#CátedrasCiber

Módulo III: Grafos en Ciberseguridad

05/03/2025

Entidad colaboradora:

Financiado por la Unión Europea NextGenerationEU

GOBIERNO DE ESPAÑA

MINISTERIO PARA LA TRANSFORMACIÓN DIGITAL Y DE LA FUNCIÓN PÚBLICA

SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

Plan de Recuperación, Transformación y Resiliencia

incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD

UC Universidad de Cantabria

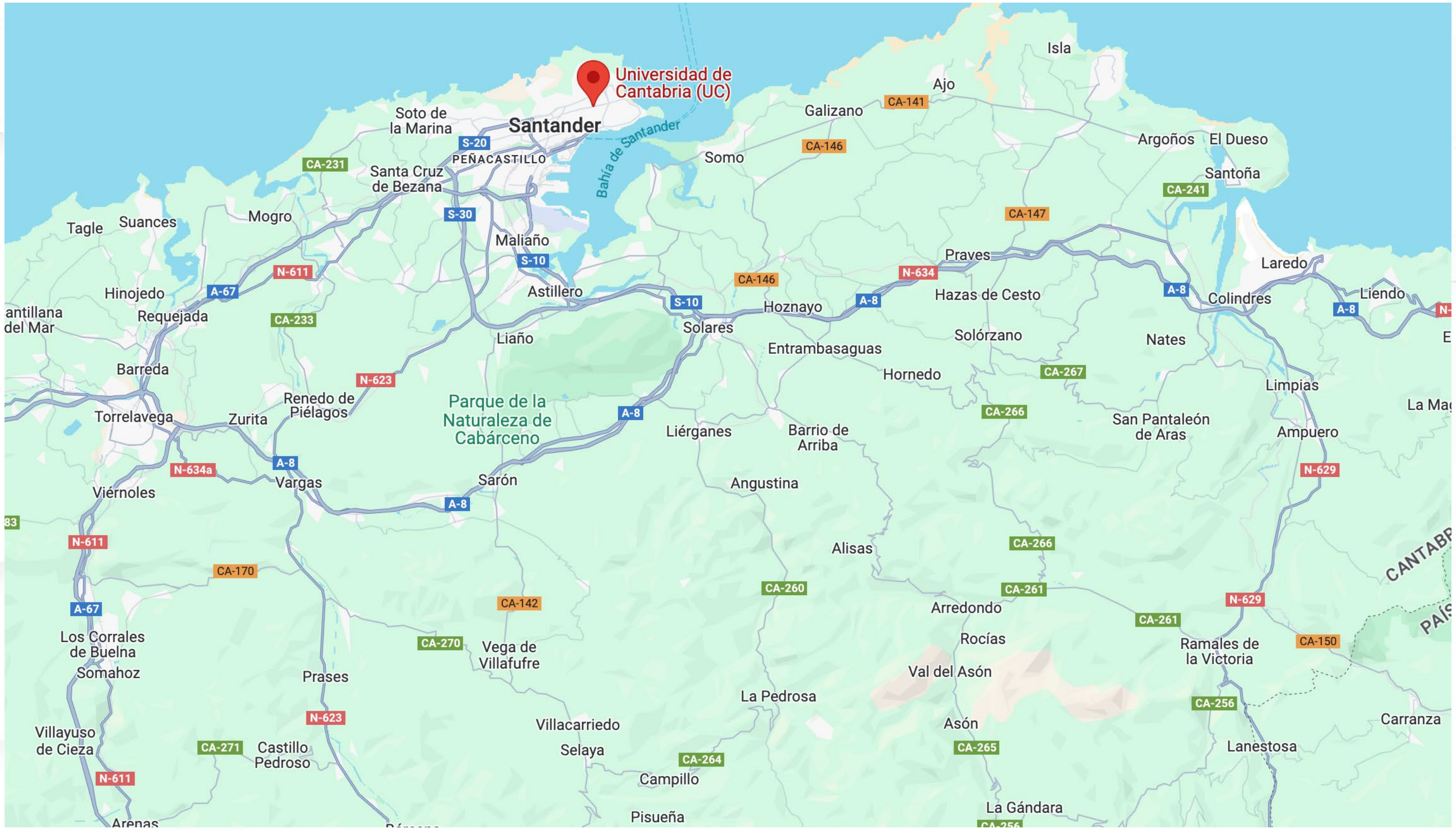
Índice

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

Introducción

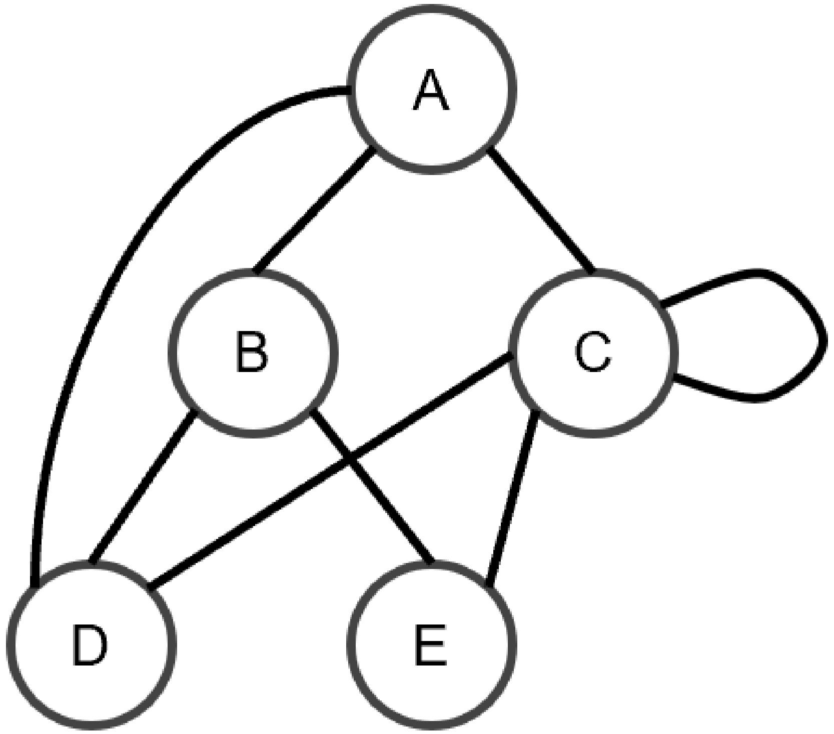
- De forma genérica: los grafos representan un conjunto de entidades y sus relaciones



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

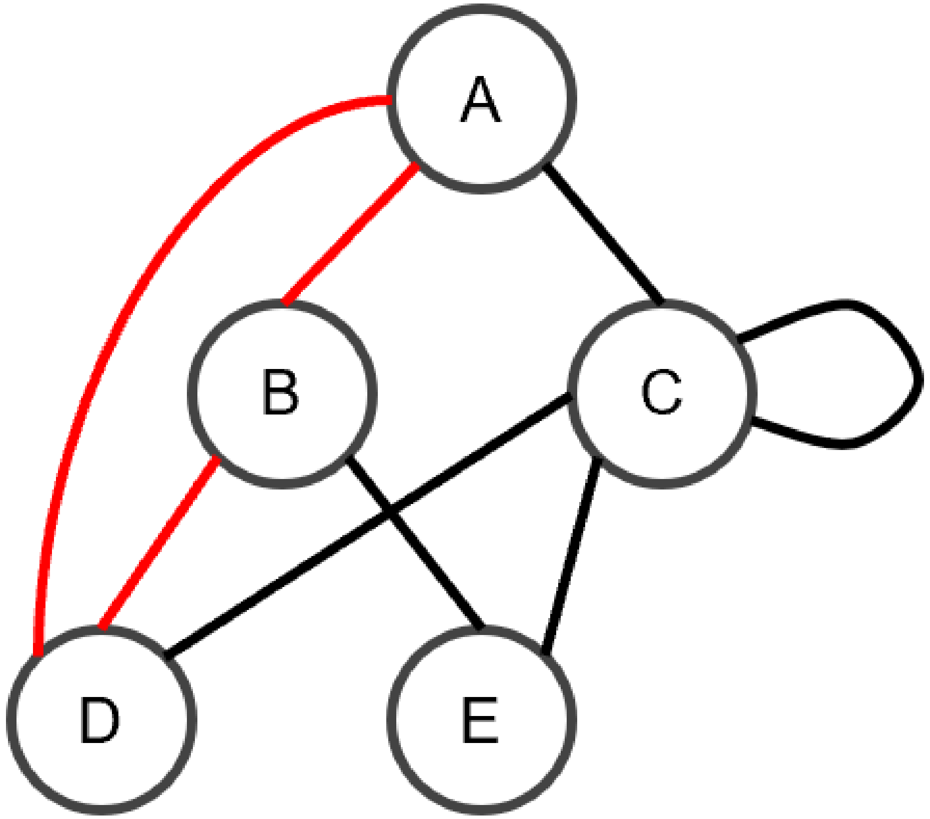
- De forma genérica: los grafos representan un conjunto de entidades y sus relaciones.
- A las entidades se les llama **nodos**, a las relaciones **aristas**



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

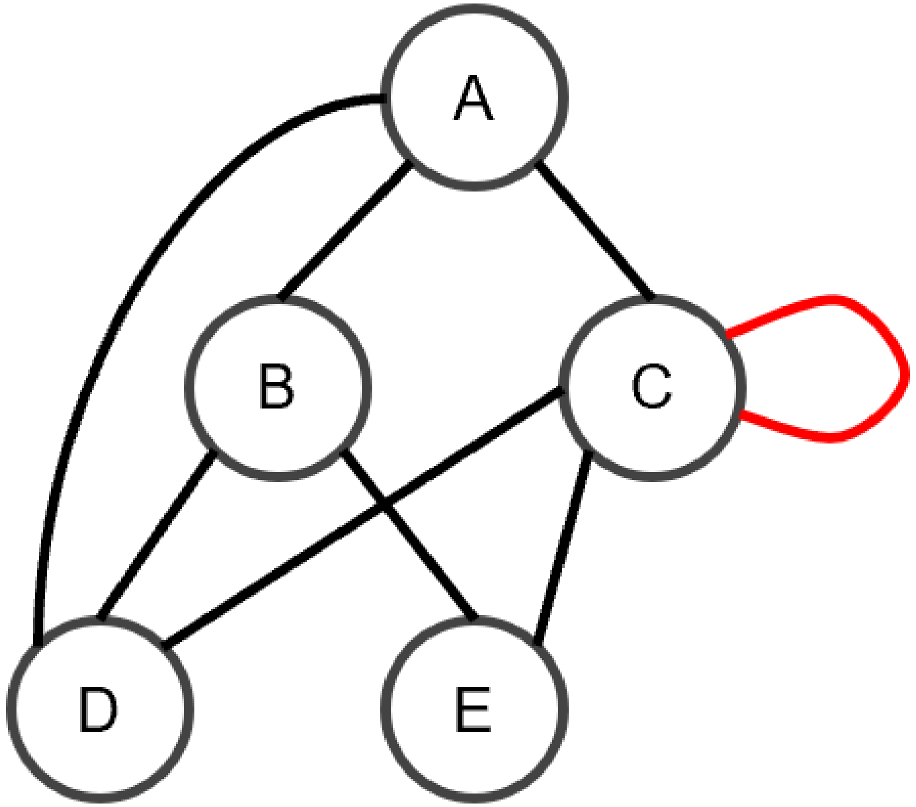
- Podemos tener ciclos



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

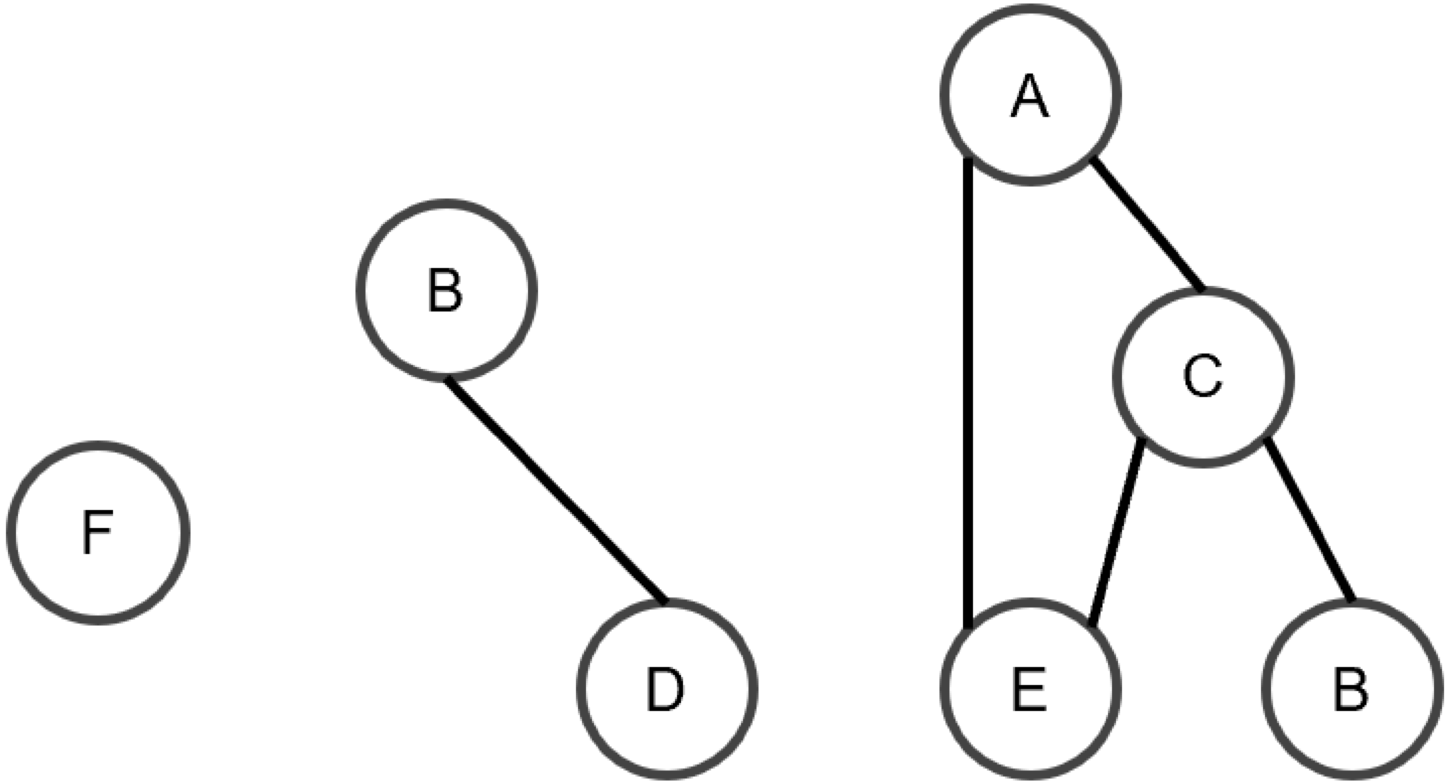
- Podríamos tener un elemento conectado consigo mismo (aunque es poco frecuente, es una posibilidad)



