

# Taller 2a – IIC2115

Matías Gaete Silva – [mzgaete@uc.cl](mailto:mzgaete@uc.cl)

## Ejercicio 1

Una secuencia de paréntesis está balanceada si cada paréntesis abierto tiene su correspondiente paréntesis cerrado y viceversa.

" ( ( ) ) ( ) "

Balanceada

" ( ( ) ) ( ) ) "

No  
Balanceada

" ) ( "

No  
Balanceada

## Ejercicio 1

Un paréntesis abierto **podría** comenzar una secuencia de paréntesis balanceada.

" ( ( ( ) ) ( ( "

No comienzan una  
secuencia balanceada

" ( ( ( ) ) ( ( "

Comienzan una  
secuencia balanceada

Si itero de izquierda a derecha y obtengo un paréntesis abierto, ¿puedo saber si ese paréntesis comenzará una secuencia balanceada?

No, ya que no tengo información o registro de lo que vendrá.

El string: " ( ( ( ) ) ( ( "

Lo que sé en la iteración  $i$ : " ( ( ( ? ? ? ? "

↑  
↓  
 $i$

## Ejercicio 1

Un paréntesis cerrado **podría** terminar una secuencia de paréntesis balanceada.

" ) ( ( ) ) "

No terminan una  
secuencia balanceada

" ) ( ( ) ) "

Terminan una secuencia  
balanceada

Si itero de izquierda a derecha y obtengo un paréntesis cerrado, ¿puedo saber si ese paréntesis termina una secuencia balanceada?

Sí! Tengo registro de lo que ha pasado en el string... pero cómo hacerlo?

Usando stacks!

## Ejercicio 1

Si obtengo un paréntesis cerrado que termina una secuencia, la idea es tener acceso a la posición anterior del paréntesis abierto que comienza la secuencia.

" ( ( ( ) ) ( ( "

$i-1$     $i$     $j$

Largo de la secuencia:  $j - (i - 1)$


Para esto, en el stack iremos guardando la posición de los paréntesis abiertos. Si nos encontramos con un paréntesis cerrado, “anularemos” el paréntesis abierto asociado a este quitando el último elemento del stack, de tal manera que ahora el nuevo último elemento del stack sea la posición anterior del paréntesis abierto que comienza la secuencia.

## Ejercicio 1

max\_long: 0

Stack: []

" ( ( ( ) ) ( ( "



0

The diagram illustrates the first step of a stack-based algorithm. A string " ( ( ( ) ) ( ( " is shown. A vertical double-headed arrow points from the first character, a double quote, down to the index 0. This indicates that the first character is being pushed onto the stack at index 0.

Stack: [0]

## Ejercicio 1

max\_long: 0

Stack: [0]

" ( ( ( ) ) ( ( "



1

Stack: [0,1]

## Ejercicio 1

max\_long: 0

Stack: [0,1]

" ( ( ( ) ) ( ( "



2

Stack: [0,1,2]



## Ejercicio 1

max\_long: 0

Stack: [0,1,2]

" ( ( ( ) ) ( ( "



3

Stack: [0,1]

Largo secuencia:  $3 - \text{stack}[-1] = 3 - 1 = 2$

max\_long: 2

## Ejercicio 1

max\_long: 2

Stack: [0,1]

" ( ( ( ) ) ( ( "



4

Stack: [0]

Largo secuencia:  $4 - \text{stack}[-1] = 4 - 0 = 4$

max\_long: 4

## Ejercicio 1

max\_long: 4

Stack: [0]

" ( ( ( ) ) ( ( "



5

Stack: [0,5]

## Ejercicio 1

max\_long: 4

Stack: [0,5]

" ( ( ( ) ) ( ( "



6

Stack: [0,5,6]

Return max\_long

## Ejercicio 1

¿Qué pasa si el primer elemento es un paréntesis abierto que comienza una secuencia?

max\_long: 0

Stack: []

" ( ) ( ) ) ( ) "



0

Stack: [0]

## Ejercicio 1

¿Qué pasa si el primer elemento es un paréntesis abierto que comienza una secuencia?

max\_long: 0

Stack: [0]

" ( ) ( ) ) ( ) "



1

Stack: []

Largo secuencia: 1 – stack[-1] (IndexError ☹)

## Ejercicio 1

¿Qué pasa si el primer elemento es un paréntesis abierto que comienza una secuencia?

“ ( ) ( ) ) ( ) ”  
↑      ↑  
?      0

Si el primer elemento es un paréntesis abierto que comienza una secuencia, entonces su posición es 0 y su posición anterior podemos considerar que es -1. Por lo tanto, debemos inicializar el stack con un -1 para incluir este caso.

