



UNRN

Universidad Nacional
de Río Negro



| unrionegro

Estructuras avanzadas

Martín René Vilugron
martinvilu@unrn.edu.ar

PROGRAMACIÓN 2
2023

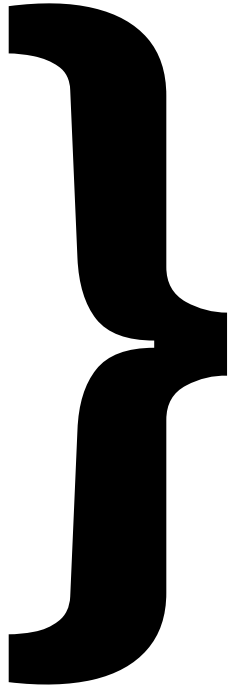
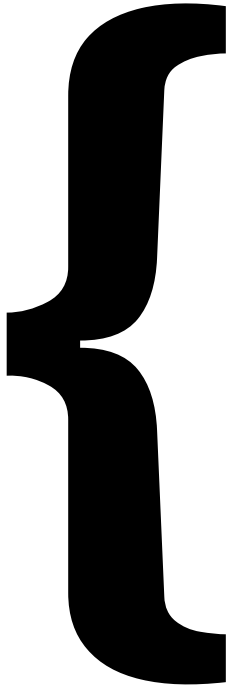
XVIII



parcial
12/06



¿Preguntas?



Repaso general con algunos agregados

Secuencias

dato	dato	dato	dato	dato	dato	dato	dato	dato	dato
0	1	2	3	4	5	6	7	8	9

`largo`

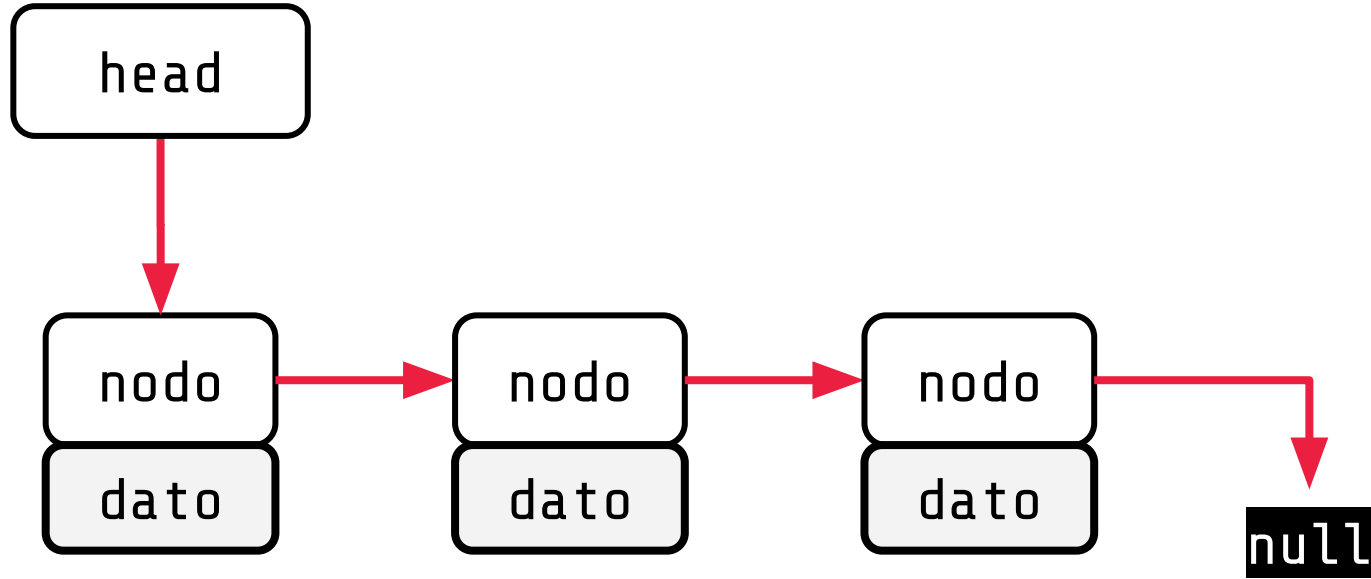


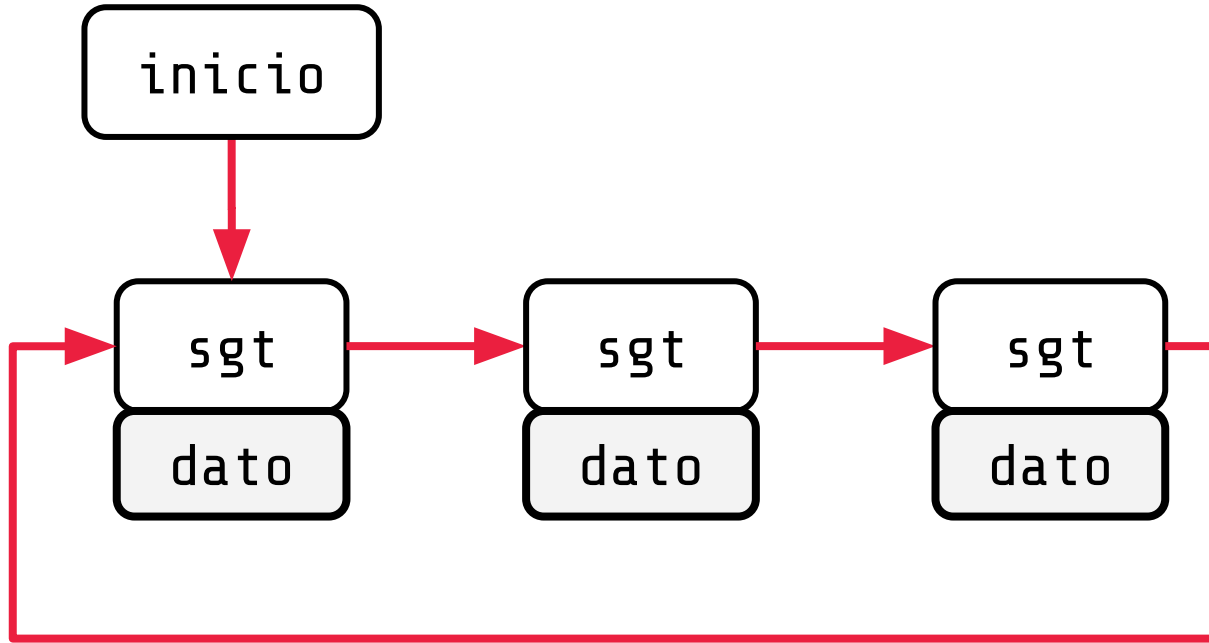
<code>dato</code>	<code>dato</code>	<code>dato</code>	<code>dato</code>	<code>dato</code>	<code>null</code>	<code>null</code>	<code>null</code>	<code>null</code>
0	1	2	3	4	5	6	7	8