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

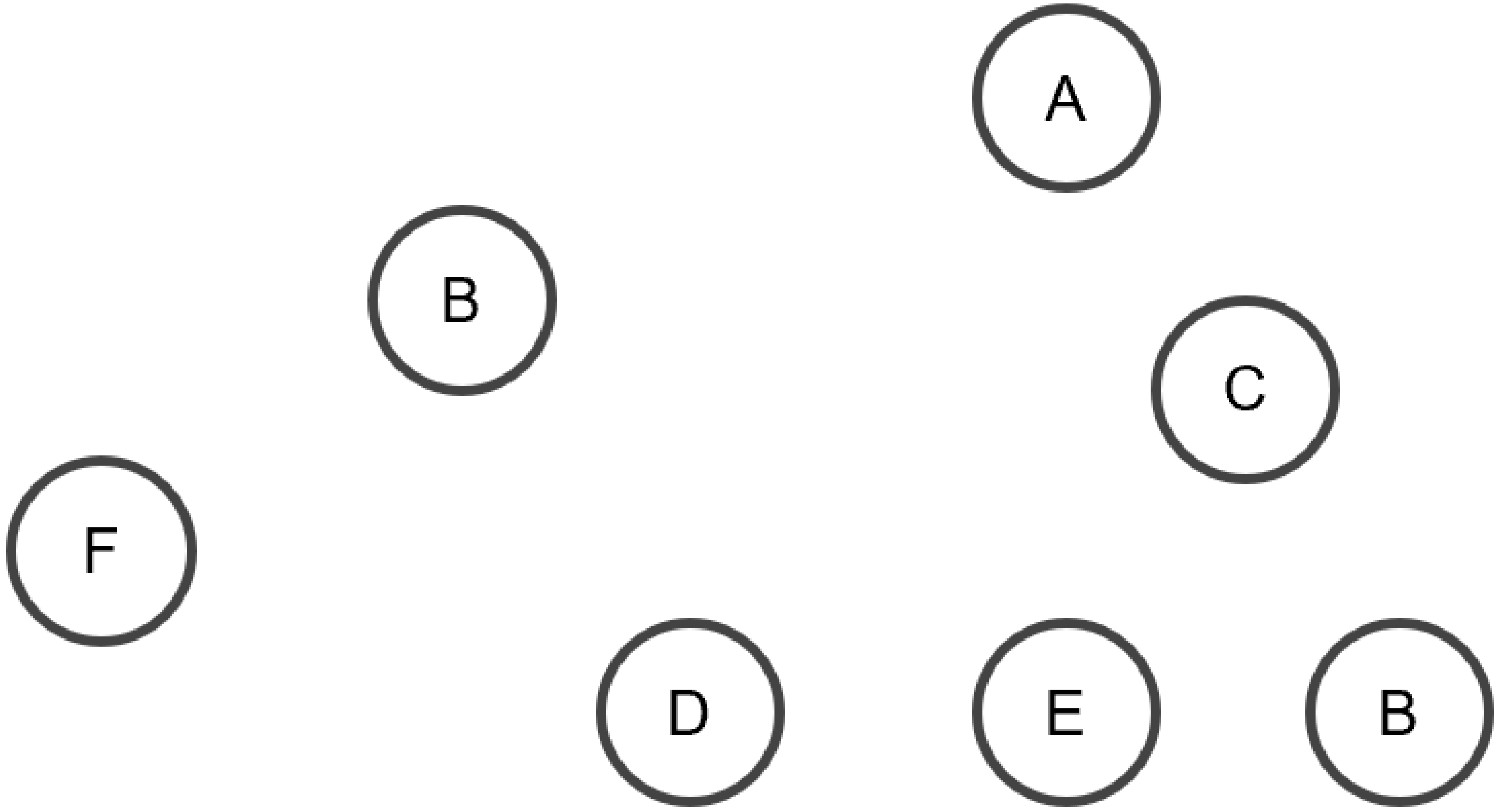
- Puede que existan nodos aislados, de forma que no es posible relacionarlos con otros nodos



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- Incluso es válido que no haya ninguna relación entre las entidades
  → todos los nodos tienen 0 aristas



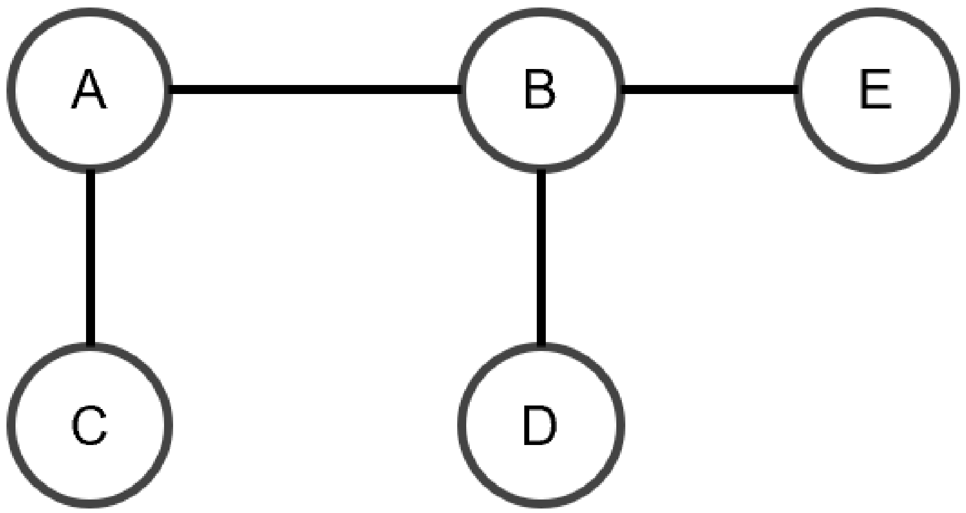
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- Definición formal: G = (**V**,**E**)

  - V es el conjunto de vértices
    V={A,B,C,D,E}

  - E es el conjunto de aristas
    E={(A,B),(A,C),(B,D),(B,E)}

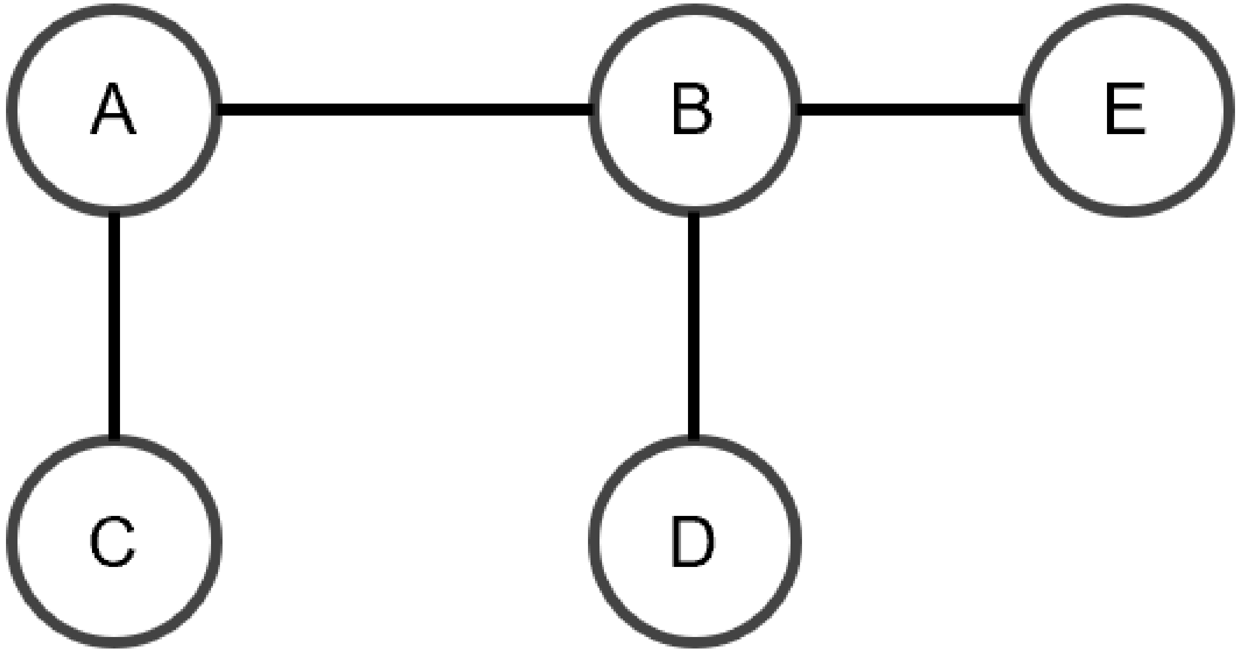


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- Los grafos pueden ser **dirigidos**, si las aristas tienen dirección, o **no dirigidos**, si (A,B) ⇔ (B,A)

- En este ejemplo, como A está conectado a B, B está conectado a A.

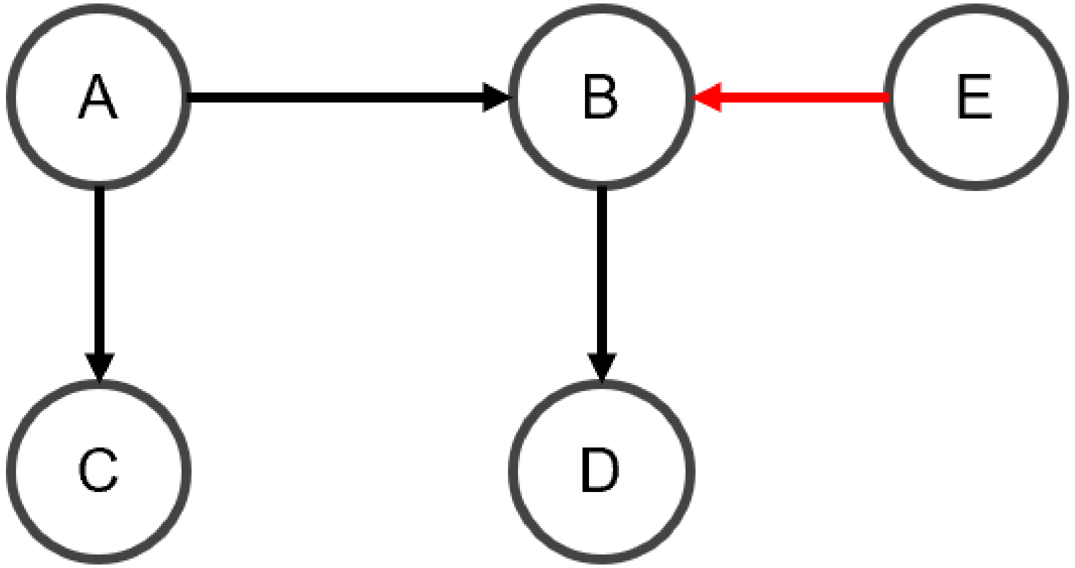


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- Si las aristas tienen dirección (representado como flechitas), el grafo es dirigido.

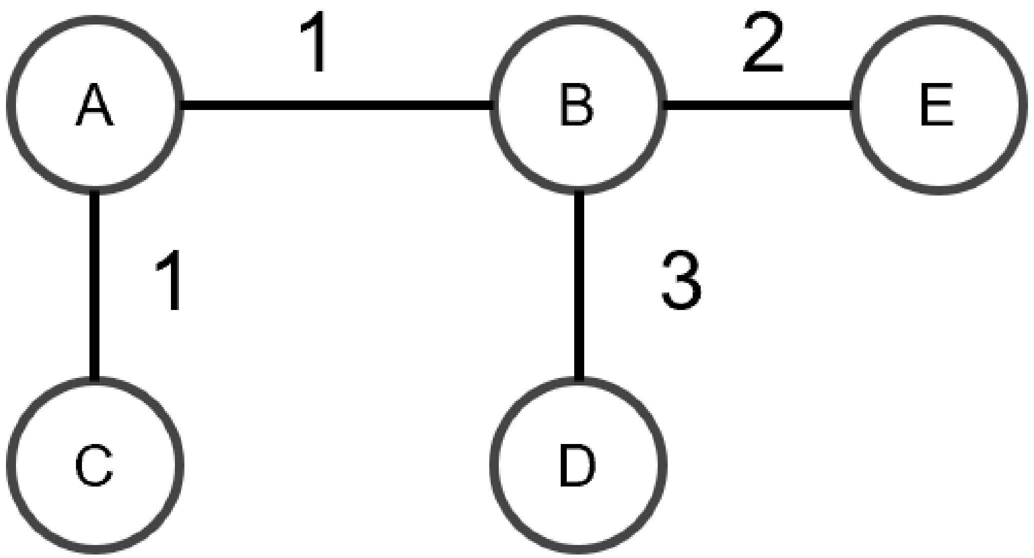
- En este ejemplo, que E esté conectado con B, no implica que B esté conectado con E.



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- Hasta ahora, las aristas implican si existe conexión o no. Además, pueden **ponderar** el tipo de conexión.

