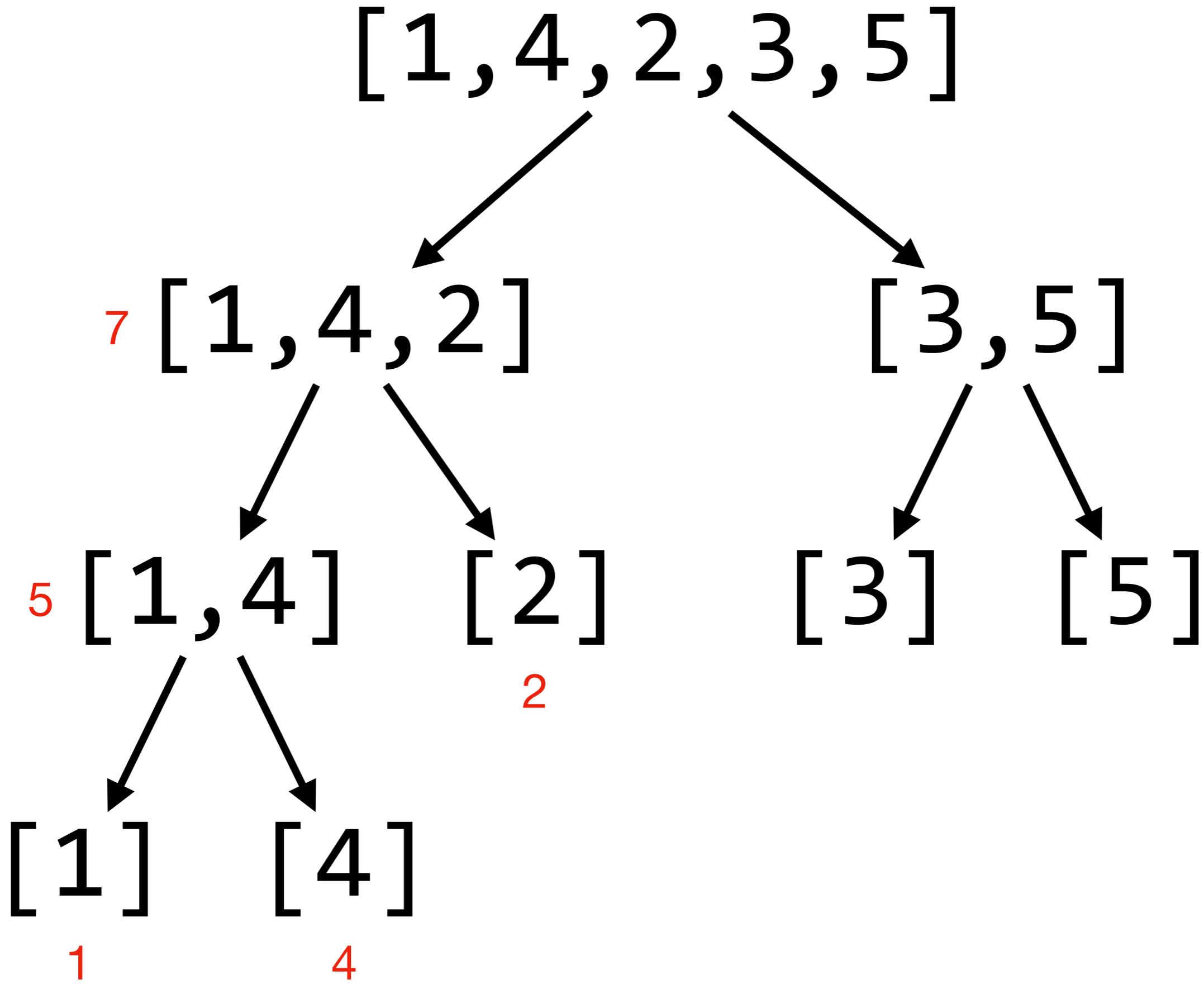


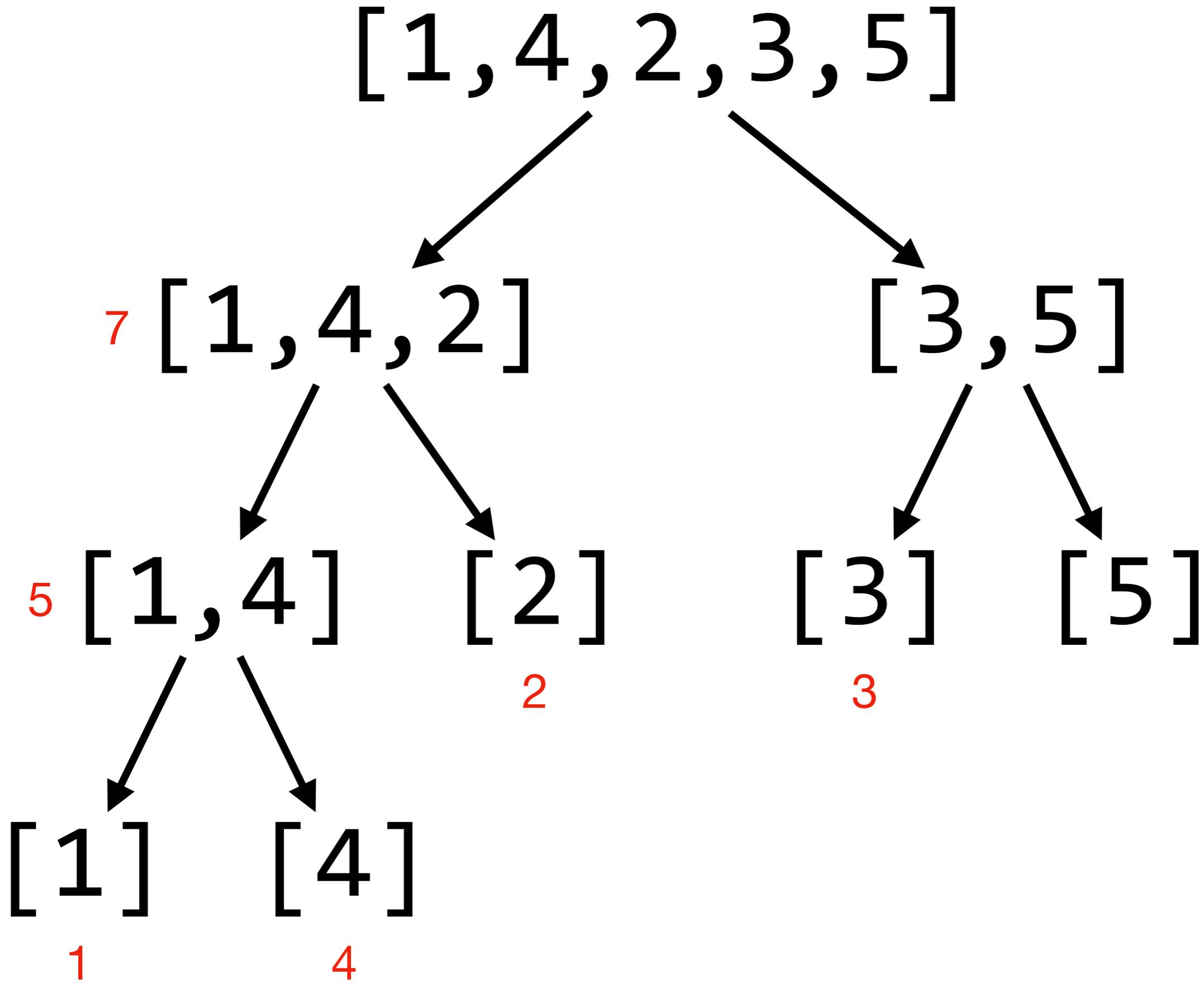


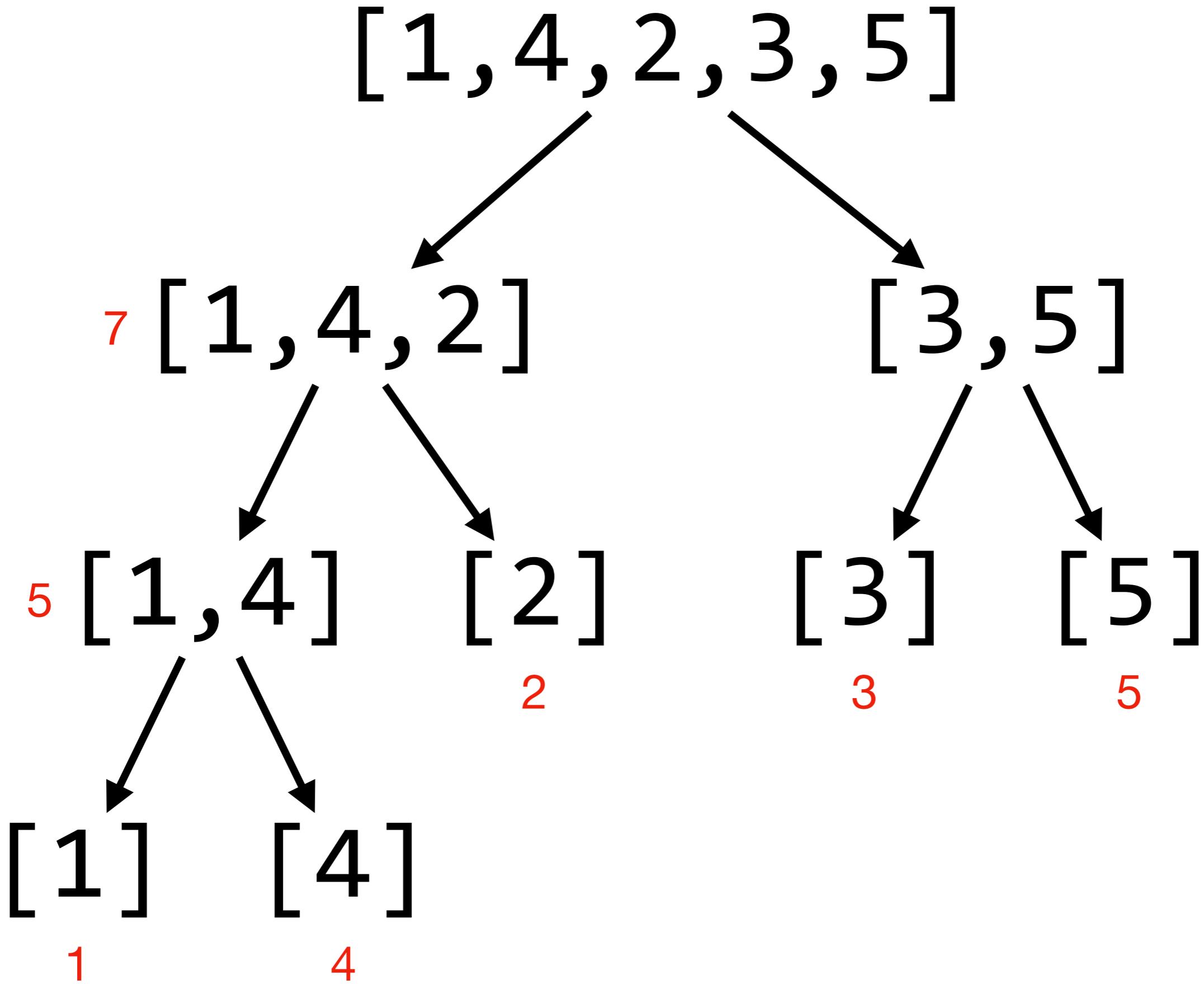


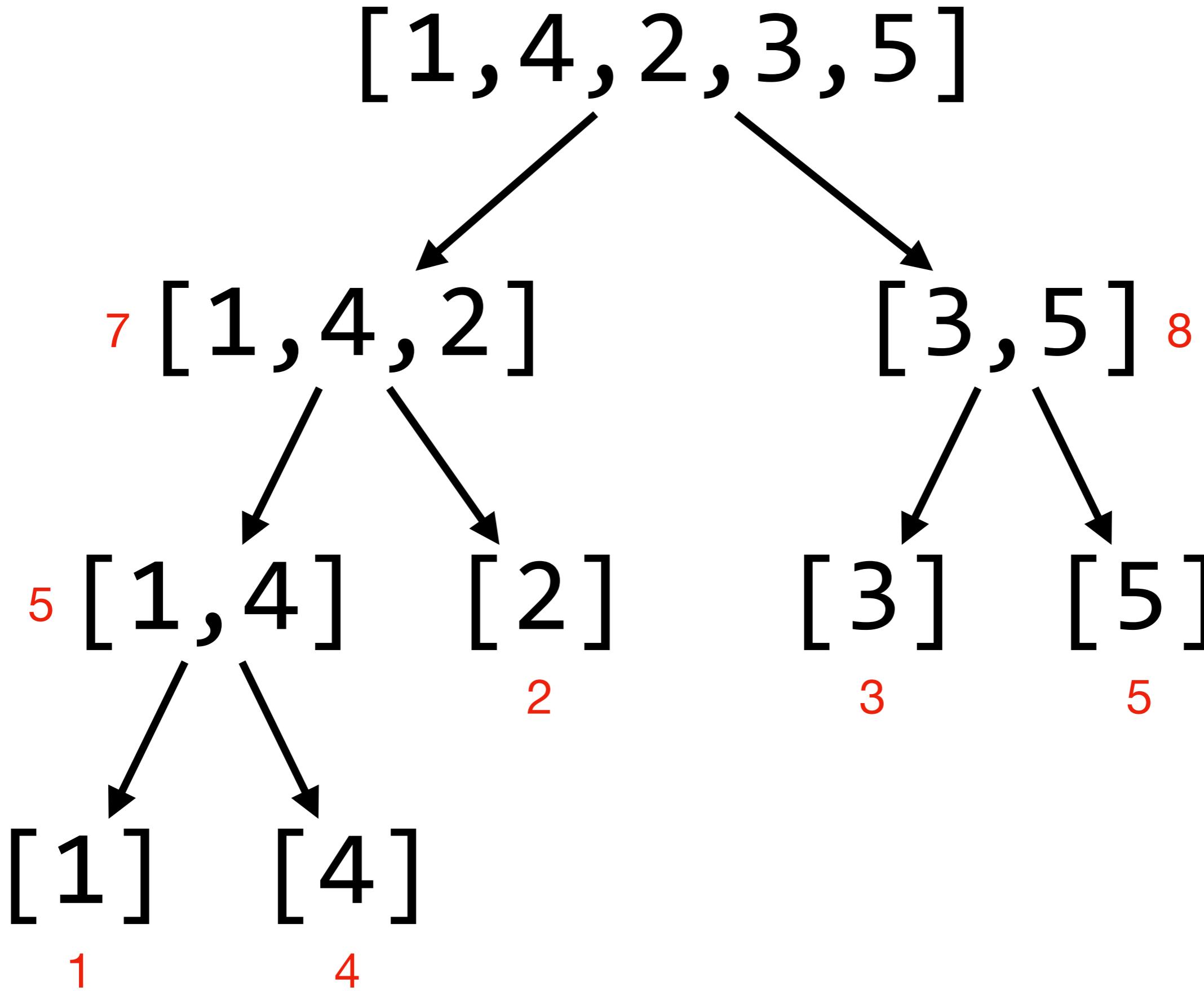


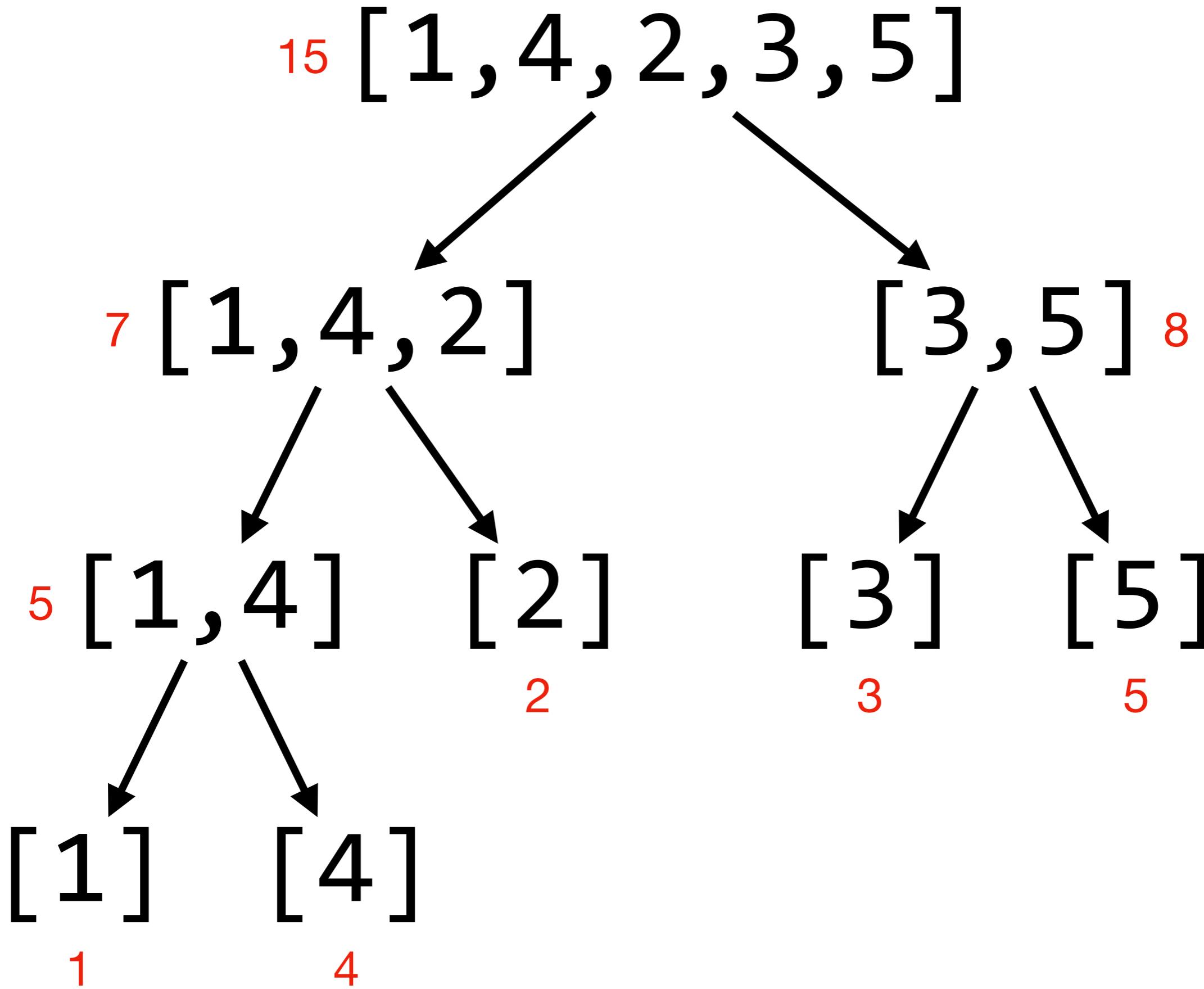












Diviser pour régner

- On veut calculer la somme d'un tableau A de longueur n , donc de la position 0 à la position $n - 1$
- On **divise** le problème en deux **sous-problèmes plus petits** :
 - Calculer **récursivement** la somme de la première moitié de A
 - Calculer **récursivement** la somme de la seconde moitié de A
- Enfin, on **combine** les deux (avec $+$) pour avoir le total
- Si le morceau de A est **très petit** (0 ou 1) **on résout directement**

Somme d'un tableau A de $A[i]$ à $A[j]$

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
        return l + r
```

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7							
8						2	
9							

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8						1	
9							

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8							0
9							
10							
11							

Somme de [1]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2					1		
3							
5							
6							

résultat : 1

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
11							

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							
11							

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							

Somme de [4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1, 4, 2, 3, 5]	1	1				
2						1	
3							
5							
6							
7							
10							
résultat : 4							
11							

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							4

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							4

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							4
11							

Somme de [1, 4]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2						2	
3							
5							
7							
8						0	
9							1
10							4
11							

résultat : 5

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8						1	
9							5

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8						1	
9							5
10							

Somme de [2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2						1	
3							
5							
6							
7							
8							
9							
10							
résultat : 2							
10							

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

Somme de [1, 4, 2]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7						2	
8							7
9							

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7						2	
8							
9						7	

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7						2	
8							7
9							
10							

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9							
11							

Somme de [3]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2						1	
3							
5							
6							
7							
8							
9							
10							
11							

résultat : 3

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9						3	
11							

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8				3			
