

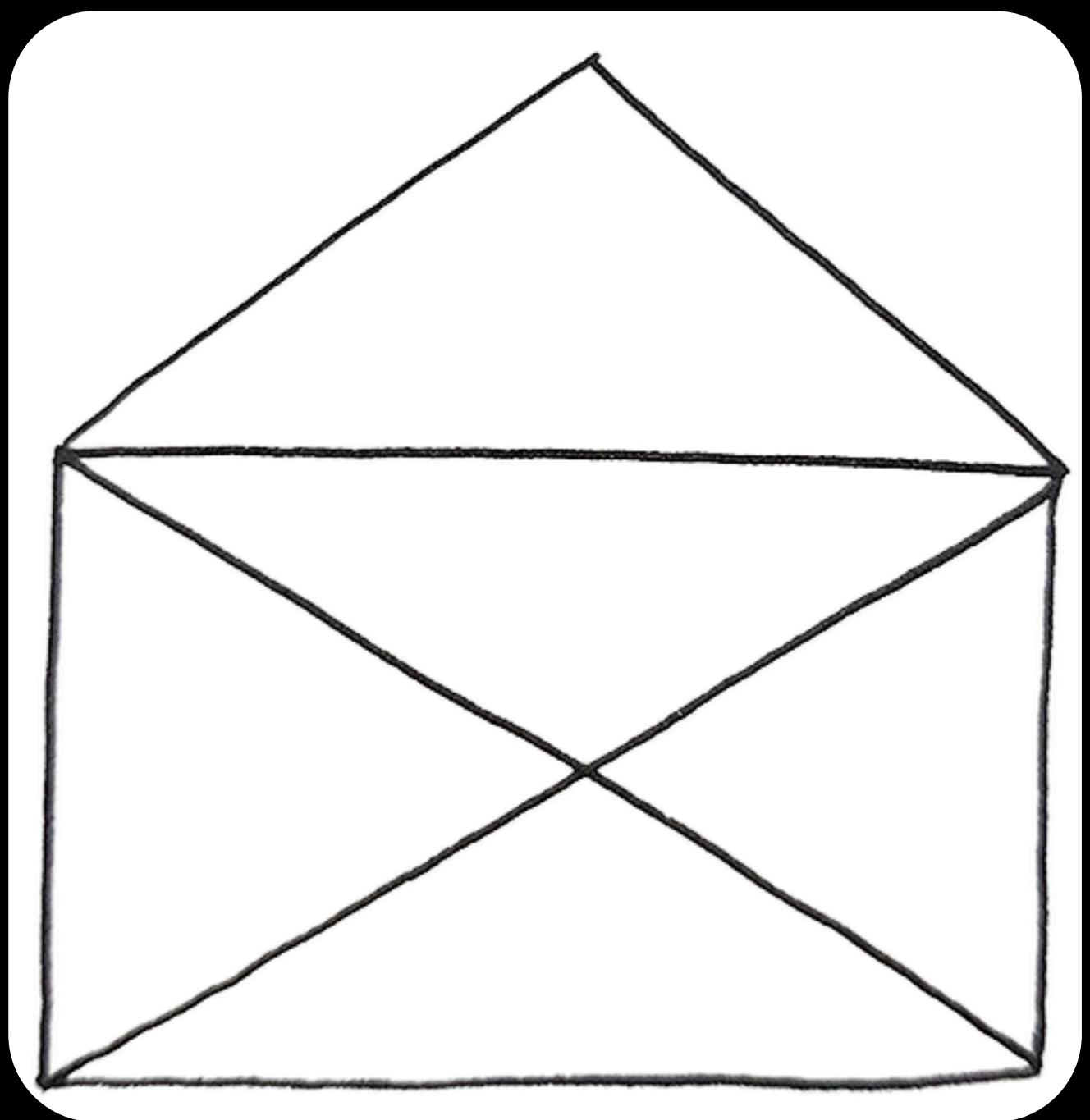
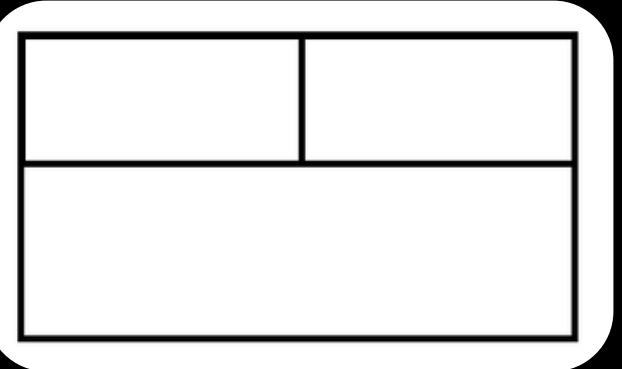
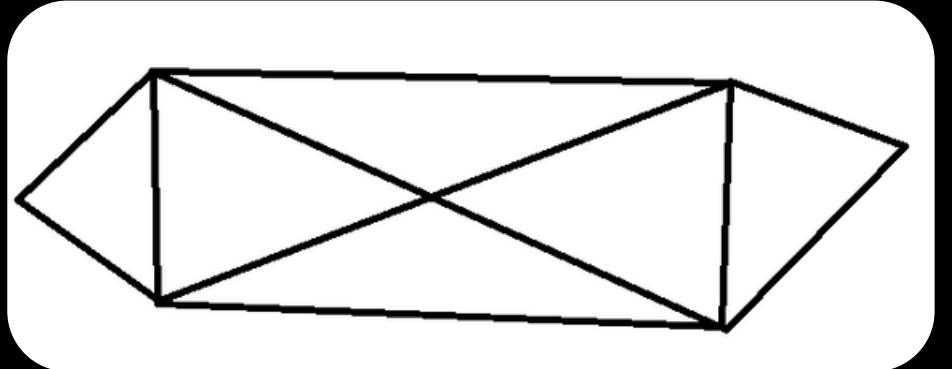
*Una introducción*  
*<Muy Algorítmica>*  
*a la Teoría de*  
**GRAFOS**

Presentado por:

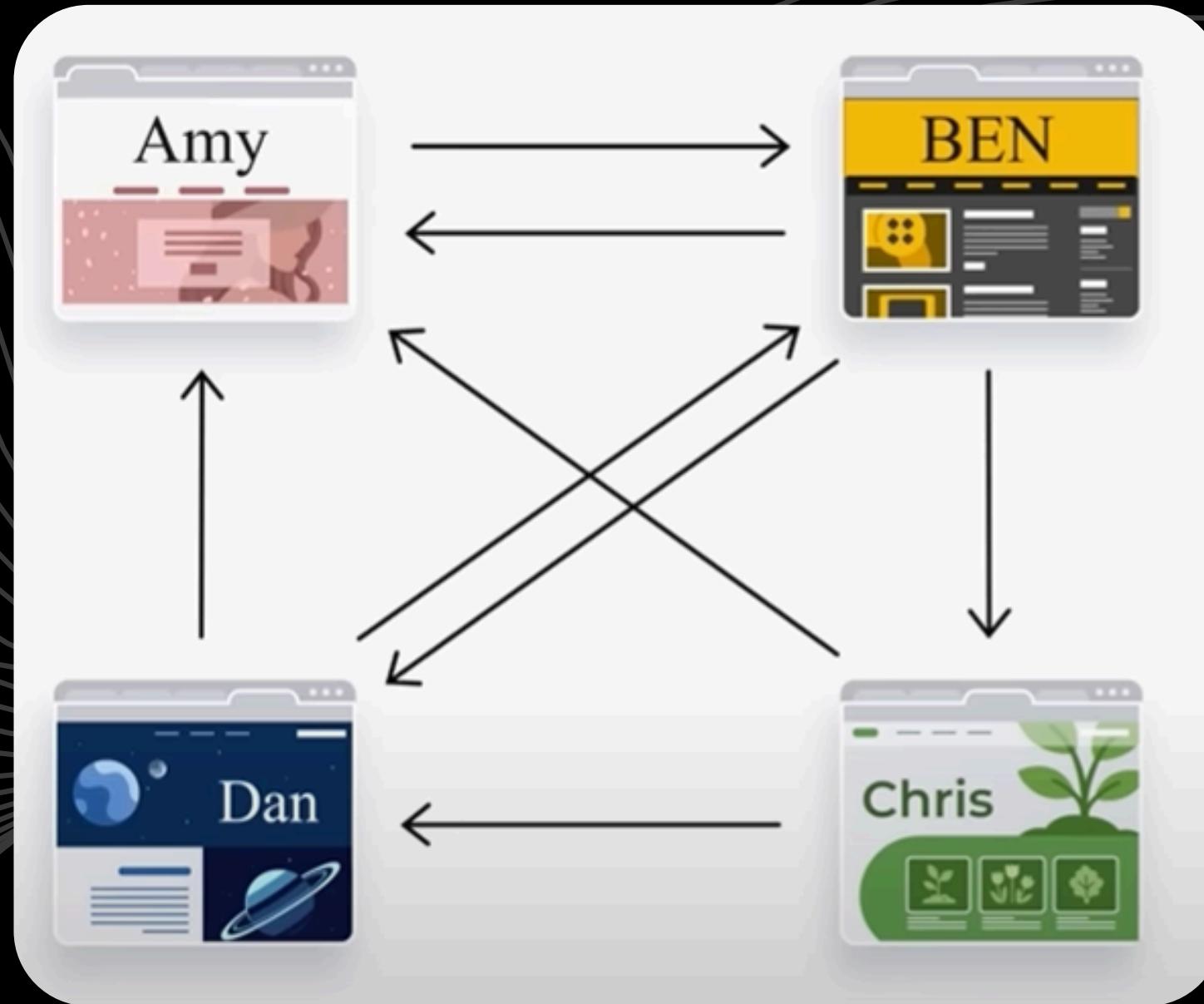
- Jerónimo Hoyos Botero.
- Daniel Fernando Verdecia Goyeneche

*¿Puedes dibujar la  
casita de un solo  
trazo?*

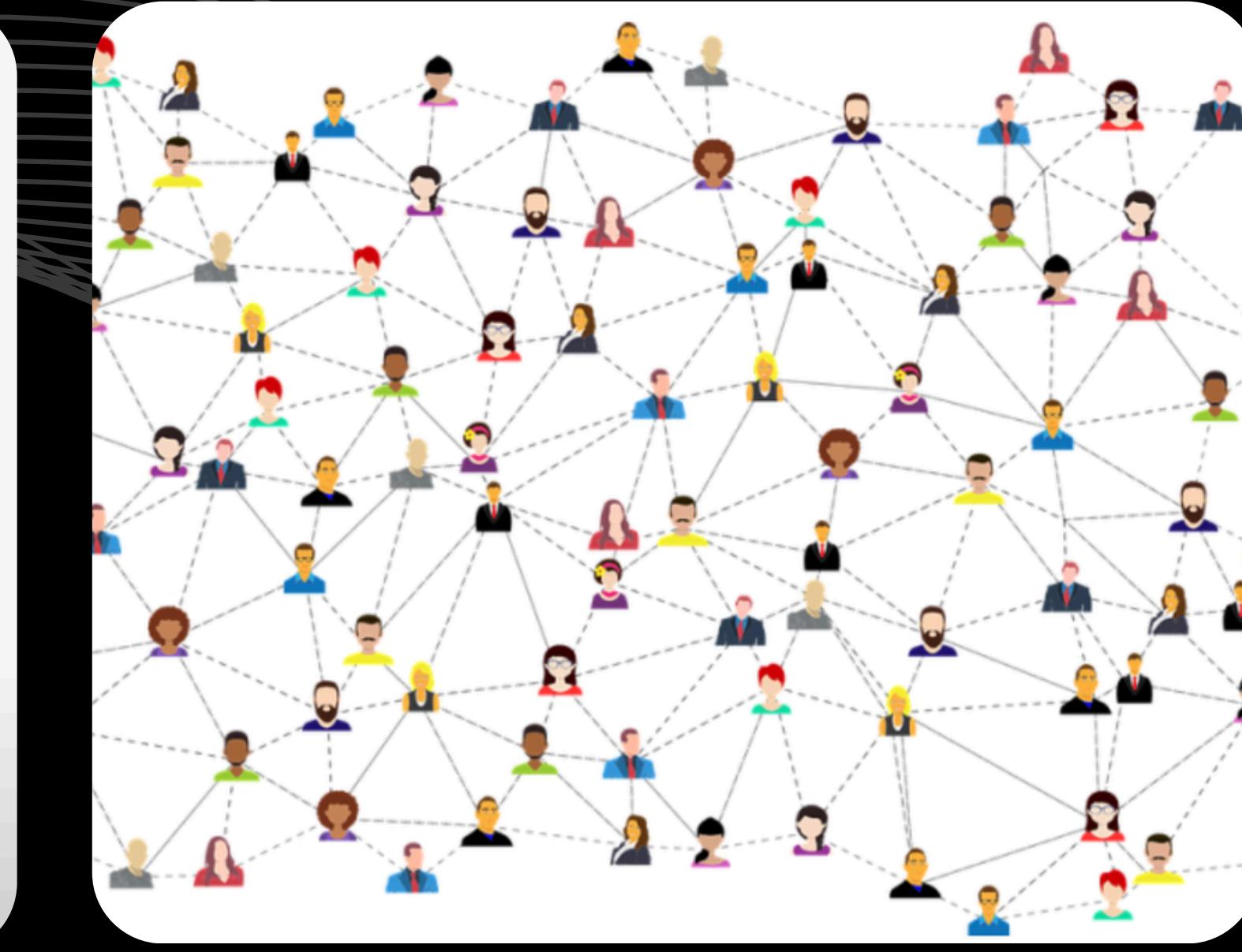
*Otros ejemplos:*



# *¡Está en todos lados!*



Páginas WEB

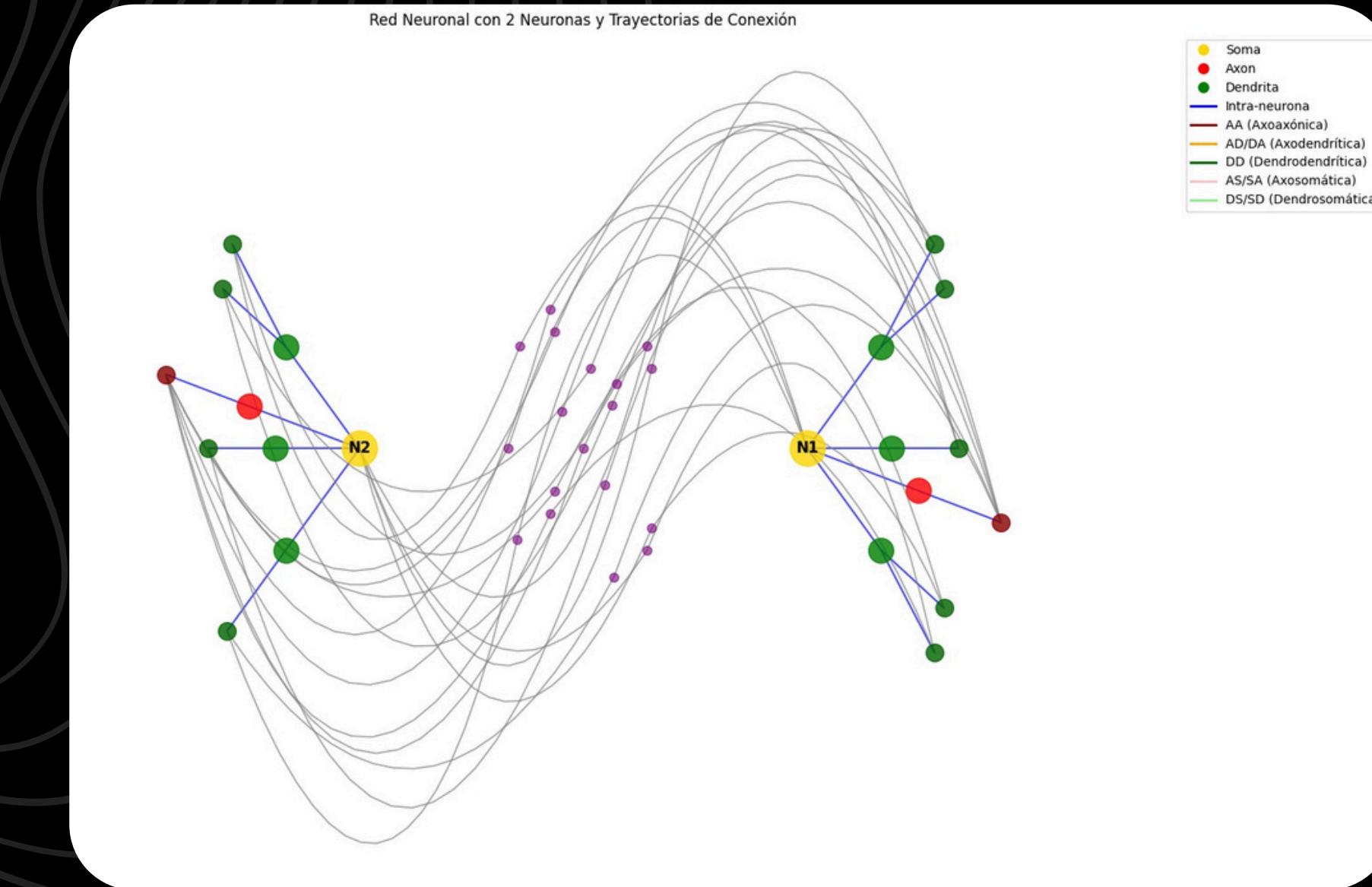
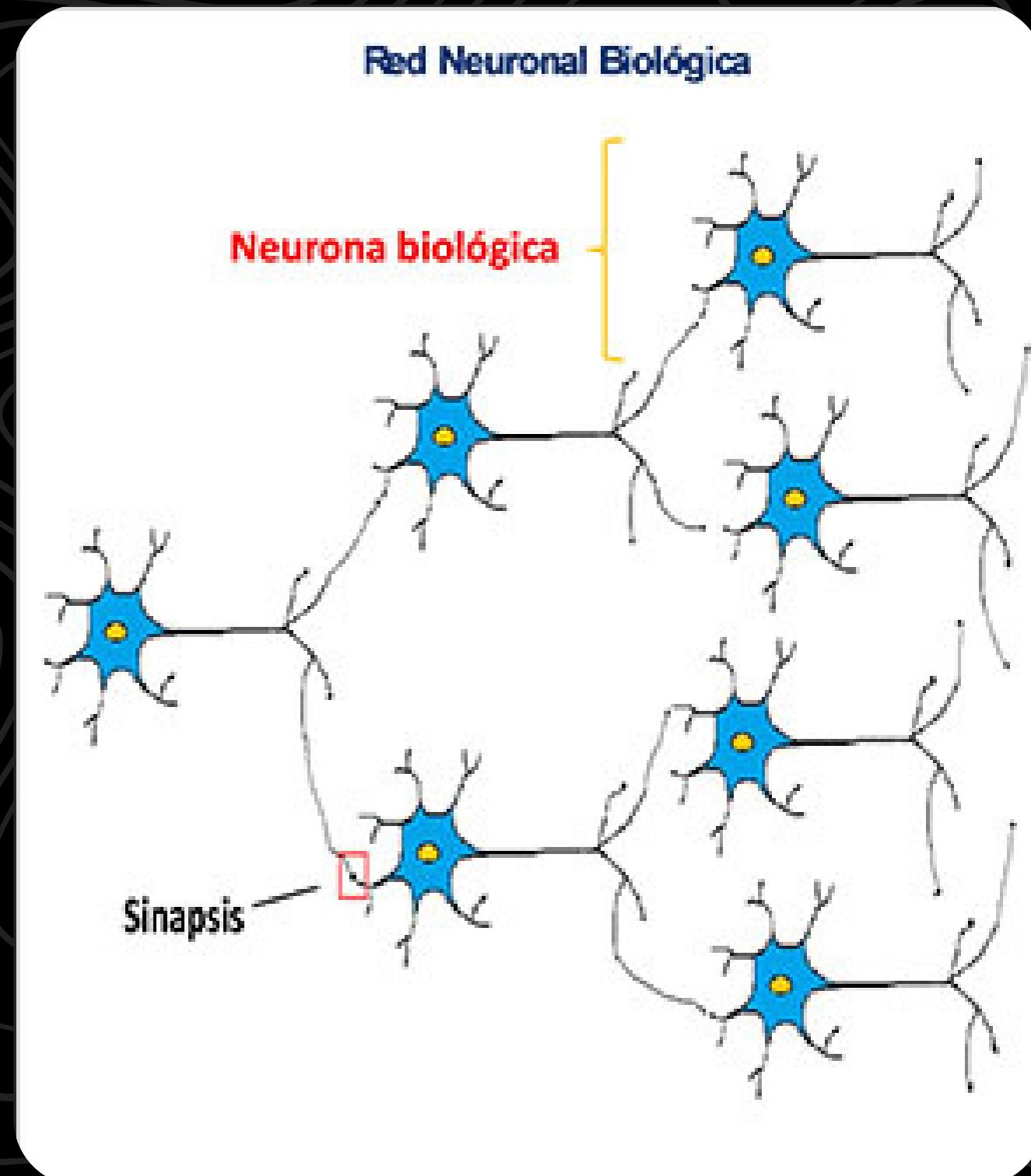


