



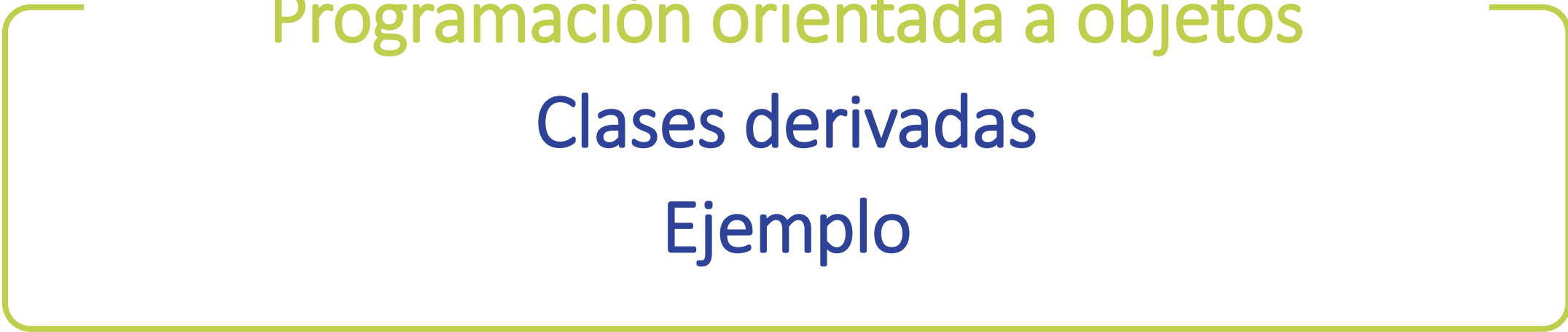
## CICLO 01

[FORMACIÓN POR CICLOS]

# Fundamentos de Programación

Ejercicios con clase derivada



A large, light green bracket graphic that frames the text on the slide.

# Programación orientada a objetos

## Clases derivadas

### Ejemplo

Presentamos el algoritmo para sumar dos enteros representados en *altaPrecision*

Es un algoritmo cuya estructura es la misma del algoritmo de intercalación de dos vectores ordenados:

Recorremos ambos vectores simultáneamente de derecha a izquierda.

Sumamos los datos en esas posiciones y al resultado le separamos el último dígito.

Lo almacenamos en la posición correspondiente del resultado y calculamos el acarreo para incluirlo en la suma de las dos siguientes posiciones.

El método para la suma podemos definirlo de dos formas:

- un método denominado sumar (Versión 1)

- sobrecargando el operador de suma (Versión 2)

Adicionalmente, se trabaja una variable global, la cual llamamos *acarreo*.

Una variable se define global cuando se necesita que sea compartida por más de un subprograma.

Dichas variables se deben definir como global en todos los subprogramas donde se compartan.

## Versión 1

```
def sumar(self, b):
    global acarreo
    i = self.tamaño()
    j = b.tamaño()
    k = mayor(i, j) + 2
    c = altaPrecision(k)
    acarreo = 0
    while i > self.V[0] and j > b.V[0]:
        r = self.sumaYacarreo(self.V[i], b.V[j])
        c.V[k] = r
        i = i - 1
        j = j - 1
        k = k - 1
    while i > self.V[0]:
        r = self.sumaYacarreo(self.V[i])
        c.V[k] = r
        i = i - 1
        k = k - 1
    while j > b.V[0]:
        r = self.sumaYacarreo(b.V[j])
        c.V[k] = r
        j = j - 1
        k = k - 1
    if acarreo > 0:
        c.V[k] = acarreo
        k = k - 1
    c.V[0] = k
    return c
```

## Versión 2

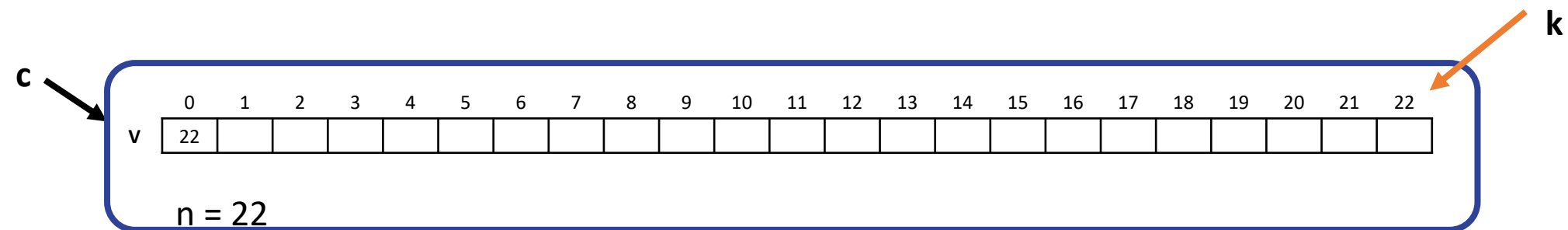
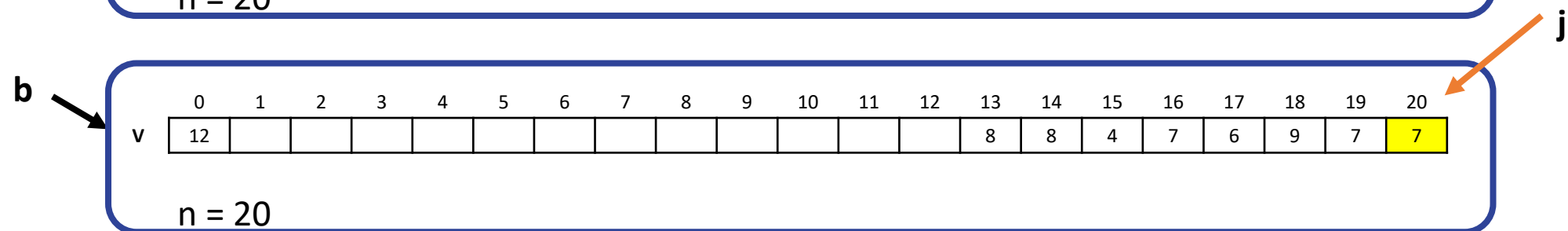
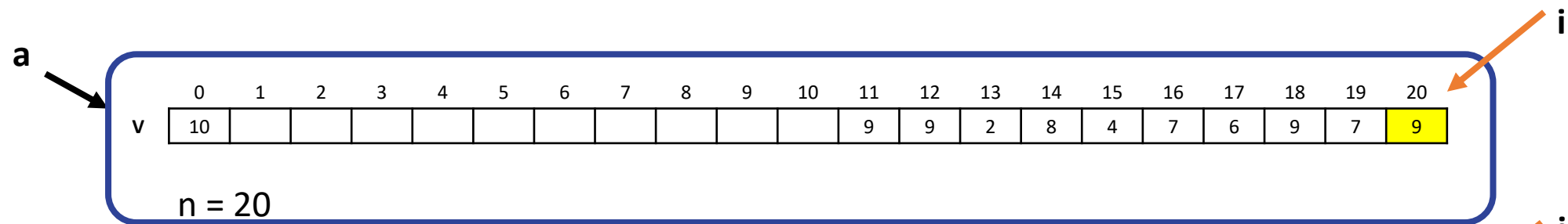
```
def add(self, b):
    global acarreo
    i = self.tamaño()
    j = b.tamaño()
    k = mayor(i, j) + 2
    c = altaPrecision(k)
    acarreo = 0
    while i > self.V[0] and j > b.V[0]:
        r = self.sumaYacarreo(self.V[i], b.V[j])
        c.V[k] = r
        i = i - 1
        j = j - 1
        k = k - 1
    while i > self.V[0]:
        r = self.sumaYacarreo(self.V[i])
        c.V[k] = r
        i = i - 1
        k = k - 1
    while j > b.V[0]:
        r = self.sumaYacarreo(b.V[j])
        c.V[k] = r
        j = j - 1
        k = k - 1
    if acarreo > 0:
        c.V[k] = acarreo
        k = k - 1
    c.V[0] = k
    return c
```

```
def sumaYacarreo(self, a, b=0):
    global acarreo
    s = a + b + acarreo
    if s > 9:
        acarreo = s // 10
        s = s - 10
    else:
        acarreo = 0
    return s
```

