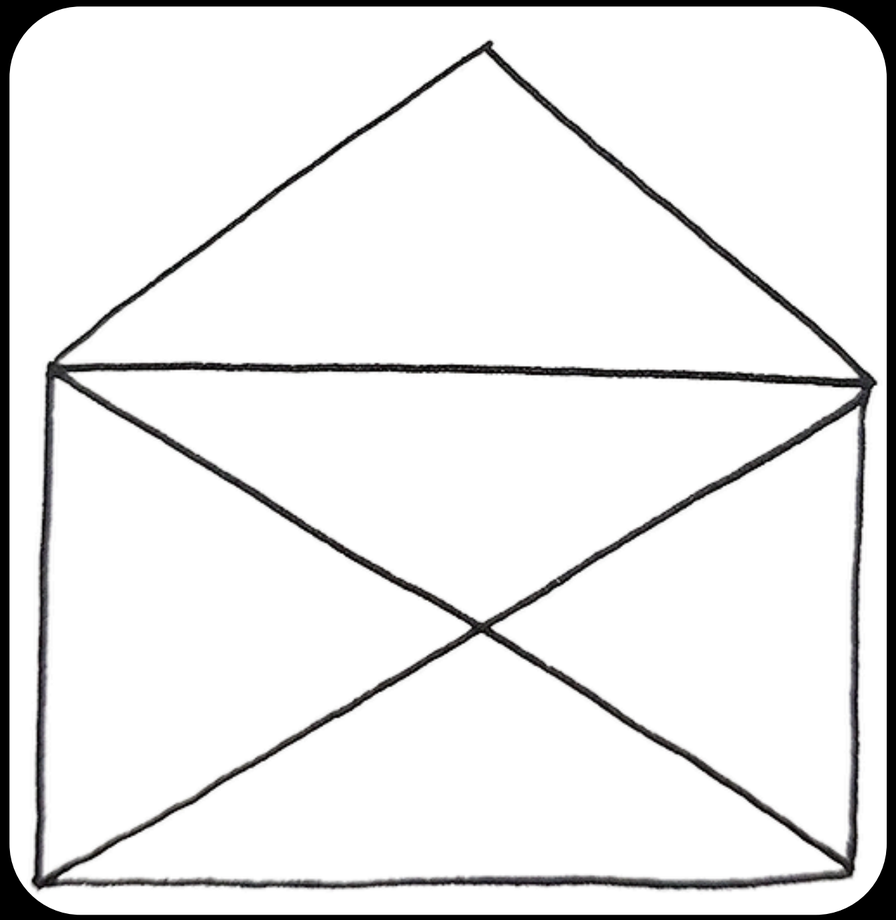
Una introducción
<Muy Algorítmica>
a la Teoría de
GRAFOS

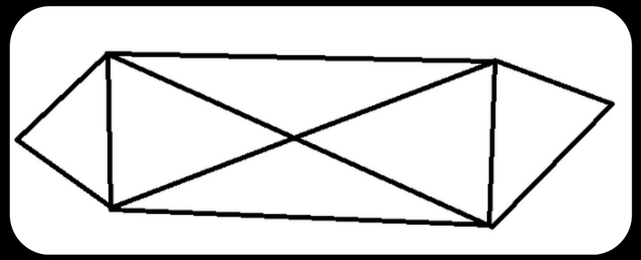
Presentado por:

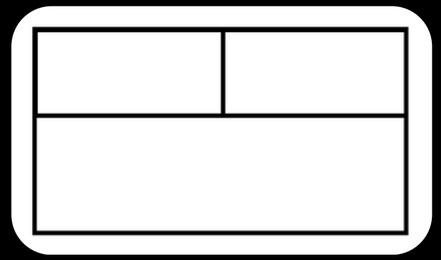
- Jerónimo Hoyos Botero.
- Daniel Fernando Verdecia Goyeneche

¿Puedes dibujar la casita de un solo trazo?



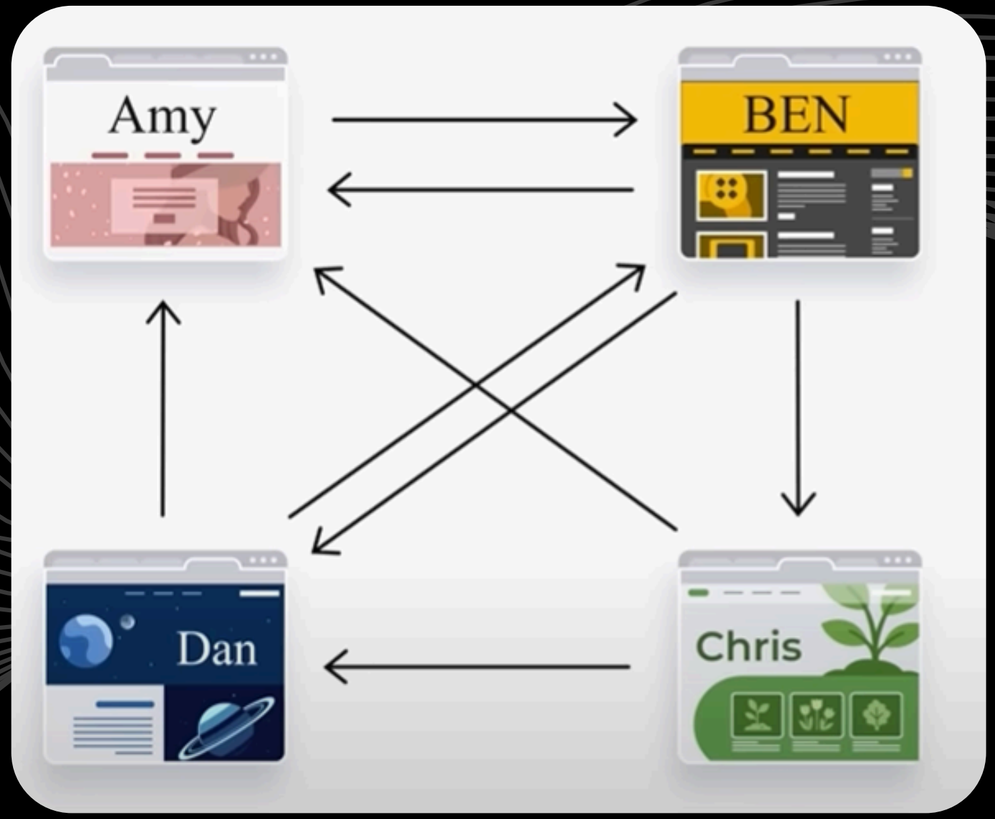
Otros ejemplos:



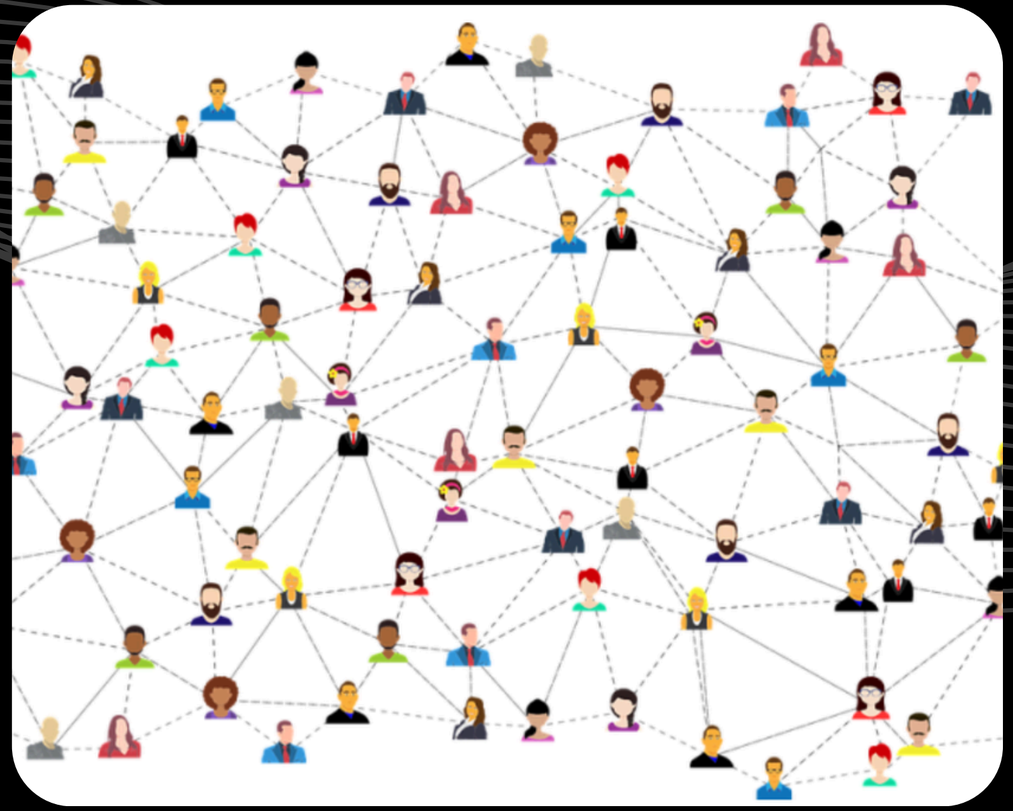


UNIVERSIDAD NACIONAL DE COLOMBIA

¡Está en todos lados!



