Estructuras avanzadas

XX

I

2025

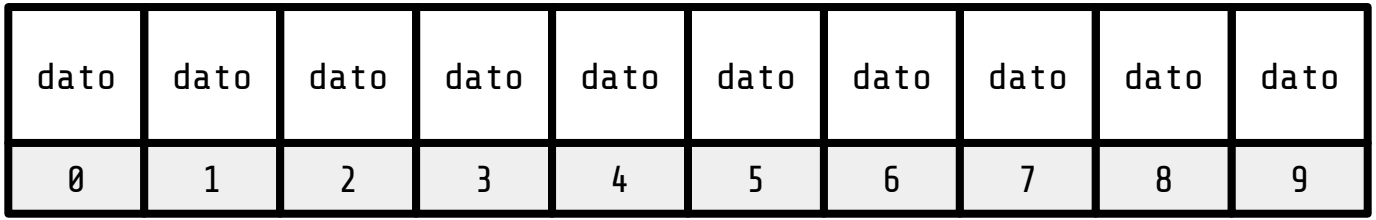
UNRN

Universidad Nacional
de **Río Negro**

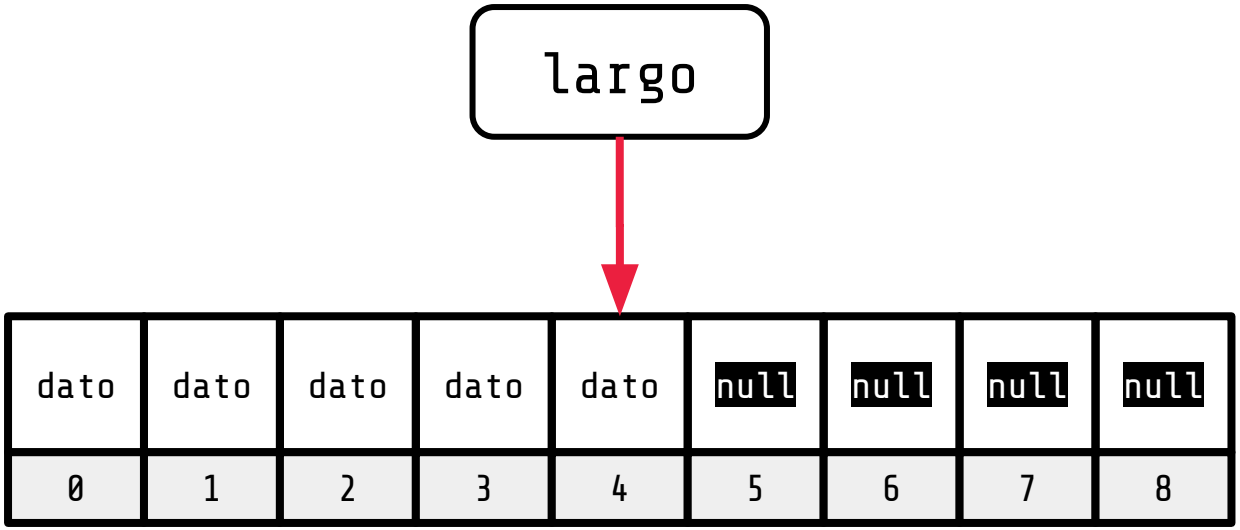
r27

Lineales

Arreglos



Arrays de tamaño fijo

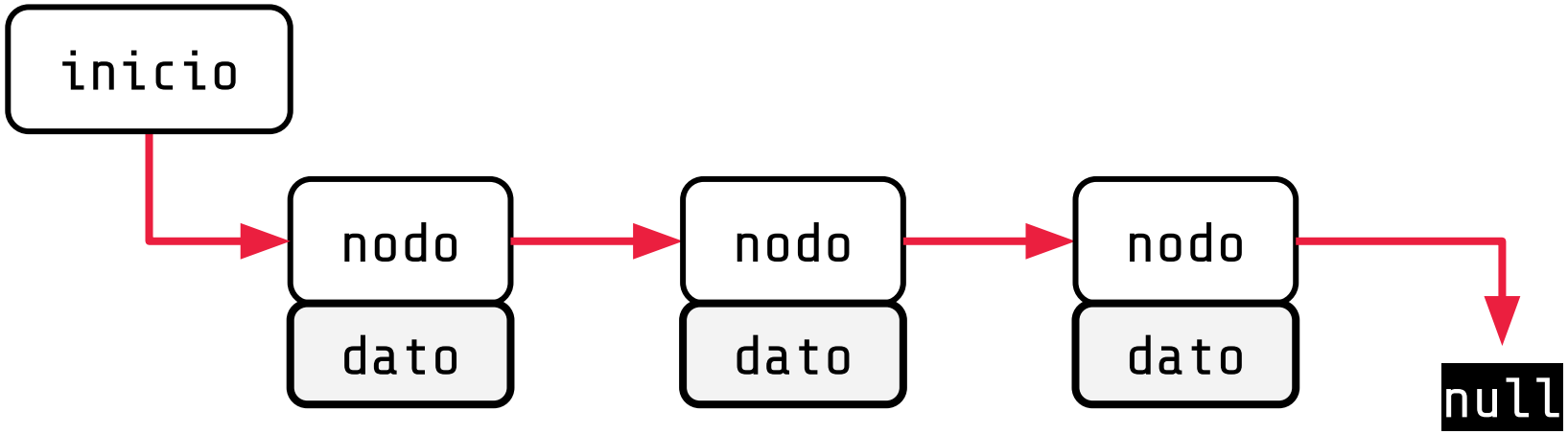

largo
dato
dato
dato
dato
dato
null
null
null
null
0
1
2
3
4
5
6
7
8

Arreglo

- **Acceso: O(1) / Ω(1)**
- **Modificación: O(1) / Ω(1)**
- ***Inserción*: O(n) / Ω(n)**
- ***Borrado*: O(n) / Ω(n)**
- **Ordenamiento*: $O(n^2)$ / Ω(n)**
- ***Ampliación*: O(n) / Ω(n)**

Listas Enlazadas

Conjunto dinámico



Simple, dinámica con nodos

Listas enlazadas

- Acceso: $O(n)$ / $\Omega(1)$
- Modificación: $O(n)$ / $\Omega(1)$
- Inserción: $O(n)$ / $\Omega(1)$
- Borrado: $O(n)$ / $\Omega(1)$
- Ordenamiento: $O(n^2)$ / $\Omega(1)$
- Ampliación: $\Omega(1)$ a $O(n)$

inicio	4									
enlaces	1	-1	0	6	3	-1	2	-1	-1	-1
datos	dato	dato	dato	dato	dato	dato	dato	dato	dato	dato
	0	1	2	3	4	5	6	7	8	9

Podemos usar un doble arreglo

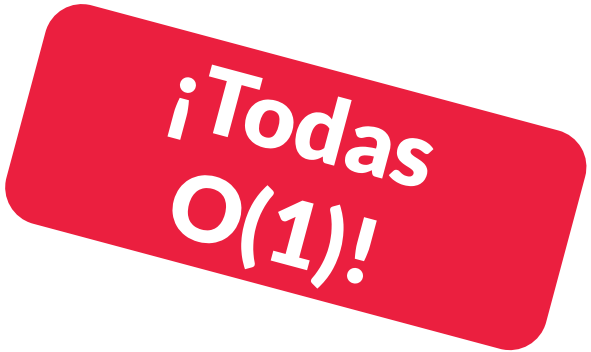
Pila / stack
LIFO

Operaciones

Stack:

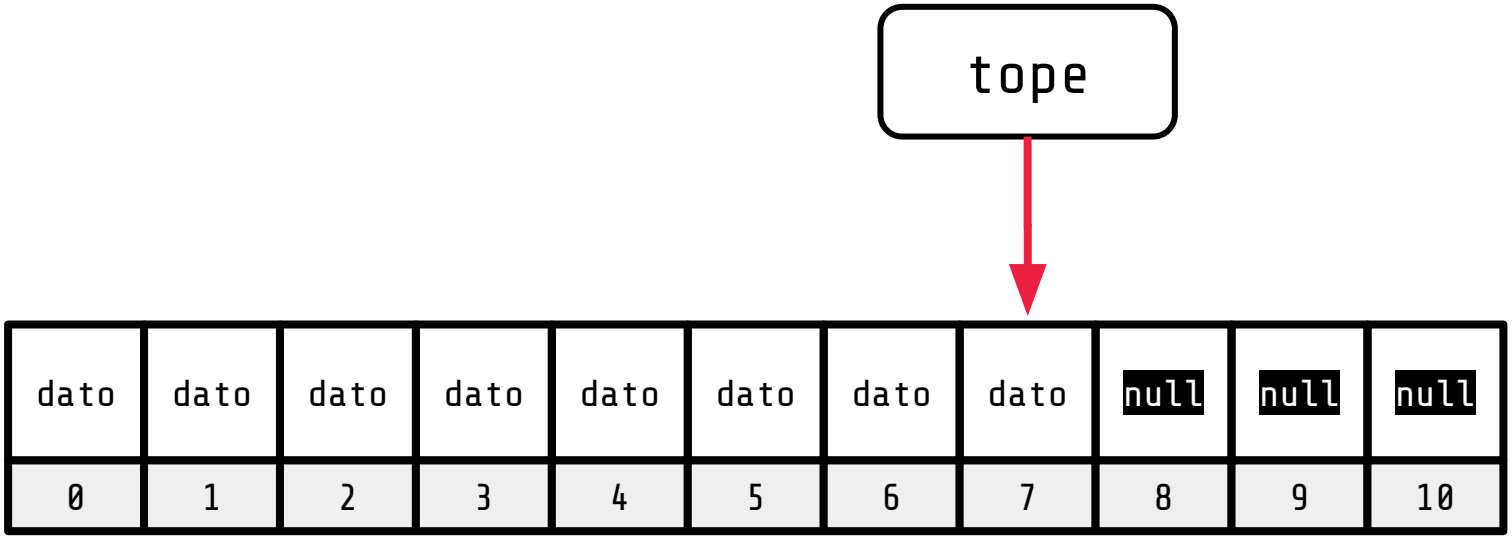
- push - agregar un elemento a la pila
- pop - extraer un elemento
- top - ver el siguiente elemento que se extraería
- isEmpty - si hay elementos en la Pila.

Operaciones

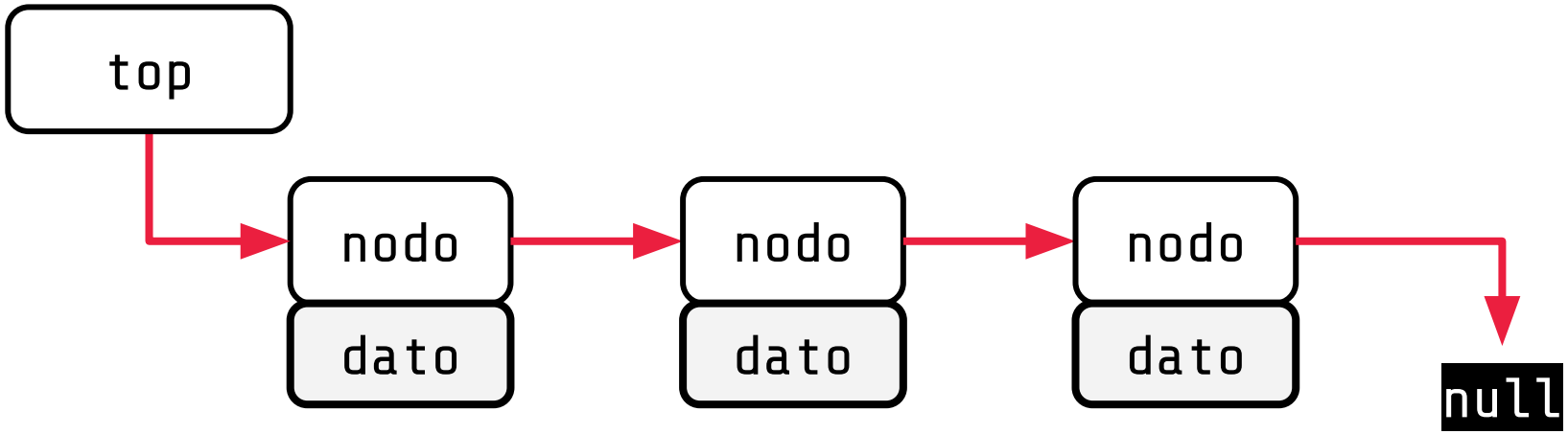


Stack:

- push - agregar un elemento a la pila
- pop - extraer un elemento
- top - ver el siguiente elemento que se extraería
- isEmpty - si hay elementos en la Pila.



Simple, fija con arreglos



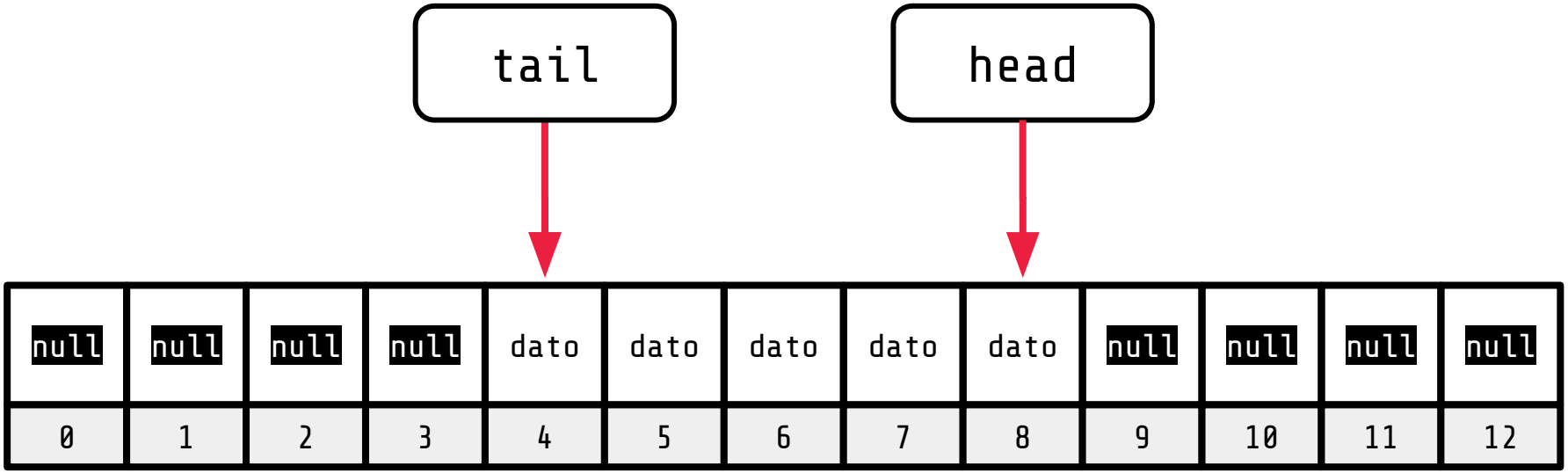
Simple, dinámica con nodos

Cola / queue
FIFO

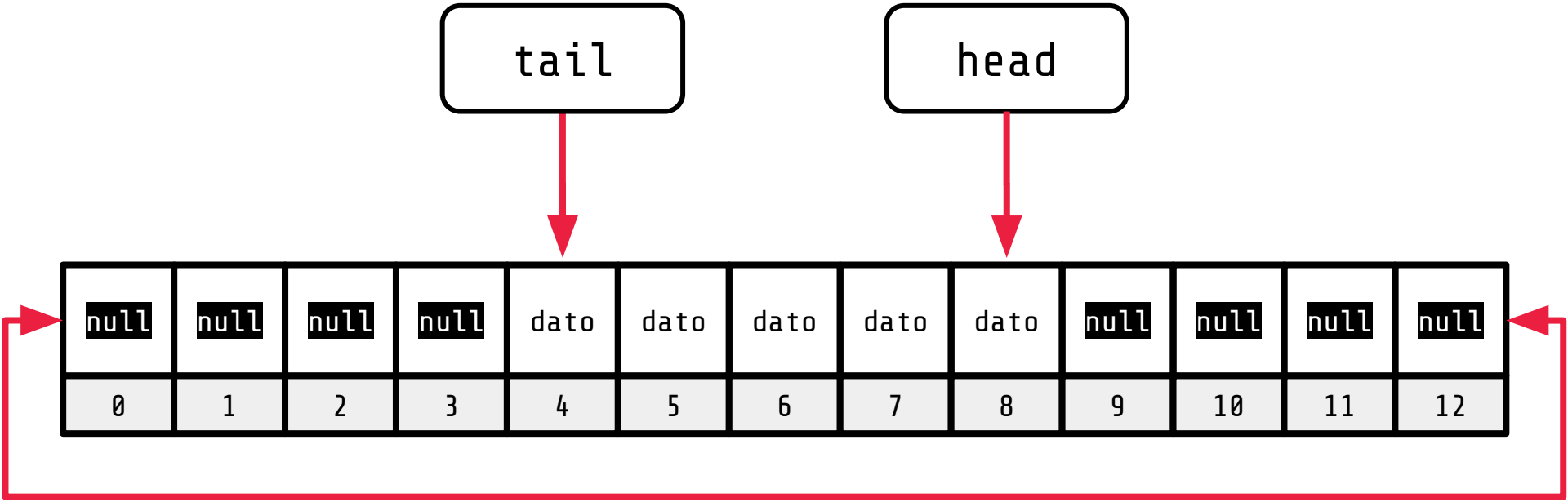
Operaciones

Queue:

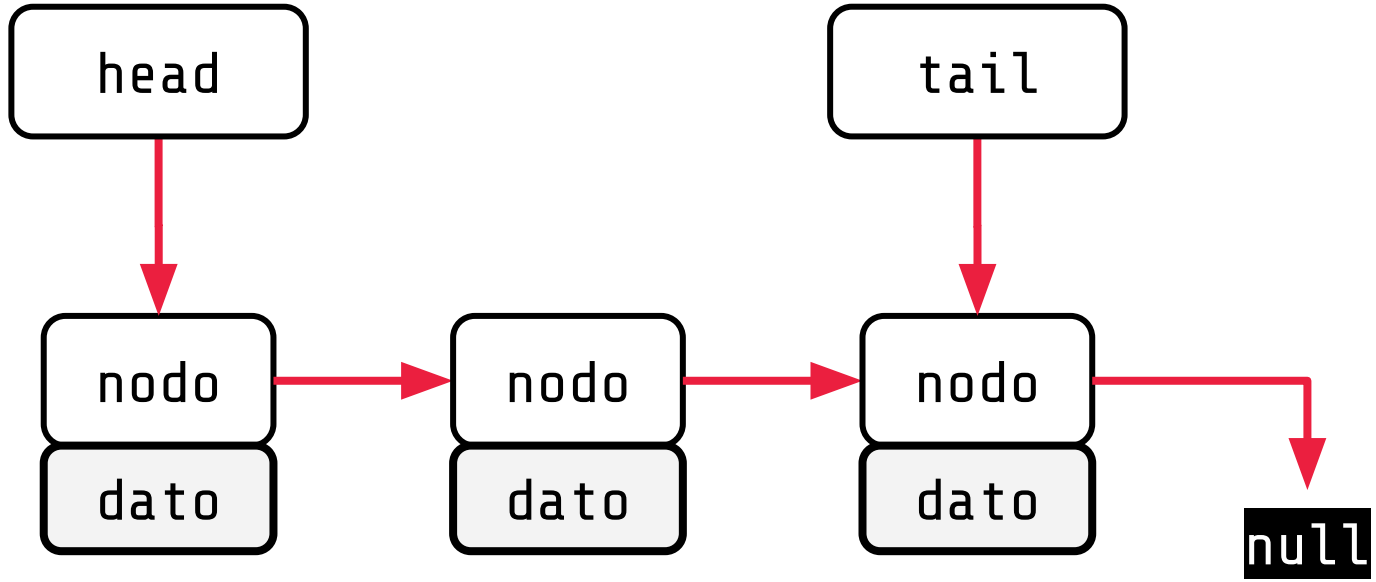
- put: agregar un elemento por el final (tail).
- get: tomar un elemento del inicio (head)
- isEmpty: Si no hay elementos
- peek: el siguiente elemento de la Queue sin sacarlo.