## Ejercicio 1

¿Qué pasa si el primer elemento es un paréntesis abierto que comienza una secuencia?

max\_long: 0

Stack: [-1]

" ( ) ( ) ) ( ) "



0

Stack: [-1,0]



## Ejercicio 1

¿Qué pasa si el primer elemento es un paréntesis abierto que comienza una secuencia?

max\_long: 0

Stack: [-1,0]

" ( ) ( ) ) ( ) "



1

Stack: [-1]

Largo secuencia:  $1 - \text{stack}[-1] = 1 - (-1) = 2$  😊

max\_long: 2

## Ejercicio 1

max\_long: 2

Stack: [-1]

" ( ) ( ) ) ( ) "



2

Stack: [-1,2]

## Ejercicio 1

max\_long: 2

Stack: [-1,2]

" ( ) ( ) ) ( ) "



3

Stack: [-1]

Largo secuencia:  $3 - \text{stack}[-1] = 3 - (-1) = 4$

max\_long: 4

## Ejercicio 1

max\_long: 4

Stack: [-1]

" ( ) ( ) ) ( ) "



4

Stack: []

Largo secuencia: 4 – stack[-1] (IndexError ☹)

¿Qué pasó aquí?

## Ejercicio 1

Hasta el momento habíamos asumido que todo paréntesis cerrado termina una secuencia, es decir, que existía un paréntesis abierto asociado a él. El caso visto nos indica que cuando el stack queda vacío, el paréntesis cerrado no termina una secuencia. Para incluir este caso, agregamos la posición del paréntesis cerrado al stack. Esto lo hacemos con dos intenciones:

1. Para que el largo de la secuencia de 0.
2. Para que sirva como posición anterior de un paréntesis abierto que comience una secuencia.

## Ejercicio 1

max\_long: 4

Stack: [-1]

" ( ) ( ) ) ( ) "



4

Stack: []

Stack: [4]

Largo secuencia:  $4 - \text{stack}[-1] = 4 - 4 = 0$

## Ejercicio 1

max\_long: 4

Stack: [4]

" ( ) ( ) ) ( ) "



5

Stack: [4,5]

## Ejercicio 1

max\_long: 4

Stack: [4,5]

" ( ) ( ) ) ( ) "



6

Stack: [4]

Largo secuencia:  $6 - \text{stack}[-1] = 6 - 4 = 2$

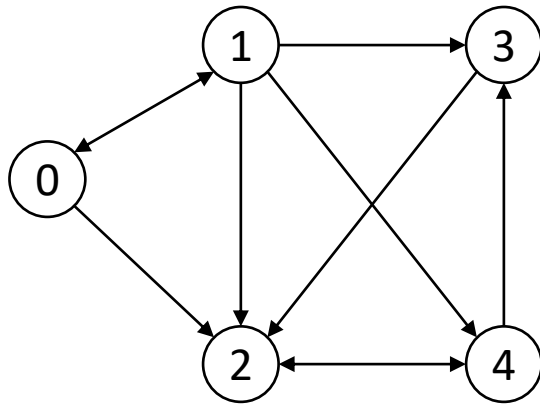
Return max\_long



## Ejercicio 2

Dado un grafo dirigido, un nodo de origen  $o$  y un nodo de destino  $d$ , encuentre la cantidad de caminos de largo  $m$  que van de  $o$  hasta  $d$ . Se recomienda basar la solución en el algoritmo BFS.

Un camino de largo  $m$  es una secuencia finita y ordenada de nodos  $[i_0, i_1, \dots, i_m]$  en donde cada par consecutivo de nodos forma un arco.



Caminos con origen 0 y destino 2 de largo 3:

$[0,1,0,2]$

$[0,1,4,2]$

$[0,1,3,2]$

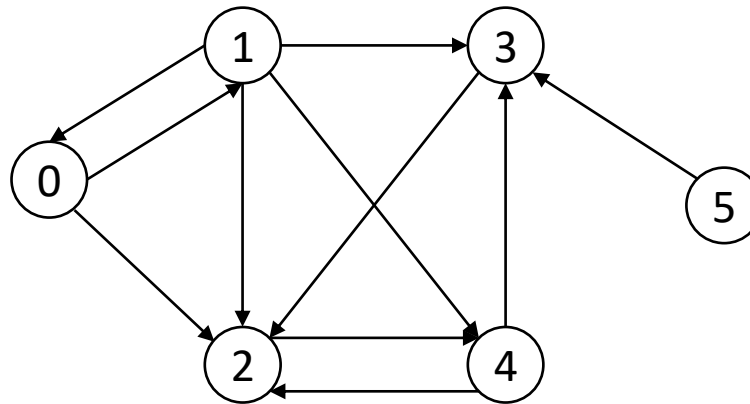
$[0,2,4,2]$

## Ejercicio 2

BFS

visitados: {}

cola: [0]