Personas



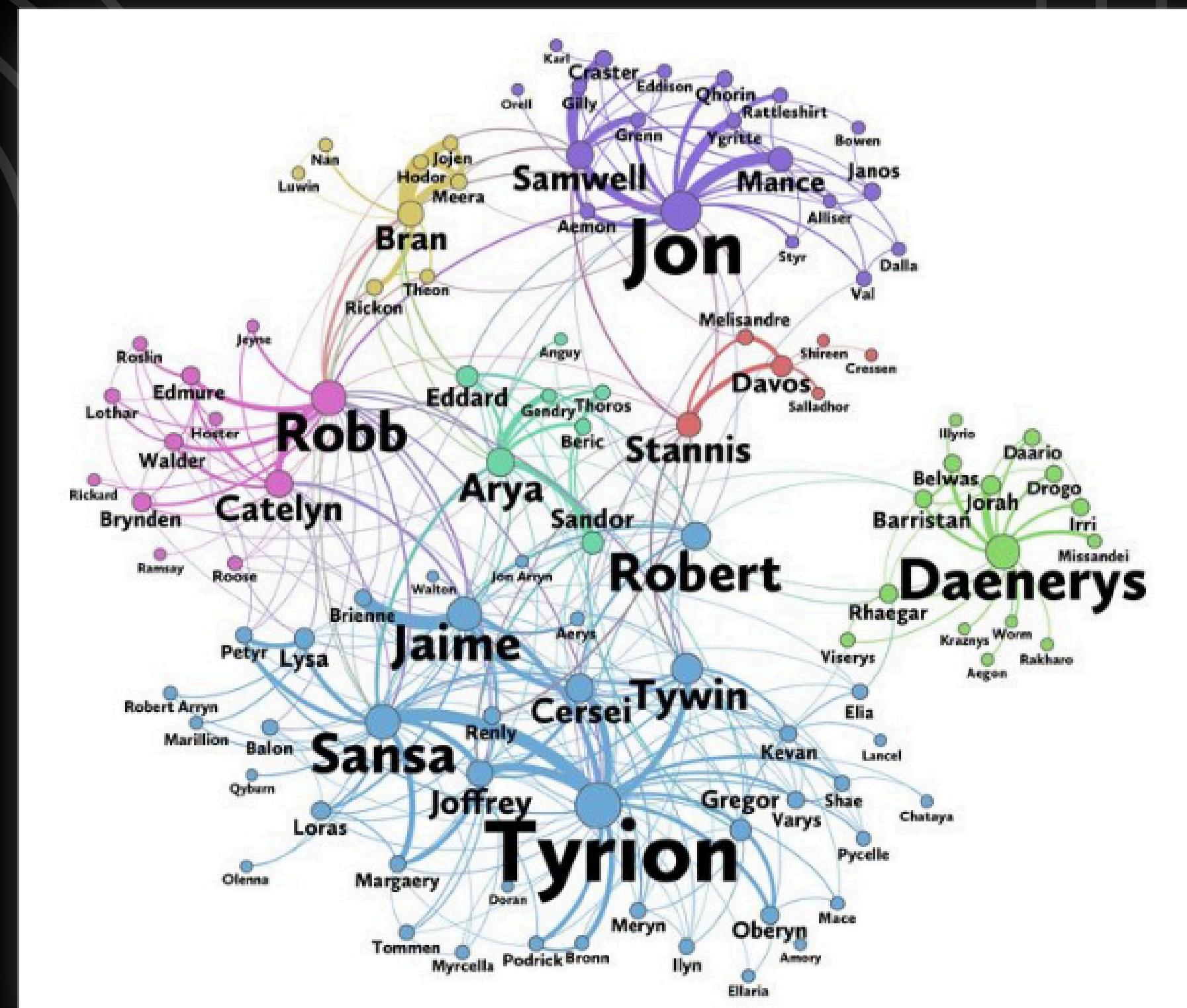
UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# Analogía Fisiológica- Grafos



¡Investigación en proceso por  
semilleristas de neuroco!

# *¿Quién es el protagonista de GOT?*



# KÖNIGSBERG



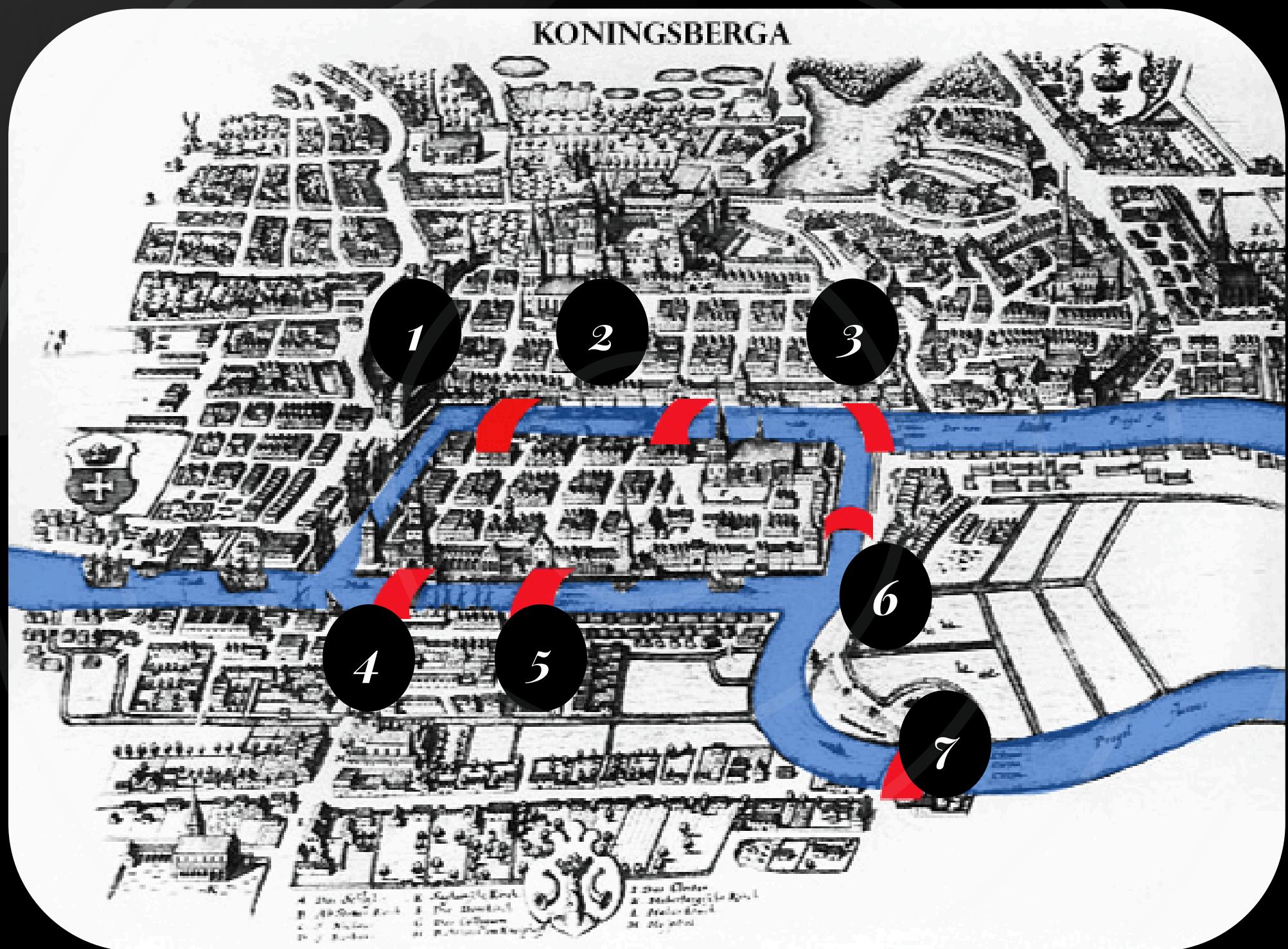
## Kant

*Estudio introductorio*  
José Luis Villaçañas

Crítica de la razón pura

Tierra de  
Immanuel Kant

*¿Se puede hacer un camino donde se pasen solo una vez cada puente?*



## Posibilidades

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

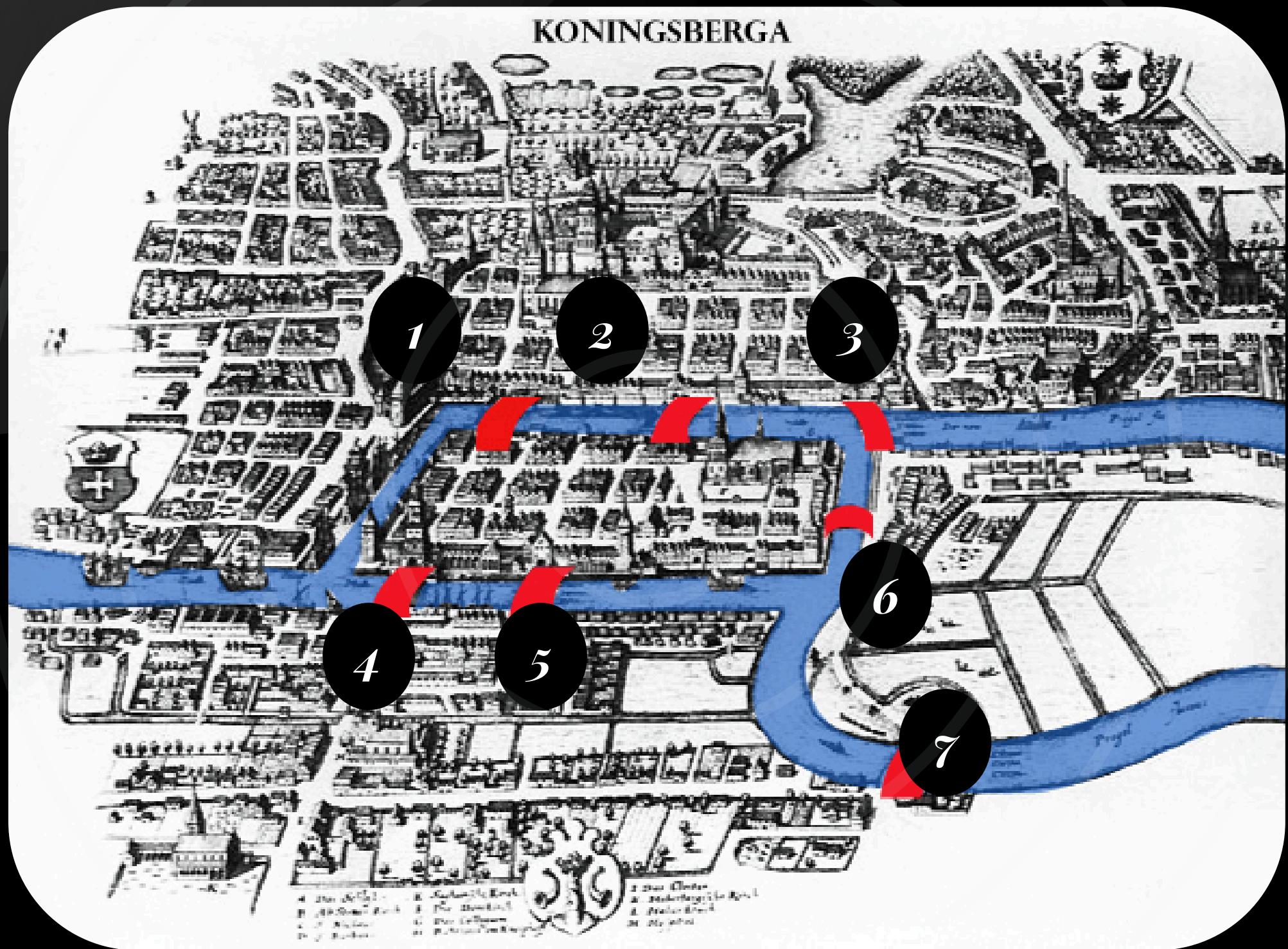
$$1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7$$

$$7! = 5040$$



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

*¿Se puede hacer un camino donde se pasen solo una vez cada puente?*

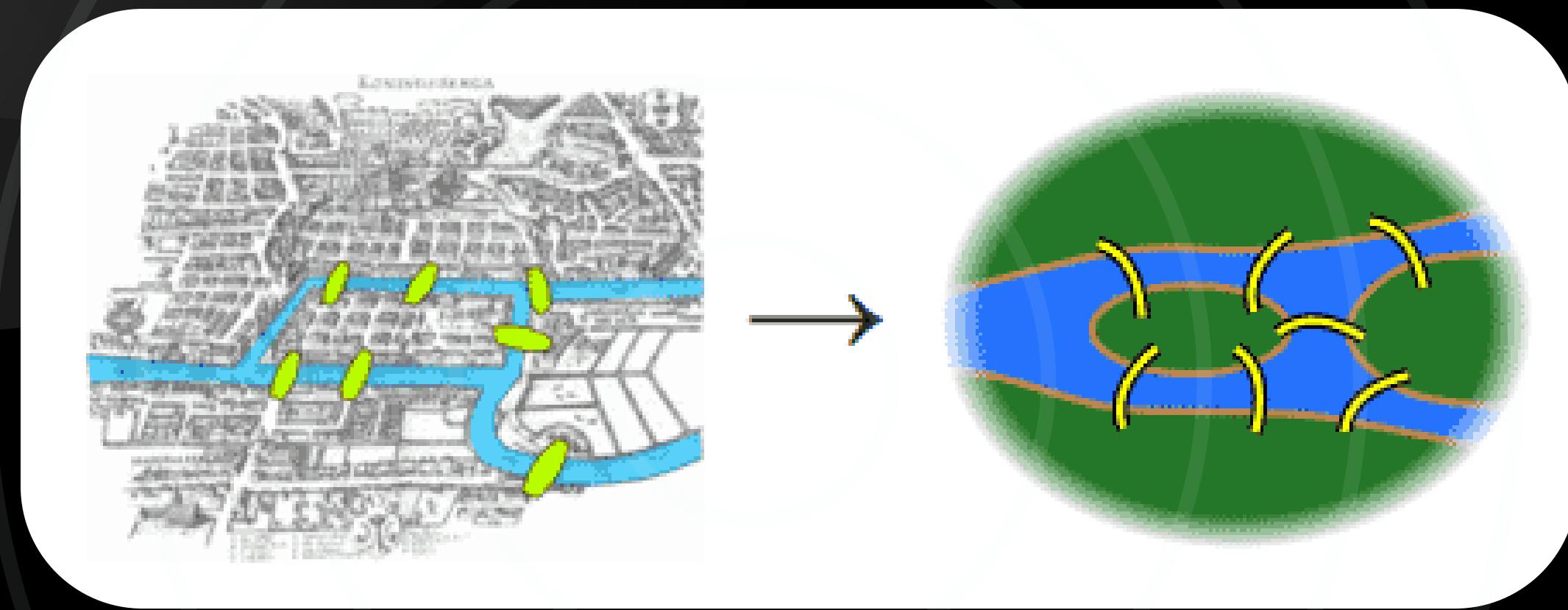


Leonhard Euler



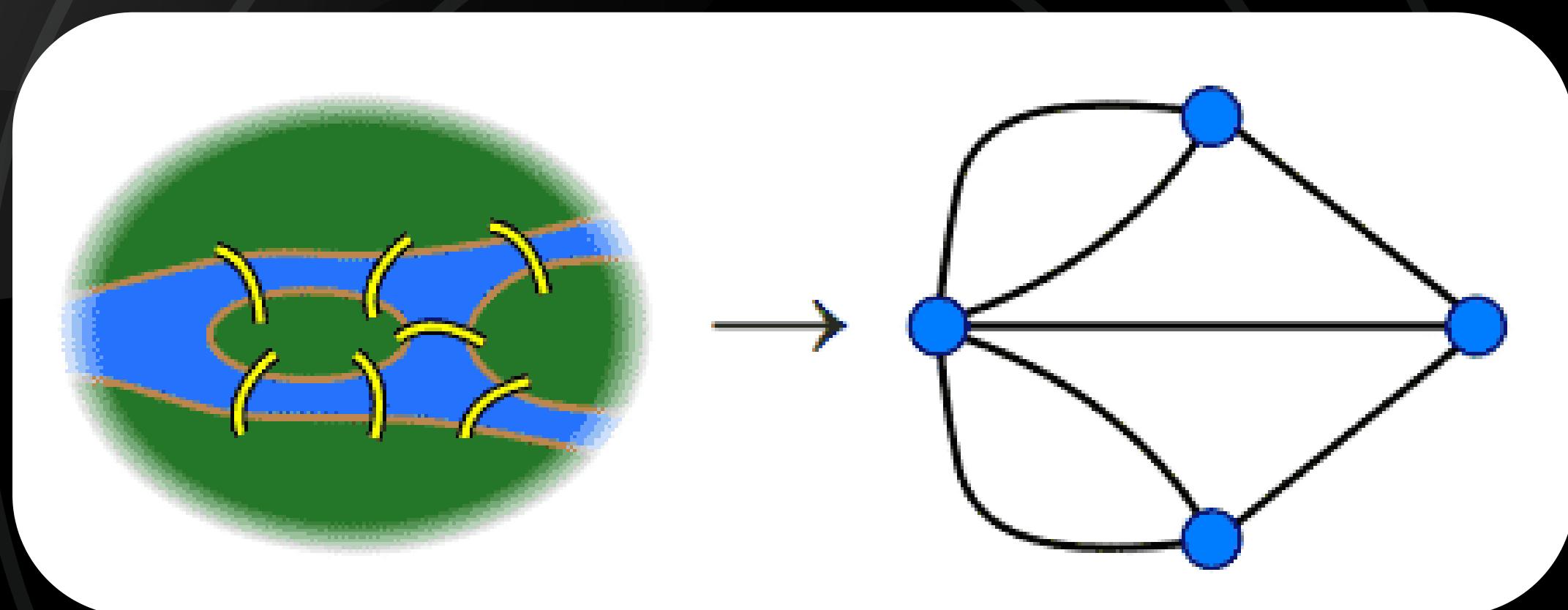
UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

