

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2115 - Programación como herramienta para la ingeniería

Capítulo 2 - Parte 1: Estructuras de datos

Profesores: Francisco Garrido Valenzuela
Hans Löbel

Estudiar por su cuenta

- Les recordamos que esta introducción no cubre todo el material disponible en GitHub, sólo lo fundamental.
- Es importante que estudien la materia del curso para profundizar los contenidos
- Luego, ¡¡Hacer ejercicios!!

Pensar, pensar, pensar y luego al código

- Entre más problemas resuelvo, más fácil será la resolución de futuros problemas
- Es importante entender completamente el problema antes de escribir líneas de código
- Usar ejemplos pequeños para facilitar el entendimiento
- Plantear una posible solución, entenderla y luego programarla

¿Qué son las Estructuras de Datos?

- Corresponden a un tipo de dato especializado, **diseñado para agrupar, almacenar o acceder a la información de manera más eficiente** que un tipo de dato básico.
- La elección adecuada de la estructura de datos es fundamental para el desarrollo de un buen programa y **hace la diferencia entre un programador y un buen programador.**

Listas, tuplas, stacks y colas

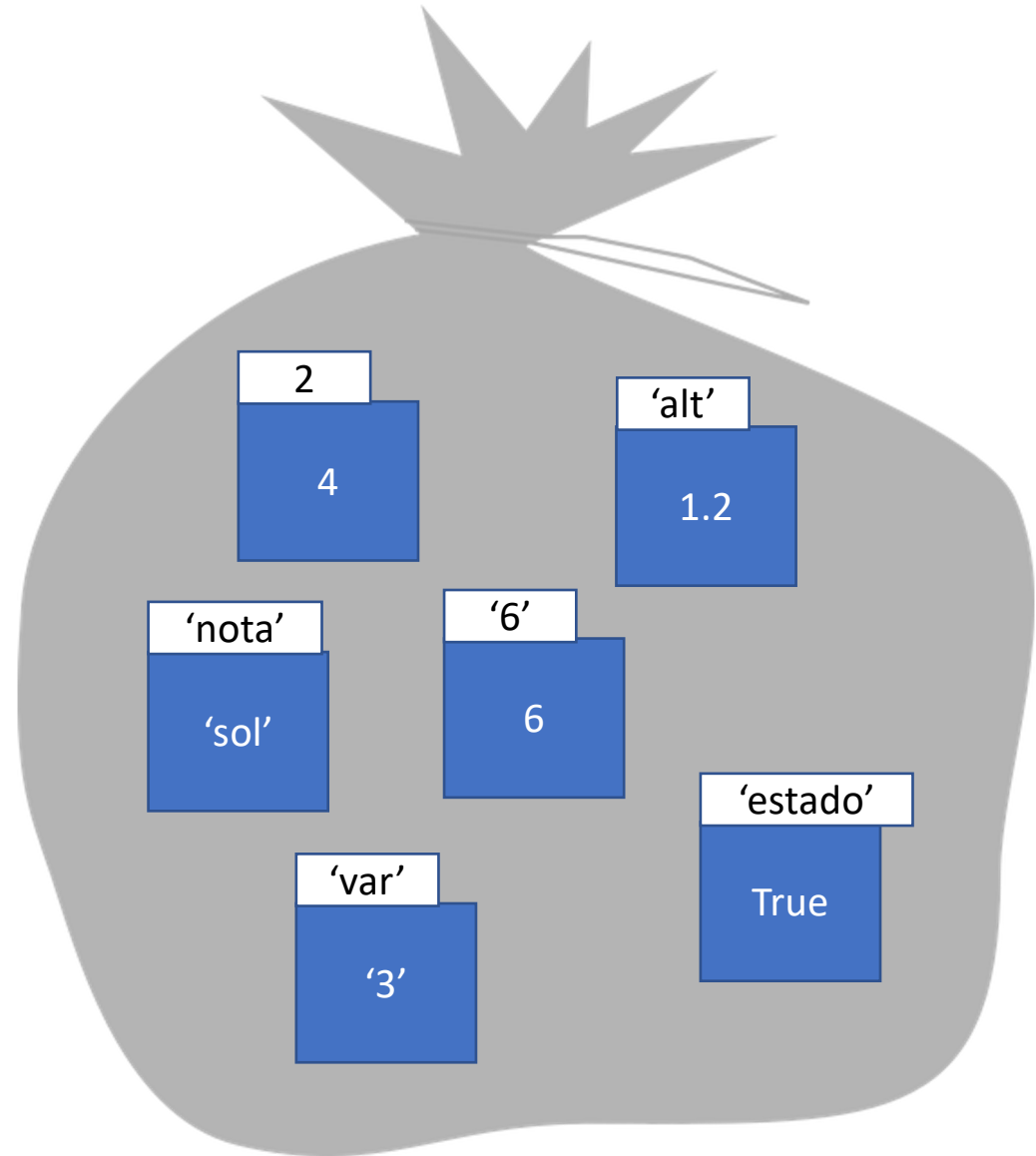
- Listas: []
- Tuplas: ()

4	6	12	21	...	1	1.2
[0]	[1]	[2]	[3]		[-2]	[-1]

- Stacks: LIFO
- Colas: FIFO

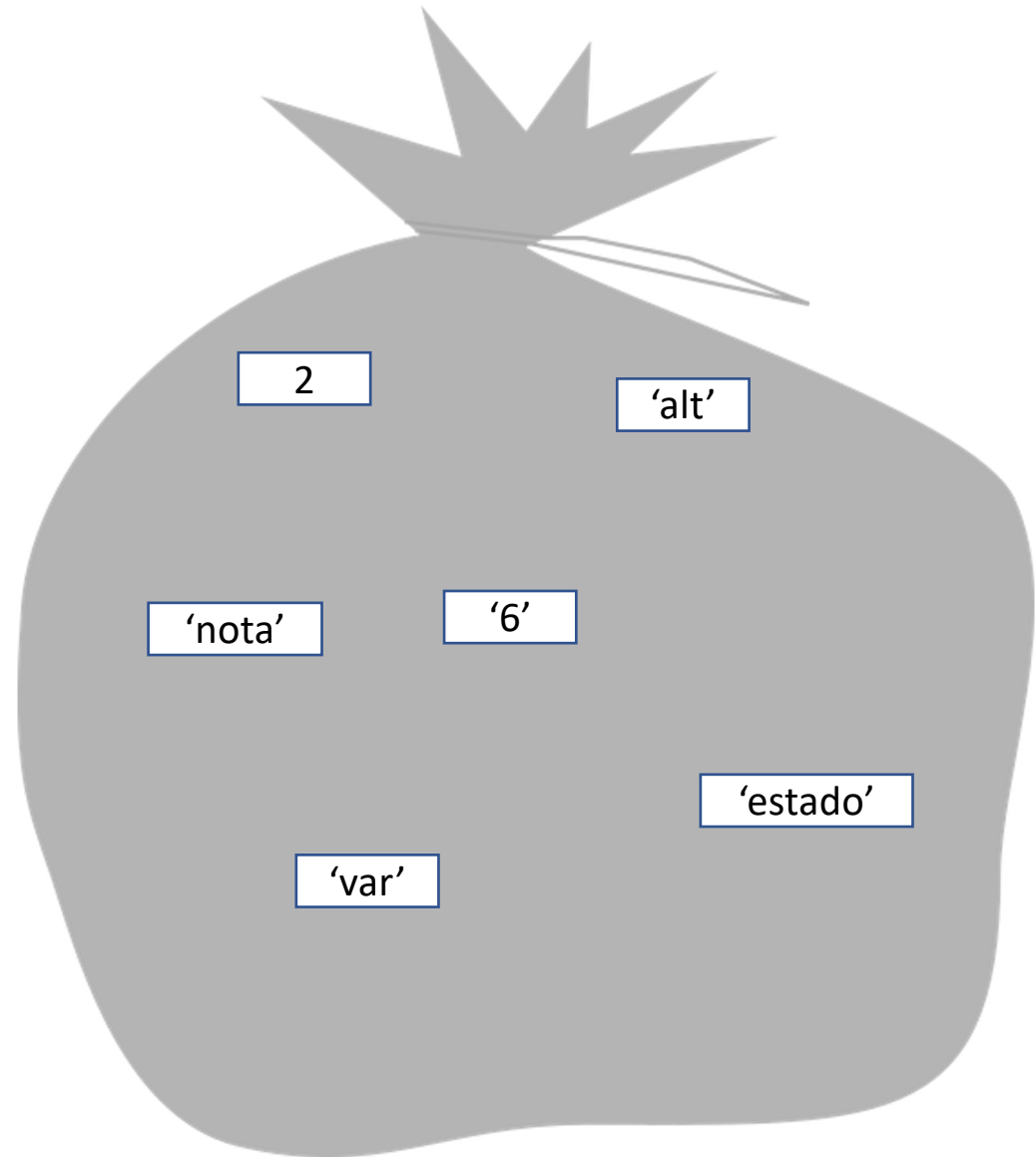
Diccionarios y Sets

- Diccionarios: {}
- Sets: {} (solo llaves)