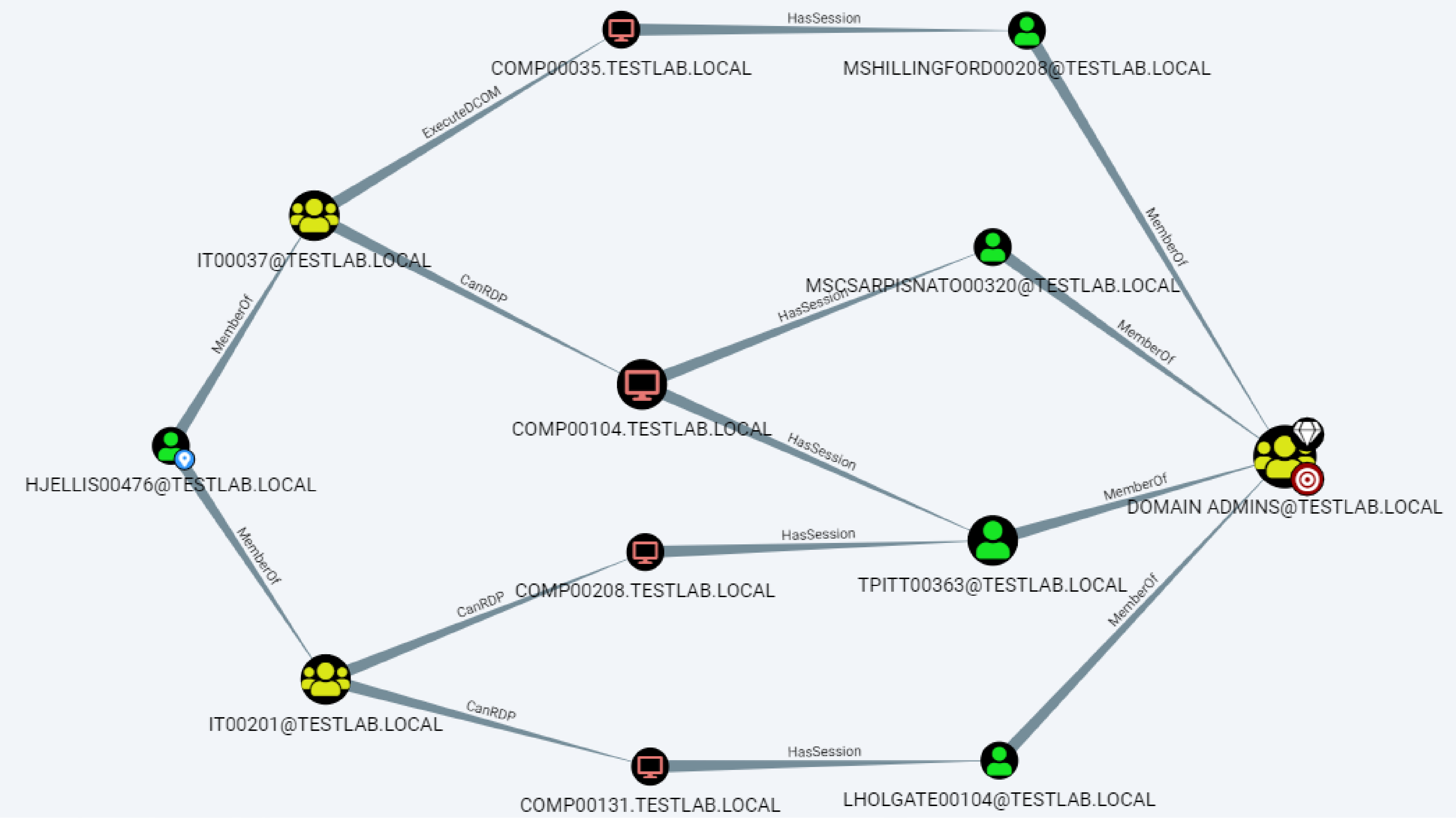


- **Ejemplo**: en un mapa, además de saber si dos ciudades están conectadas, nos puede indicar la **duración del trayecto** correspondiente.

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

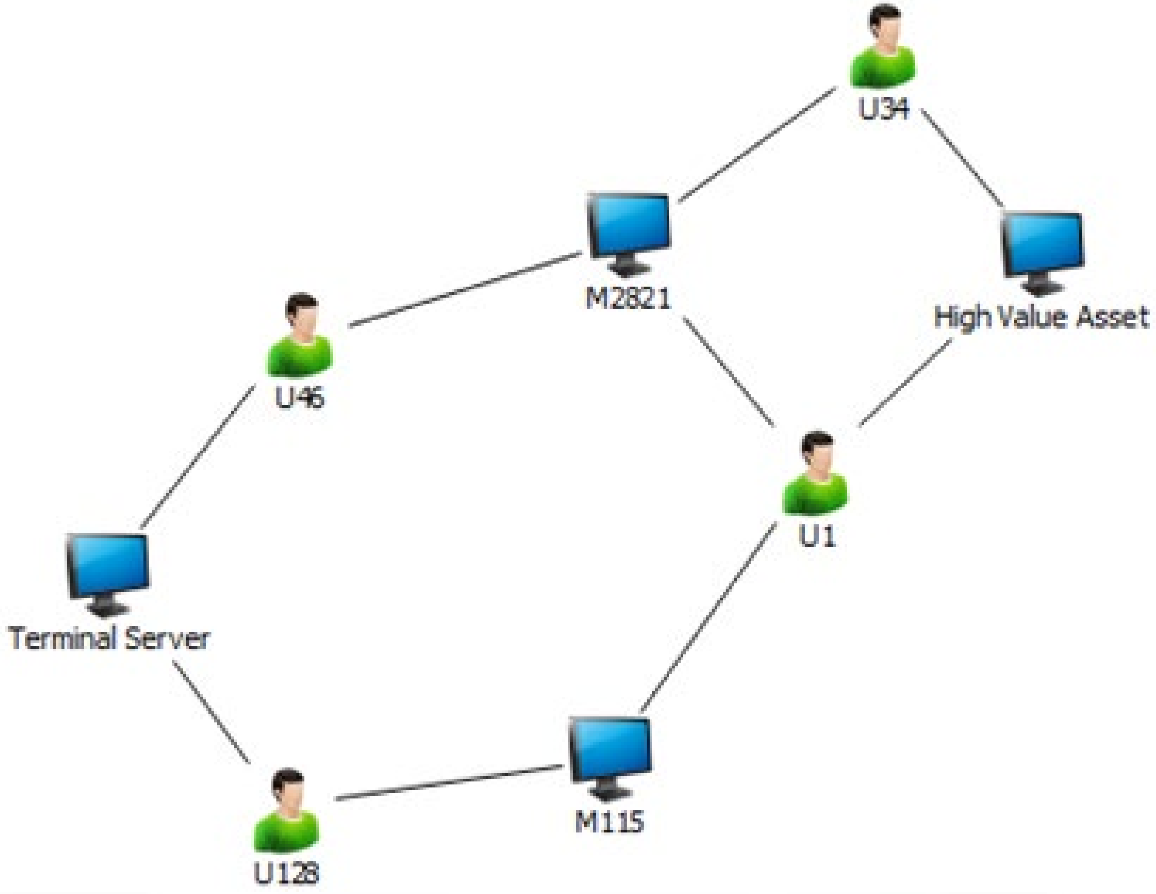
- Ejemplo clásico de ciberseguridad: **relaciones en directorio activo.**



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Introducción

- **“Defenders think in lists. Attackers think in graphs. As long as this is true, attackers win.”** @JohnLaTwC



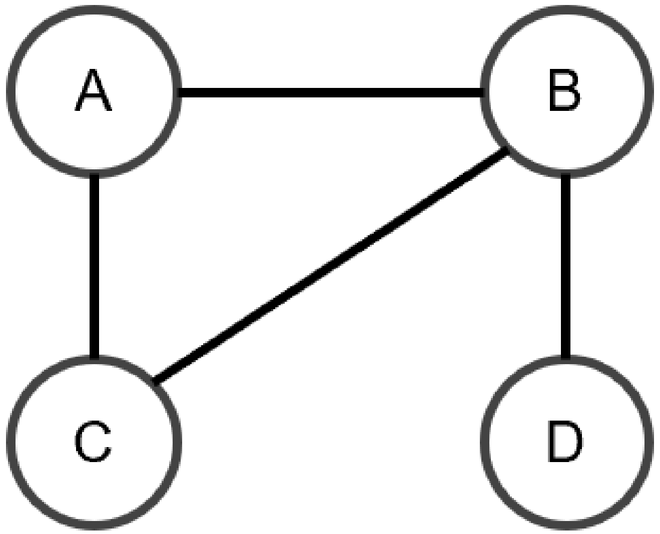
- Lectura importante:
  https://github.com/JohnLaTwC/Shared/blob/master/Defenders%20think%20in%20lists.%20Attackers%20think%20in%20graphs.%20As%20long%20as%20this%20is%20true%2C%20attackers%20win.md

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación

Implementación

- **Matriz de adyacencia**: array de dos dimensiones.

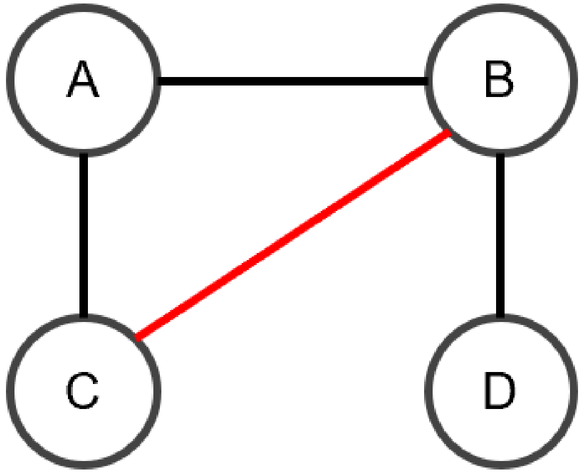


	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	0
D	0	1	0	0

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación

- Si el grafo es no ponderado y no dirigido: existencia de arista se puede representar como 1 / 0 o true / false.



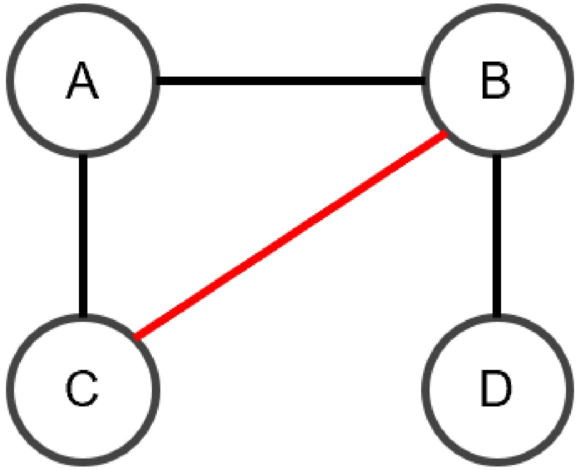
	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	0
D	0	1	0	0

- ¿Se me olvida algo?

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación

- Si el grafo es no ponderado y no dirigido: existencia de arista se puede representar como 1 / 0 o true / false.



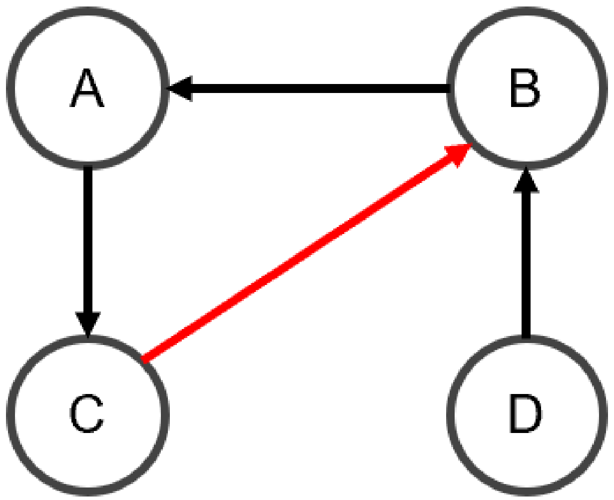
	A	B	C	D
A	0	1	1	0
B	1	0	1	1
		1	0	0
D	0	1	0	0

- No dirigido: si B conecta con C, entonces C conecta con B.

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación

- Si el grafo es **dirigido**, la matriz **no es simétrica**



	A	B	C	D
A	0	0	1	0
B	1	0	0	0
C	0	1	0	0
D	0	1	0	0

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación

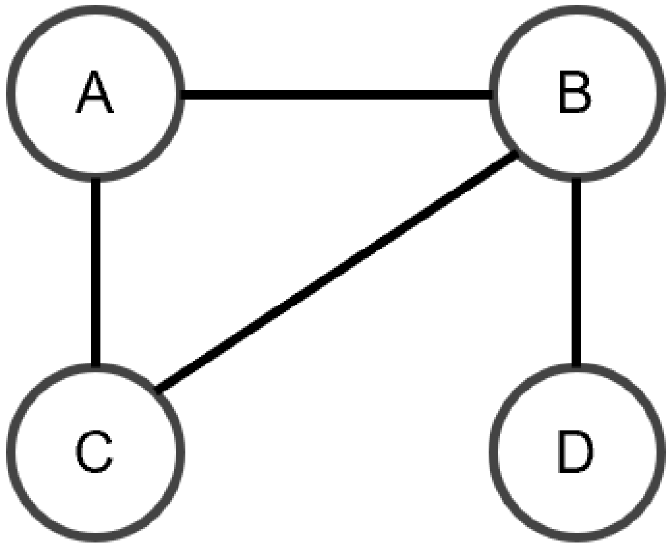
- La matriz de adyacencia es simple de implementar, pero **el consumo de memoria escala muy mal**:
  - Ej: Si cada arista consume 1 byte, como es un array de dos dimensiones:
    - 100 nodos → 100*100 = 10KB RAM
    - 1000 nodos → 1MB RAM
    - 1MK nodos → 1000GB RAM

  - ¿Existe alguna forma de almacenar solo las conexiones que necesitamos?

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación 2

- **Listas de adyacencia**: enumeramos las aristas de cada vértice.
  - En función del lenguaje: mapa de listas, diccionario de sets, etc.



A	⇒	{B,C}
B	⇒	{A,C,D}
C	⇒	{A,B}
D	⇒	{B}

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Implementación 2

- **Listas de adyacencia**: consumo de memoria menor en grafos dispersos: |V| + |E|
  - |V|: número de vértices (las entradas del mapa/diccionario)
  - |E|: número de aristas (cada elemento almacenado en la lista).