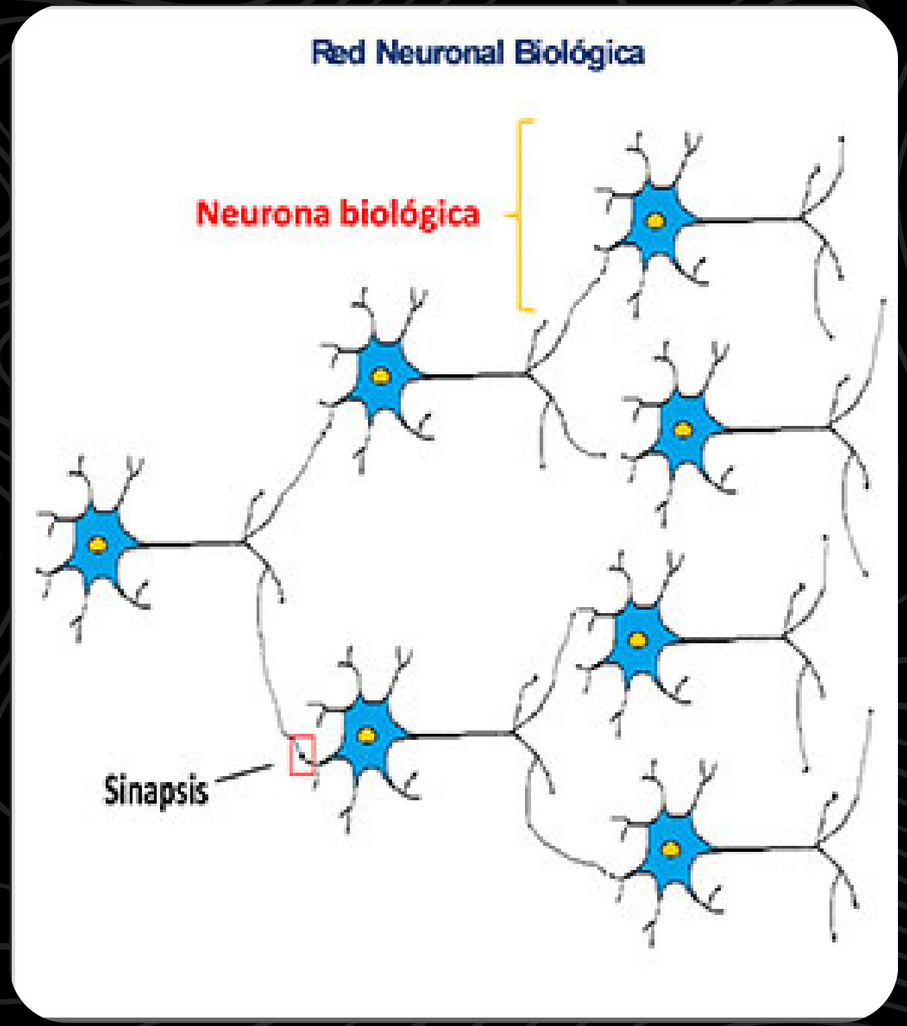
Páginas WEB

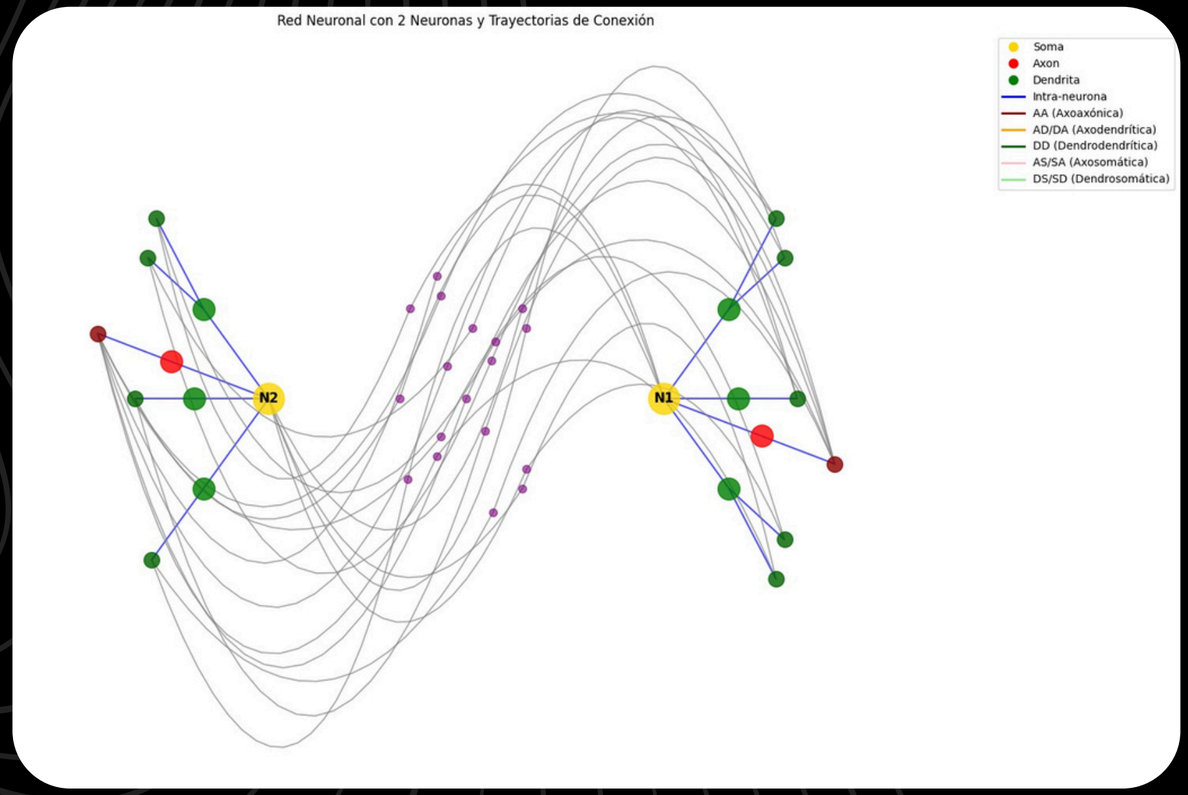


Personas

UNIVERSIDAD NACIONAL DE COLOMBIA

Analogía Fisiológica-Grafos

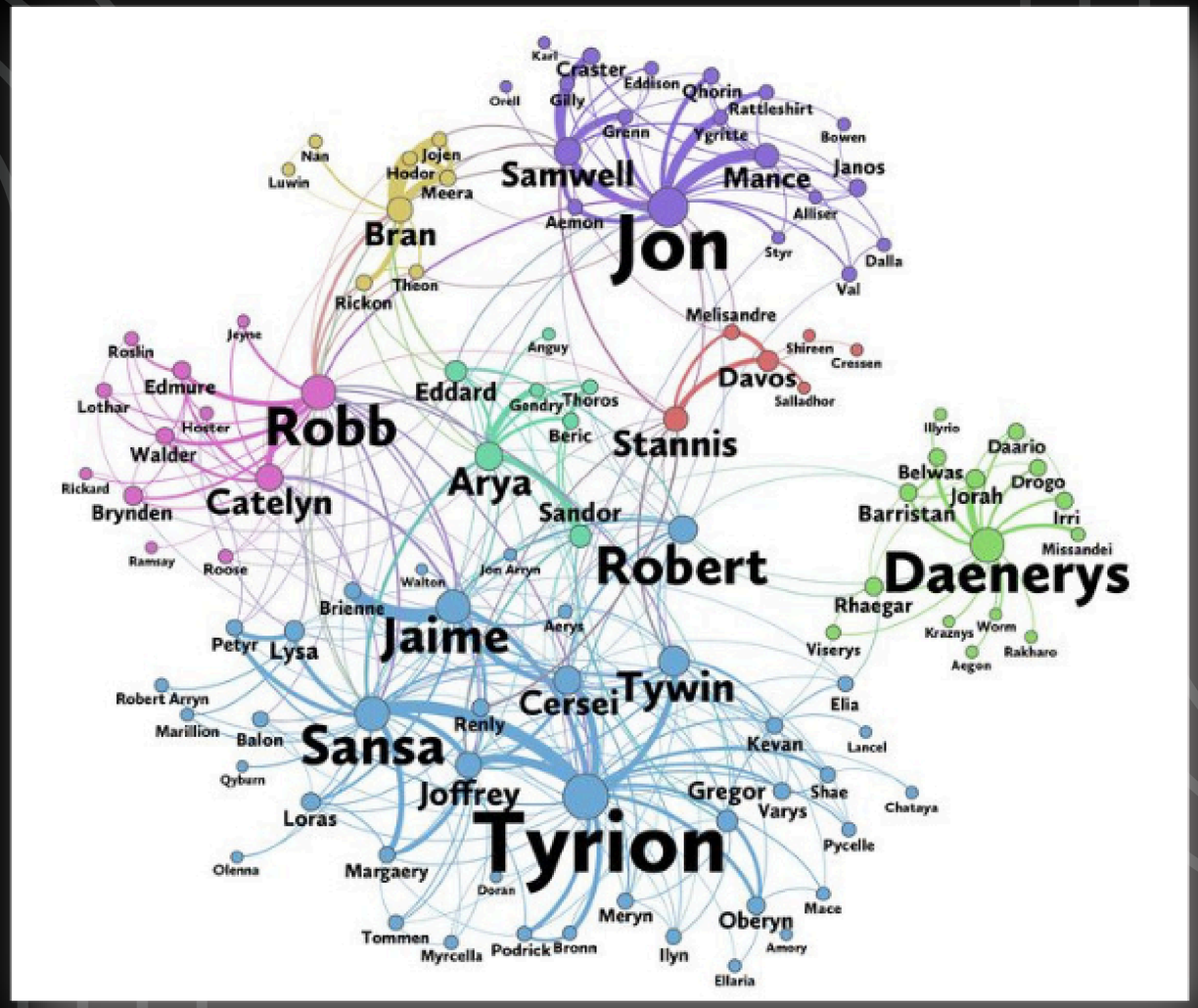




¡Investigación en proceso por semilleristas de neuroco!

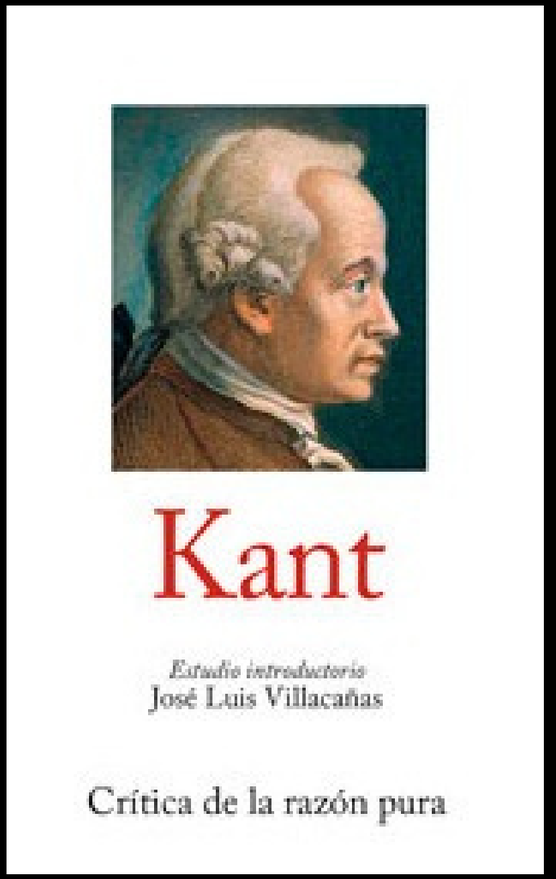
UNIVERSIDAD NACIONAL DE COLOMBIA

¿Quién es el protagonista de GOT?



UNIVERSIDAD NACIONAL DE COLOMBIA

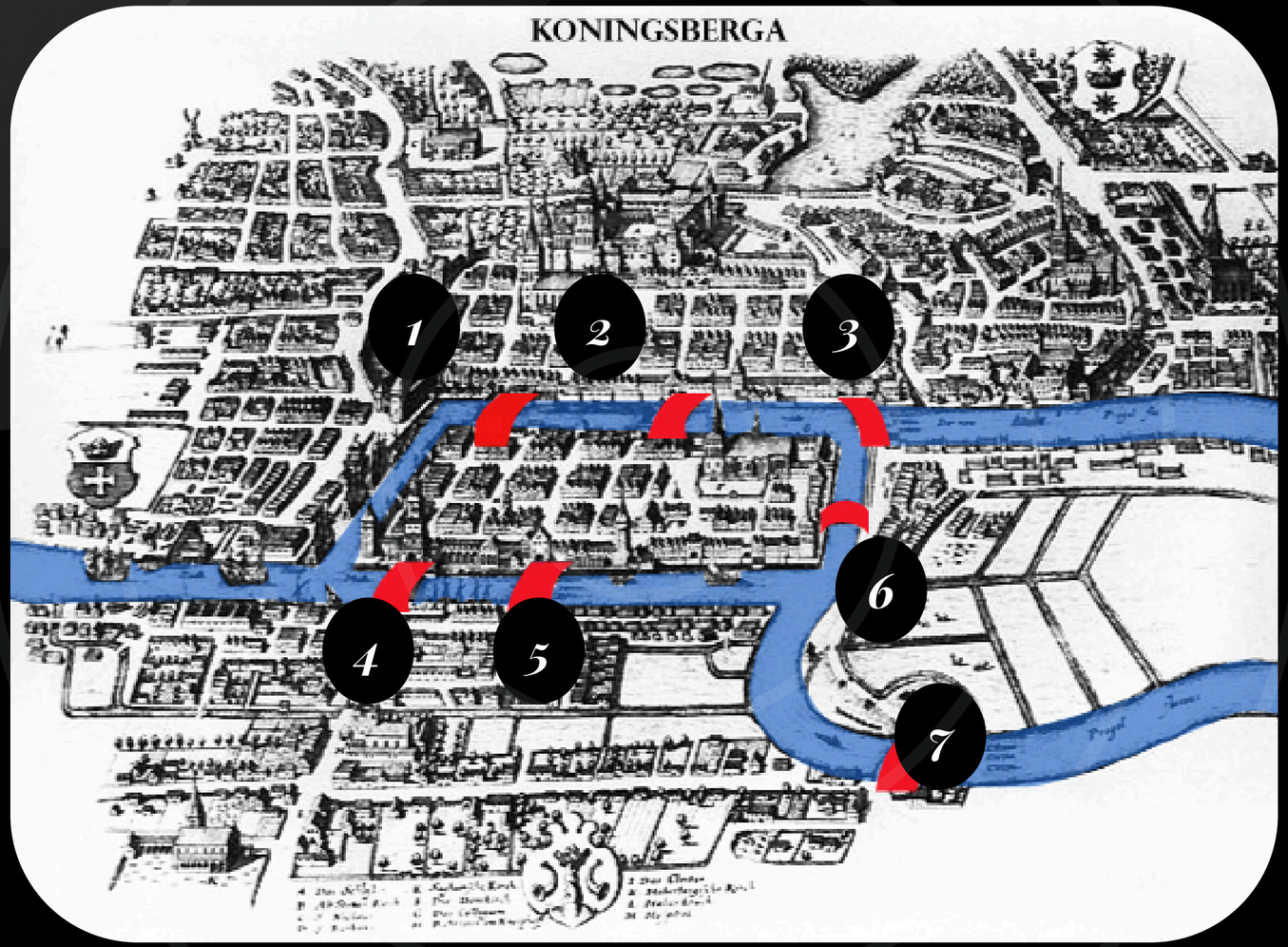
KÖNIGSBERG



Tierra de
Immanuel Kant

UNIVERSIDAD
NACIONAL
DE COLOMBIA

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



Posibilidades

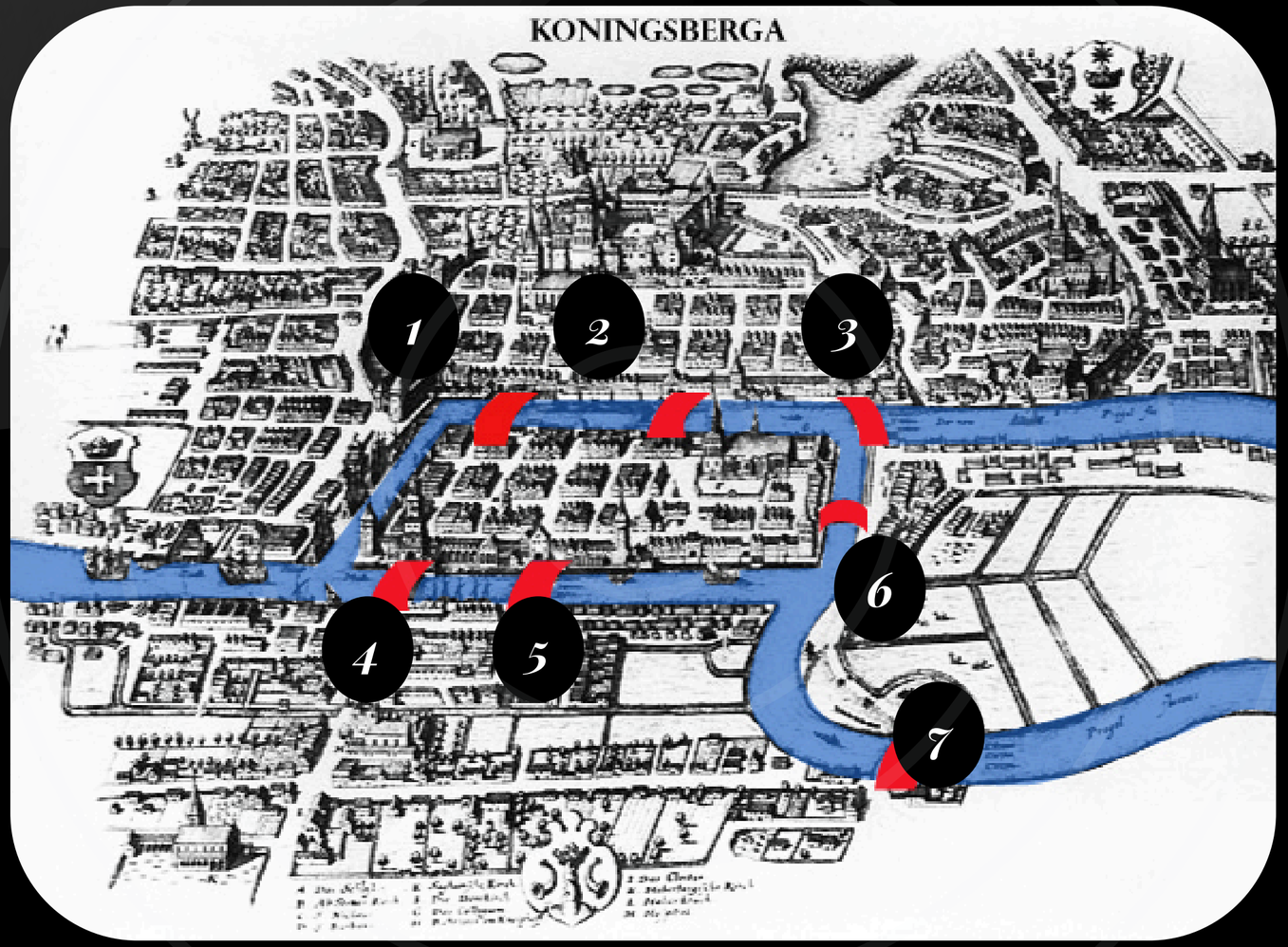
- [1,2,3,4,5,6,7]
- [6,7,2,1,5,4,3]
- [7,6,2,6,3,4,1]
- -
- -

1x2x3x4x5x6x7

7!=5040

UNIVERSIDAD NACIONAL DE COLOMBIA

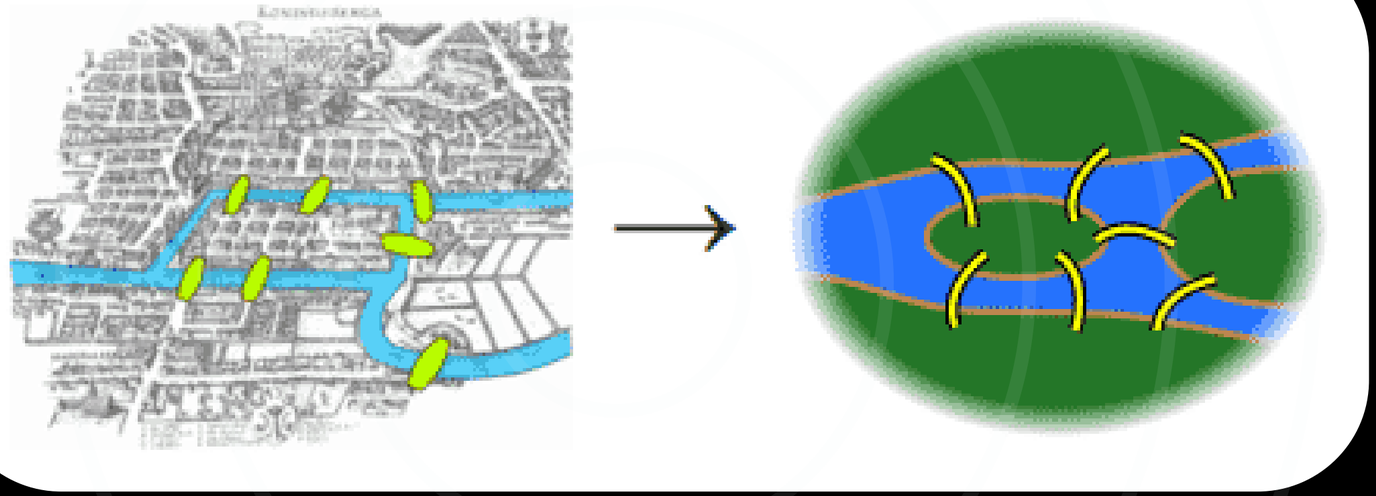
¿Se puede hacer un camino donde se pasen solo una vez cada puente?



Leonhard Euler

UNIVERSIDAD NACIONAL DE COLOMBIA

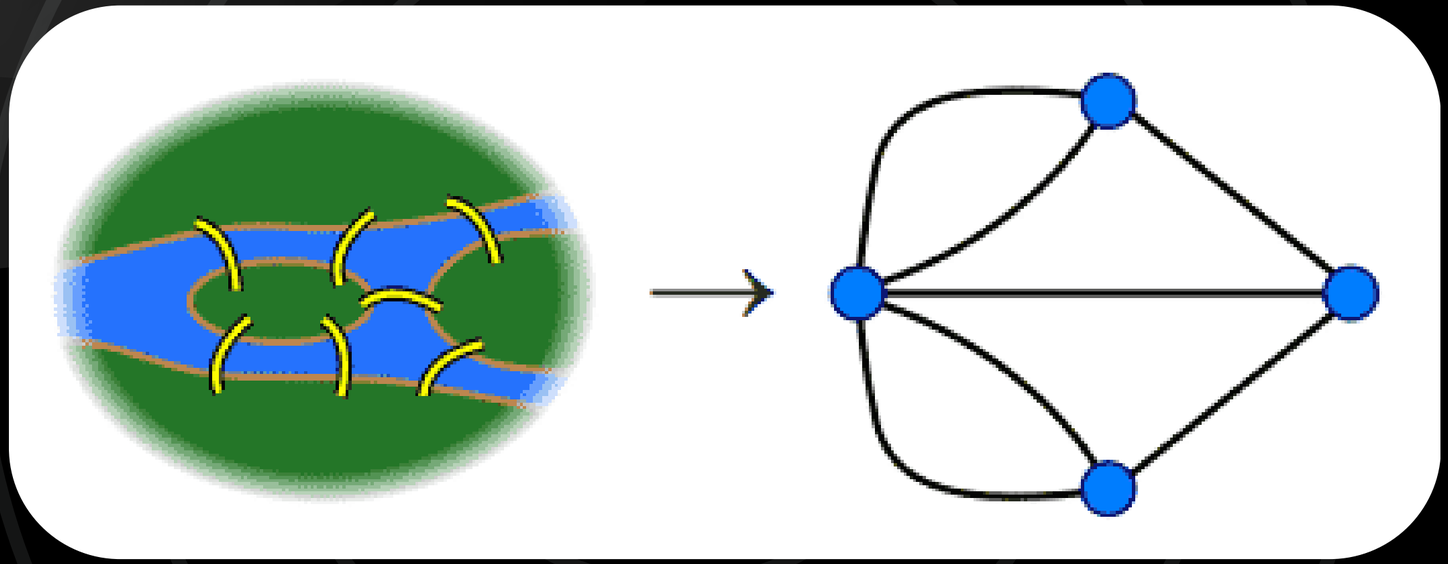
¿Se puede hacer un camino donde se pasen solo una vez cada puente?