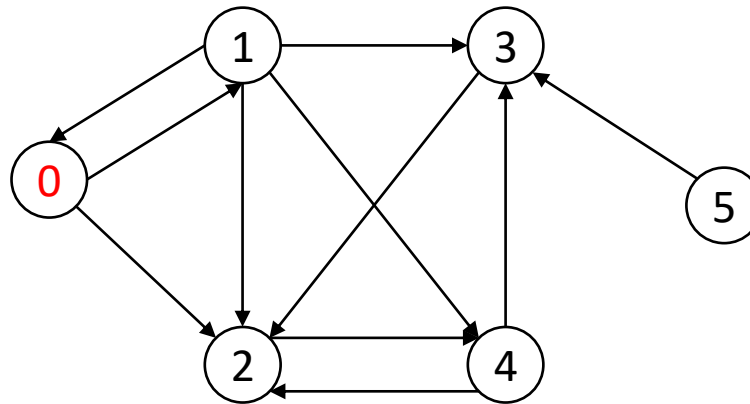
¿Está la cola vacía? No, saco el primer elemento.

## Ejercicio 2

BFS

visitados: {}

cola: []



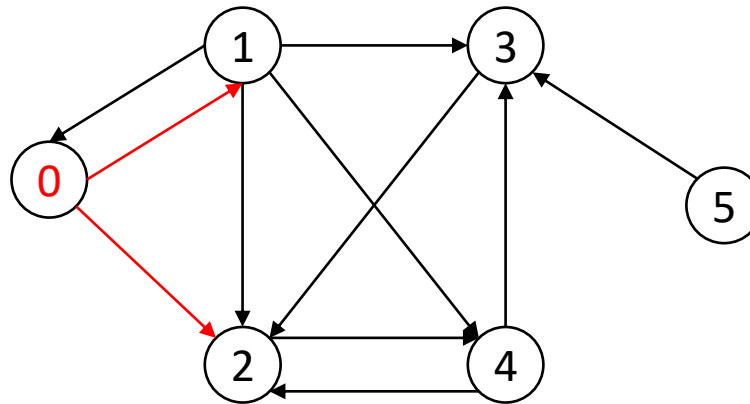
¿Está 0 en visitados? No, lo agrego.

## Ejercicio 2

BFS

visitados: {0}

cola: []



Reviso nodos posteriores:

¿Está 1 en visitados? No, lo agrego a la cola.

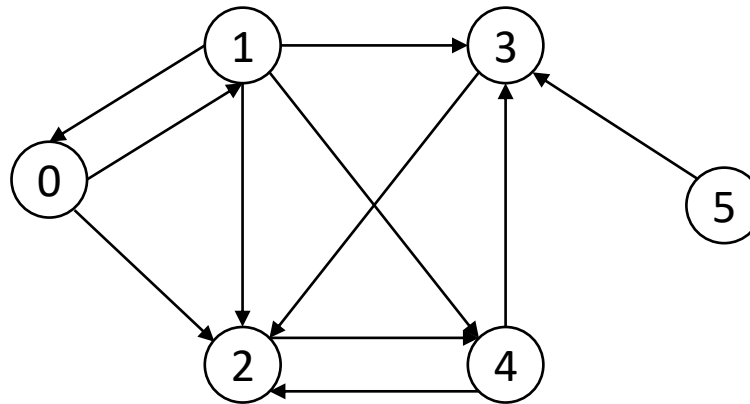
¿Está 2 en visitados? No, lo agrego a la cola.

## Ejercicio 2

BFS

visitados: {0}

cola: [1,2]



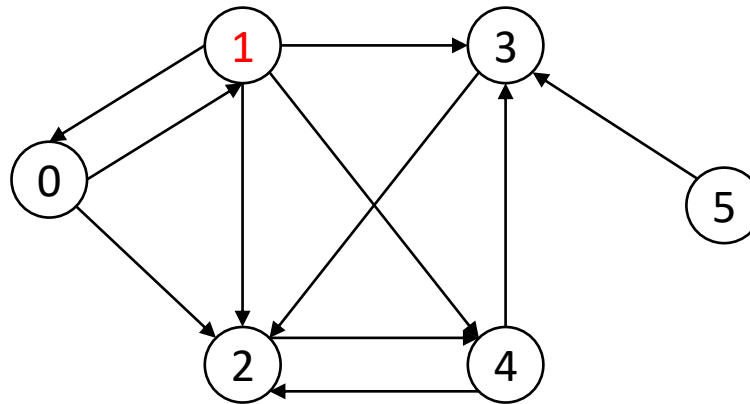
¿Está la cola vacía? No, saco el primer elemento.