*¿Se puede hacer un camino donde se pasen solo una vez cada puente?*



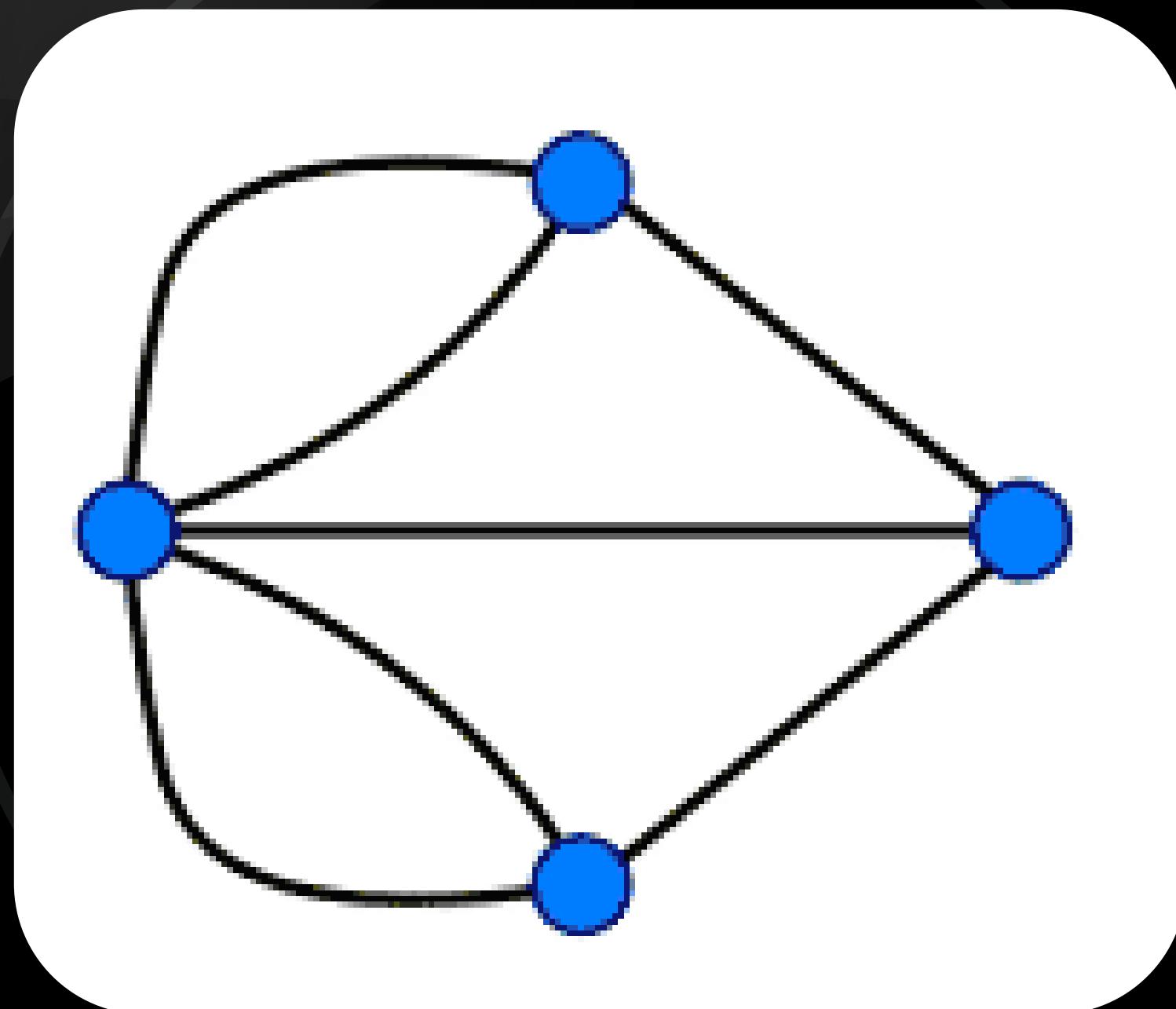
Leonhard Euler

*¿Se puede hacer un camino donde se pasen solo una vez cada puente?*



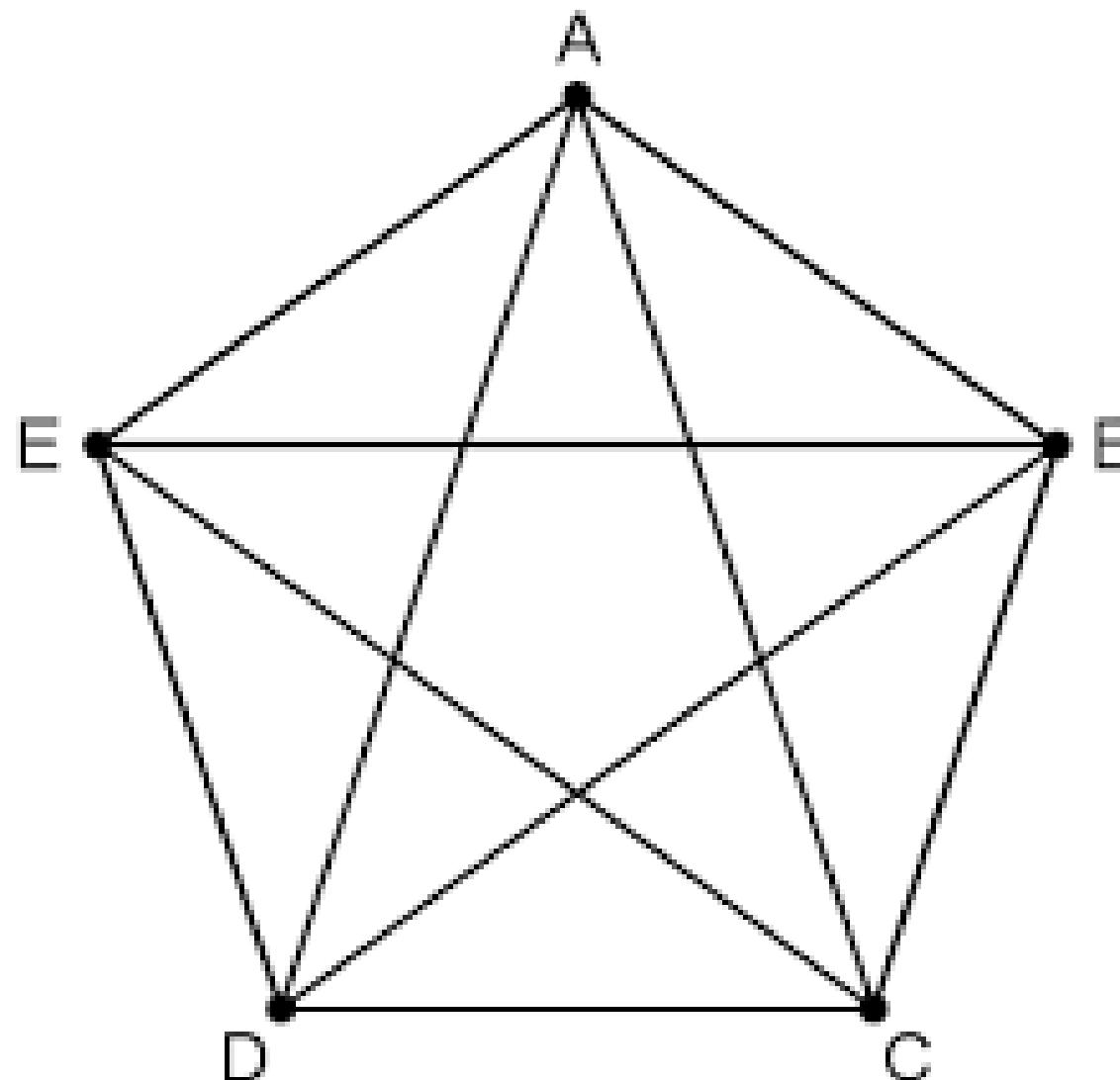
Leonhard Euler

*¿Se puede hacer un camino donde se pasen solo una vez cada puente?*



Leonhard Euler

# *El Teorema de Euler*



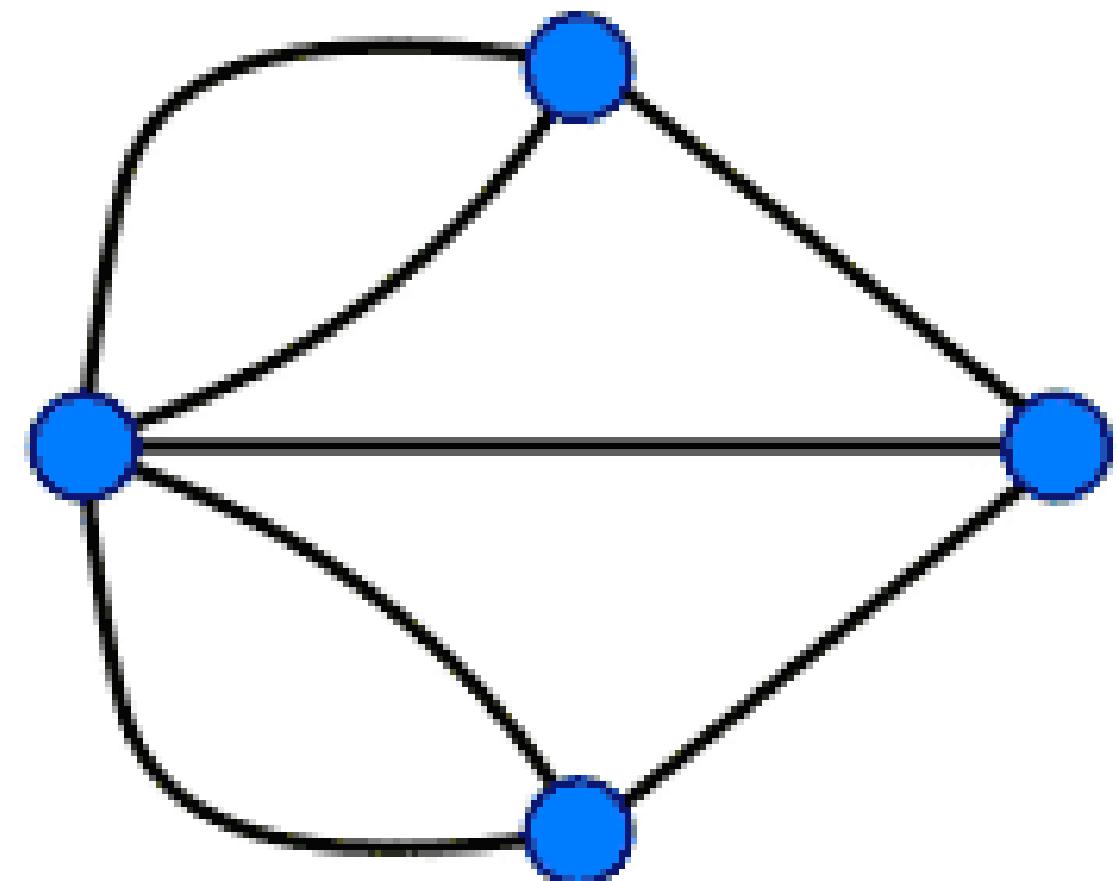
Es un Grafo Euleriano

Un grafo es Euleriano



Es conexo y todos sus vértices son pares.

# *El Teorema de Euler*



No es un Grafo Euleriano

Un grafo es Euleriano

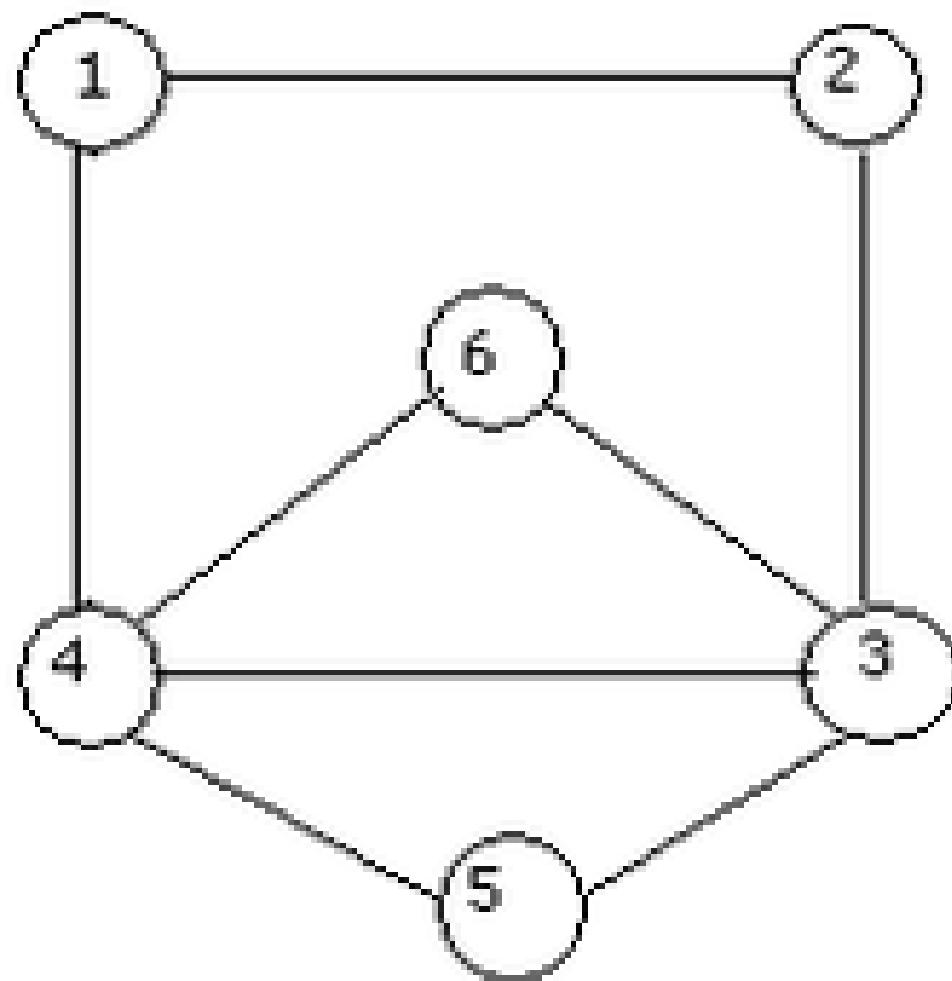


Es conexo y todos sus vértices son pares.



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# *El Teorema de Euler*



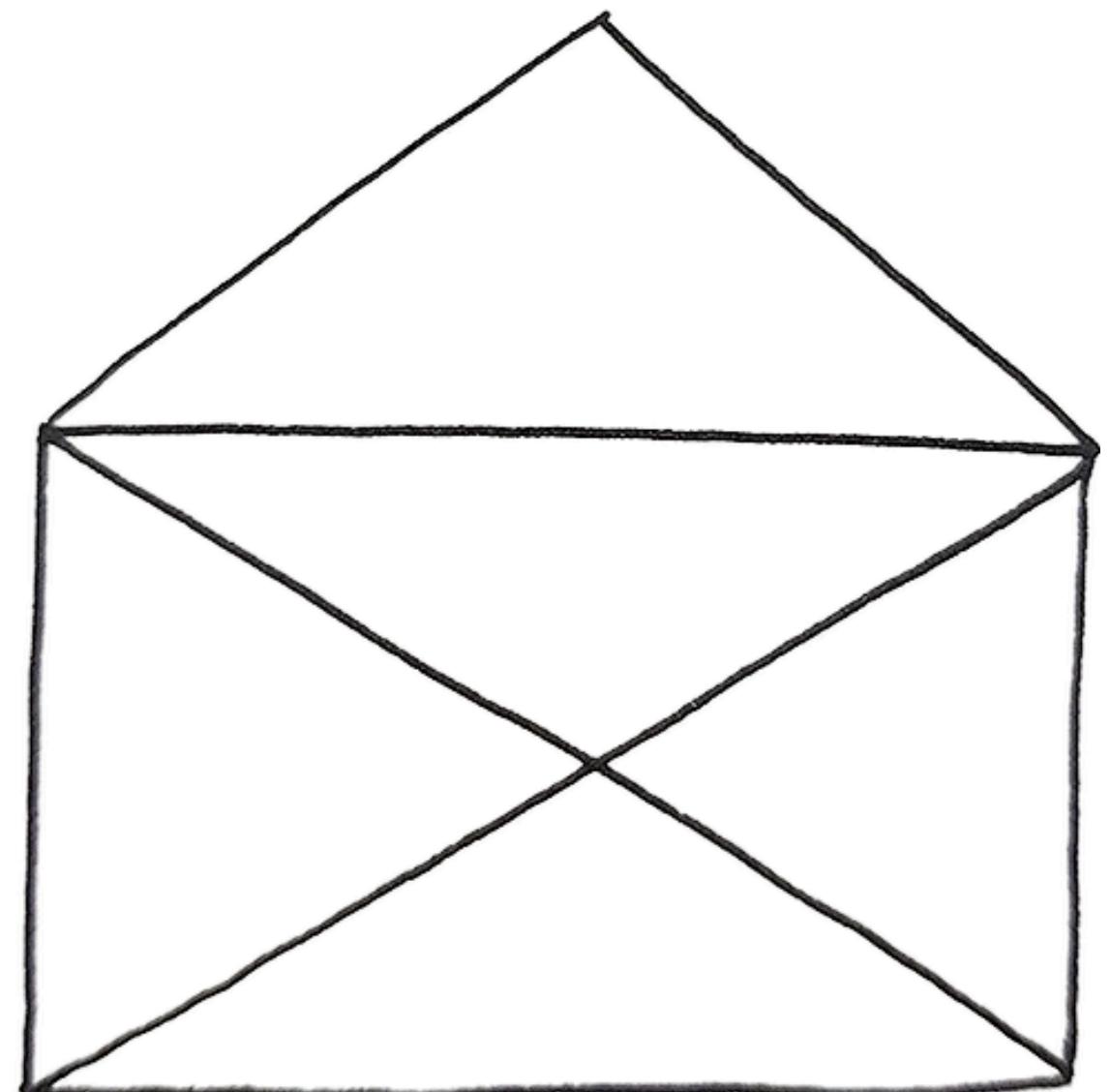
Es un Grafo Euleriano

Un grafo es Euleriano



Es conexo y todos sus vértices son pares.

# *El Teorema de Euler*



No es un Grafo Euleriano

Un grafo es Euleriano



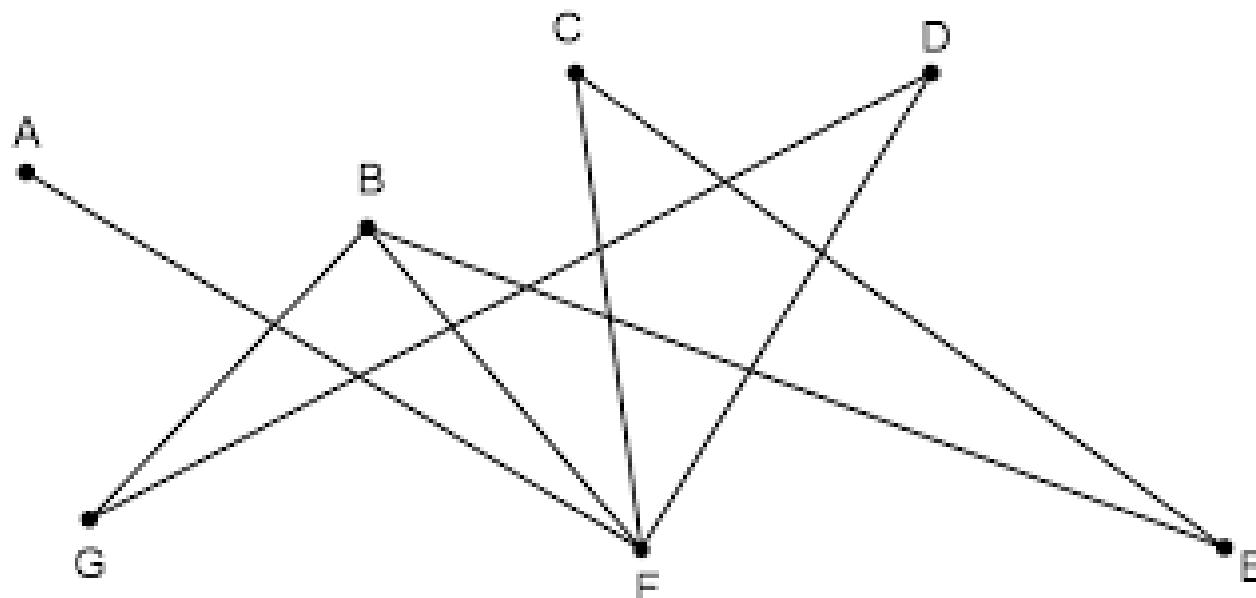
Es conexo y todos sus vértices son pares.



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# *El Teorema de Euler*

Círcuito de Euler NO es lo mismo que Camino de Euler



Un grafo es Euleriano



Es conexo y todos sus vértices son pares.

No es un grafo de Euler

No tiene circuito Euleriano (no todos los vértices tienen grado par).

Sí tiene camino Euleriano.

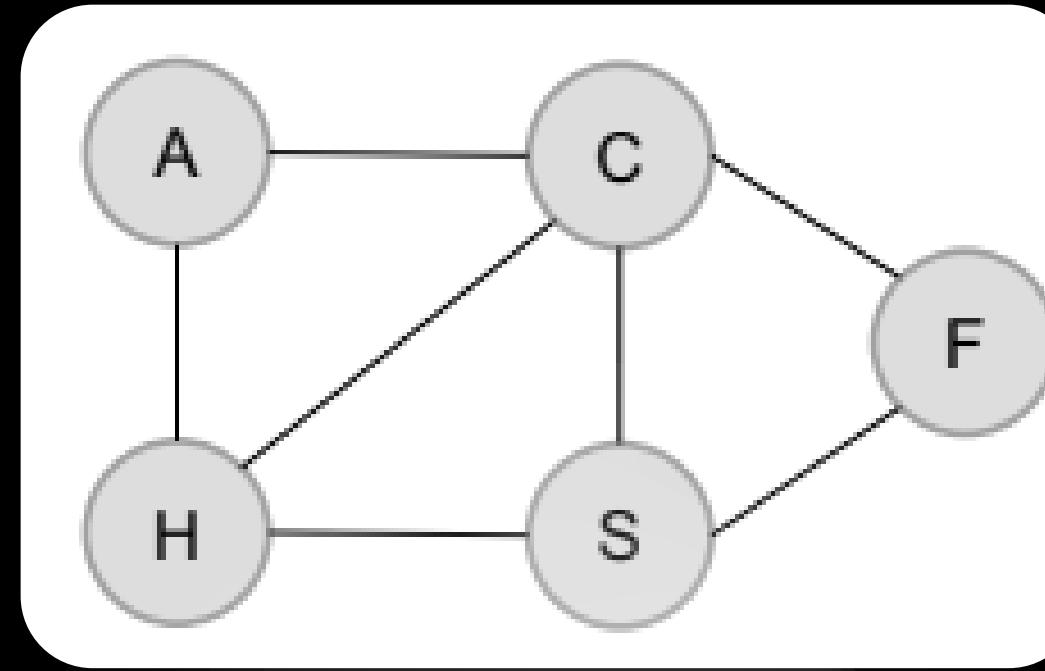
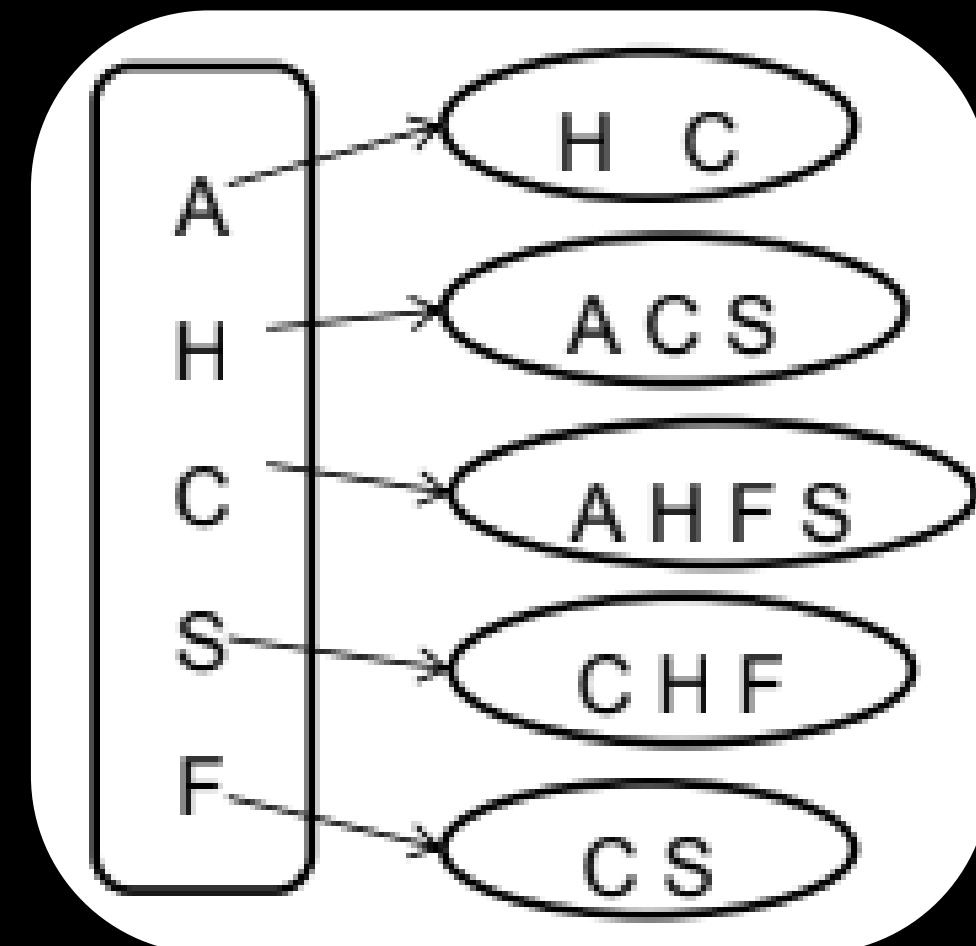
*¿CÓMO LOS  
PUEDO  
USAR?*