9					3		
11						3	

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9						3	
10							

Somme de [5]

```
1 def sum_array(A, i, j):  
2     n = j - i + 1  
3     if n == 0:  
4         return 0  
5     elif n == 1:  
6         return A[i]  
7     else:  
8         m = (i + j) // 2  
9         l = sum_array(A, i, m)  
10        r = sum_array(A, m+1, j)  
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2						1	
3							
5							
6							
7							
8							
9							
10							
11							

résultat : 5

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9						3	
10							5

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8				3			
9					3		
10						5	

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9						3	
10							5
11							

Somme de [3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8					3		
9						3	
10							5
11							

résultat : 8

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7						2	
8							7
9							8
10							

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7						2	
8							7
9							
10							8

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7						2	
8							7
9							8
10							
11							

Somme de [1, 4, 2, 3, 5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

sum_array([1,4,2,3,5], 0, 4)

Somme de [1,4,2,3,5]

```
1 def sum_array(A, i, j):
2     n = j - i + 1
3     if n == 0:
4         return 0
5     elif n == 1:
6         return A[i]
7     else:
8         m = (i + j) // 2
9         l = sum_array(A, i, m)
10        r = sum_array(A, m+1, j)
11        return l + r
```

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7						2	
8							7
9							8
10							
11							

résultat : 15

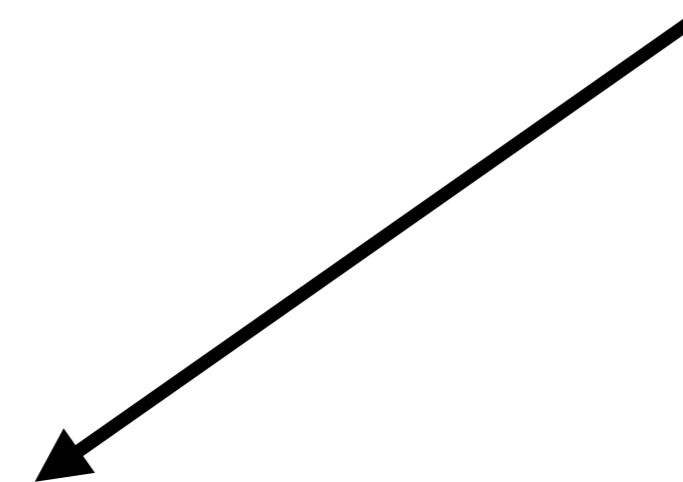
Arbre de récursion



sum_array([1,4,2,3,5], 0, 4)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7							
8				1			
9					5		
10						2	
11							

résultat : 7

sum_array([1,4,2,3,5], 0, 4)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

sum_array([1,4,2,3,5], 0, 2)

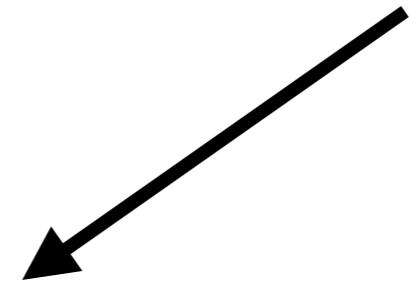
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7					1		
8						5	
9							2
10							
11							

résultat : 7

sum_array([1,4,2,3,5], 0, 1)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7					0		
8						1	
9							4
10							
11							

résultat : 5



sum_array([1,4,2,3,5], 0, 4)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

sum_array([1,4,2,3,5], 0, 2)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

sum_array([1,4,2,3,5], 0, 1)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8					0		
9						1	
10							4
11							

résultat : 5

sum_array([1,4,2,3,5], 0, 0)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2				1			
3							
5							
6							

résultat : 1



sum_array([1,4,2,3,5], 0, 4)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

sum_array([1,4,2,3,5], 0, 2)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