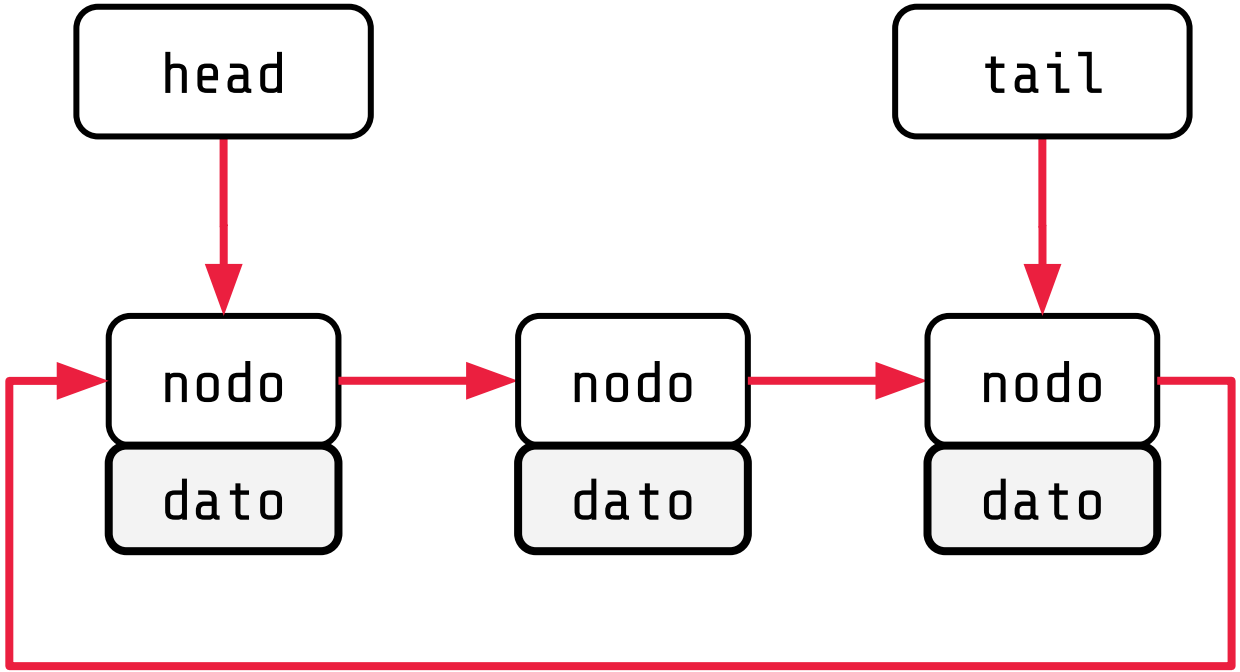
Simple, fija con arreglos



Circular con arreglos



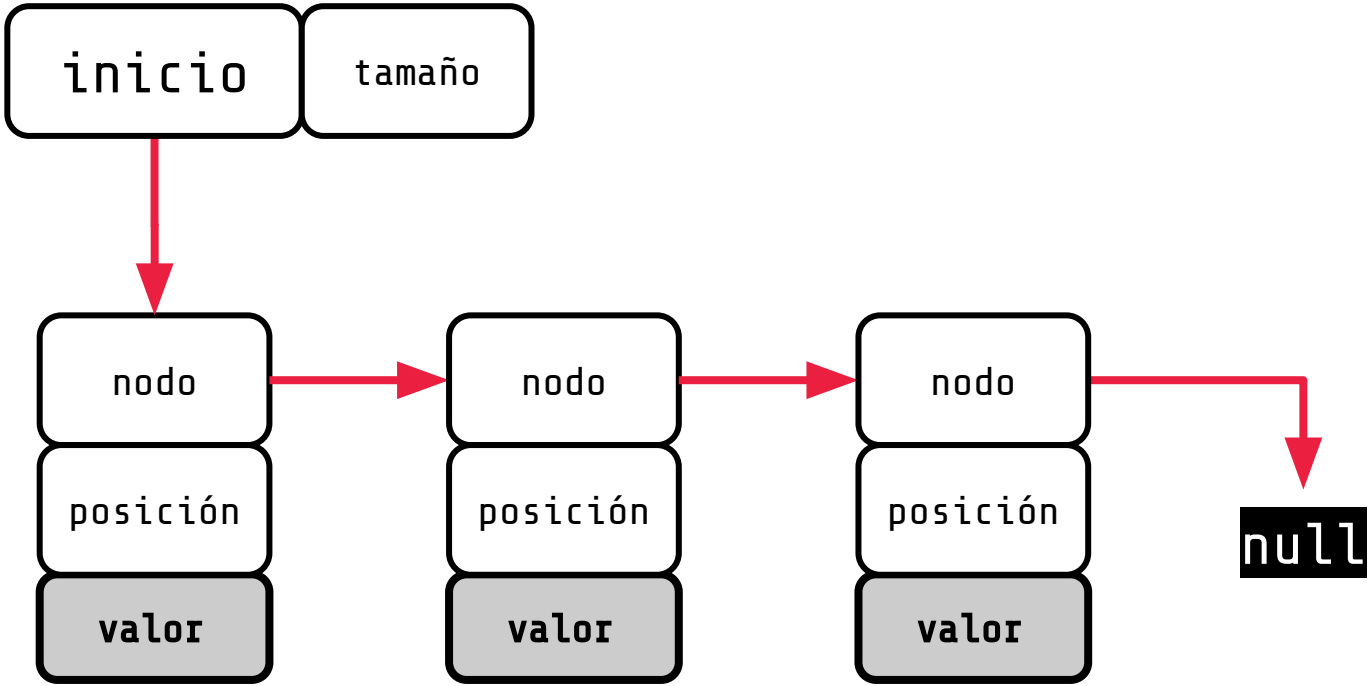
Simple, dinámica con nodos



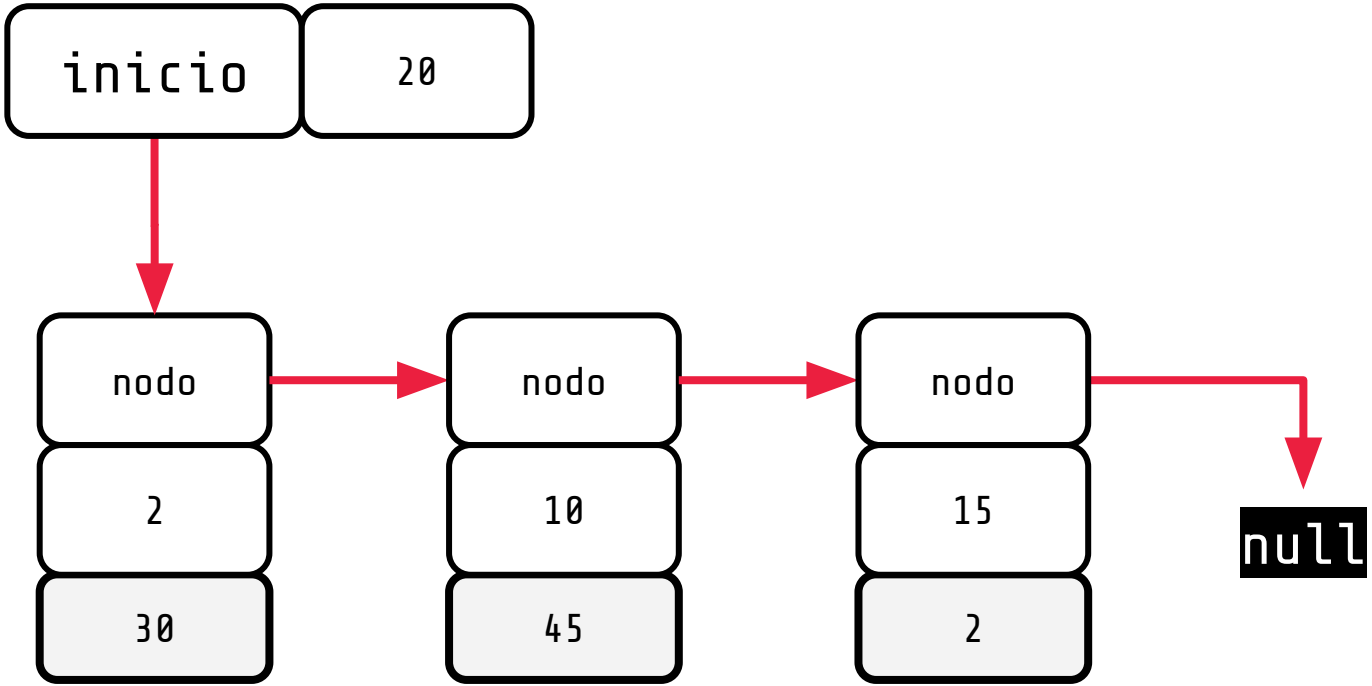
Circular, dinámica con nodos

Arrays dispersos

(sparse)



Como una forma de ahorrar espacio

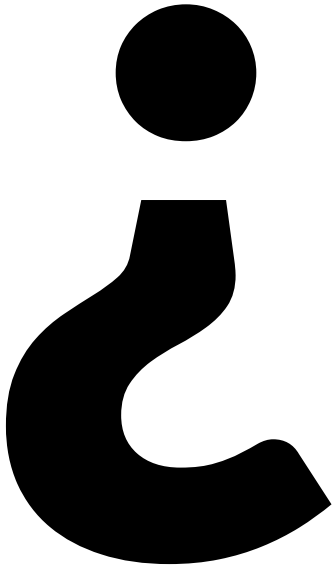


Este `array` que tiene tres valores de interés

El arreglo sería algo como

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

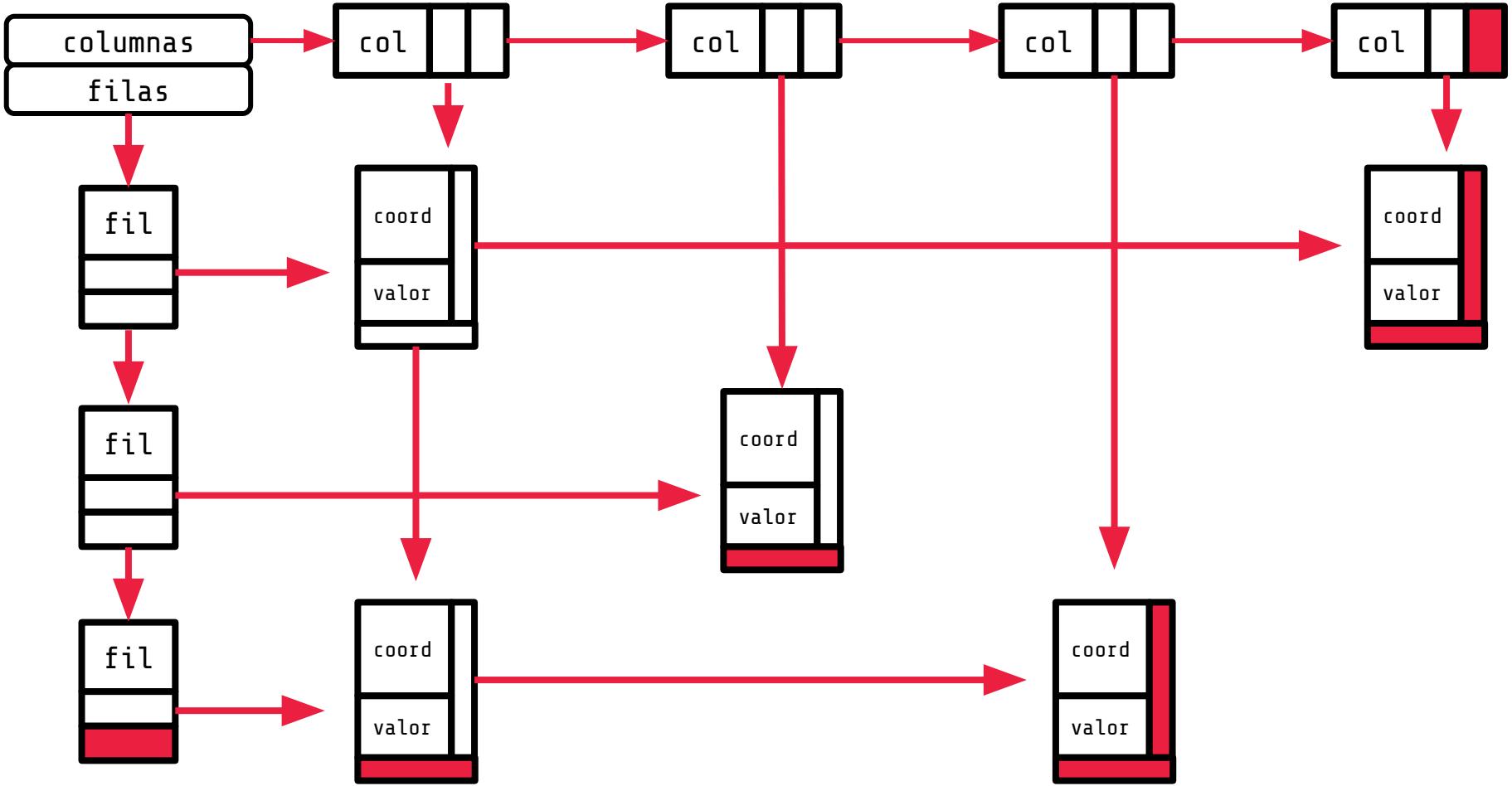
{ Ahorramos espacio sacrificando tiempo }

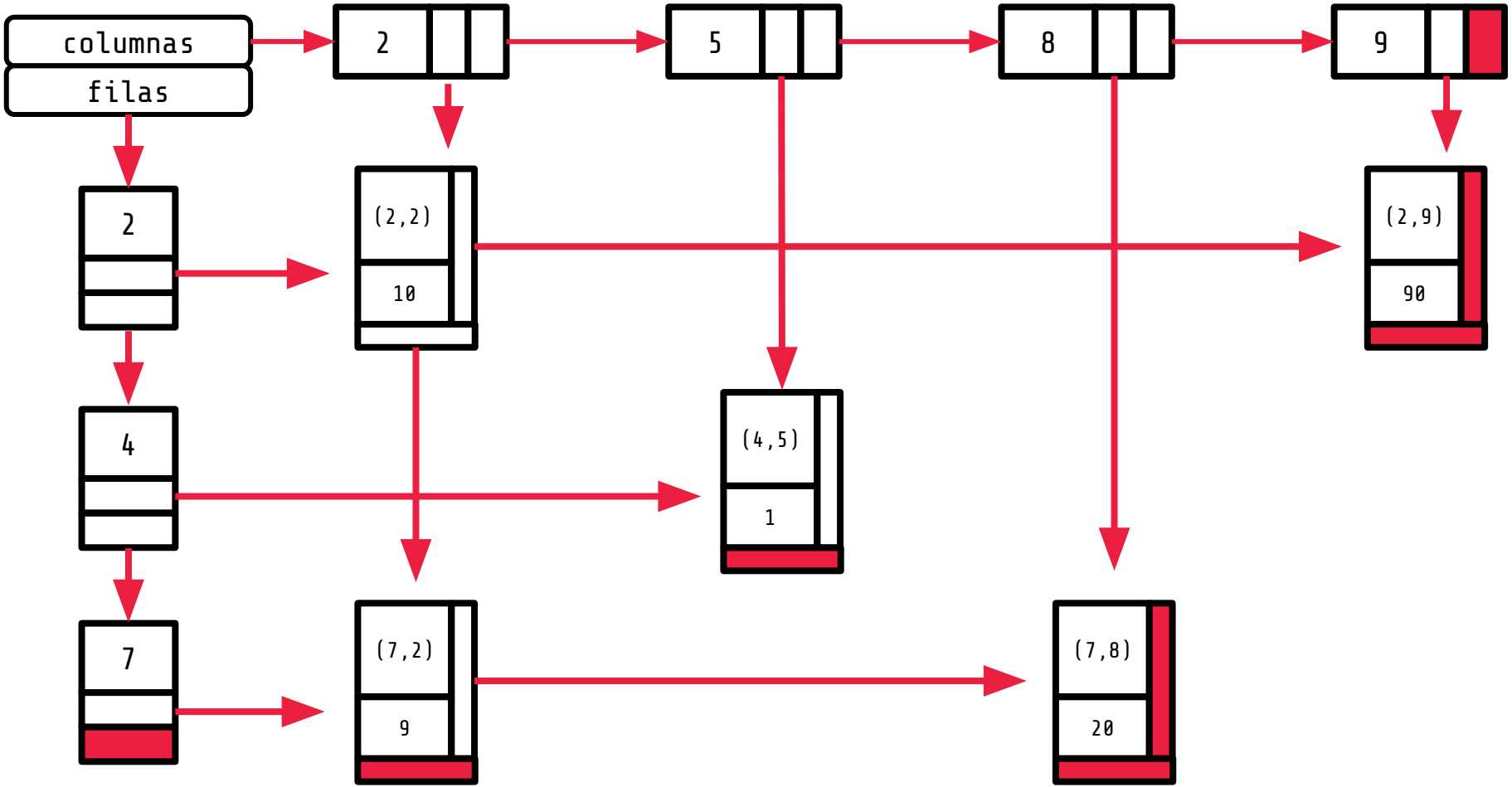


Y una matriz

Matrices dispersas

(una de las implementaciones posibles)

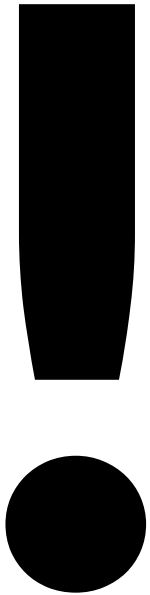

columnas
filas
col
col
col
col
fil
fil
fil
coord
valor
coord
valor
coord
valor
coord
valor
coord
valor


columnas
filas
2
5
8
9
2
4
7
(2,2)
10
(2,9)
90
(4,5)
1
(7,2)
9
(7,8)
20

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

Puede ser 3D también



Aplicaciones

Simulación de elementos finitos

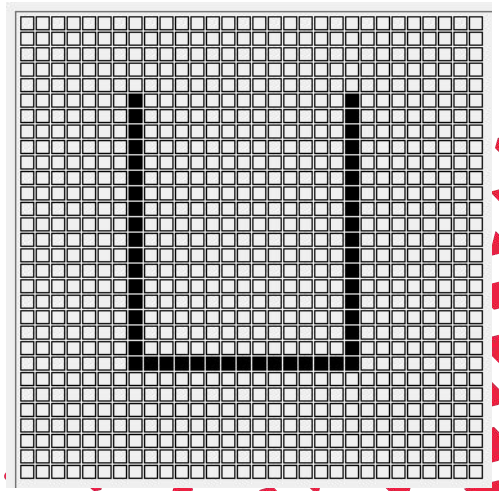
(¡con matrices 3D!)

UNRN | Universidad Nacional de Río Negro

Inteligencia Artificial

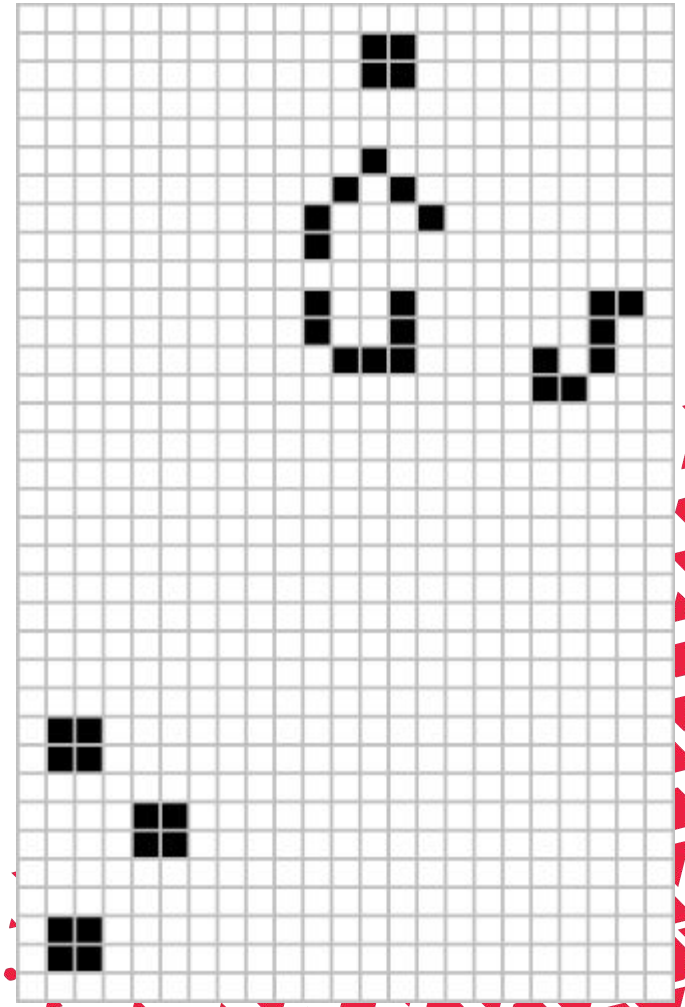
Visión de máquina

Redes neuronales convolucionales



UNRN | Universidad Nacional de Río Negro

El juego de la vida de Conway



https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

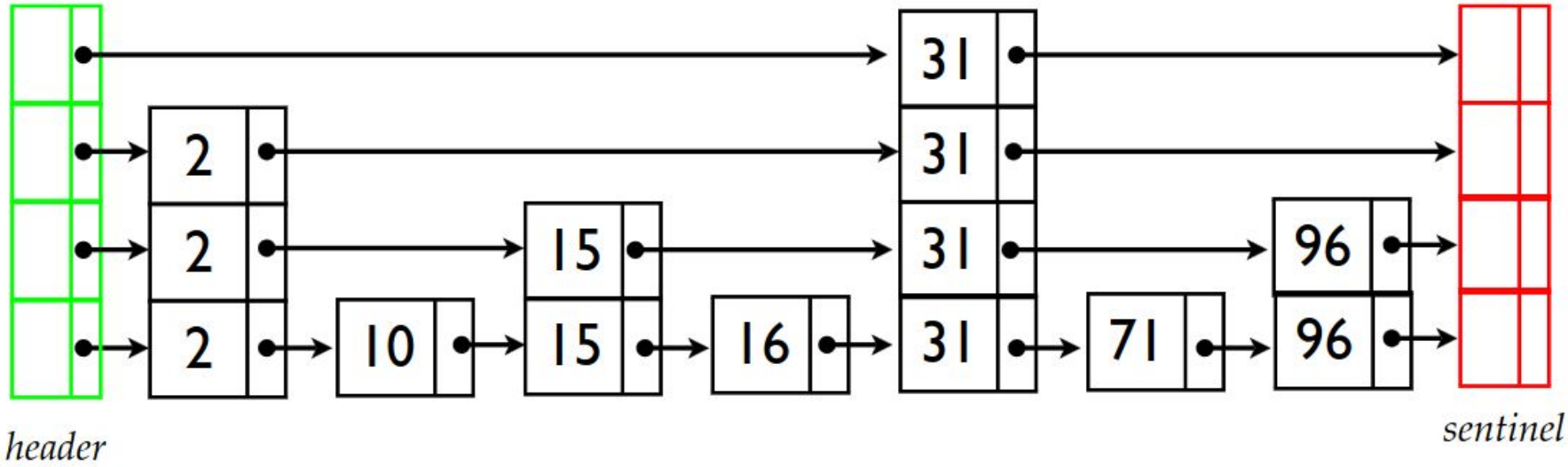
Algoritmos de grafos
caminos y mapas

UNRN | Universidad Nacional de Río Negro


¿Preguntas?