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# *IMPLEMENTACIÓN*

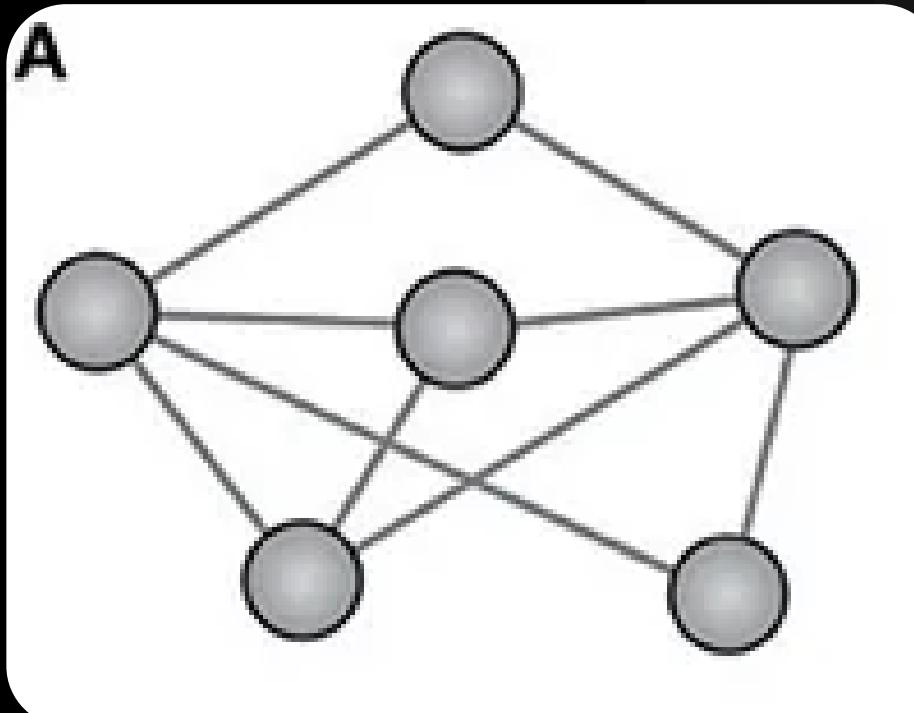
## Diccionarios



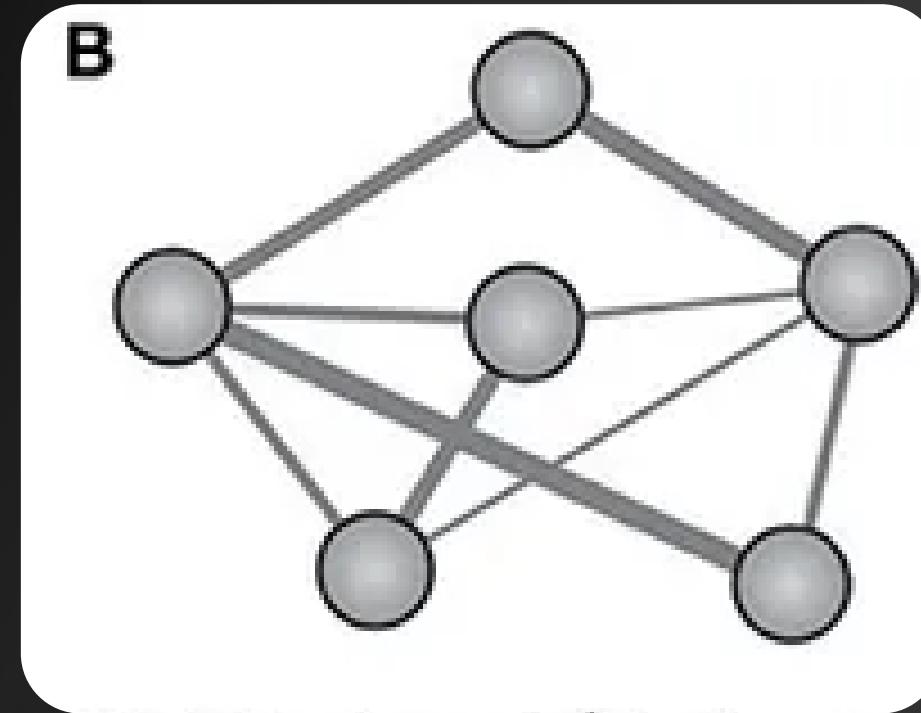
## Matriz Adyacencia

	A	C	F	S	H
A	-	1	0	0	1
C	1	-	1	1	1
F	0	1	-	1	0
S	0	1	1	-	1
H	1	1	0	1	-

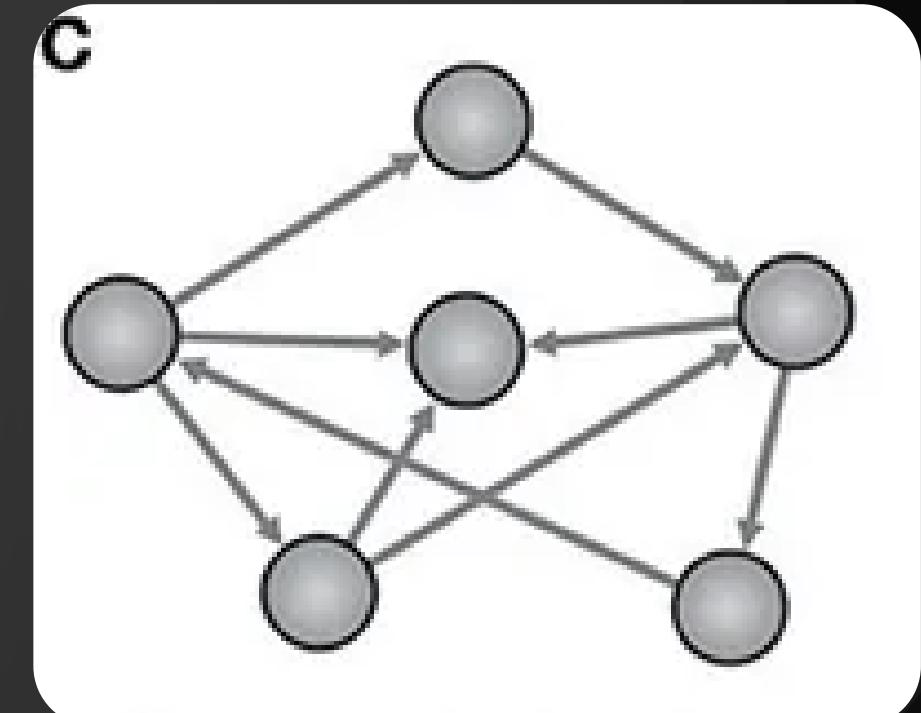
# *Algunos tipos de Grafos:*



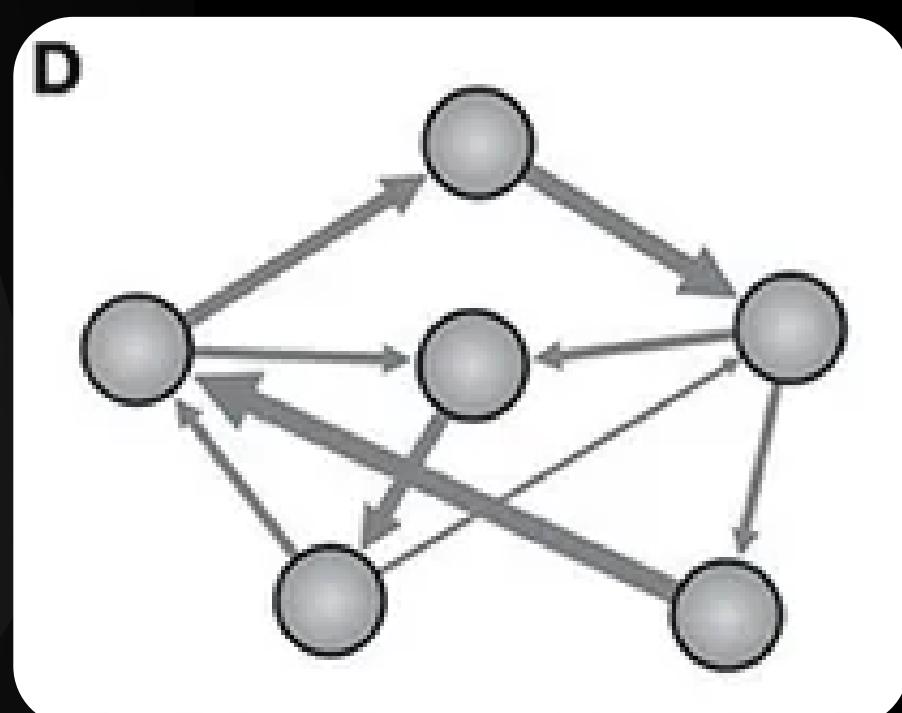
Grafo binario



Grafo con pesos



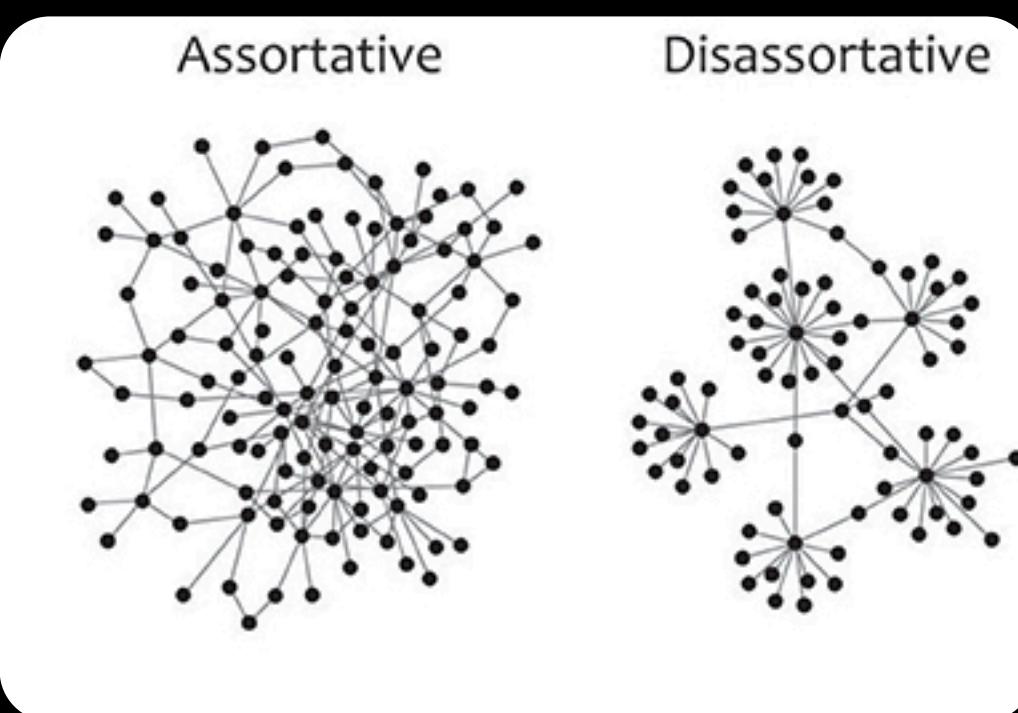
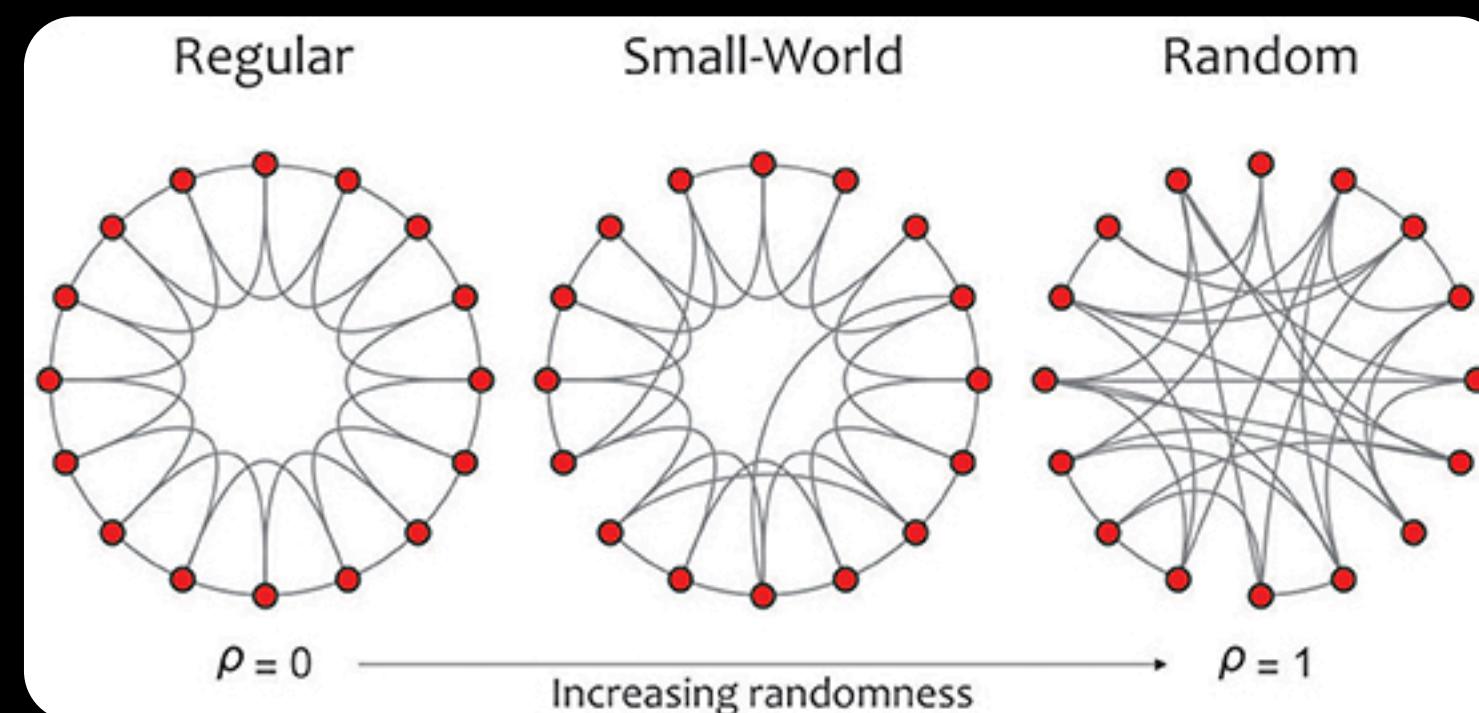
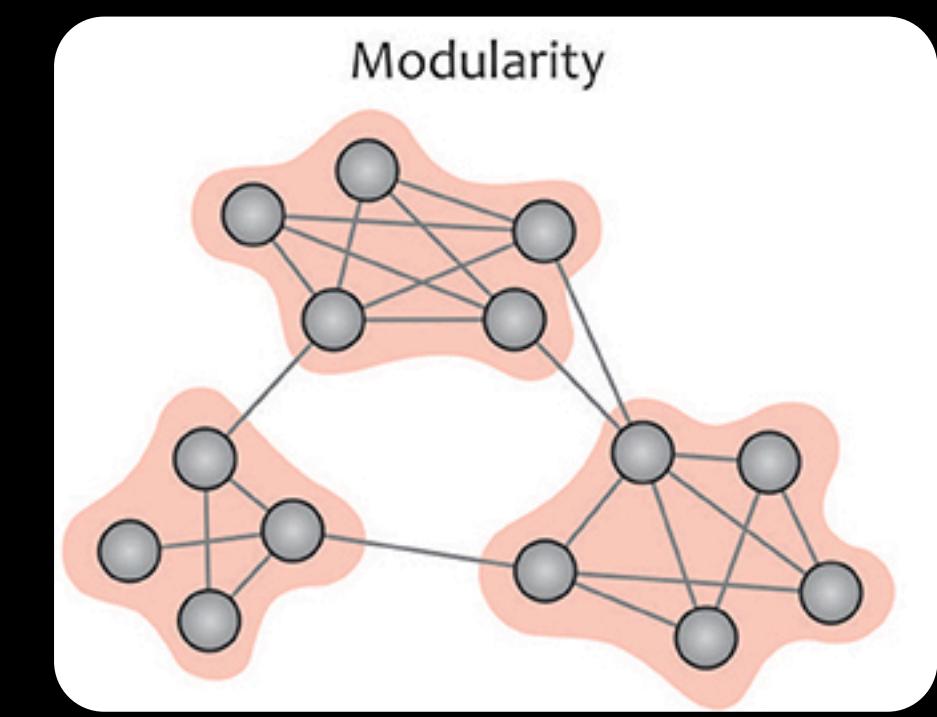
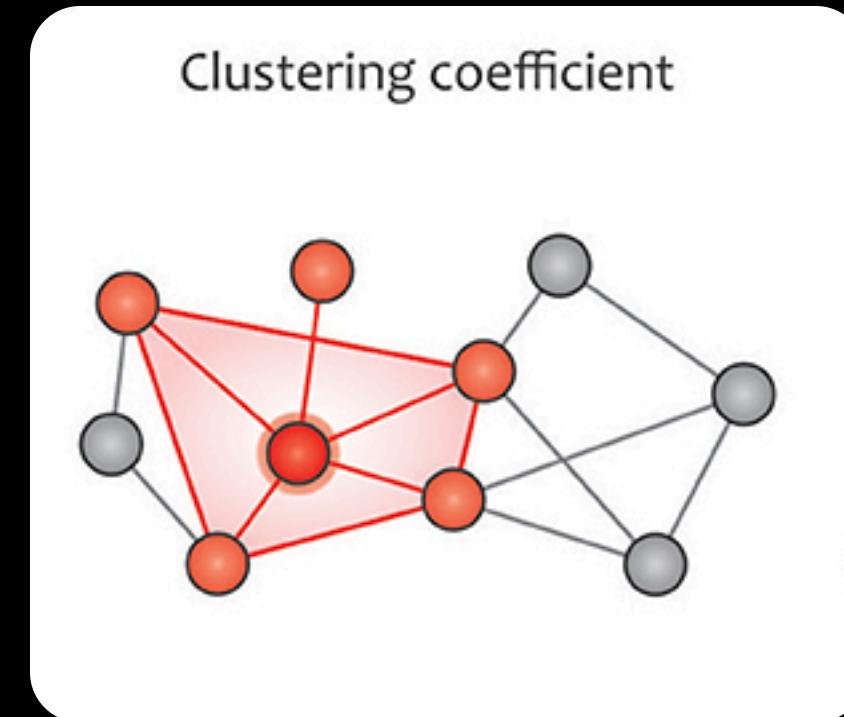
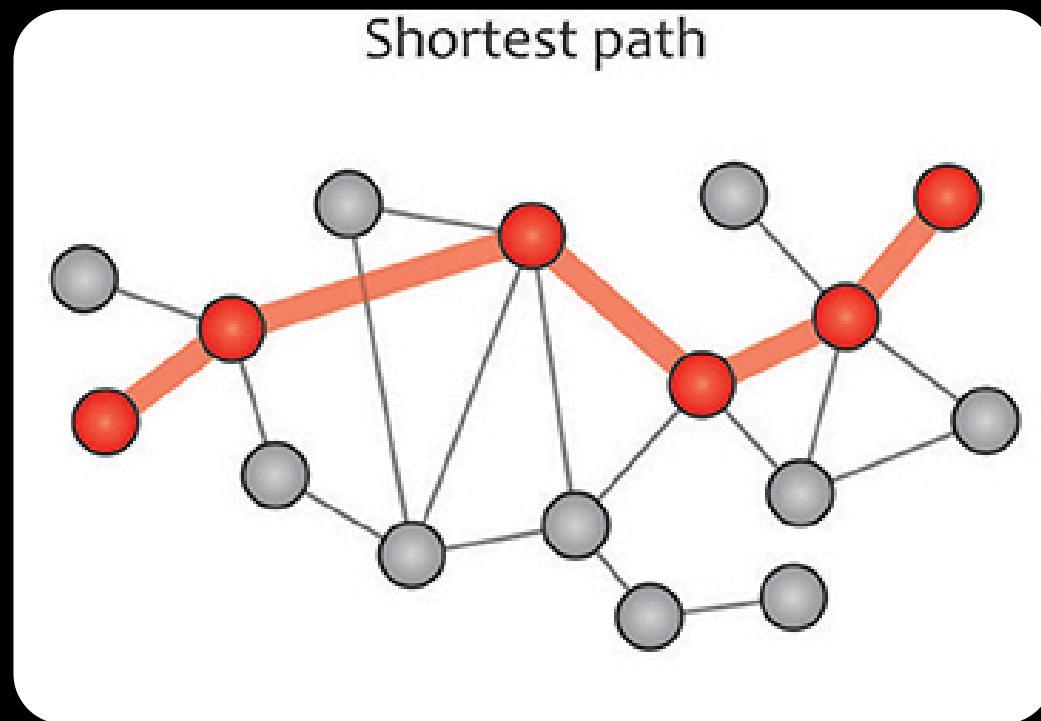
Grafo dirigido