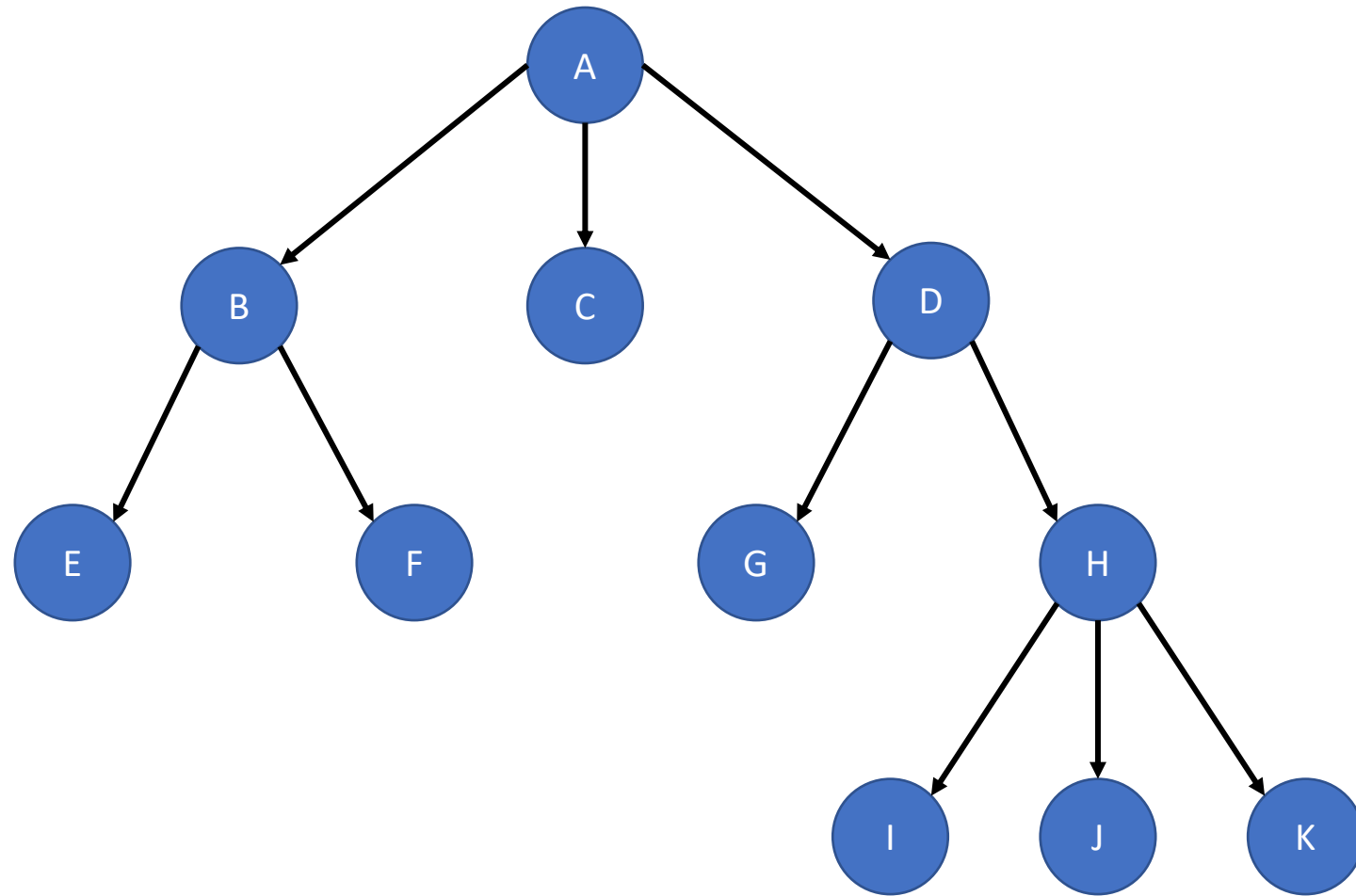
Diccionarios y Sets

- Diccionarios: {}
- Sets: {} (solo llaves)