Skip Lists

Para listas ordenadas

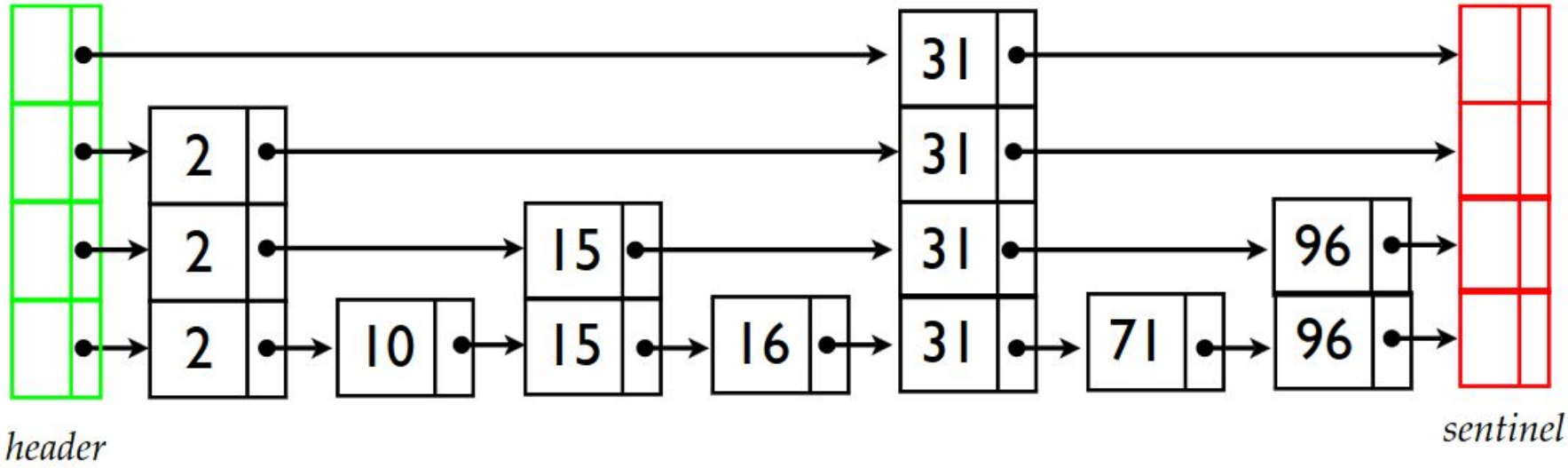


El nodo centinela (derecha) está como tope

Características

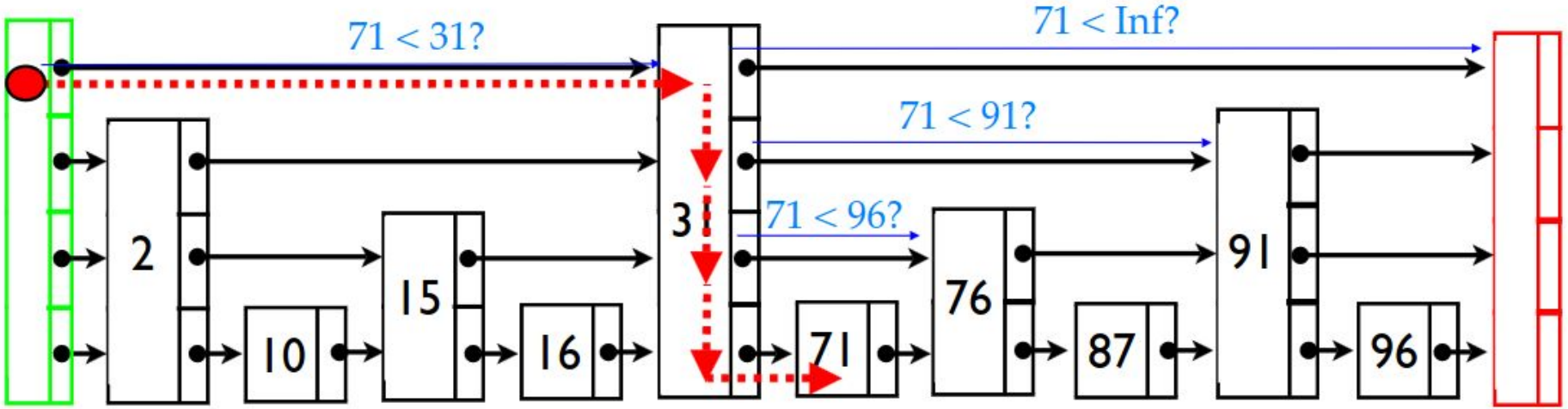
- Los valores están ordenados
- $\log n$ niveles
- La cabecera y centinela están en todos
- Cada nivel superior contiene la mitad de elementos que el inferior

Busquemos 71



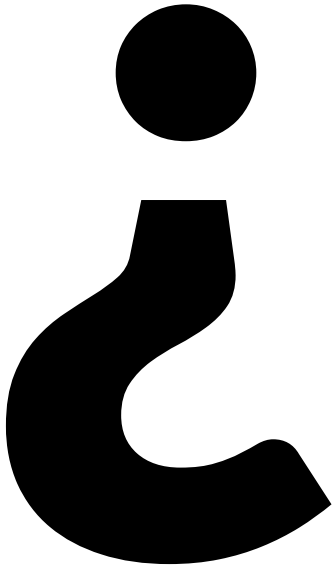
¿Cómo buscamos?

Busquemos 71



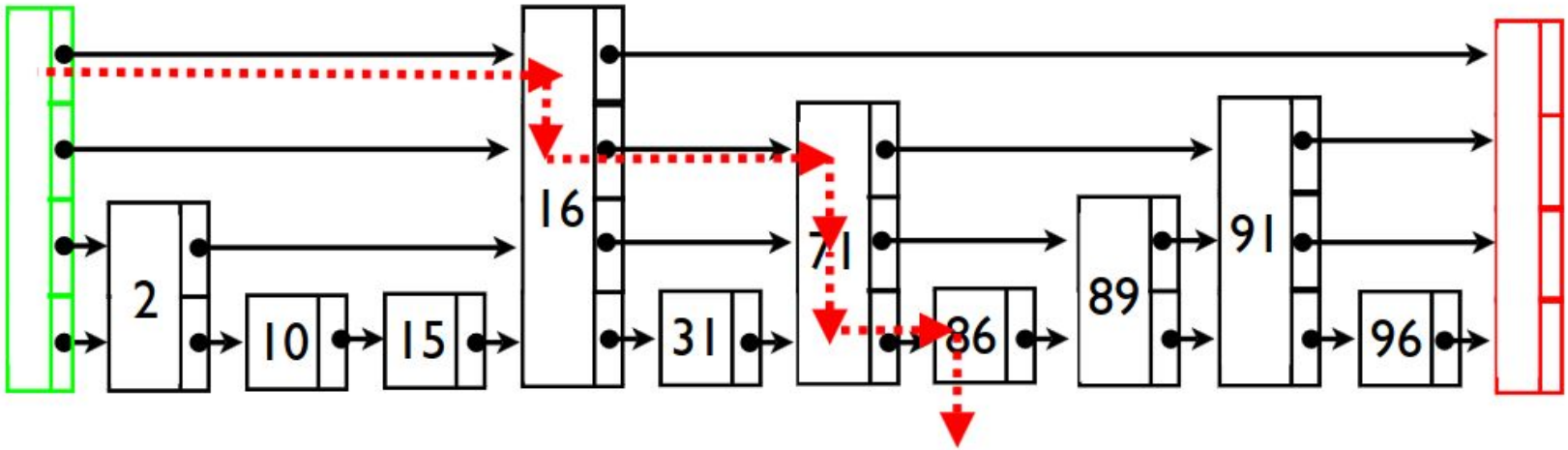
¿Cómo buscamos?

{ Esto para skip lists perfectamente distribuidas }

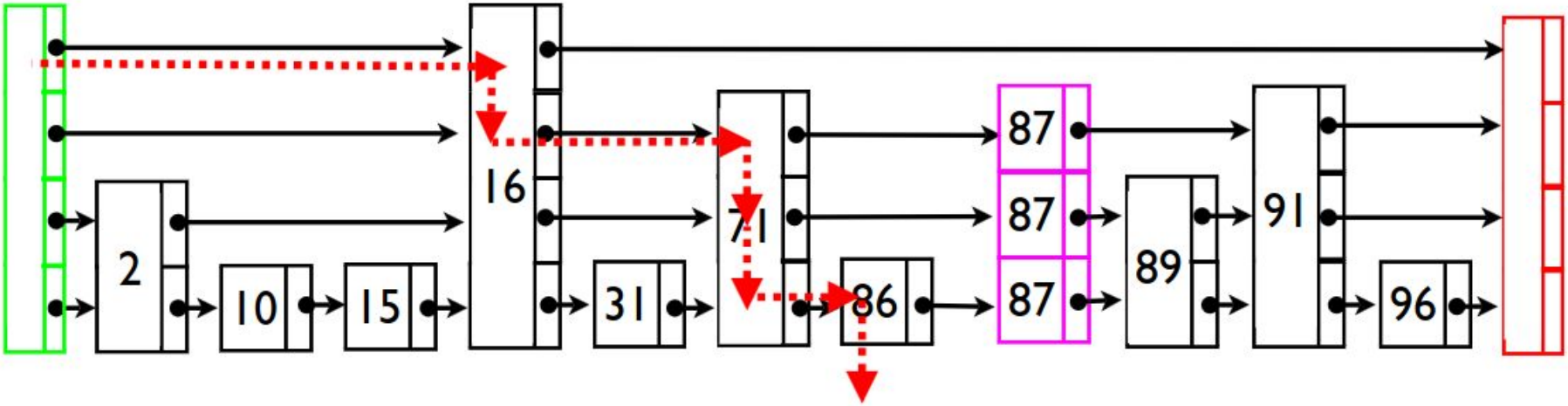


Y las inserciones, borrado

Se *afloja* un poco el balance de contenido



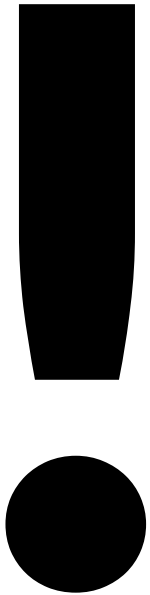
Para insertar 87



buscamos k
se inserta en el nivel 0
nivel = 0
while **random(0,1)** == 1 {
    insertar(k, nivel)
    nivel++

Random?

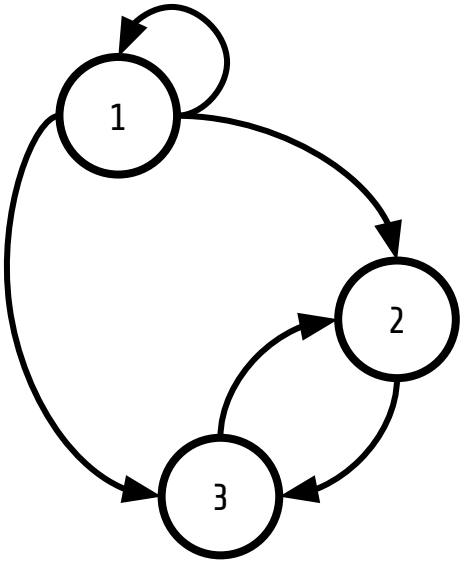
Aflojar es tirar una moneda

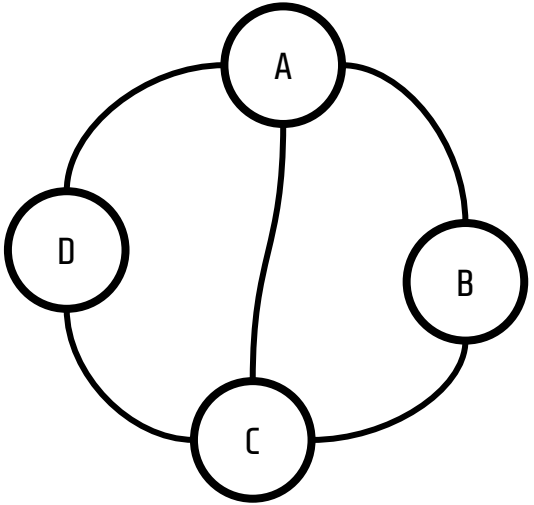


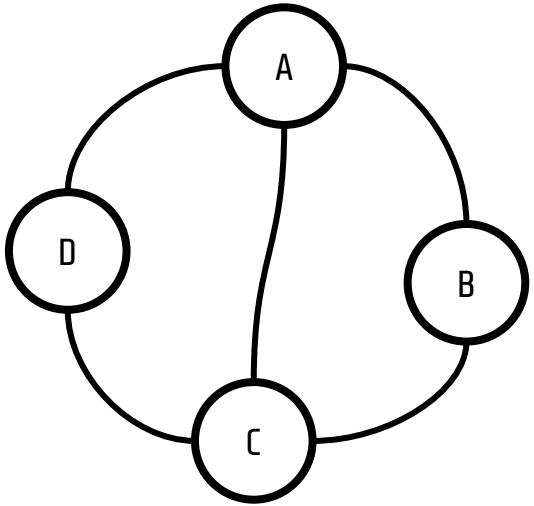
En el peor de los casos se transforma en una lista enlazada

Para más información

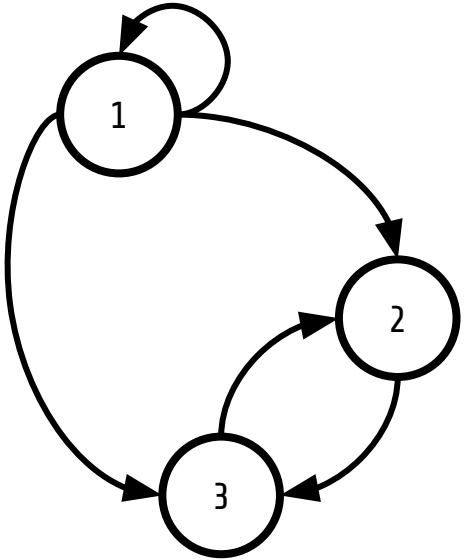
Grafos


1
2
3

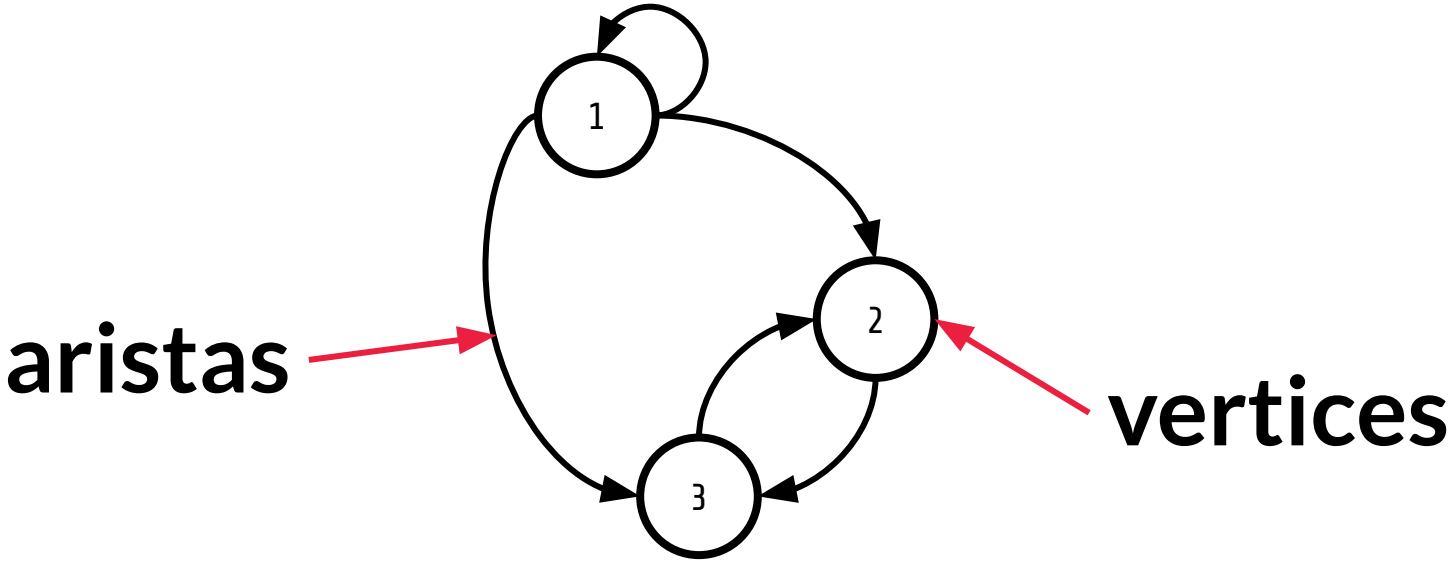

A
D
B
C



no dirigidos

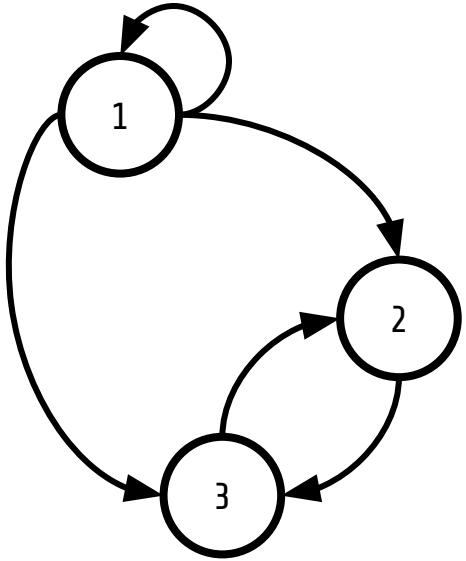


dirigidos


aristas
1
2
3
vertices

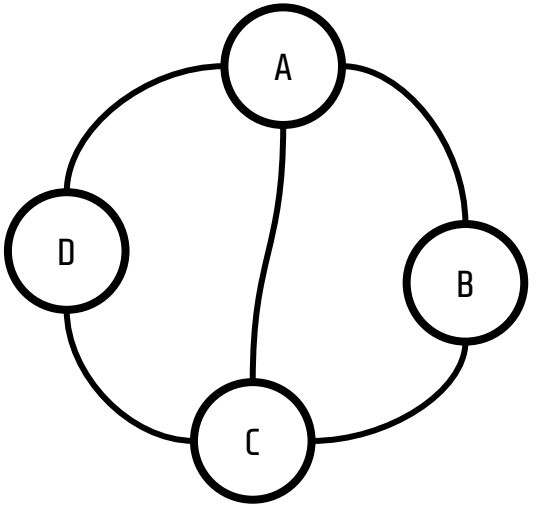
Par ordenado de aristas

```
G = (V, A)
V = {1, 2, 3}
A = {{1,1}, {1,2}, {2,3},
     {3,2}, {1,3}}
```



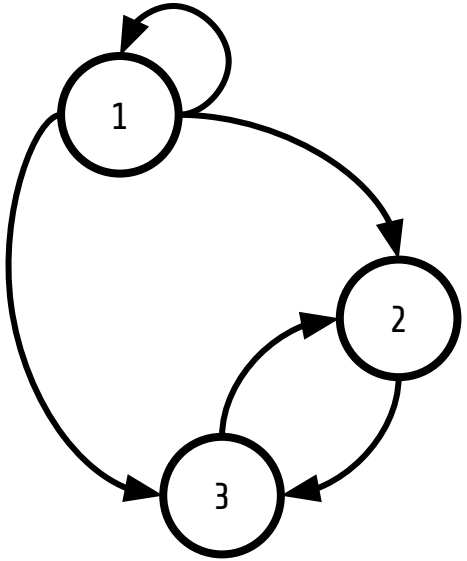
Pares de aristas

```
G = (V, A)
V = {A, B, C, D}
A = { {A, B}, {B, A},
      {B, C}, {C, B},
      {A, C}, {C, A},
      {A, D}, {D, A},
      {D, C}, {C, D}
    }
```