Grafo con pesos y  
dirigido

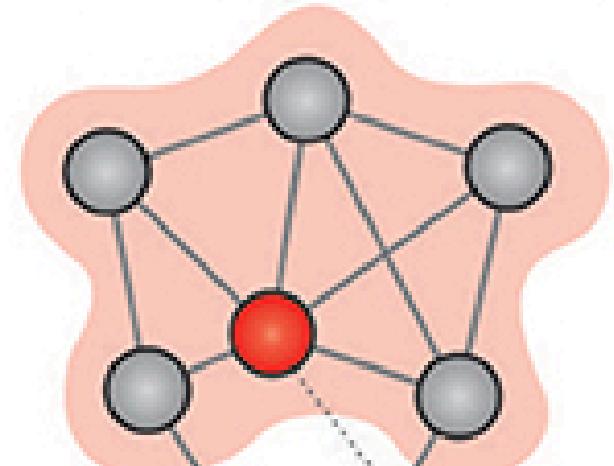
# MÉTRICAS

# Métricas Globales

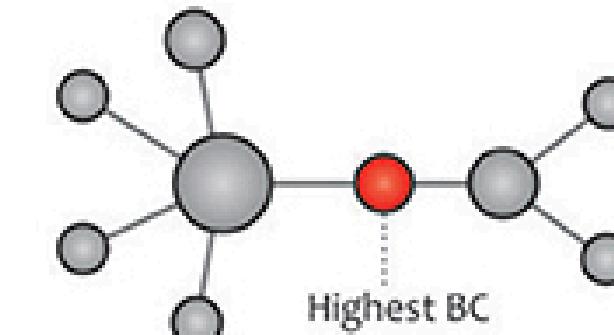


# Métricas Locales

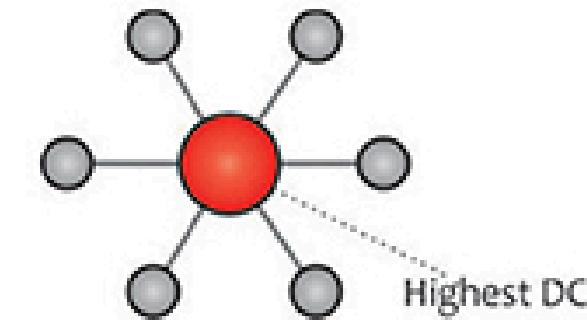
Centrality and hubs



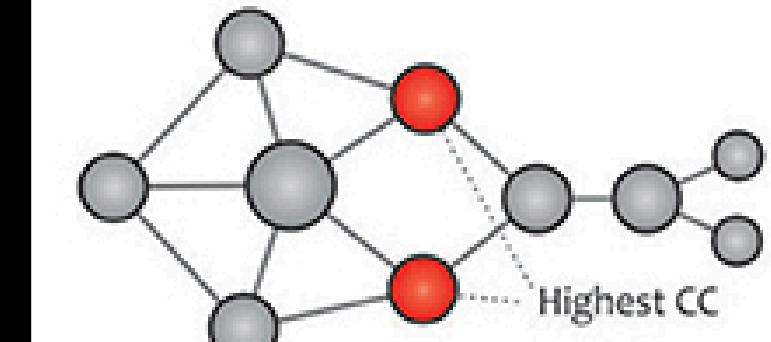
Betweenness centrality



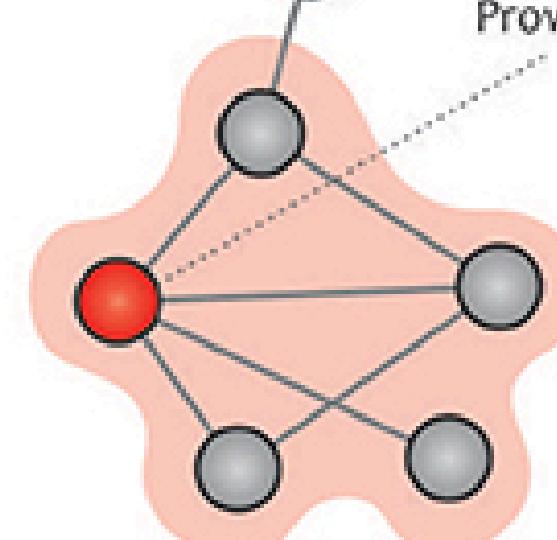
Degree centrality



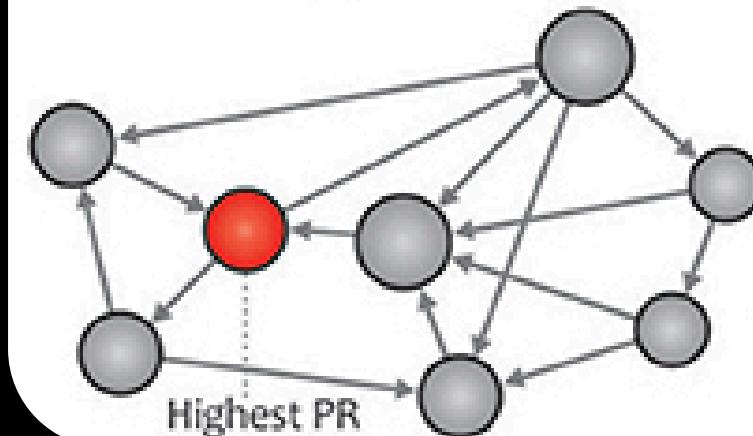
Closeness centrality



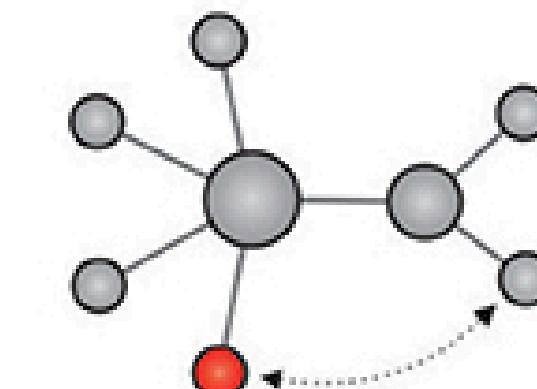
Connector hub.....  
Provincial hubs



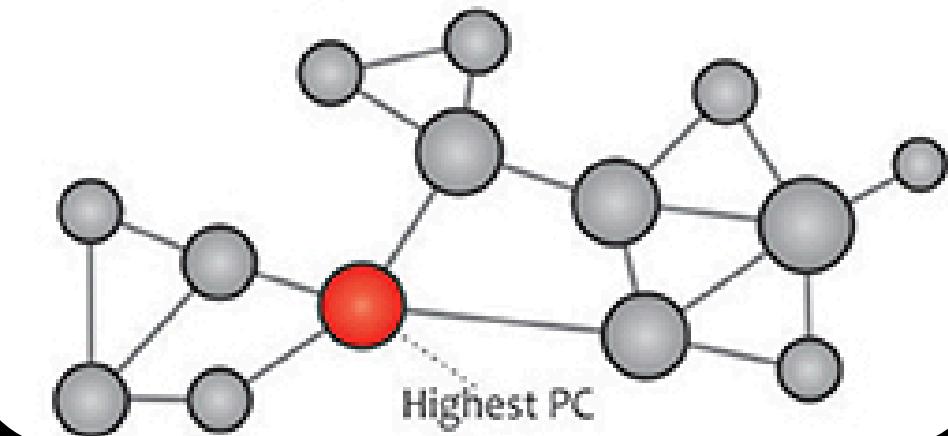
PageRank



Eigenvector centrality



Participation coefficient



# <ALGORITMOS>



UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

# *Recorridos de Grafos*

El objetivo es, dado un nodo de partida:

- 1) explorar tantos nodos restantes como sea posible pero
- 2) no explorar ningún nodo más de una vez.

**BFS (Breadth-First Search) o  
búsqueda en amplitud**

Explora los nodos por “capas”

Puede implementarse con una cola

**DFS (Depth-First Search) o  
búsqueda en profundidad**

Explora los nodos “agresivamente”

Puede implementarse con una pila

# Recorridos de Grafos

## <BFS>

```
BFS(grafo G, nodo a):  
    for each nodo u:  
        u.visited = false  
    a.visited = true  
    q = new queue  
    q.push(a)  
    while (q is not empty):  
        u = q.pop()  
        for each arista (u, v):  
            if v.visited = false:  
                v.visited = true  
                q.push(v)
```

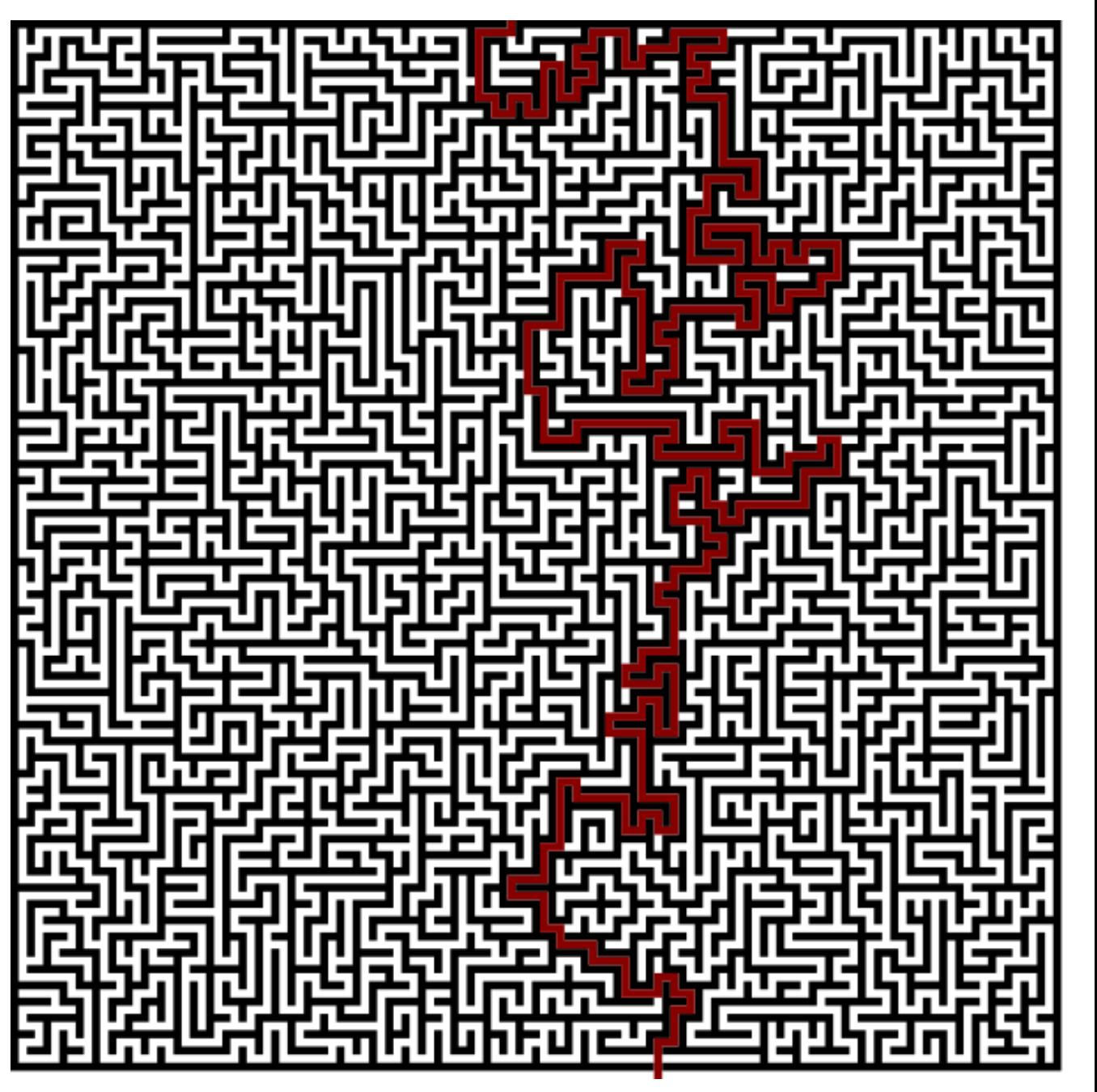
## <DFS>

```
DFS(grafo G, nodo a):  
    for each nodo u:  
        u.visited = false  
    s = new stack  
    s.push(a)  
    while (s is not empty):  
        u = s.pop()  
        if u.visited = false:  
            u.visited = true  
            for each arista (u, v):  
                if v.visited = false:  
                    s.push(v)
```

# *Cantidad Mínima de saltos*

El objetivo es, dado un nodo de partida, determinar cuál es la mínima cantidad de “saltos” o aristas por las que se debe pasar para llegar al resto de nodos

```
countHops(grafo G, nodo a):
    for each nodo u:
        u.visited = false
    a.visited = true
    a.hops = 0
    q = new queue
    q.push(a)
    while (q is not empty):
        u = q.pop()
        for each arista (u, v):
            if v.visited = false:
                v.visited = true
                v.hops = u.hops + 1
                q.push(v)
```

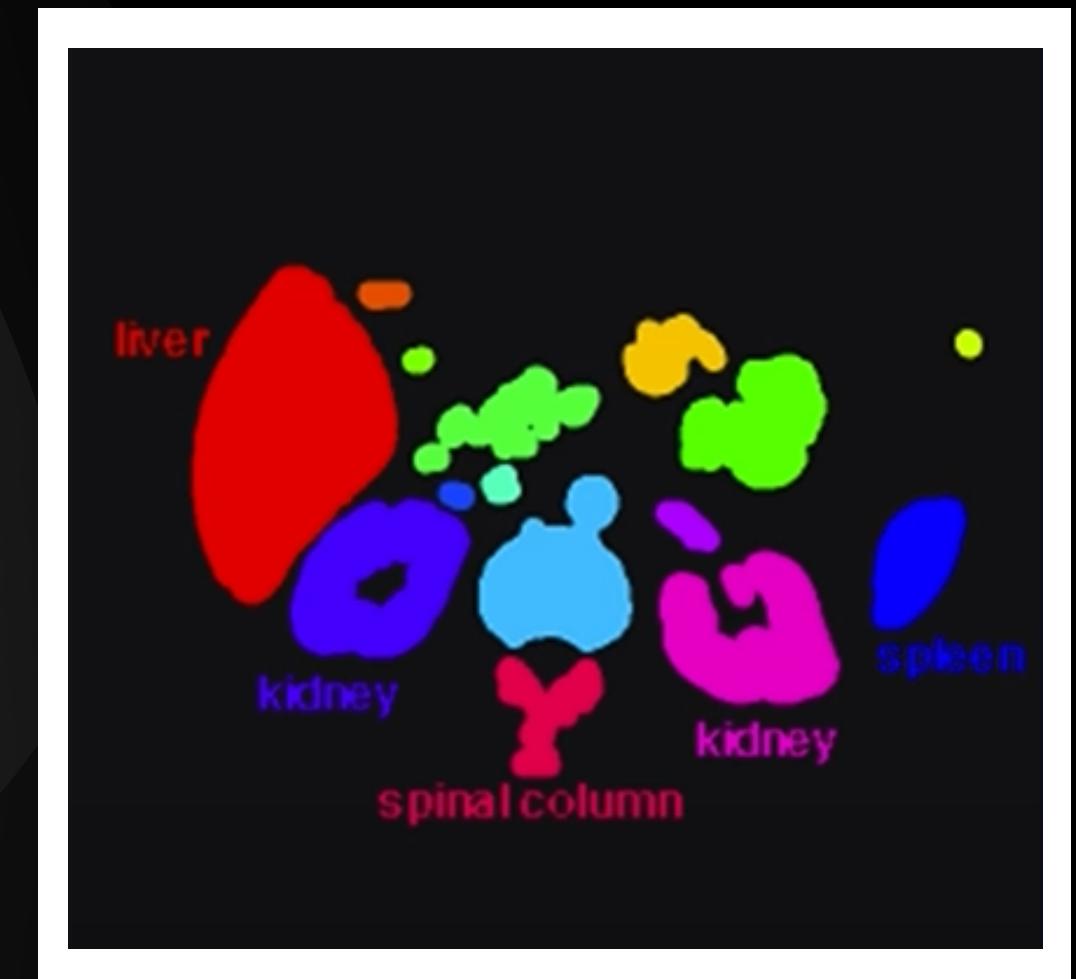
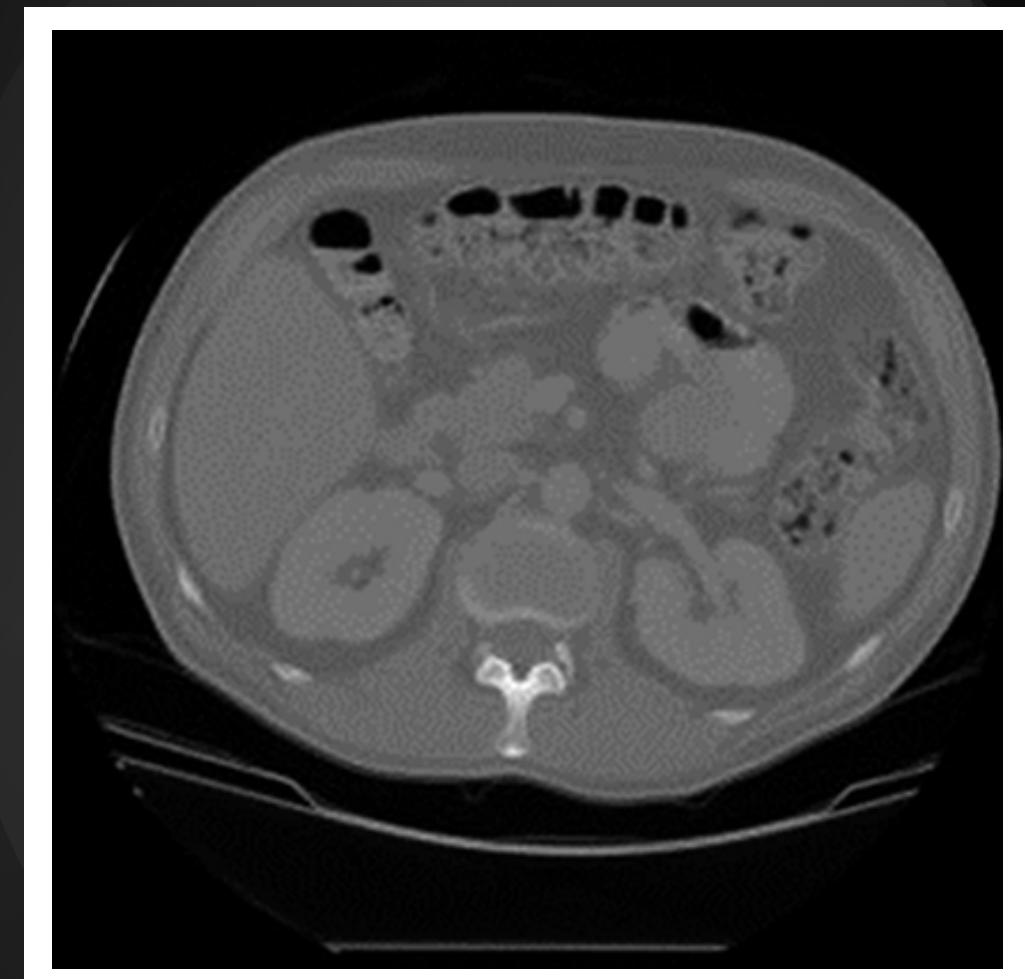


*Resolver un laberinto*

# Cantidad Mínima de saltos

El objetivo es, dado un grafo, ¿cuántos subgrafos conectados hay?

```
connectedComponents(grafo G):
    for each nodo u:
        u.visited = false
    c = 0
    for each nodo a:
        if a.visited = false:
            c += 1
            s = new stack
            s.push(a)
            while (s is not empty):
                u = s.pop()
                if u.visited = false:
                    u.visited = true
                    for each arista (u, v):
                        if v.visited = false:
                            s.push(v)
```



Este algoritmo aporta la segmentación de imágenes médicas

# EL CAMINO MÁS CORTO

**Input:** Un grafo simple conexo y con pesos en el que todos los pesos son positivos.

**Output:**  $L(z)$ , la longitud del camino más corto de  $a$  a  $z$

```
1: function DIJKSTRA( $w, a, z, L$ )
2:   for all  $x \in \mathcal{V}$  do
3:      $L(x) = \infty$ 
4:    $L(a) = 0$ 
5:    $T =$  conjunto de todos los vértices
6:   while  $z \in T$  do
7:     Tomar  $v \in T$  con mínimo  $L(v)$ .
8:      $T = T - \{v\}$ 
9:     for  $x \in T$  adyacente con  $v$  do
10:       $L(x) = \min\{L(x), L(v) + w(v, x)\}$ 
11:   return  $L(z)$ 
```

El algoritmo de Dijkstra es la base de todos los algoritmos de caminos más cortos. Representa una introducción fundamental para entender cómo funcionan los algoritmos en grafos, por eso lo estudiaremos hoy.