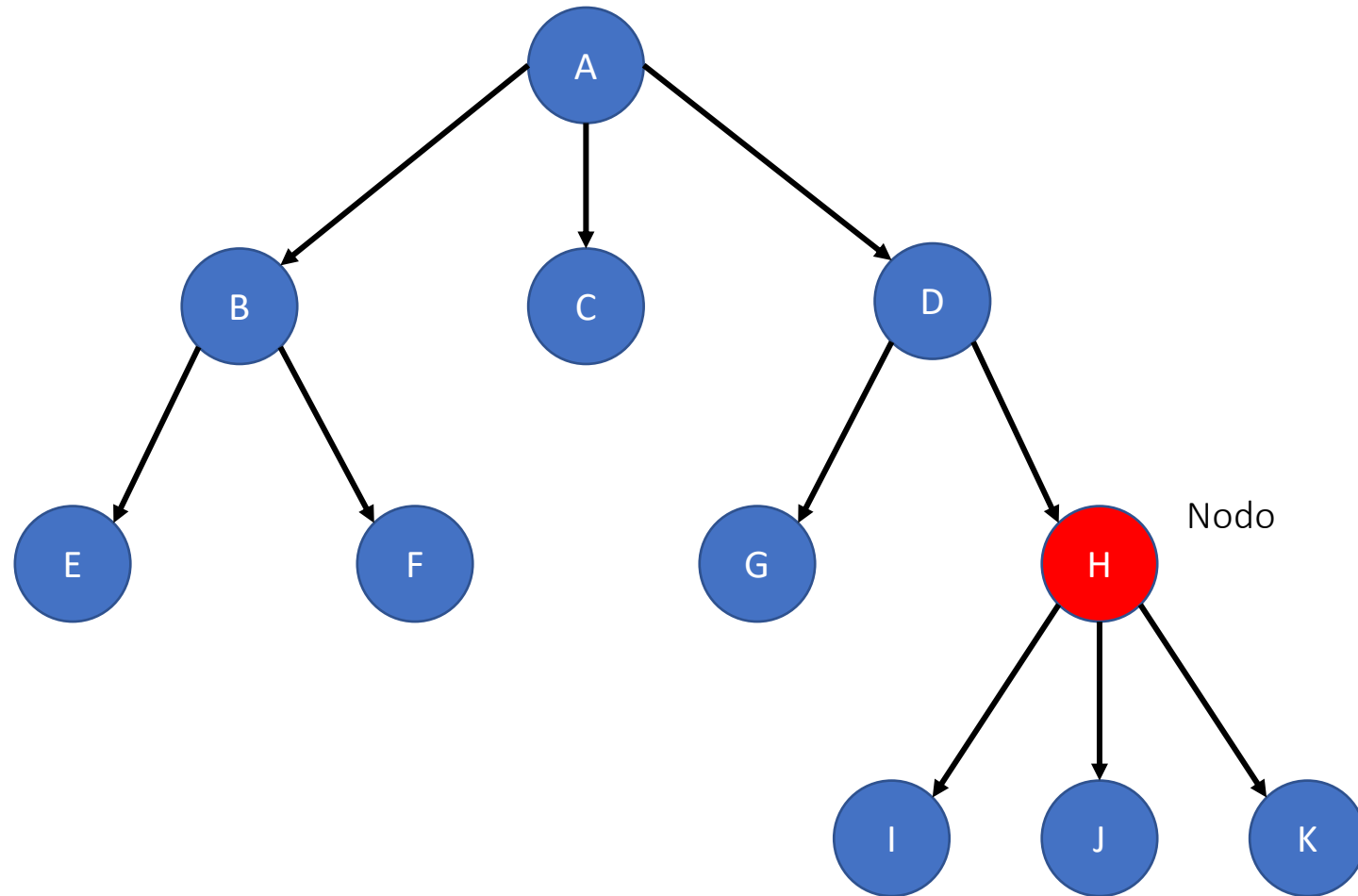
Arboles

Arbol: POO



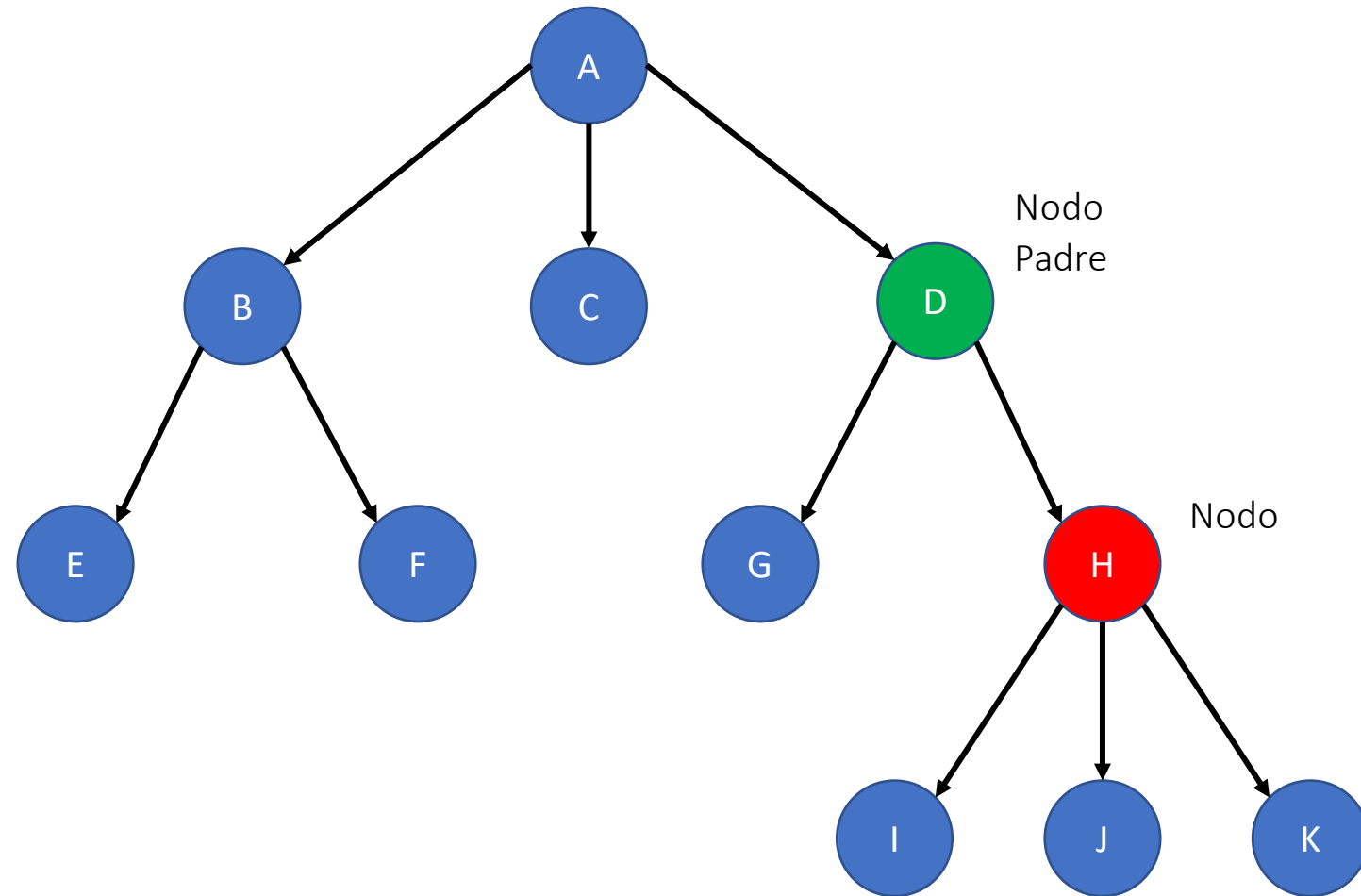
Arboles

Arbol: POO



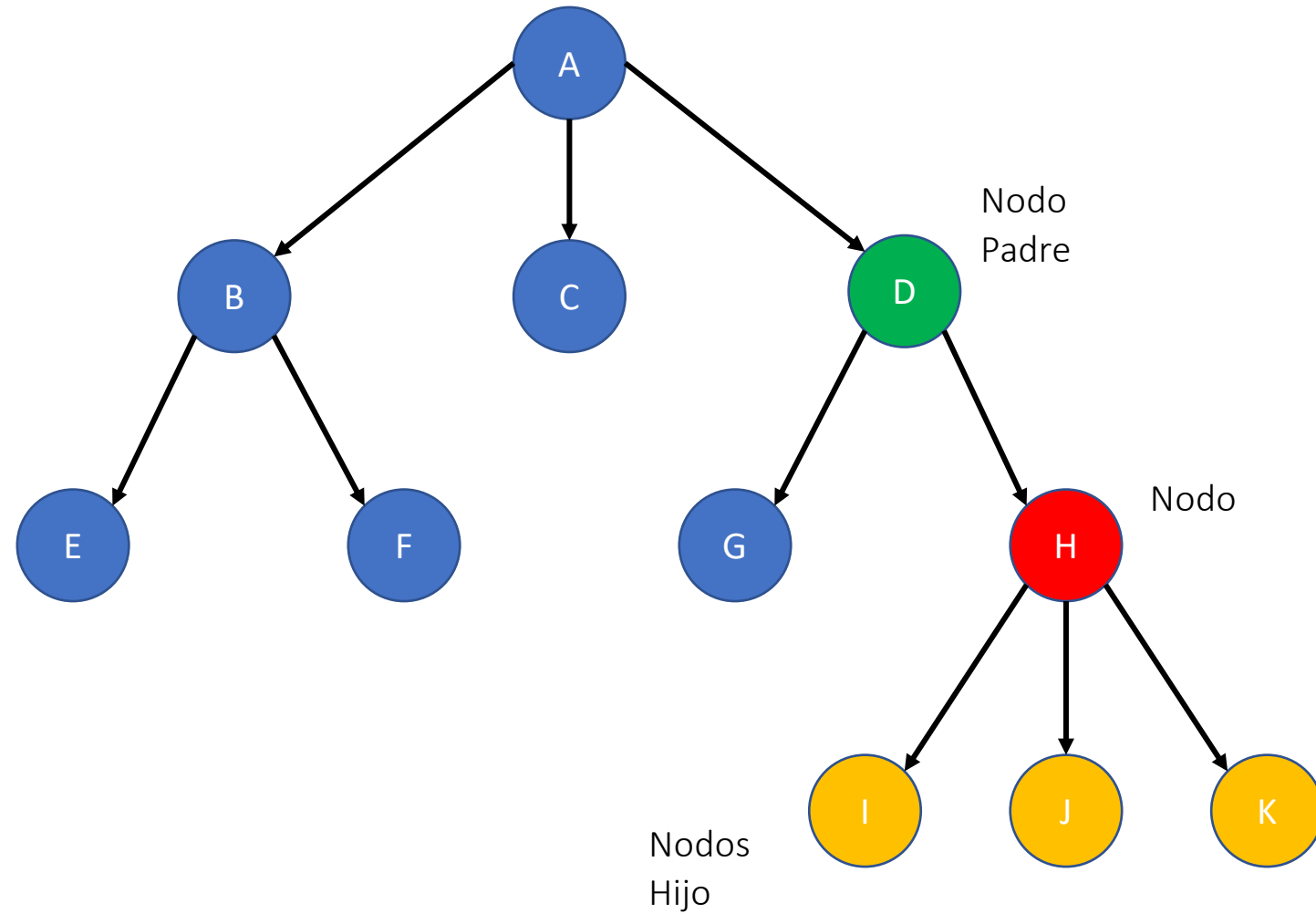
Arboles

Arbol: POO

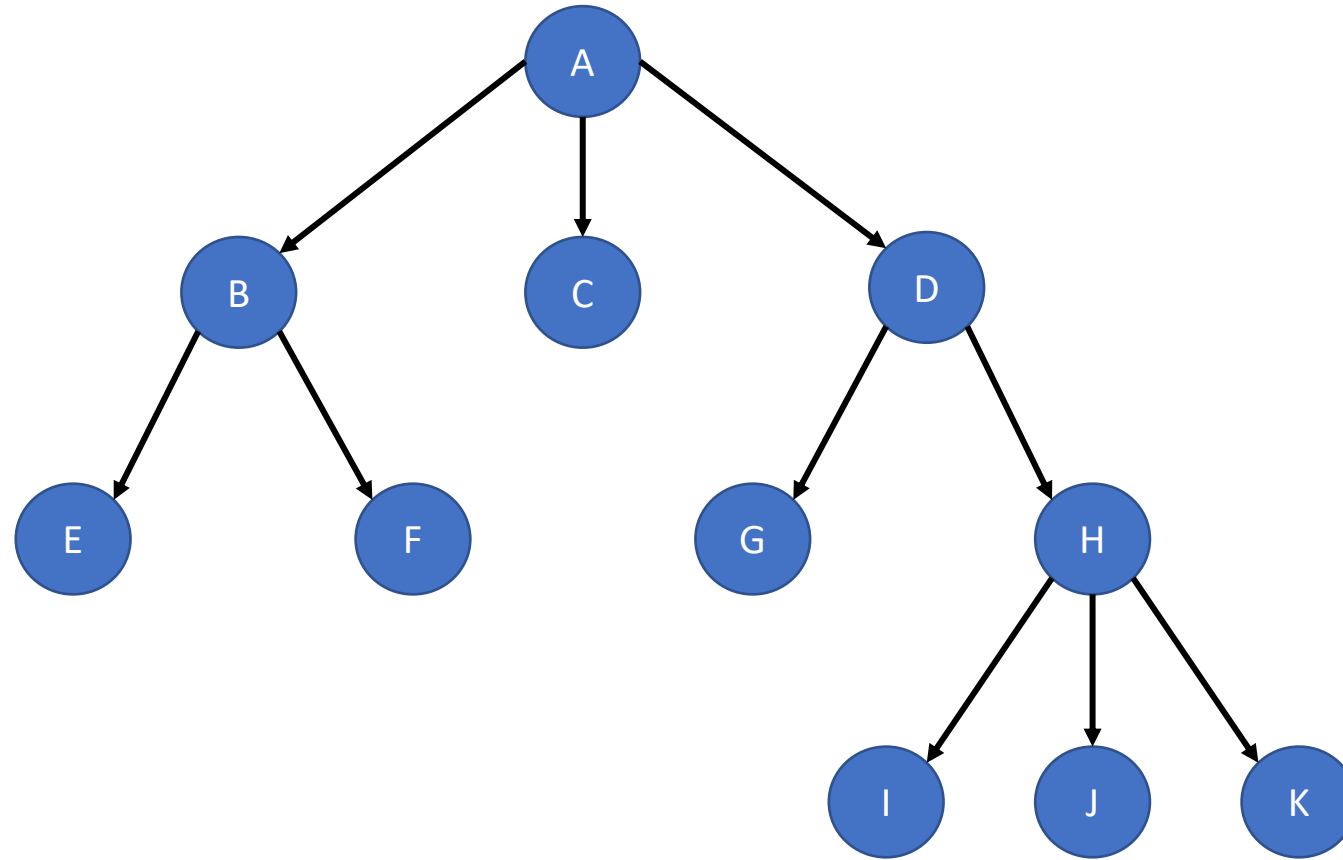


Arboles

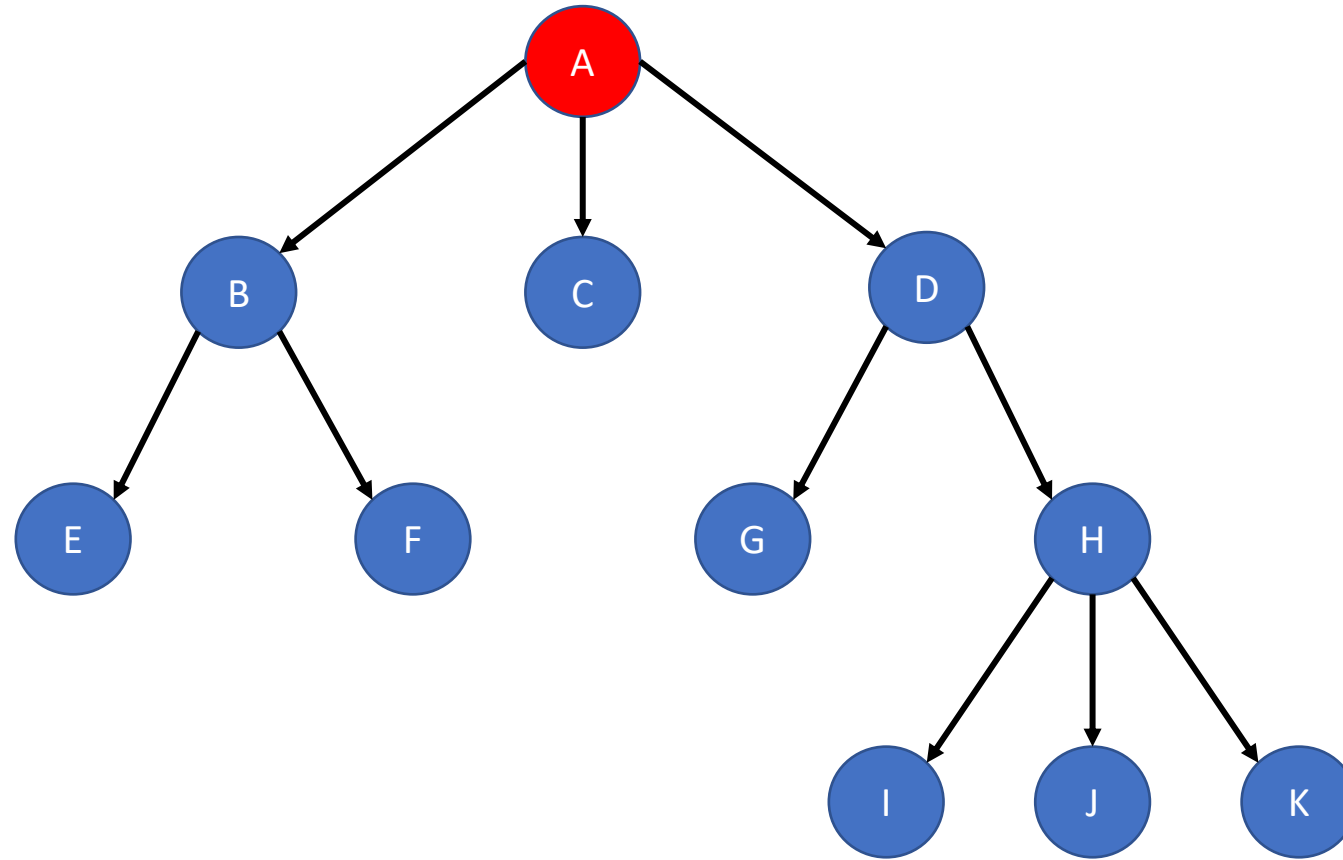