Forma de llamado:

Versión 1:  $c = a.sumar(b)$

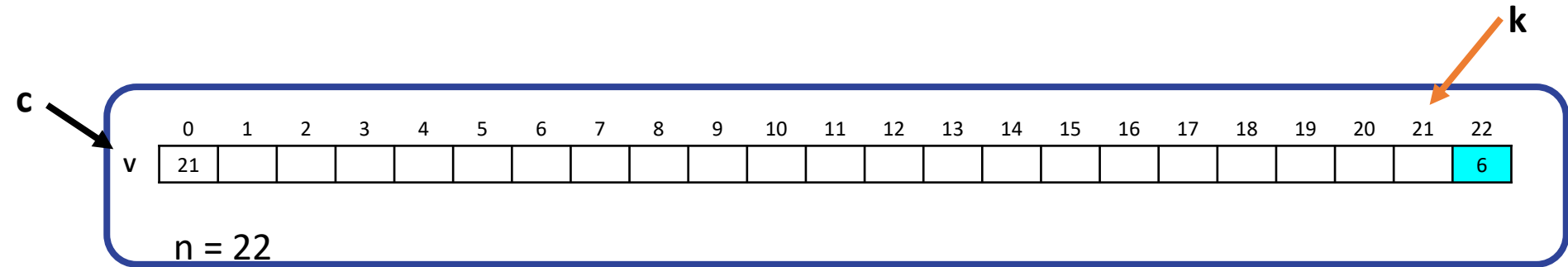
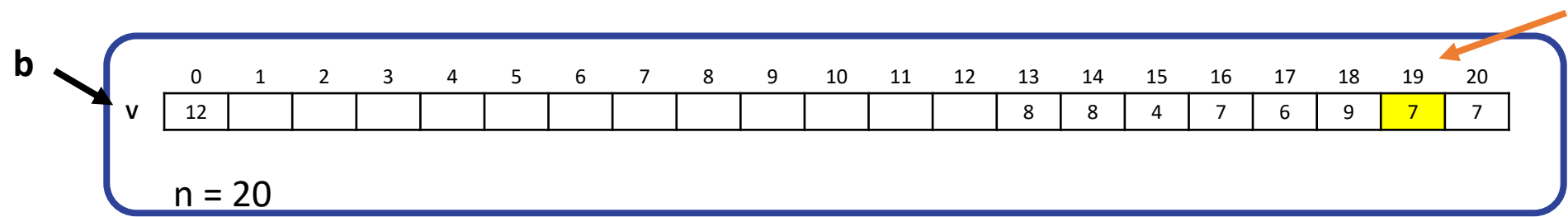
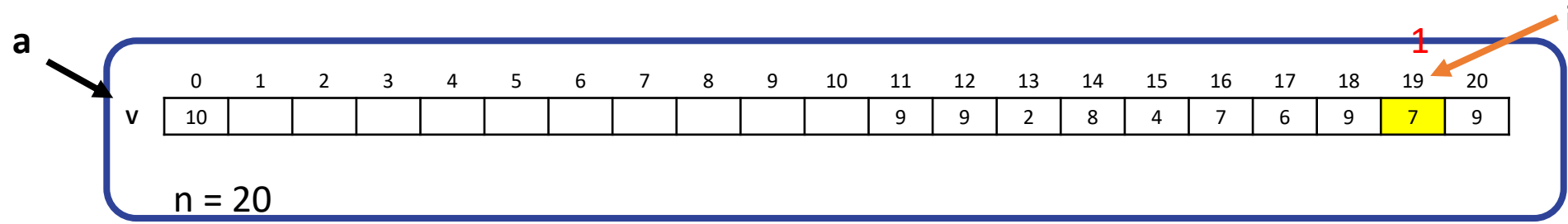
Versión 2:  $c = a + b$



```

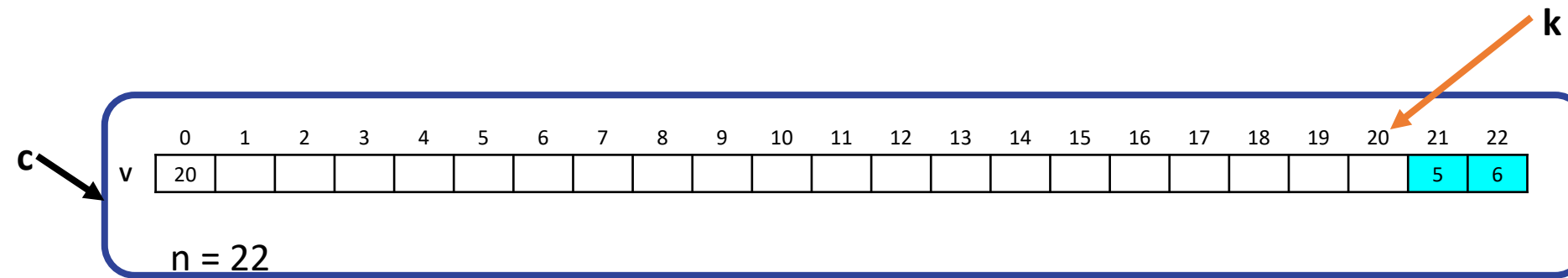
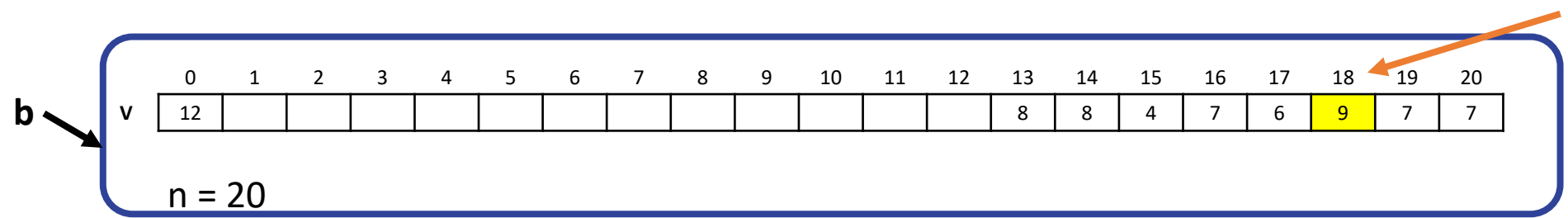
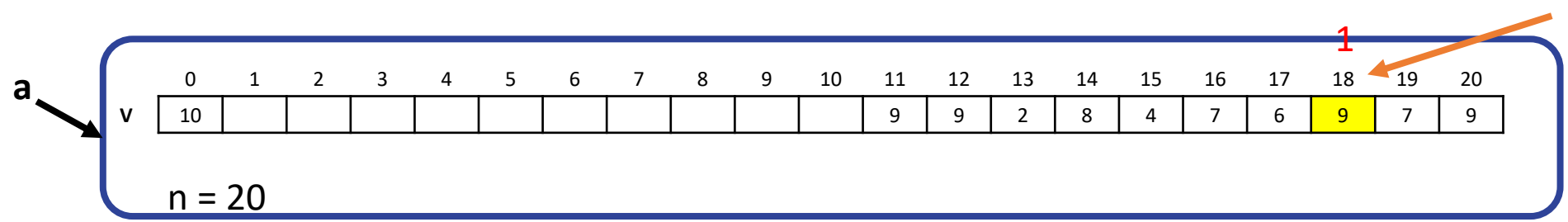
i = self.tamaño()
j = b.tamaño()
k = mayor(i, j) + 2
c = altaPrecision(k)
acarreo = 0