Ejemplo Python 3:

```
from collections import defaultdict
grafo = defaultdict(set)
# Conectar A y B
graph[A].add(B) # Si es no dirigido: Tambien B con A
# Mirar si B y C estan conectados
conectaCyB = C in graph[B]
```

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsqueda

Búsquedas

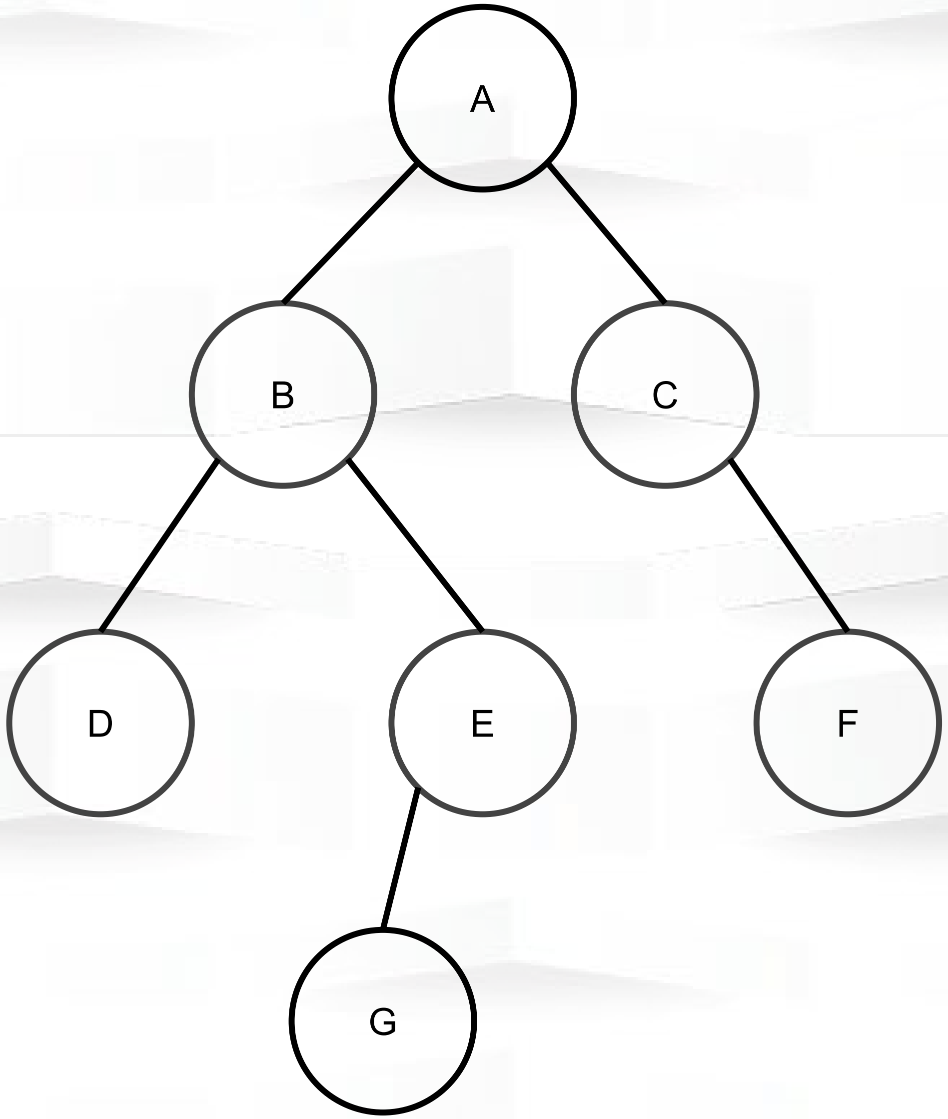
- **BFS:** Breadth First Search
  - Búsqueda en anchura
- **DFS:** Depth First Search
  - Búsqueda en profundidad

- **¡Importante!**
  - Los grafos pueden tener ciclos. Si no **controlamos los nodos ya visitados**, es muy fácil entrar en bucles infinitos.
  - Los grafos pueden tener “nodos sueltos” (componentes no conexas). **Explorar desde 1 nodo puede no ser suficiente.**

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

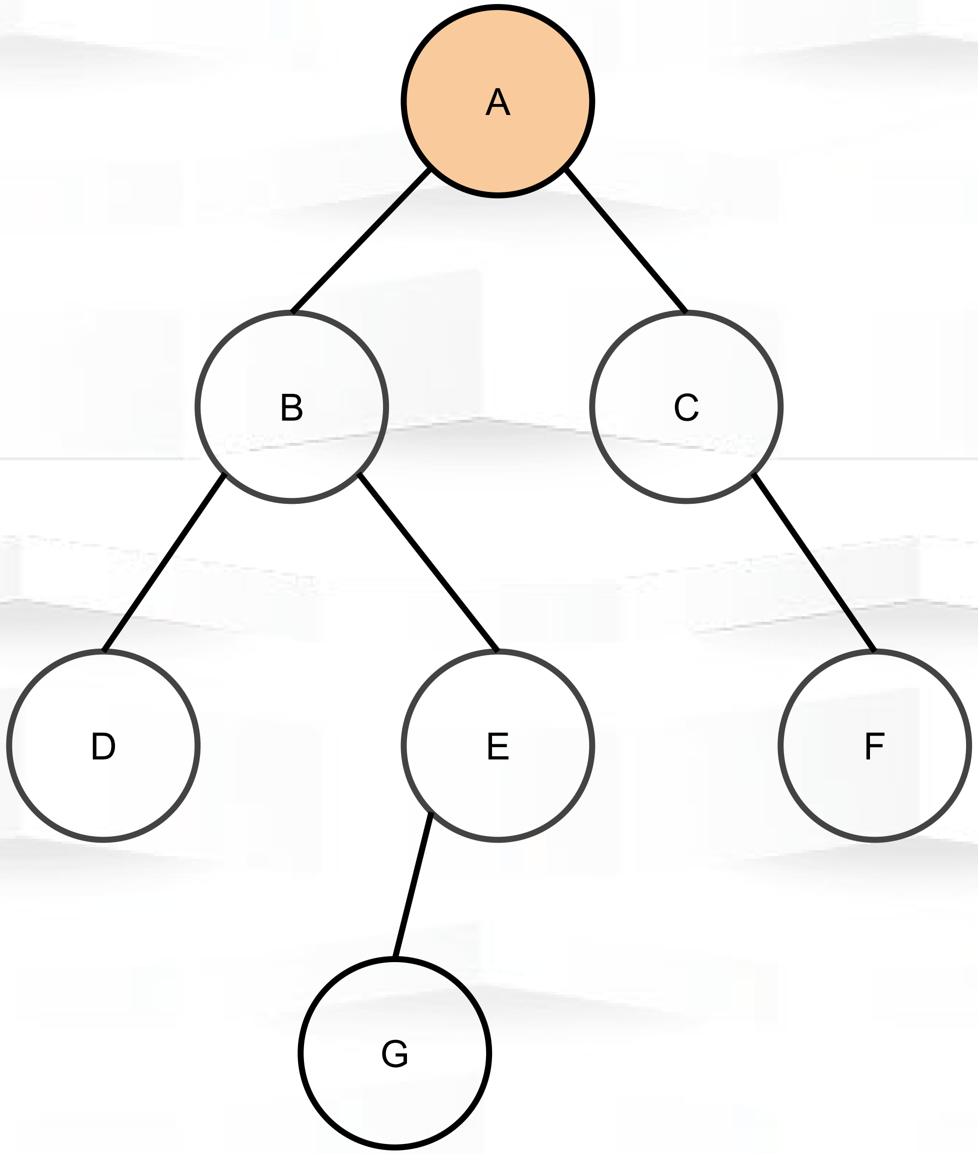


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura
  - Ej: empezamos desde A
  - Vamos recorriendo por "niveles" (distancia al nodo inicial)