Arbol: POO



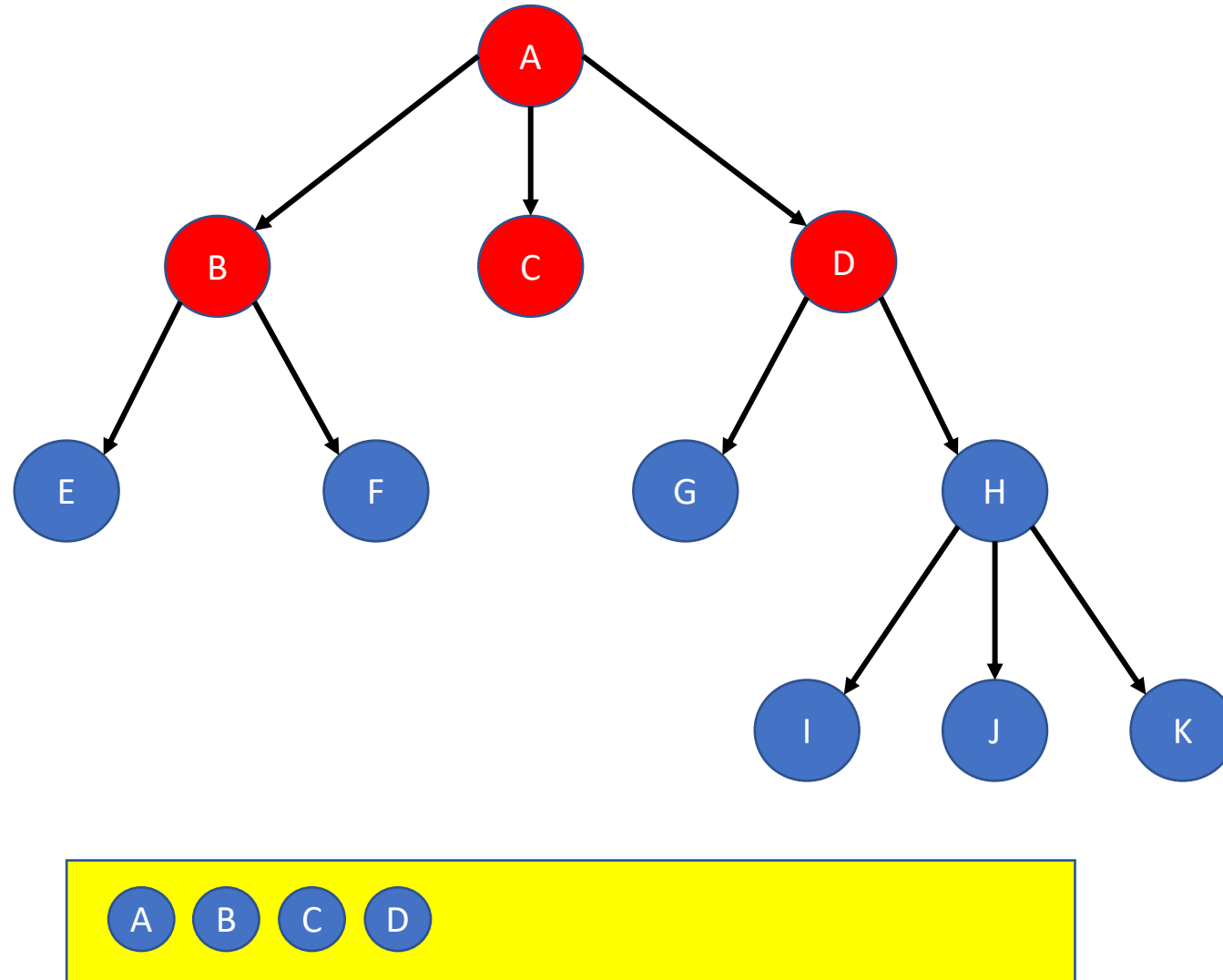
Recorriendo arboles – BFS (Busqueda en amplitud)



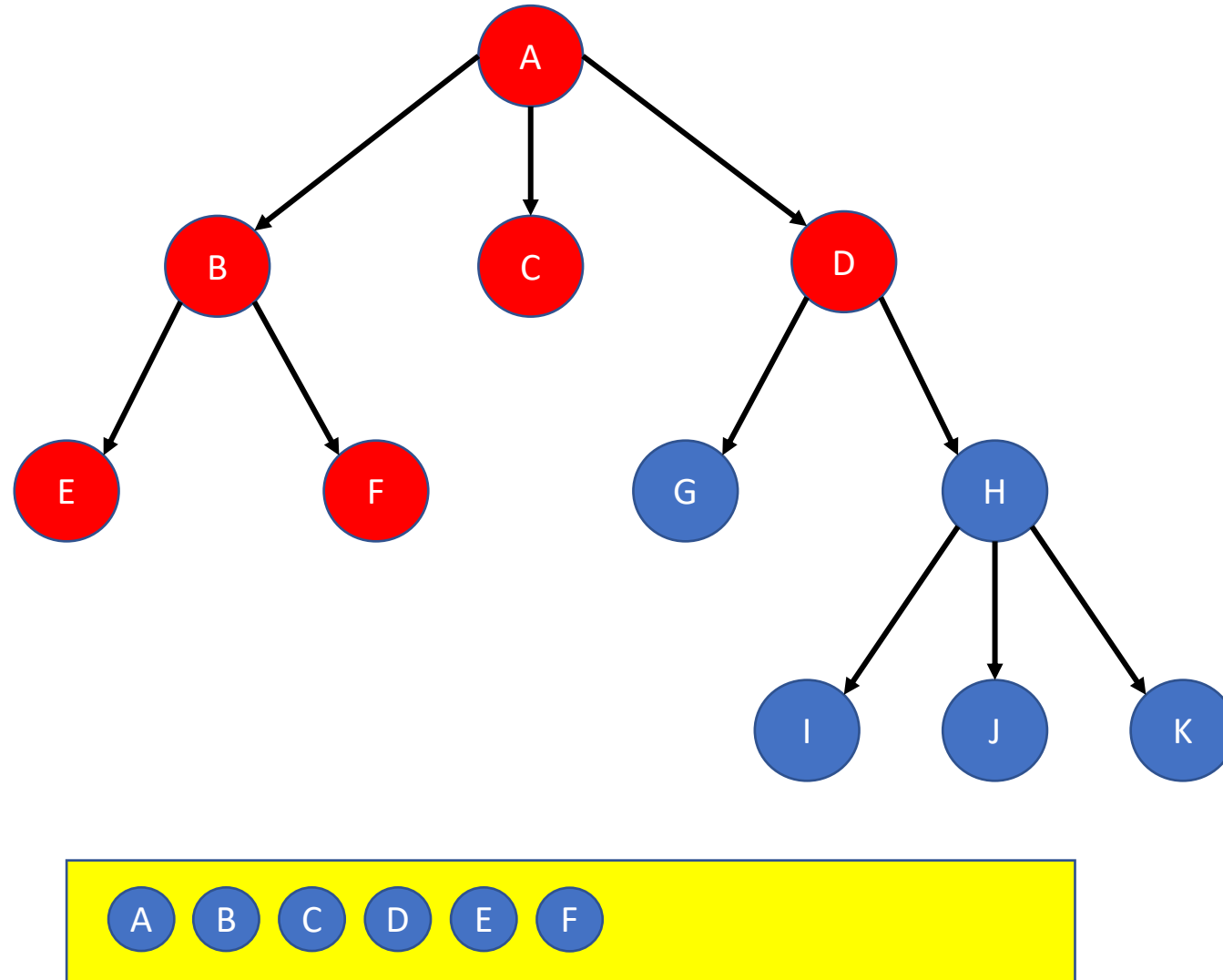
Recorriendo arboles – BFS (Busqueda en amplitud)



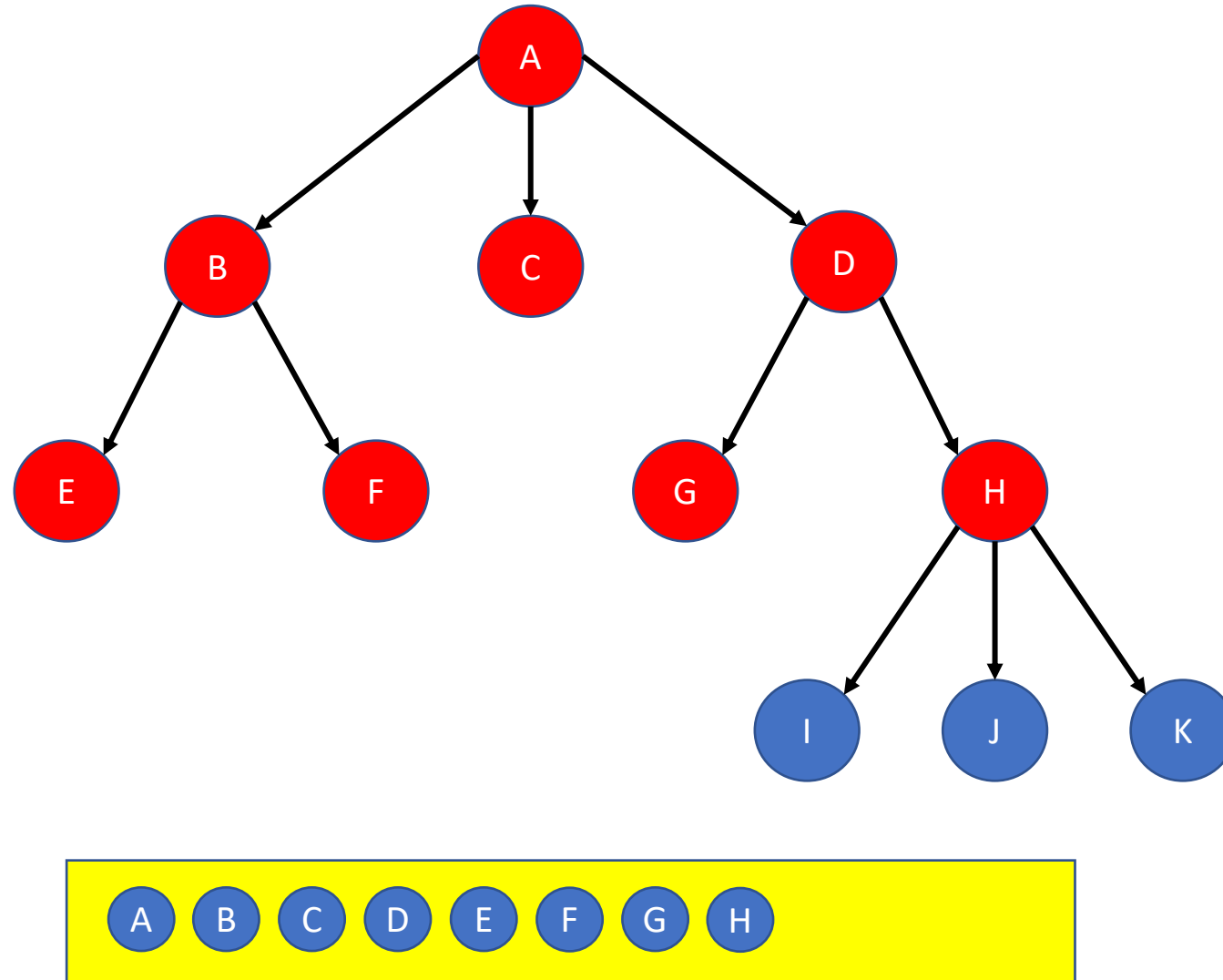
Recorriendo arboles – BFS (Busqueda en amplitud)