```



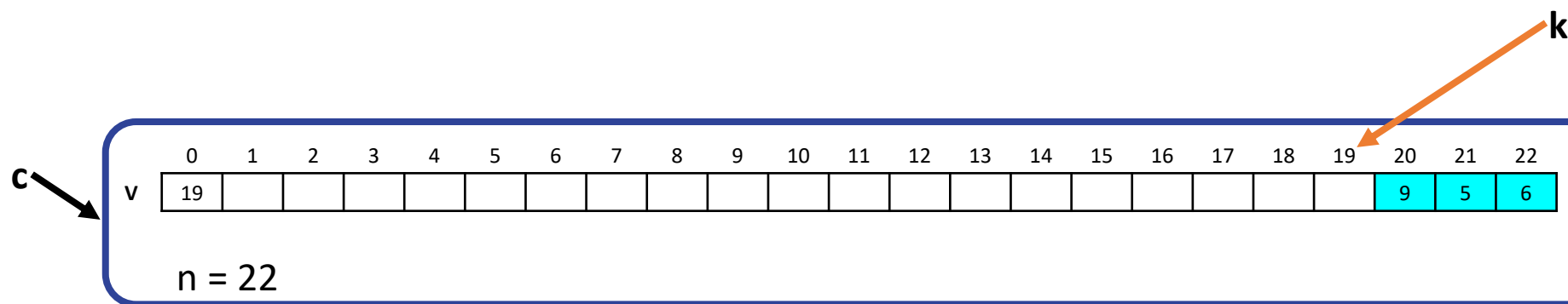
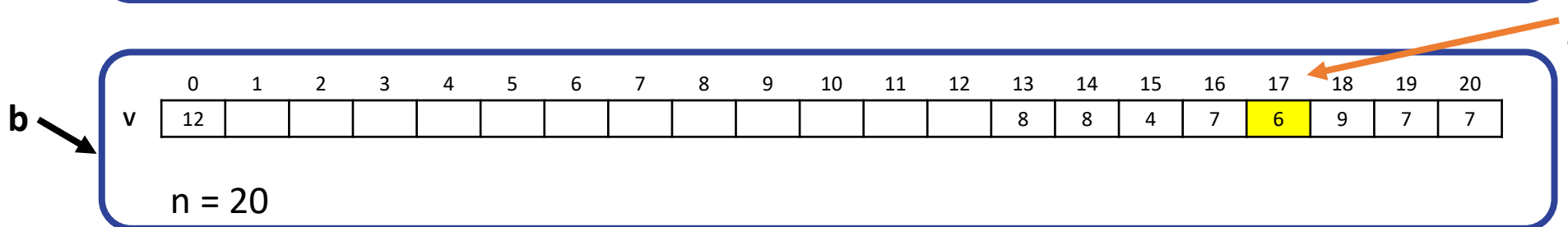
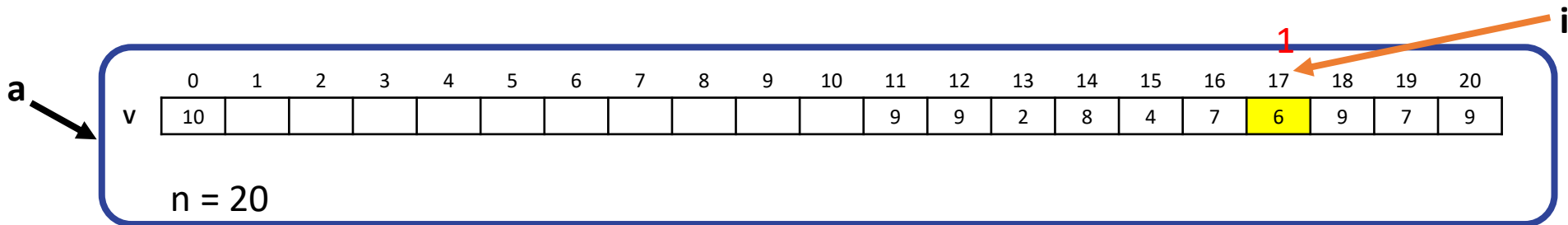
```

while i > self.V[0] and j > b.V[0]:
    r = self.sumaYacarreo(self.V[i], b.V[j])
    c.V[k] = r
    i = i - 1
    j = j - 1
    k = k - 1
  