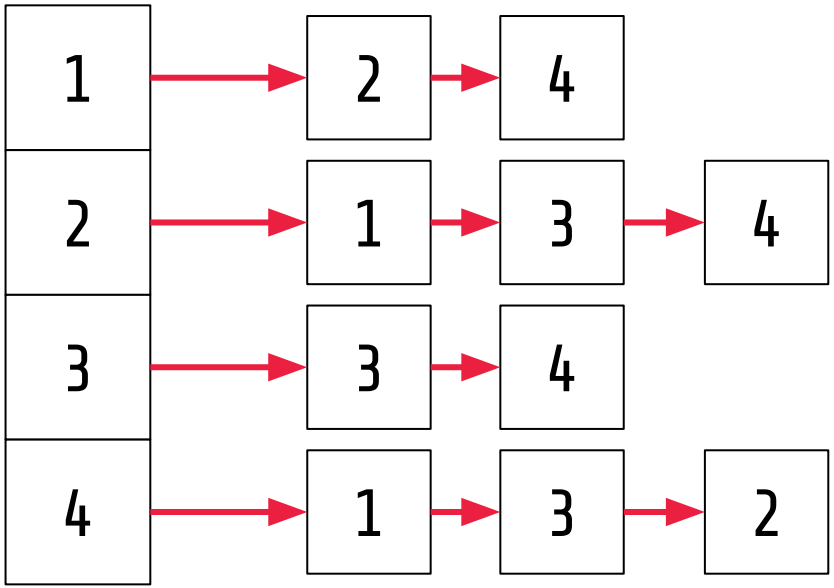


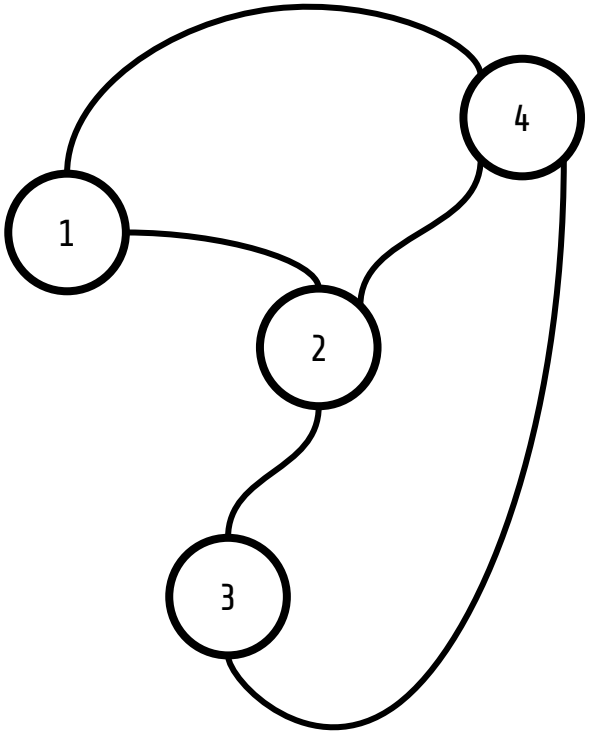
Matriz de adjacencia

	1	2	3
1	1	1	1
2	0	0	1
3	0	1	0



Lista de adjacencia





Algoritmos de grafos

https://link.springer.com/book/10.1007/978-3-319-73235-0

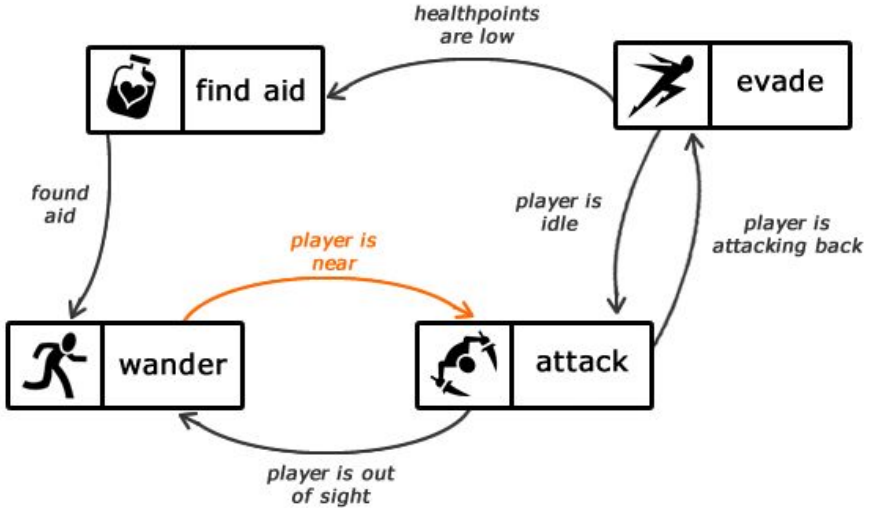
Complejidades

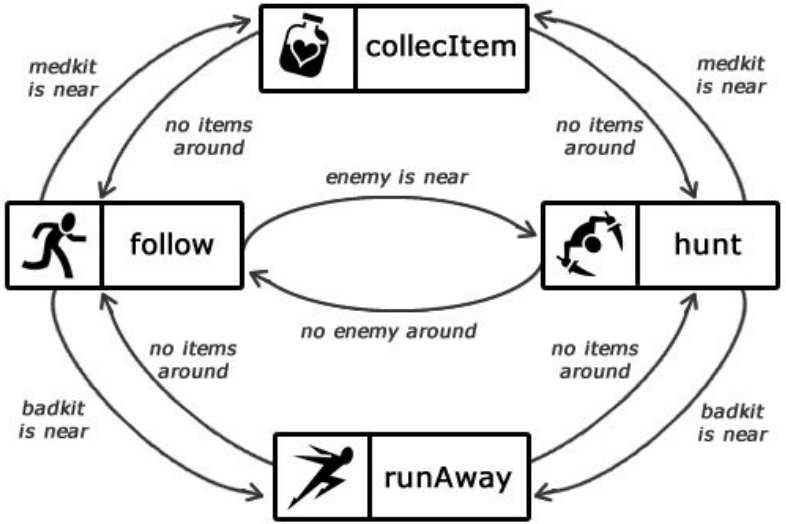
Operación	Matriz de adyacencia	Lista de adyacencia
Verificar adyacencia	O(1)	O(grado del vértice)
Encontrar vecinos	O(V)	O(grado del vértice)
Agregar vértice	$O(V^2)$	O(1)
Agregar arista	O(1)	O(1)
Eliminar vértice	$O(V^2)$	O(V+E)
Eliminar arista	O(1)	O(grado del vértice)

Distancia entre nodos

Aplicaciones

Máquinas de estados

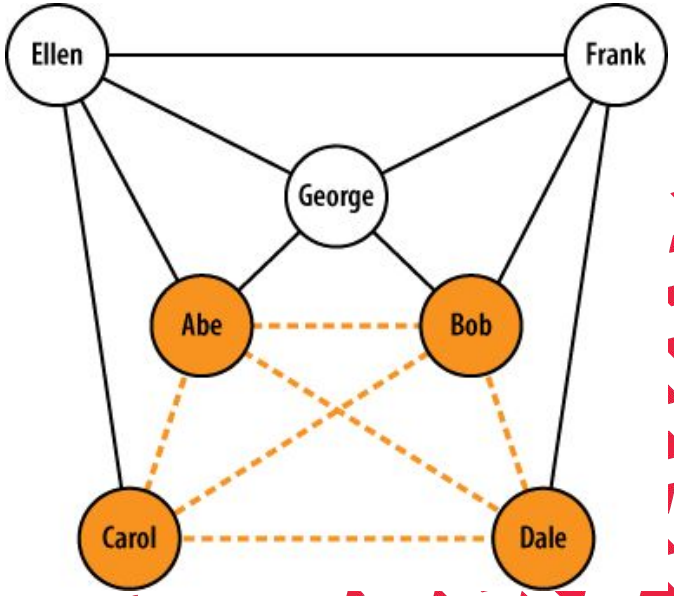




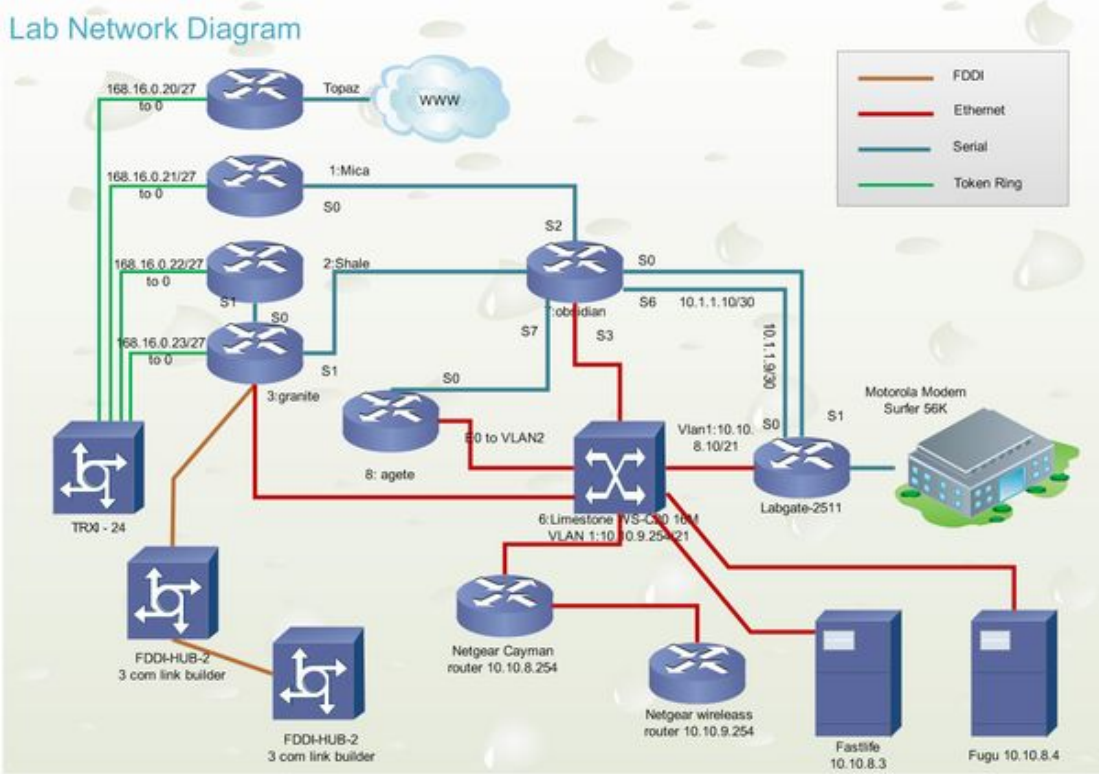
https://en.wikipedia.org/wiki/Finite-state_machine

Grafos sociales

Quien conoce a quien



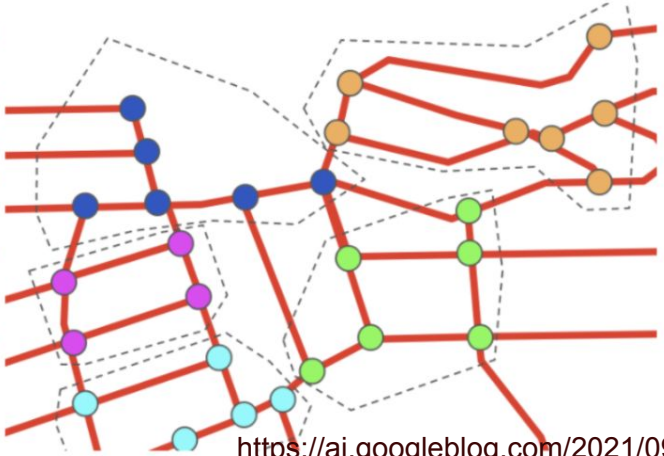
Modelado de redes

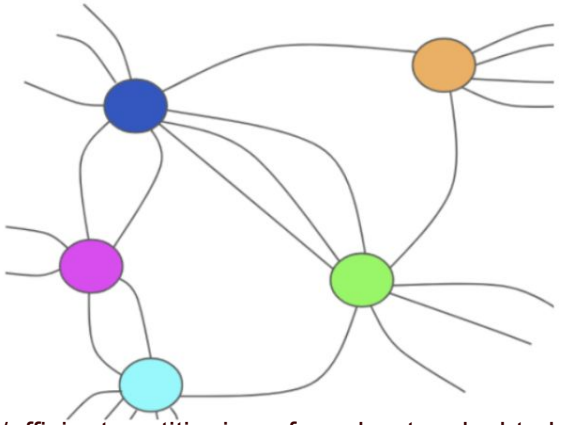


UNRN | Universidad Nacional de Río Negro

https://link.springer.com/book/10.1007/978-1-4471-5173-9

Tráfico y conexiones de caminos





https://ai.googleblog.com/2021/09/efficient-partitioning-of-road-networks.html


¿Preguntas?

Estructuras avanzadas

XX
II
2025

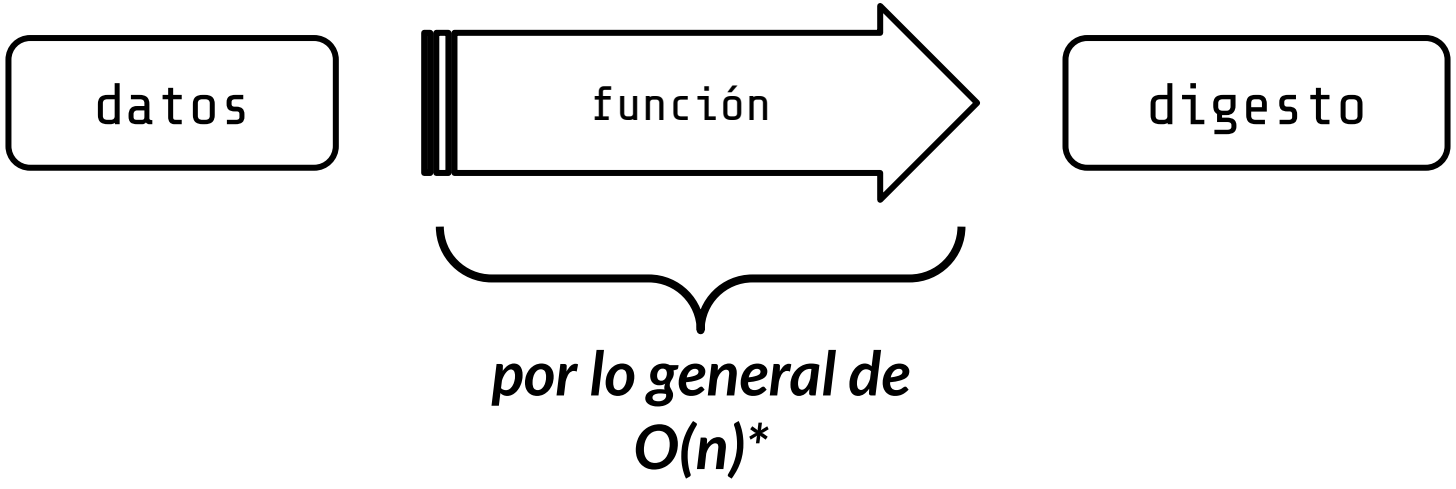
UNRN
Universidad Nacional
de **Río Negro**

r27

Diccionarios

Introducción a las funciones de extracto

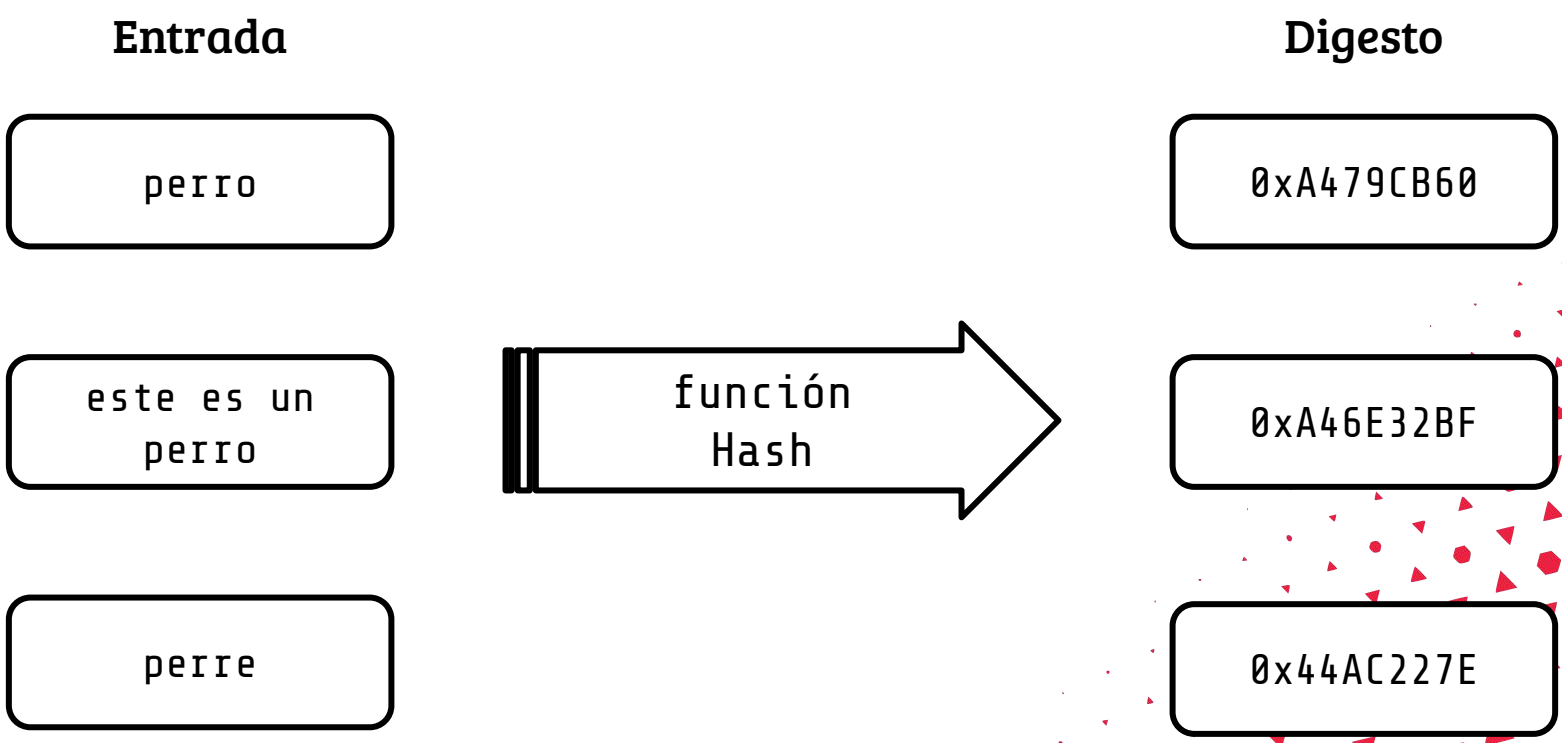
hash/extract/extracto/digesto



La función nos resume los datos en un digesto

Lo que entra, sale con ancho fijo

(como secuencia de bits)


Entrada
perro
este es un perro
perre
función Hash
Digesto
0xA479CB60
0xA46E32BF
0x44AC227E

El contenido original no es reconstruible

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

*hay muchos mas

https://emn178.github.io/online-tools/crc32.html

¡Pueden existir dos datos con el mismo valor!

Esto es llamado **colisión**

Aplicaciones

Control de versiones

git hace uso extensivo

Blockchain

Los datos se guardan con el digesto de lo anterior

Monitoreo de salud de datos y archivos

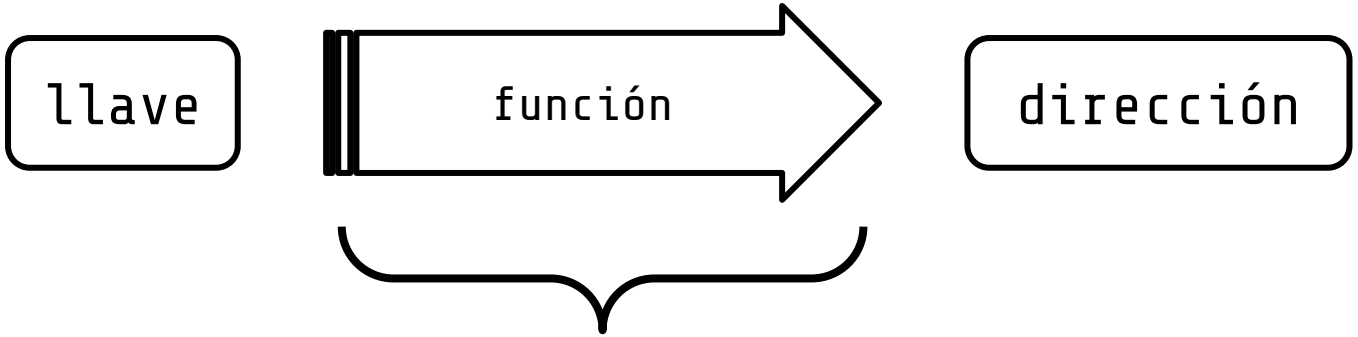
(cualquier cambio, altera el digesto)

Diccionarios*
Tablas de Hash

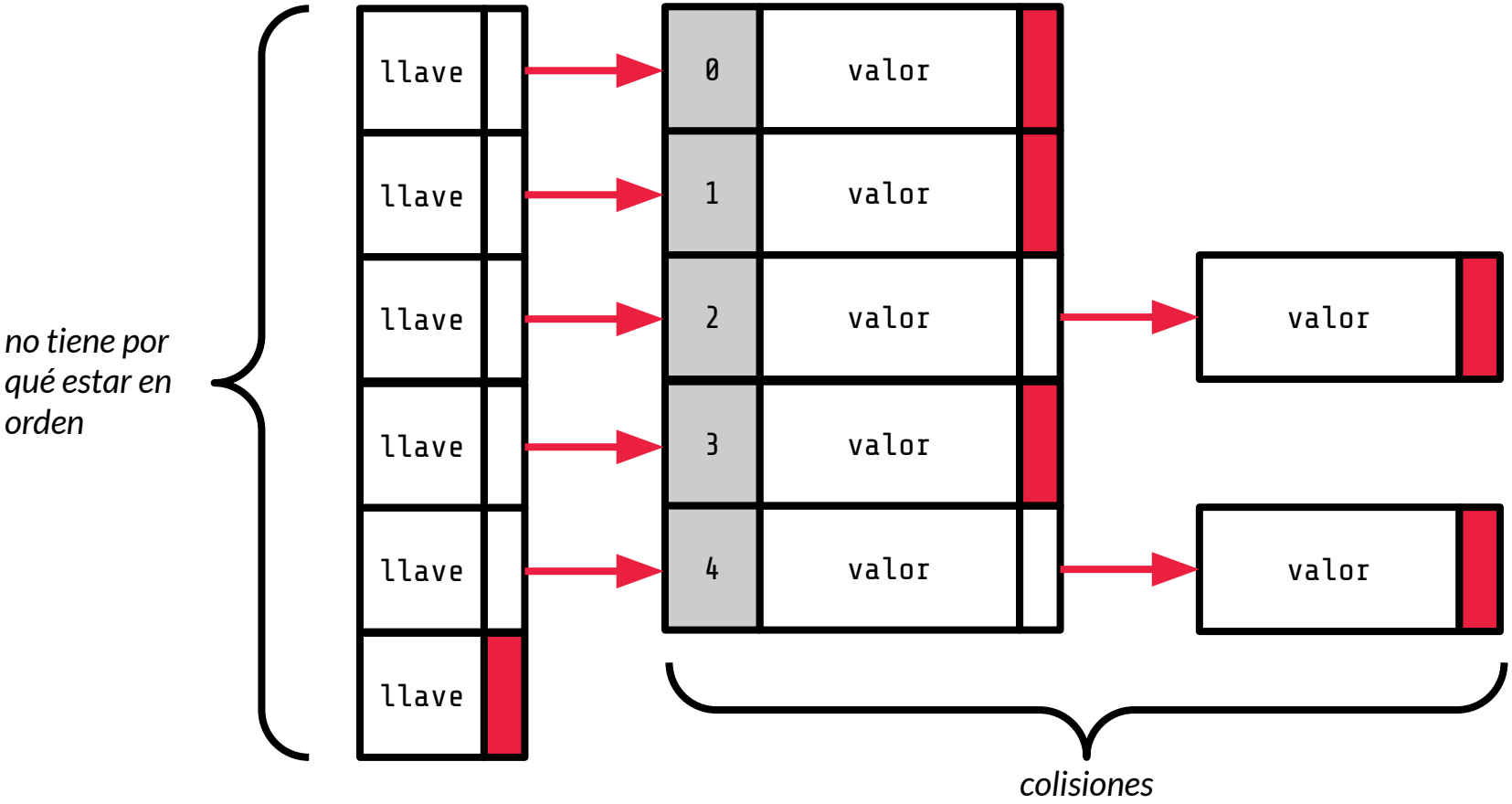
*conceptualmente

{ 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



Una implementación simple (y simplificada)