{A}

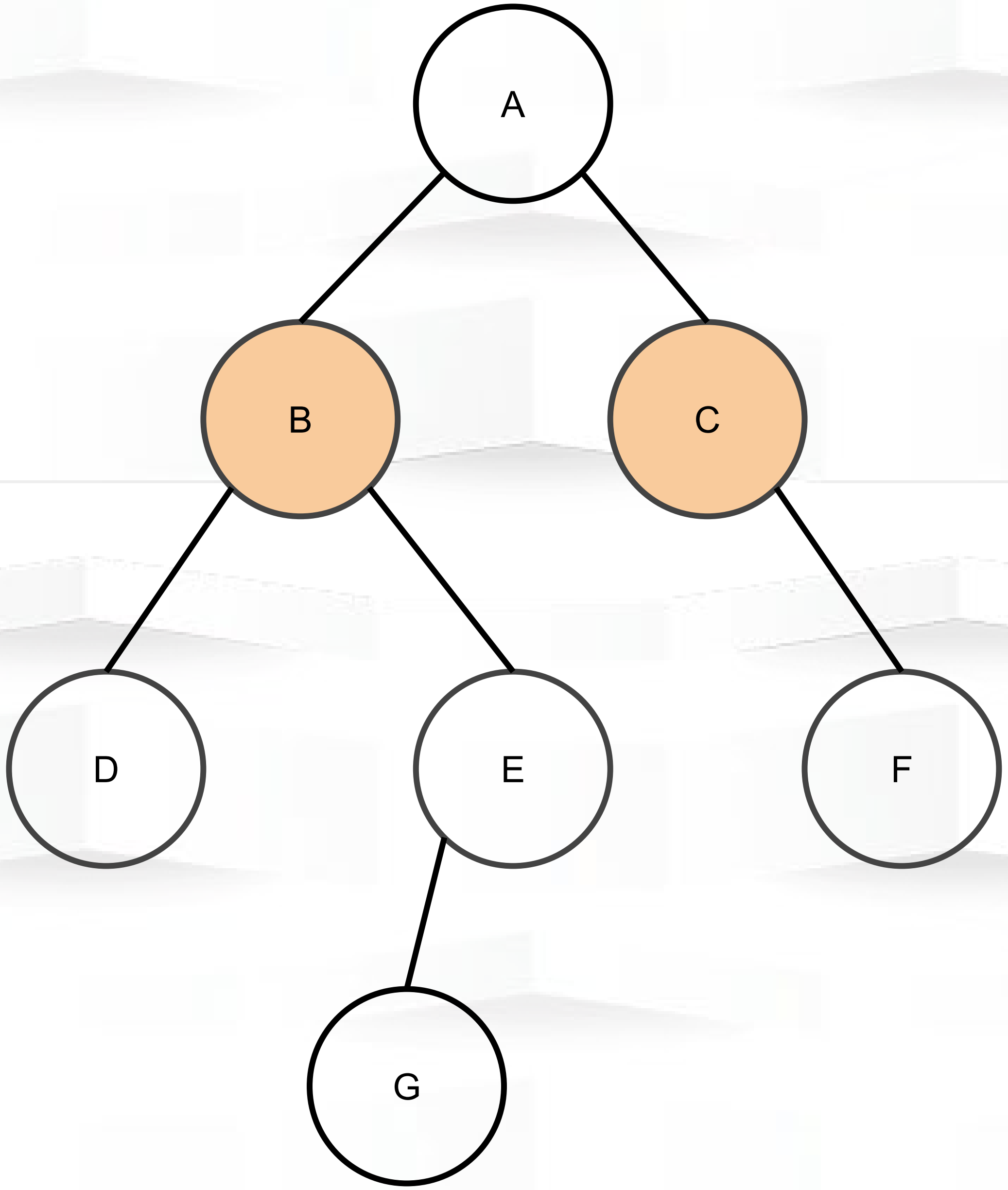


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

{A} ⇒ {B, C}

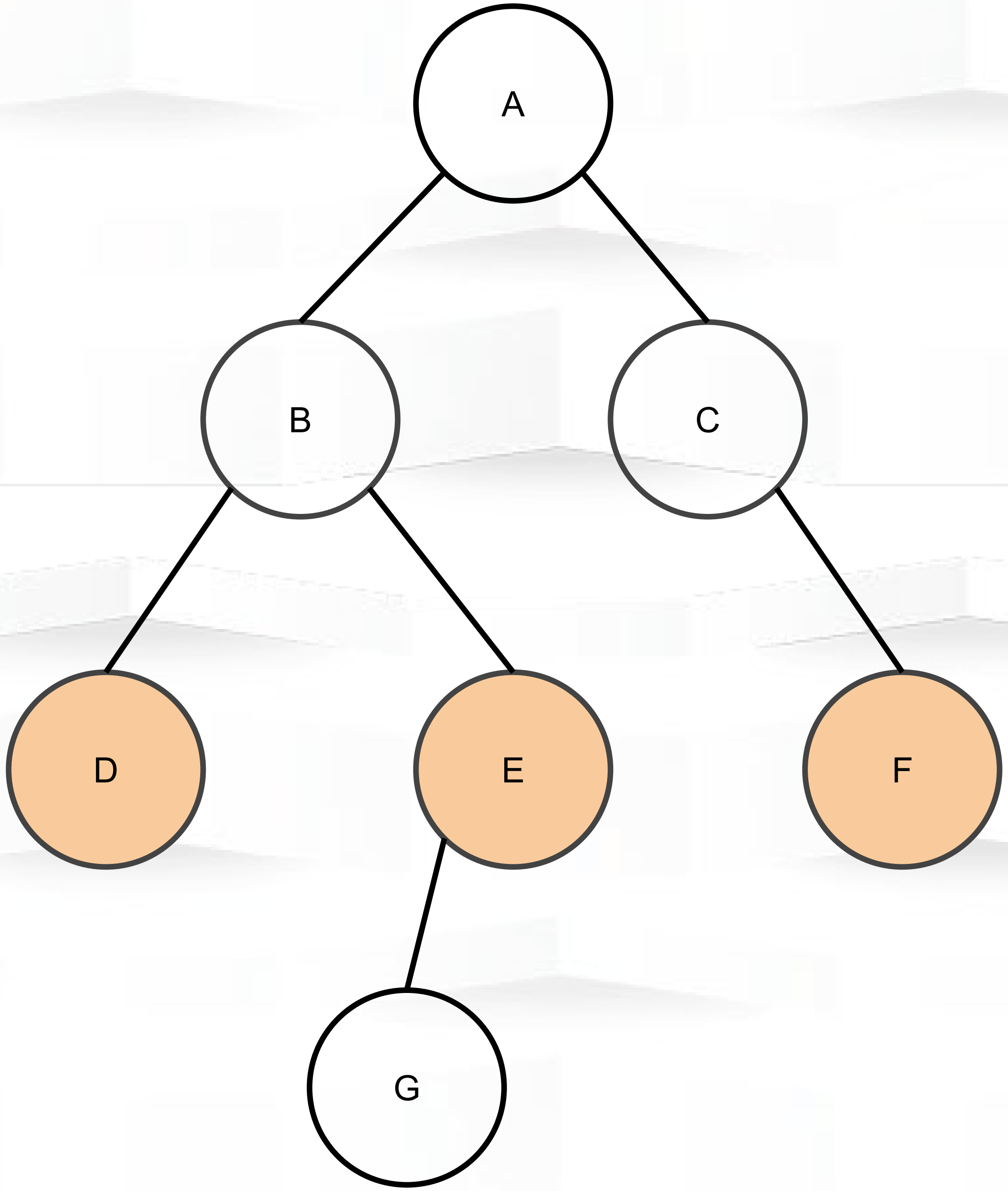


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

{A} ⇒ {B, C} ⇒ {D, E, F}

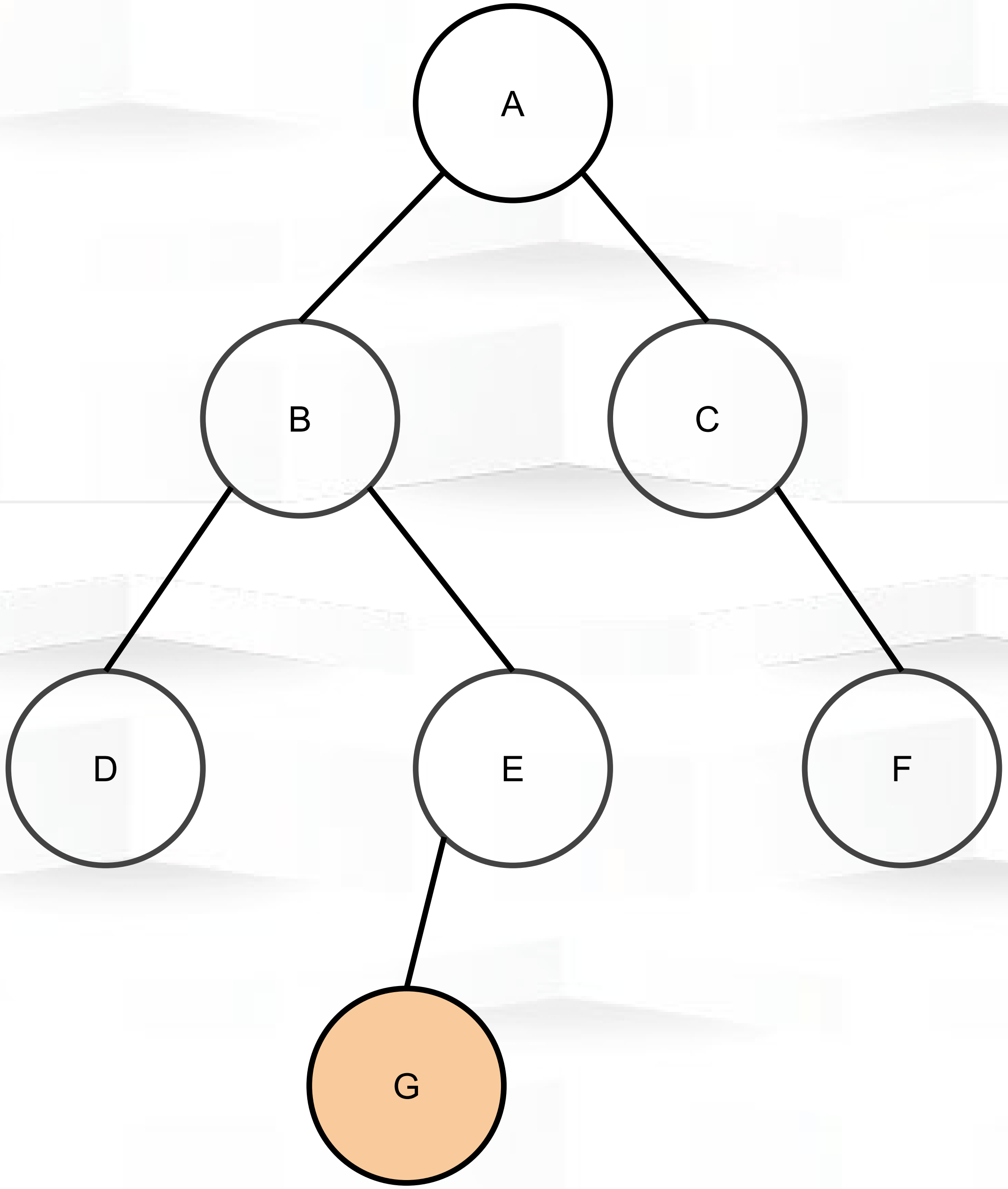


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

$\{A\} \Rightarrow \{B, C\} \Rightarrow \{D, E, F\} \Rightarrow \{G\}$

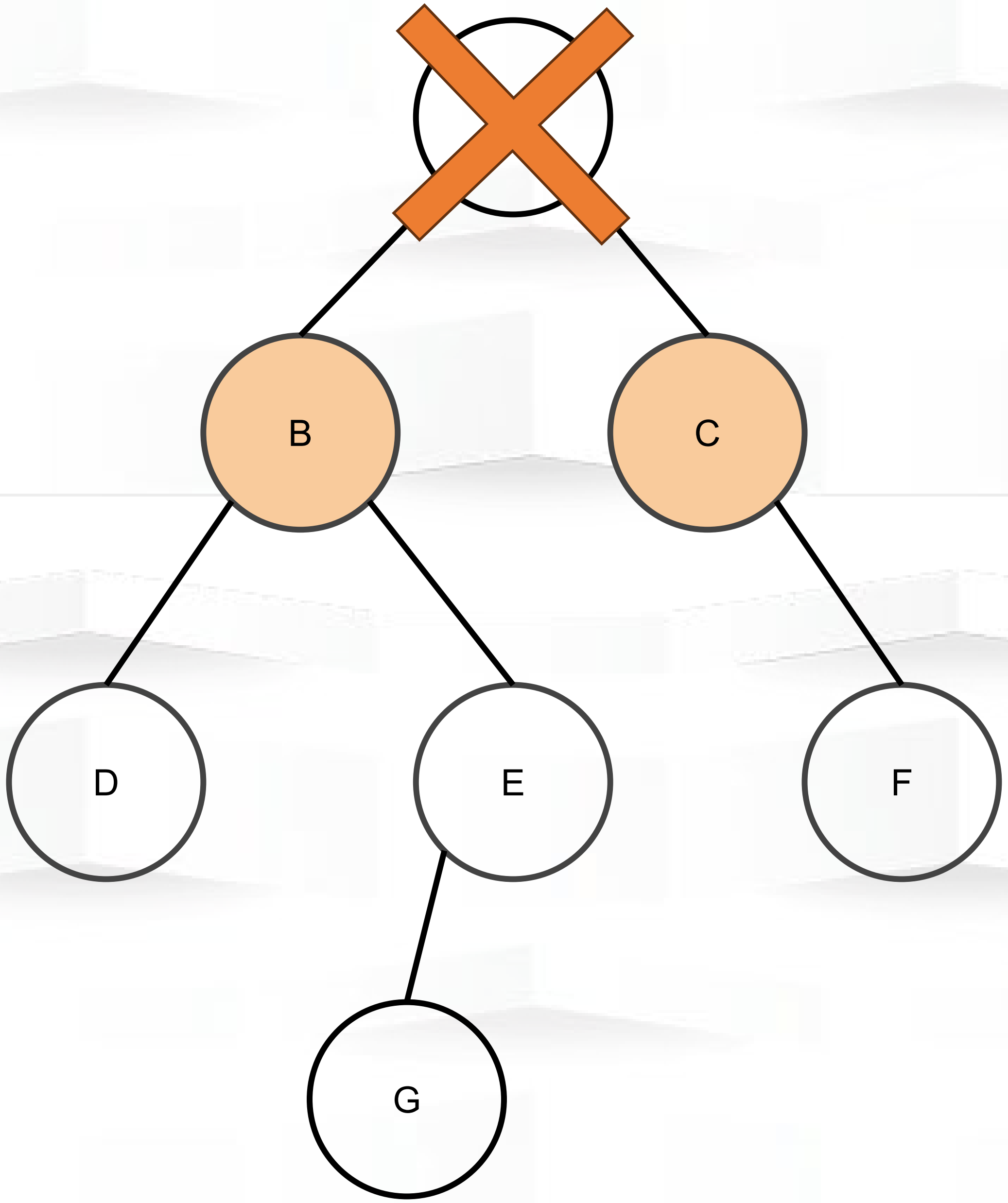


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

- **En el paso actual no volvemos a A, porque ya ha sido visitado**
- → Necesitamos apuntar qué nodos han sido visitados para no repetir: **set de visitados**

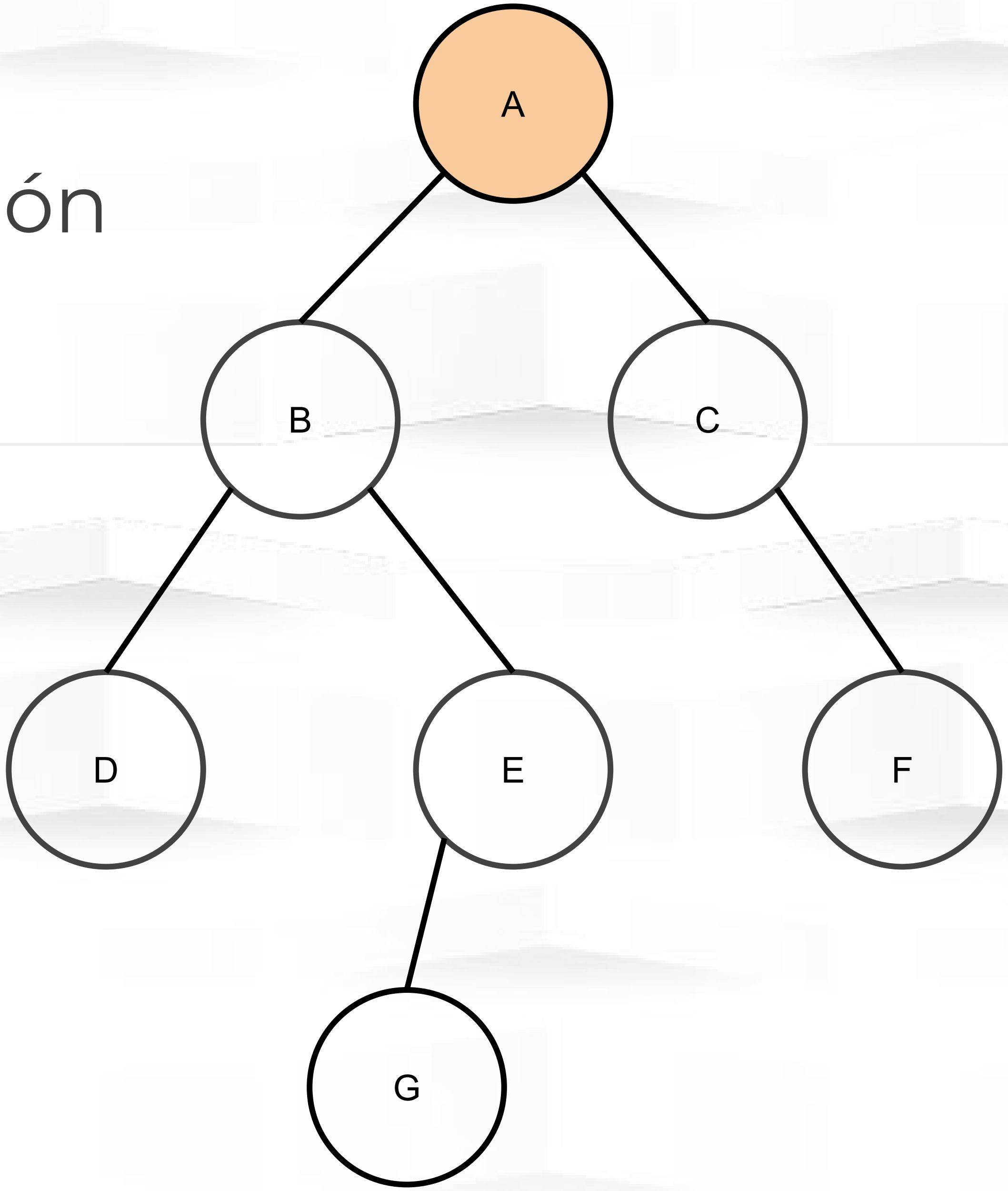


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura
- **Implementación:** inicialización

```
visitado = {}
pendientes = [A]
```

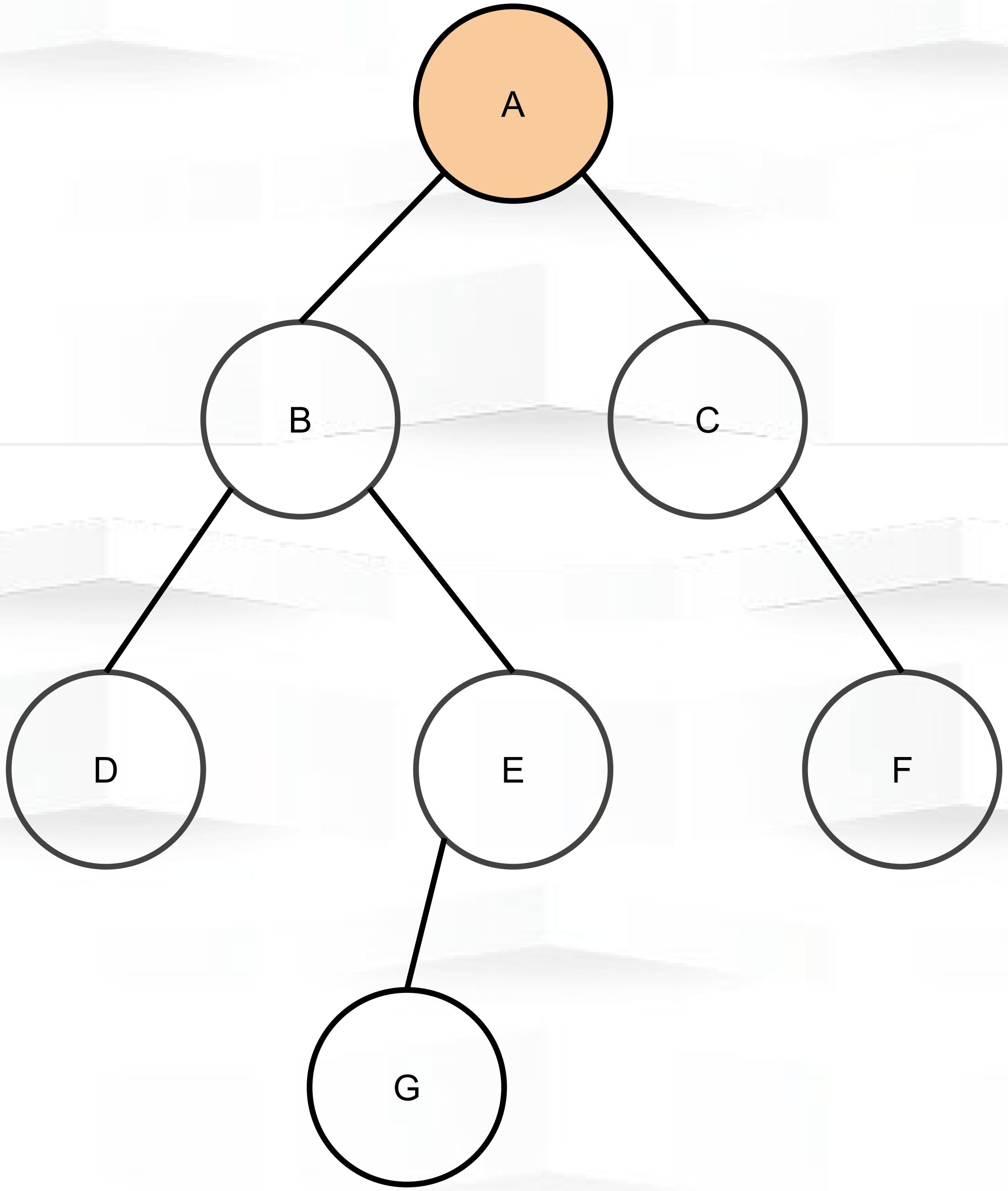


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura
- **Implementación:** recorrido

```
visitado = {}
cola = [A]
while cola:
   v = cola.primero()
   if visitado[v2]:
       ignorar
   visitado[v2] = True
   for v2 vecino v:
       cola.add(v2)
```