```



```

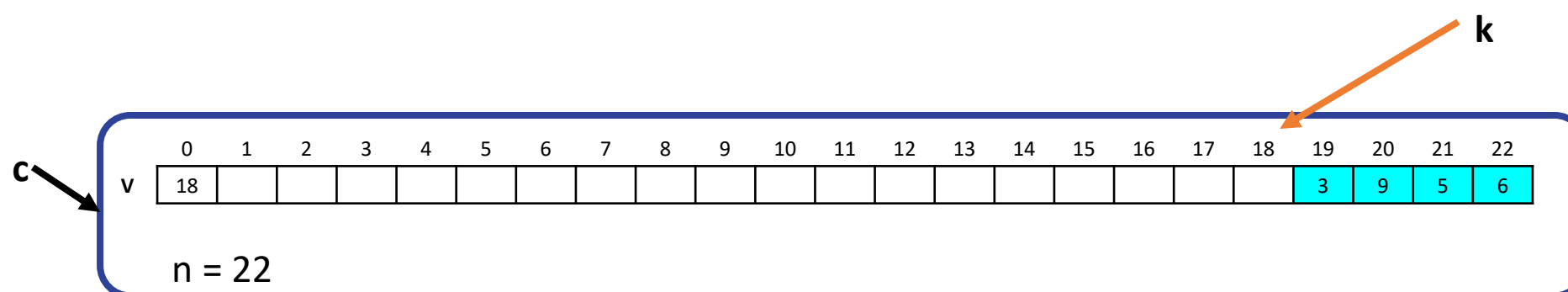
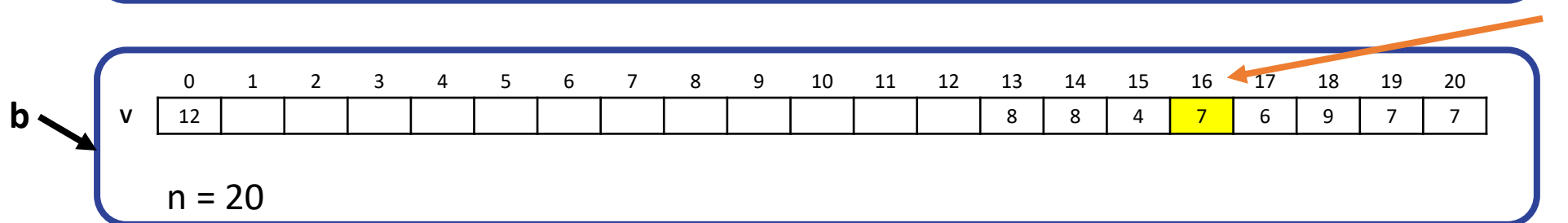
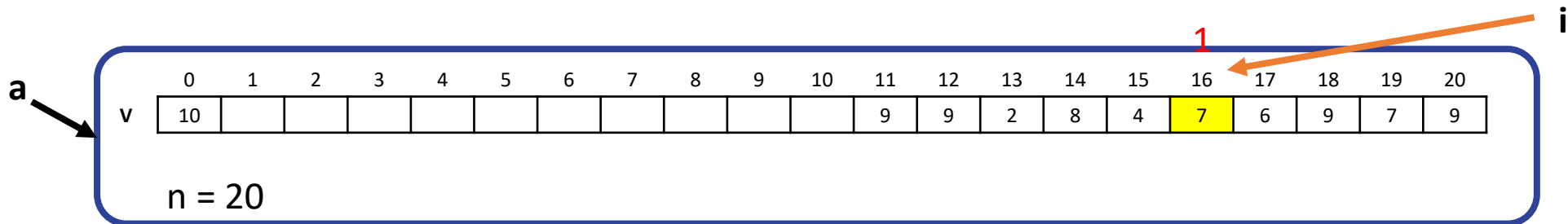
while i > self.V[0] and j > b.V[0]:
    r = self.sumaYacarreo(self.V[i], b.V[j])
    c.V[k] = r
    i = i - 1
    j = j - 1
    k = k - 1
  