sum_array([1,4,2,3,5], 0, 1)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8					0		
9						1	
10							4
11							

résultat : 5

sum_array([1,4,2,3,5], 0, 0)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

résultat :



sum_array([1,4,2,3,5], 0, 4)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

sum_array([1,4,2,3,5], 0, 2)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

sum_array([1,4,2,3,5], 0, 1)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8					0		
9						1	
10							4
11							

résultat : 5

sum_array([1,4,2,3,5], 0, 0)

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

résultat :



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							
résultat : 15							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							
résultat : 7							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8					0		
9						1	
10							4
11							
résultat : 5							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							
résultat : ..							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							
résultat : 4							



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8						0	
9							1
10							4
11							

résultat : 5



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							

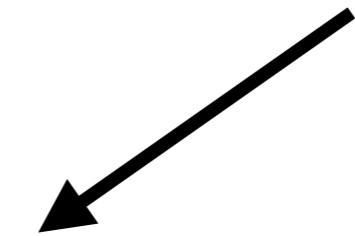
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9							7
10							8
11							
résultat : 15							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							
résultat : 7							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8					0		
9						1	
10							4
11							
résultat : 5							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							
résultat : 0							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							
résultat : 0							



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							
résultat : 15							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							
résultat : 7							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8							
9							
10							
11							
résultat : 6							



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							
résultat : 0							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							
résultat : 0							





#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

résultat : 7

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8							
9							
10							
11							

résultat : 4

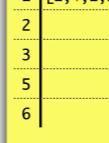


#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 0

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 0



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9							7
10							8
11							
résultat : 15							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10						2	
11							
résultat : 7							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2				1			
3							
5							
7							
8							
9							
10							
11							
résultat : 2							