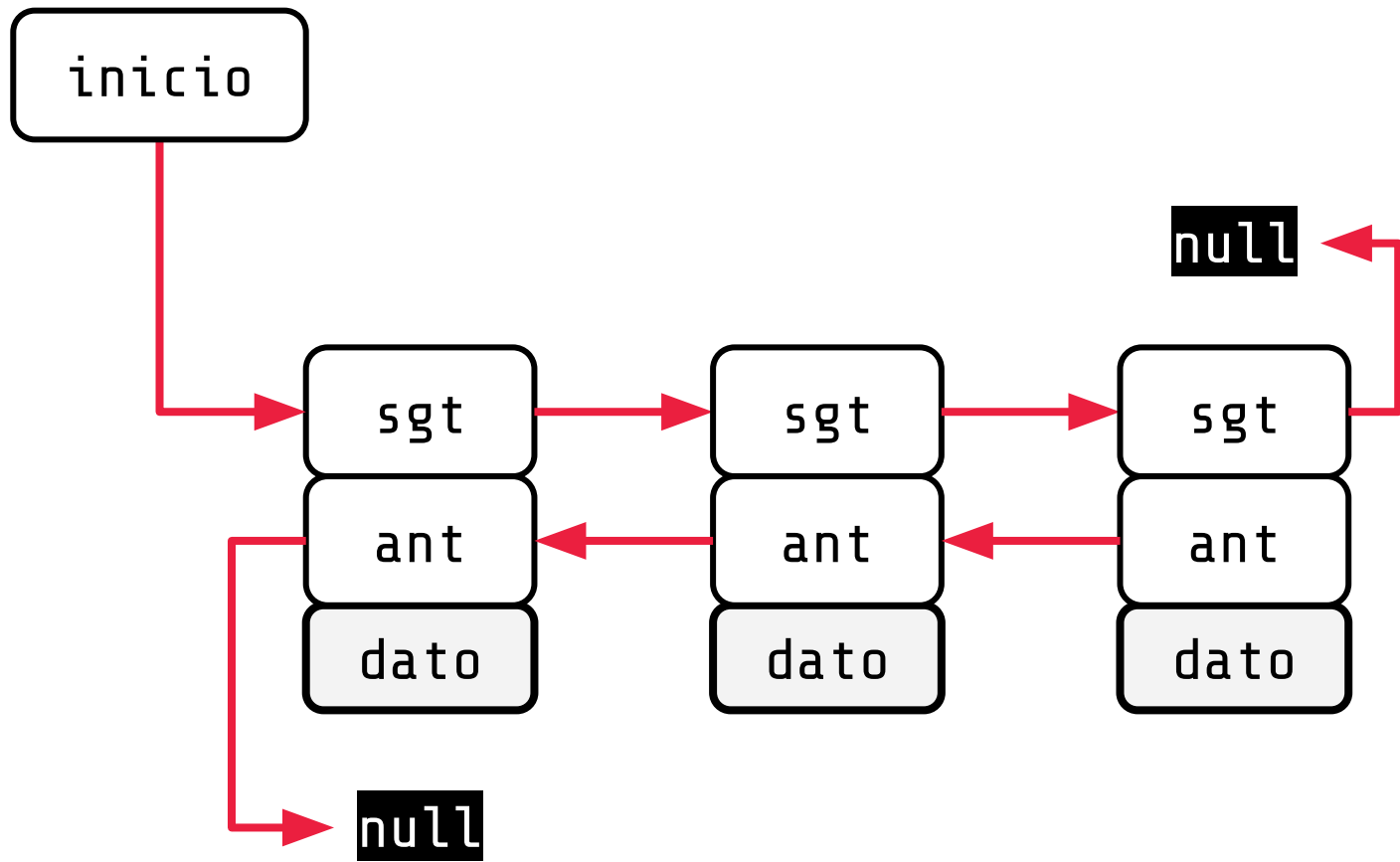
Listas enlazadas

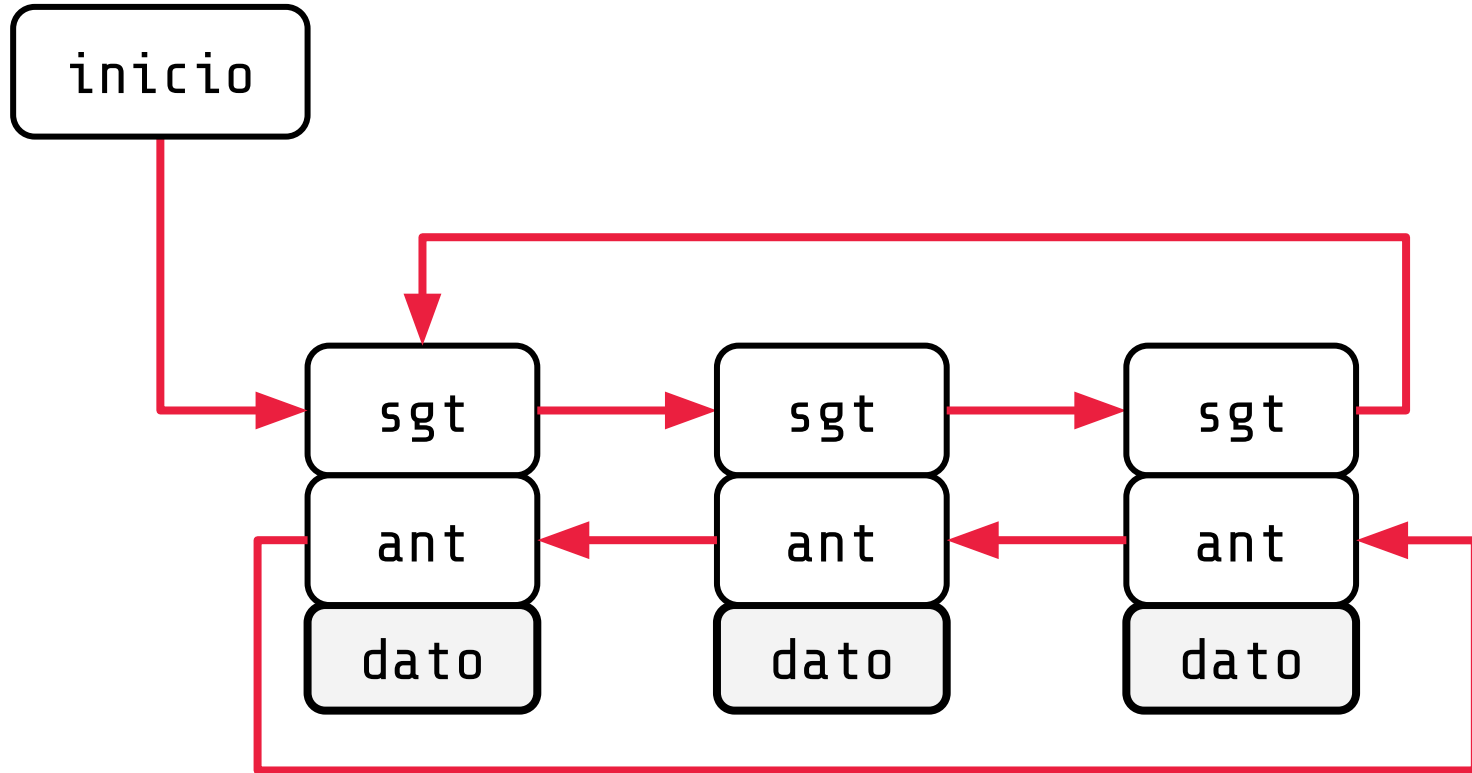
lineales/circulares

simples/dobles





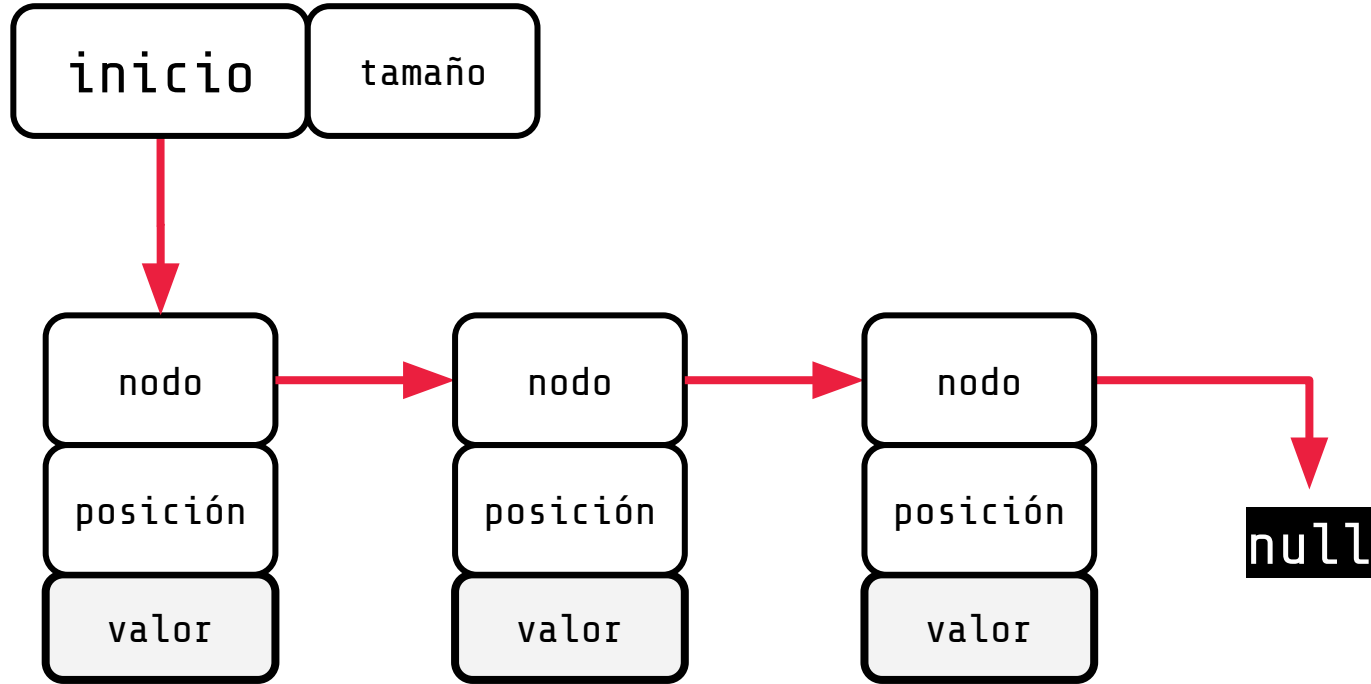




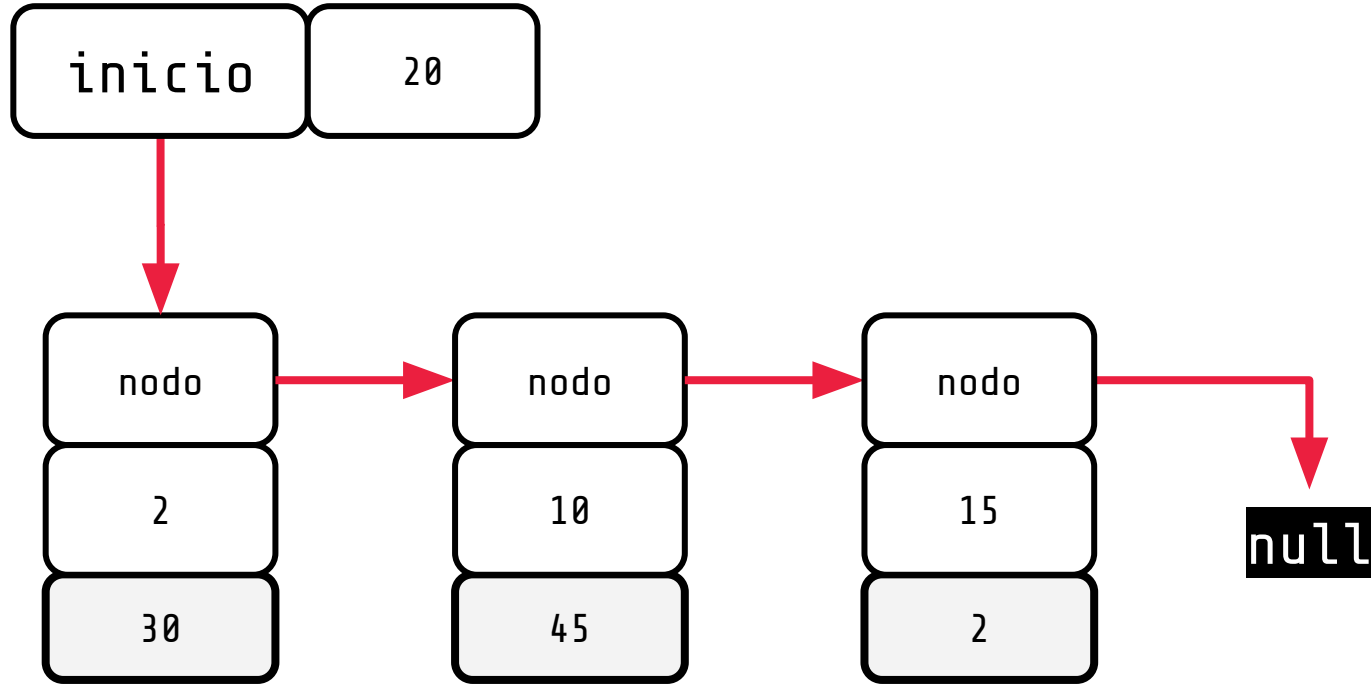


¿Preguntas?