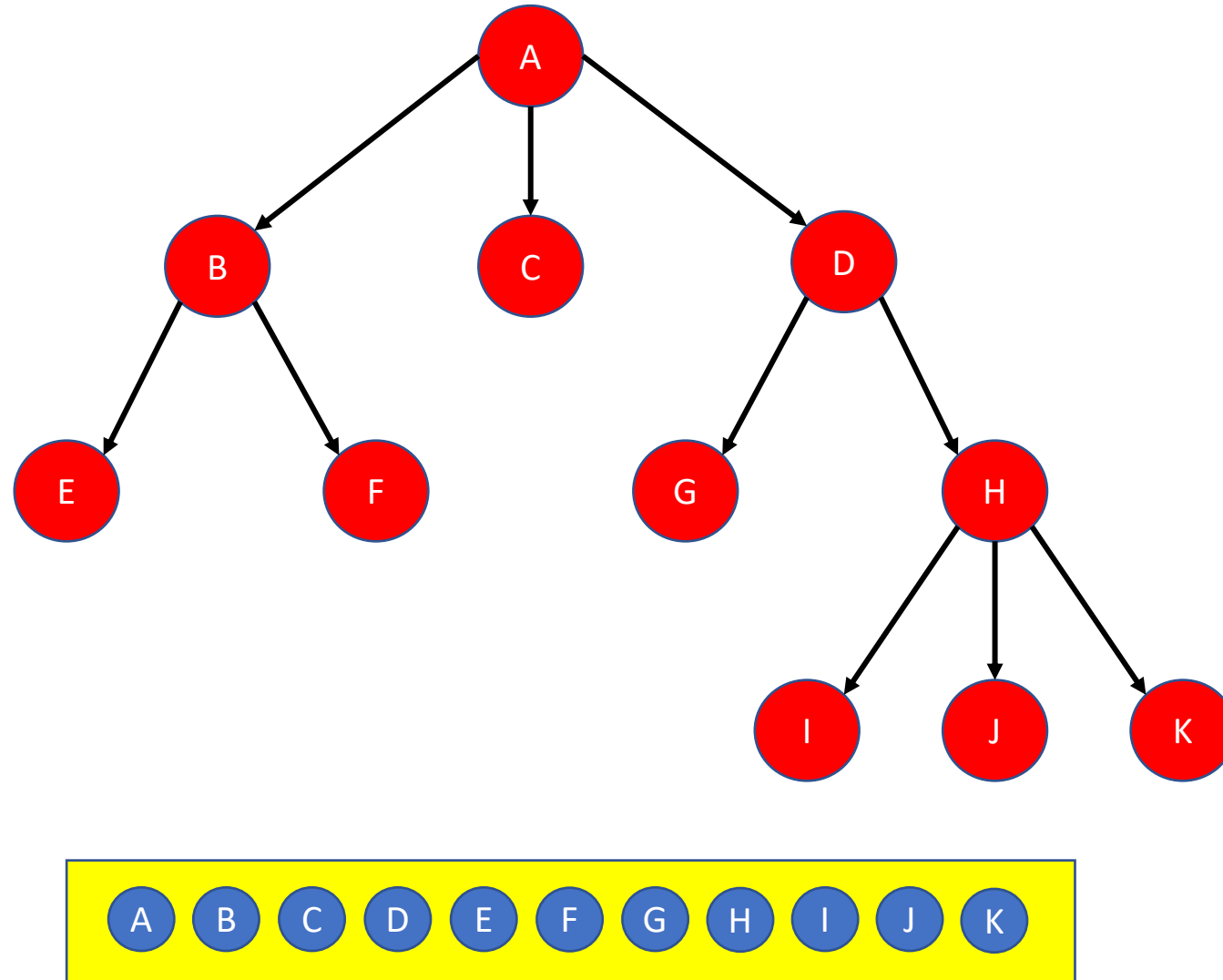
Recorriendo arboles – BFS (Busqueda en amplitud)



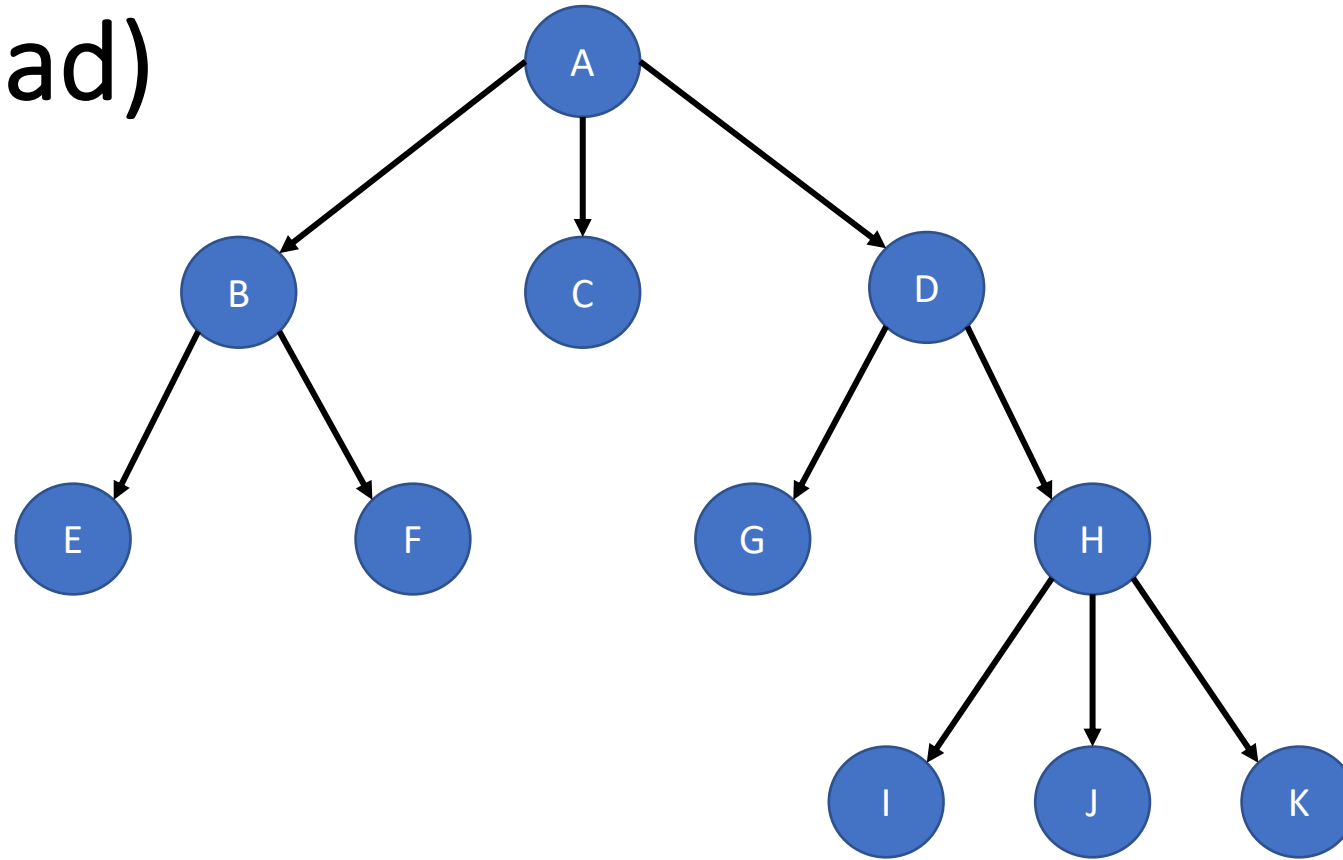
Recorriendo arboles – BFS (Busqueda en amplitud)



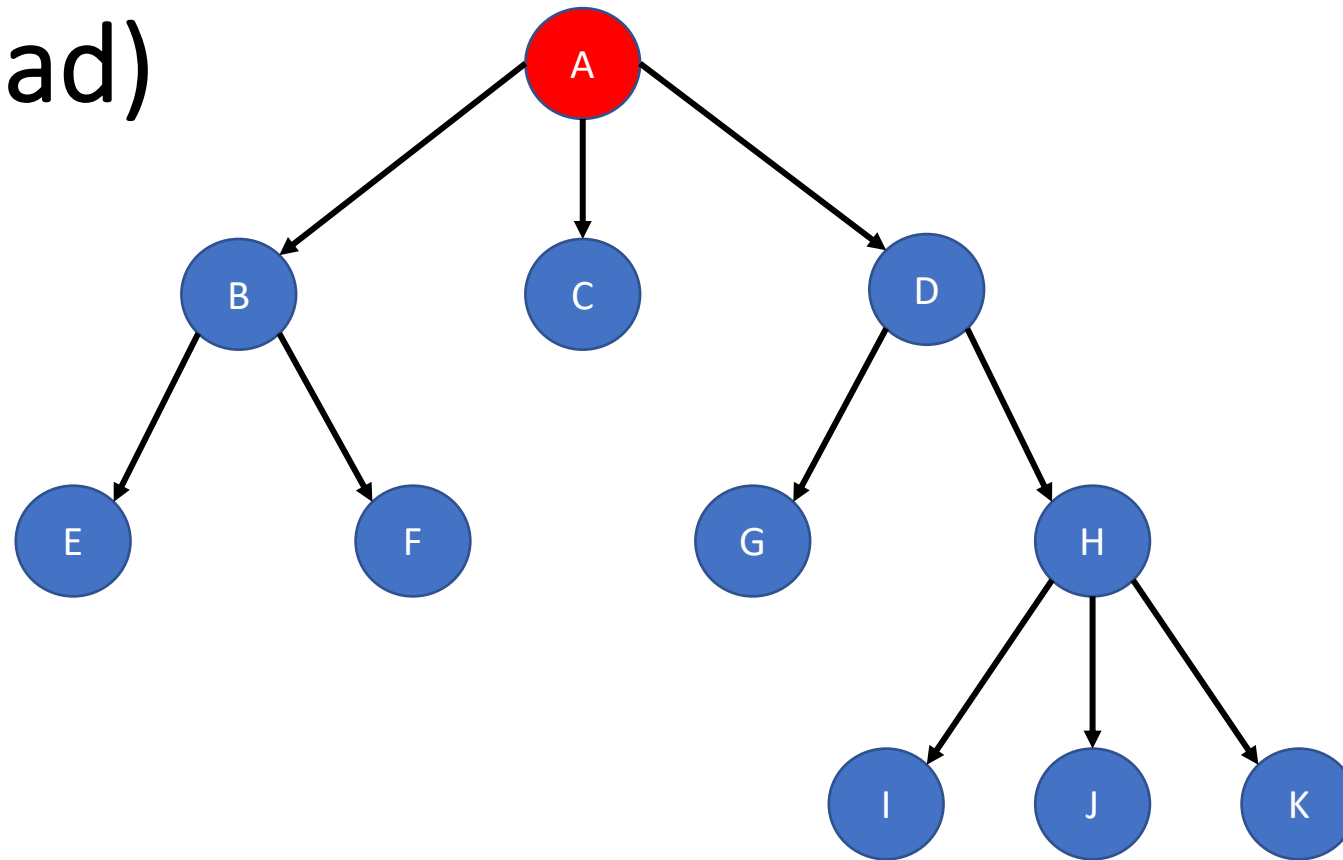
Recorriendo arboles – BFS (Busqueda en amplitud)