## ALGORITMO DE DIJKSTRA

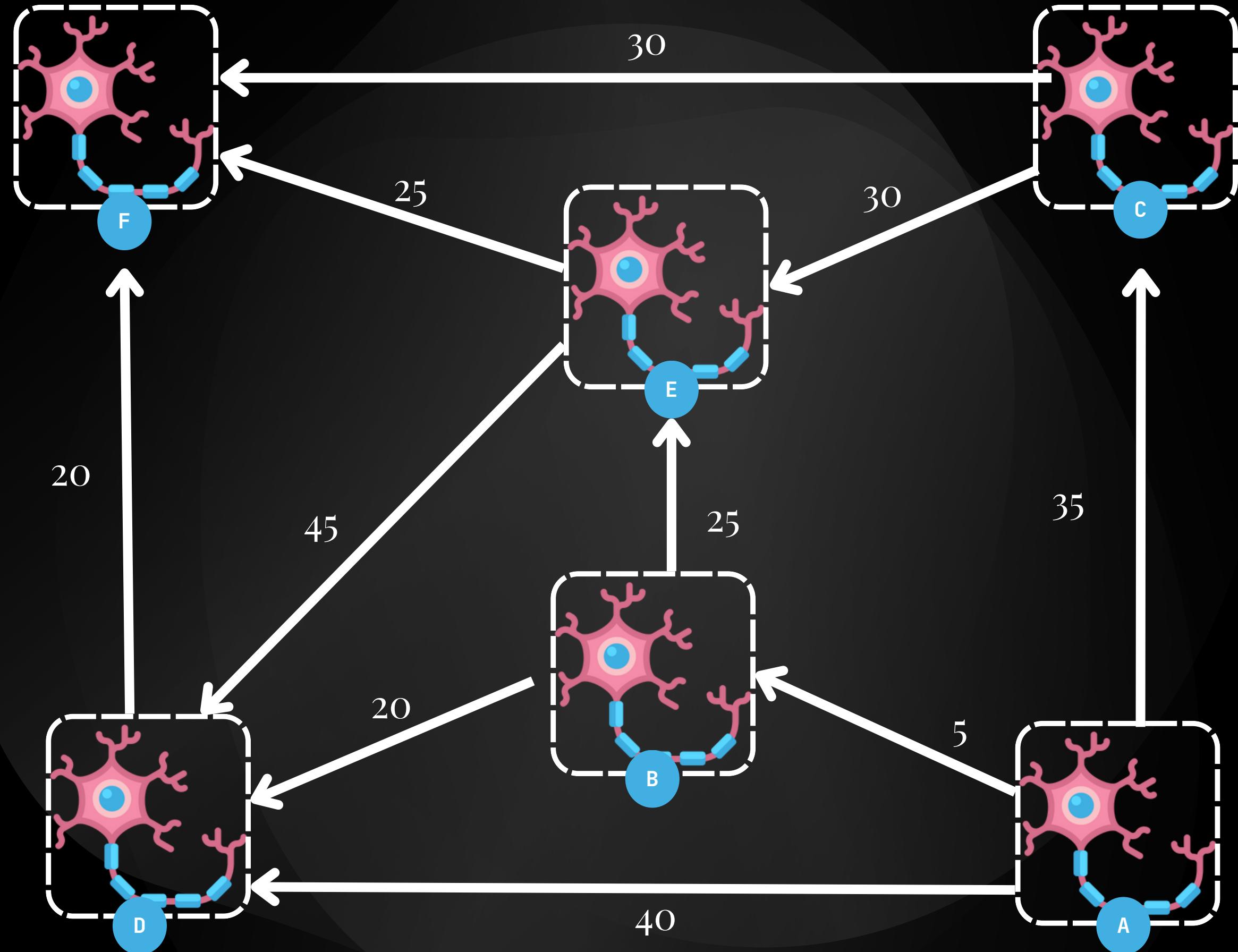


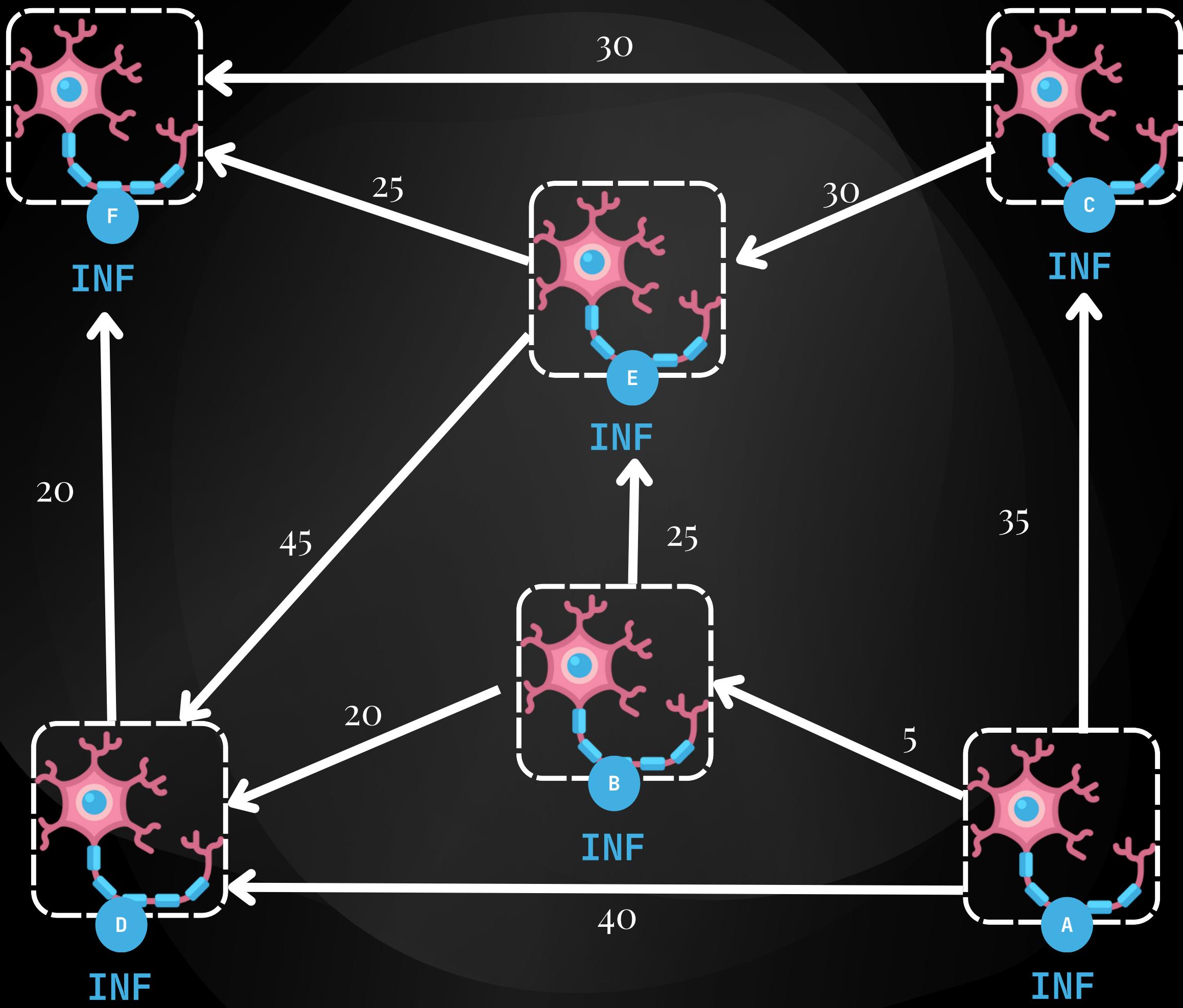
UNIVERSIDAD  
NACIONAL  
DE COLOMBIA

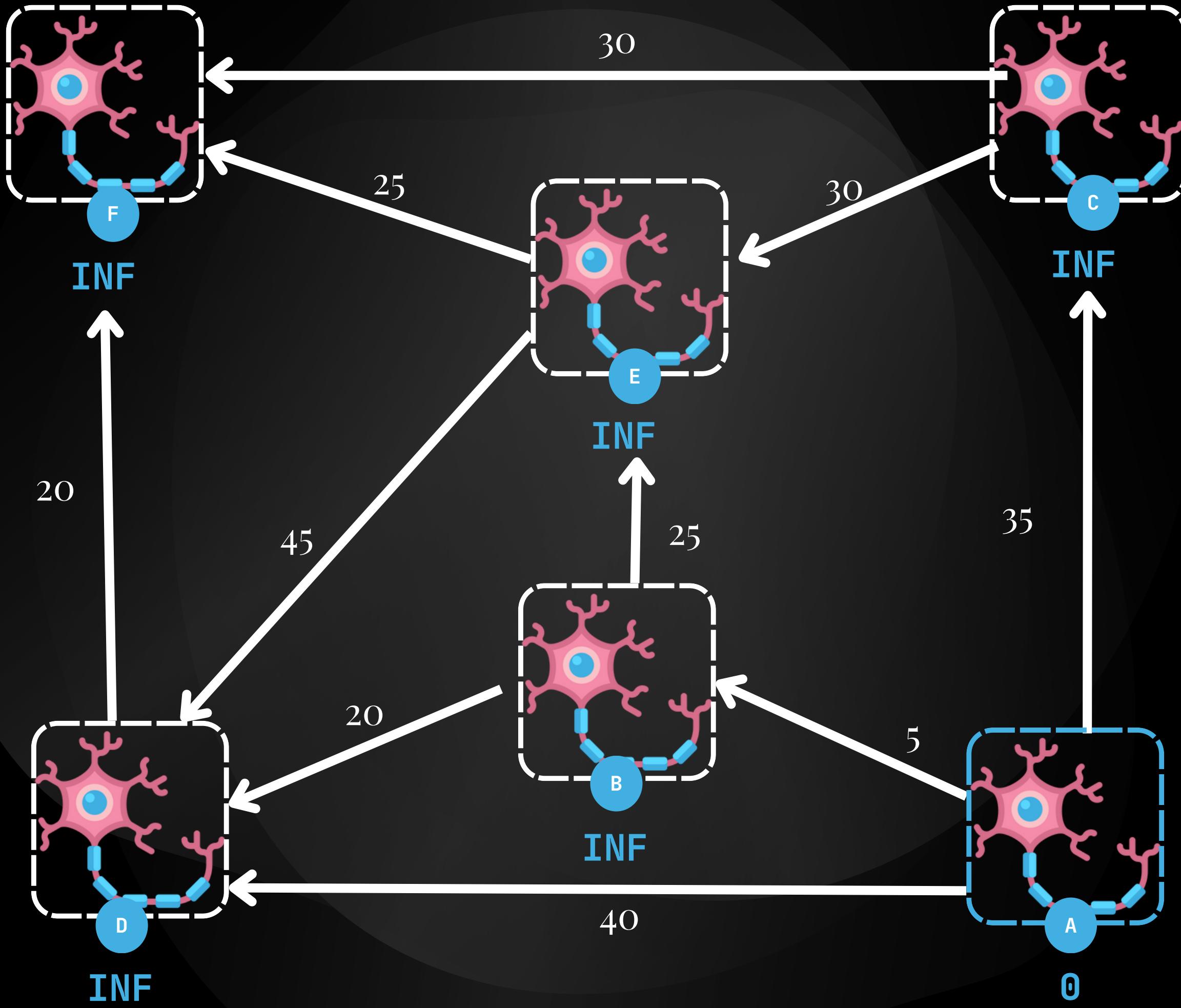
*¿Cuál es el camino más corto entre A y F?*

