¿Se puede hacer el proyecto?

- Tenemos un proyecto complejo dividido en varias tareas
- Algunas tareas tienen como requisito otras tareas

¿Cómo sabemos si es posible realizar el proyecto completo?

Requisitos inconsistentes

Si la tarea B tiene como requisito la tarea A, entonces escribimos

$$A \to B$$

Si existe alguna secuencia "circular" de requisitos:

$$X \to R \to \cdots \to K \to X,$$

… entonces no es posible realizar el proyecto

¿Es la única condición?

¿Cómo lo verificamos en el computador?

Si recibimos la lista de tareas y de requisitos

… ¿cómo hacemos un programa que revise esto?

¿Cuál será la forma más eficiente de hacerlo?

¿Qué datos recibo?

0: —*la tarea 0 no tiene prerrequsitos*

1: 0 —*la tarea 1 tiene como prerrequisito la tarea 0*

2: 1 —*la tarea 2 tiene como prerrequisito la tarea 1*

3: 2 6 8 —*la tarea 3 tiene como prerrequisitos las tareas 2, 6 y 8*

4: 3 5 9 —*… y así sucesivamente*

5: 6 8

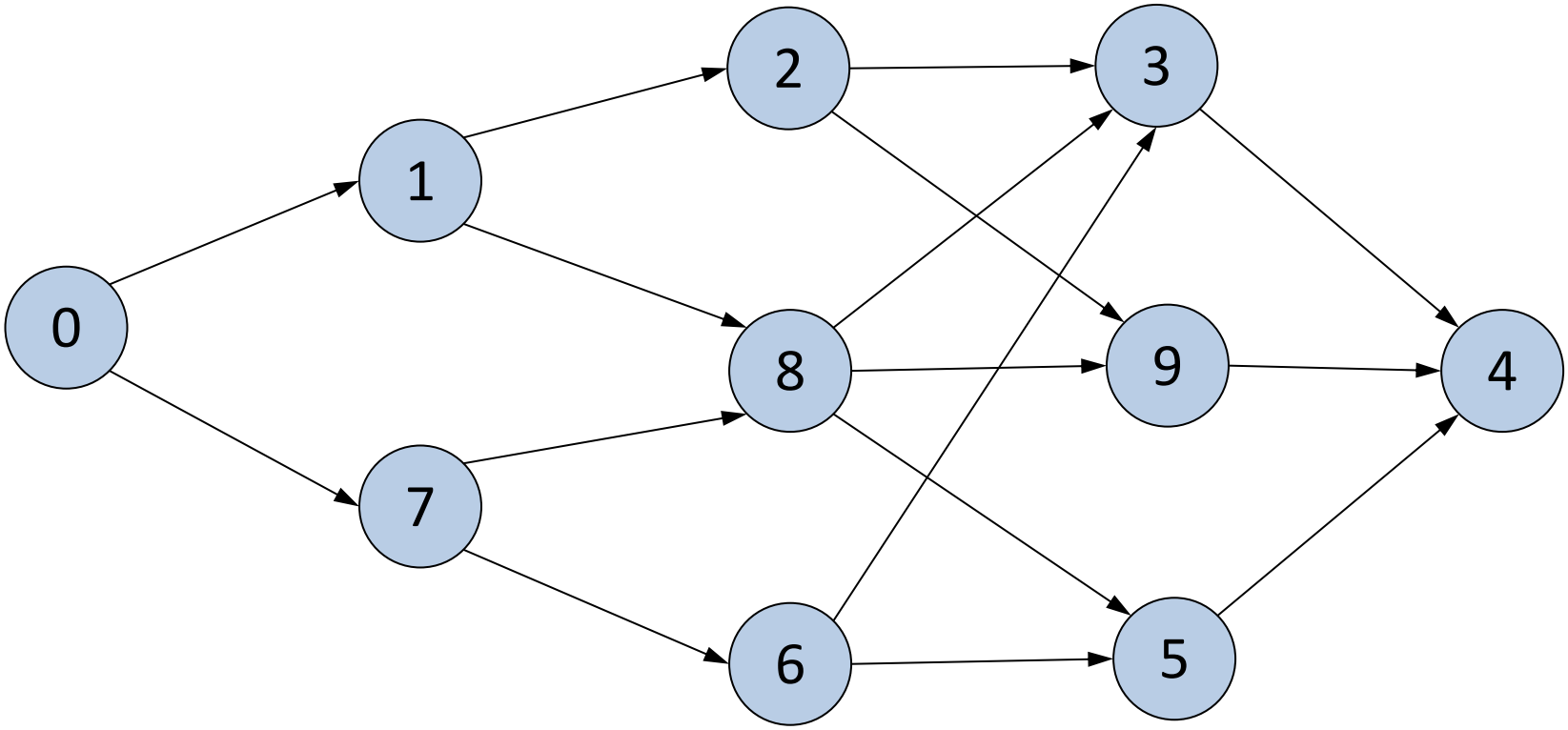
6: 7

7: 0

8: 1 7

9: 2 8

¿Cómo represento los datos visualmente?



La representación visual anterior se conoce como **grafo**

La representación visual anterior se conoce como **grafo**:

- un conjunto *V* de **vértices**, o nodos
- … + un conjunto *E* de **aristas**, o arcos

La complejidad de un algoritmo tiene ahora dos parámetros incluidos en una misma expresión matemática: $|V|$ y $|E|$

P.ej., $O(|V|+|E|)$, que escribimos $O(V+E)$

$O(|E|\log|V|)$, que escribimos $O(E \log V)$

Hay dos tipos principales de grafos

1. **Direccionales** (como el del ejemplo):

   cada arista tiene una *dirección* —va *de un vértice al otro*

2. **No direccionales:**

   las aristas no tienen dirección —van, o están*, entre vértices*

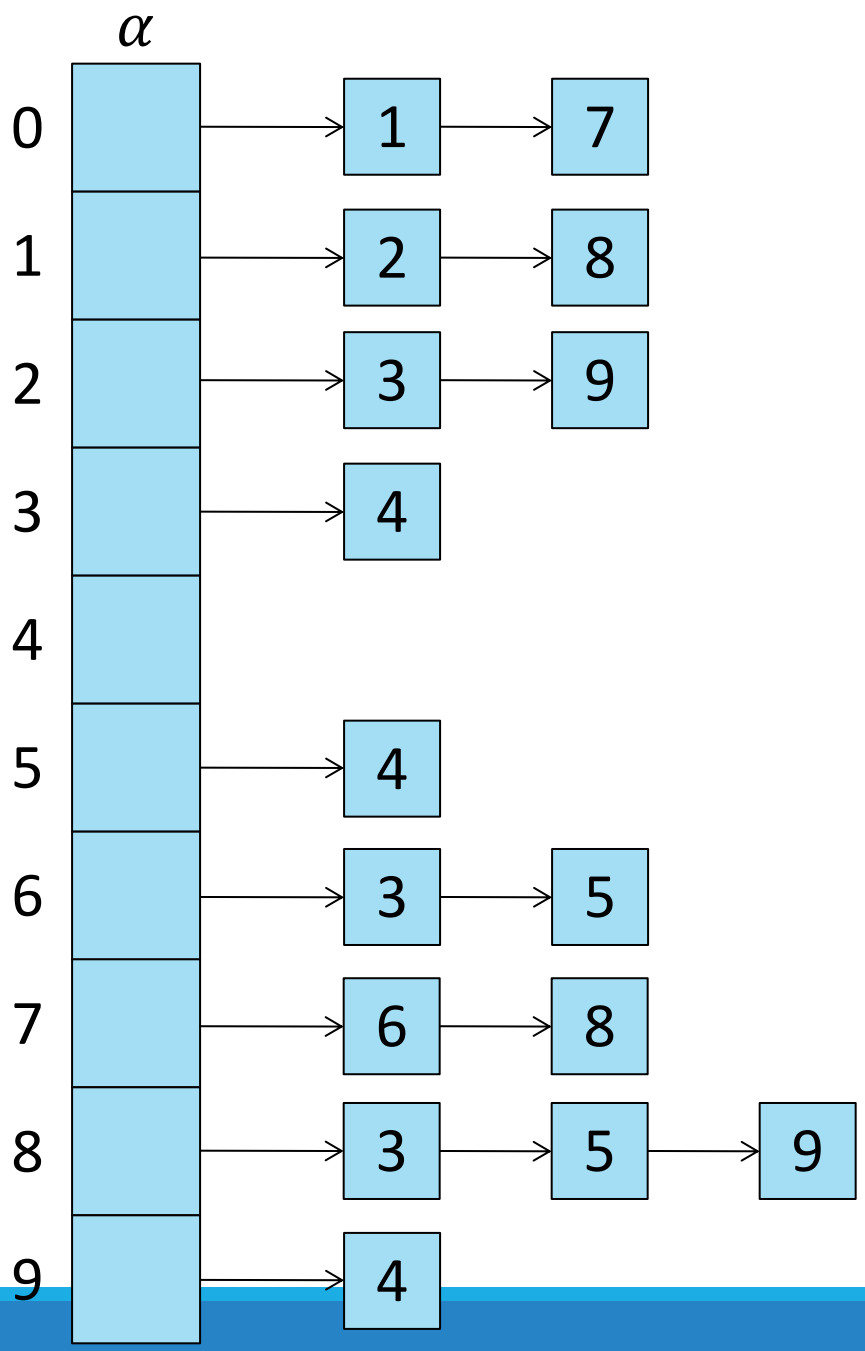
Hay dos formas de representar un grafo en la memoria del computador