## Ejercicio 2

BFS

visitados: {0}

cola: [2]



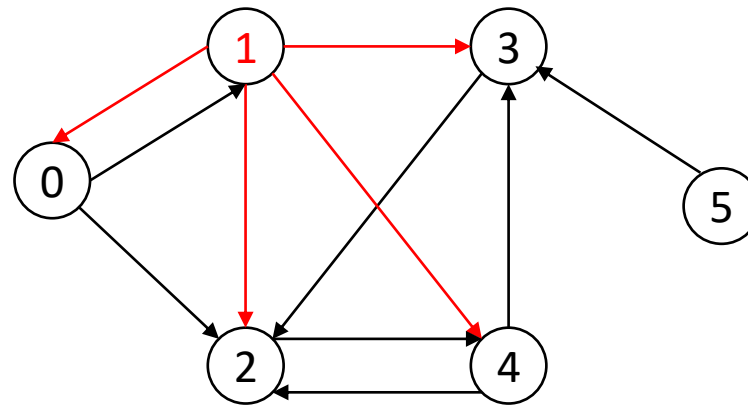
¿Está 1 en visitados? No, lo agrego.

## Ejercicio 2

BFS

visitados: {0,1}

cola: [2]



Reviso nodos posteriores:

¿Está 0 en visitados? Sí, continúo.

¿Está 2 en visitados? No, lo agrego a la cola.

¿Está 3 en visitados? No, lo agrego a la cola.

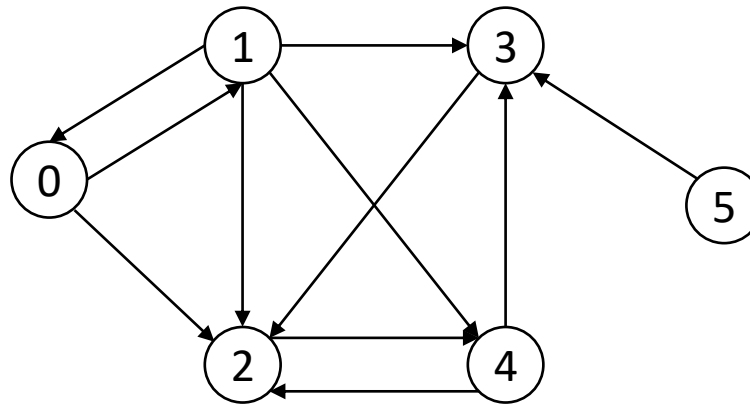
¿Está 4 en visitados? No, lo agrego a la cola.

## Ejercicio 2

BFS

visitados: {0,1}

cola: [2,2,3,4]



¿Está la cola vacía? No, saco el primer elemento.

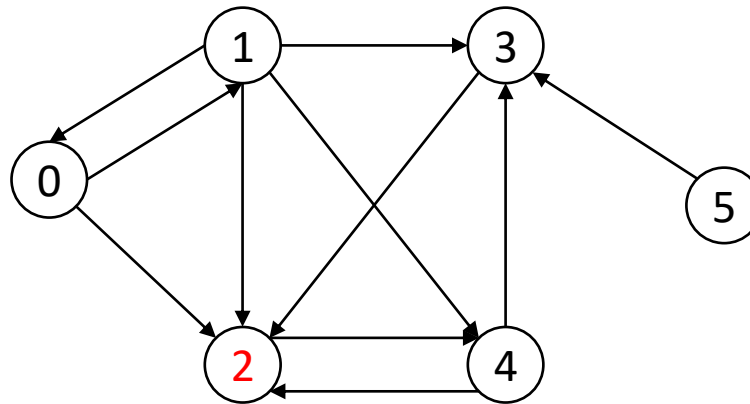


## Ejercicio 2

BFS

visitados: {0,1}

cola: [2,3,4]



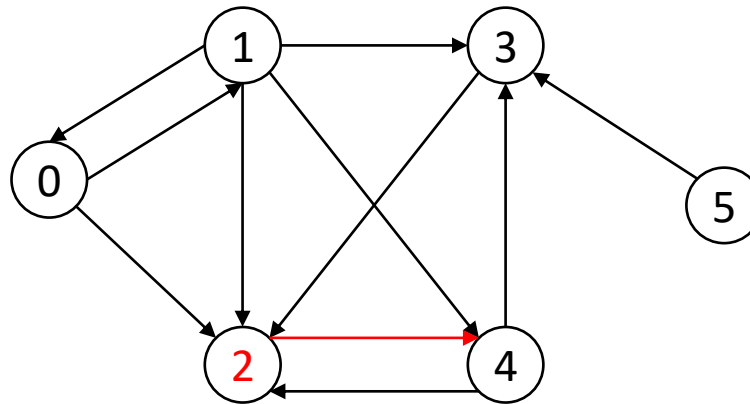
¿Está 2 en visitados? No, lo agrego.