Arrays dispersos (sparse)



Como una forma de ahorrar espacio



Este array que tiene tres valores de interés

Versión densa

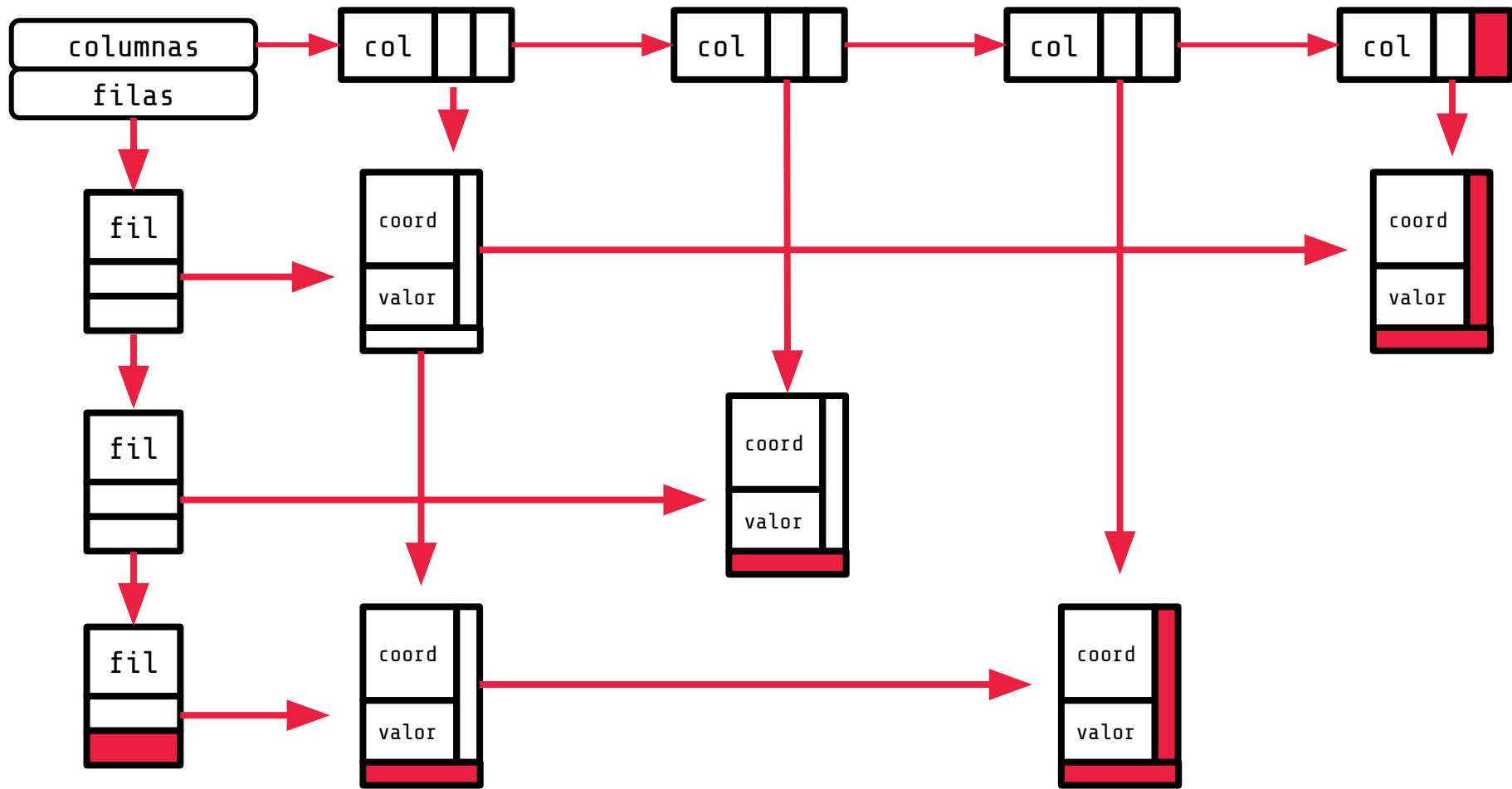
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	30	0	0	0	0	0	0	0	45	0	0	0	0	2	0	0	0	0

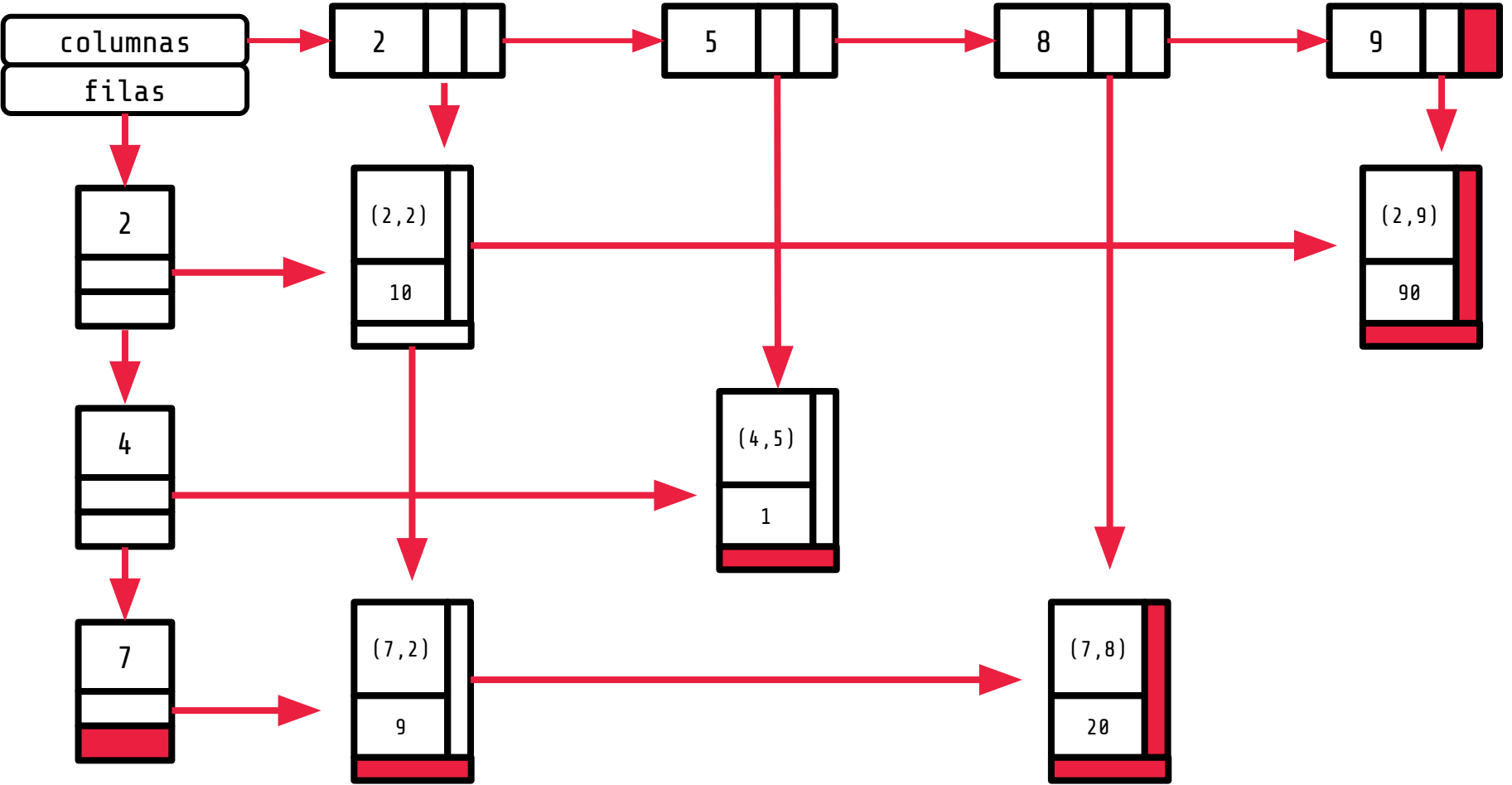


¿Preguntas?