Leonhard Euler

UNIVERSIDAD NACIONAL DE COLOMBIA

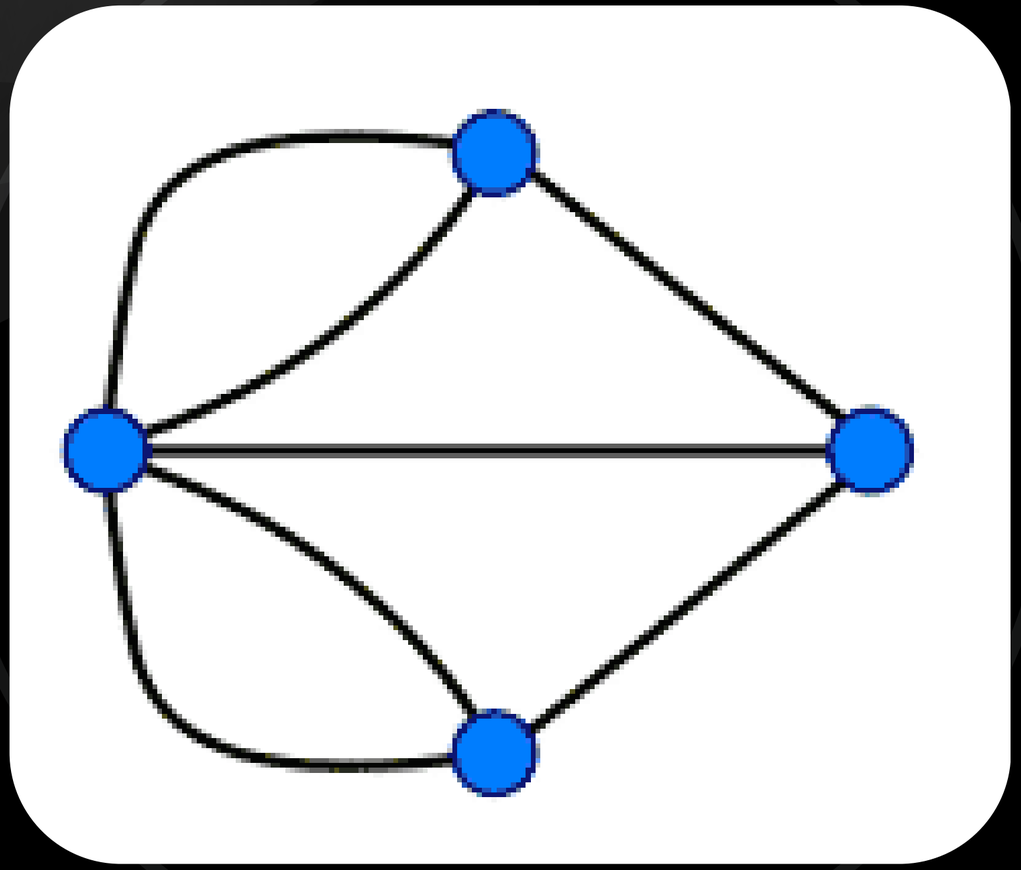
¿Se puede hacer un camino donde se pasen solo una vez cada puente?



Leonhard Euler

UNIVERSIDAD NACIONAL DE COLOMBIA

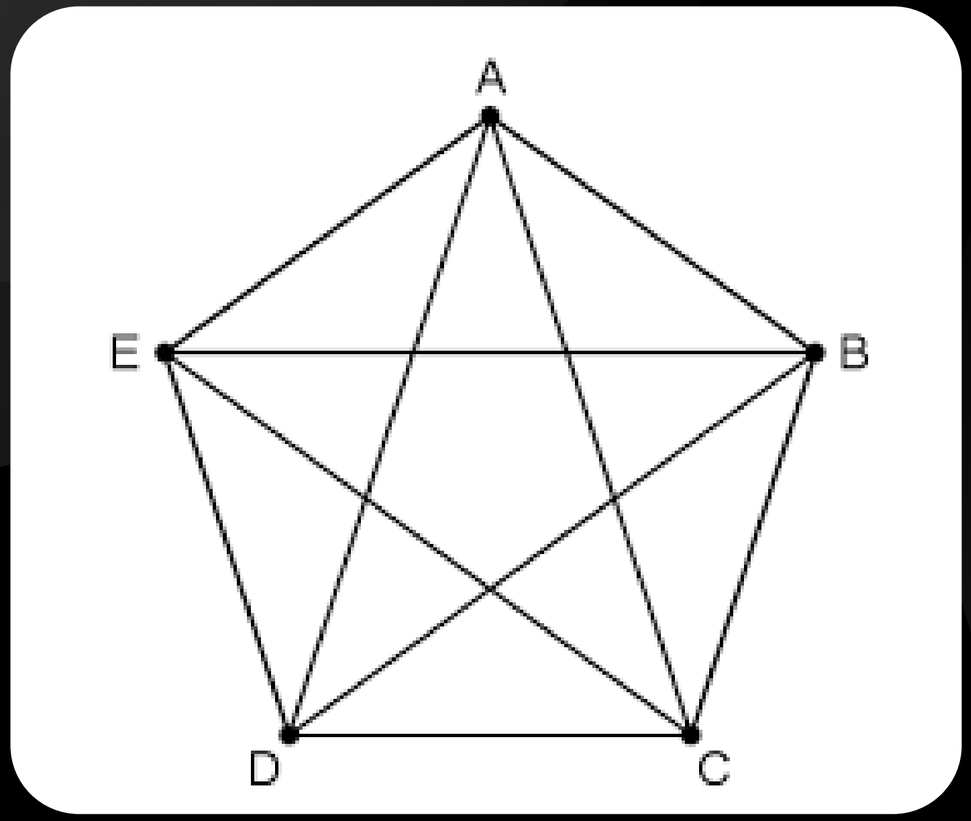
¿Se puede hacer un camino donde se pasen solo una vez cada puente?



Leonhard Euler

UNIVERSIDAD NACIONAL DE COLOMBIA

El Teorema de Euler



Es un Grafo Euleriano

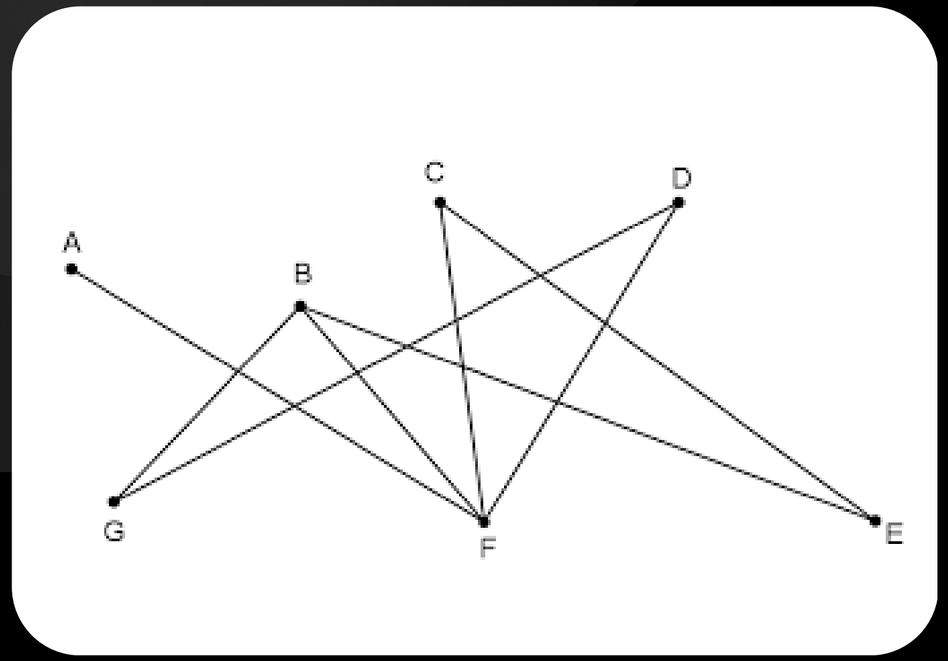
Un grafo es Euleriano

Es **conexo** y todos sus **vértices** son **pares.**

UNIVERSIDAD NACIONAL DE COLOMBIA

El Teorema de Euler

Circuito de Euler NO es lo mismo que Camino de Euler



Un grafo es Euleriano

Es **conexo** y todos sus **vértices** son **pares.**

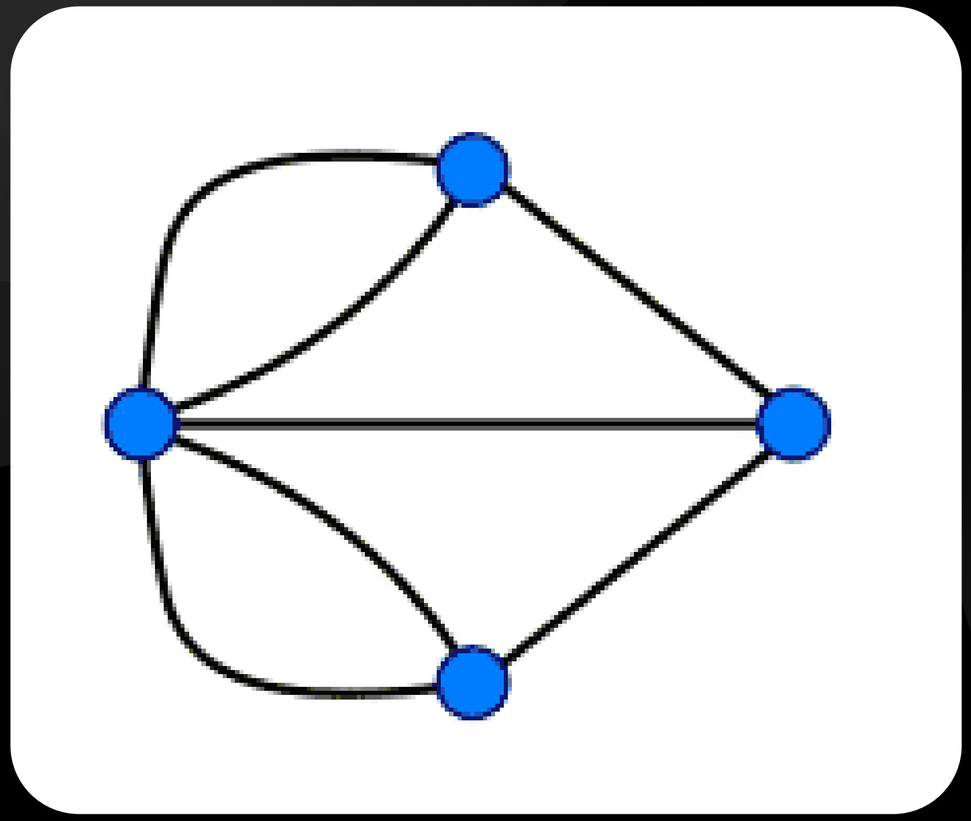
No es un grafo de Euler

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

Sí tiene camino Euleriano.

UNIVERSIDAD NACIONAL DE COLOMBIA

El Teorema de Euler



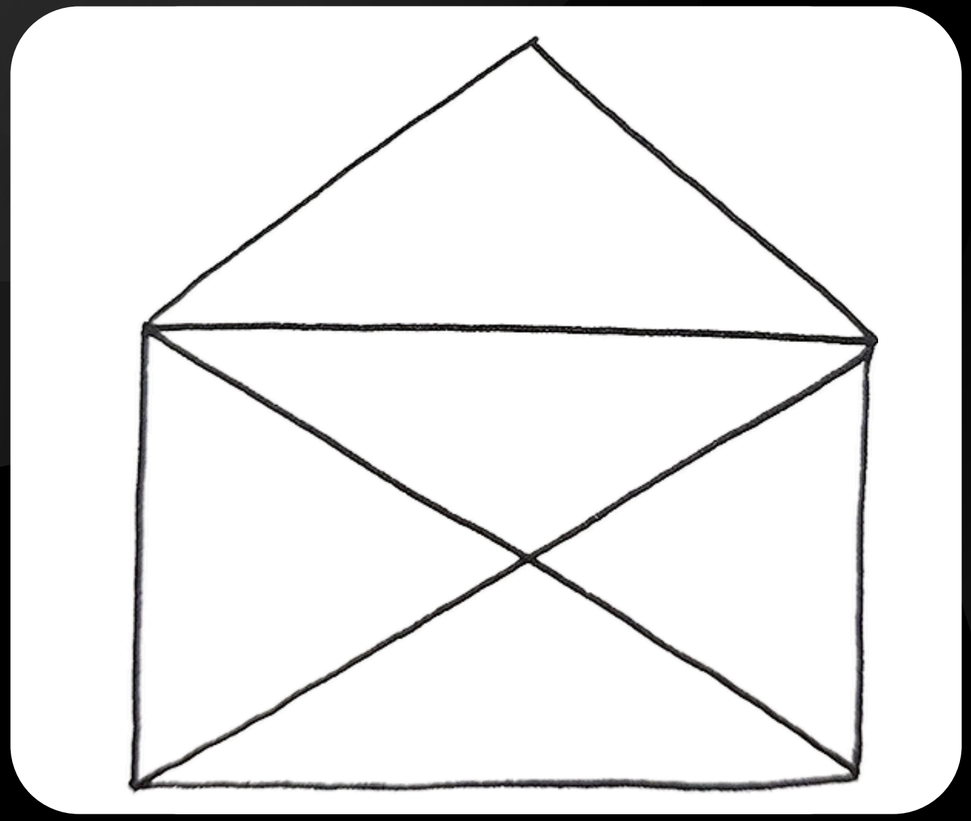
No es un Grafo Euleriano

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.**

UNIVERSIDAD NACIONAL DE COLOMBIA

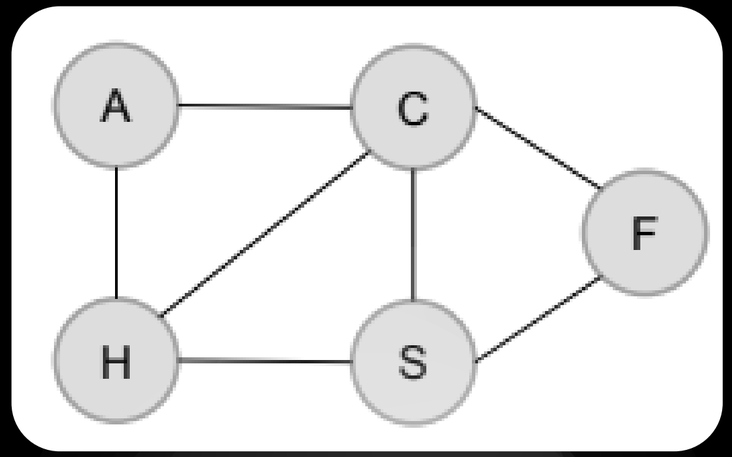
¿CÓMO LOS PUEDO USAR?