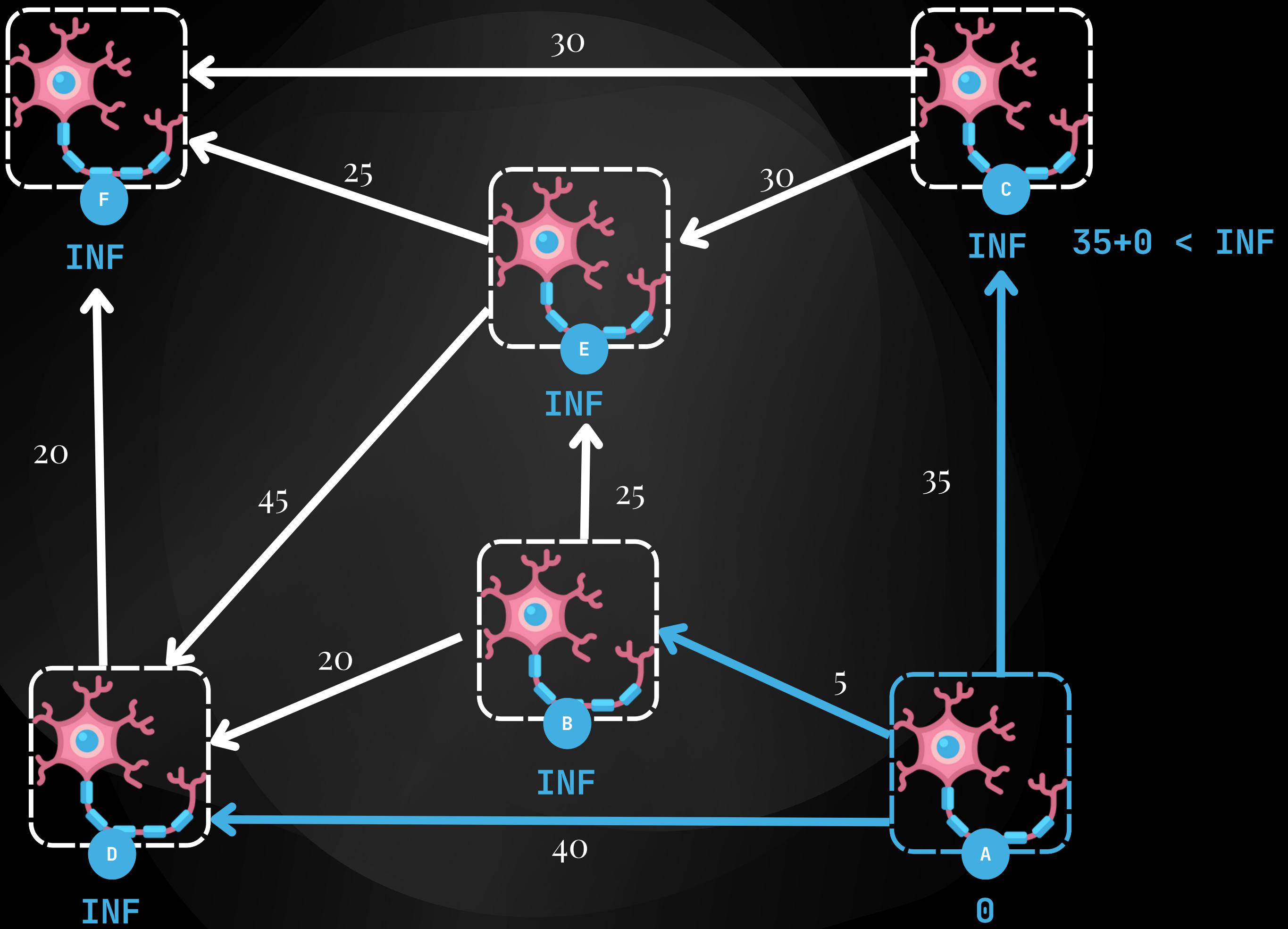


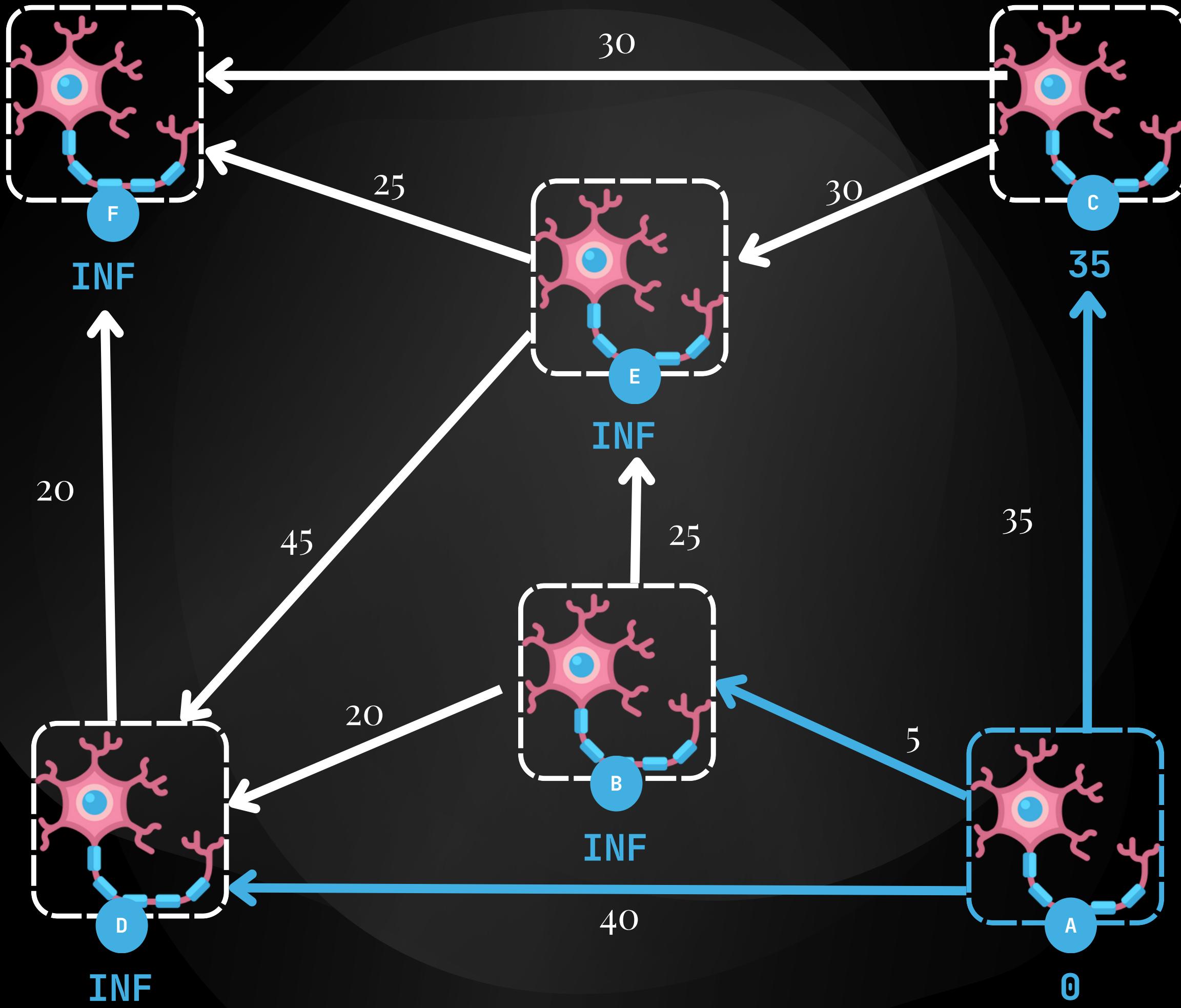
Edsger Dijkstra











Predecesores

$$A \rightarrow B$$

$$A \rightarrow C$$

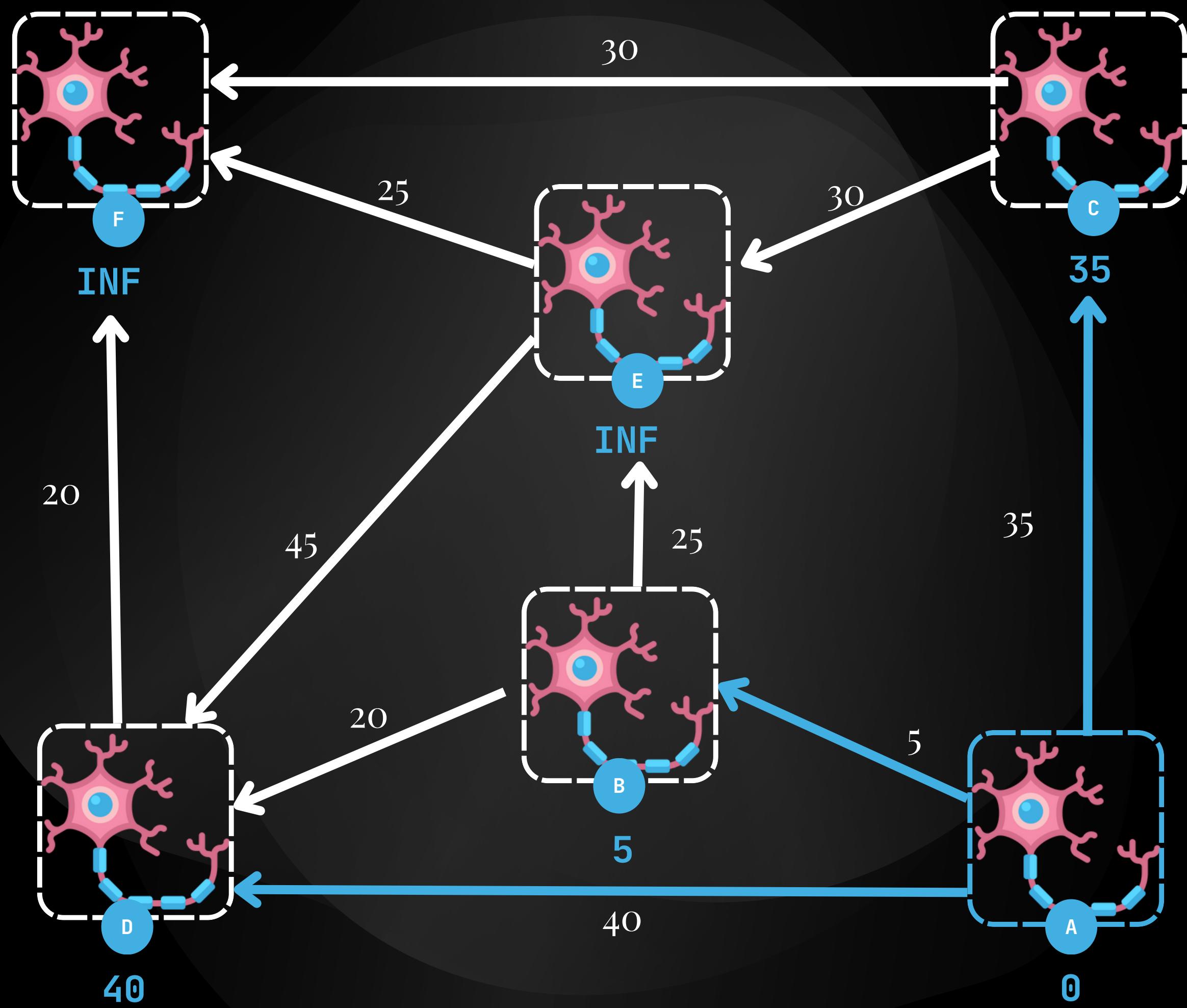
$$A \rightarrow D$$

Fila prioridad

$$B = 5$$

$$C = 35$$

$$D = 40$$



Predecesores

$$A \rightarrow B$$

$$A \rightarrow C$$

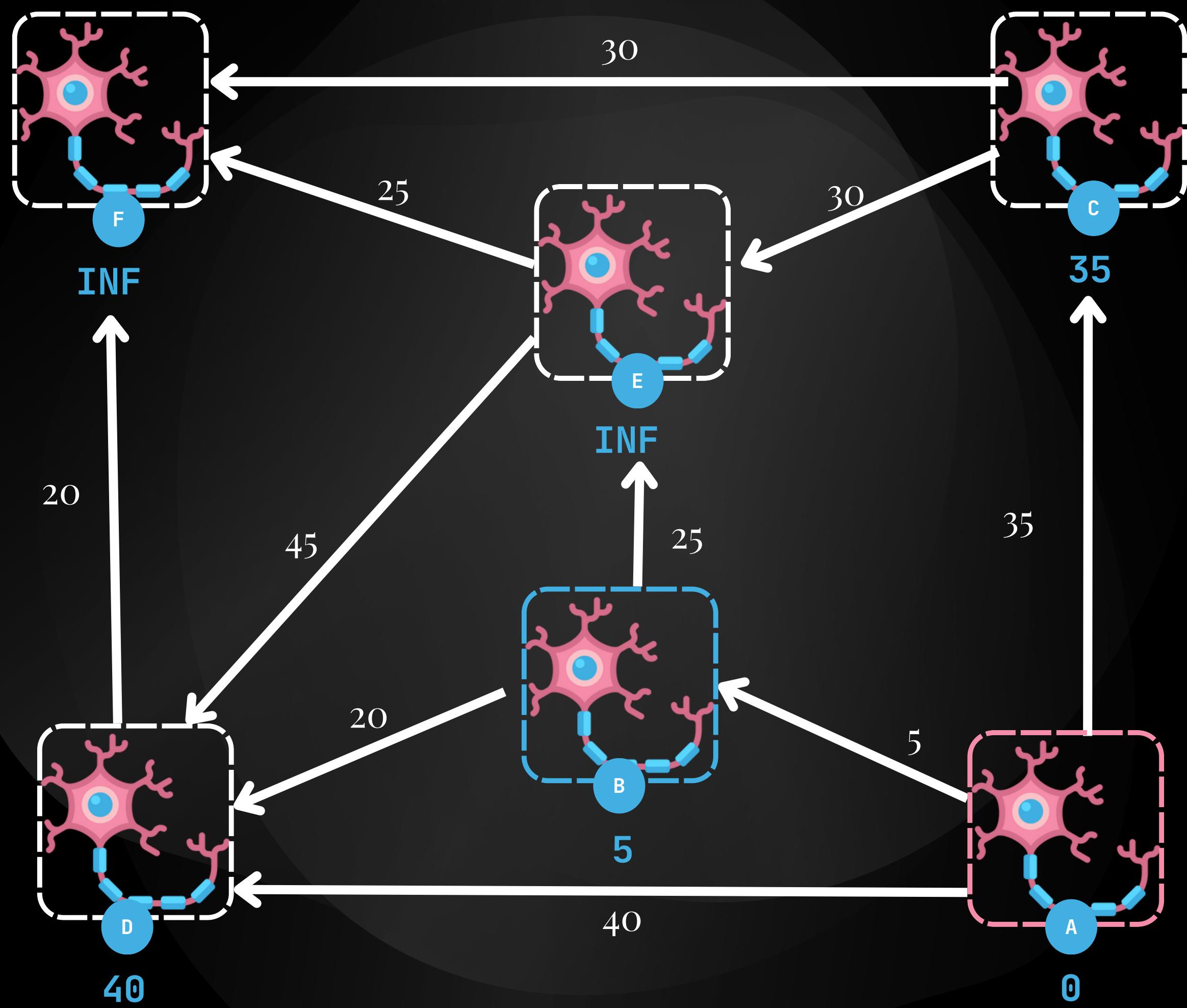
$$A \rightarrow D$$

Fila prioridad

$$B = 5$$

$$C = 35$$

$$D = 40$$



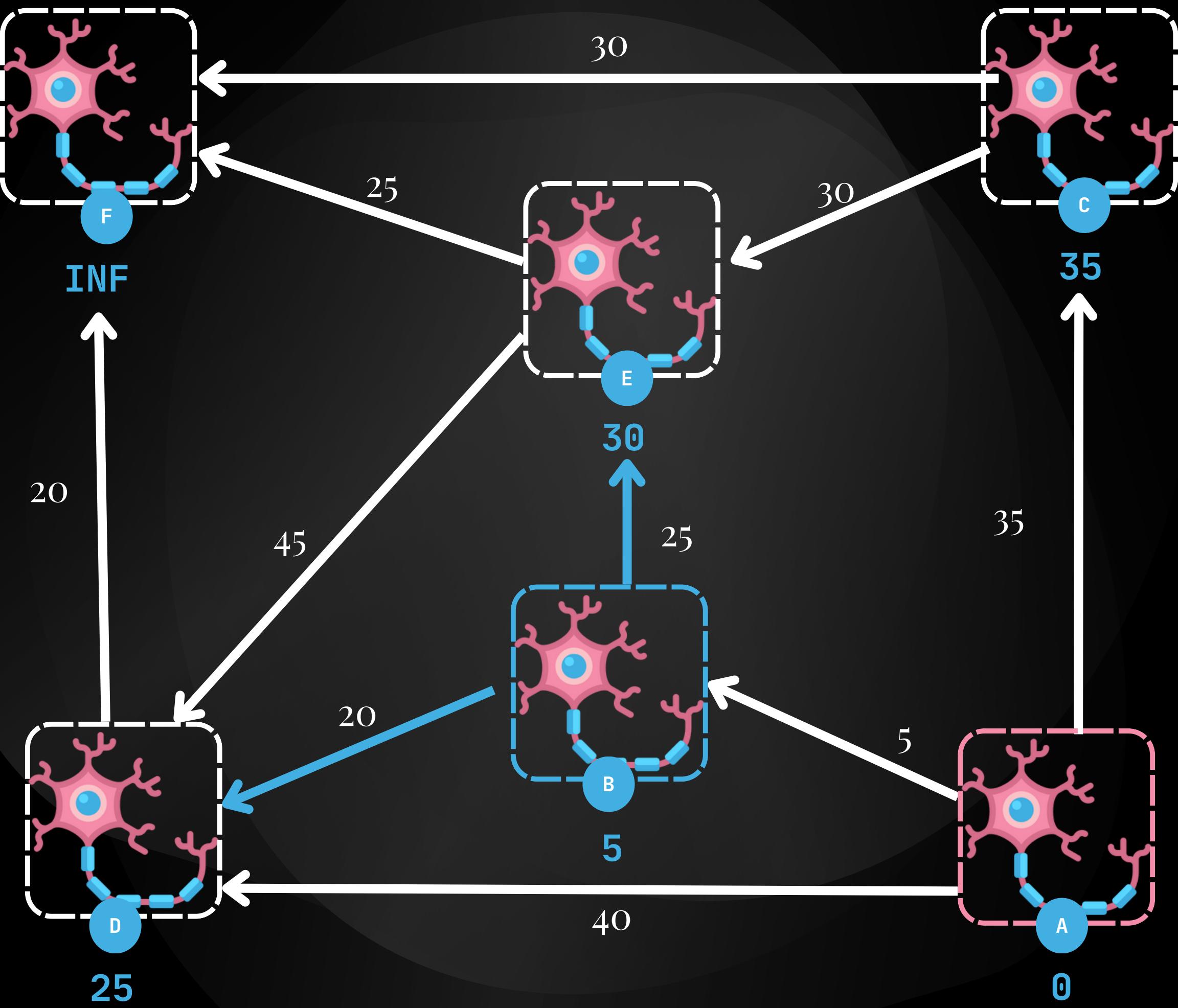
Predecesores

$$A \rightarrow B$$

$$A \rightarrow C$$

$$B \rightarrow D$$

$$B \rightarrow E$$



Fila prioridad

$$D = 25$$

$$E = 30$$

$$C = 35$$

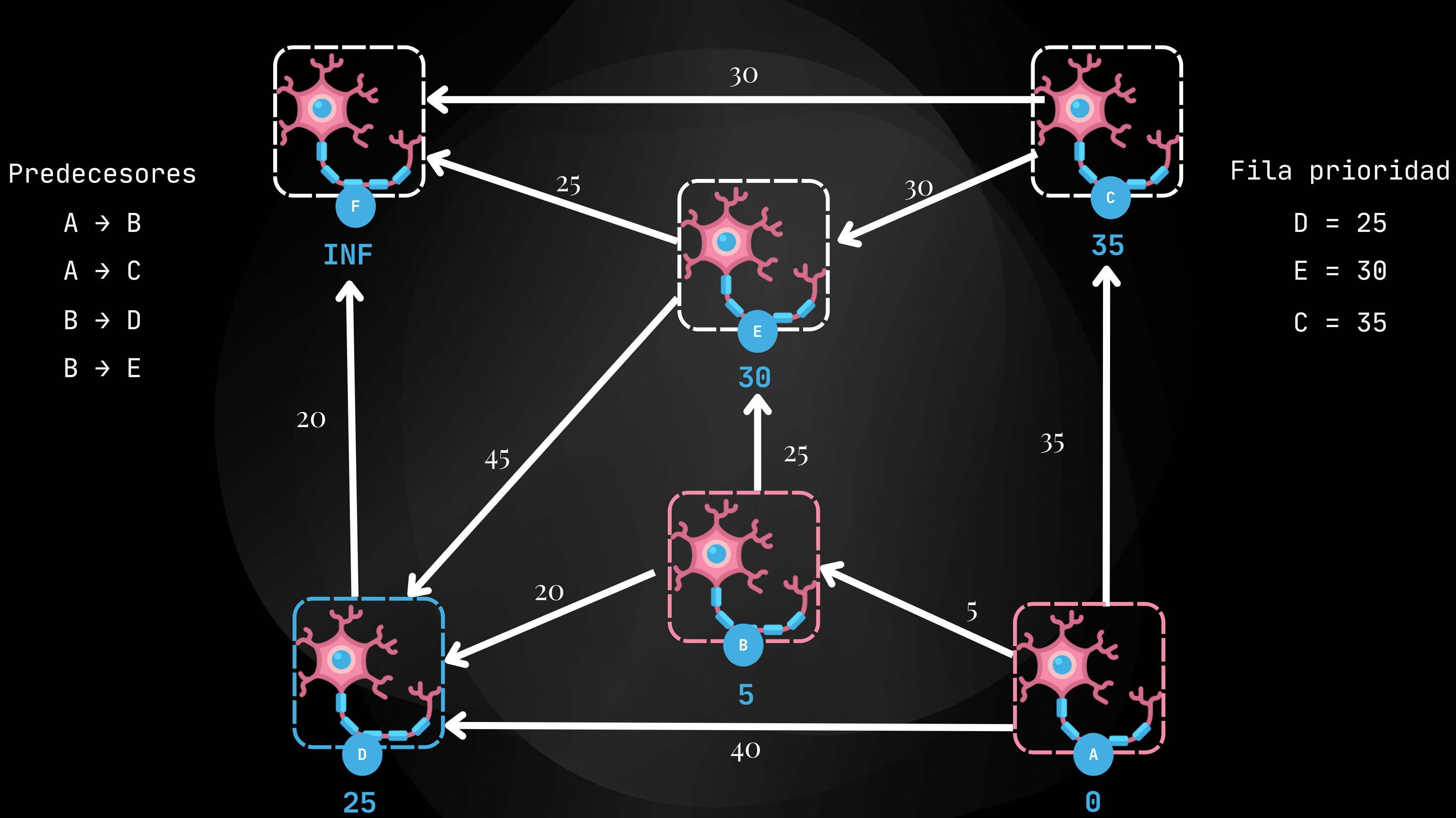
Predecesores

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow D$

$B \rightarrow E$



Predecesores

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow D$

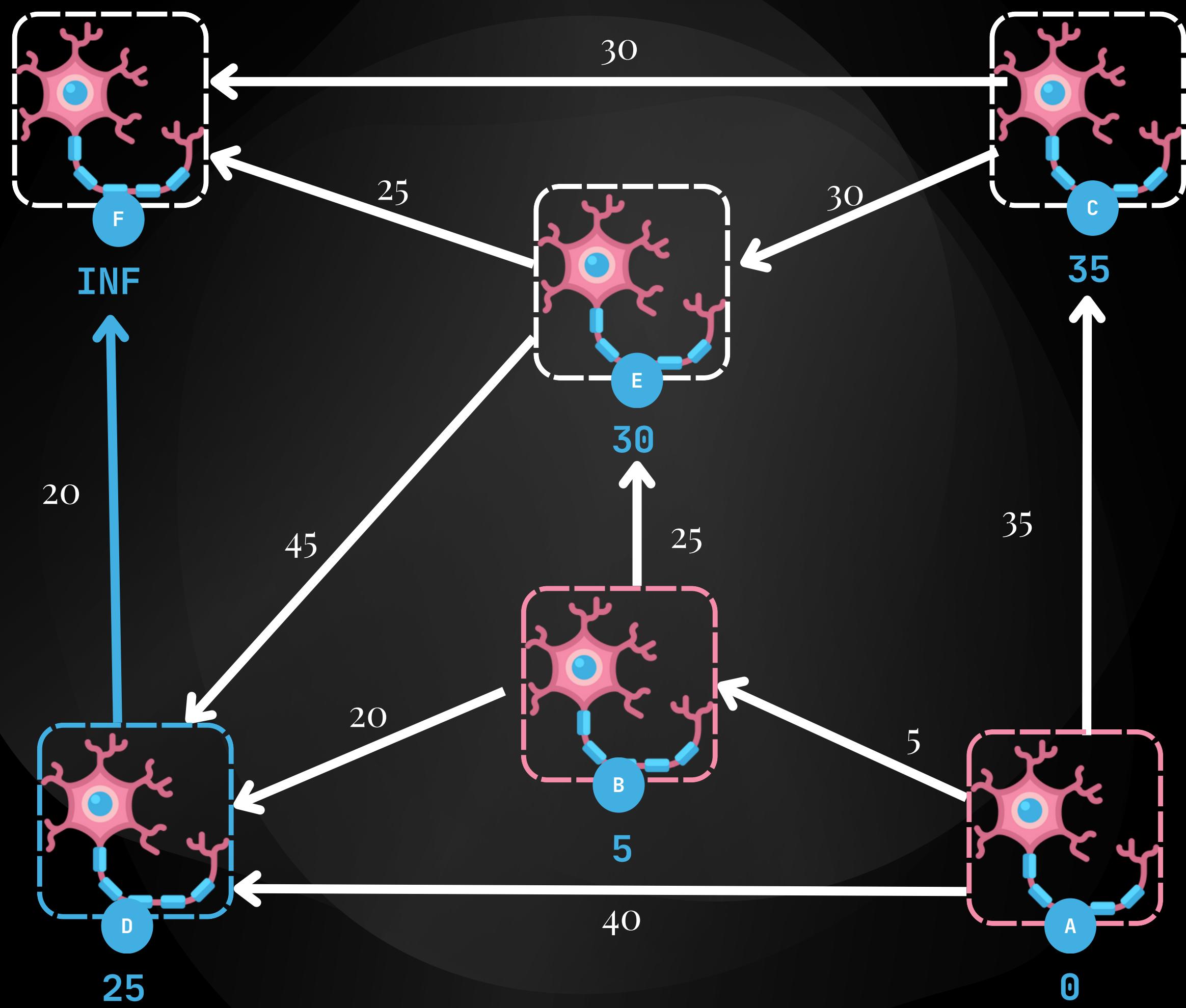
$B \rightarrow E$

Fila prioridad

$D = 25$

$E = 30$

$C = 35$



Predecesores

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow D$

$B \rightarrow E$

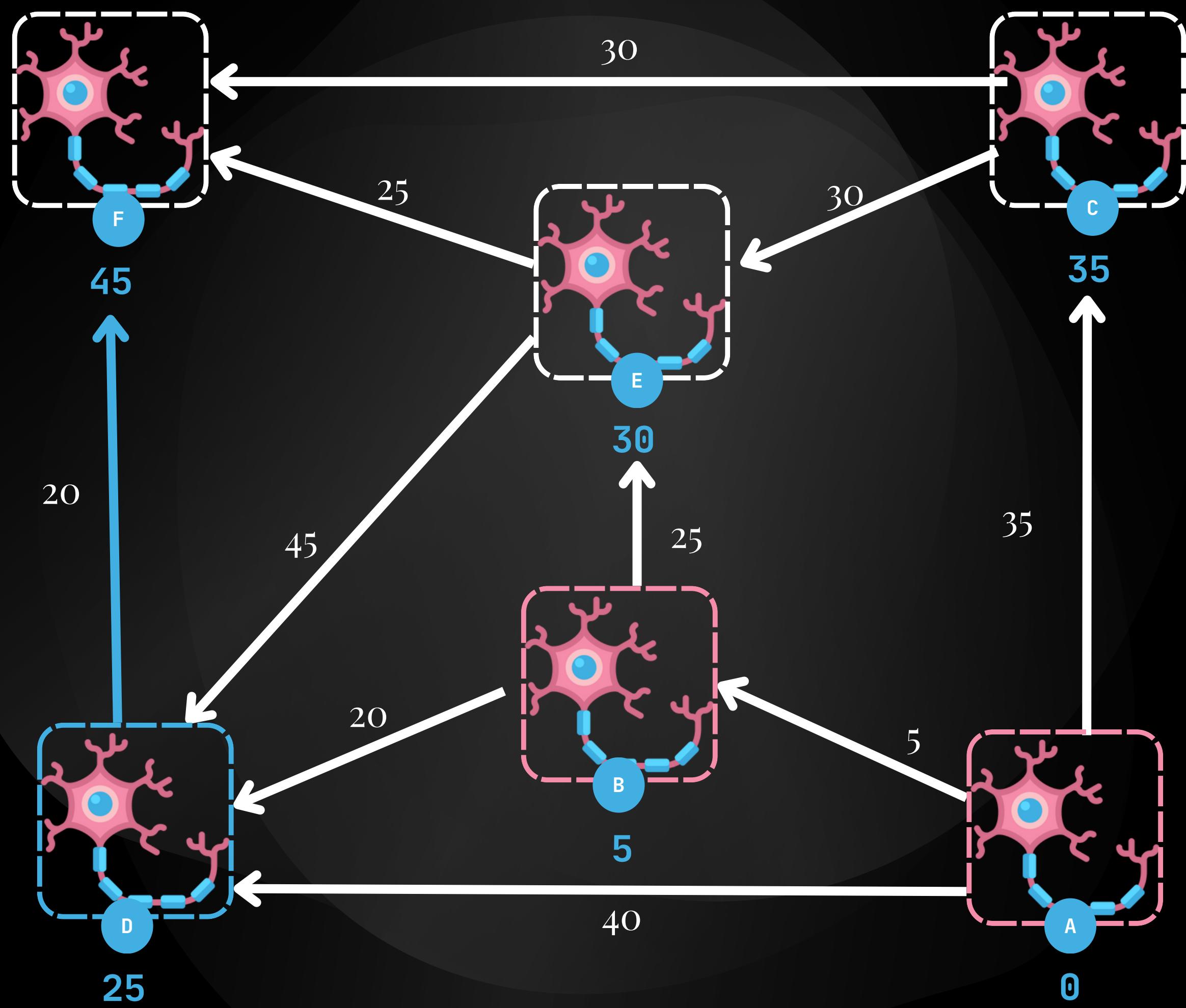