Matrices dispersas

(una de las implementaciones posibles)





Versión densa

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	10	0	0	0	0	0	0	90
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	9	0	0	0	0	0	0	20	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0

Aplicaciones

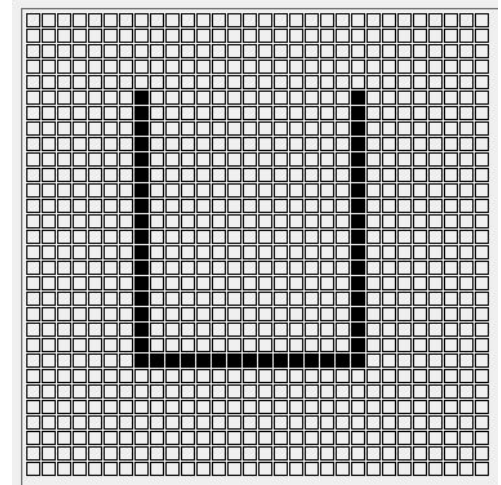
Simulación de elementos finitos

(¡con matrices 3D!)

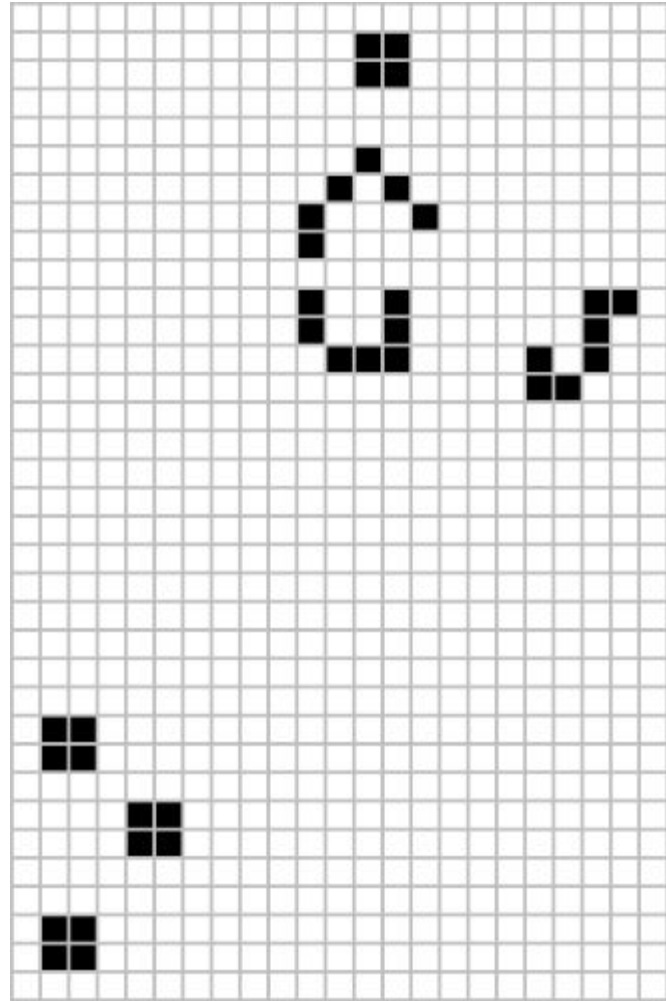
Inteligencia Artificial

Visión de máquina

(reconocimiento de patrones)