UNIVERSIDAD NACIONAL DE COLOMBIA

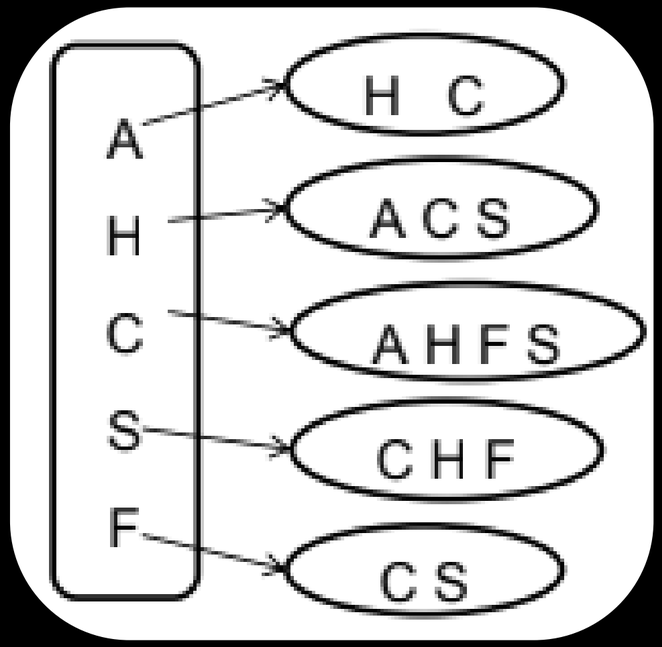
IMPLEMENTACIÓN

UNIVERSIDAD
NACIONAL
DE COLOMBIA

UNIVERSIDAD NACIONAL DE COLOMBIA



Diccionarios

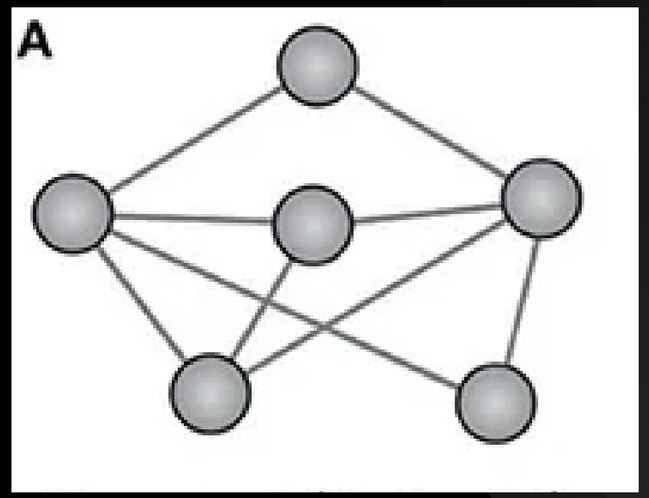


Matriz Adyacencia

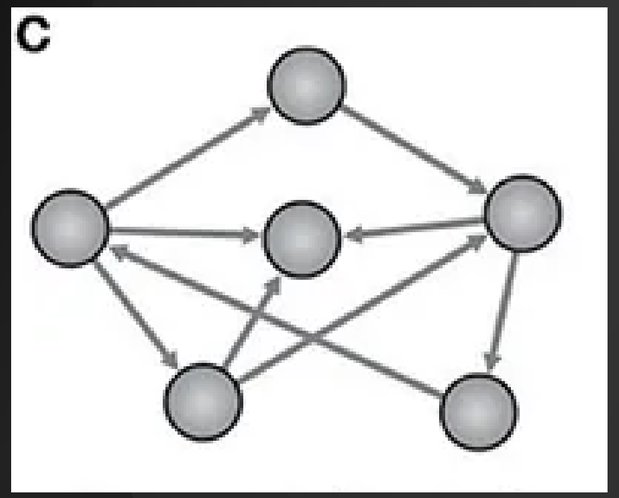
	A	C	F	S	H
A	-	1	0	0	1
C	1	-	1	1	1
F	0	1	-	1	0
S	0	1	1	-	1
H	1	1	0	1	-

UNIVERSIDAD NACIONAL DE COLOMBIA

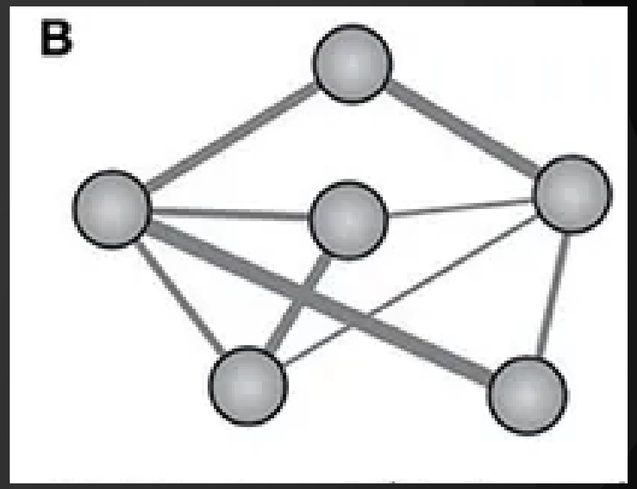
Algunos tipos de Grafos:



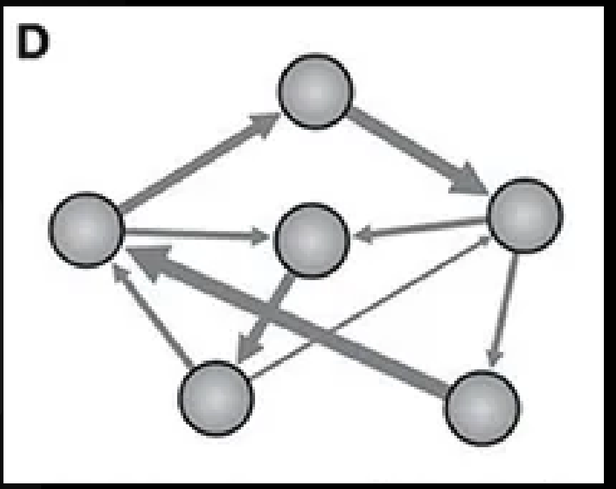
Grafo binario



Grafo dirigido



Grafo con pesos



Grafo con pesos y dirigido

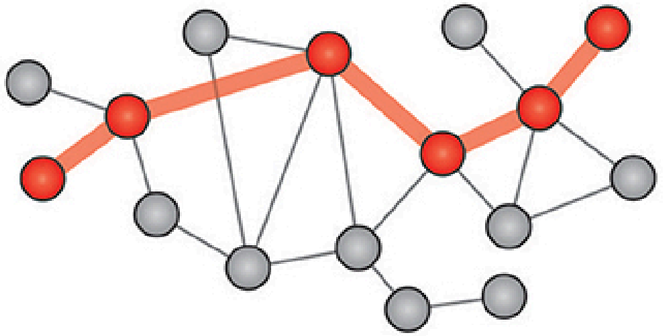
MÉTRICAS

UNIVERSIDAD
NACIONAL
DE COLOMBIA

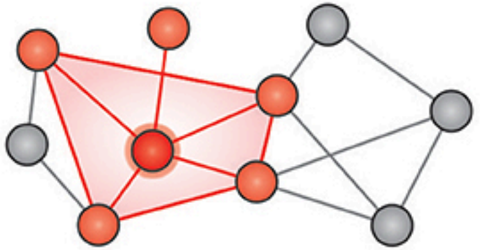
Métricas Globales

UNIVERSIDAD NACIONAL DE COLOMBIA

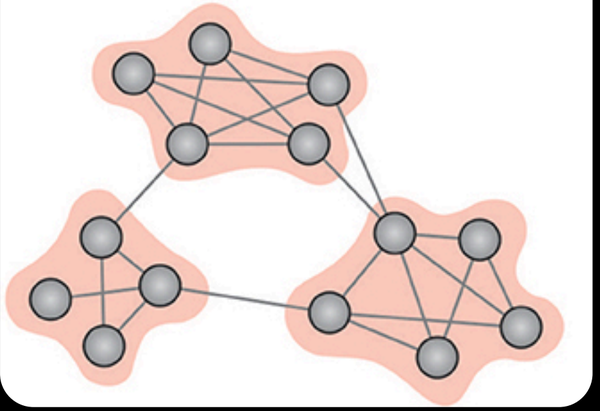
Shortest path



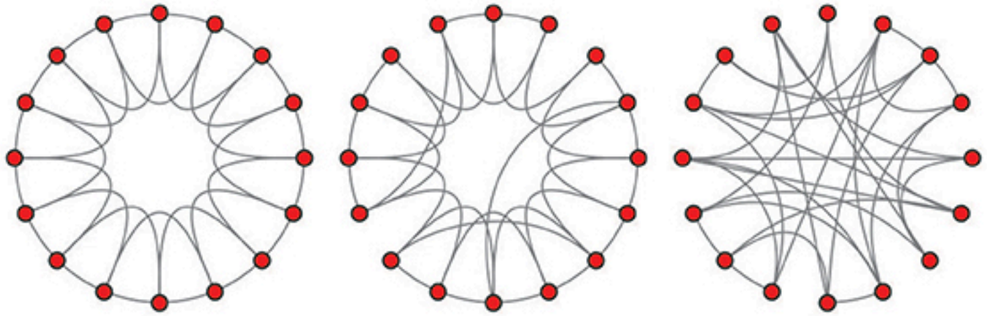
Clustering coefficient



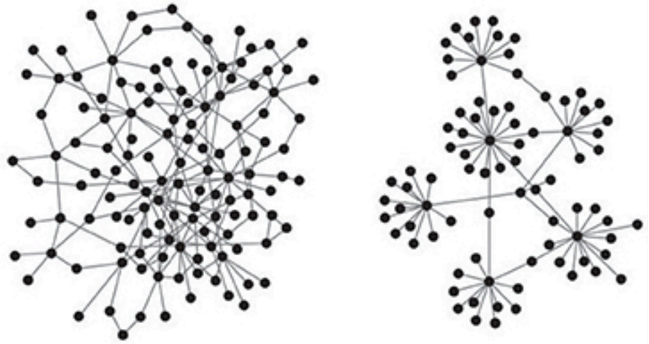
Modularity



Regular Small-World Random



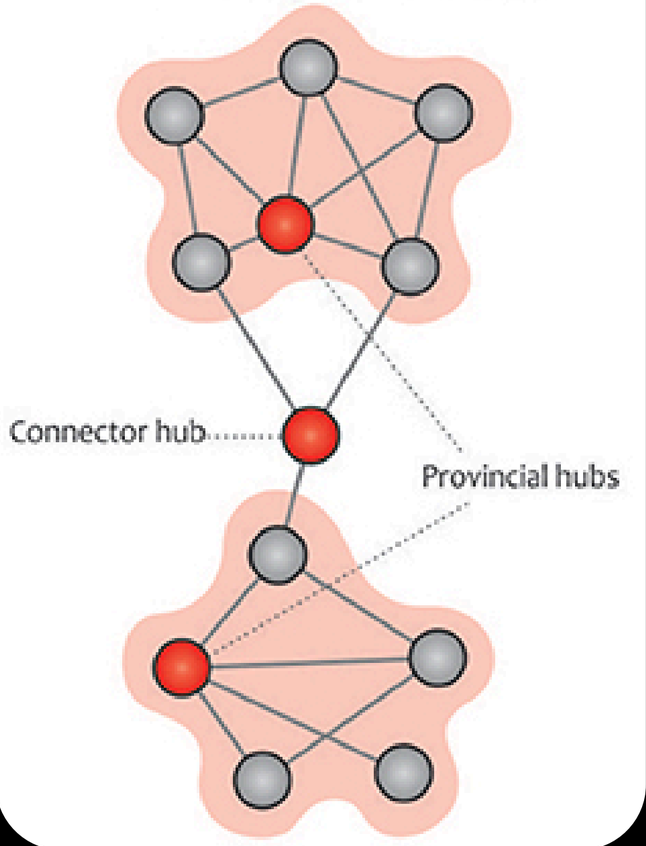
Assortative Disassortative



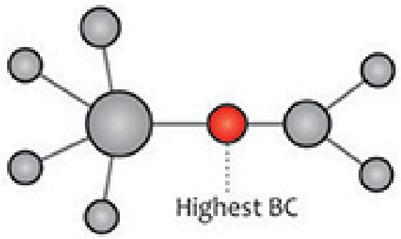
UNIVERSIDAD NACIONAL DE COLOMBIA

Métricas Locales

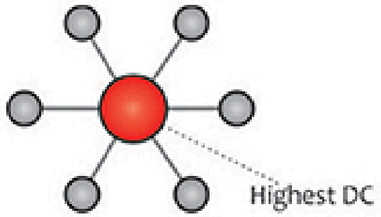
Centrality and hubs



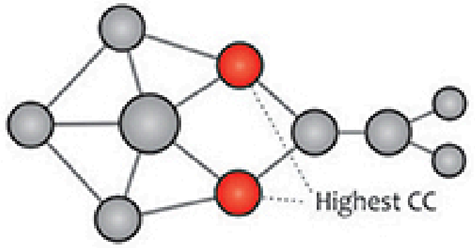
Betweenness centrality



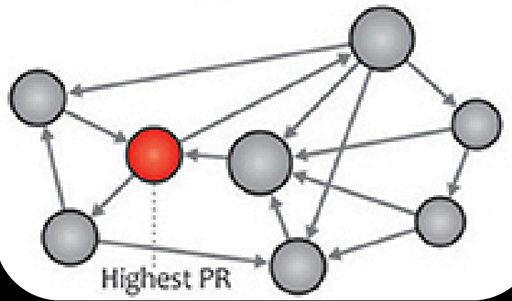
Degree centrality



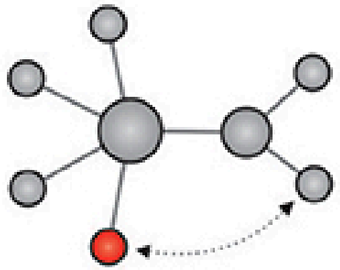
Closeness centrality



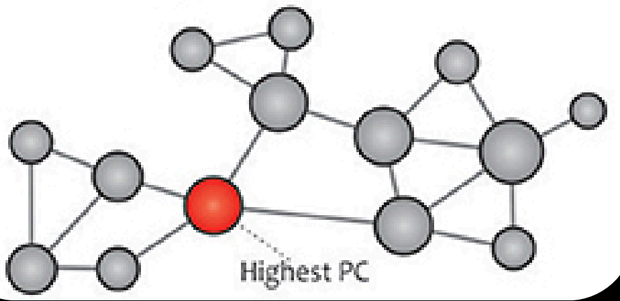
PageRank



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

UNIVERSIDAD NACIONAL DE COLOMBIA

<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>

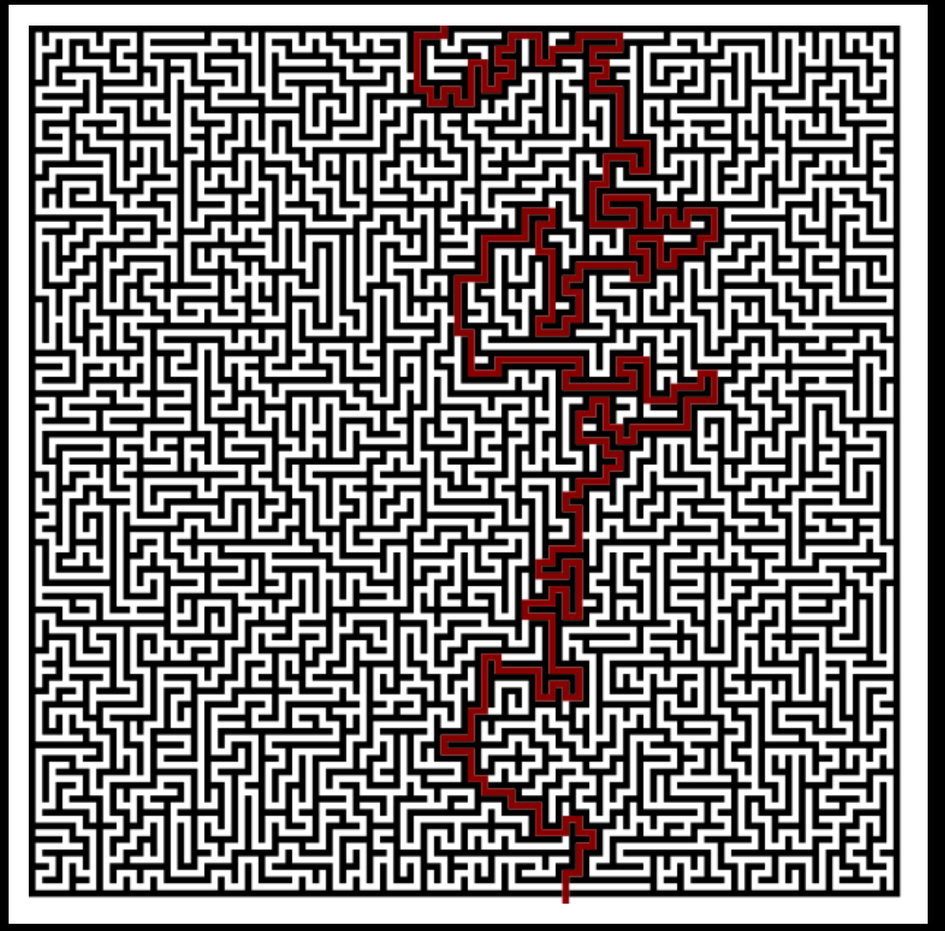
```
DFS(grafo G, nodo a):
   for each nodo u:
      u.visited = false
   s = new stack
   s.push(a)
   while (s is not empty):
      u = s.pop()
      if u.visited = false:
         u.visited = true
         for each arista (u, v):
            if v.visited = false:
               s.push(v)
```