```



```

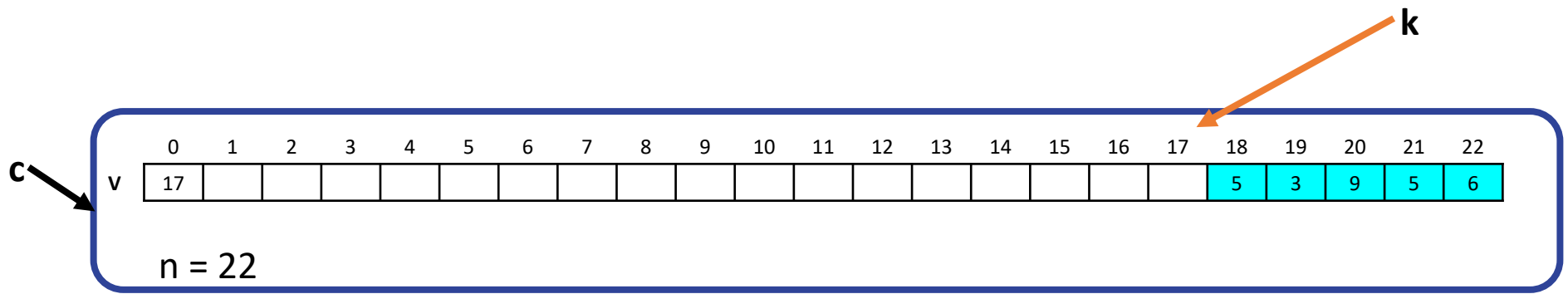
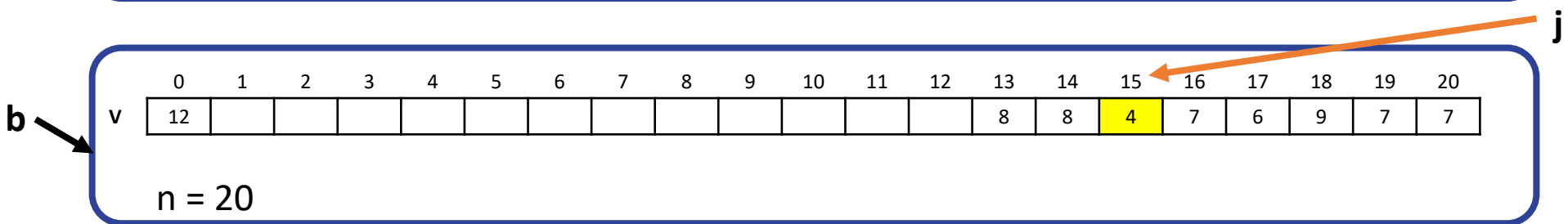
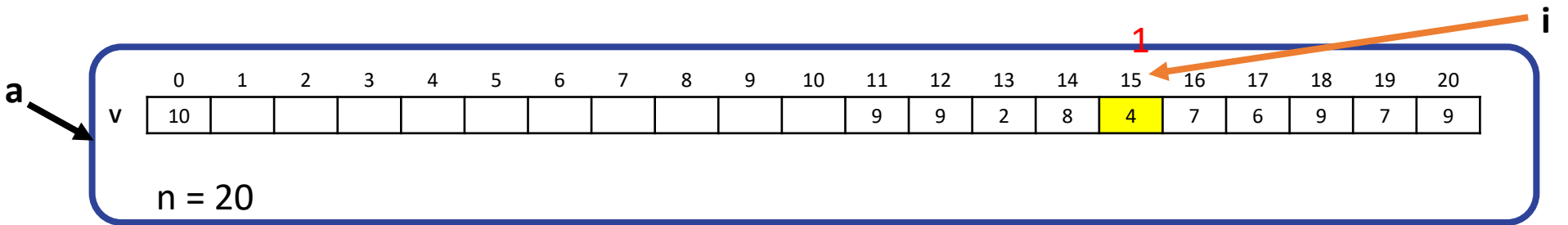
while i > self.V[0] and j > b.V[0]:
    r = self.sumaYacarreo(self.V[i], b.V[j])
    c.V[k] = r
    i = i - 1
    j = j - 1
    k = k - 1
  
```



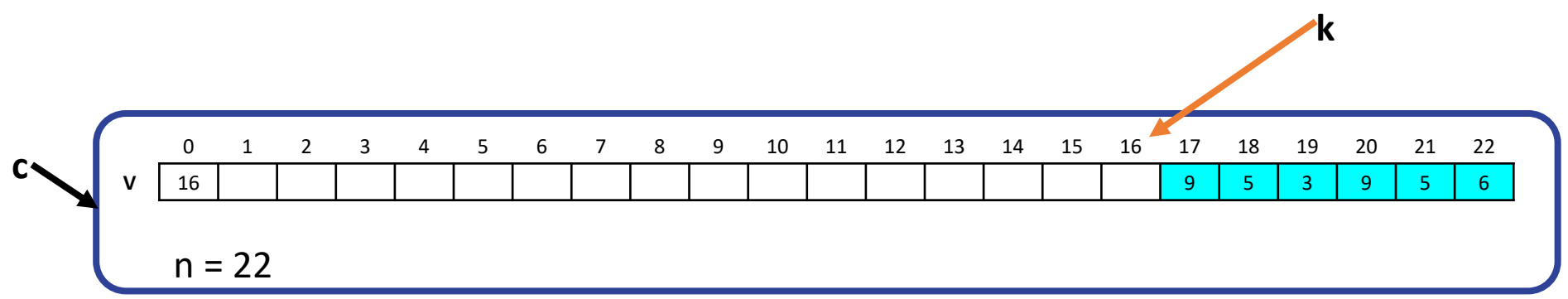
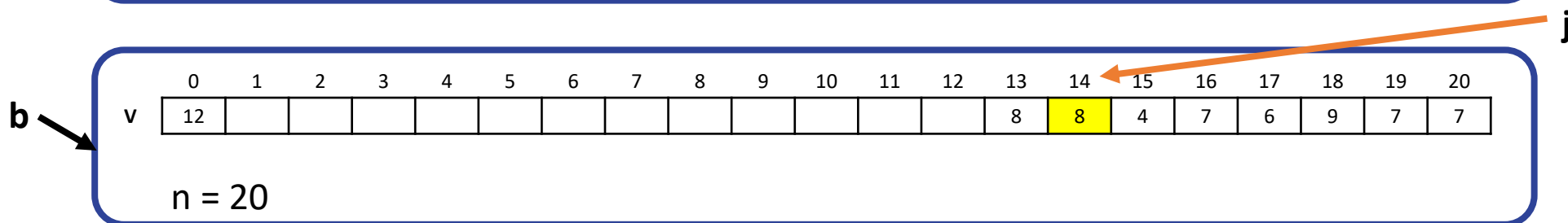
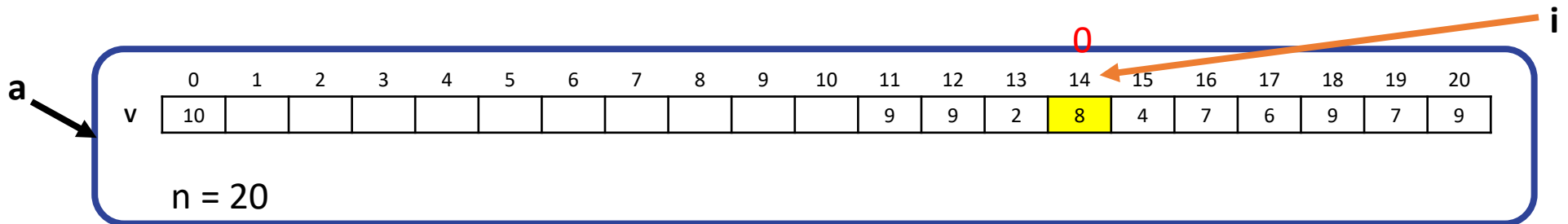


```

while i > self.V[0] and j > b.V[0]:
    r = self.sumaYacarreo(self.V[i], b.V[j])
    c.V[k] = r
    i = i - 1
    j = j - 1
    k = k - 1
  