{

Un ejemplo simple

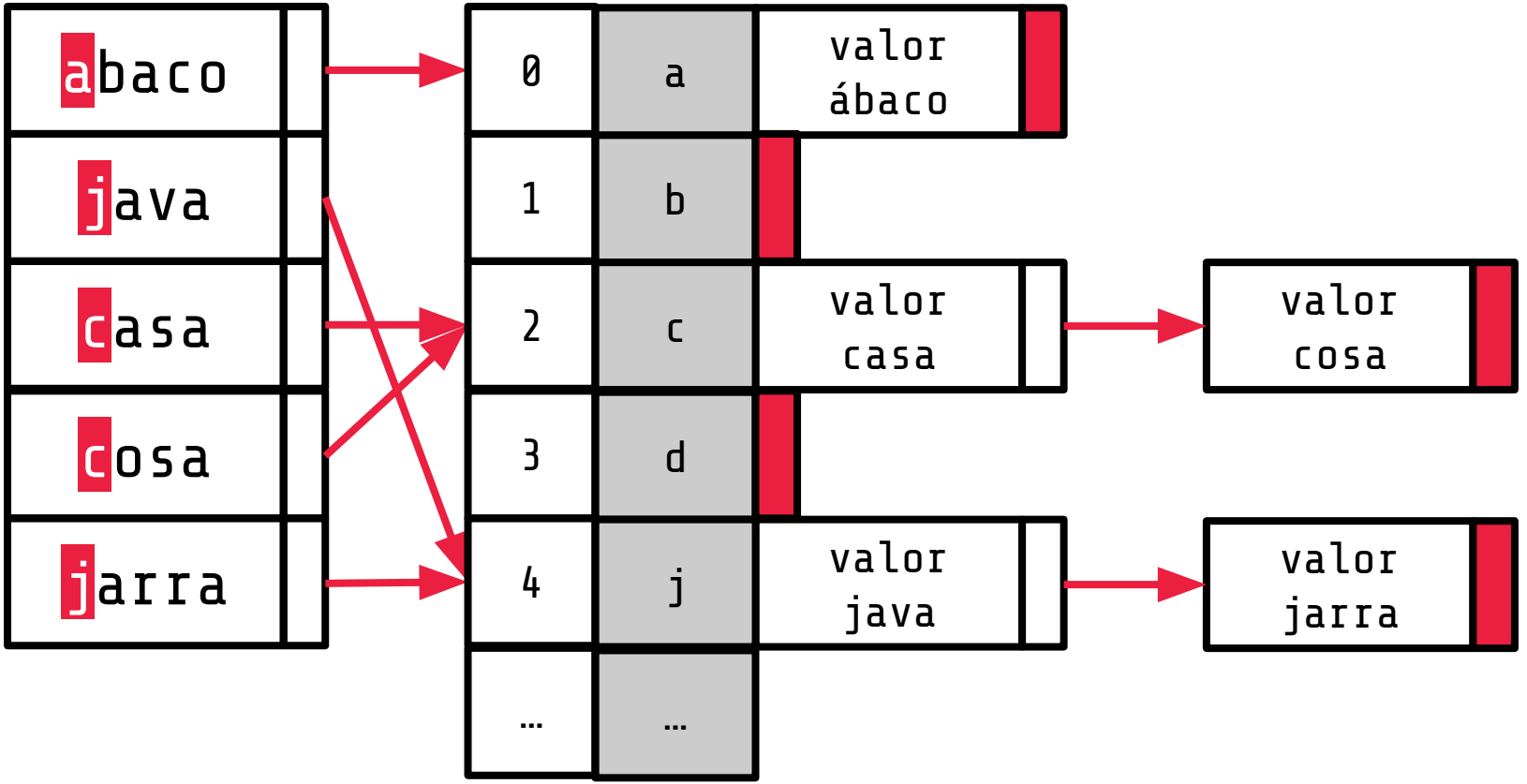
}

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

Una implementación simple (demasiado)

```
static int hashimple(String cadena){
    char llave = cadena.charAt(0);
    return llave;
}
```

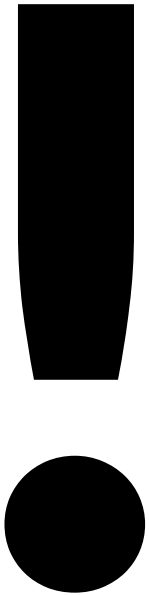
*Tiene sus propios problemas ser **tan simple***



Usamos la primera letra como llave

Un ejemplo más práctico

compactando hashCode



Siendo que `int hashCode()`

Tiene unos 4.294.967.296 valores disponibles como `int`

Lo podemos dividir para que ‘entre’ en un espacio más reducido, dividiéndolo por los espacios disponibles.

Las colisiones *se pueden* resolver como una lista enlazada

Pero mejor sí usamos árbol binario de búsqueda

Búsqueda
Inserción
O(n)
Ω(1)

En la implementación simple

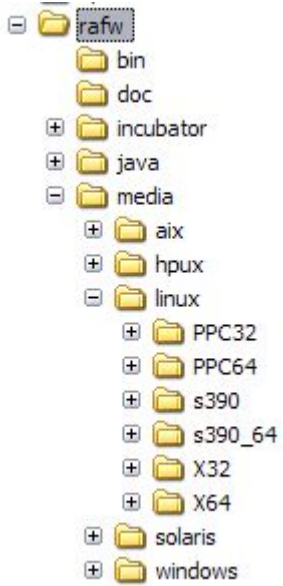
En espacio
O(n)

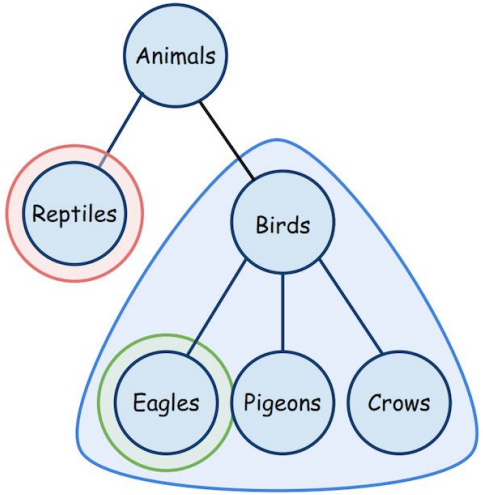
Es necesario reservar lugar para cada elemento

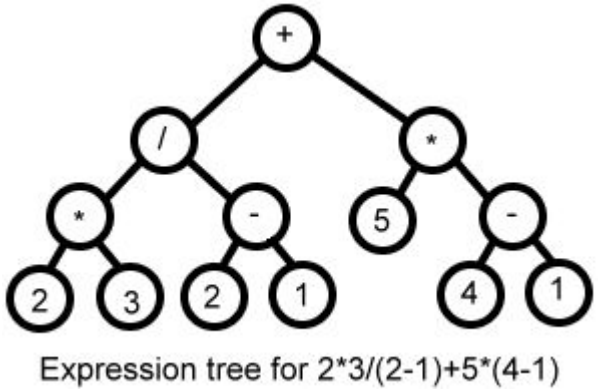

¿Preguntas?

Árboles

Estructuras de datos jerárquicas



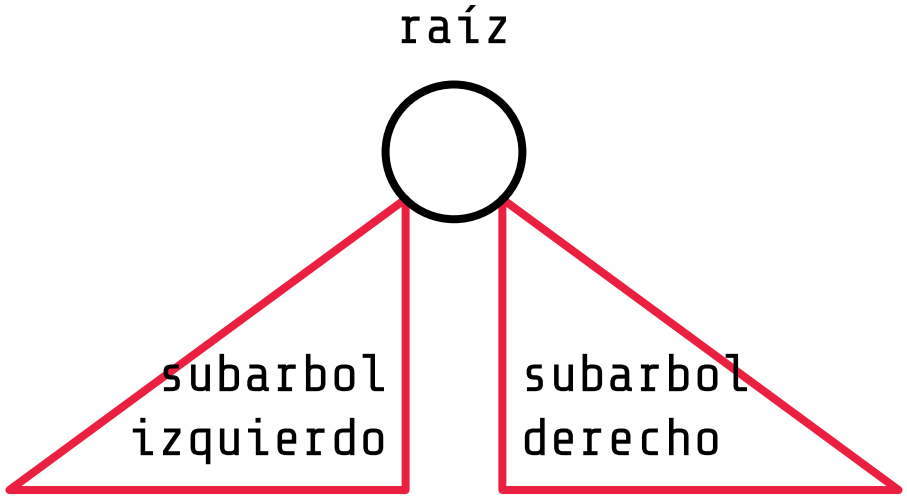




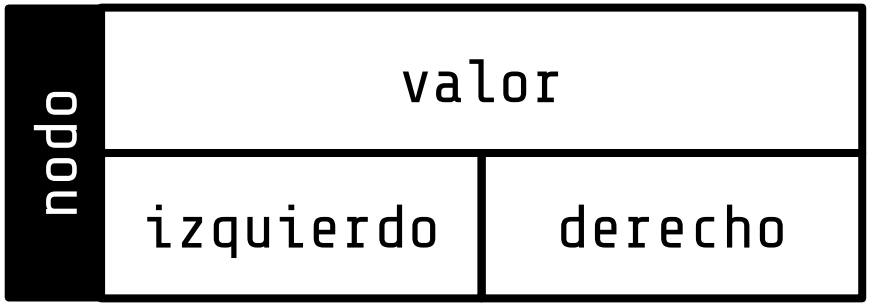
Expression tree for 2*3/(2-1)+5*(4-1)

Árbol binario

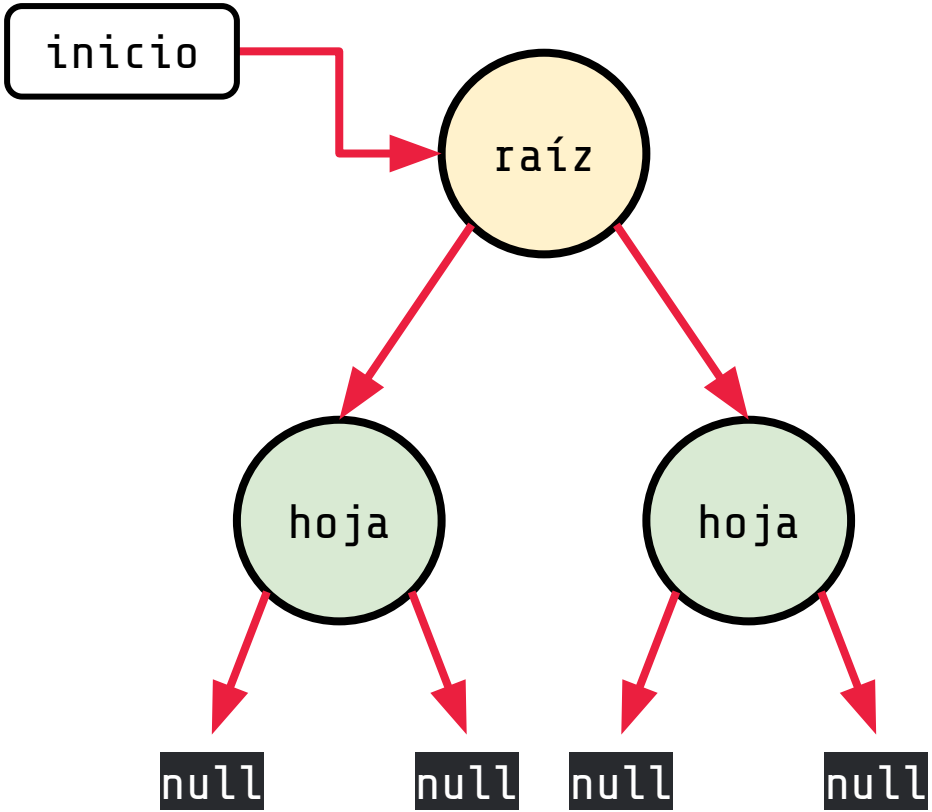
Definido recursivamente



En donde cada nodo está compuesto de



Componentes y estructura de un árbol



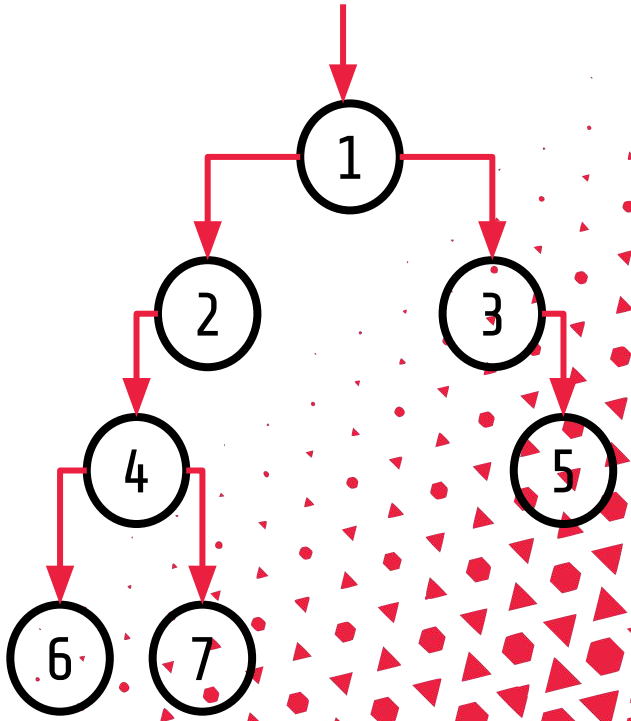
Conceptos

Arco

La conexión entre dos nodos

Por ejemplo

(1,2)
(4,7)



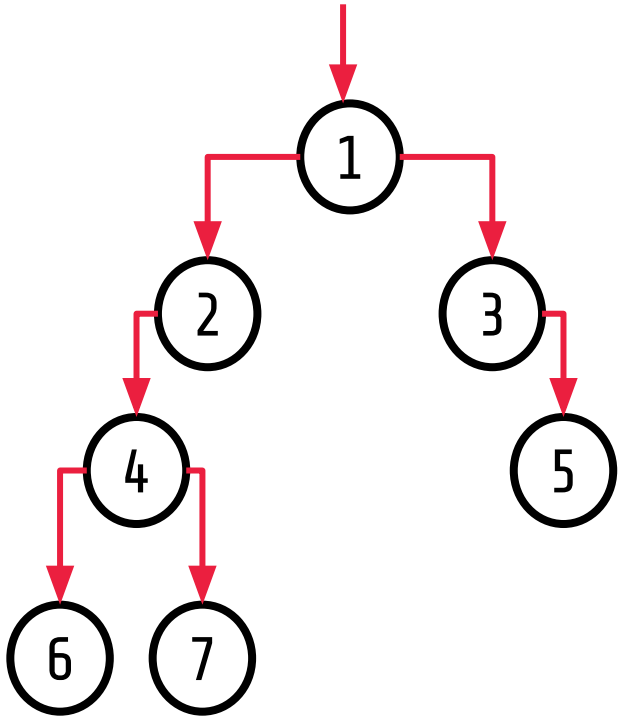
Camino

Una secuencia entre nodos

Por ejemplo

(1,2,4,7)

(1,3,5)

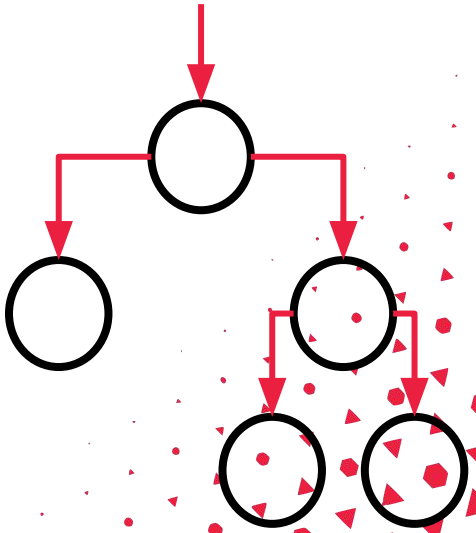


***más sobre esto un poco más adelante**

Características

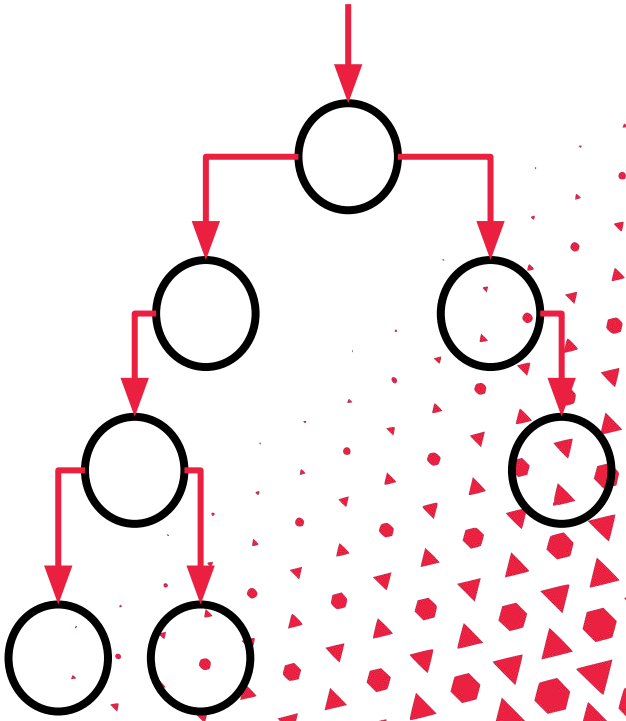
Árbol ‘propio’

Los nodos tienen cero o dos hijos



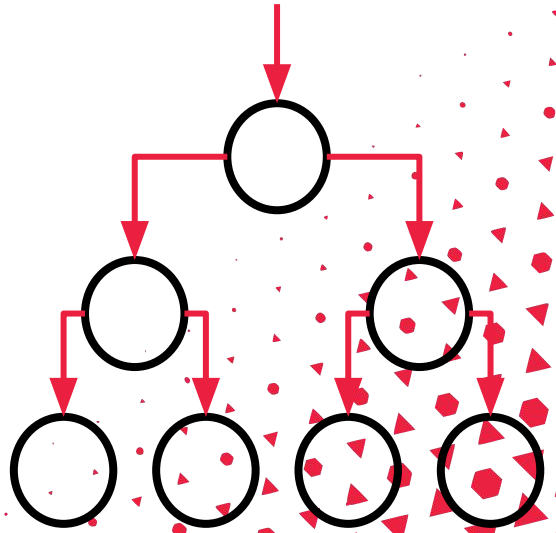
Árbol ‘impropio’

Los nodos pueden no tener ambos hijos



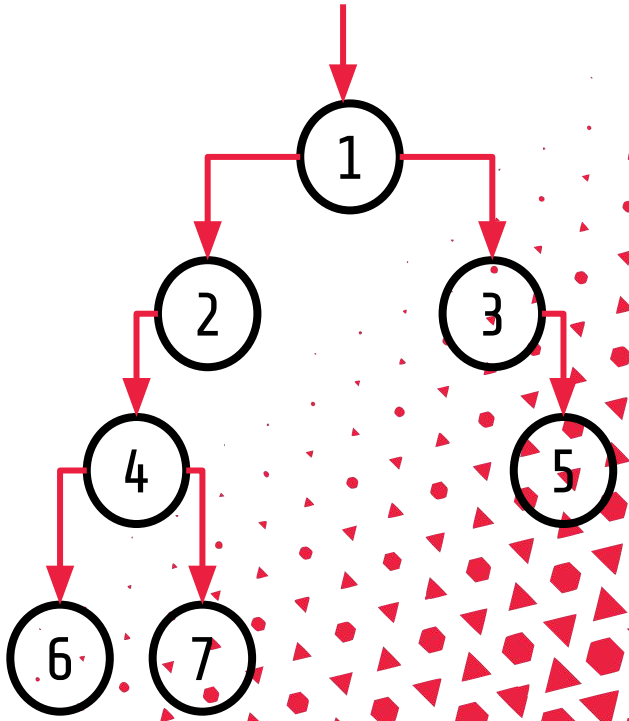
Árbol ‘lleno’

Todos los nodos tienen dos hijos (excepto las hojas)