Cantidad Mínima de saltos

UNIVERSIDAD NACIONAL DE COLOMBIA

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)
```

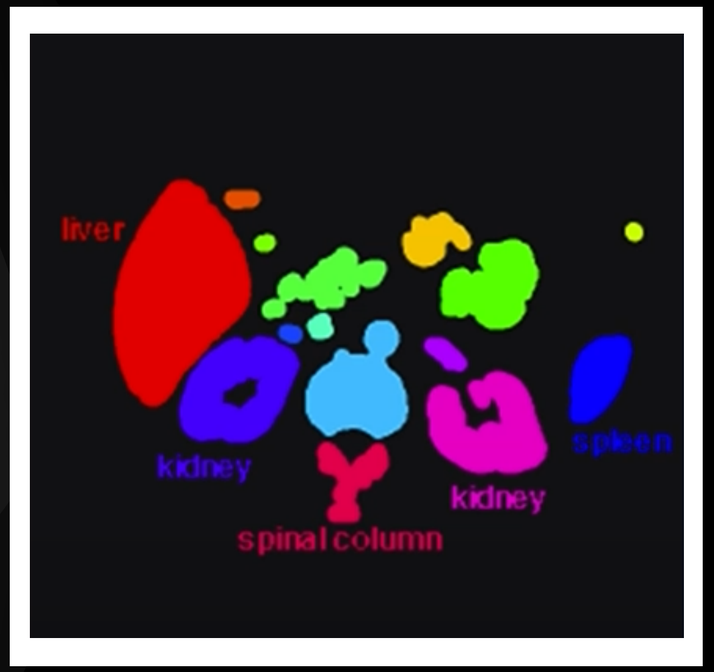


Resolver un laberinto

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 imagenes 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 ∈ V do
3:          L(x) = ∞
4:      L(a) = 0
5:      T = conjunto de todos los vértices
6:      while z ∈ T do
7:          Tomar v ∈ T con mínimo L(v).
8:          T = T − {v}
9:          for x ∈ T adyacente con v do
10:             L(x) = mín{L(x), L(v) + w(v, x)}
11:     return L(z)
```

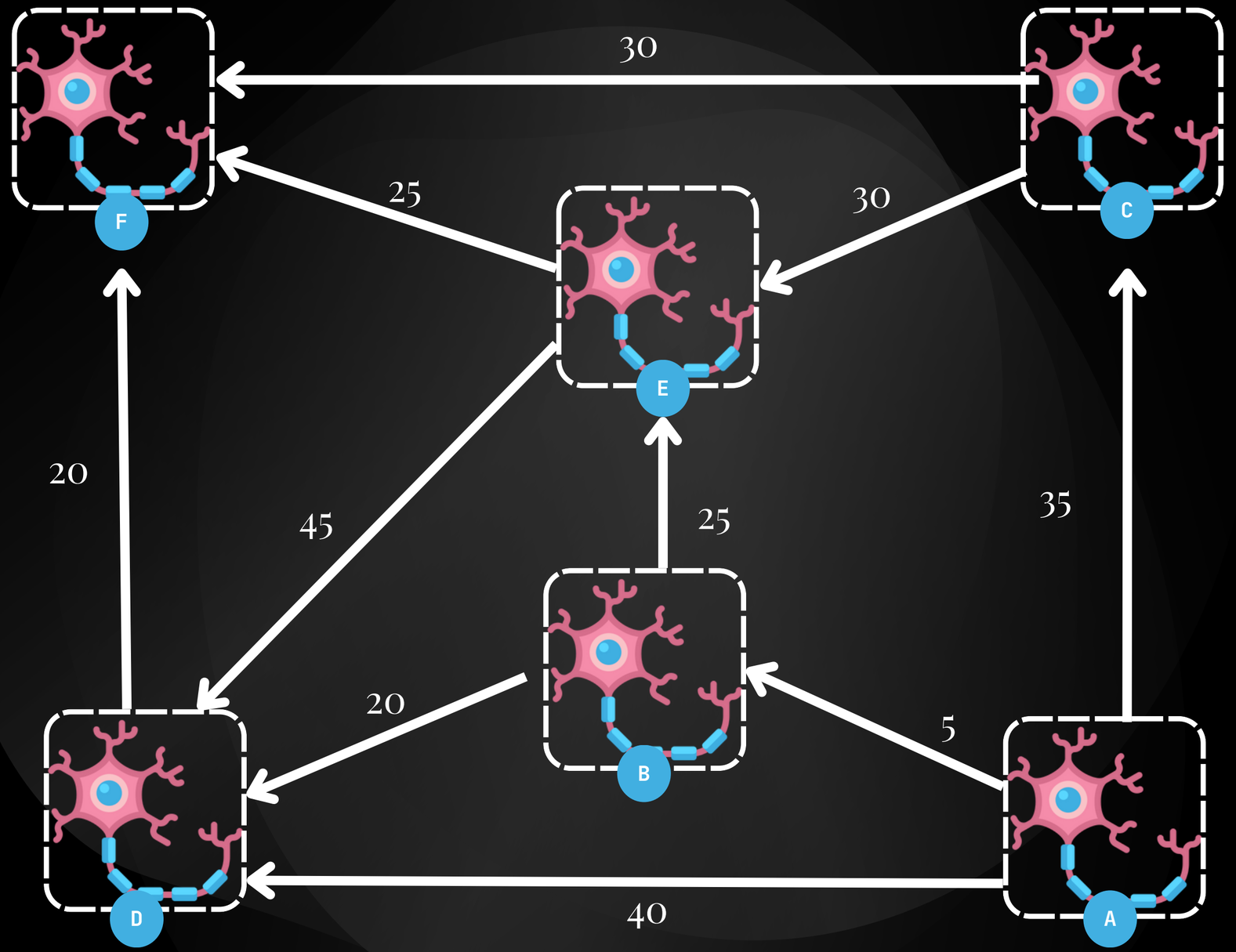
ALGORITMO DE DIJKSTRA

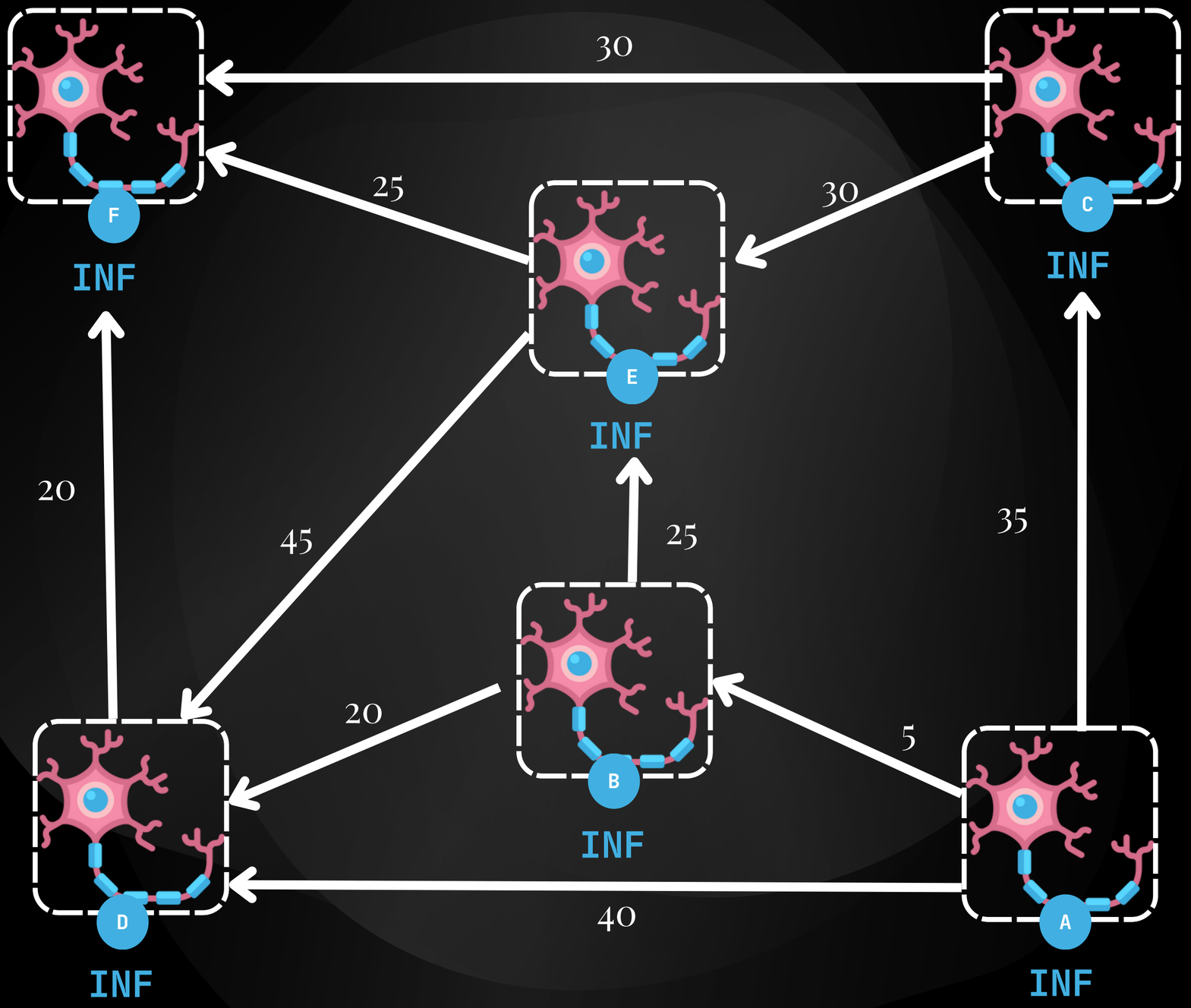
El algoritmo de Dijkstra es **la base de todos los algoritmos** de caminos más cortos. Representa una introducción fundamental para entender **cómo funcionan** los algoritmos en grafos, por eso lo estudiaremos hoy.

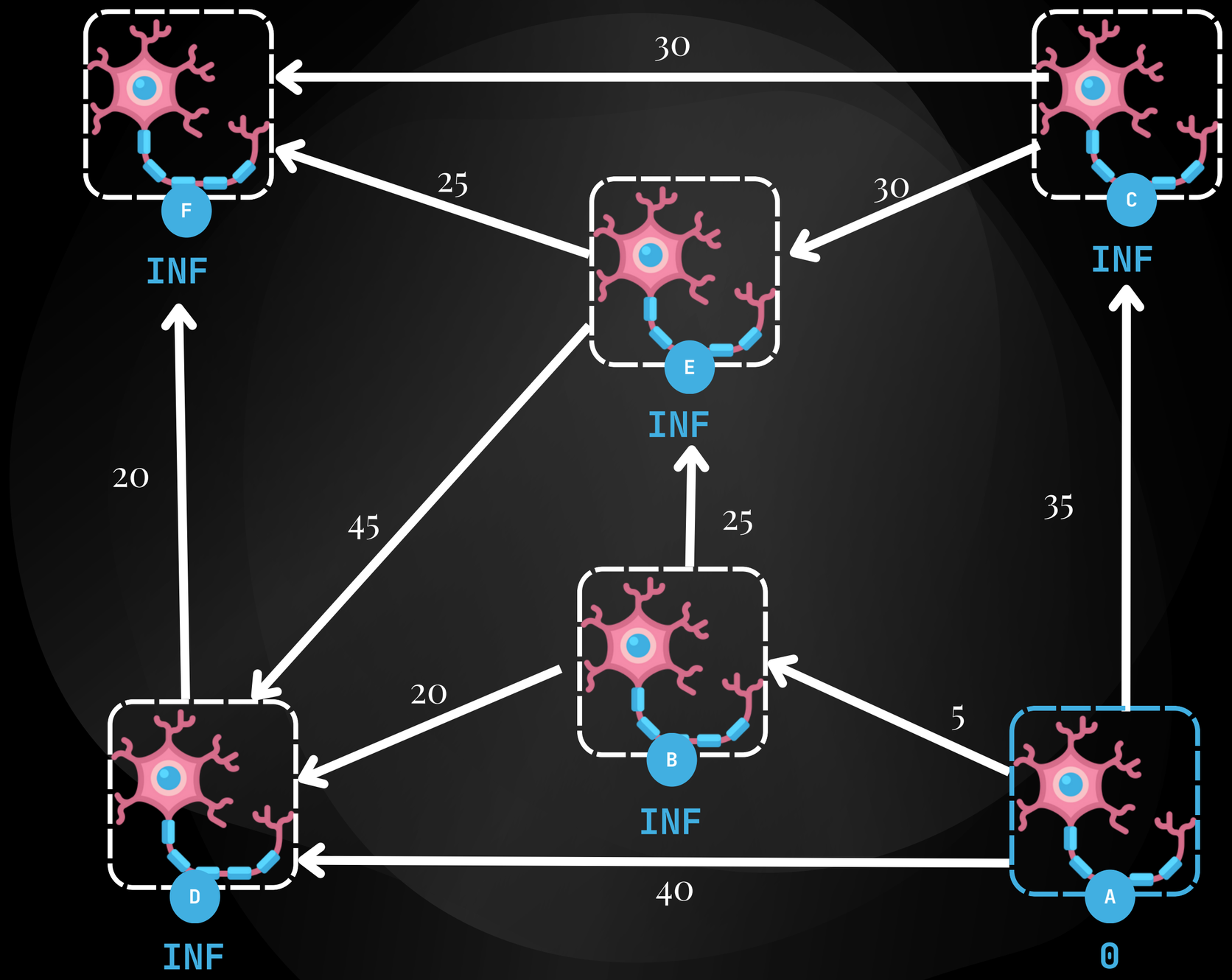
UNIVERSIDAD NACIONAL DE COLOMBIA

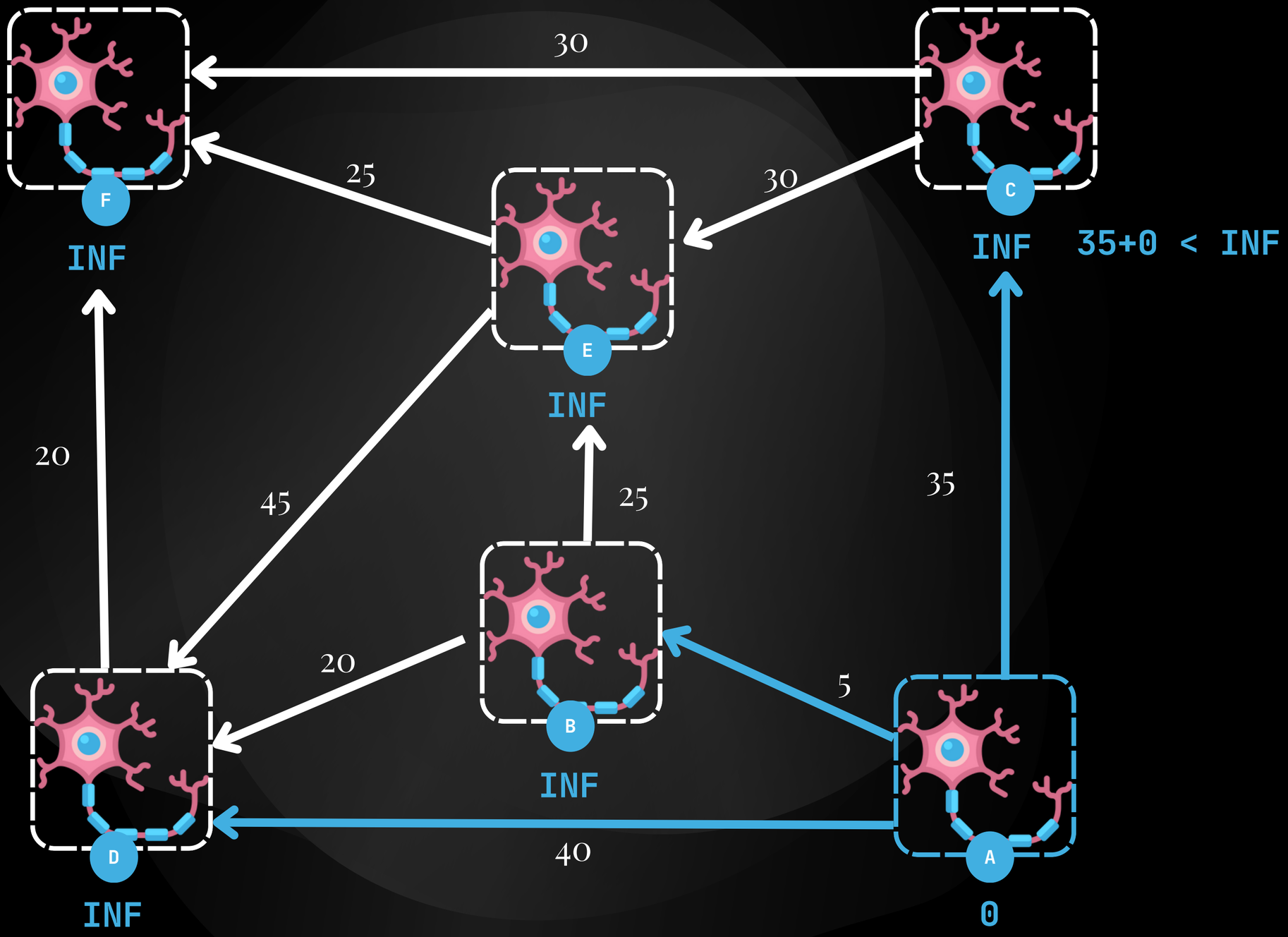
¿Cuál es el camino más corto entre A y F?