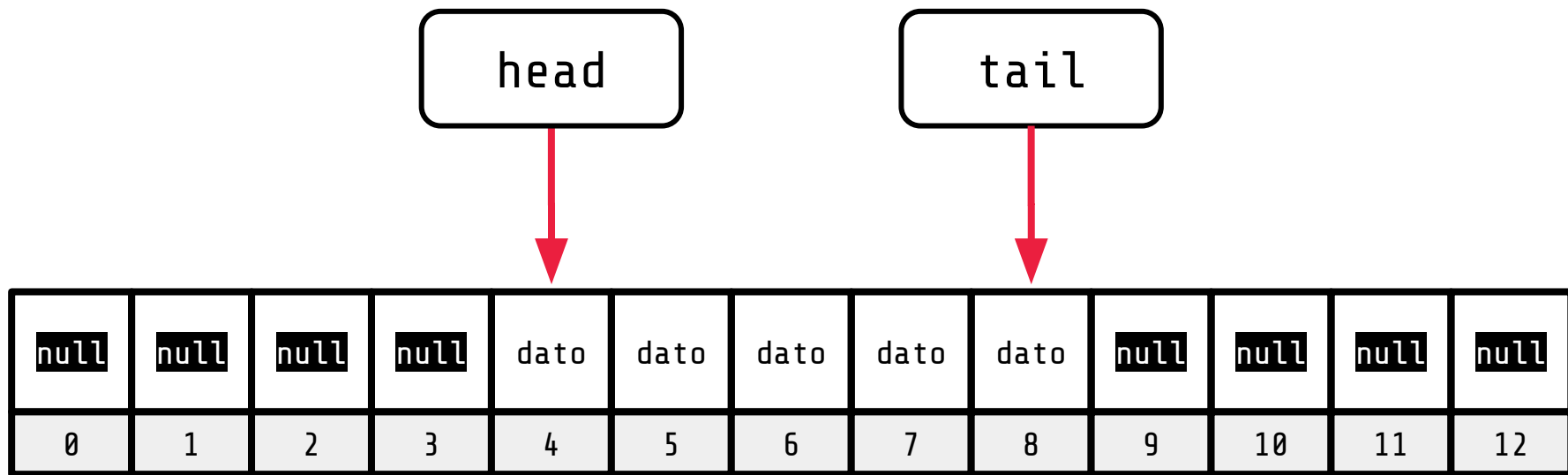
El juego de la vida de Conway



Algoritmos de caminos y mapas*

Colas

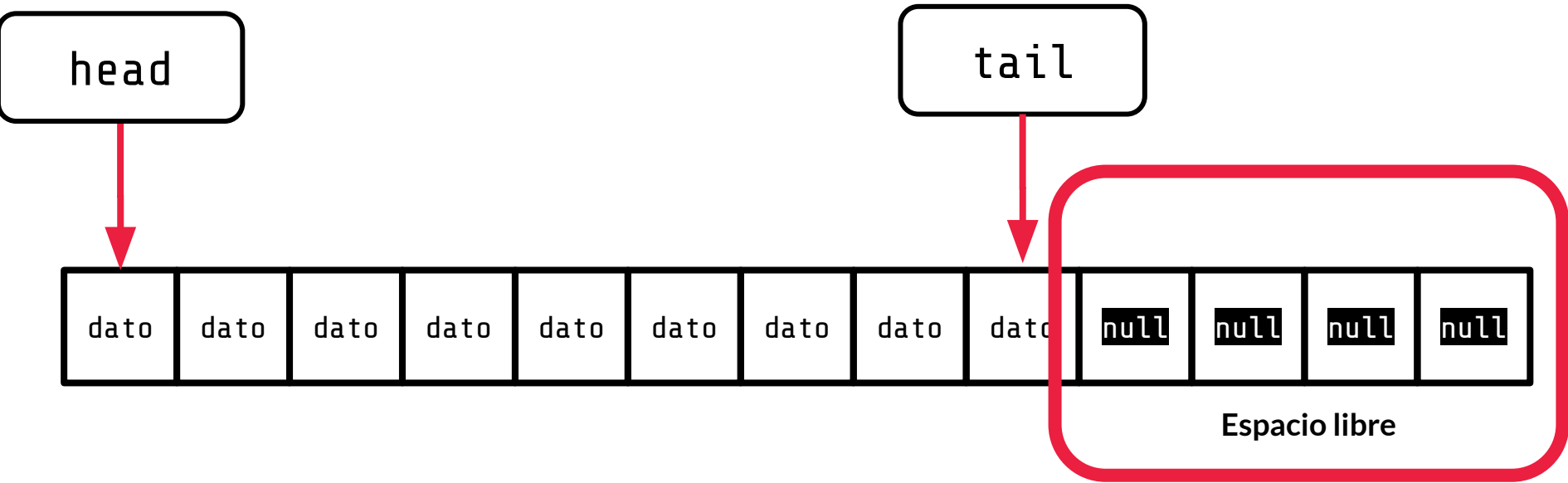
fijas/dinámicas



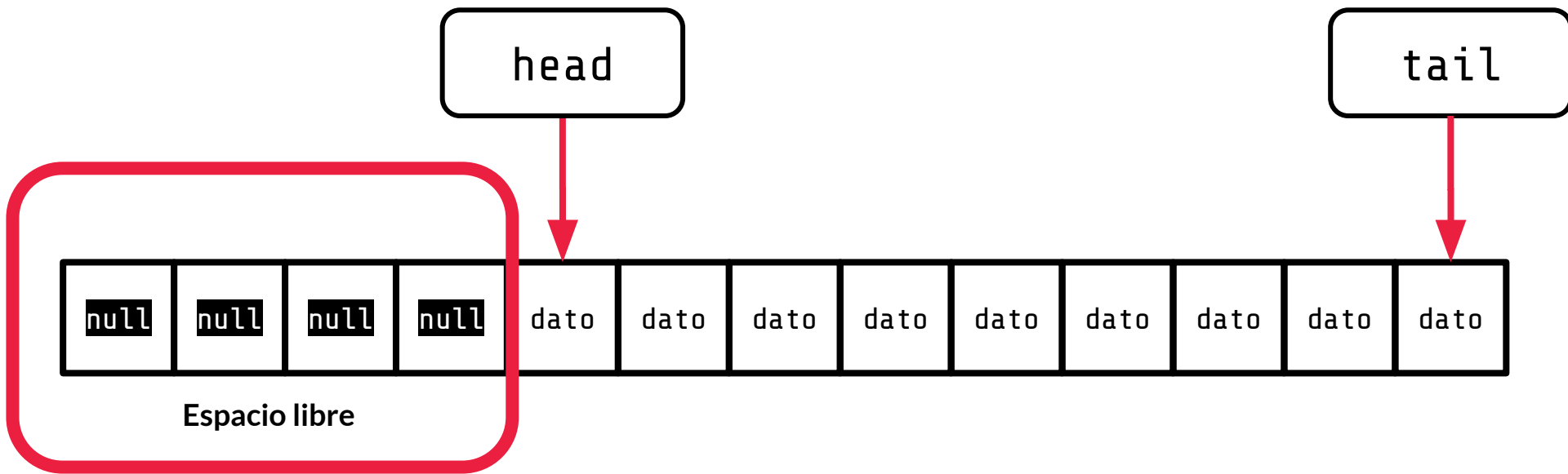


operaciones especiales

Balancear

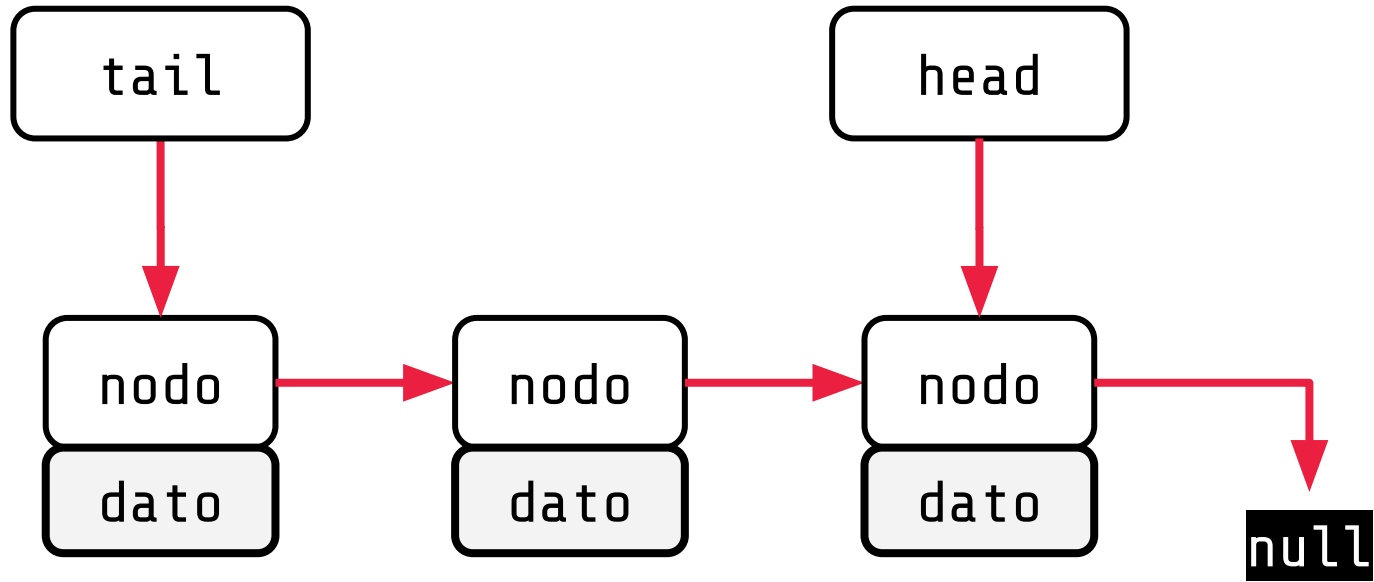


Cuando no podemos seguir 'encolando'