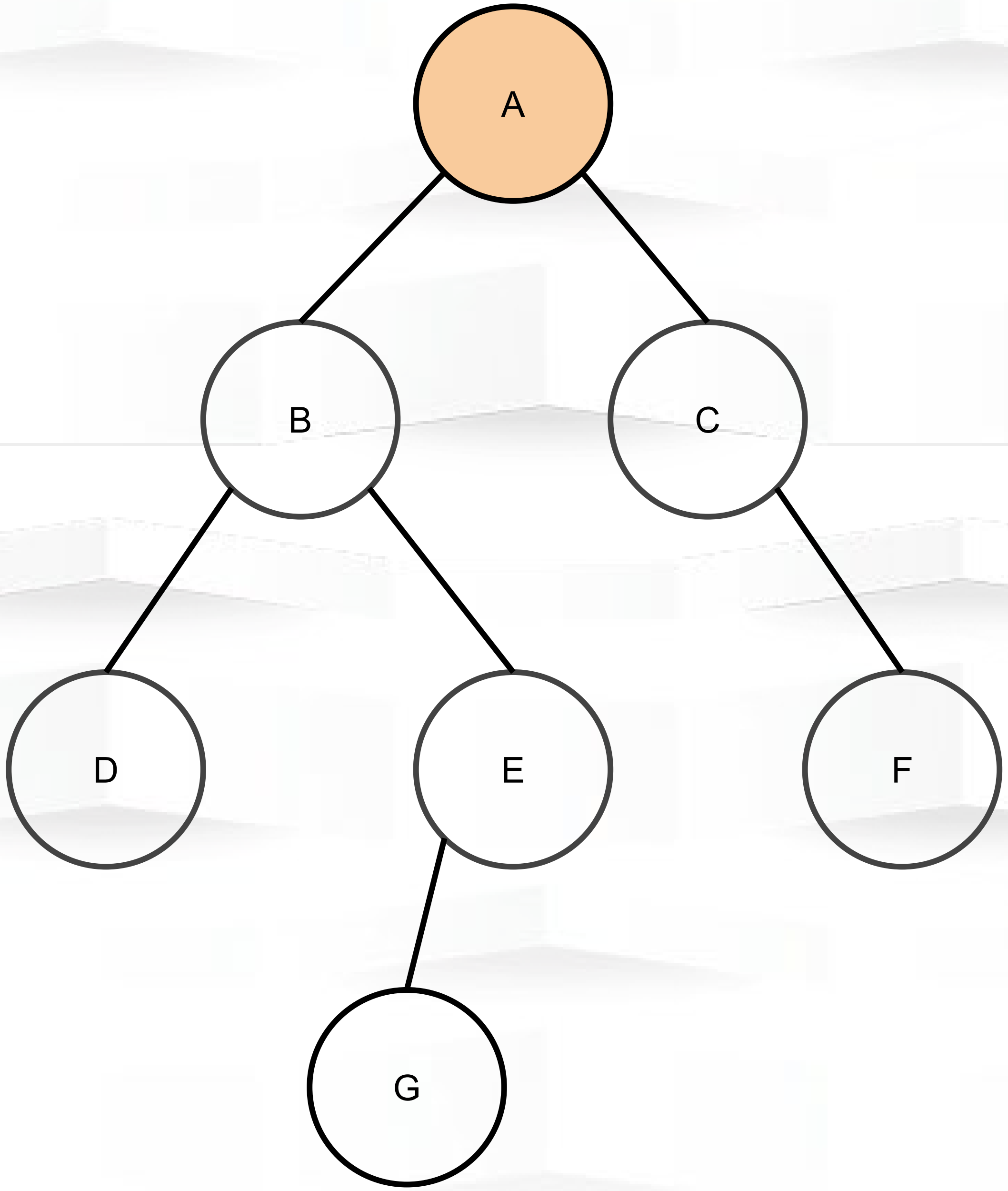
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura

Resultado orden de recorrido:

{A} ⇒ {B, C} ⇒ {D, E, F} ⇒ {G}

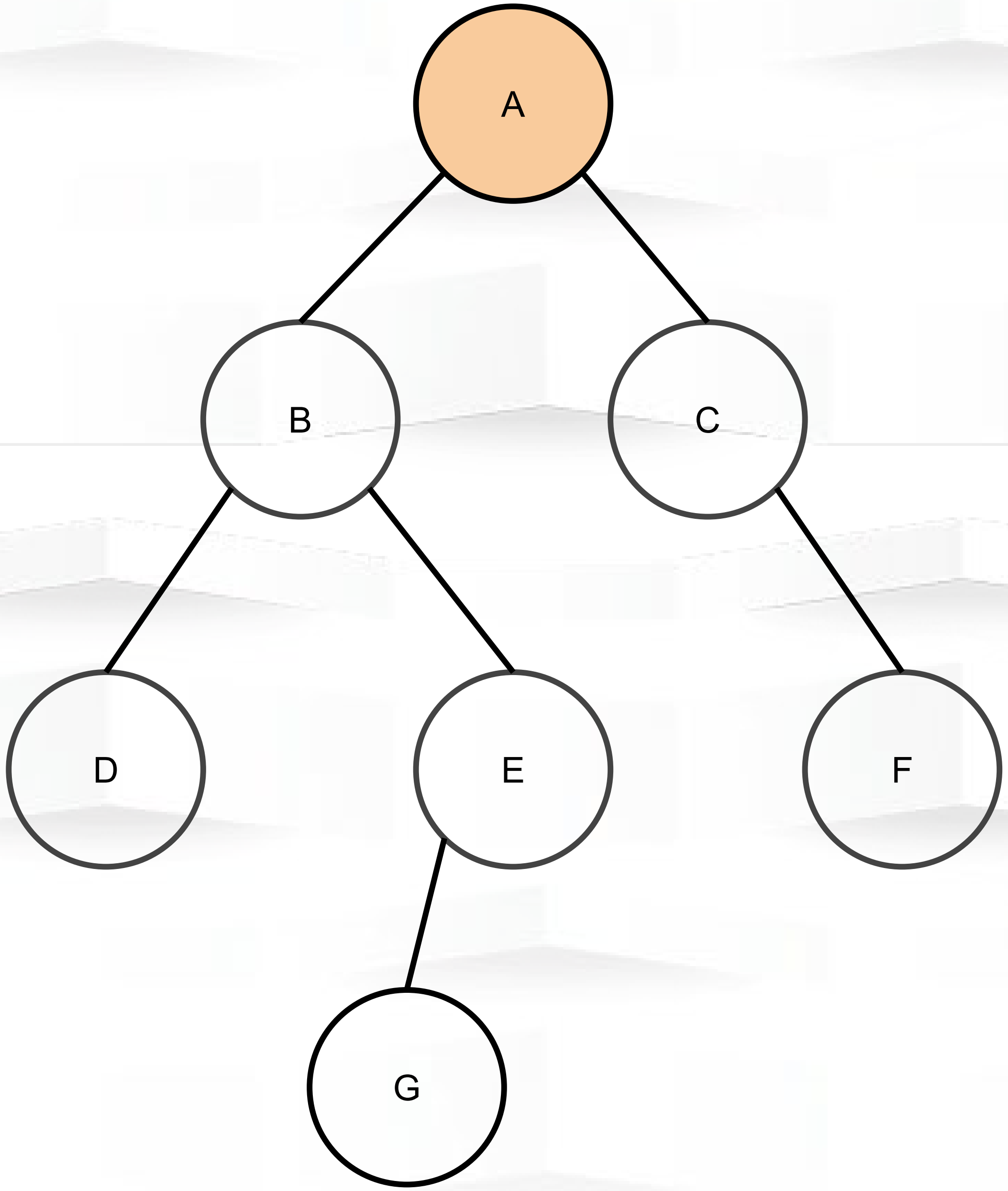


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **BFS:** Búsqueda en anchura
- **Implementación:** recorrido

```
visitado = {}
cola = [A]
while cola:
   v = cola.primero()
   if visitado[v2]:
        ignorar
   visitado[v2] = True
   for v2 vecino v:
        cola.add(v2)
```



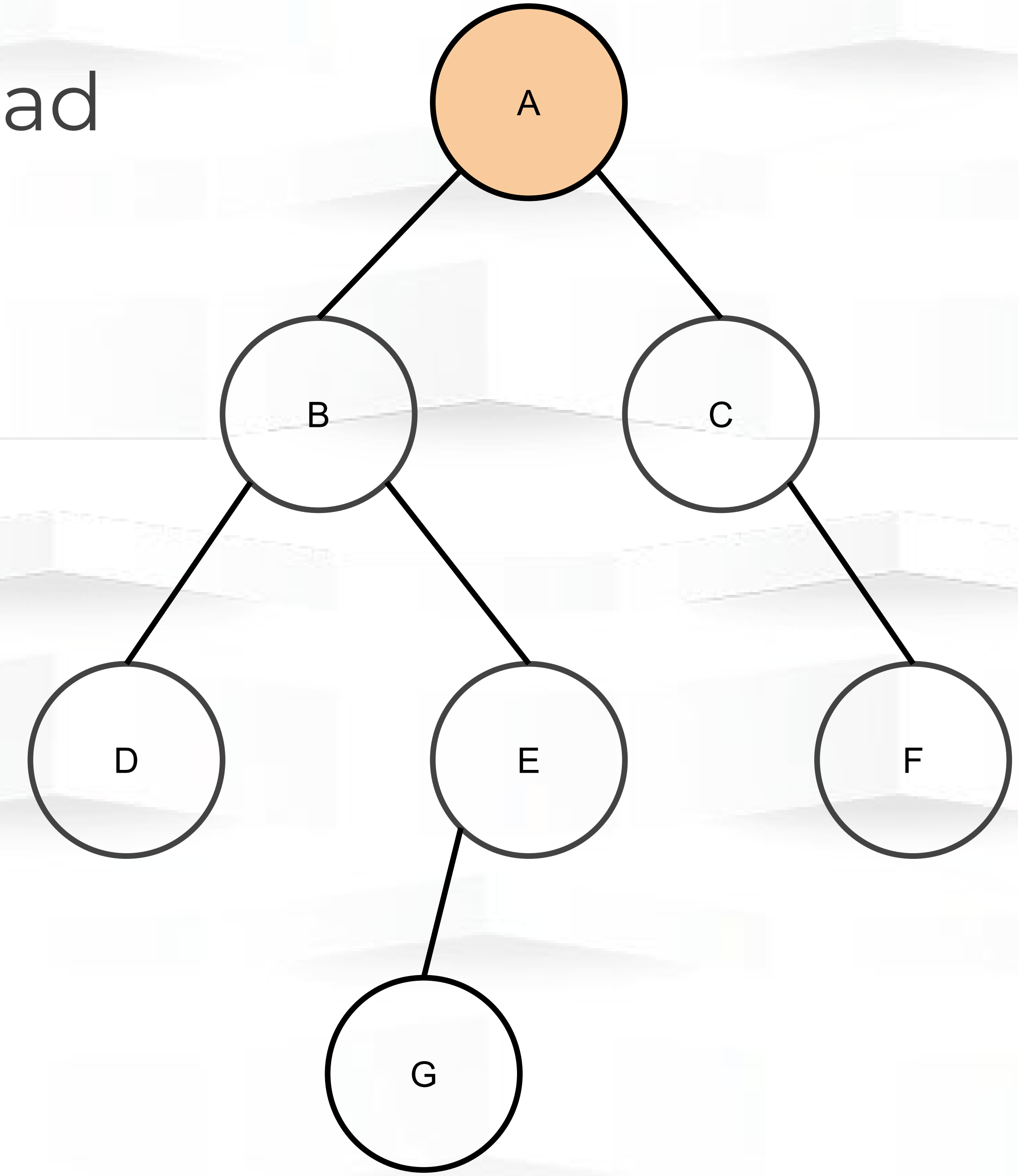
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **DFS:** Búsqueda en profundidad

En vez de recorrer por niveles, recorremos la rama entera.

Ej: A ⇒ C ⇒ F ⇒ B ⇒ E ⇒ G ⇒ D

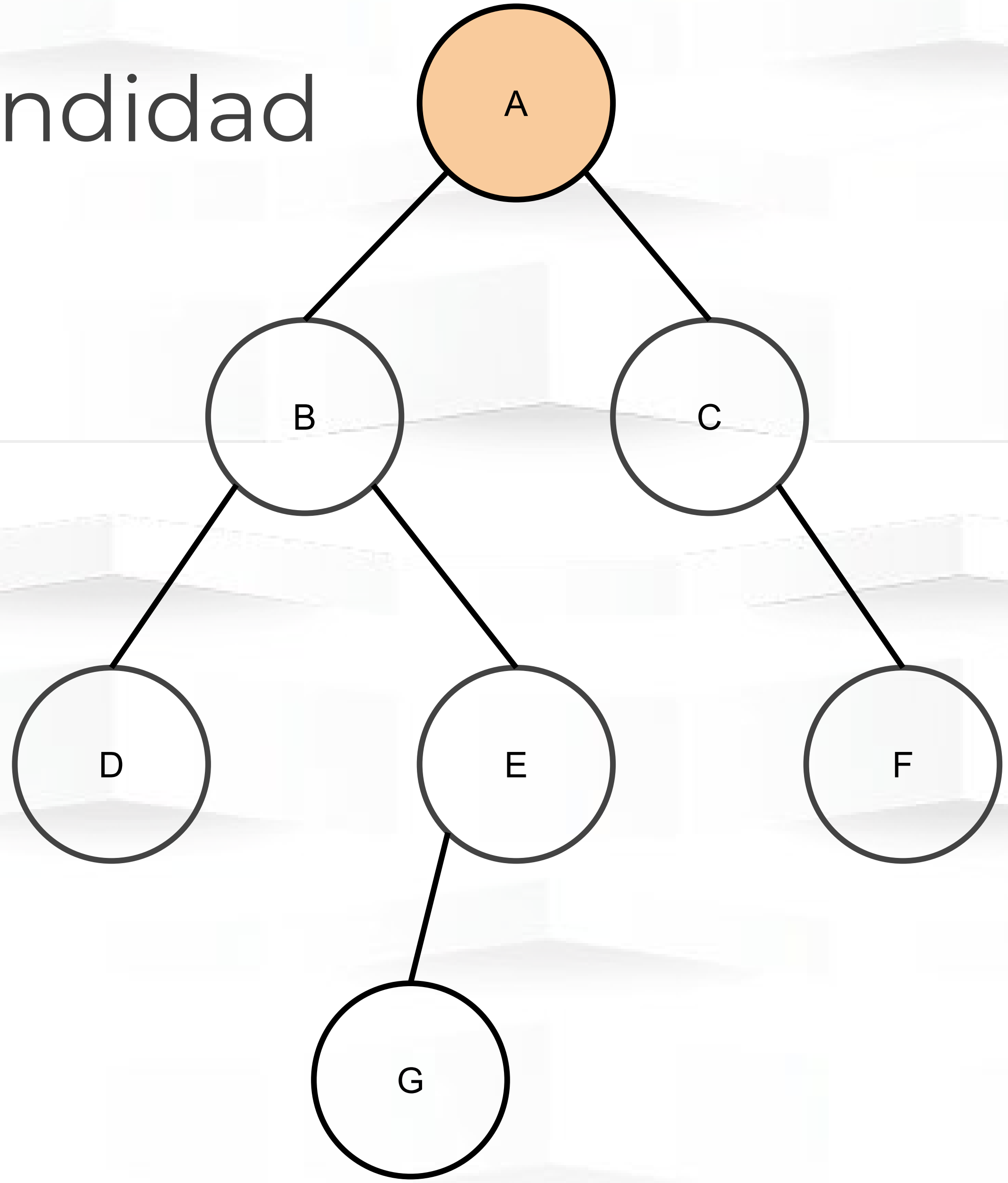


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **DFS:** Búsqueda en profundidad

```
visitado = {}
stack = [A]
while stack :
  v = stack.primero()
  if visitado[v2]:
    ignorar
  visitado[v2] = True
  for v2 vecino v:
    stack.add(v2)
```