## Ejercicio 2

BFS

visitados: {0,1,2}

cola: [2,3,4]



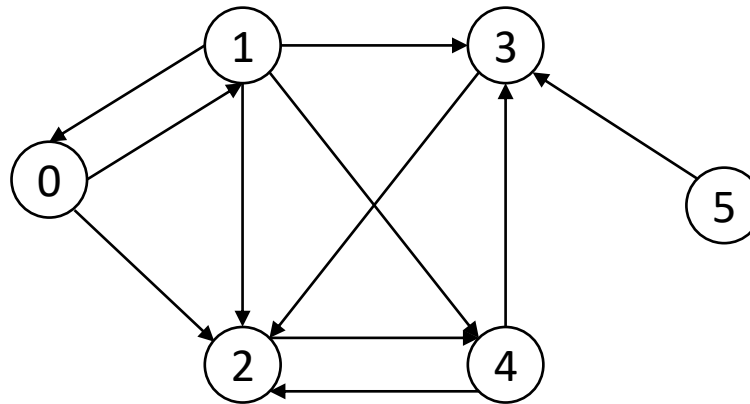
Reviso nodos posteriores:  
¿Está 4 en visitados? No, lo agrego a la cola.

## Ejercicio 2

BFS

visitados: {0,1,2}

cola: [2,3,4,4]



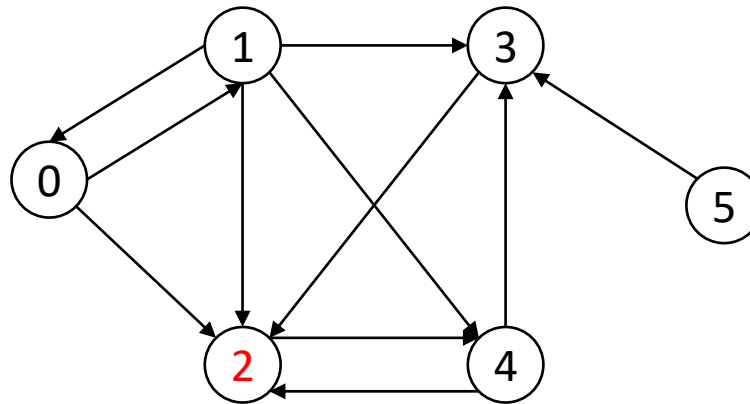
¿Está la cola vacía? No, saco el primer elemento.

## Ejercicio 2

BFS

visitados: {0,1,2}

cola: [3,4,4]



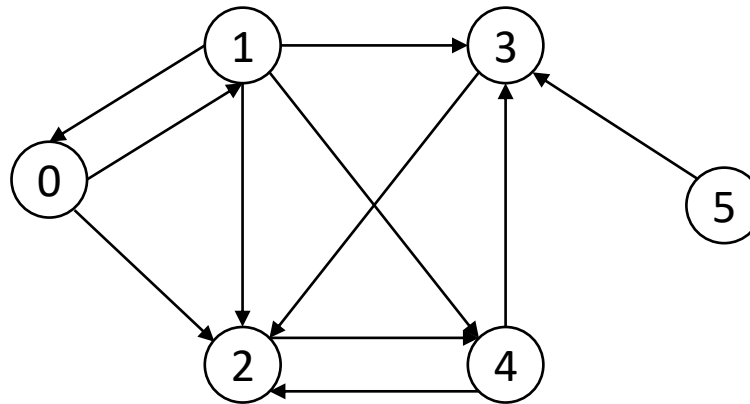
¿Está 2 en visitados? Sí, continúo.

## Ejercicio 2

BFS

visitados: {0,1,2}

cola: [3,4,4]



¿Está la cola vacía? No, saco el primer elemento.

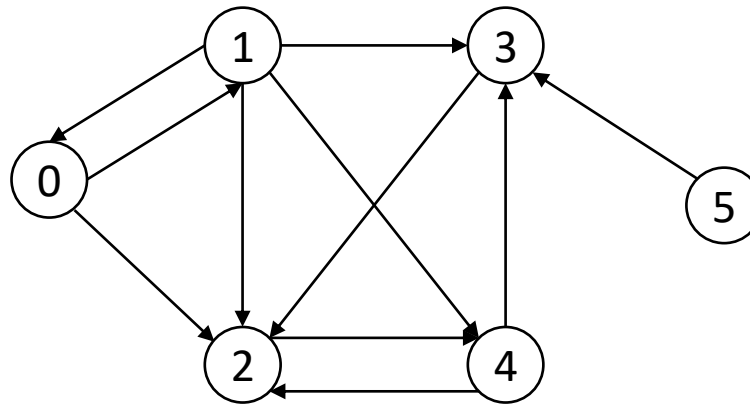
...