1. **Listas de adyacencias**

   Cada nodo tiene una lista de los nodos a los que tiene una arista

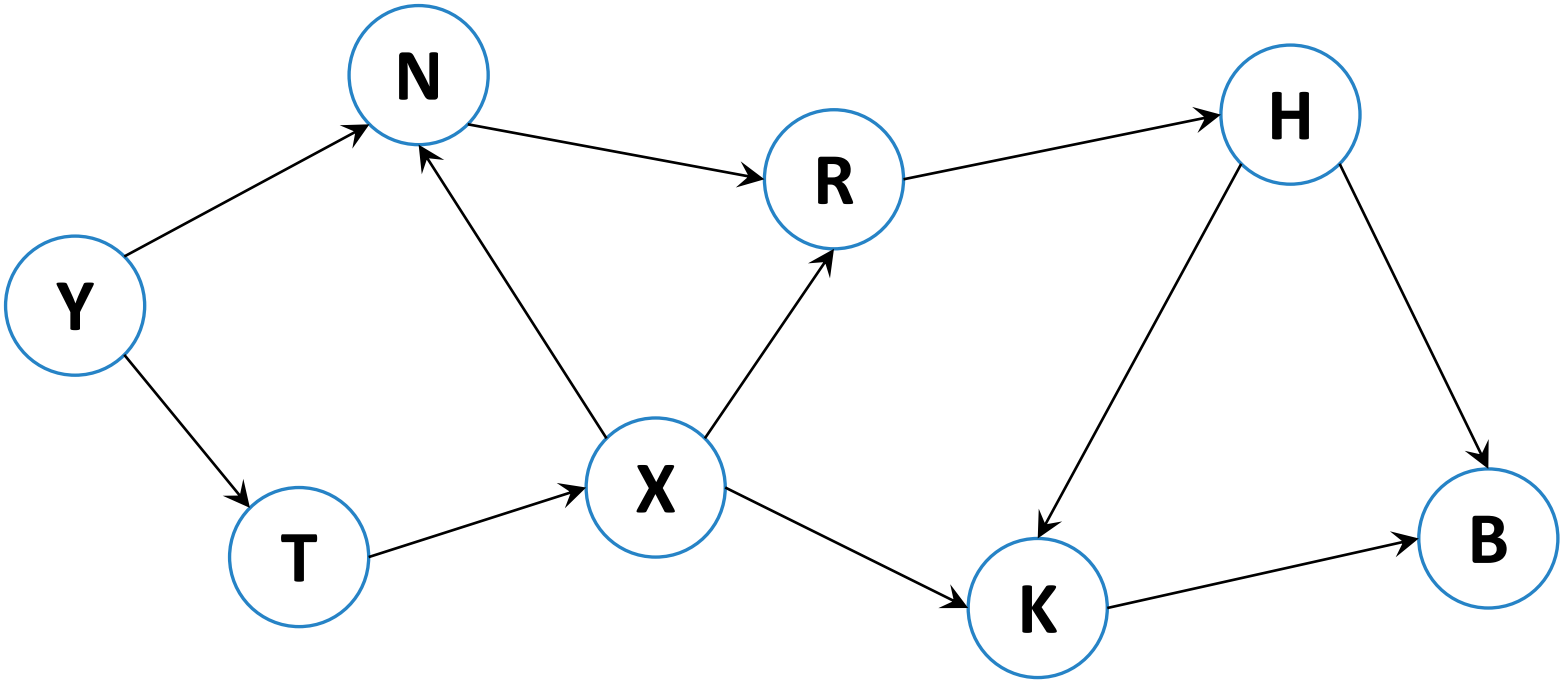
2. **Matriz de adyacencias**

   La coordenada x, y de la matriz indica si la arista (x, y) está en el grafo