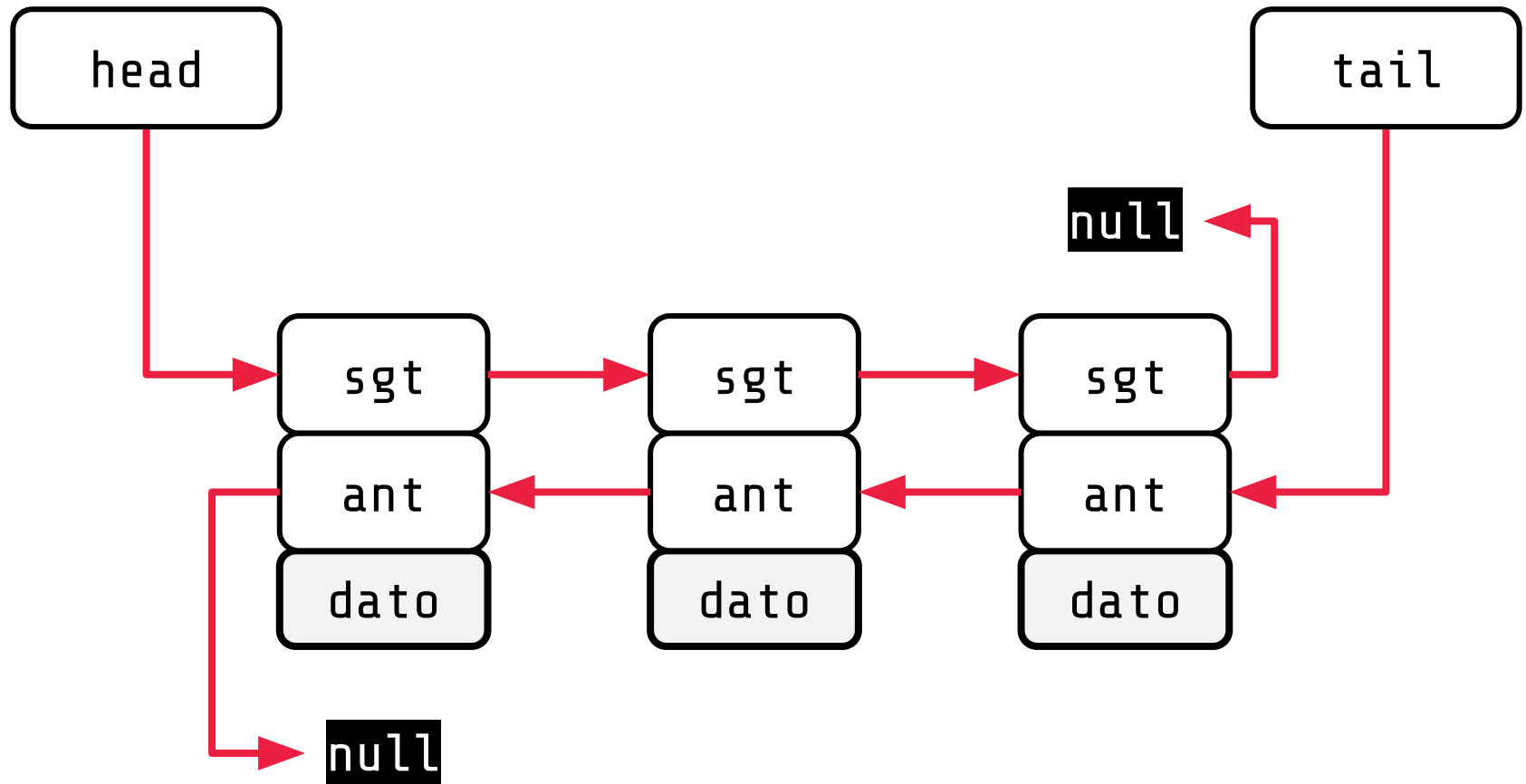


Tenemos que hacer lugar

Colas dinamicas



Dequeues



Se puede push/pop por ambos extremos

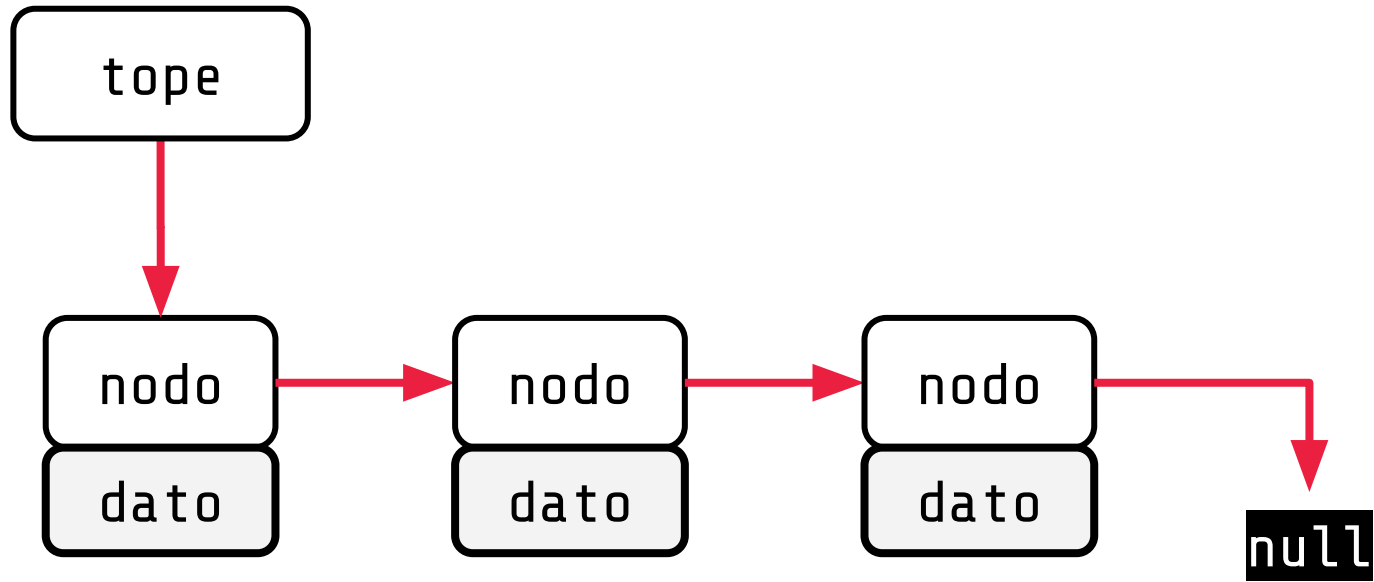
Pilas

fijas/dinámicas

tope



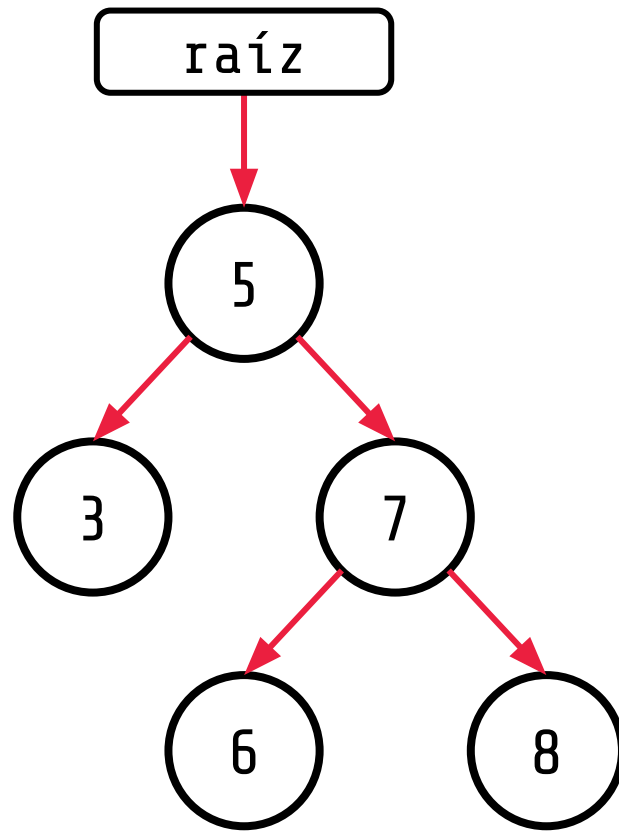
dato	dato	dato	dato	dato	dato	dato	dato	null	null	null
0	1	2	3	4	5	6	7	8	9	10