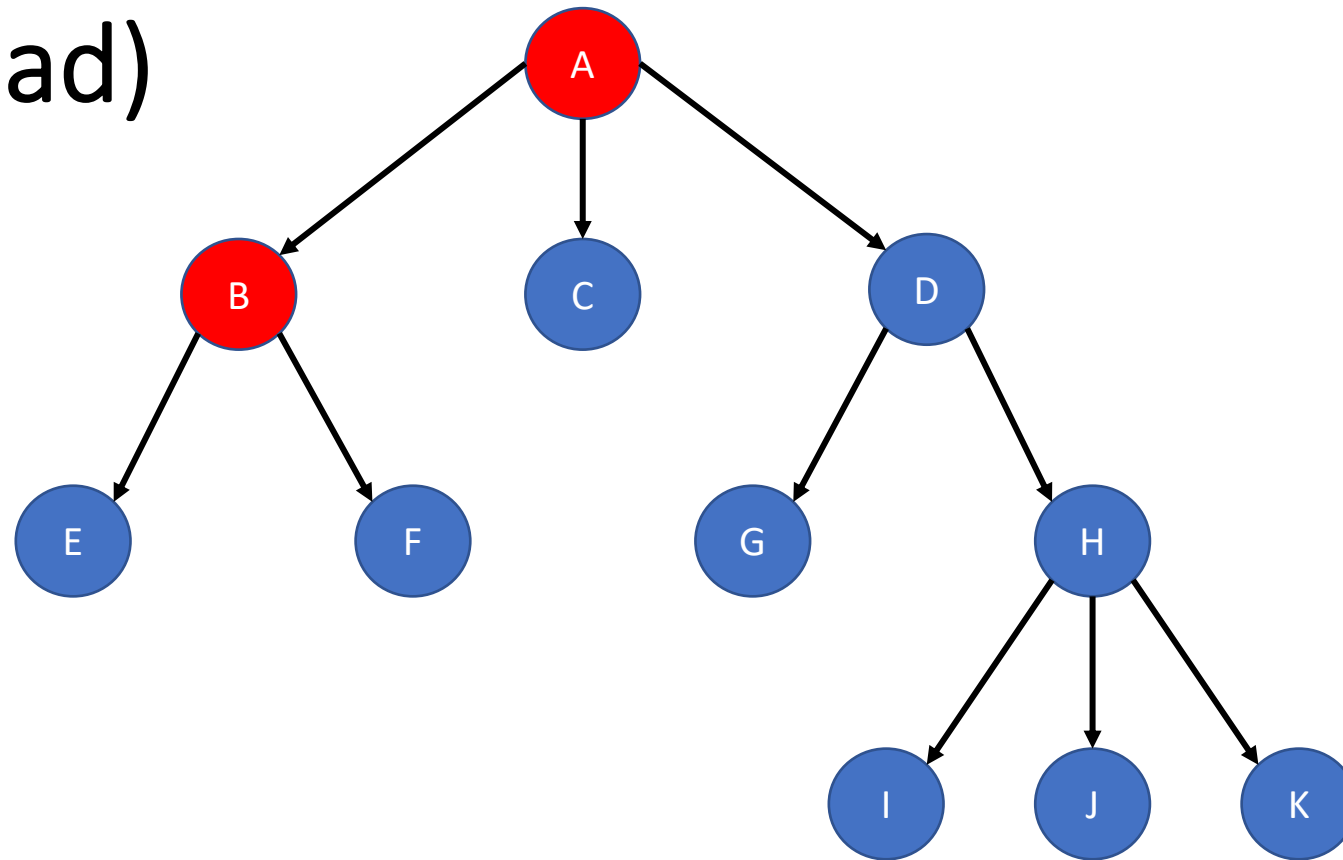
Recorriendo arboles – DFS (Busqueda en profundidad)



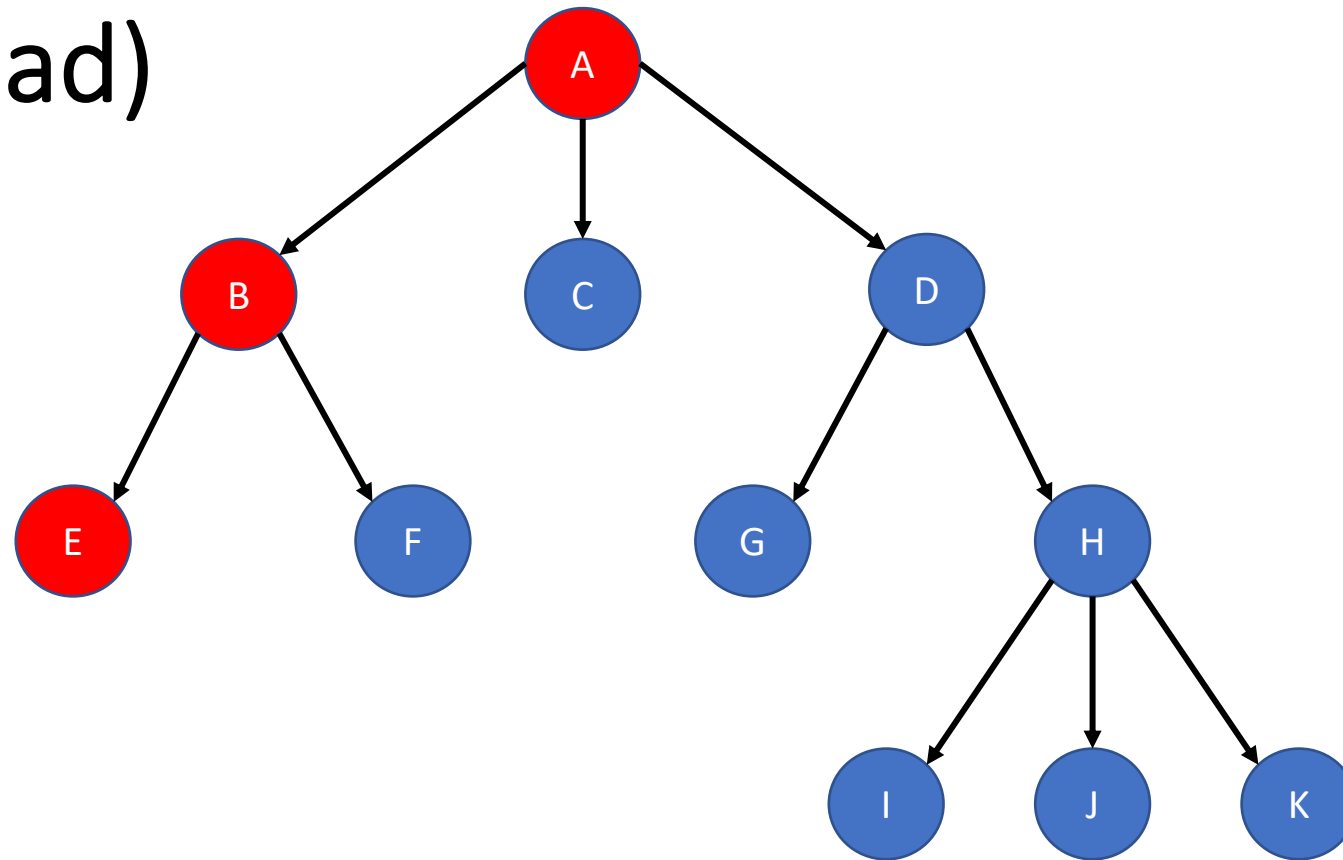
Recorriendo arboles – DFS (Busqueda en profundidad)