```

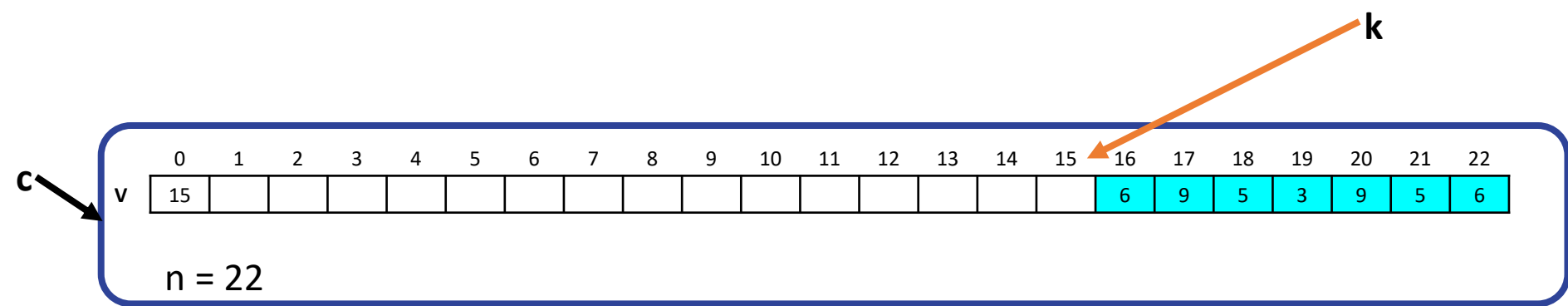
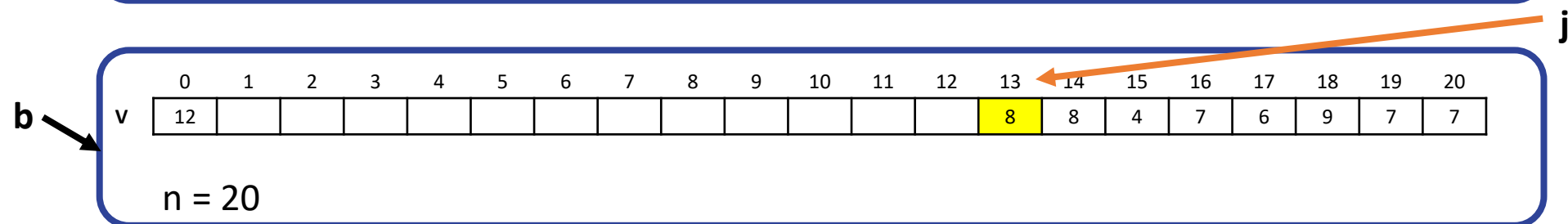
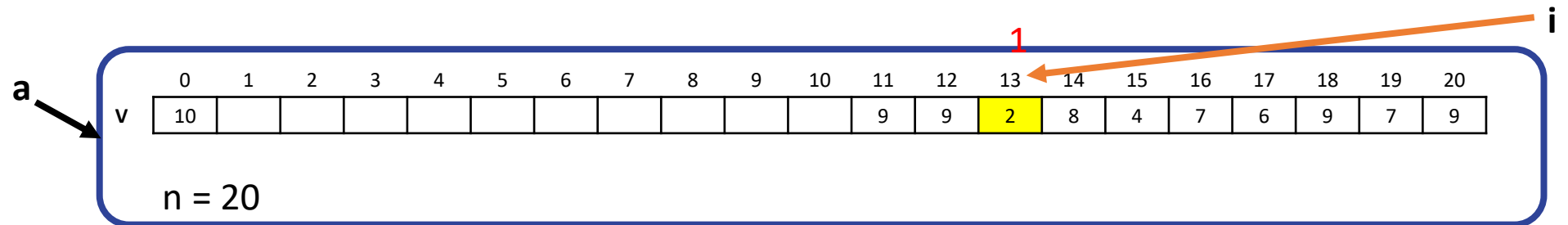


```
while i > self.V[0] and j > b.V[0]:  
    r = self.sumaYacarreo(self.V[i], b.V[j])  
    c.V[k] = r  
    i = i - 1  
    j = j - 1  
    k = k - 1
```