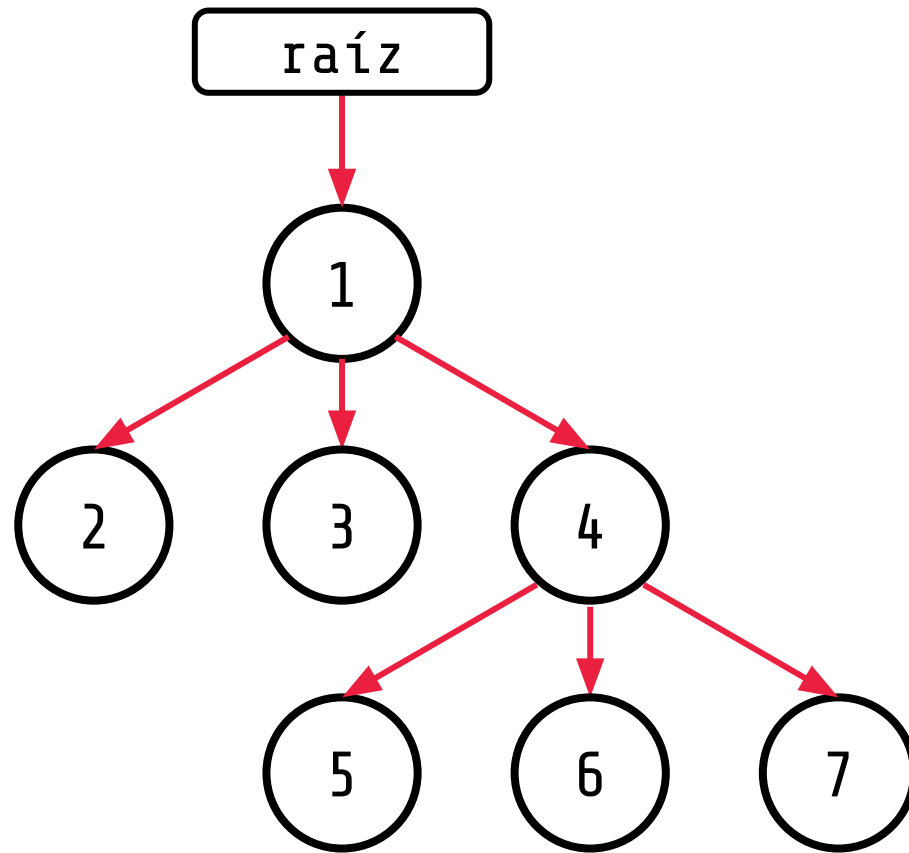


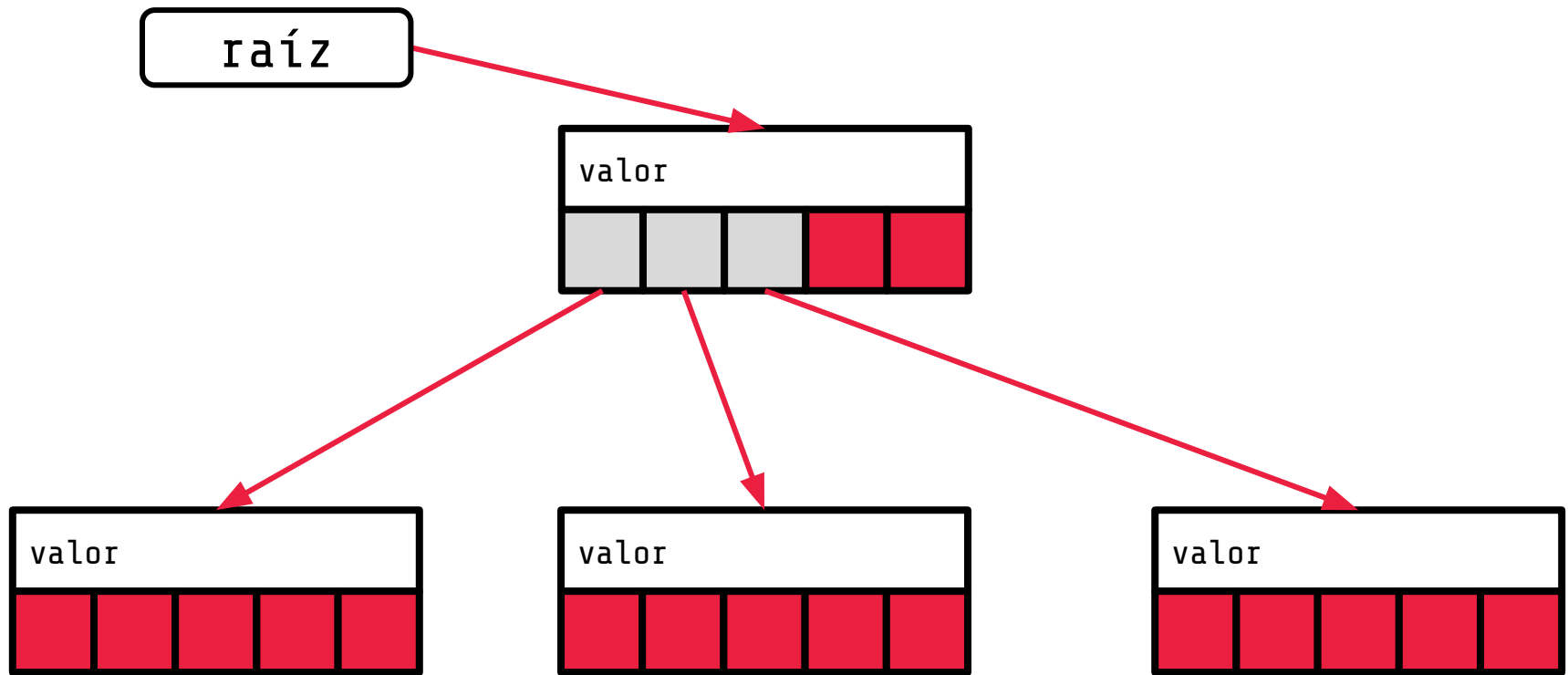


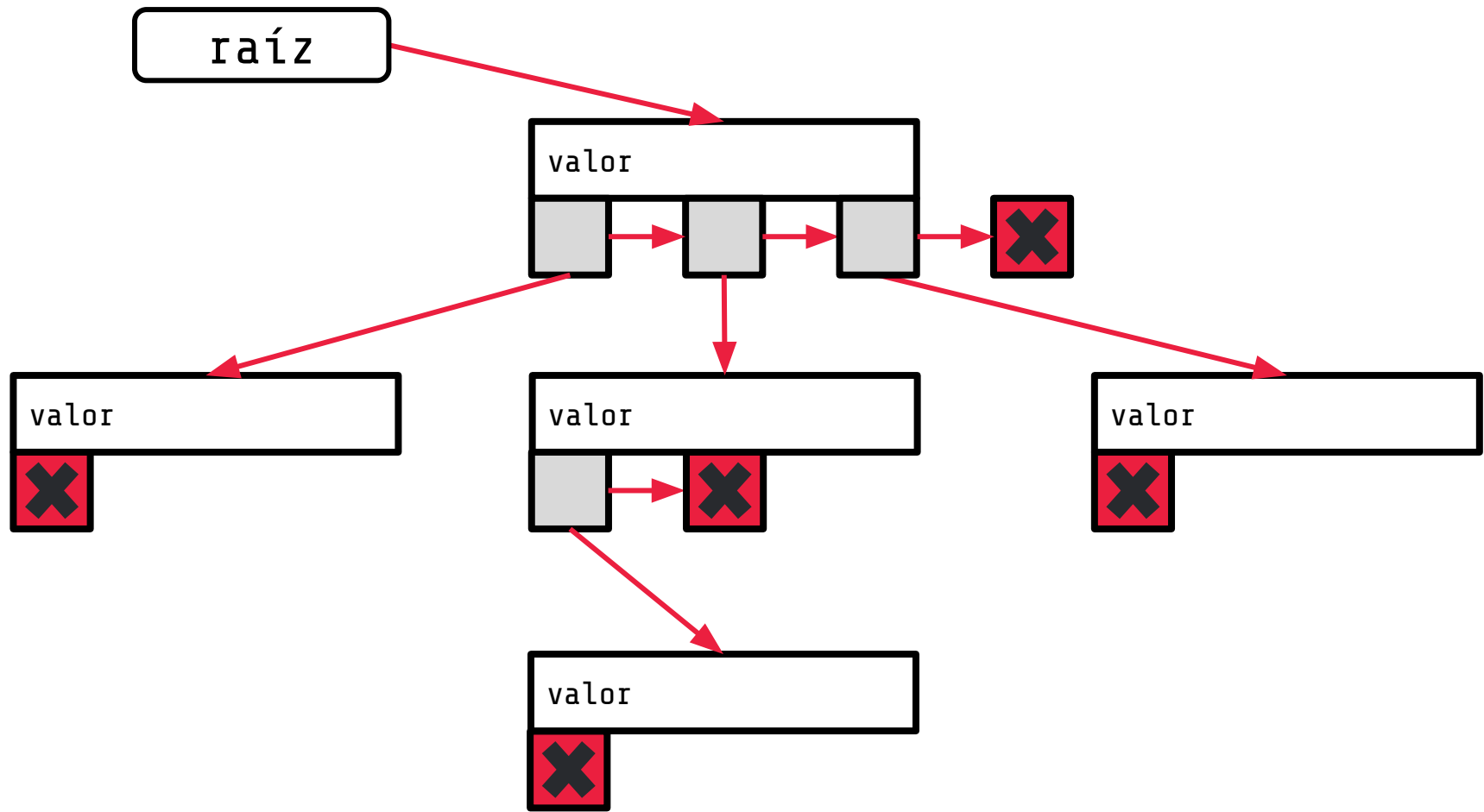
¿Preguntas?

Árboles











¿Preguntas?

Los recorridos en profundidad

preorden

1 se *visita* el nodo

2 seguimos con el izquierdo

3 seguimos con el derecho

En orden

1 vamos al izquierdo

2 se *visita* el nodo

3 vamos al derecho

Pos orden

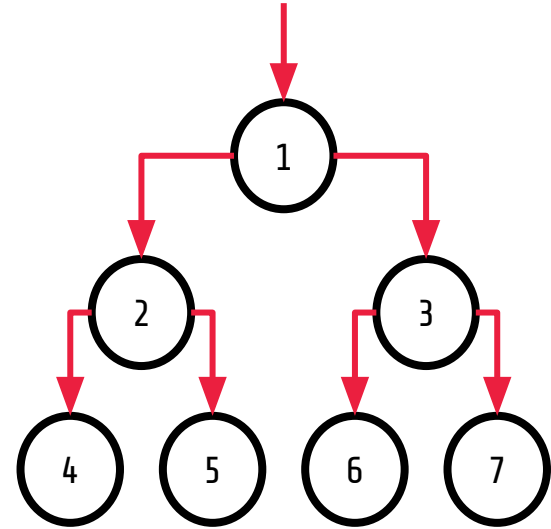
1 vamos al izquierdo

2 vamos al derecho

3 se *visita* el nodo

Recorrido en amplitud

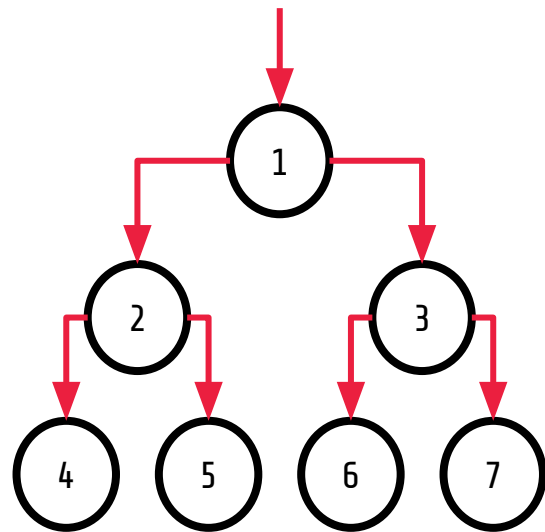
[1,2,3,4,5,6,7]



Requiere de una Queue

Iterativo es más simple

```
public static ListaEnlazada<T> enAmplitud(Nodo<T> raiz) {  
    if (raiz == null) {  
        return;  
    }  
    Queue<Nodo> cola = new Queue<>();  
    ListaEnlazada<T> recorrido = new ListaEnlazada<T>();  
    cola.agregar(raiz);  
  
    while (!cola.estaVacia()) {  
        Nodo nodo = cola.obtener();  
        recorrido.insertar(nodo.valor());  
  
        if (nodo.izquierda() != null) {  
            cola.agregar(nodo.izquierda());  
        }  
  
        if (nodo.derecha() != null) {  
            cola.agregar(nodo.derecha());  
        }  
    }  
    return recorrido;  
}
```

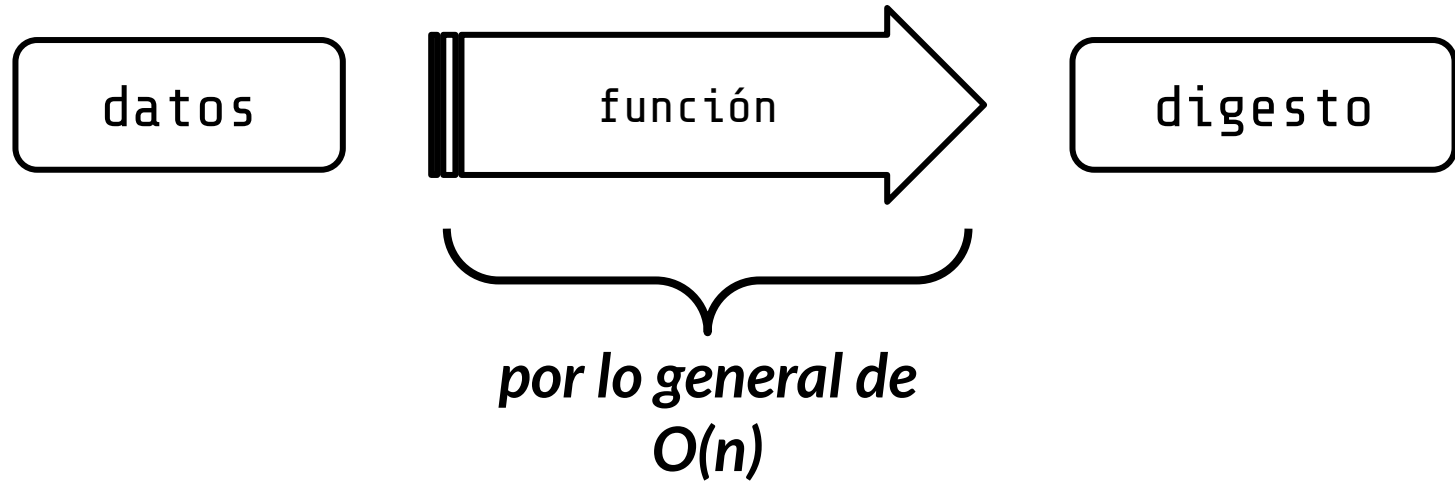




¿Preguntas?

Introducción a las funciones de extracto

hash/extract/extracto/digesto



La función nos resume los datos en un digesto

**Lo que entra, sale
con ancho fijo**

perro

0xA479CB60

este es un
perro

función
CRC-32

0xA46E32BF

perre

0x44AC227E

**El contenido original
no es reconstruible**

**Hay múltiples
algoritmos
CRC-8/16/32
SHA-1 (160bits)**

***hay muchos mas**

**¡Pueden existir dos datos
con el mismo valor!**

Esto es llamado **colisión**

Aplicaciones

Control de versiones

Blockchain

**Los datos se guardan con el
digesto de lo anterior**

Monitoreo de salud de datos y archivos

(cualquier cambio, altera el
digesto)