## Ejercicio 2

BFS

visitados: {0,1,2,3,4}

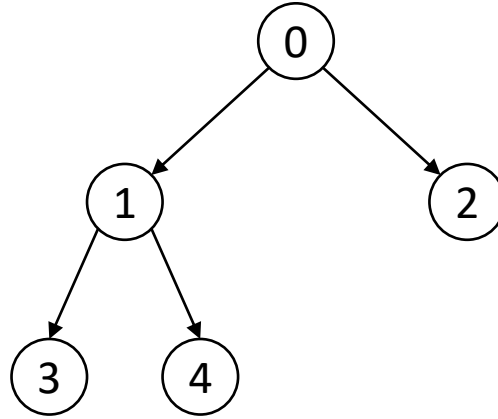
cola: []



¿Está la cola vacía? Sí, fin.

## Ejercicio 2

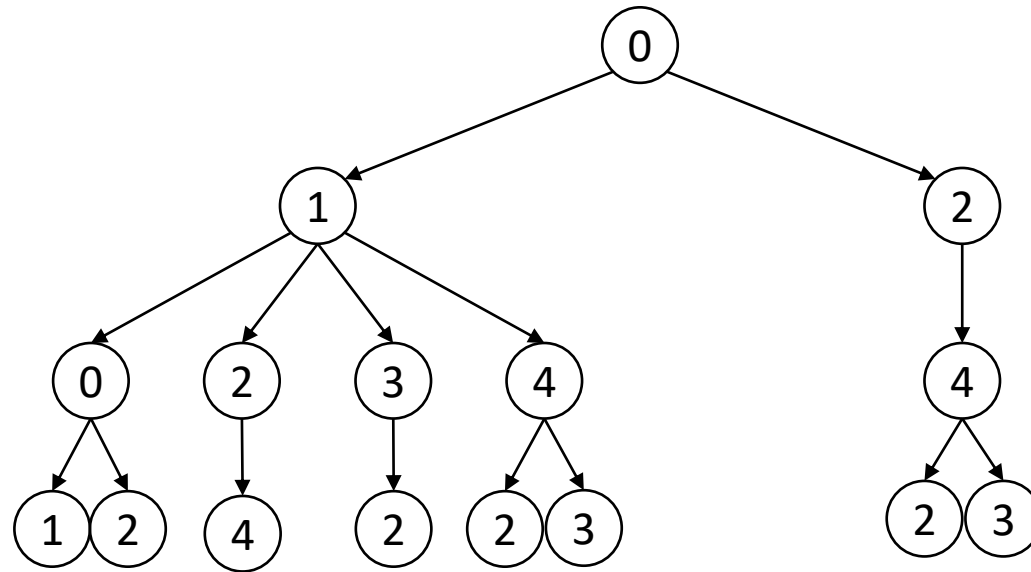
BFS: Árbol de búsqueda



visitados impide repetir nodos y buscar caminos

## Ejercicio 2

BFS sin visitados

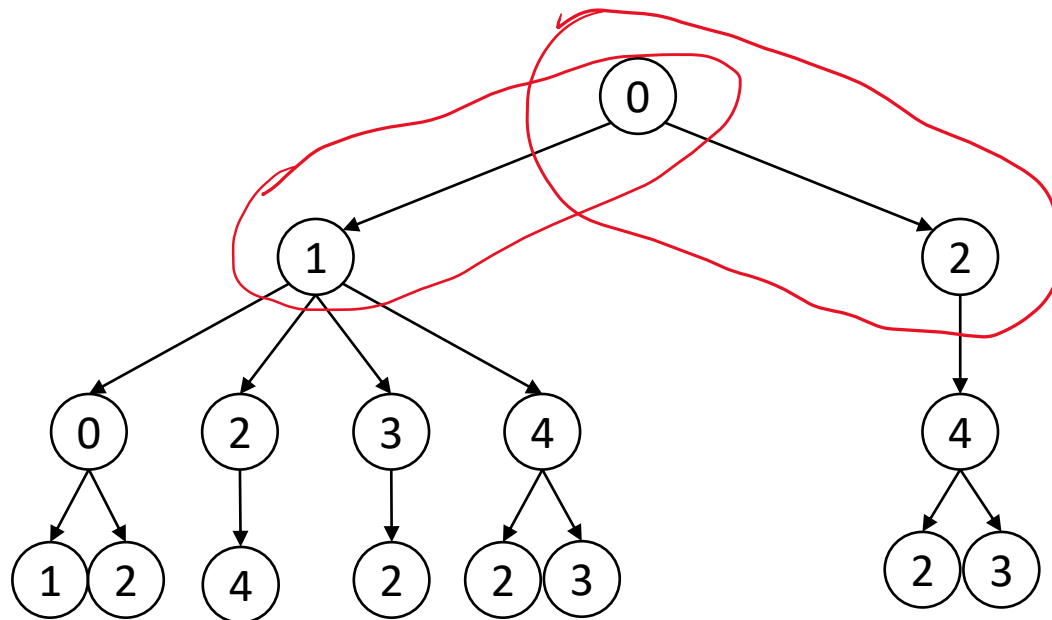


...