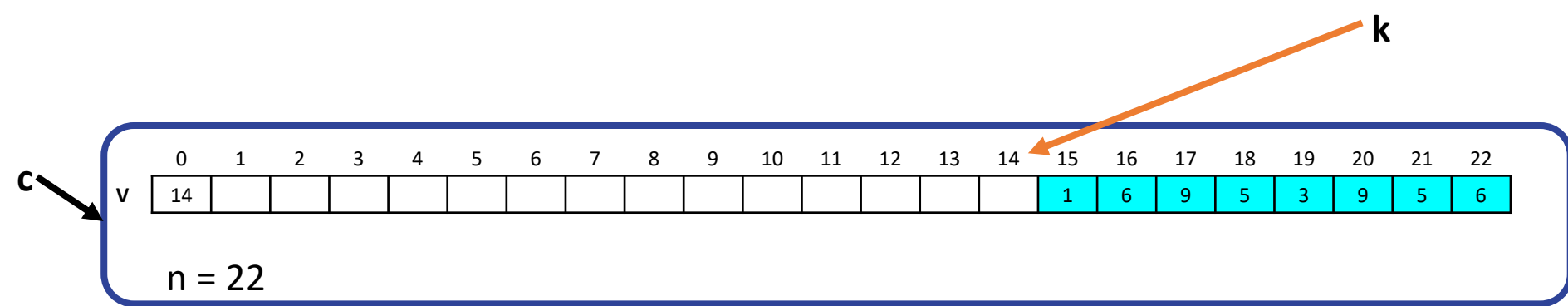
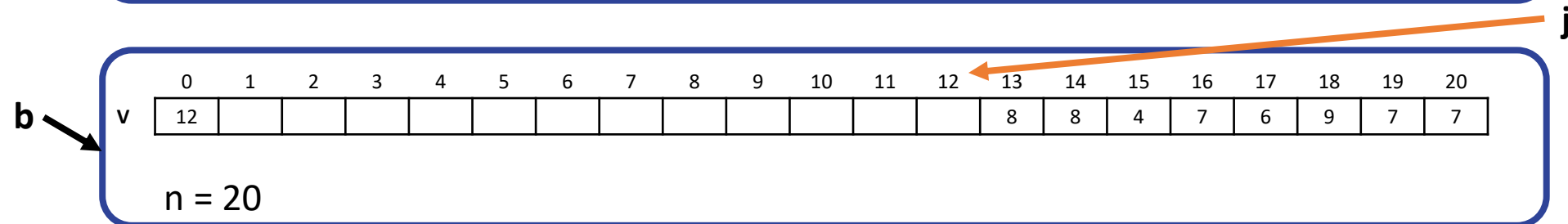
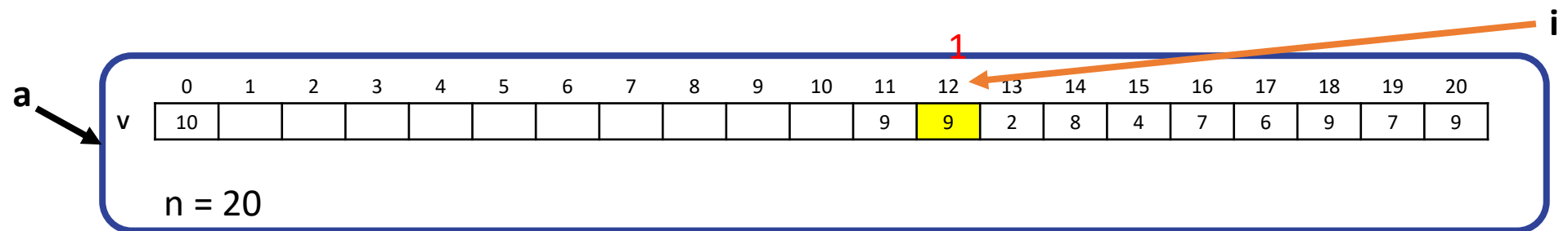


```

while i > self.V[0] and j > b.V[0]:
    r = self.sumaYacarreo(self.V[i], b.V[j])
    c.V[k] = r
    i = i - 1
    j = j - 1
    k = k - 1
  