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							
résultat : 0							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							
résultat : 0							



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9							7
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7					1		
8							
9					5		
10						2	
11							

résultat : 7



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5							
7							
8							
9							
10							
11							

résultat : 4

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2							
3							
5							
6							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

résultat : 0

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							

résultat : 15



sum_array([1,4,2,3,5], 0, 4)							
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

sum_array([1,4,2,3,5], 0, 2)							
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8					1		
9						5	
10							2
11							

#	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8							
9							
10							
11							

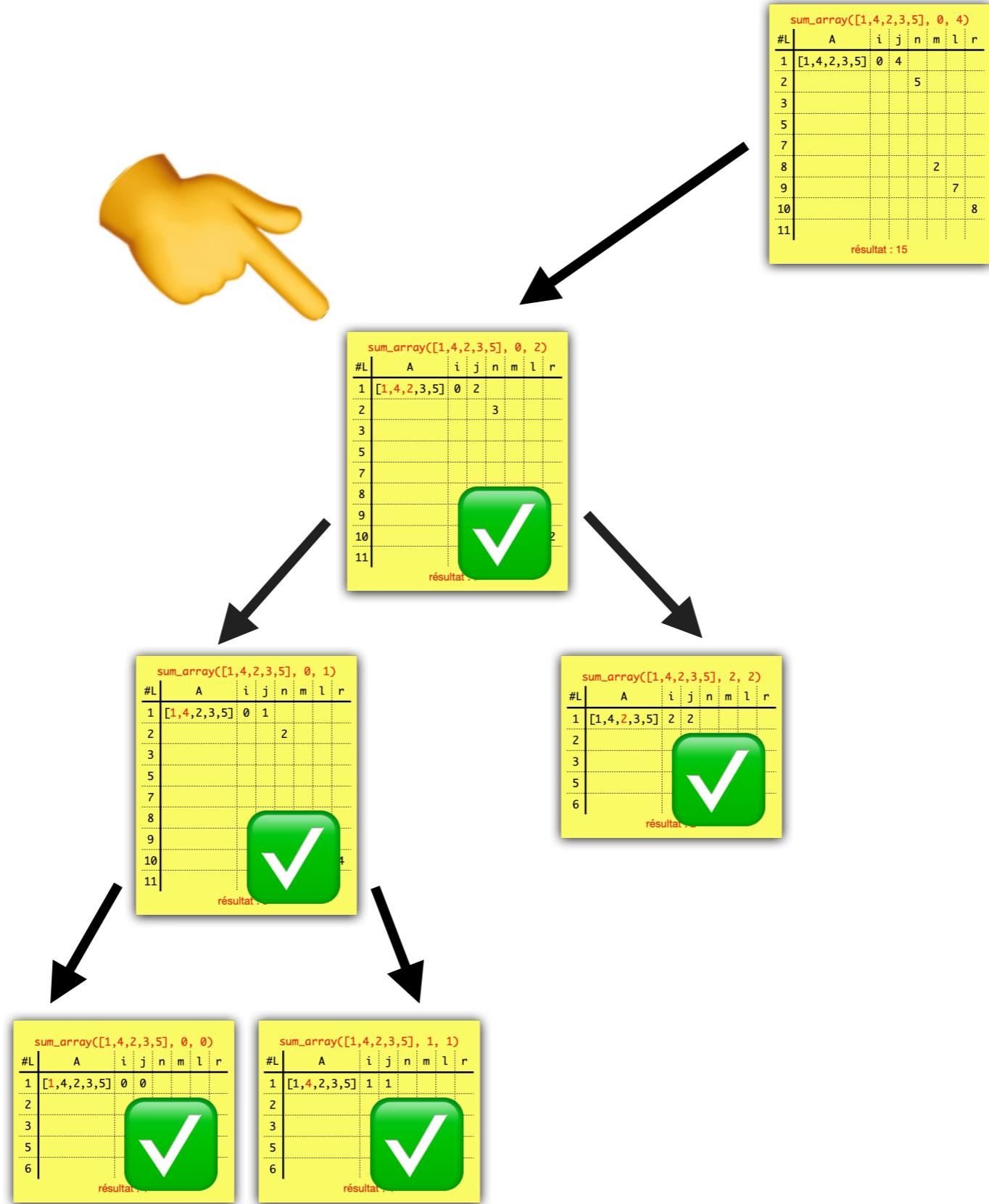
#L	A	i	j	n	m	l	r
1	[1,4, 2 ,3,5]	2	2				
2							
3							
5							
6							

réultat : 

sum_array([1,4,2,3,5], 0, 0)							
#	L	A	i	j	n	m	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

sum_array([1,4,2,3,5], 1, 1)							
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							

résultat : 





#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8					2		
9						7	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8						2	
9							
10							
11							

résultat : 15



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8							
9							
10							
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2							
3							
5							
6							

résultat : 15



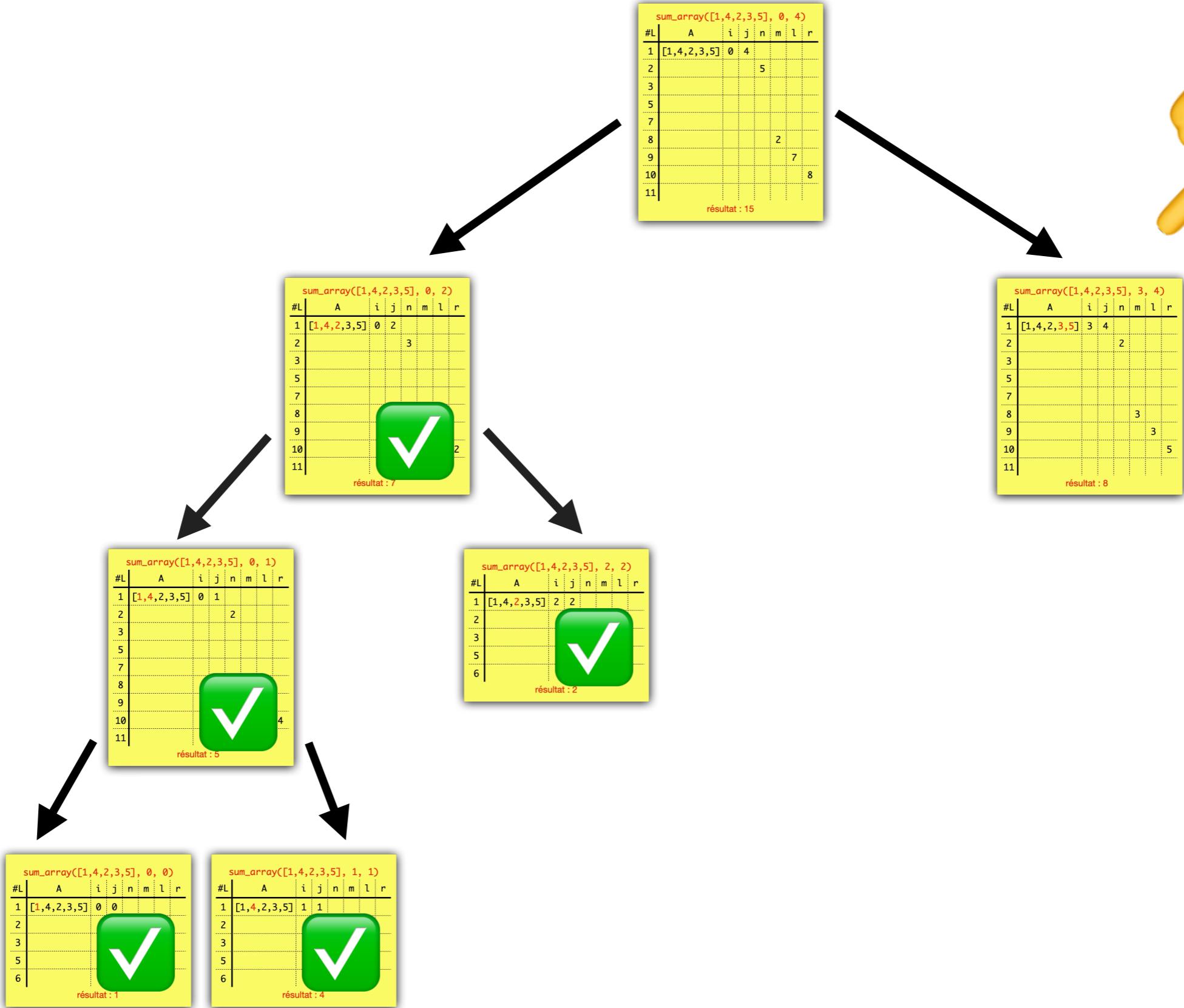
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
6							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
6							