Peso

Como la cantidad total de nodos

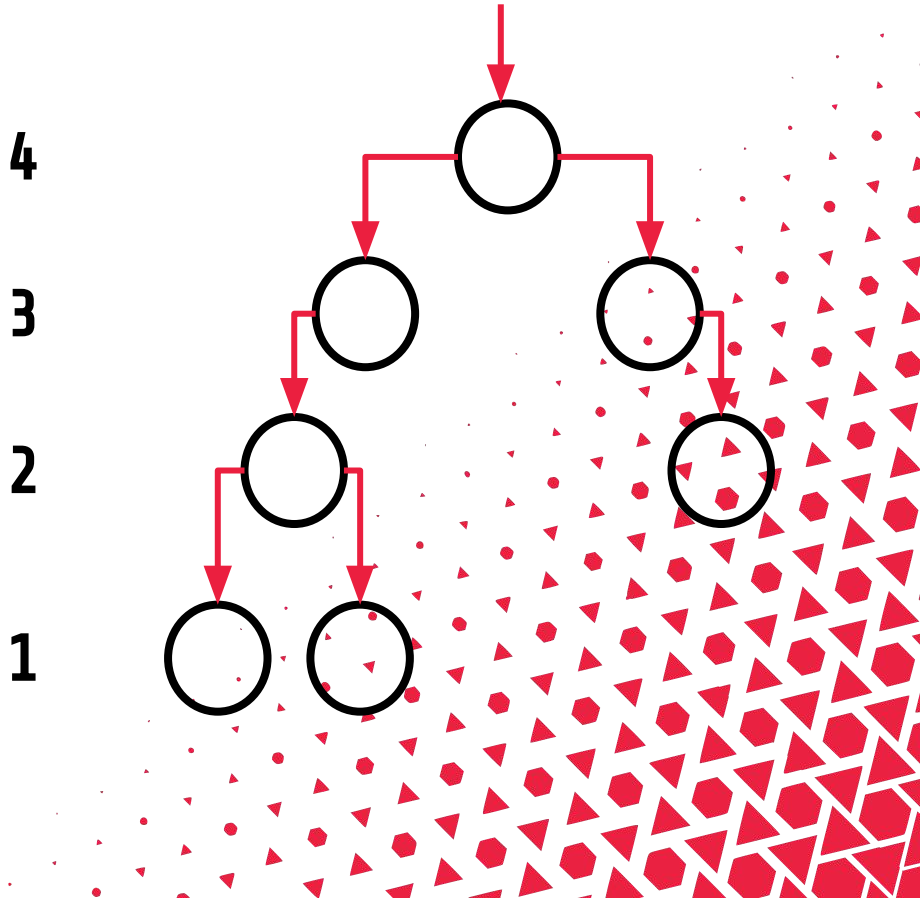


Cálculo recursivo

```
public static int peso(Arbol raiz){
    if (raiz == null){
        return 0;
    }
    int izquierda = peso(root.left);
    int derecha = peso(root.right);
    return 1 + izquierda + derecha;
}
```

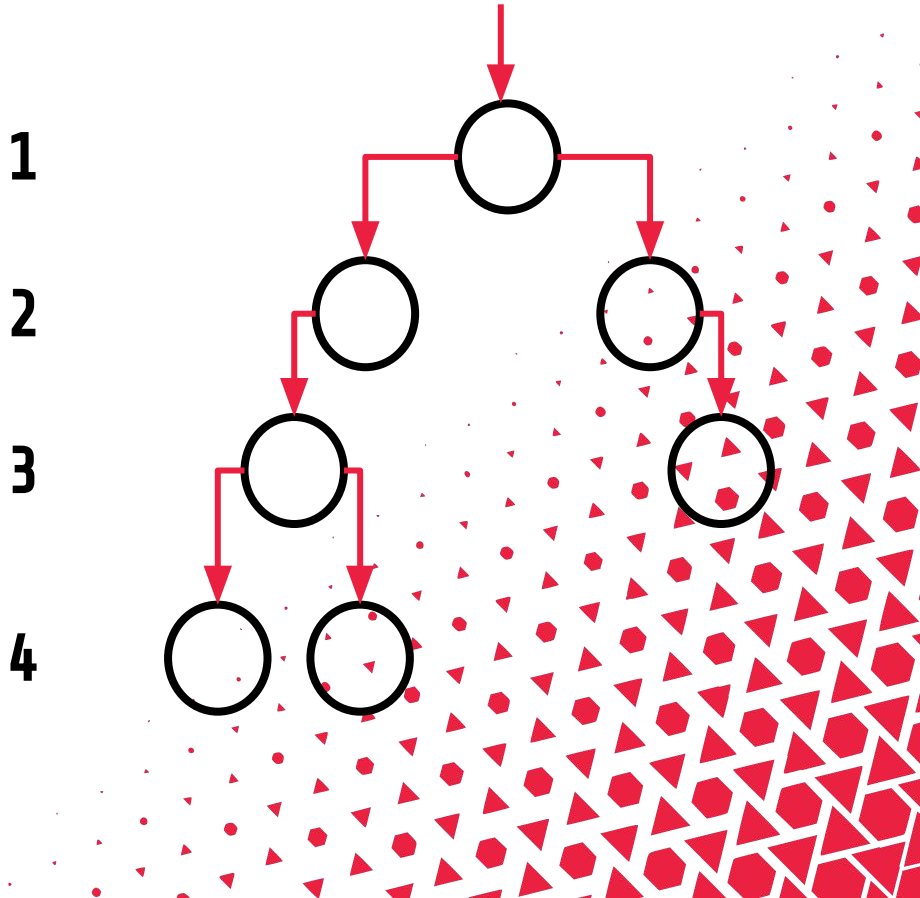
Altura

Cuantos niveles tiene el árbol



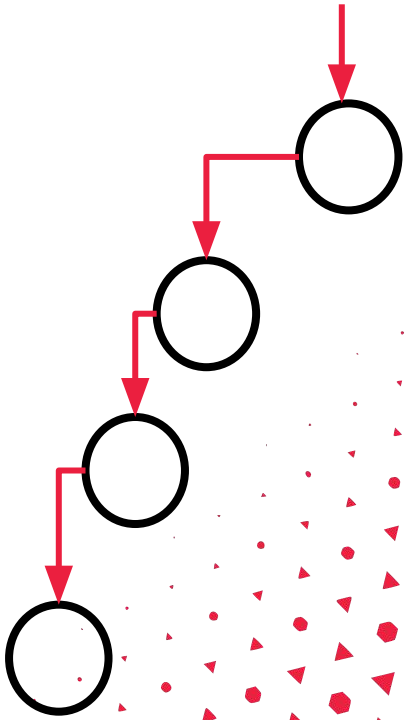
Profundidad

El camino mas largo hacia abajo



Degeneración

A lista enlazada



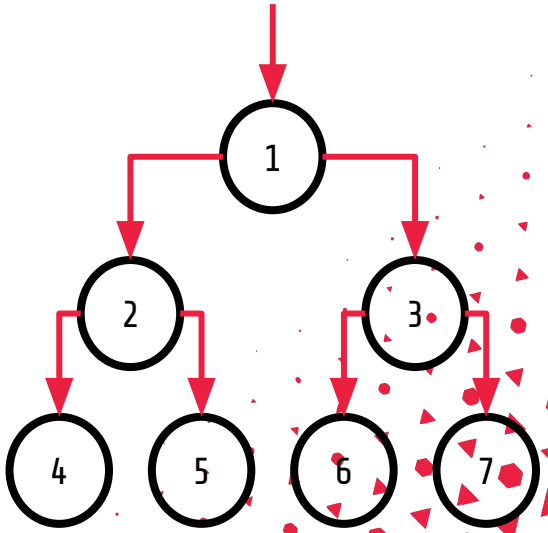
recorridos

En orden

1. vamos al izquierdo
2. se *visita* el nodo
3. vamos al derecho

En orden

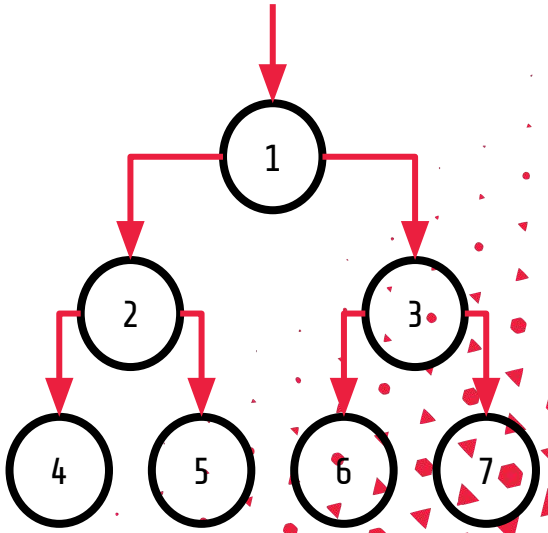
1 vamos al izquierdo
2 se *visita* el nodo
3 vamos al derecho

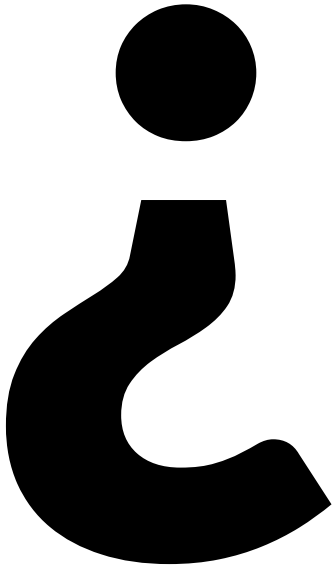


En orden

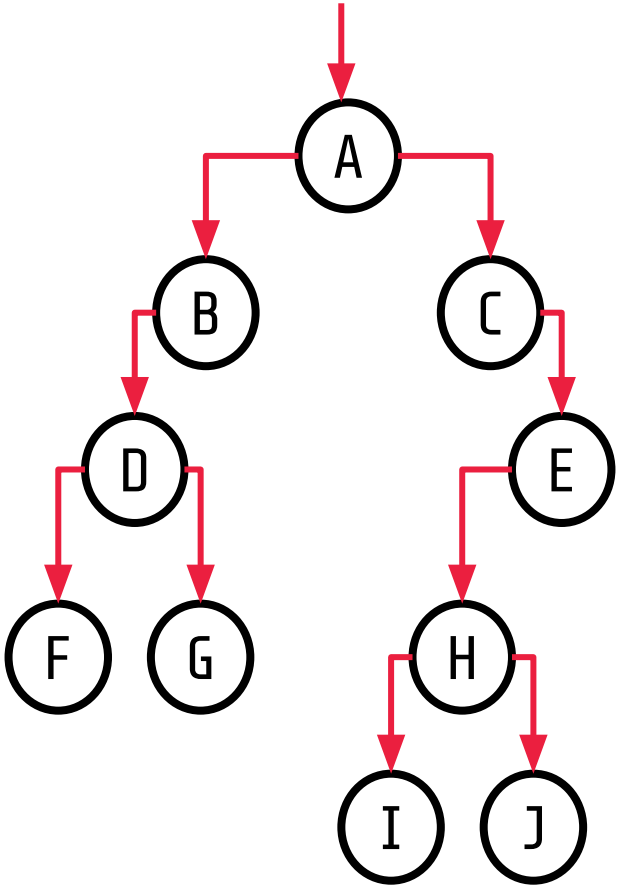
1 vamos al izquierdo
2 se *visita* el nodo
3 vamos al derecho

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





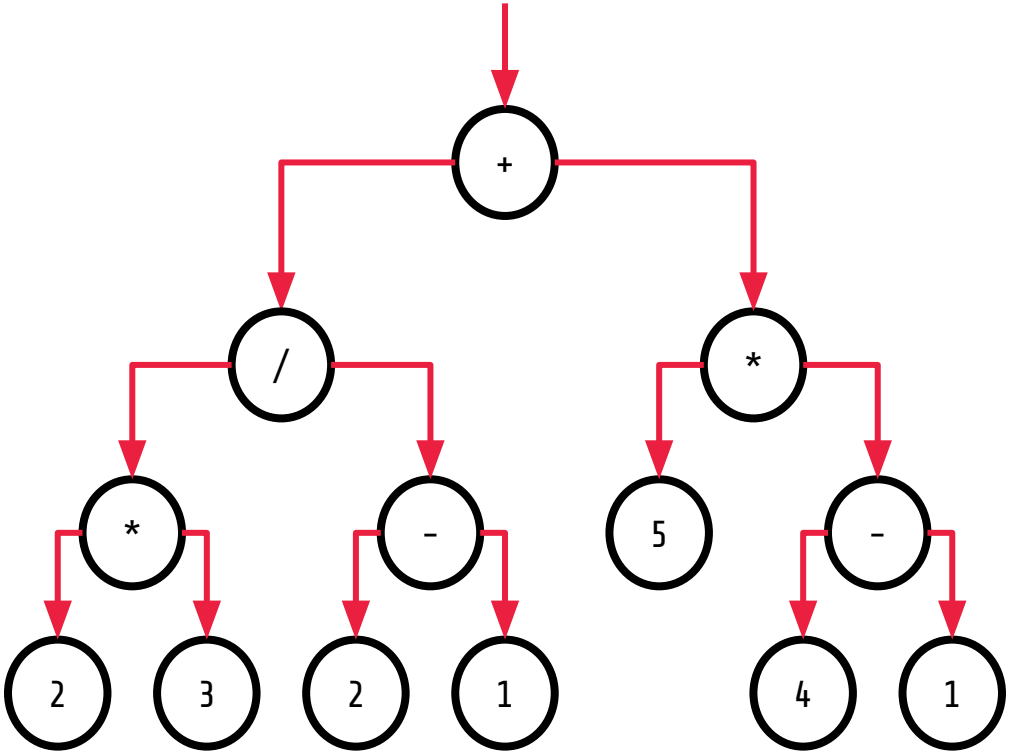
Y en el árbol más complicado


A
B
C
D
E
F
G
H
I
J

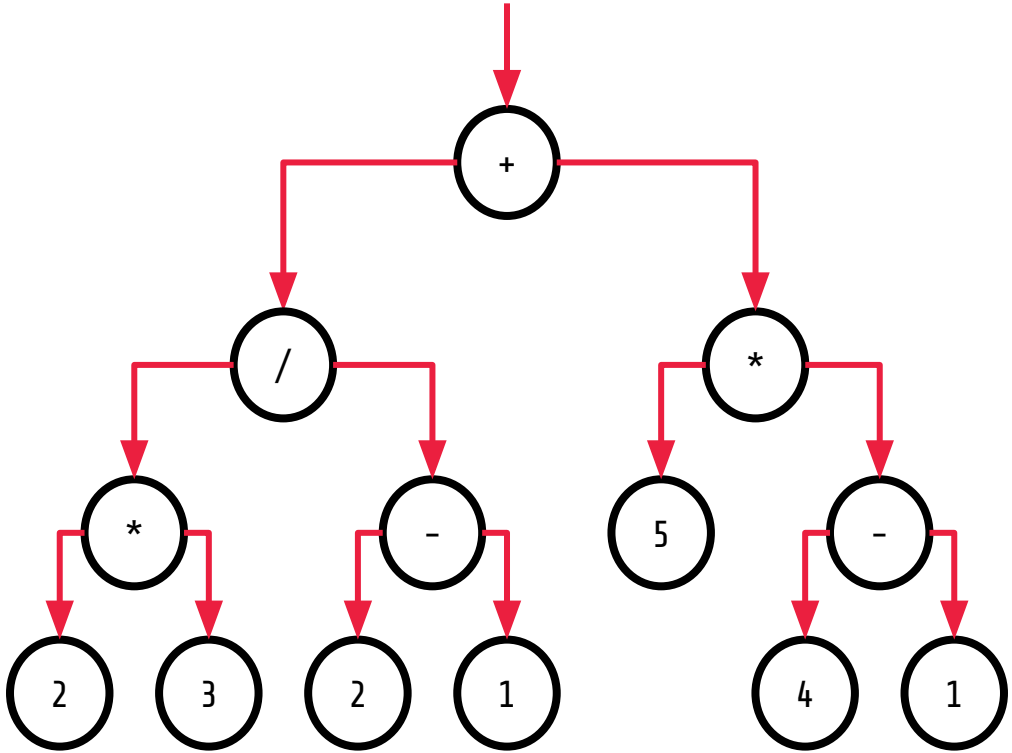

¿Preguntas?

Podemos también, sacar una aplicación

¿Cuál es el recorrido enorden de este árbol?



¿Cuál es el recorrido enorden de este árbol?



(2*3)/(2-1)+5*(4-1)

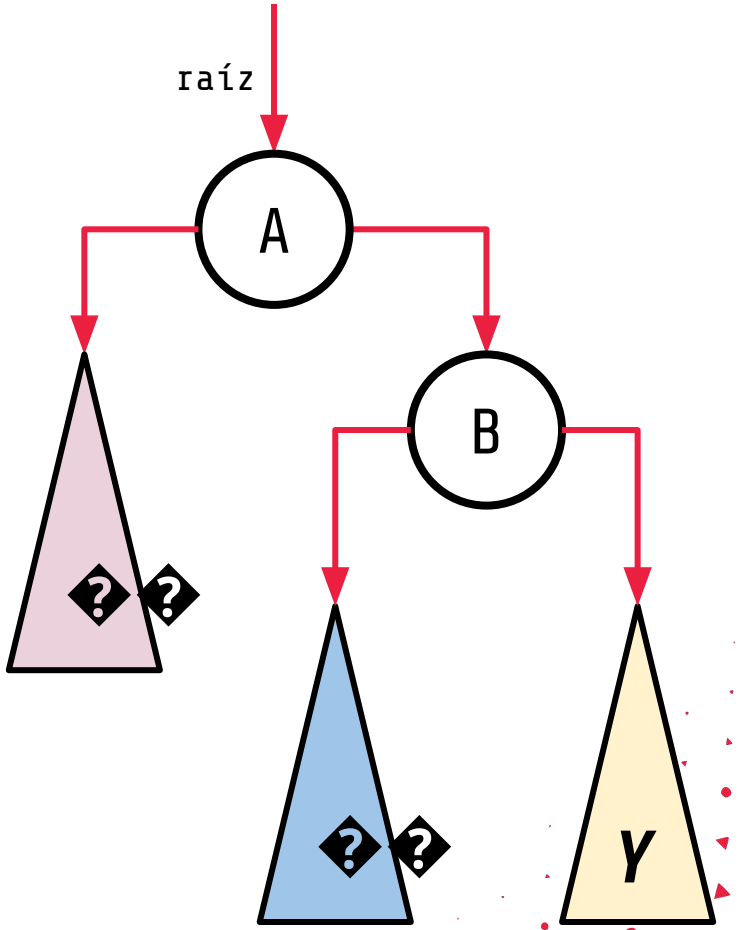

¿Preguntas?