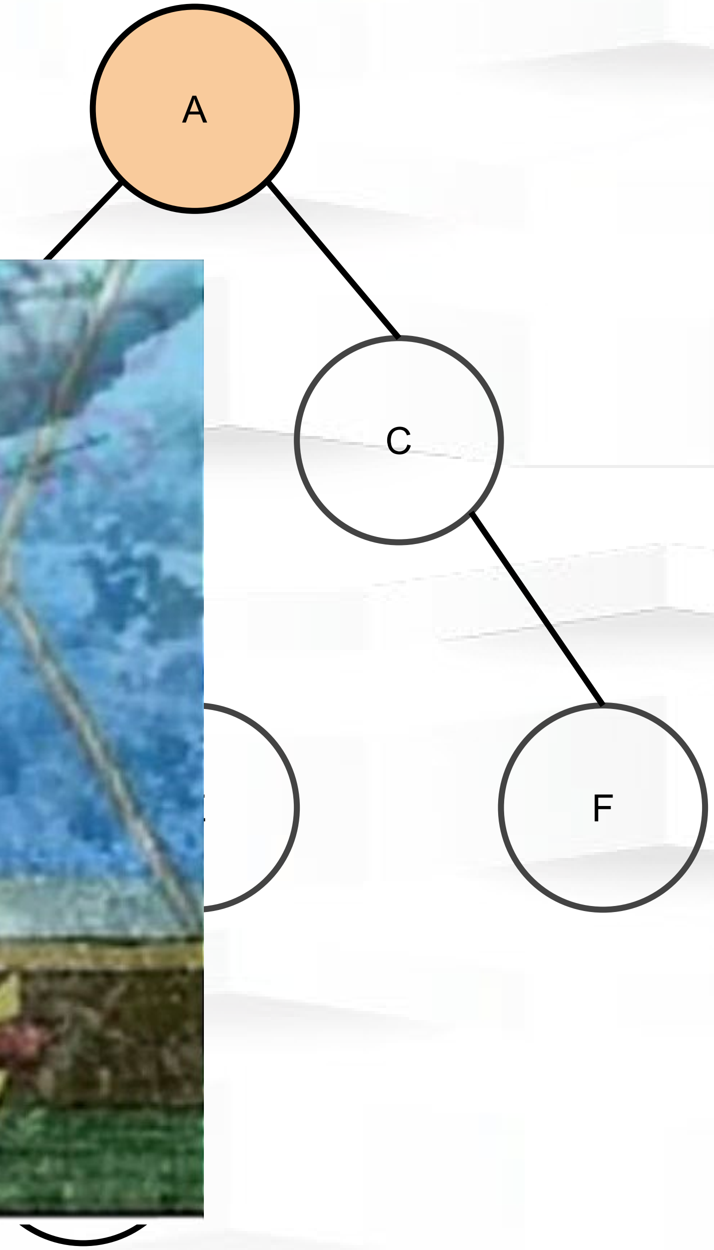


Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **DFS:** Búsqueda en profundidad

```
visitado = {}
stack = [A]
while stack
  v = stack.
  if visitad
    ignorar
  visitado[v
  for v2 vec
    stack.add(v2)
```



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Búsquedas

- **DFS:**

```
visitado = {}
stack = [A]
while stack :
   v = stack.primero()
   if visitado[v2]:
      ignorar
   visitado[v2] = True
   for v2 vecino v:
      stack.add(v2)
```

- **BFS:**

```
visitado = {}
cola = [A]
while cola:
   v = cola.primero()
   if visitado[v2]:
      ignorar
   visitado[v2] = True
   for v2 vecino v:
      cola.add(v2)
```

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas

Búsquedas: aplicaciones prácticas

- **Camino más corto** entre dos nodos:
  - Solo funciona si las aristas no están ponderadas.

- Contar el **número de componentes** ("grupos") de un grafo

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas

Contar el **número de componentes** ("grupos") de un grafo

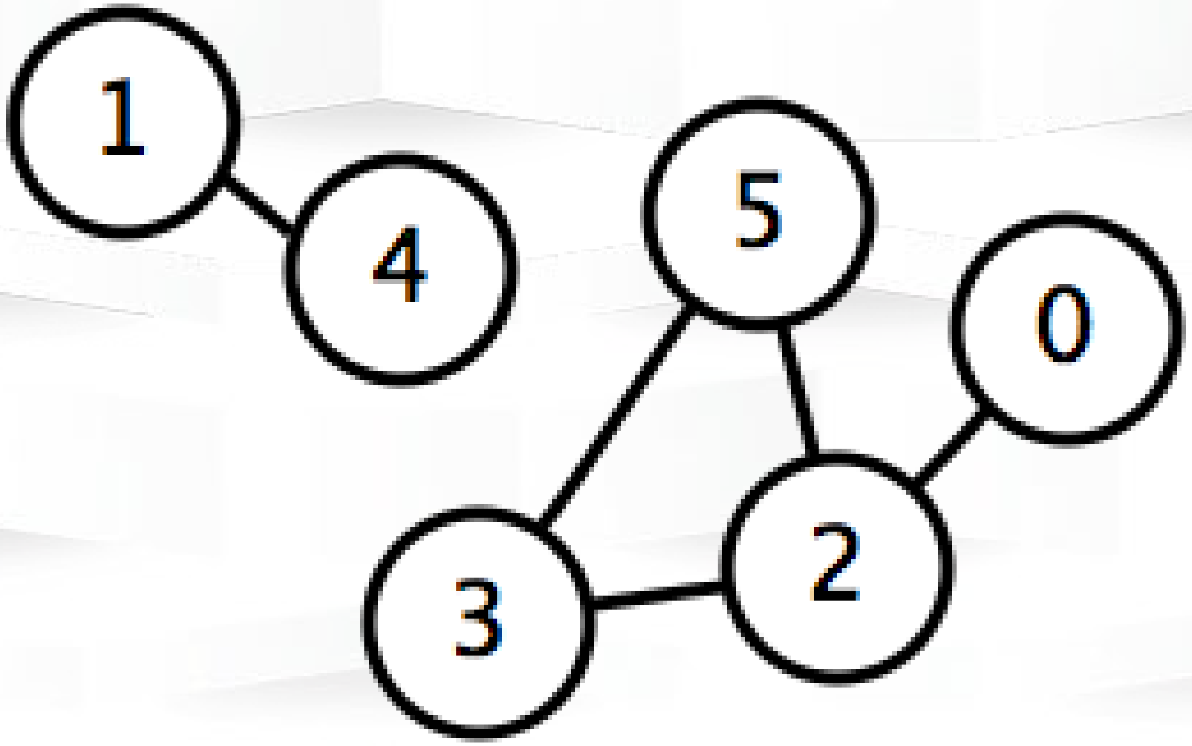
Lanzamos un BFS desde cada nodo del grafo no visitado.

El número de componentes sería el **número de veces que lanzamos un BFS**.

¿Podemos usar DFS?

Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

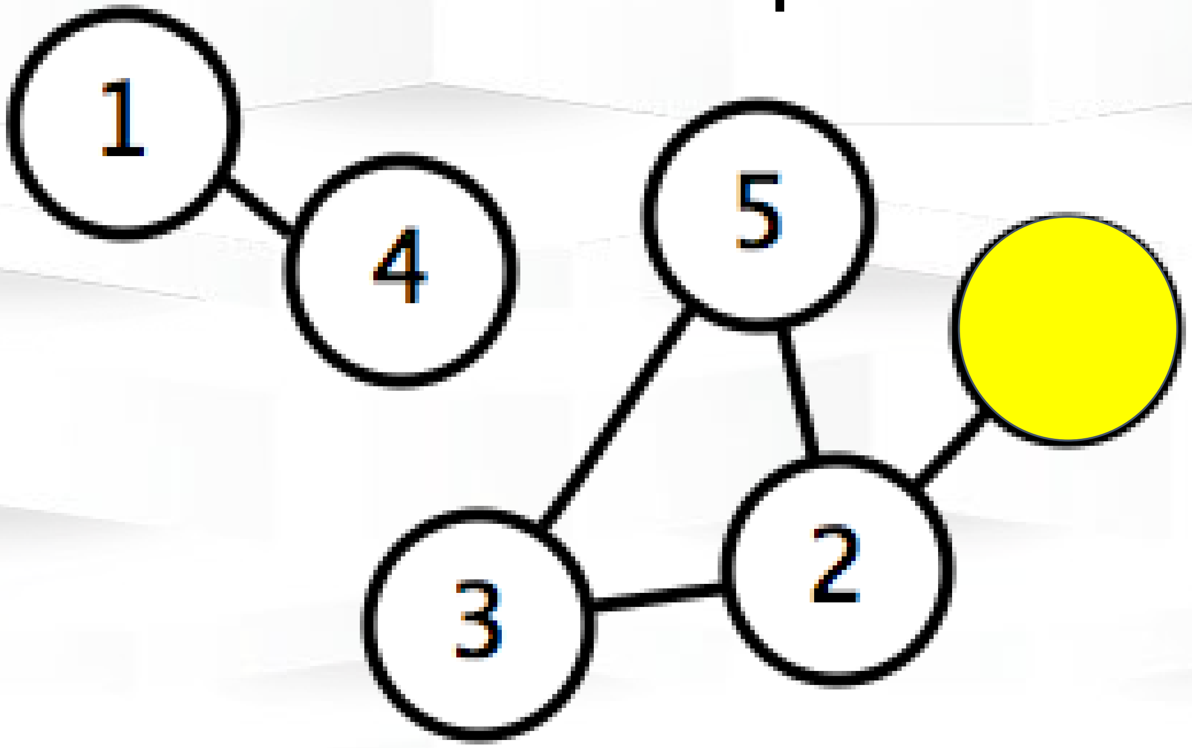
Componentes conexas



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

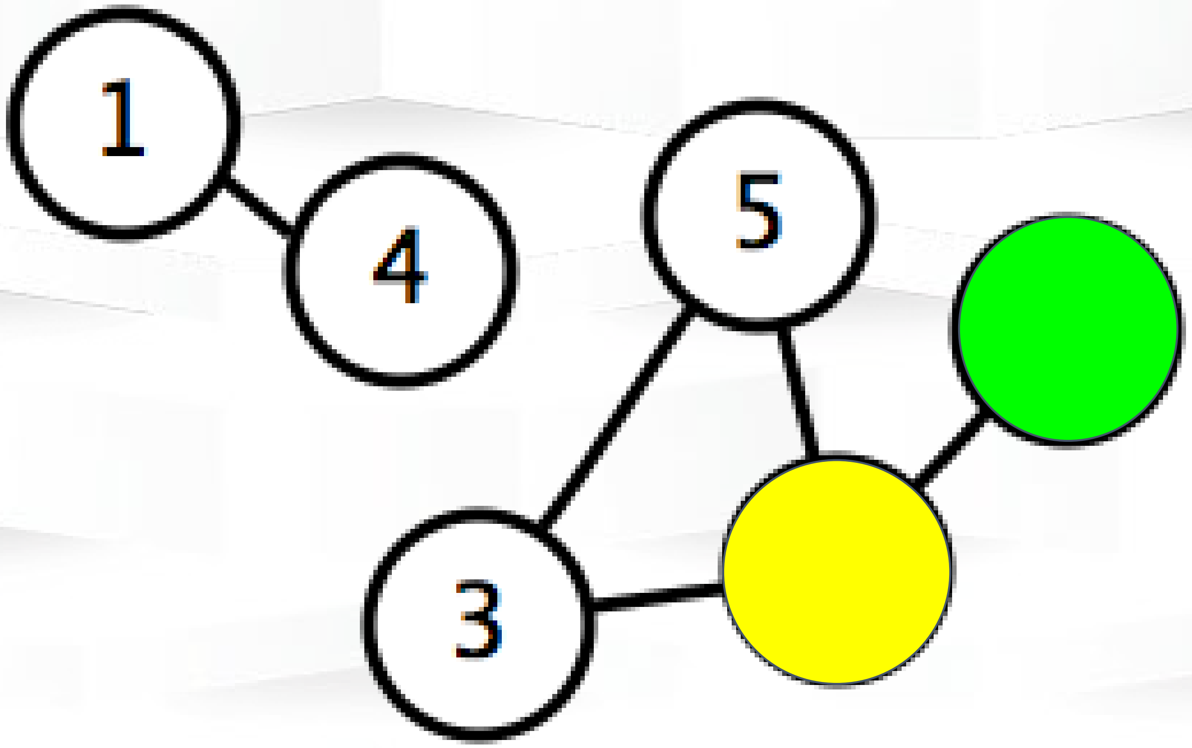
Componentes conexas

+1 Componente Conexa (1)