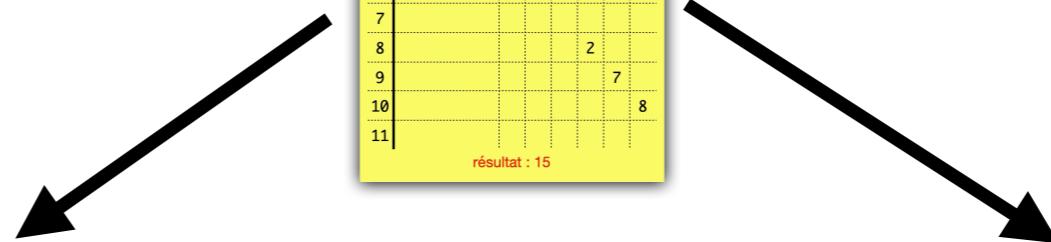
résultat : 15





#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
5							
7							
8				2			
9					7		
10						8	
11							

résultat : 15



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
5							
7							
8							
9							
10				2			
11							

résultat : 7



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
5							
7							
8				3			
9					3		
10						5	
11							

résultat : 8



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
5							
7							
8							
9							
10				4			
11							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2				2			
3							
5							
6							

résultat : 2

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2				1			
3							
5							
6							

résultat : 3

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2				0			
3							
5							
6							

résultat : 1

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2				1			
3							
5							
6							

résultat : 4

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2				5			
3							
4							
5							
6							
7							
8					2		
9						7	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2				3			
3							
4							
5							
6							
7							
8							
9							
10					2		
11							

résultat : 7

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2				2			
3							
4							
5							
6					3		
7						3	
8							5
9							
10							
11							

résultat : 8

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2				2			
3							
4							
5							
6							
7							
8							
9							
10					4		
11							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2				2			
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 2

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2				3			
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 3

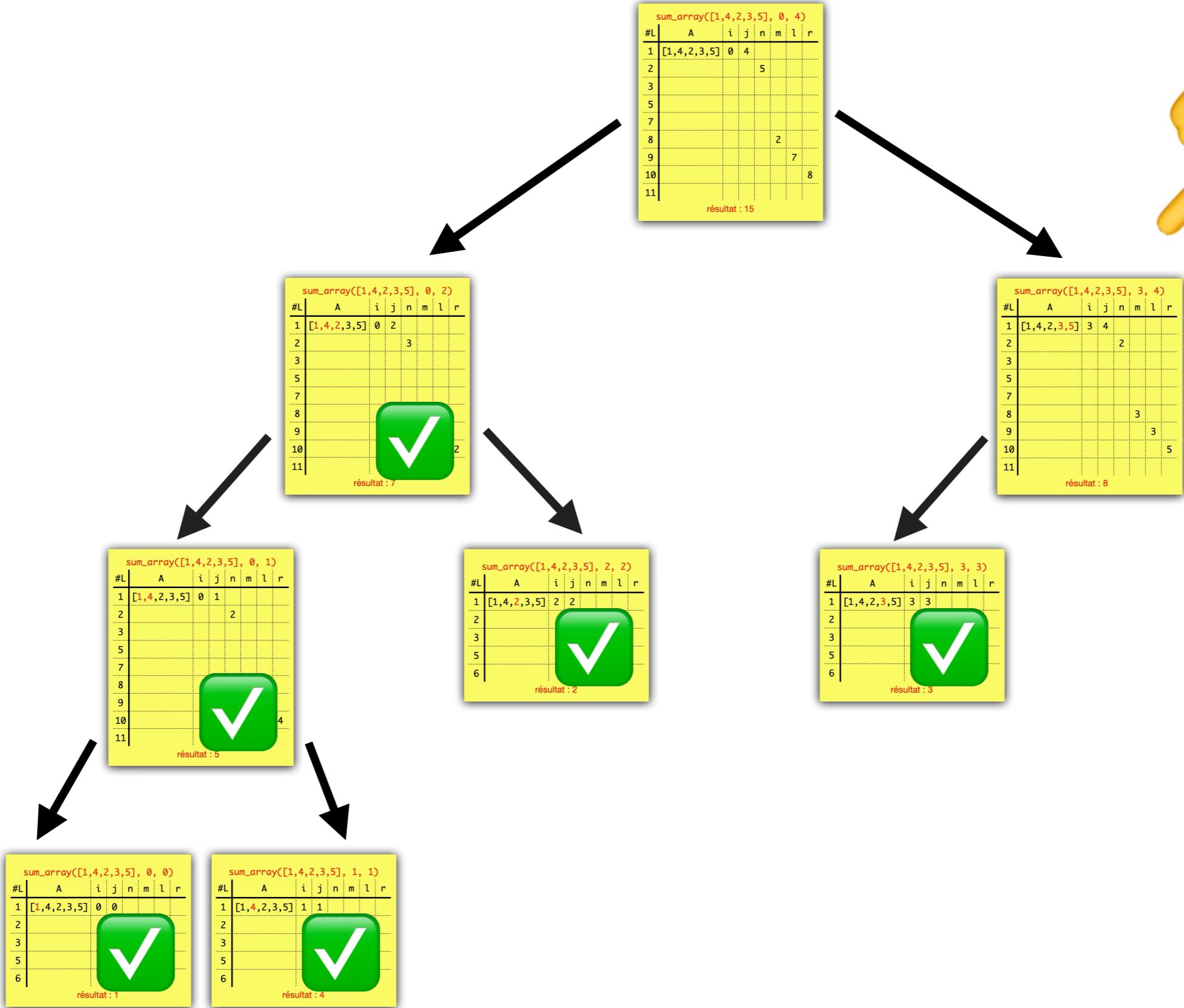
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2				0			
3							
4							
5							
6							
7							
8							
9							
10							
11							