## Ejercicio 2

Idea: en la cola ir guardando **los caminos**

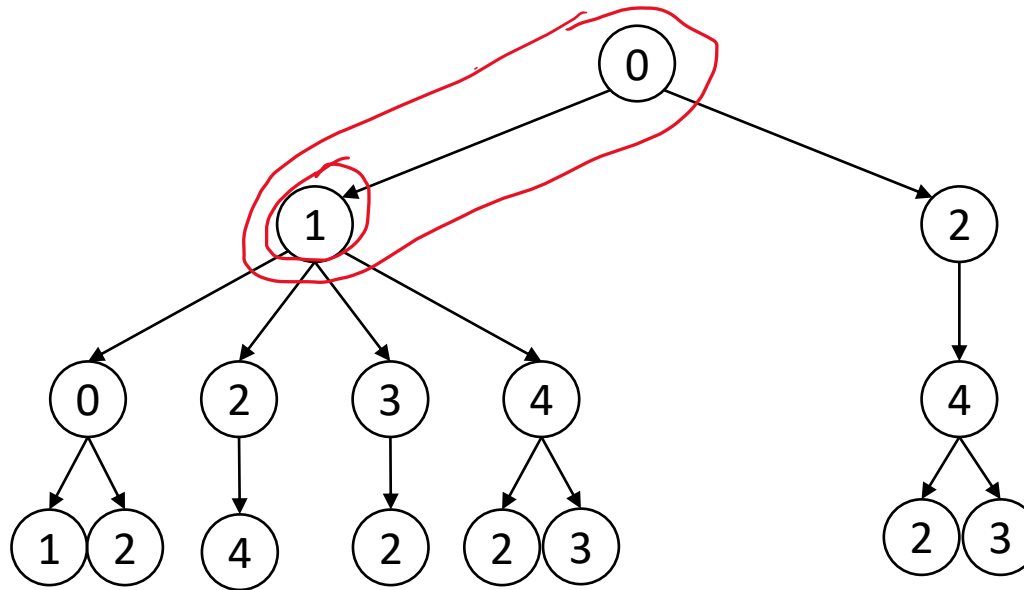


cola:  $[[0,1],[0,2]]$

## Ejercicio 2

Al sacar un camino de la cola, tomar el último nodo del camino.

1° Caso: si el largo del camino es menor que  $m - 1$ , recorrer los nodos posteriores del último nodo del camino y agregar a la cola los nuevos caminos.



camino: [0,1]  
ultimo\_nodo: 1

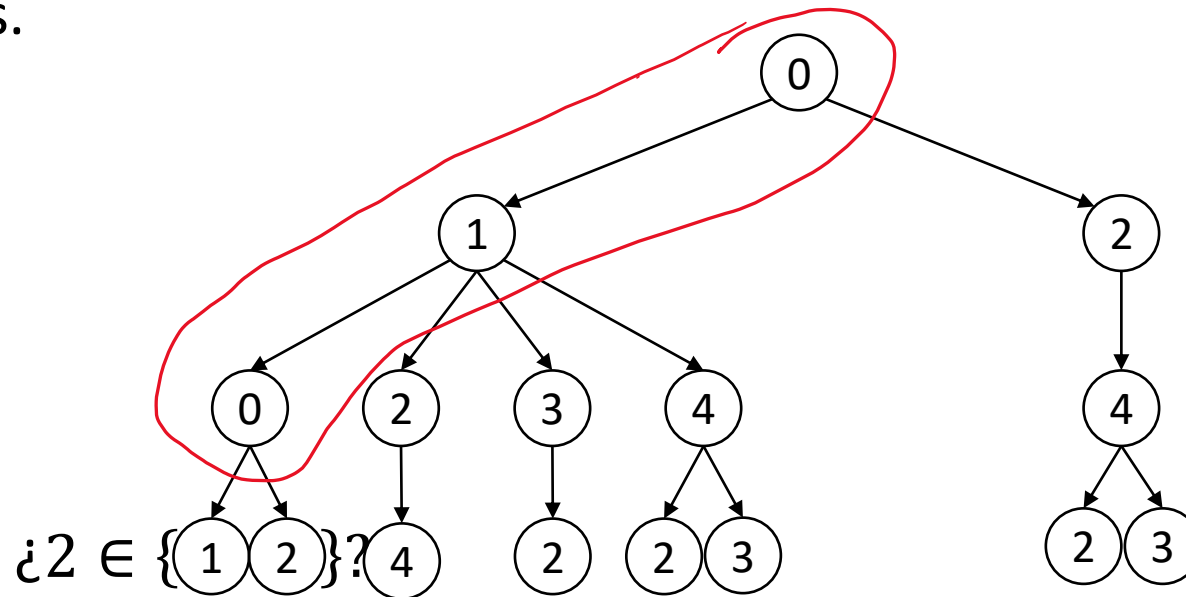
⋮

cola: [[0,2], [0,1,0], [0,1,2], [0,1,3], [0,1,4]]