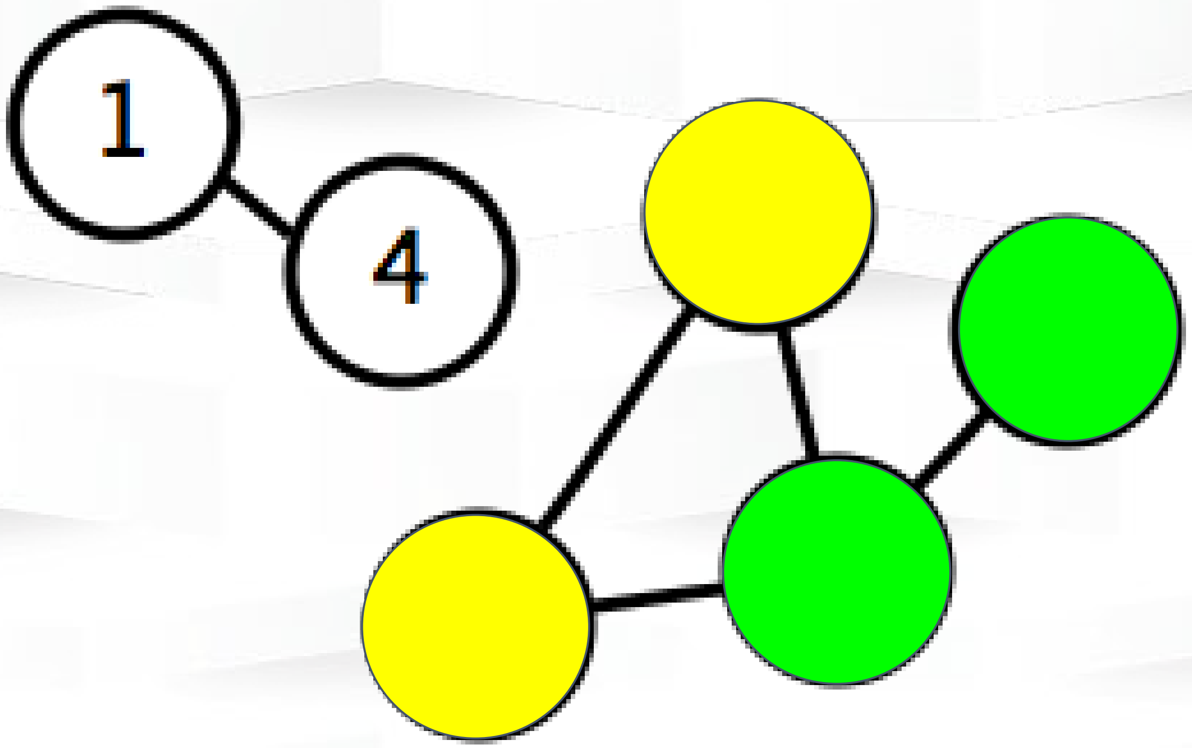
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas



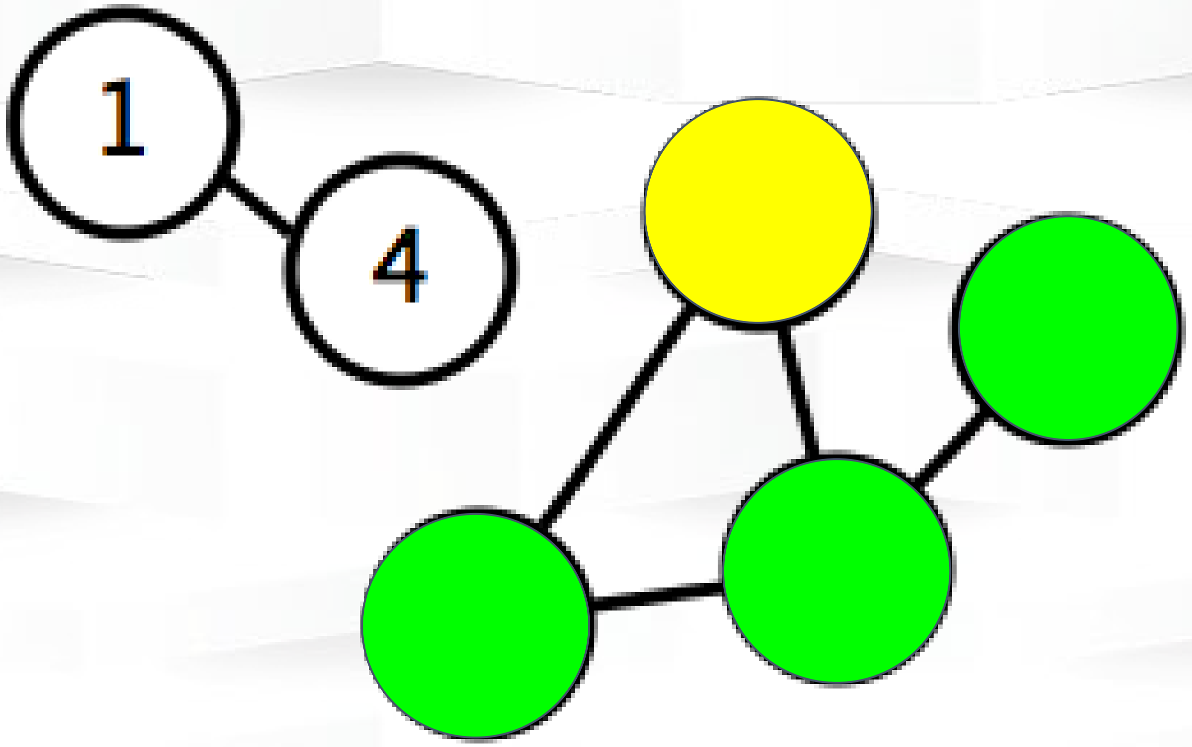
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

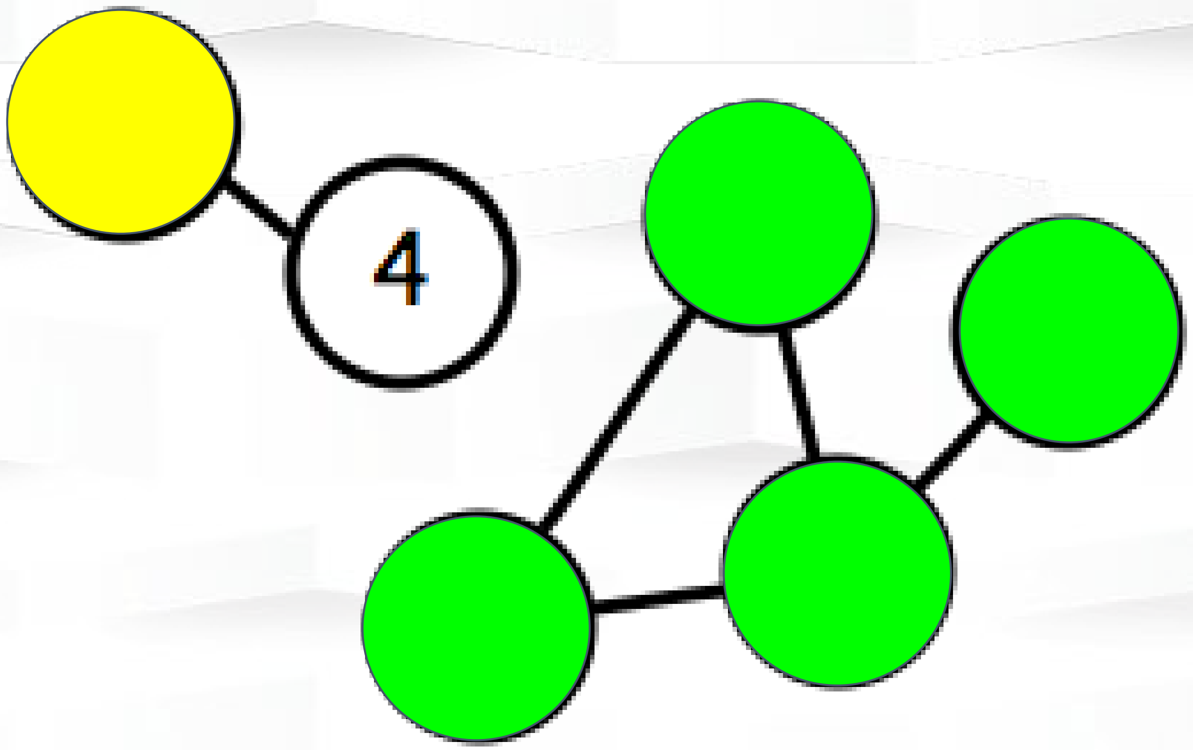
Componentes conexas



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

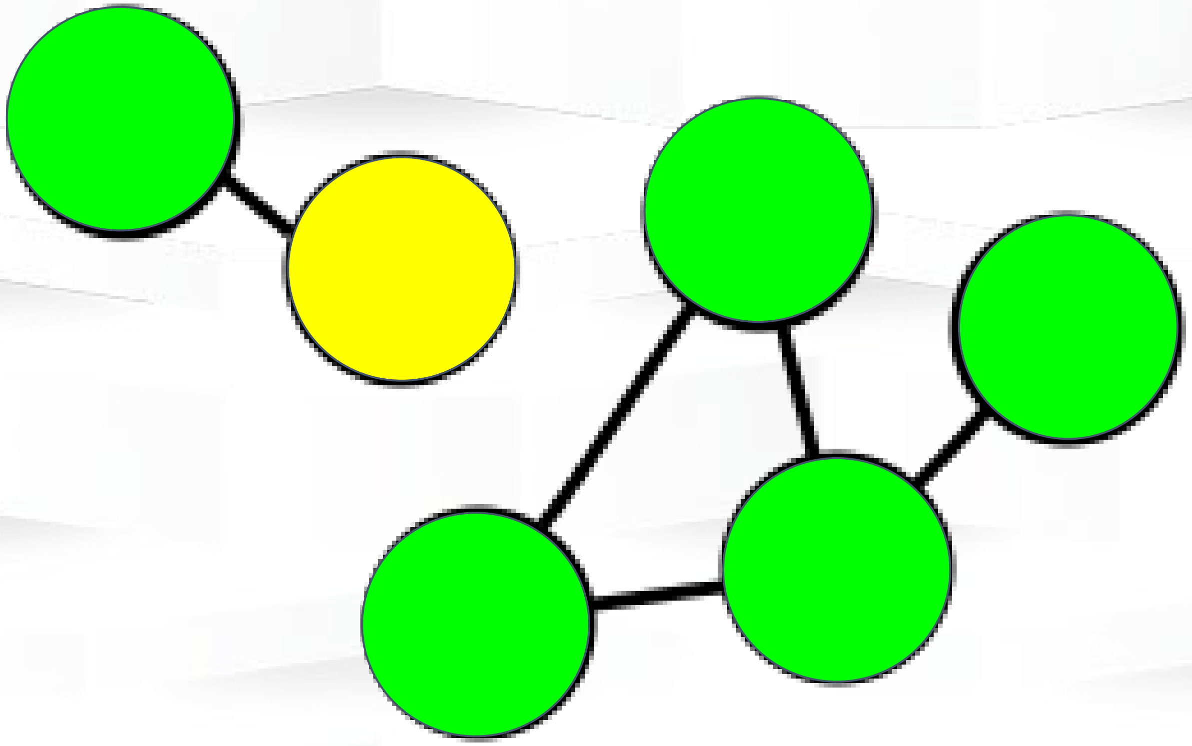
Componentes conexas

+1 Componente Conexa (2)



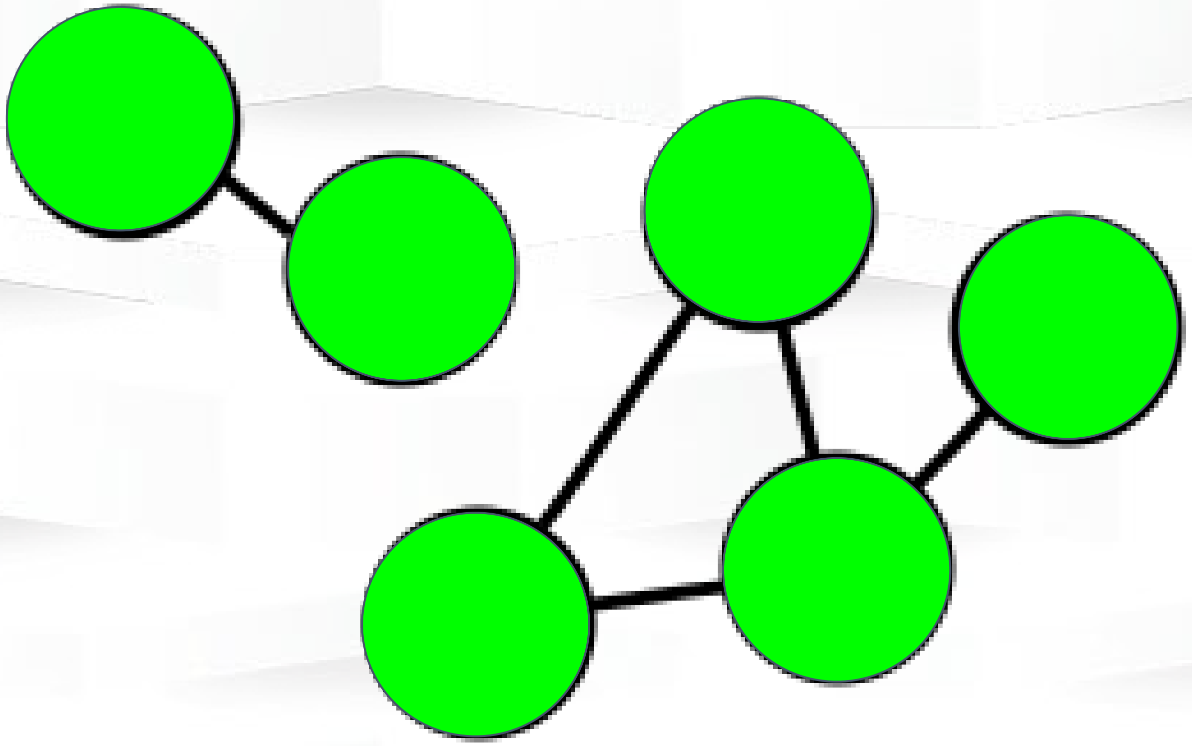
Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

Componentes conexas



Raúl Martín Santamaría
Isaac Lozano Osorio
Sergio Pérez Peló

#CátedrasCiber

Módulo III: Grafos en Ciberseguridad

Entidad colaboradora:

Financiado por la Unión Europea NextGenerationEU

GOBIERNO DE ESPAÑA
MINISTERIO PARA LA TRANSFORMACIÓN DIGITAL Y DE LA FUNCIÓN PÚBLICA
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

Plan de Recuperación, Transformación y Resiliencia

incibe_ INSTITUTO NACIONAL DE CIBERSEGURIDAD

UC Universidad de Cantabria