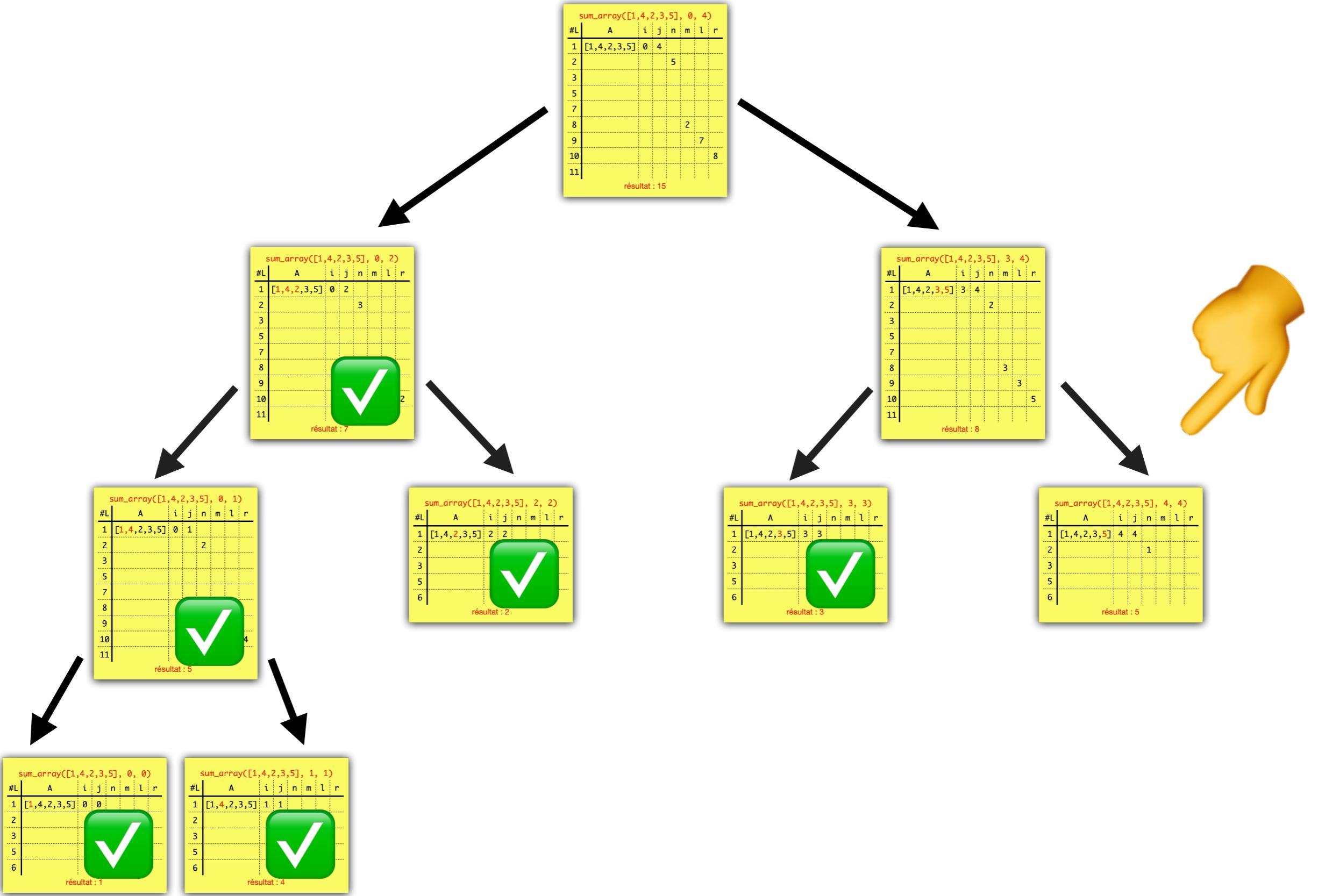
résultat : 1

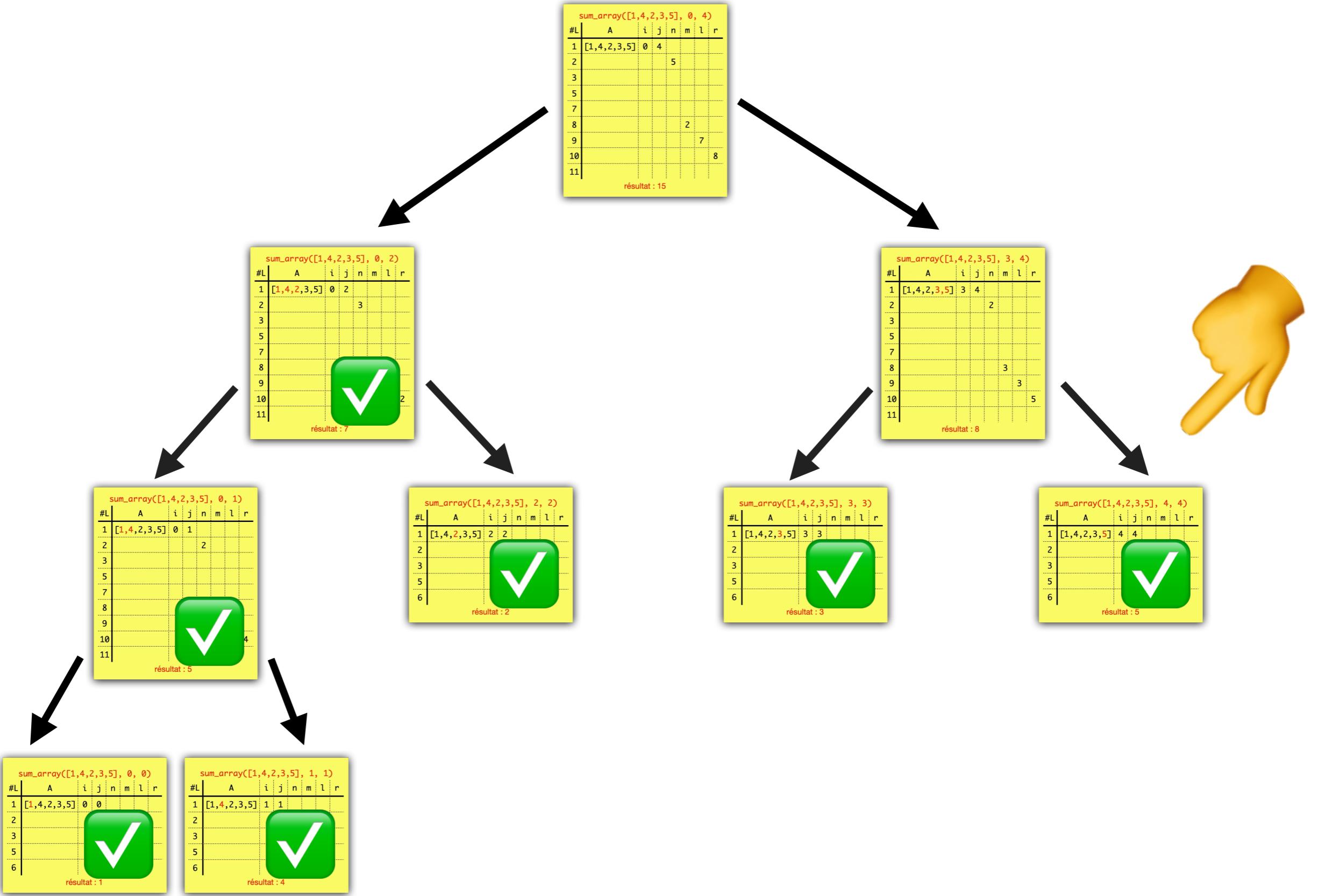
#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2				1			
3							
4							
5							
6							
7							
8							
9							
10							
11							