Diccionarios*

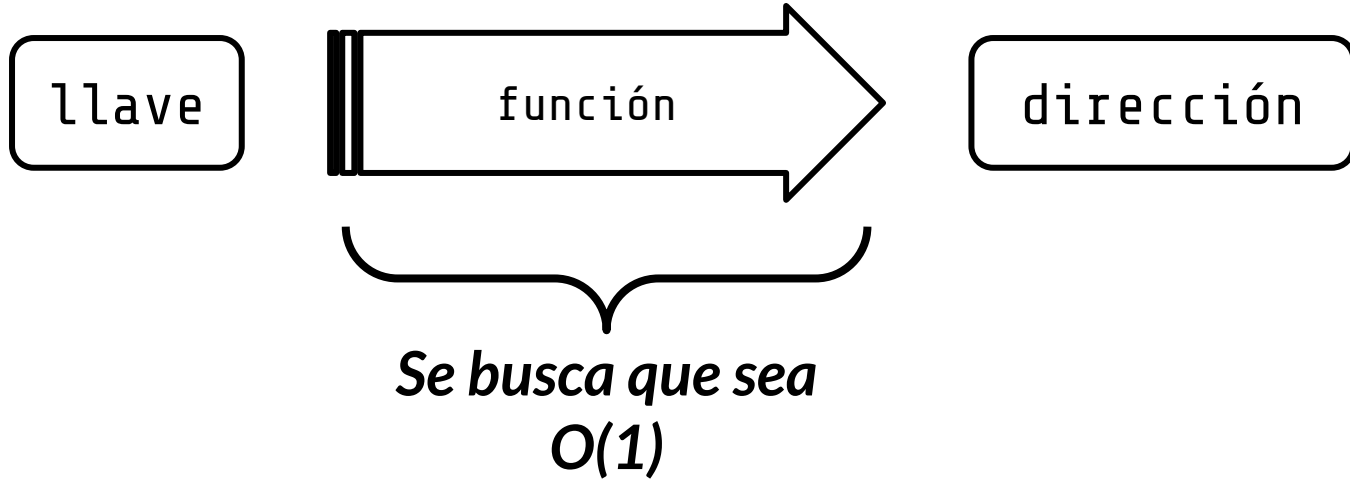
Tablas de Hash

***conceptualmente**



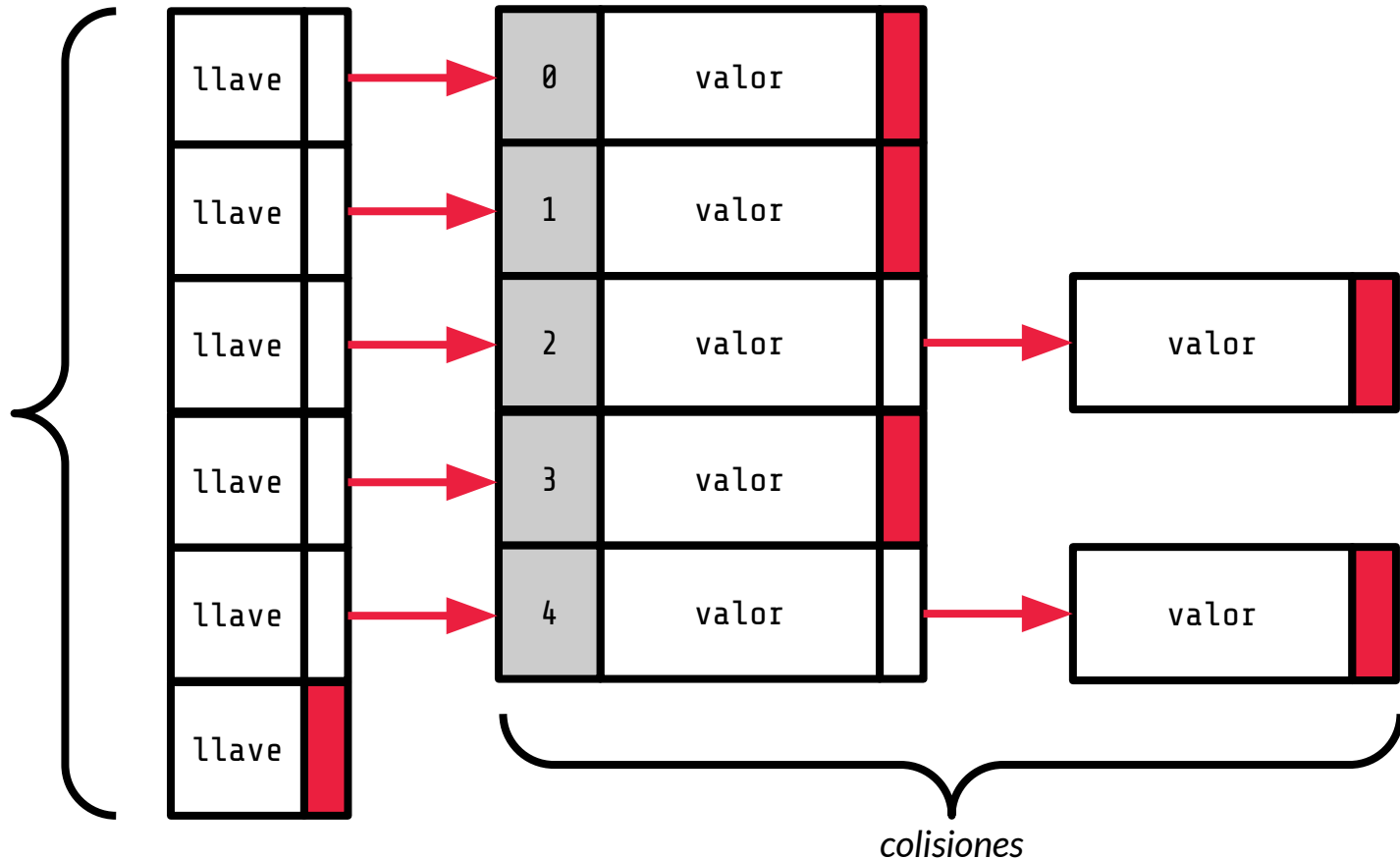
**Las funciones de
digesto son
costosas en tiempo**

**Por eso se busca
una función que sea
 $O(1)$**



Esta nos indicará donde guardar un valor

*no tiene por
qué estar en
orden*



Una implementación simple (y simplificada)

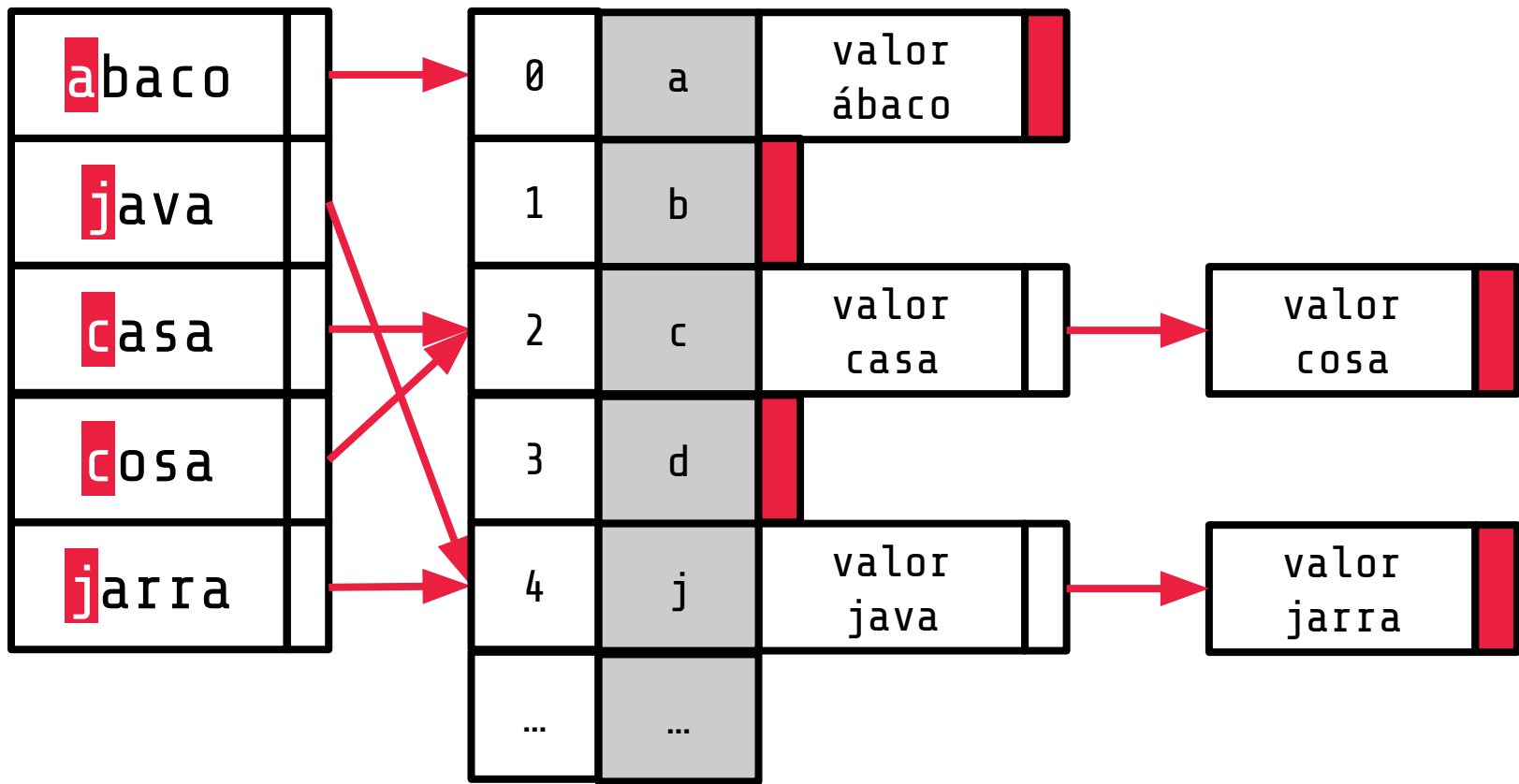
Un ejemplo simple

**Una función de
hash, puede tomar
la primera letra de
una cadena**

Una implementación simple (demasiado)

```
static int hashimple(String cadena){  
    char[] caracteres = cadena.toCharArray();  
    int valor = caracteres[0] - 'A';  
    return valor;  
}
```

*Tiene sus propios problemas ser **tan simple***



Usamos la primera letra como llave

**Las colisiones se
pueden resolver
como una lista
enlazada**

unrn.edu.ar