Más operaciones con nodos

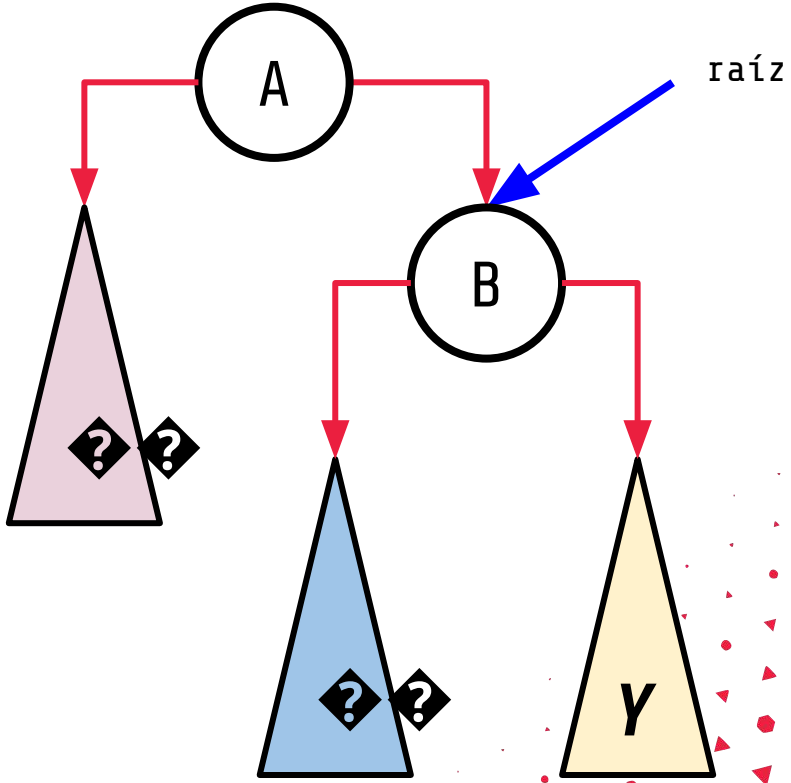
Rotación

https://people.ksp.sk/~kuko/gnarley-trees/Balance.html

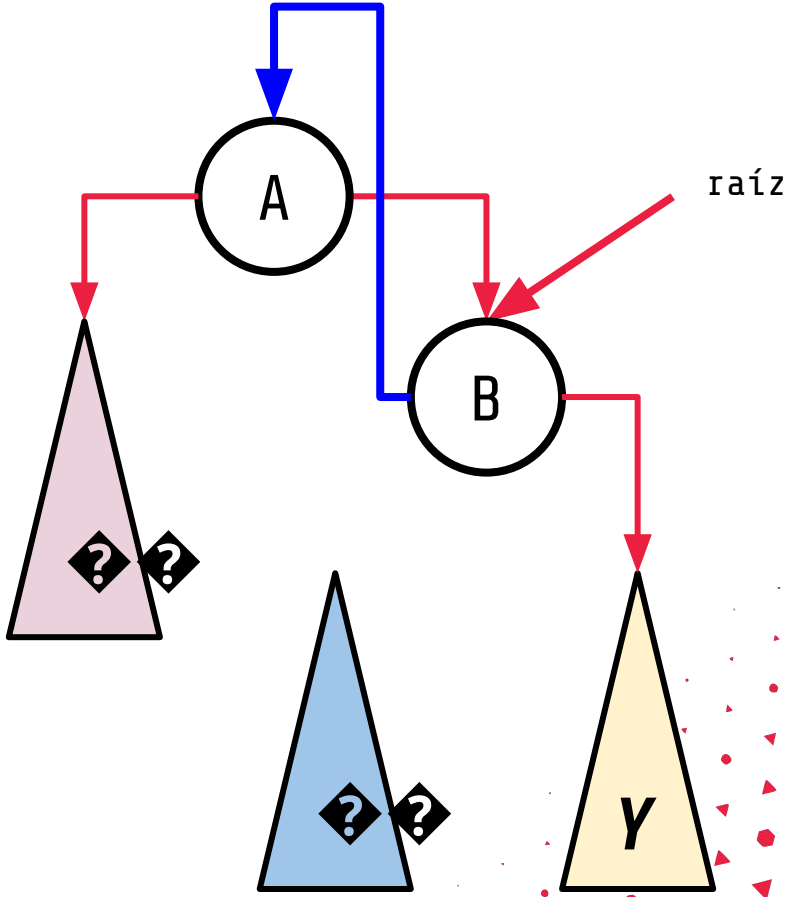
Rotación izquierda


raíz
A
B
?
?
?
?
Y

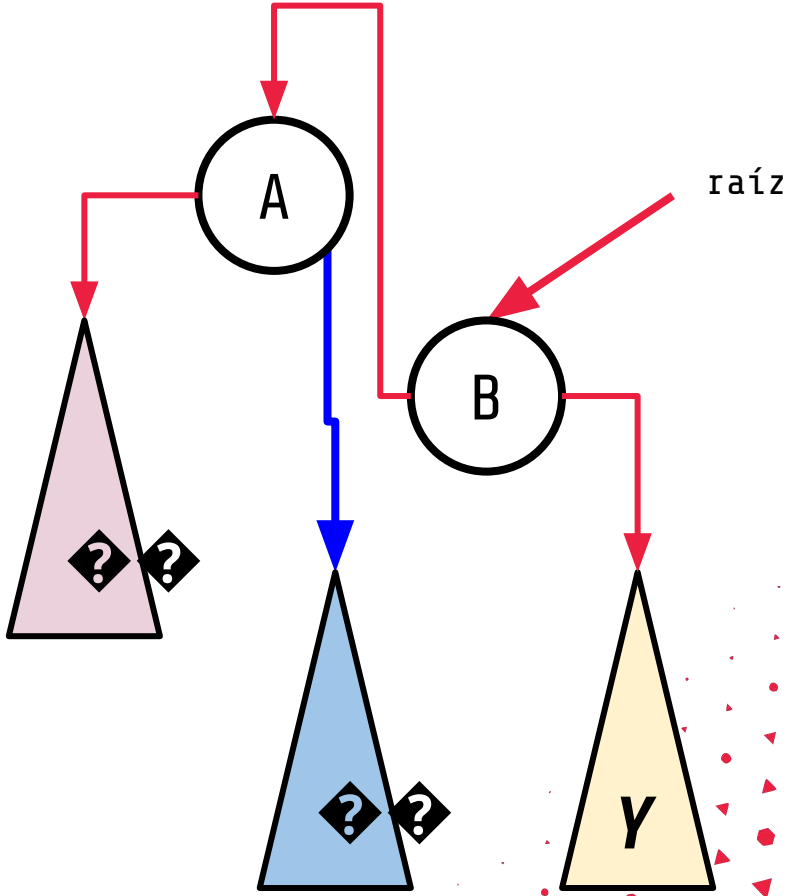
1. el nodo izquierdo ahora es la raíz

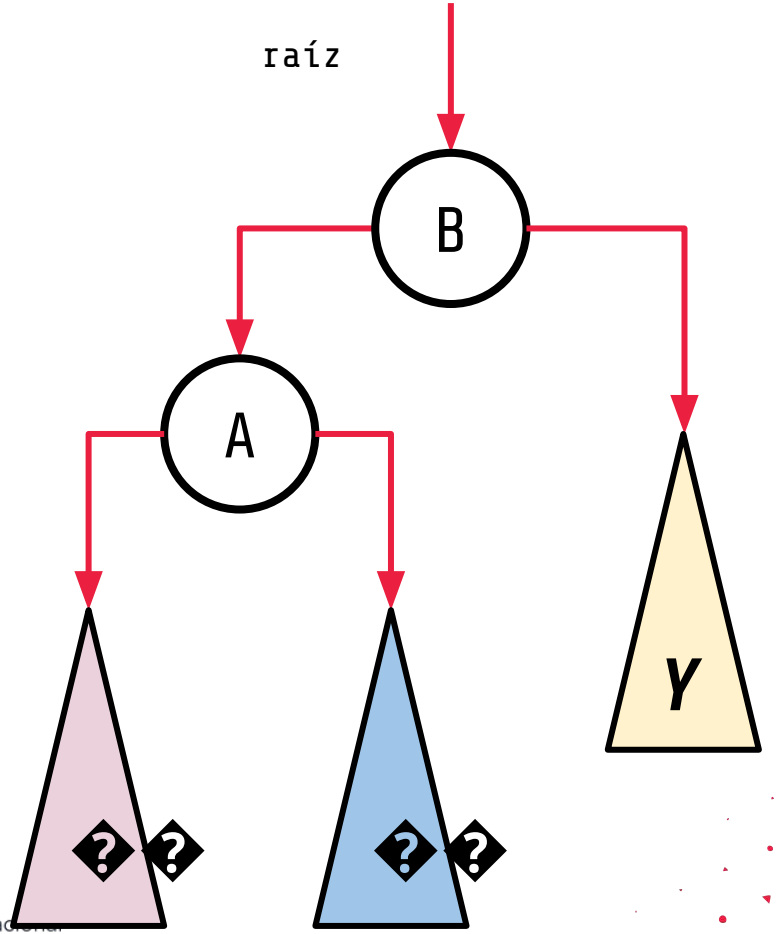


1. el nodo izquierdo ahora es la raíz
2. el nodo izquierdo apunta a la raíz anterior



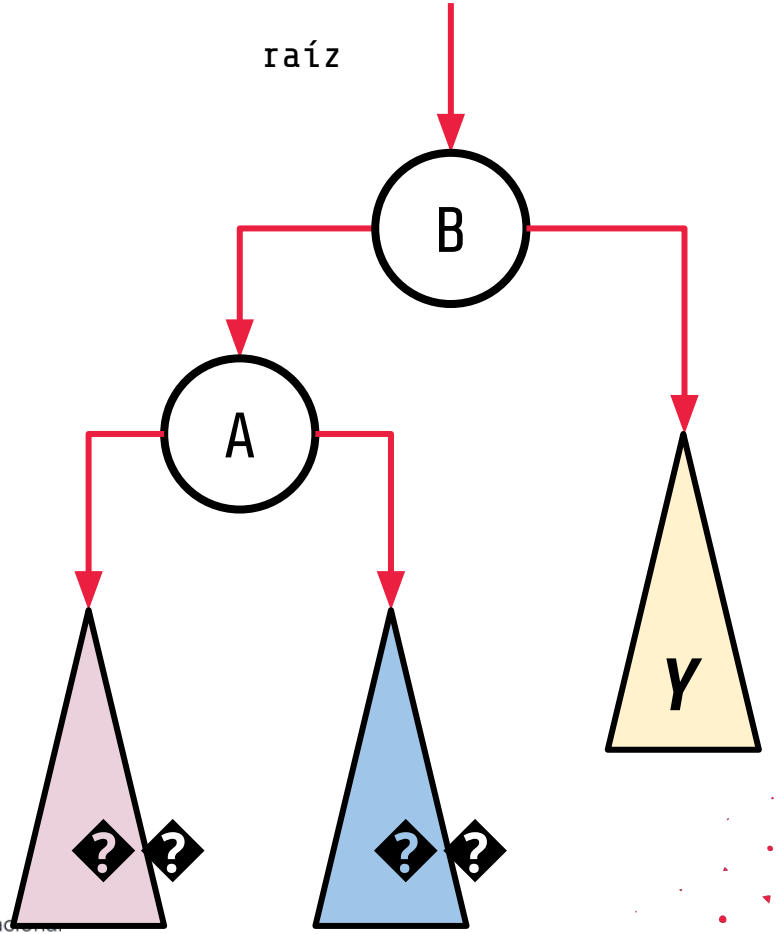
1. el nodo izquierdo ahora es la raíz
2. el nodo izquierdo apunta a la raíz anterior
3. la raíz anterior apunta al subarbol izquierdo

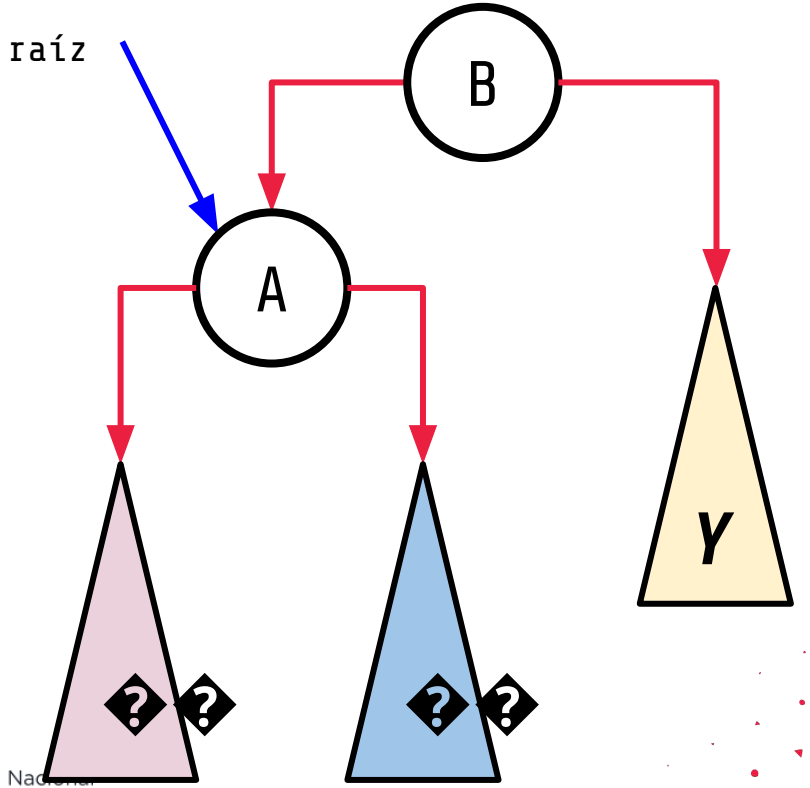


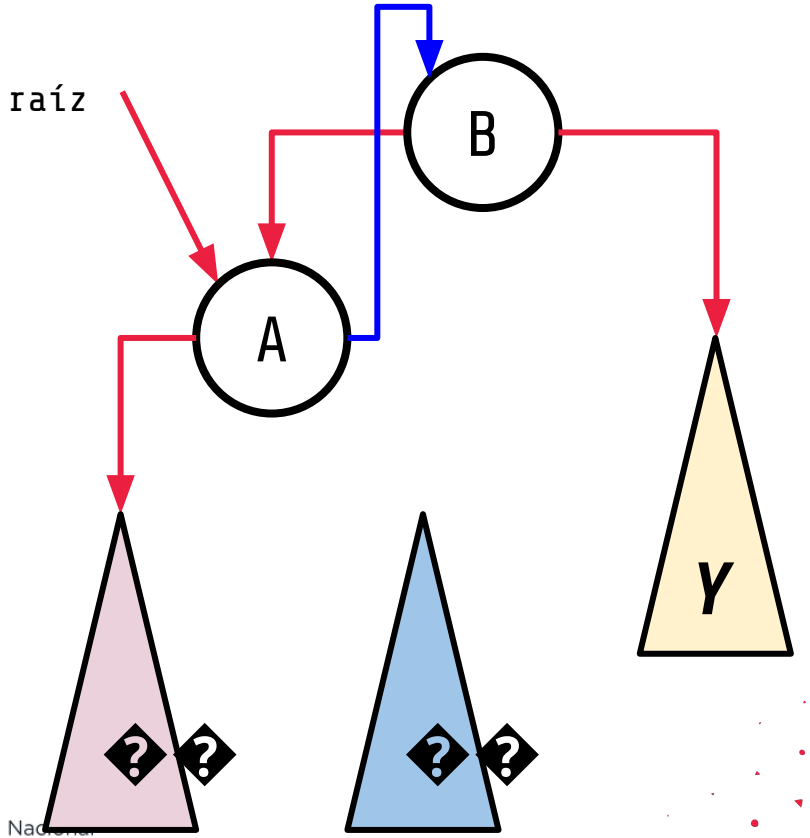

raíz
B
A
Y
?
?
?
?

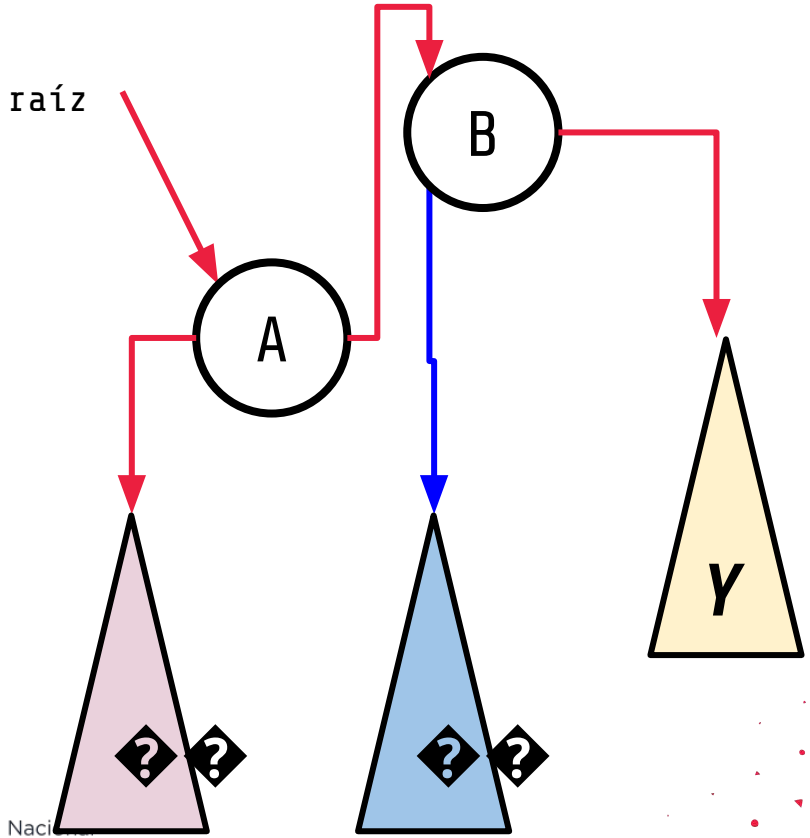

¿Preguntas?

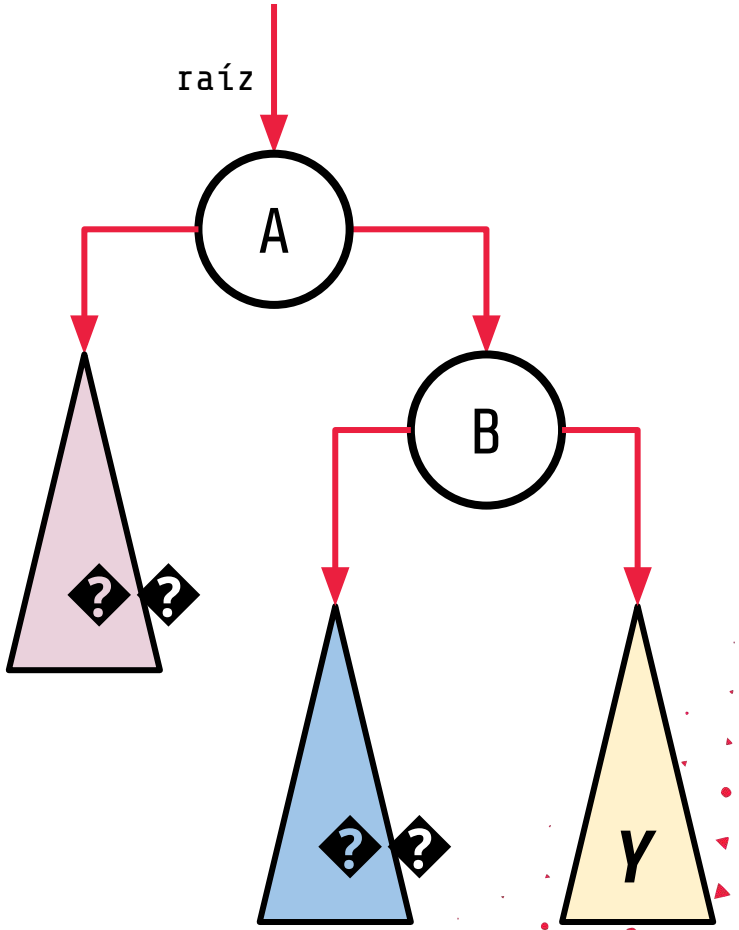
Y hacia la derecha


raíz
B
A
Y
?
?
?
?

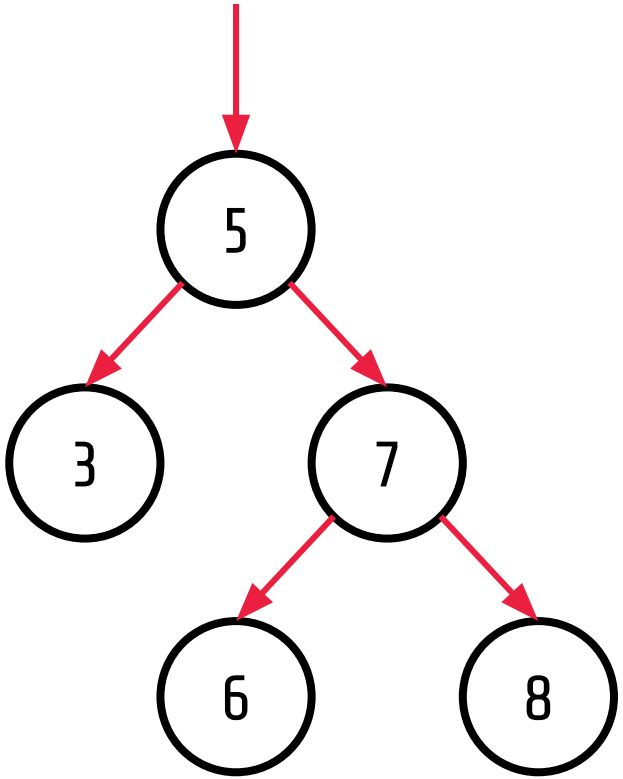

raíz
B
A
Y
?
?
?
?

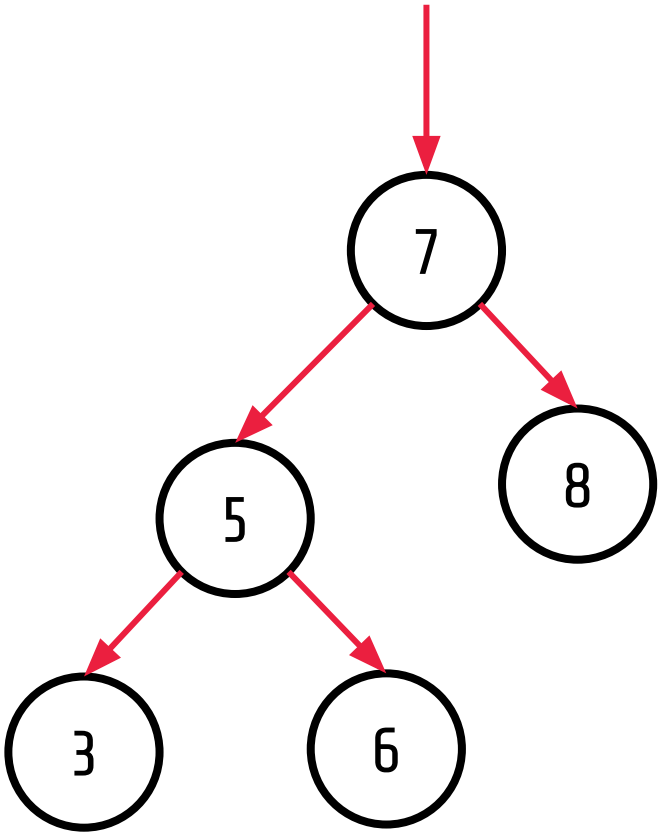

raíz
B
A
Y
?
?
?
?


raíz
B
A
Y
?
?
?
?