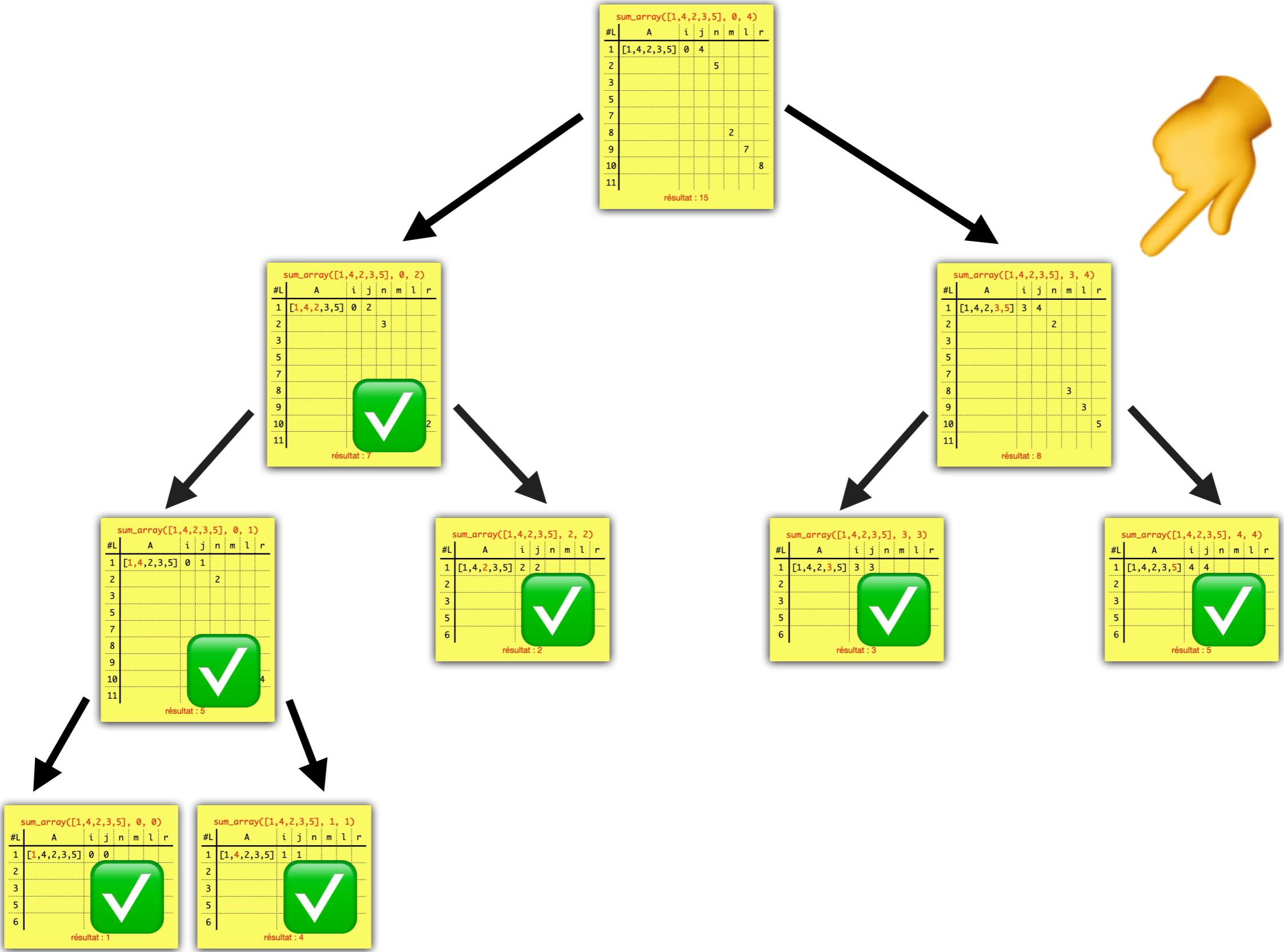
résultat : 4

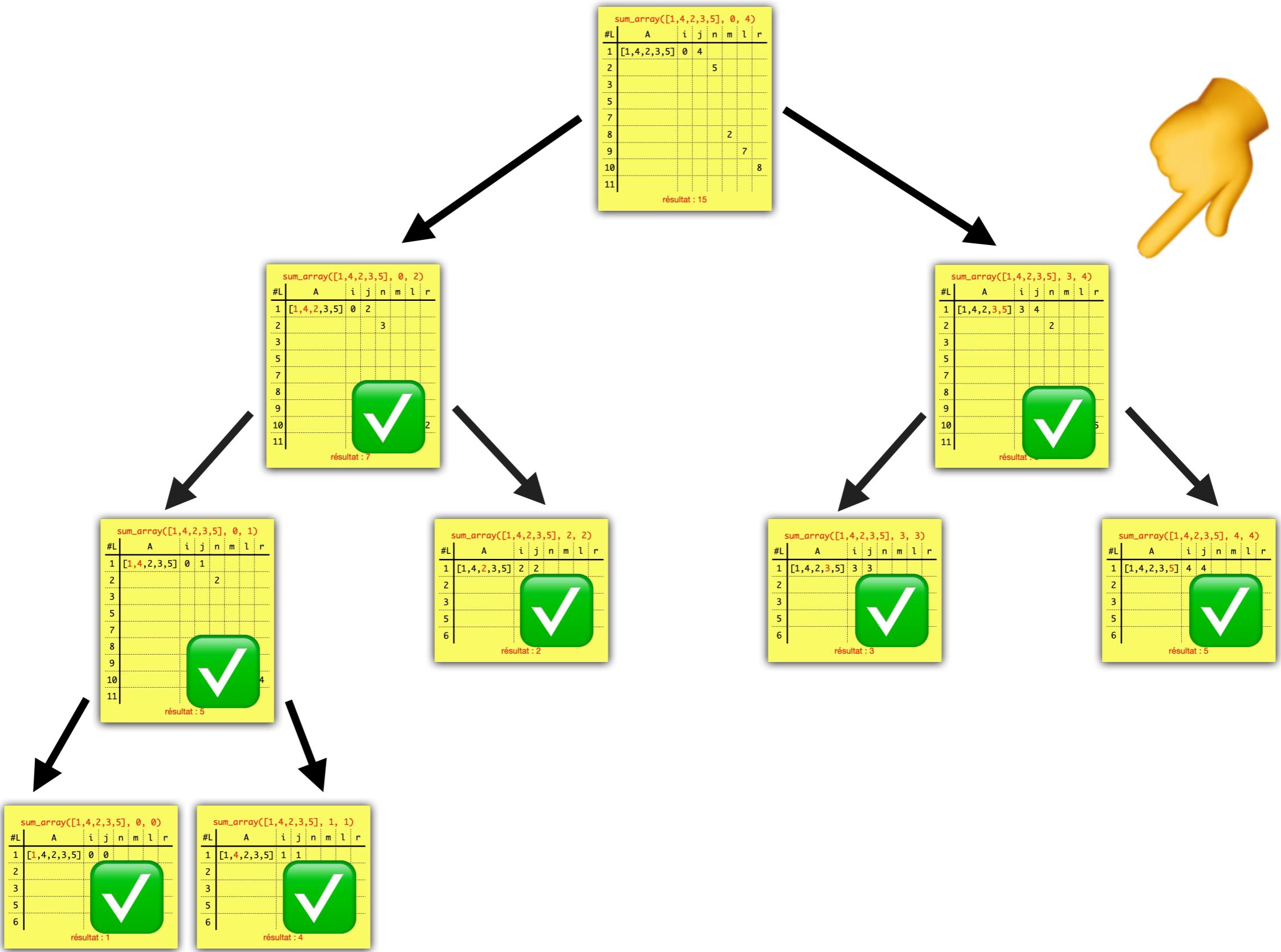














#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7				2			
8					7		
9						8	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7							
8						2	
9							
10							
11							

résultat : 7

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2						2	
3							
5							
7							
8						5	
9							
10							
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5							
7							
8						4	
9							
10							
11							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 2

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 3

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2							
3							
5							
7							
8							
9							
10							
11							

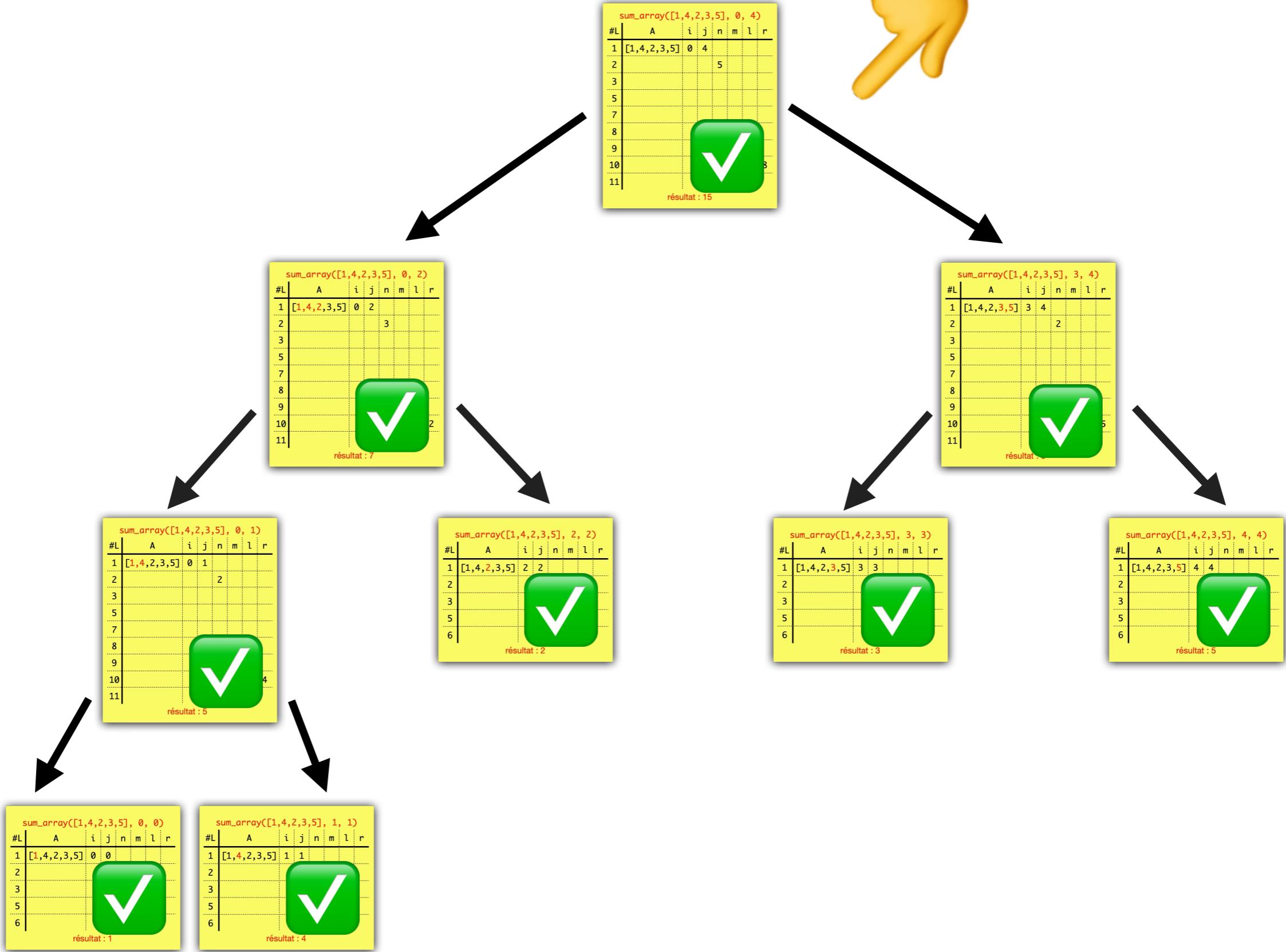
résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 1

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2							
3							
5							
7							
8							
9							
10							
11							

résultat : 4



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 15



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 7



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 15



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 5



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2			3				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 2



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2			4				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 3



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2			5				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 5



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2			1				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 1