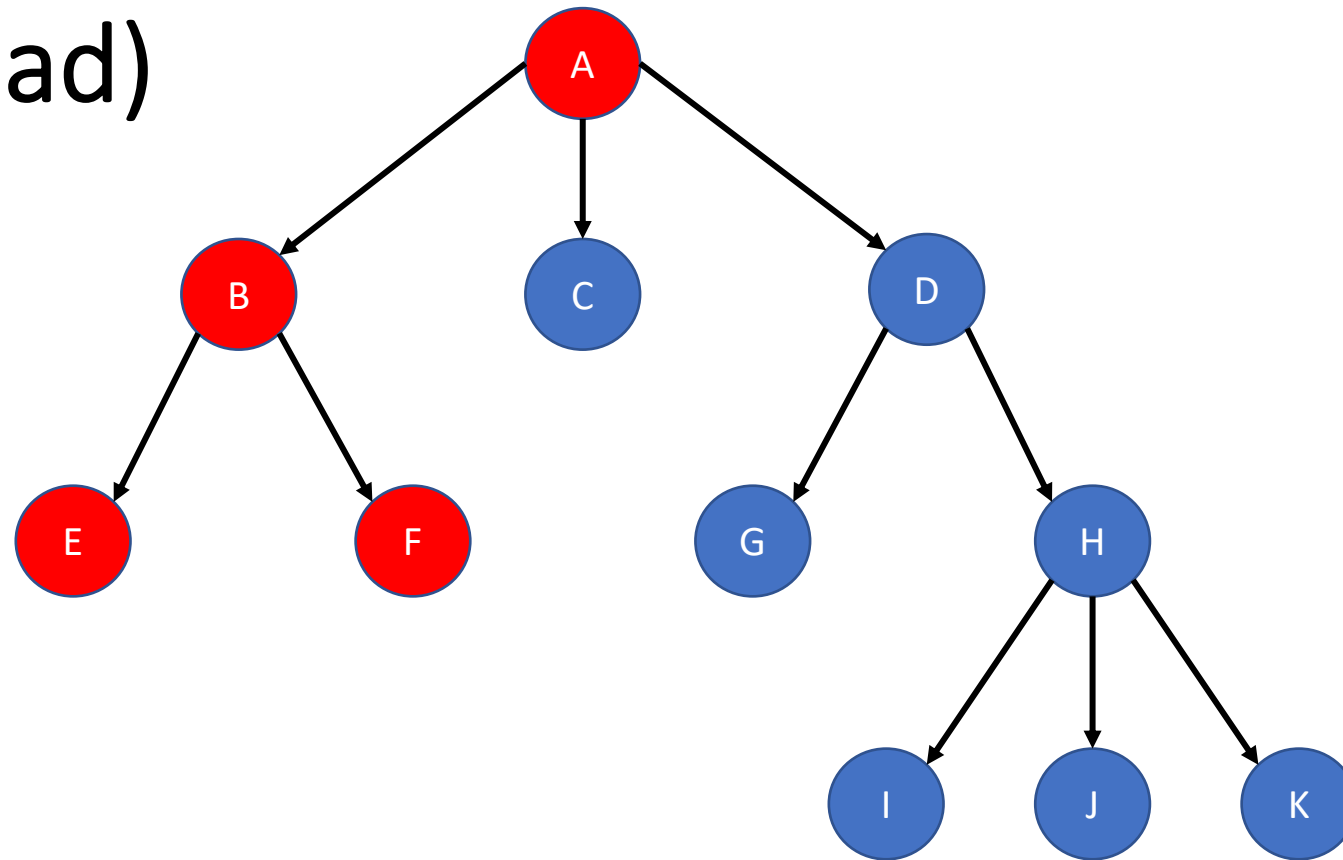
Recorriendo arboles – DFS (Busqueda en profundidad)



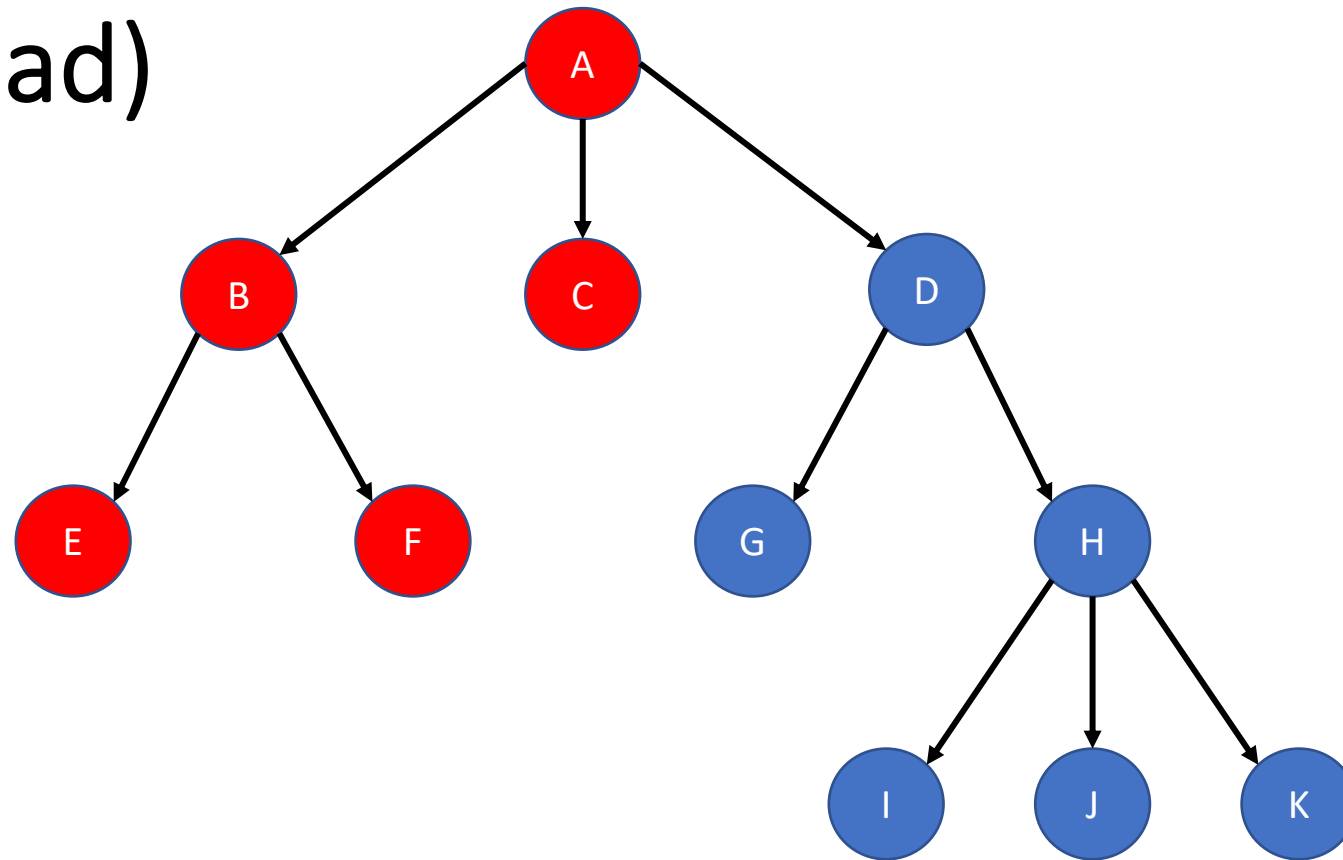
Recorriendo arboles – DFS (Busqueda en profundidad)



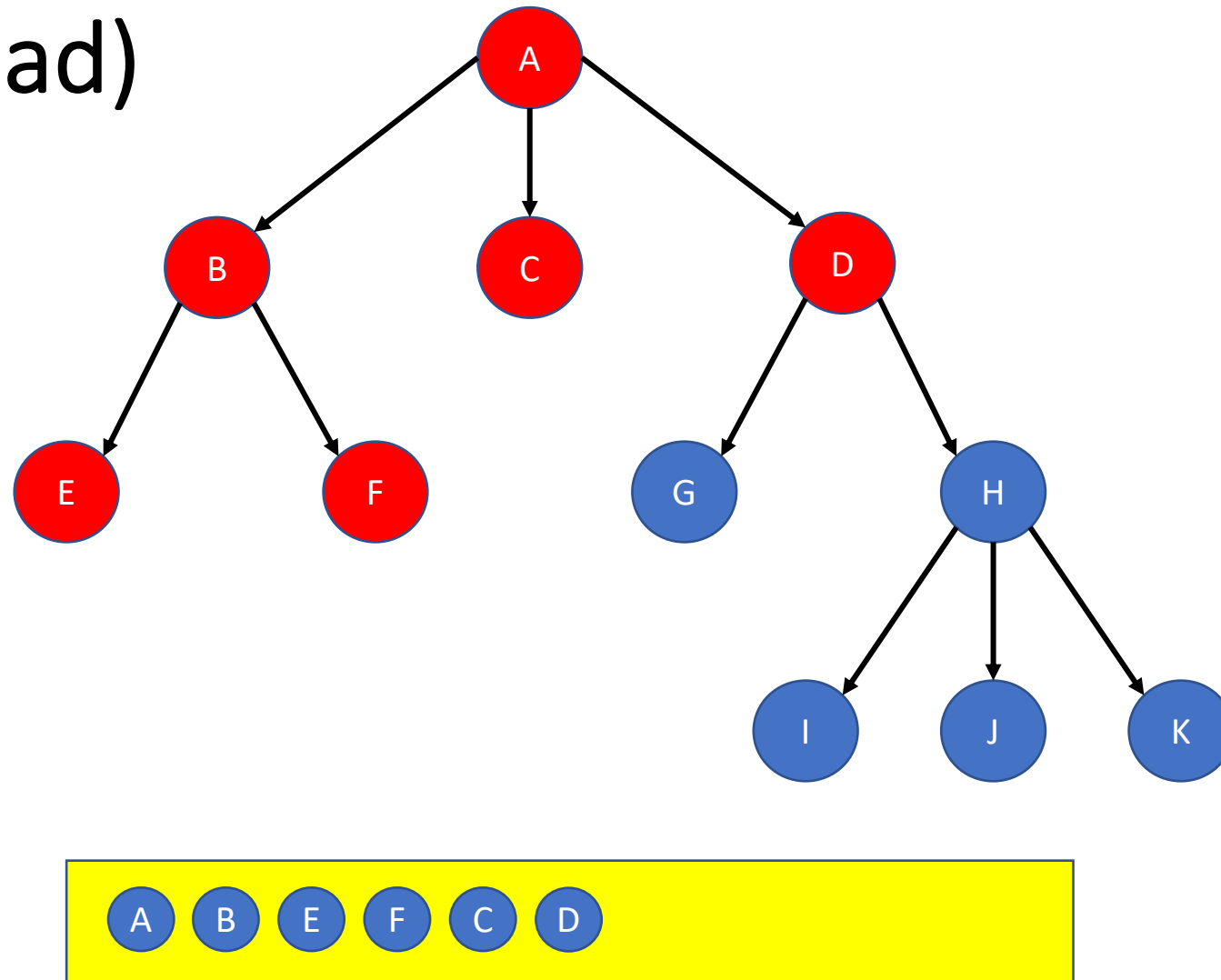
Recorriendo arboles – DFS (Busqueda en profundidad)