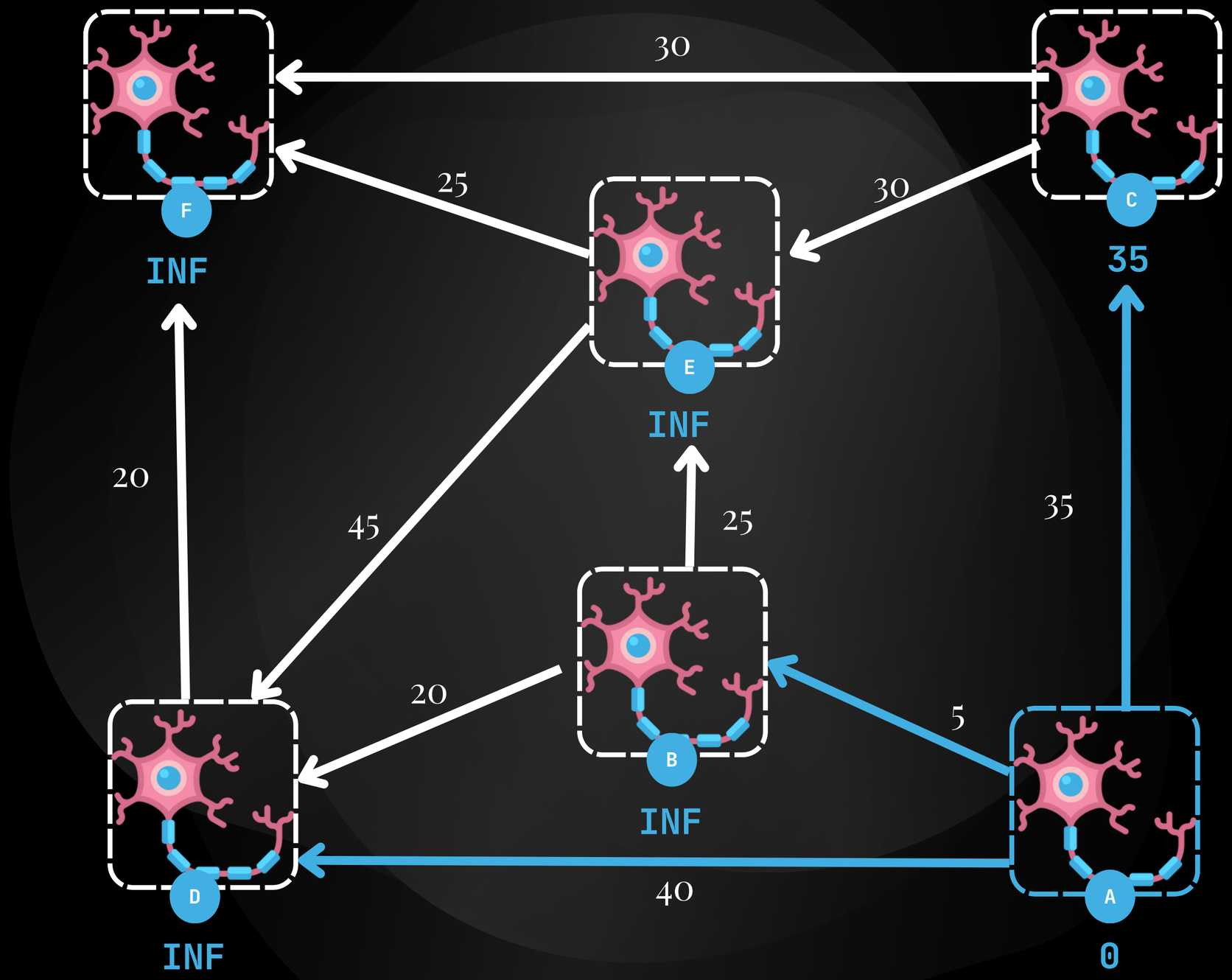
Edsger Dijkstra




30
25
30
20
45
25
35
20
5
40
F
INF
C
INF
E
INF
B
INF
D
INF
A
INF


30
25
30
20
45
25
35
20
5
40
F
INF
C
INF
E
INF
B
INF
D
INF
A
0


30
25
30
F
C
INF
INF
35+0 < INF
E
INF
20
45
25
35
20
5
B
INF
D
A
40
INF
0

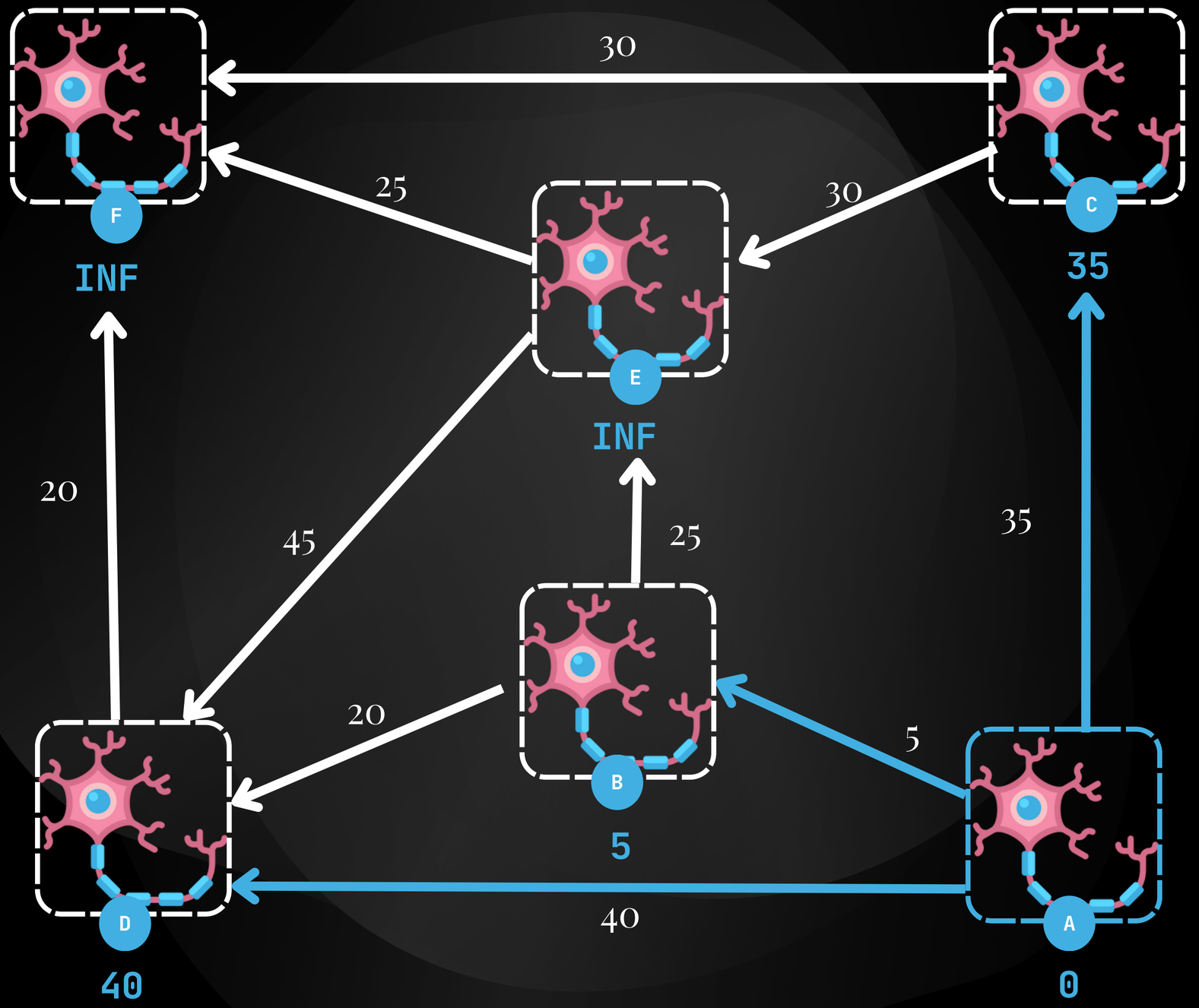

30
25
30
F
INF
C
35
E
INF
20
45
25
35
20
5
B
INF
D
INF
40
A
0

Predecesores

A → B

A → C

A → D



Fila prioridad

B = 5

C = 35

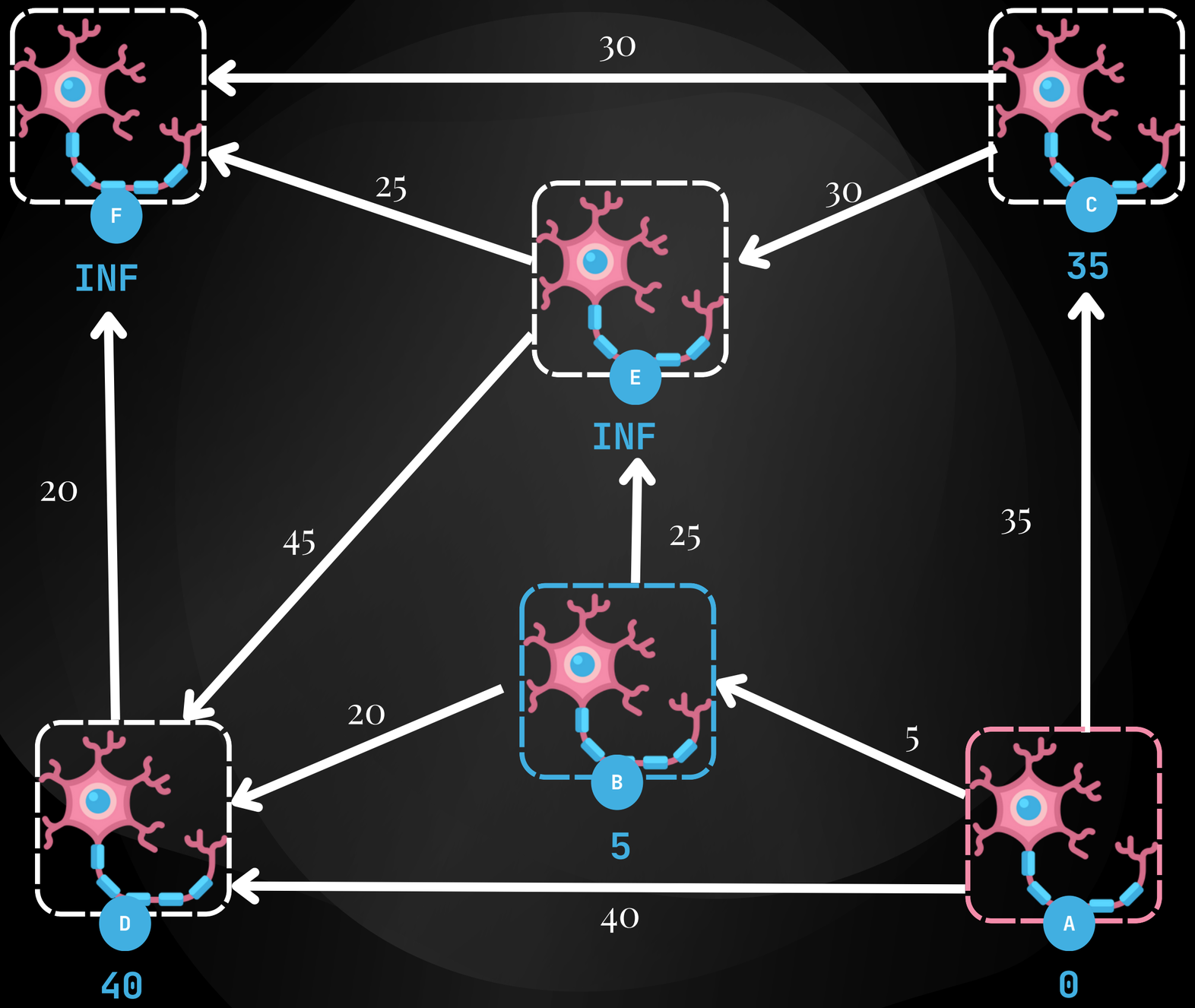
D = 40

Predecesores

A → B

A → C

A → D



Fila prioridad

B = 5

C = 35

D = 40