```

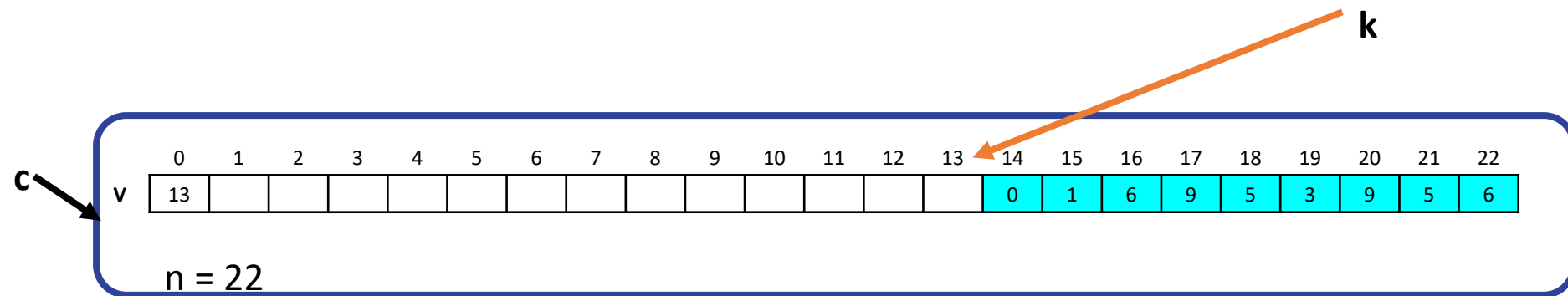
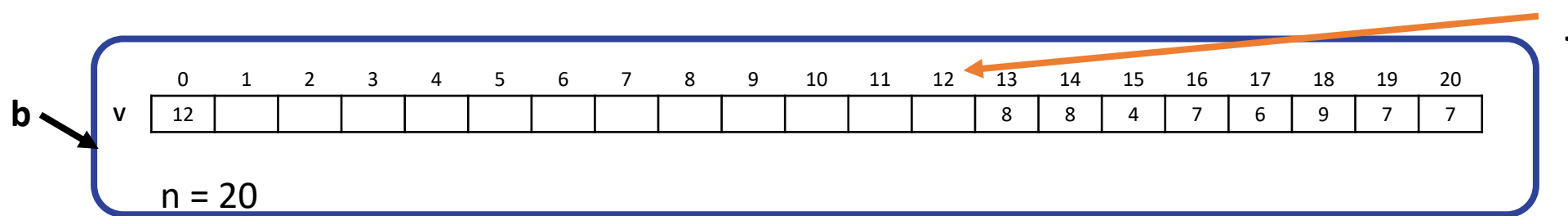
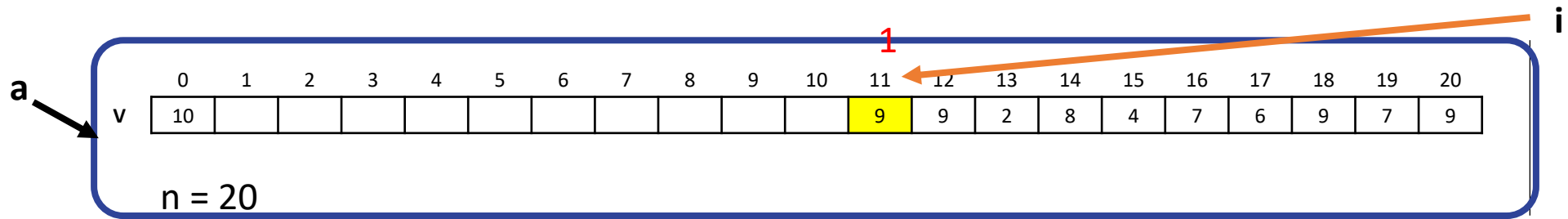


```
while i > self.V[0] and j > b.V[0]:  
    r = self.sumaYacarreo(self.V[i], b.V[j])  
    c.V[k] = r  
    i = i - 1  
    j = j - 1  
    k = k - 1
```



```

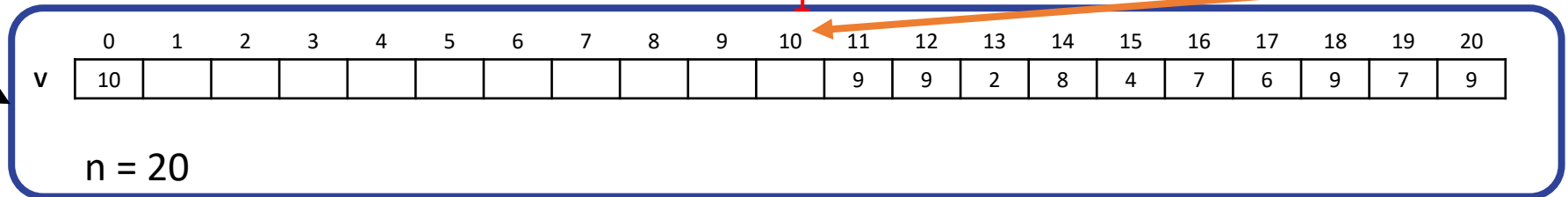
while i > self.V[0]:
    r = self.sumaYacarreo(self.V[i])
    c.V[k] = r
    i = i - 1
    k = k - 1
  
```



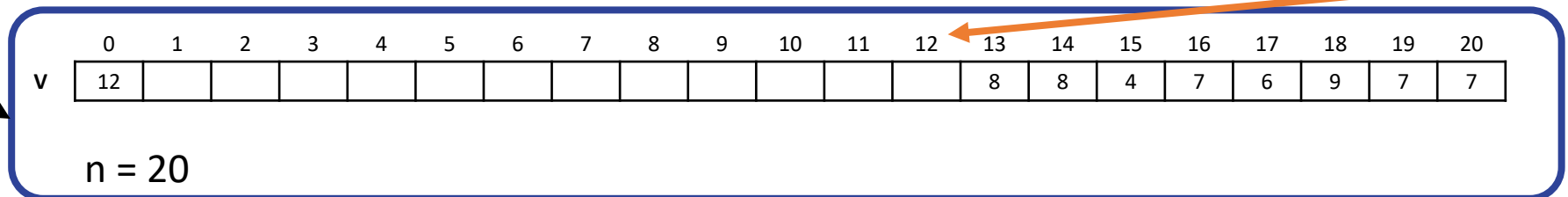
```

while i > self.V[0]:
    r = self.sumaYacarreo(self.V[i])
    c.V[k] = r
    i = i - 1
    k = k - 1
  
```

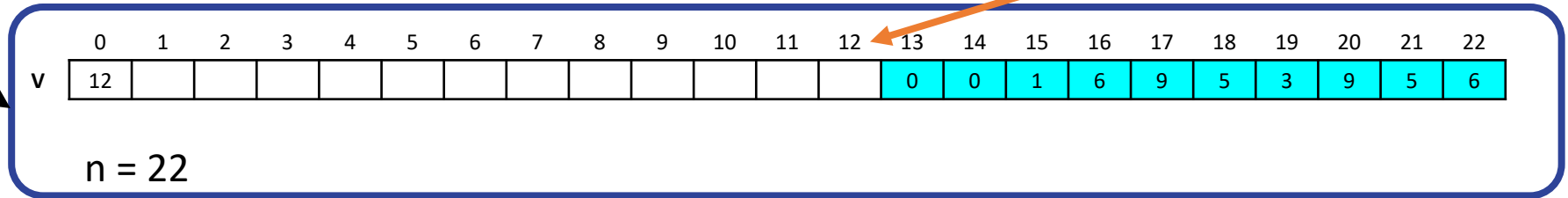
a



b



c



if acarreo > 0:  
 $c.V[k] = \text{acarreo}$   
 $k = k - 1$

a

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	10											9	9	2	8	4	7	6	9	7	9

n = 20

b

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
v	12													8	8	4	7	6	9	7	7

n = 20

c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
v	11												1	0	0	1	6	9	5	3	9	5	6

n = 22

c.V[0] = k  
return c