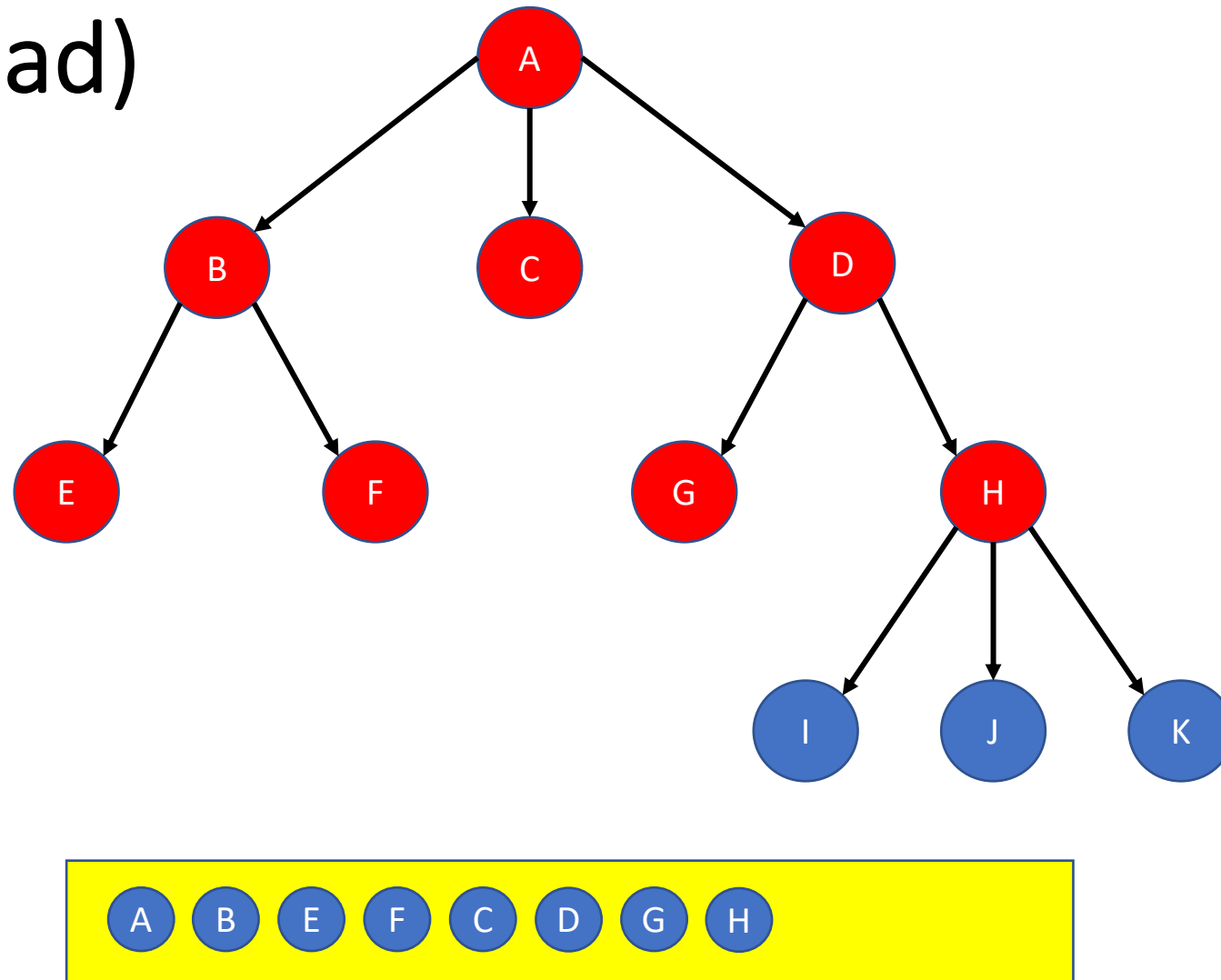
Recorriendo arboles – DFS (Busqueda en profundidad)



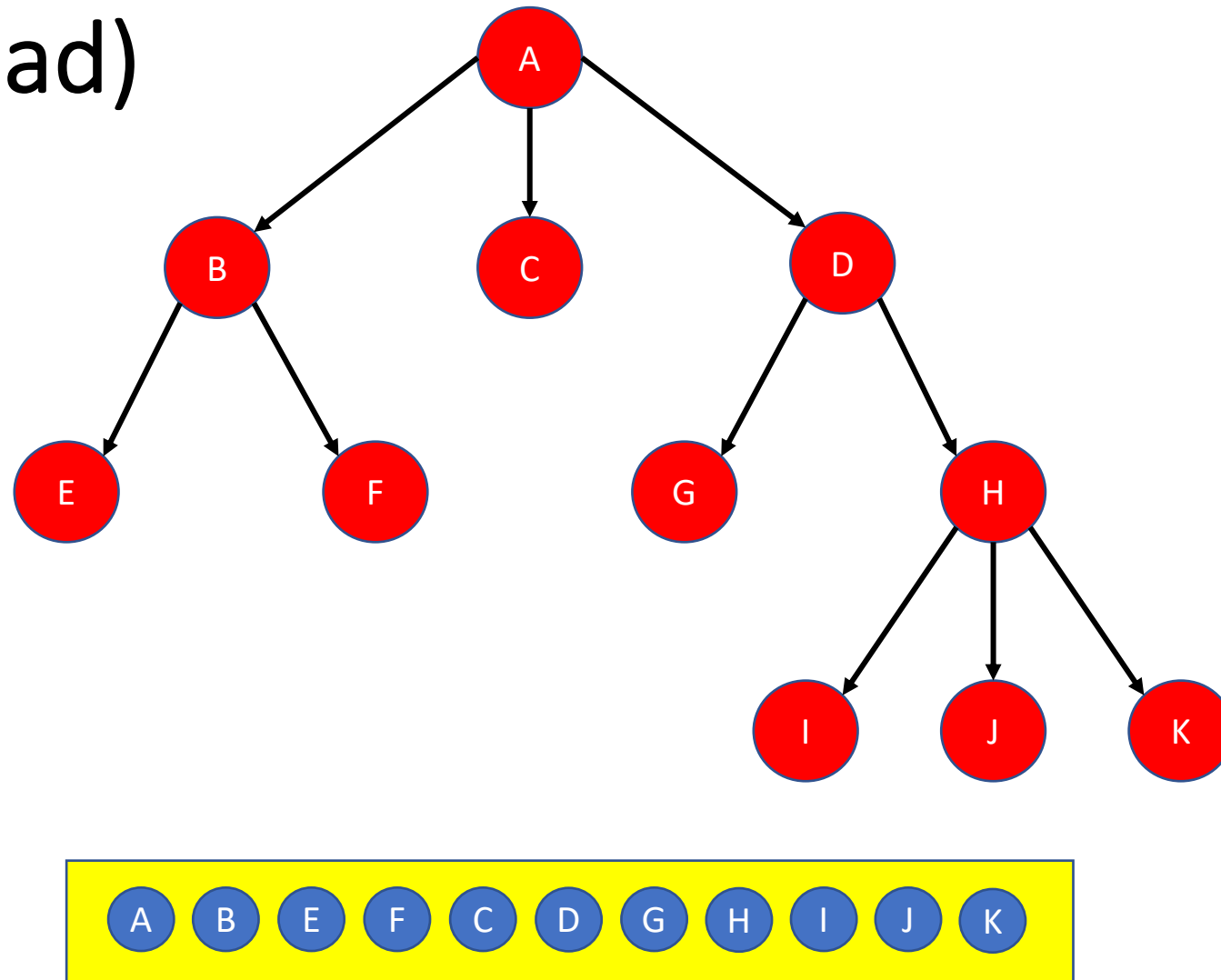
Recorriendo arboles – DFS (Busqueda en profundidad)



Recorriendo arboles – DFS (Busqueda en profundidad)

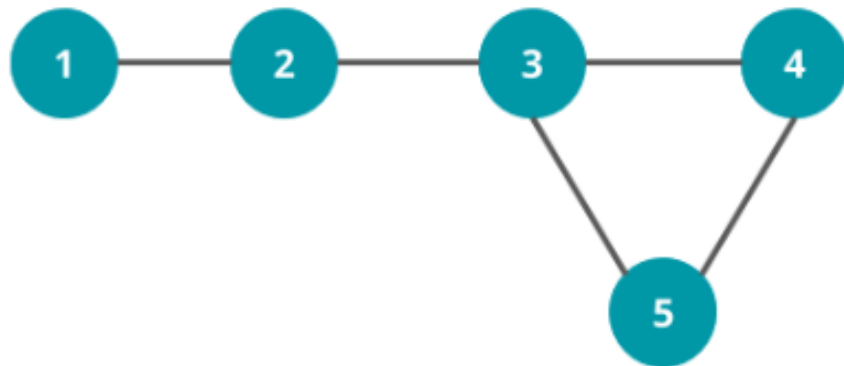


Recorriendo arboles – DFS (Busqueda en profundidad)

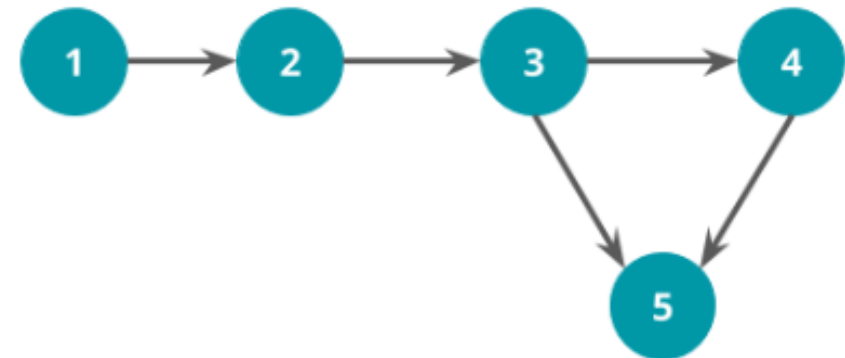


Grafos

Grafo no dirigido



Grafo dirigido



Pensemos algunos ejercicios

“Dada una lista de enteros, retorne la misma lista, pero con todos los 0s al final de esta”

1. Entender el enunciado
2. Darnos ejemplo sencillos para entender la mecánica
3. Pensar formas de abordar el problema
4. Puedo mejorarla?
5. Programo...

Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]

2	0	6	0	4
---	---	---	---	---

k = 0

--	--	--	--	--

Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



k = 0



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



k = 0



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 1$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



k = 1



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



k = 1



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



k = 1



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 2$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 2$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 2$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 2$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$k = 3$



Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$$k = 3.5$$

Demos un ejemplo

Si recibieramos la lista [2,0,6,0,4] debiesemos retornar [2,6,4,0,0]



$$k = 3.5$$

Problema 1

“Dada una cadena de texto que utiliza los parentesis:

() [] { }

determine si se encuentra balanceada o no”

1. Entender el enunciado
2. Darnos ejemplo sencillos para entender la mecanica
3. Pensar formas de abordar el problema
4. Puedo mejorarla?
5. Programo...

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

(

Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

(

Stack

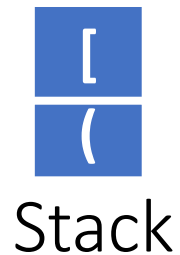
Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***




Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***



Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***


Stack

([]) { }

Si el Stack estuviera vacio...
entonces ya seria... Falso

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***



Stack



Si el Stack estuviera vacio...
entonces ya seria... Falso

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***



Stack

([]) { }

Si el que eliminamos corresponde
al opuesto del que estamos
analizando, seguimos revisando..
Sino, retornamos Falso

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

(

Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

(

Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }



Stack

Demos un ejemplo

*Si recibieramos el texto '()({[]})' debiesemos retornar **True**,
mientras que con '([])' o '({})' **False***

([]) { }

Stack

Seguimos iterando y si al finalizar
el Stack esta vacio, retornamos
True