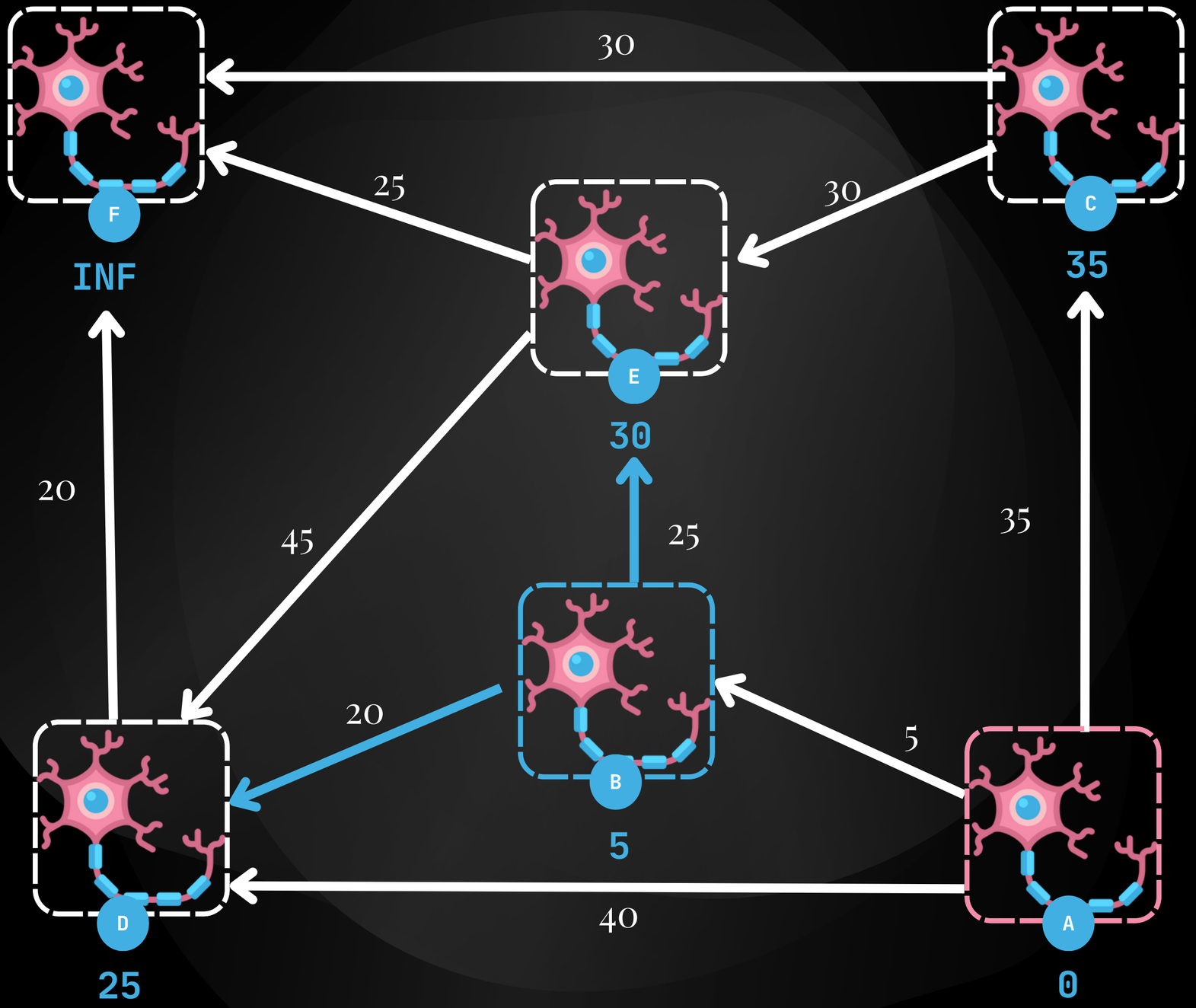
Predecesores

A → B

A → C

B → D

B → E



Fila prioridad

D = 25

E = 30

C = 35

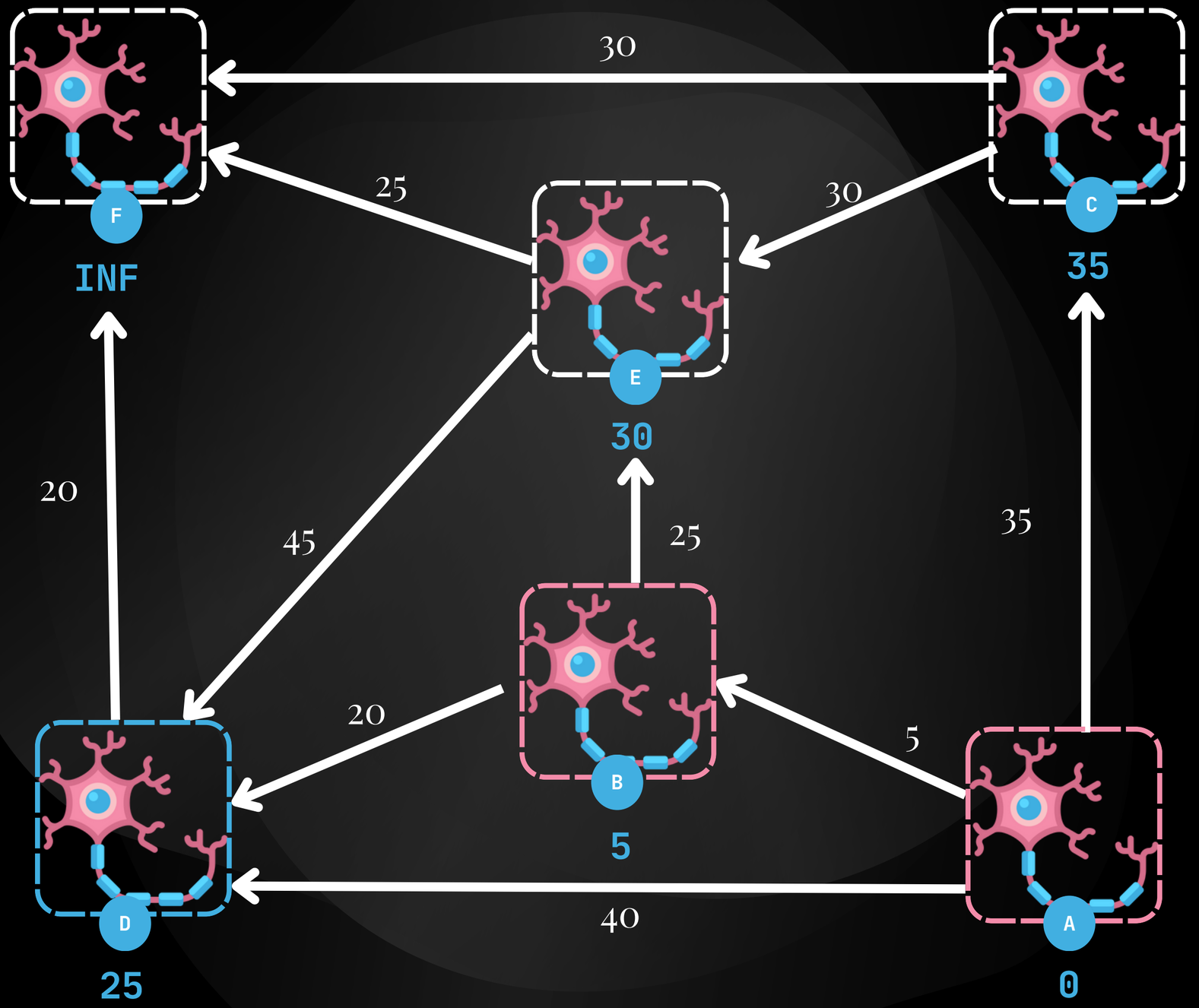
Predecesores

A → B

A → C

B → D

B → E



Fila prioridad

D = 25

E = 30

C = 35

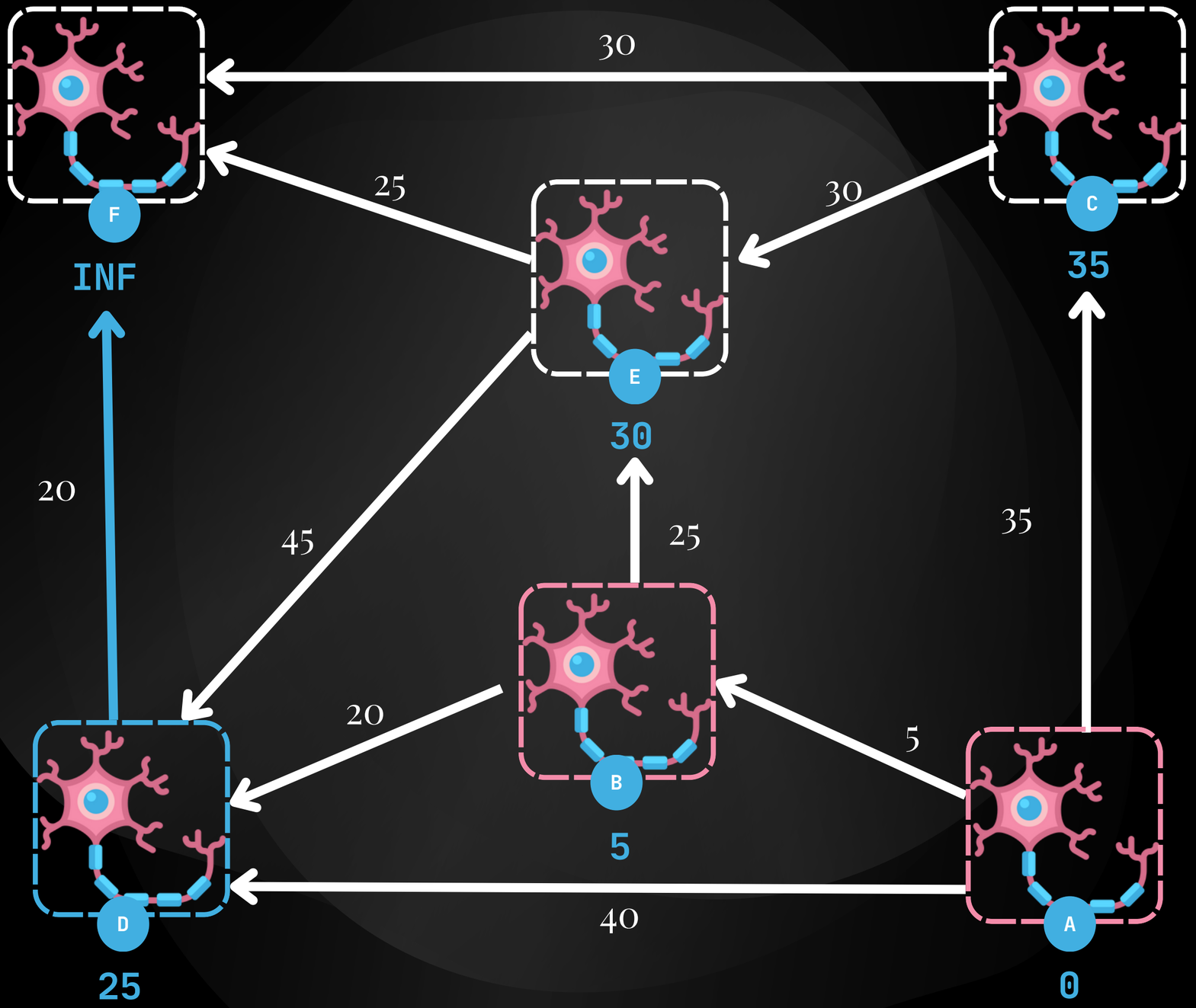
Predecesores

A → B

A → C

B → D

B → E



Fila prioridad

D = 25

E = 30

C = 35

Predecesores

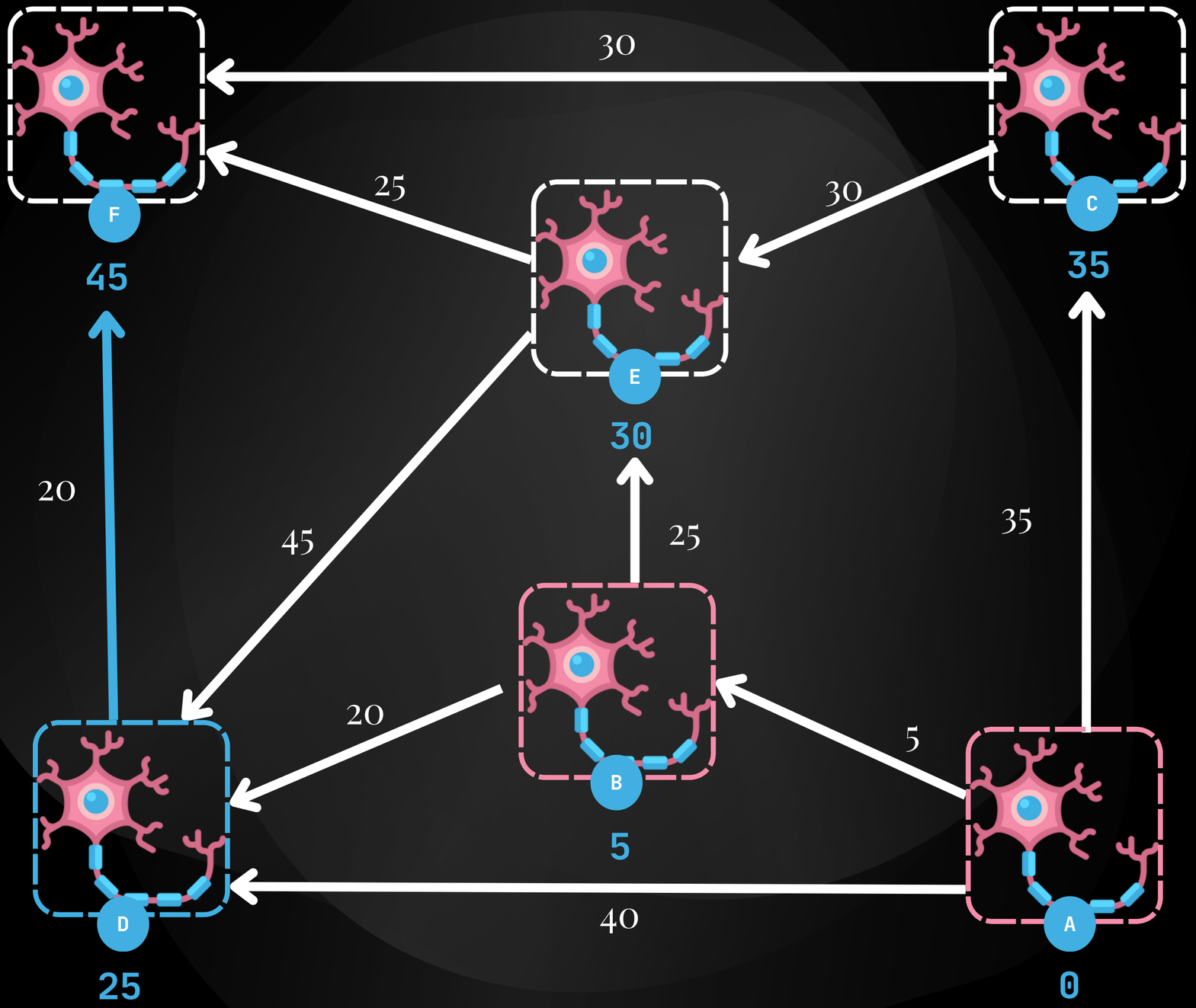
A → B

A → C

B → D

B → E

D → F



Fila prioridad

E = 30

C = 35

F = 45

Predecesores

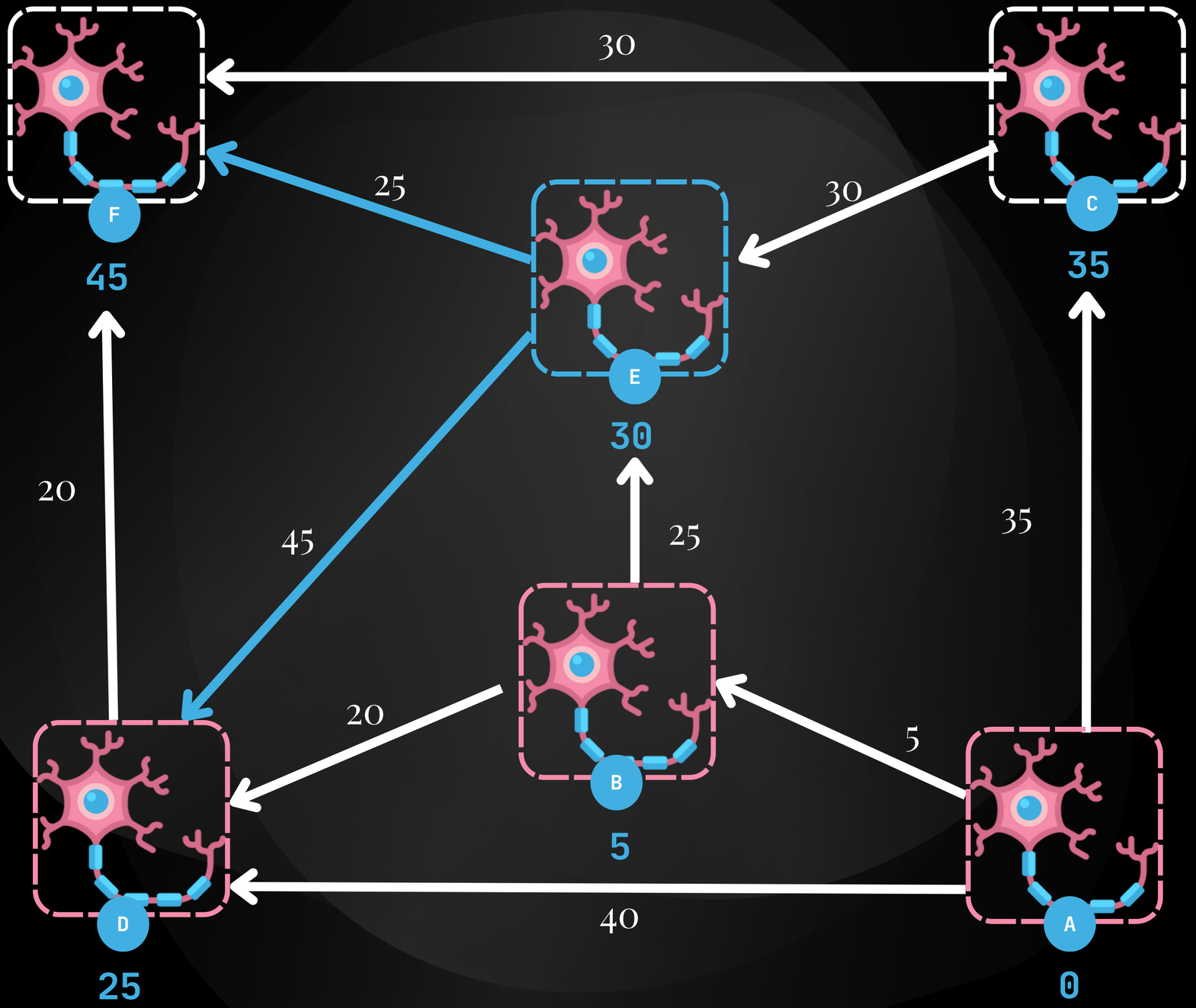
A → B

A → C

B → D

B → E

D → F