$D \rightarrow F$

Fila prioridad

$E = 30$

$C = 35$

$F = 45$



Predecesores

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow D$

$B \rightarrow E$

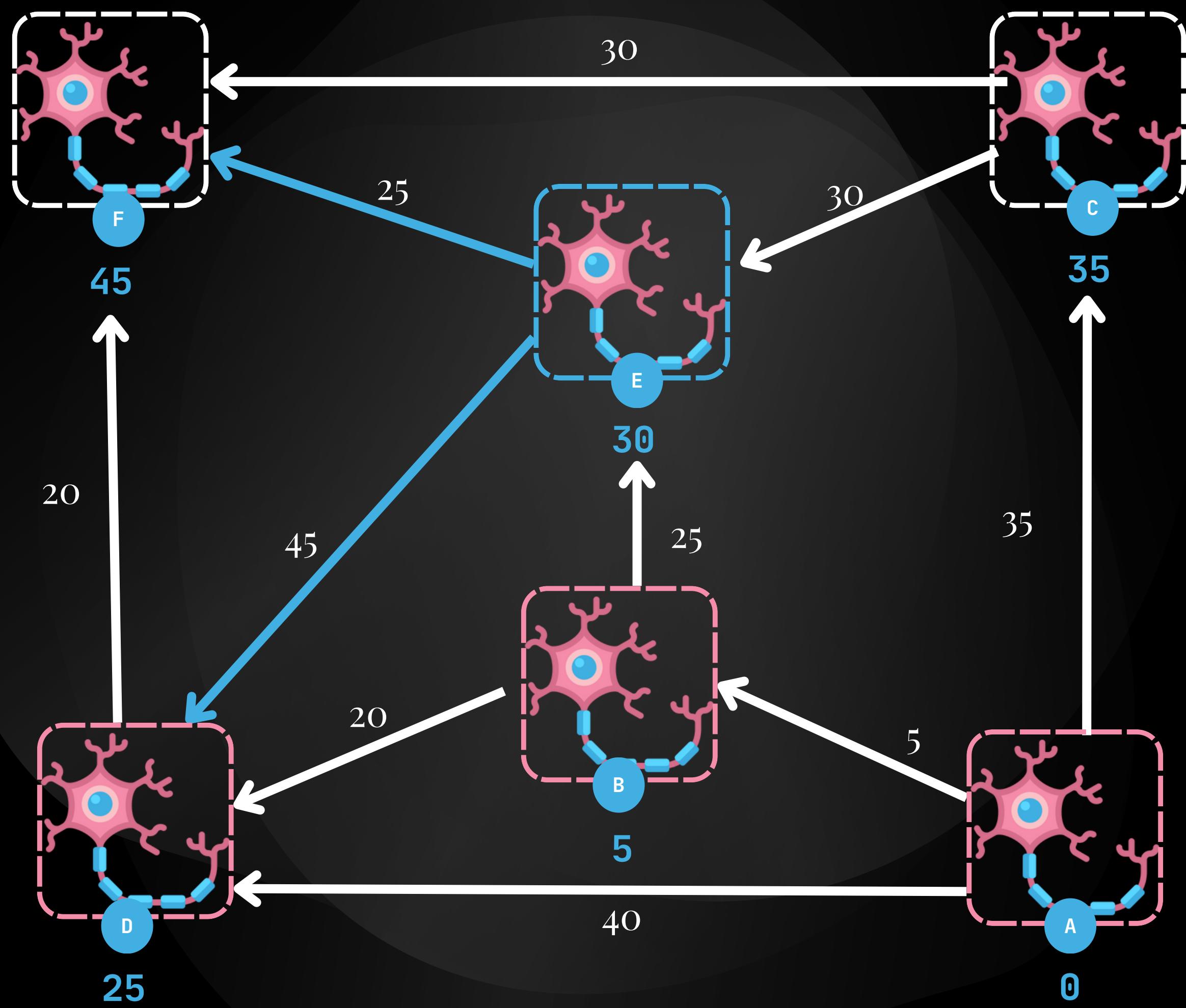
$D \rightarrow F$

Fila prioridad

$E = 30$

$C = 35$

$F = 45$



Predecesores

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow D$

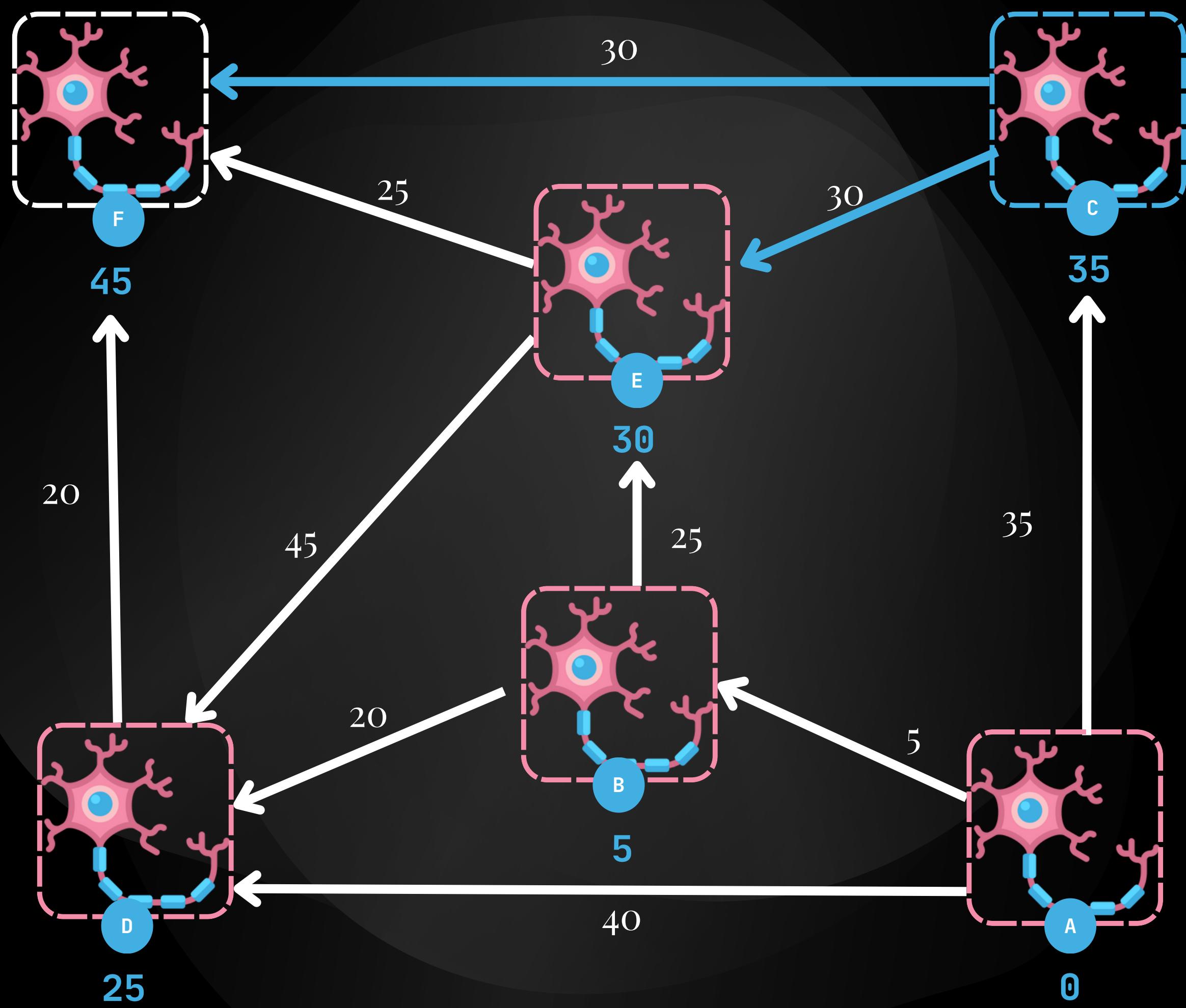
$B \rightarrow E$

$D \rightarrow F$

Fila prioridad

$C = 35$

$F = 45$



Predecesores

$A \rightarrow B$

$A \rightarrow C$

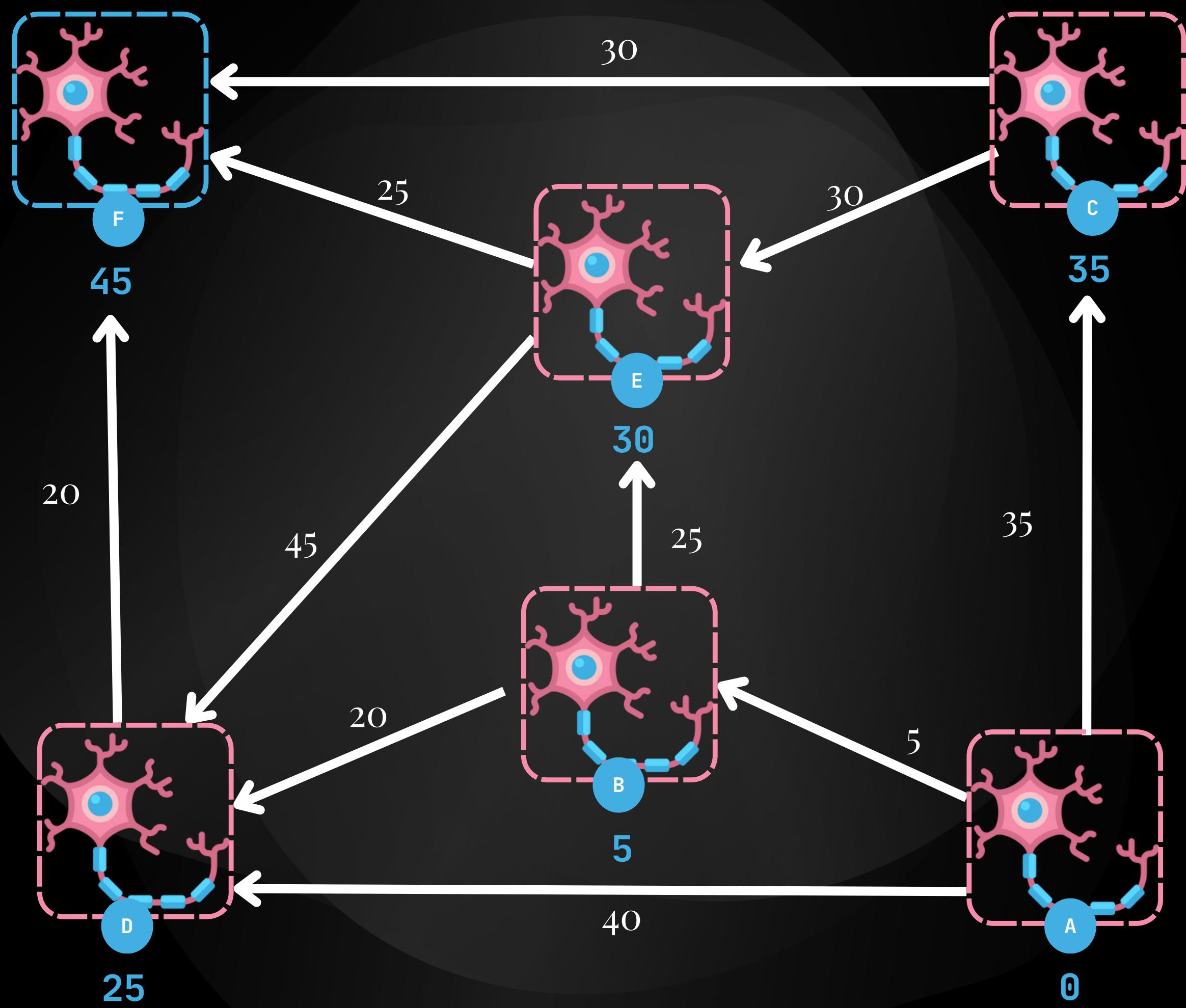
$B \rightarrow D$

$B \rightarrow E$

$D \rightarrow F$

Fila prioridad

$F = 45$



Predecesores

$A \rightarrow B$

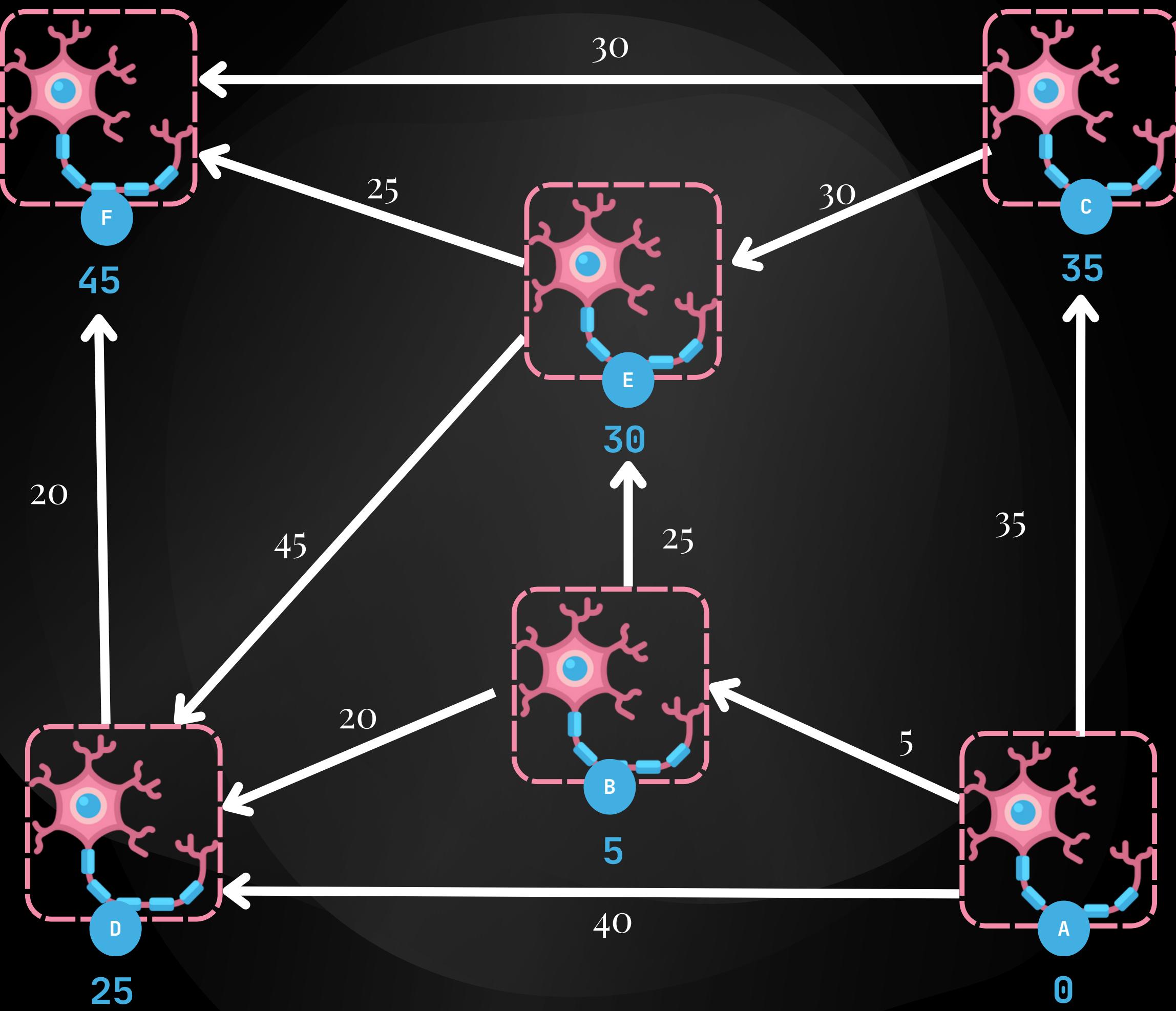
$A \rightarrow C$

$B \rightarrow D$

$B \rightarrow E$

$D \rightarrow F$

Fila prioridad



*Hallamos el camino  
más corto entre A y F*

## Predecesores

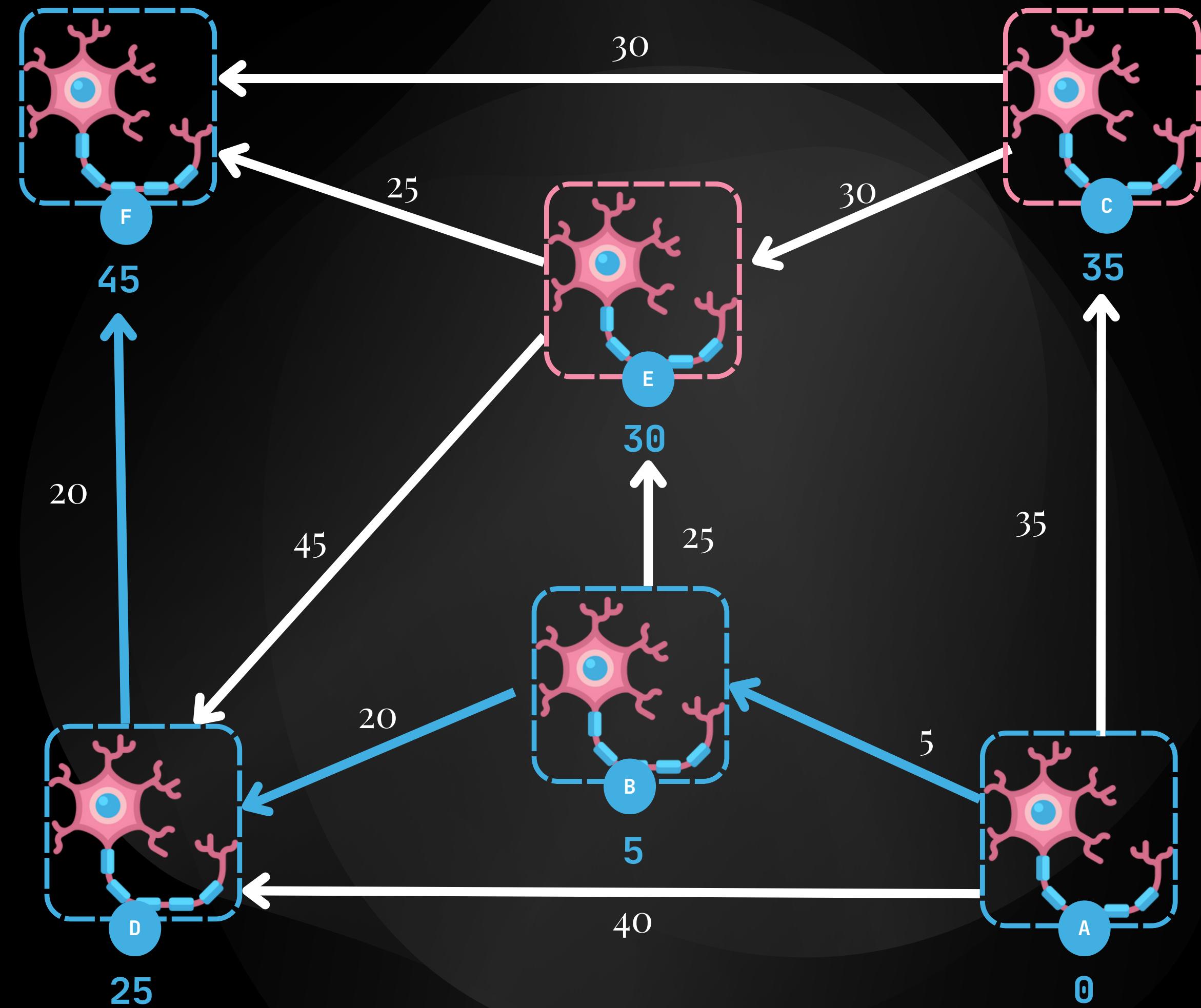
$$A \rightarrow B$$

$$A \rightarrow C$$

$$B \rightarrow D$$

$$B \rightarrow E$$

$$D \rightarrow F$$



*¡MUCHAS!*  
Gracias !



*Reto*



*Presentación*