## Ejercicio 2

Al sacar un camino de la cola, tomar el último nodo del camino.

2° Caso: si el largo del camino es  $m - 1$ , verificar si el destino está en los nodos posteriores del último nodo. Si es así, aumentar el contador de número de caminos.



camino: [0,1,0]  
ultimo\_nodo: 0

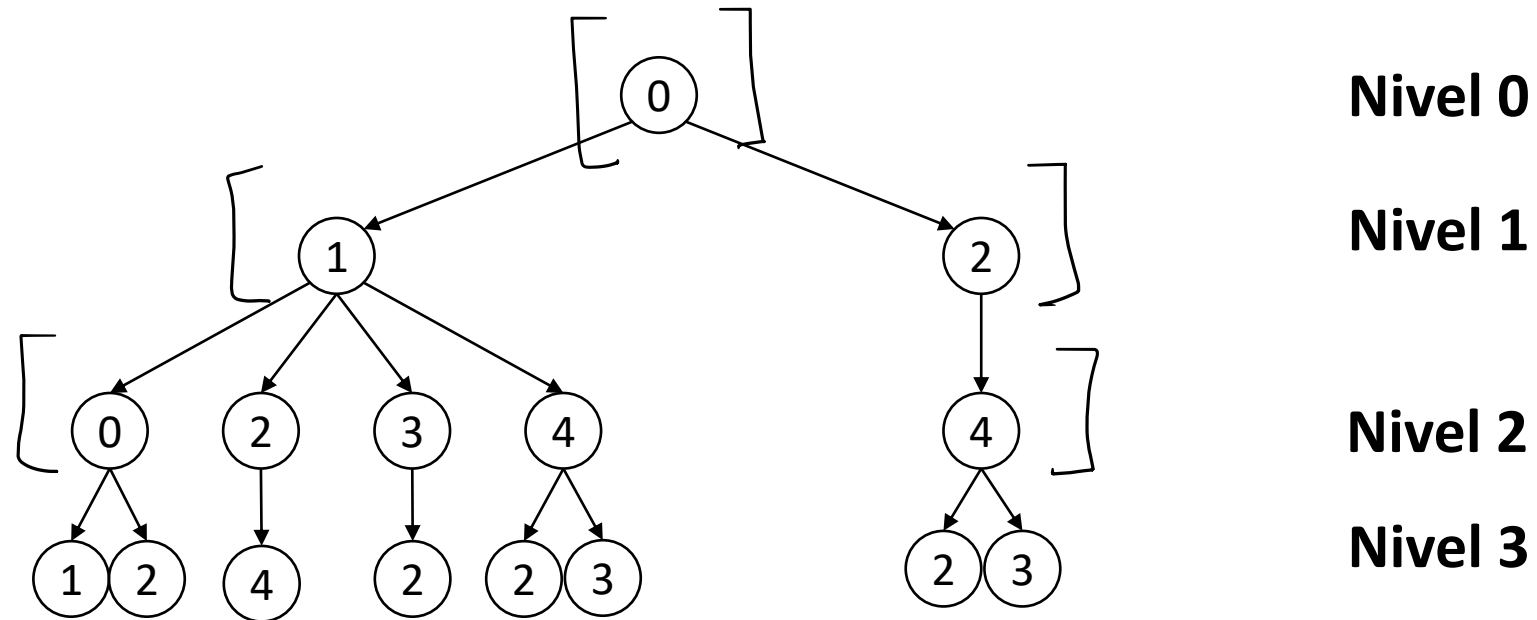
Sí! Aumentar en 1 el contador.

·  
·  
·

Observación:  $\text{len}(\text{camino}) \neq \text{largo del camino}$   
 **$\text{len}(\text{camino}) - 1 = \text{largo del camino}$**

## Ejercicio 2

2° Idea: ir trabajando con los niveles



Pros: no necesito la cola ni los caminos.

Contras: necesito formar la lista con los nodos de cada nivel (se puede).

Pierdo el largo de los caminos (se puede llevar contador de nivel)

Pierdo los caminos (nada que hacer ☹)