raíz
A
B
?
?
?
?
Y

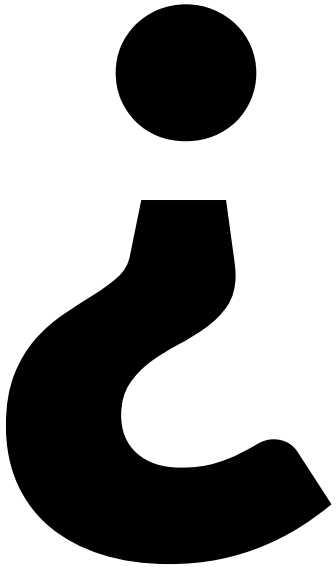
Un ejemplo mas concreto


5
3
7
6
8


7
5
8
3
6


ah

El recorrido en orden no cambia

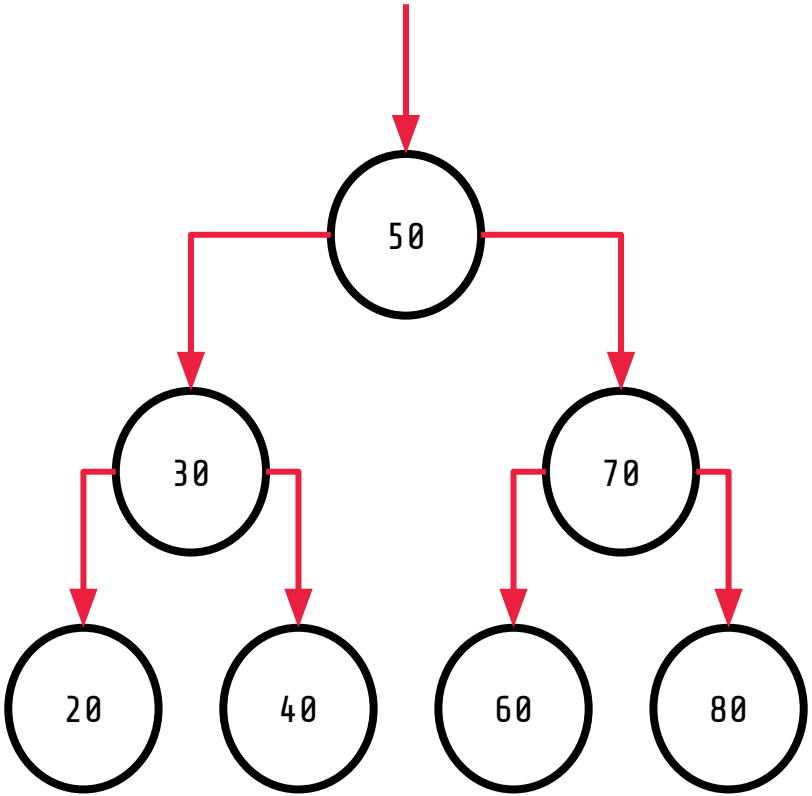


para qué se usa esta operación

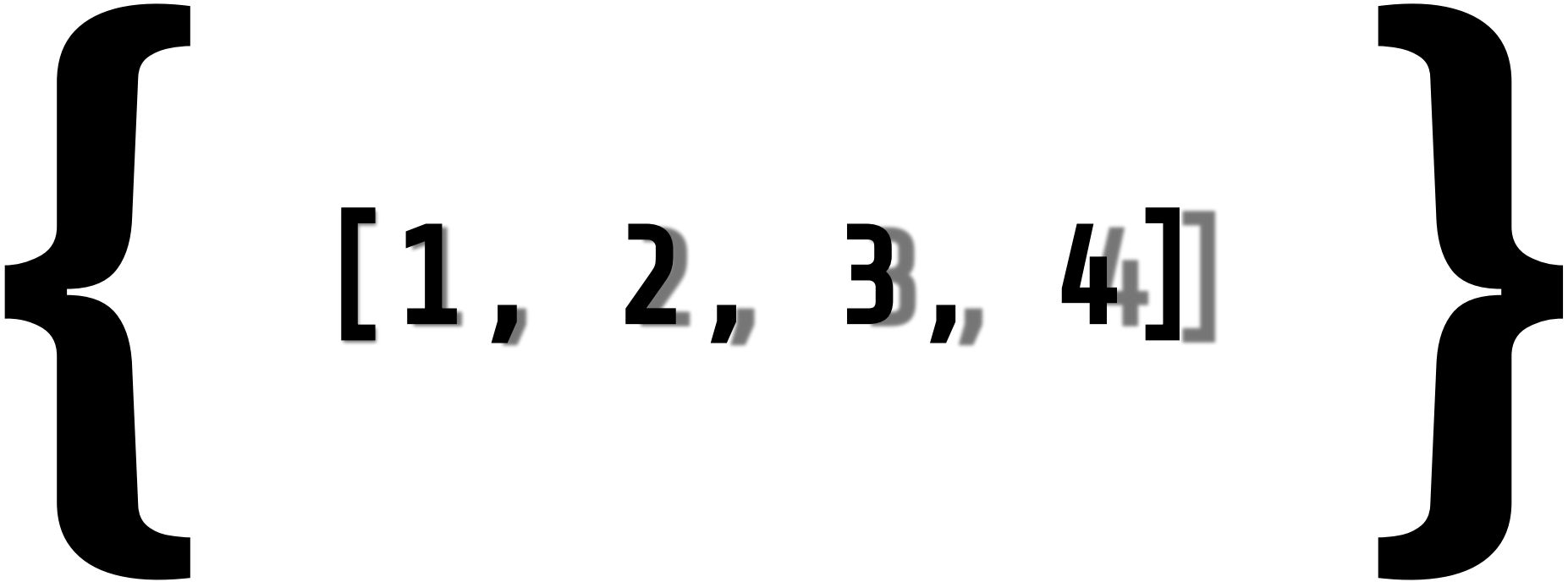
{ **Acomodan la carga**
afectando solo dos nodos }

Árbol binario de búsqueda

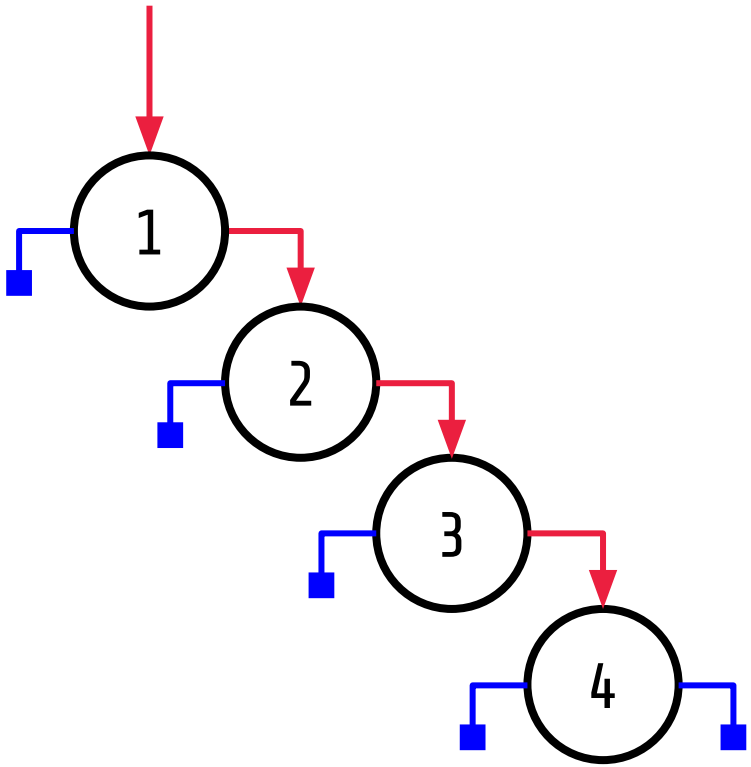
El menor a la izquierda y el mayor a la derecha



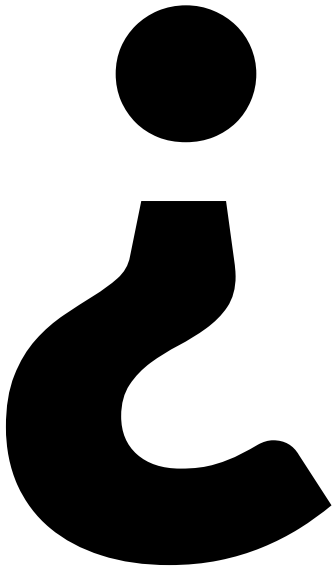
{ pero cuando la inserción es en orden... }


[1, 2, 3, 4]

Se ‘degradan’ a una lista enlazada



Se degeneran a una lista enlazada



Que podemos hacer

Es aquí donde

entran los Árboles balanceados

Arboles AVL

Adelson-Velskii, Landis (1962)

Cada nodo conserva el balance de los subárboles

Otros árboles balanceados

Arboles Red-Black

Leonidas Guibas - Robert Sedgewick (1978)

UNRN | Universidad Nacional de Río Negro

B-tree

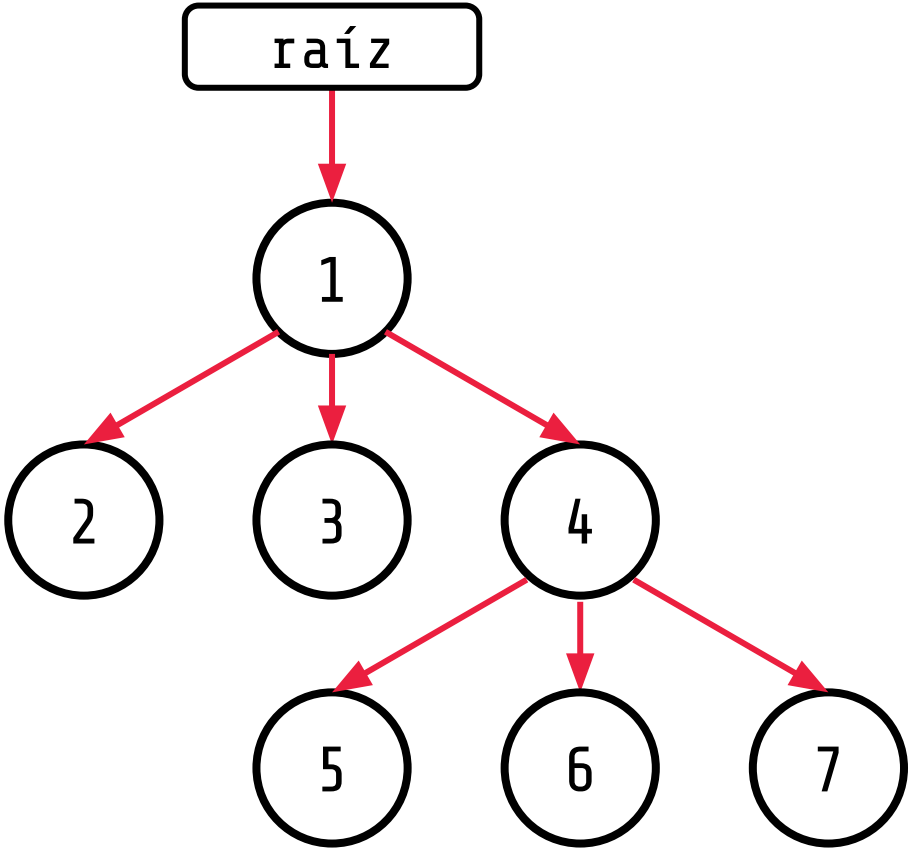
Rudolf Bayer - Edward M. McCreight (1970)

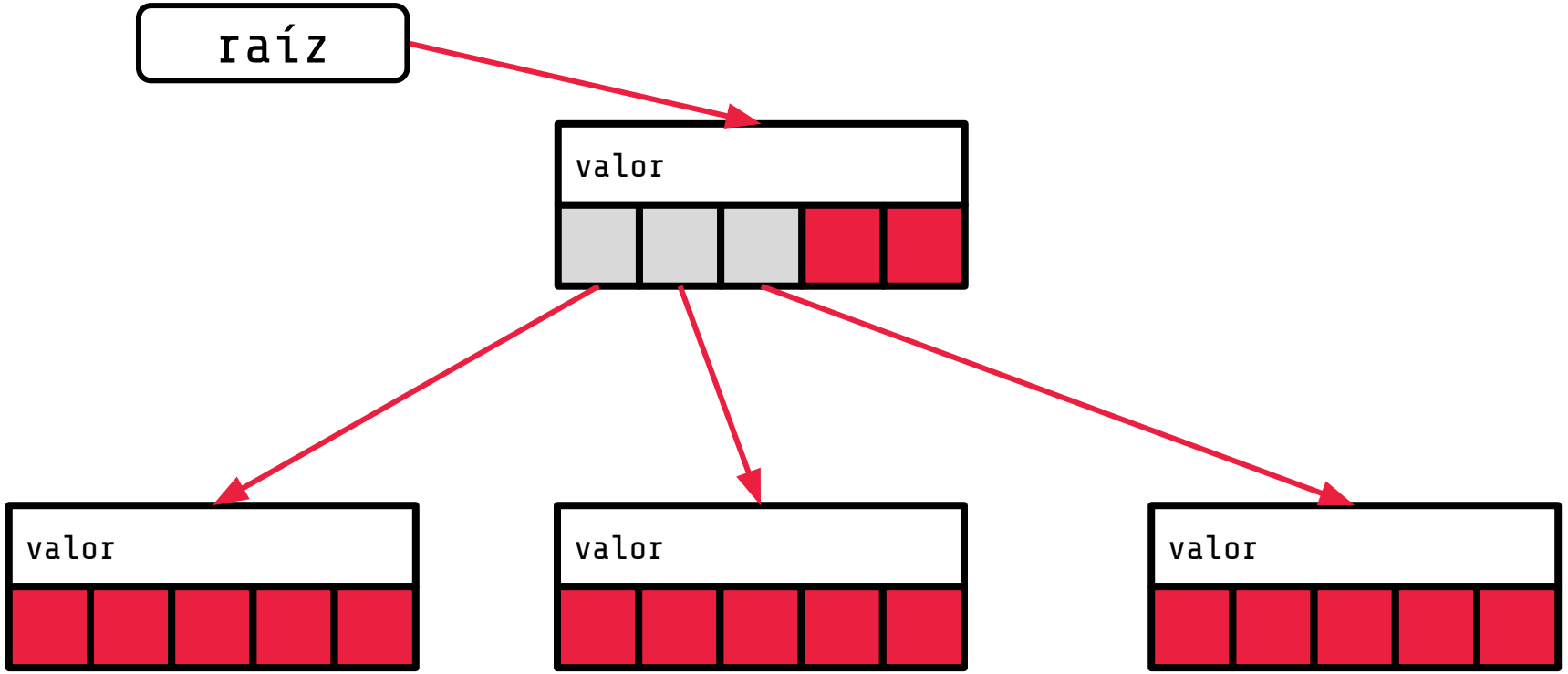

¿Preguntas?

{

Más árboles

}


raíz
1
2
3
4
5
6
7

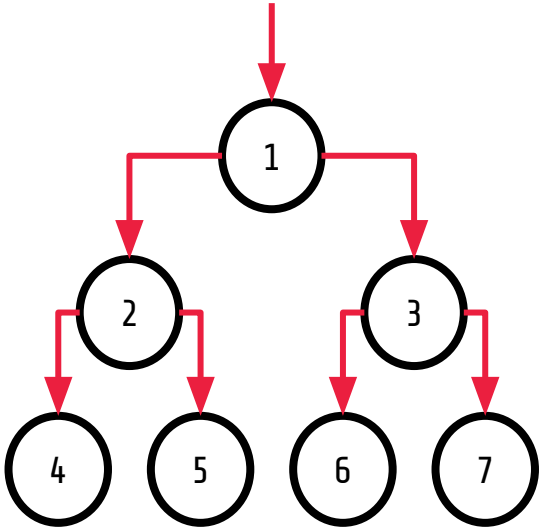


Arbol N-ario


¿Preguntas?

Recorrido en amplitud

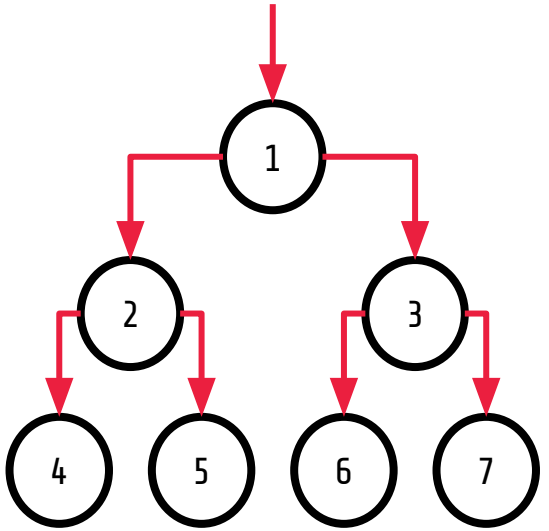
[1, 2, 3, 4, 5, 6, 7]



Requiere de una Queue

Iterativo es más simple

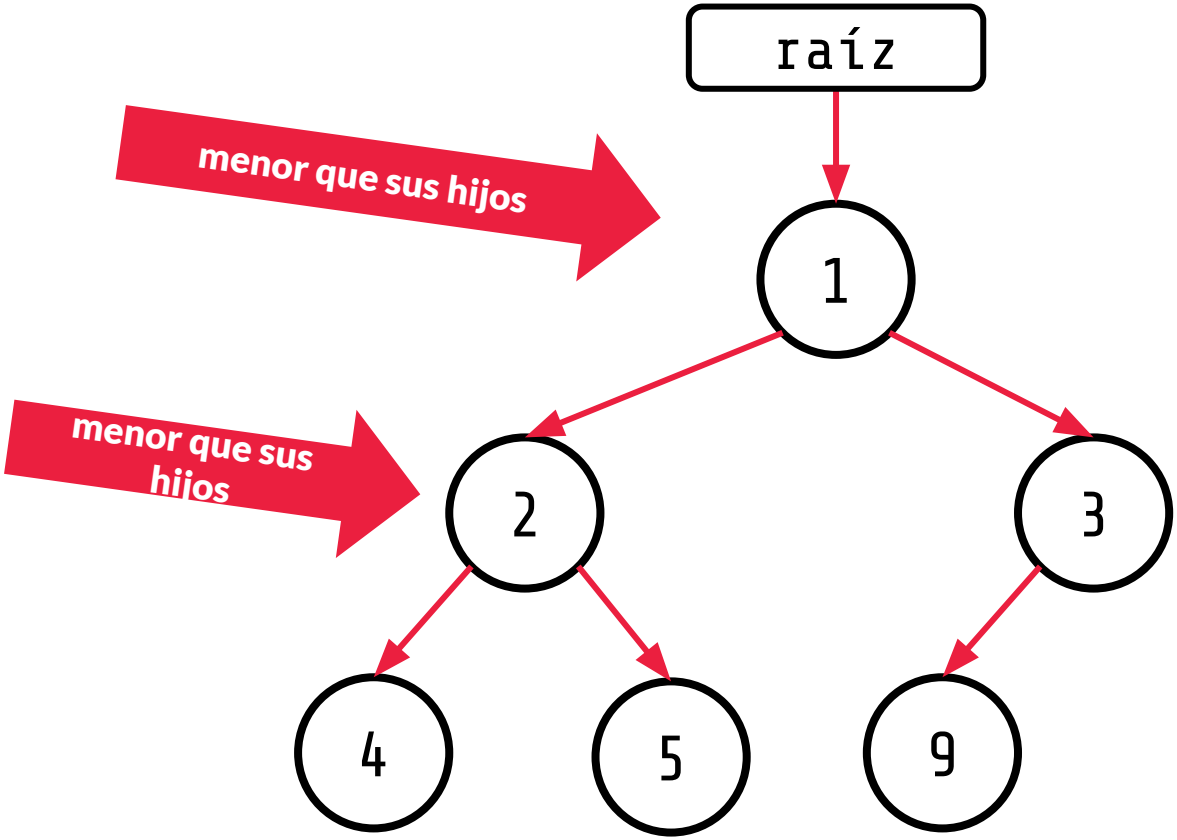
```
public staic ListaEnlazada<T> enAmplitud(Nodo<T> raiz) {
     if (raiz == null) {
          return;
     }
     Queue<Nodo> cola = new Queue<>();
     ListaEnlazada<T> recorrido = new ListaEnlazada<T>();
     cola.agregar(root);
     while (!cola.estaVacia()) {
          Nodo nodo = cola.obtener();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




¿Preguntas?

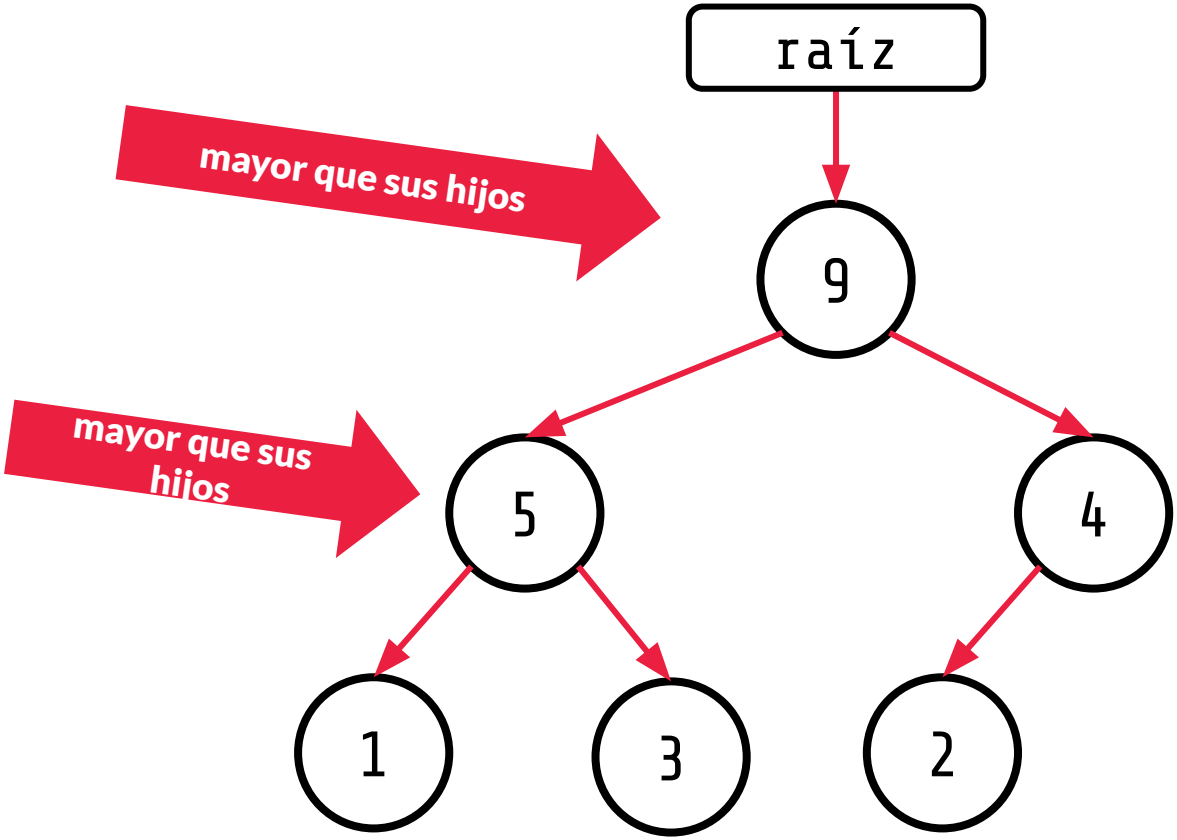
Heap/monticulo

MIN-Heap



¿Cual es el recorrido Preorden?

MAX-Heap



¿Cuál es el recorrido Preorden?


¿Preguntas?

Estructuras avanzadas

unrn.edu.ar

UNRN Universidad Nacional de Río Negro

| unrionegro