#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2			2				
3							
4							
5							
6							
7							
8							
9							
10							
11							

résultat : 4



Comptage des opérations

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7				2			
8					7		
9						8	
10							8
11							

résultat : 15

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7				1			
8					5		
9						2	
10							2
11							

résultat : 7

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	4				
2						2	
3							
5							
7						3	
8							3
9							3
10							5
11							

résultat : 8

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5				0			
7					1		
8						4	
9							4
10							
11							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2			1				
3							
5							
6							

résultat : 2

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2						1	
3							
5							
6							

résultat : 3

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2						1	
3							
5							
6							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2			1				
3							
5							
6							

résultat : 1

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2				1			
3							
5							
6							

résultat : 4

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7				2			
8					7		
9						8	
10							8
11							

résultat : 15

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7				1			
8					5		
9						2	
10							2
11							

résultat : 7

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5							
7				0			
8					1		
9						4	
10							4
11							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2			1				
3							
5							
6							

résultat : 2

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2			1				
3							
5							
6							

résultat : 3

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2			1				
3							
5							
6							

résultat : 5

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2			1				
3							
5							
6							

résultat : 1

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2			1				
3							
5							
6							

résultat : 4

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7				2			
8					7		
9						8	
10							8
11							

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7				1			
8					5		
9						2	
10							2
11							

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5							
7				0			
8					1		
9						4	
10							4
11							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2			1				
3							
5							
6							

5 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	3	3				
2					1		
3							
5							
6							

5 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	4	4				
2					1		
3							
5							
6							

5 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2			1				
3							
5							
6							

5 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2				1			
3							
5							
6							

5 op

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	4				
2			5				
3							
5							
7				2			
8					7		
9						8	
10							8
11							

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	2				
2			3				
3							
5							
7				1			
8					5		
9						2	
10							2
11							

9 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	1				
2			2				
3							
5							
7				0			
8					1		
9						4	
10							4
11							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	2	2				
2			1				
3							
5							
6							

5 op

5 op

5 op

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	0	0				
2			1				
3							
5							
6							

#L	A	i	j	n	m	l	r
1	[1,4,2,3,5]	1	1				
2			1				
3							
5							
6							

5 op

5 op

61 op

Terminaison de sum_array

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
    return l + r
```

Terminaison de sum_array

si $n = 0$ ou $n = 1$,
alors on s'arrête

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
        return l + r
```

Terminaison de sum_array

si $n = 0$ ou $n = 1$,
alors on s'arrête

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
        return l + r
```

sinon on a deux
appels récursifs...

Terminaison de sum_array

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
        return l + r
```

si $n = 0$ ou $n = 1$,
alors on s'arrête

sinon on a deux
appels récursifs...

...mais avec moins
d'éléments (on se
rapproche de 1)

Terminaison de sum_array

```
def sum_array(A, i, j):
    n = j - i + 1
    if n == 0:
        return 0
    elif n == 1:
        return A[i]
    else:
        m = (i + j) // 2
        l = sum_array(A, i, m)
        r = sum_array(A, m+1, j)
        return l + r
```

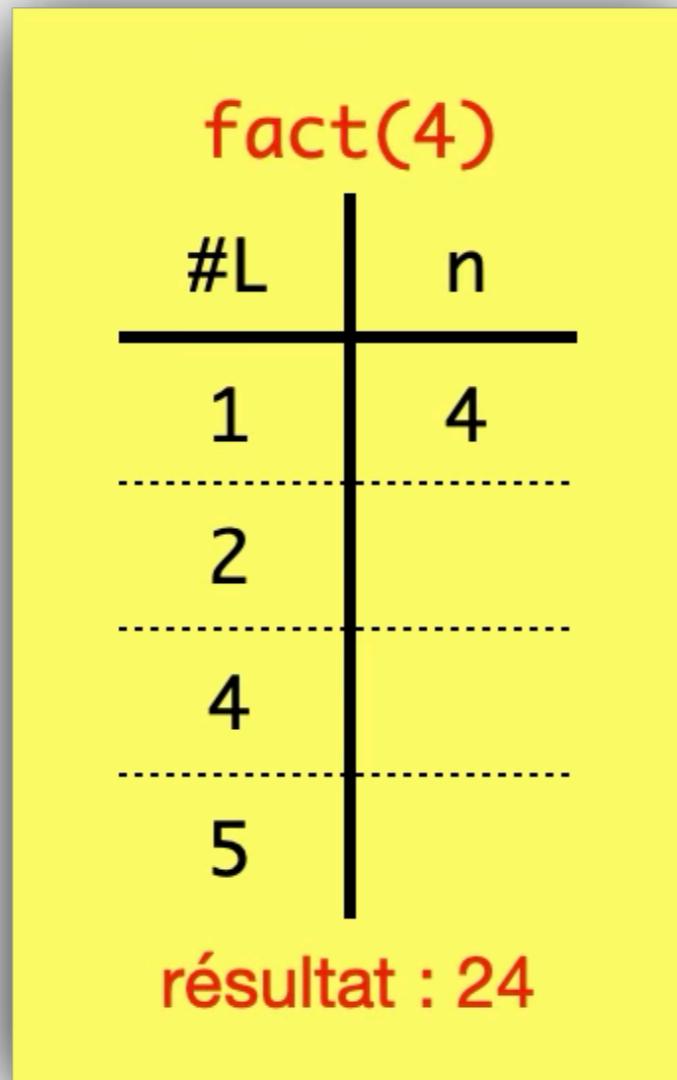
si $n = 0$ ou $n = 1$,
alors on s'arrête

sinon on a deux
appels récursifs...

...mais avec moins
d'éléments (on se
rapproche de 1)

et après on
s'arrête

Arbre de récursion de fact(4)



fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24



fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(4)	
#L	n
1	4
2	
4	
5	
résultat : 24	



fact(3)	
#L	n
1	3
2	
4	
5	
résultat : 6	



fact(2)	
#L	n
1	2
2	
4	
5	
résultat : 2	

fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)

#L	n
1	2
2	
4	
5	

résultat : 2



fact(1)

#L	n
1	1
2	
4	
5	

résultat : 1

fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)

#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)

#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)

#L	n
1	0
2	
3	

résultat : 1



fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)

#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)

#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)

#L	n
1	0
2	
3	

résultat : 



fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)

#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)

#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)

#L	n
1	0
2	
3	

résultat : ✓



fact(4)

#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)

#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)

#L	n
1	2
2	
4	
5	

résultat : 2



fact(1)

#L	n
1	1
2	
4	
5	

résultat : ✓

fact(0)

#L	n
1	0
2	
3	

résultat : ✓



fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : 1

fact(0)	
#L	n
1	0
2	
3	

résultat : 1



fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2



fact(1)	
#L	n
1	1
2	
4	
5	

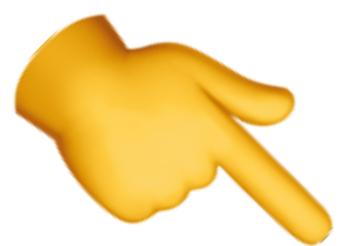
résultat : 1



fact(0)	
#L	n
1	0
2	
3	

résultat : 1





fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

↓

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

↓

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2



↓

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : 1

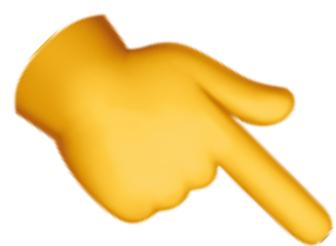


↓

fact(0)	
#L	n
1	0
2	
3	

résultat : 1





fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

↓

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : 6

↓

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : 2

↓

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : 1

↓

fact(0)	
#L	n
1	0
2	
3	

résultat : 1



fact(4)	
#L	n
1	4
2	
4	
5	

résultat : 24

↓

fact(3)	
#L	n
1	3
2	
4	
5	

résultat : ✓

↓

fact(2)	
#L	n
1	2
2	
4	
5	

résultat : ✓

↓

fact(1)	
#L	n
1	1
2	
4	
5	

résultat : ✓

↓

fact(0)	
#L	n
1	0
2	
3	

résultat : ✓



fact(4)	
#L	n
1	4
2	
4	
5	
résultat	

fact(3)	
#L	n
1	3
2	
4	
5	
résultat	

fact(2)	
#L	n
1	2
2	
4	
5	
résultat	

fact(1)	
#L	n
1	1
2	
4	
5	
résultat	

fact(0)	
#L	n
1	0
2	
3	
résultat	

fact(4)	
#L	n
1	4
2	
4	
5	
résultat	

↓

fact(3)	
#L	n
1	3
2	
4	
5	
résultat	

↓

fact(2)	
#L	n
1	2
2	
4	
5	
résultat	

↓

fact(1)	
#L	n
1	1
2	
4	
5	
résultat	

↓

fact(0)	
#L	n
1	0
2	
3	
résultat	

Exercices 2 et 3

du TD6