Fila prioridad

E = 30

C = 35

F = 45

Predecesores

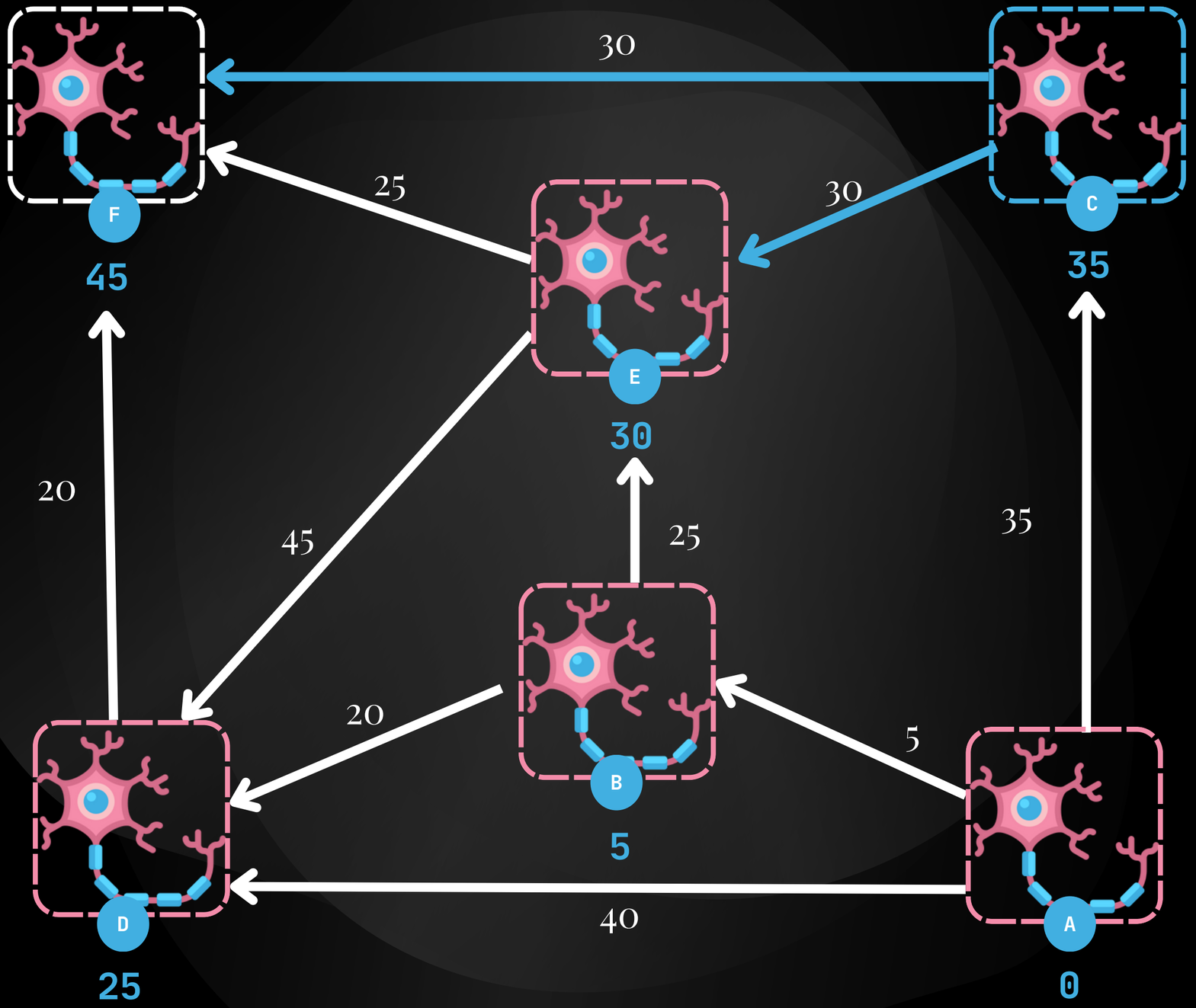
A → B

A → C

B → D

B → E

D → F



Fila prioridad

C = 35

F = 45

Predecesores

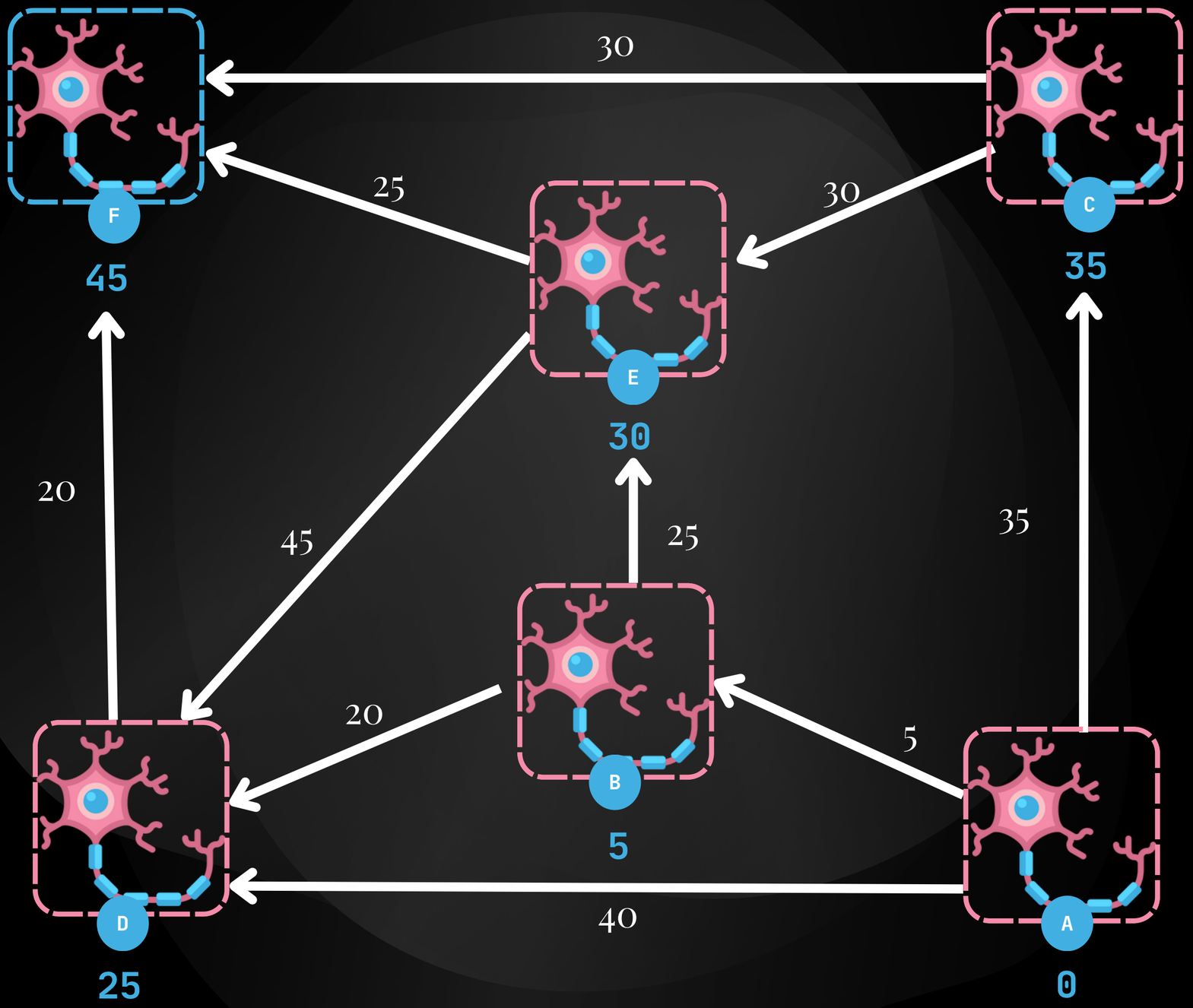
A → B

A → C

B → D

B → E

D → F



Fila prioridad

F = 45

Predecesores

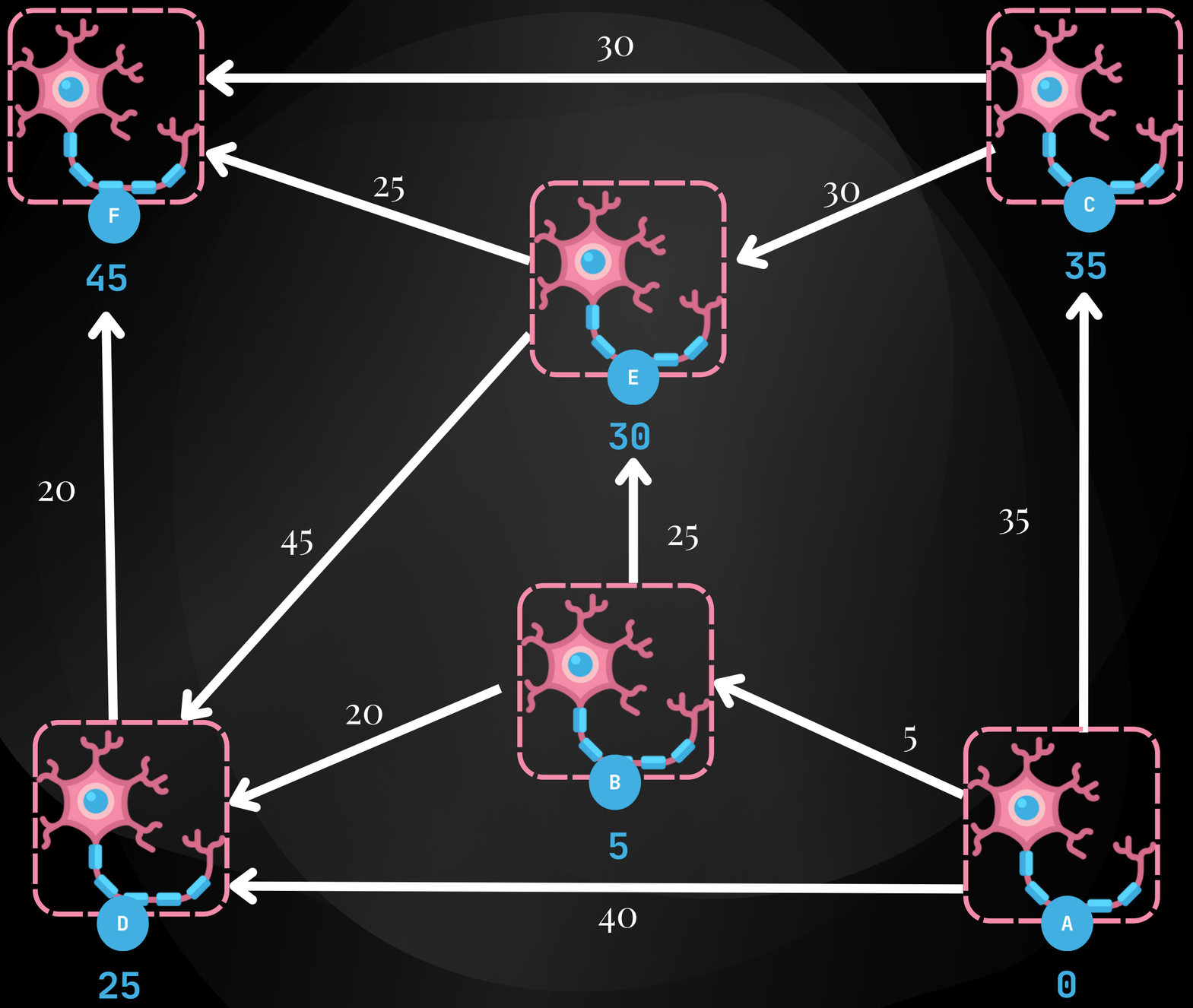
A → B

A → C

B → D

B → E

D → F



Fila prioridad

Hallamos el camino más corto entre A y F

Predecesores

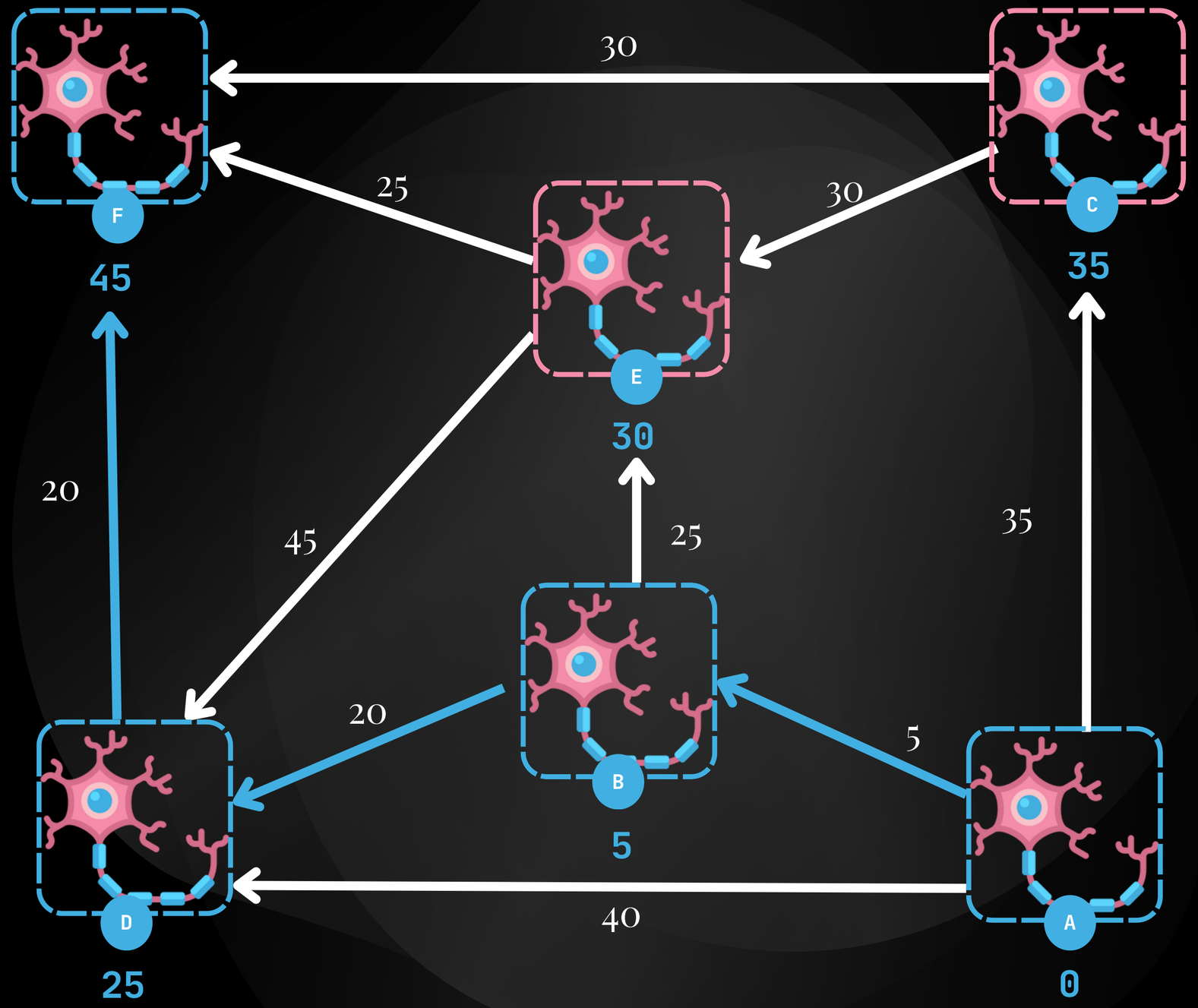
A → B

A → C

B → D

B → E

D → F



¡MUCHAS Gracias!

Reto

Presentación

UNIVERSIDAD NACIONAL DE COLOMBIA