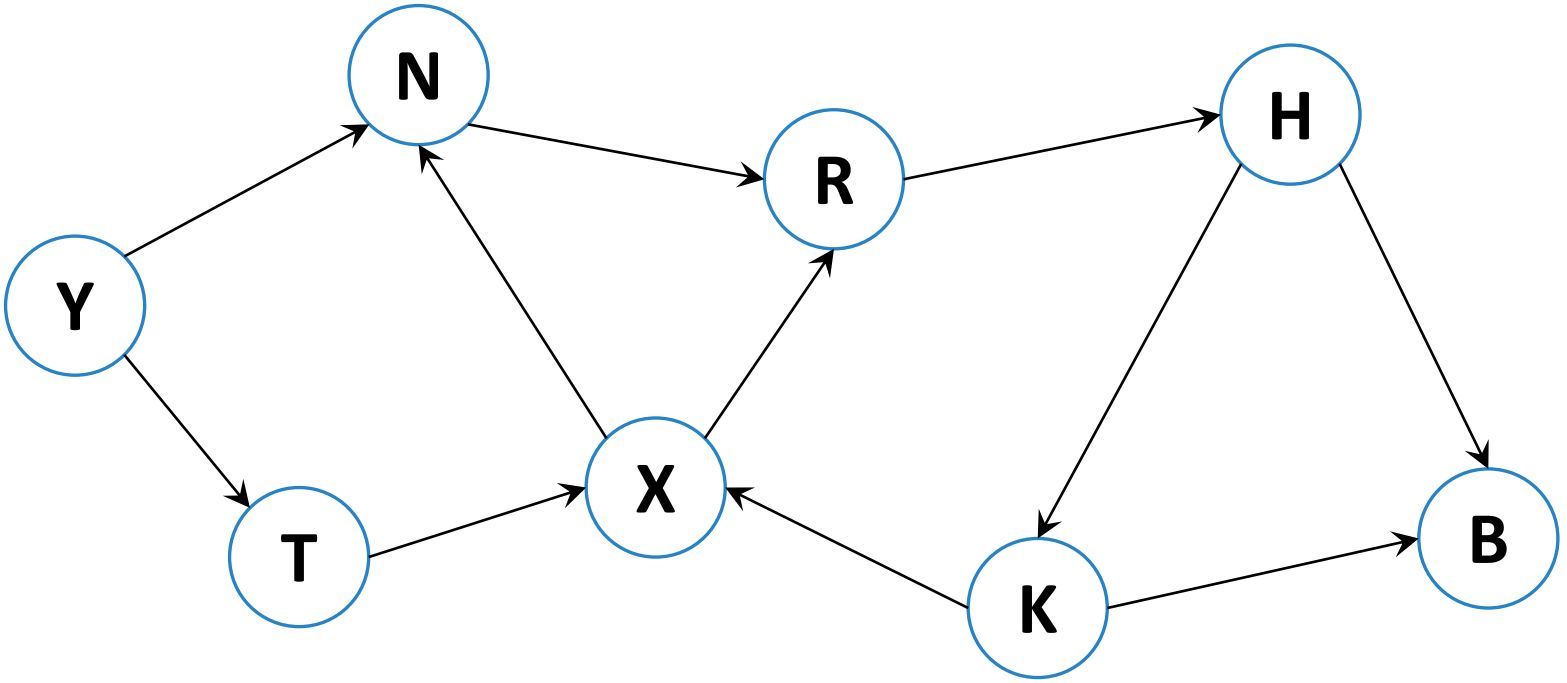
El grafo direccional anterior representado por 10 listas de adyacencias, $\alpha[i]$

Dibujemos el grafo de un proyecto



¿Algún problema con este proyecto?

¿Qué pasa en este caso?



¿Algún problema con este proyecto?

Los grafos direccionales pueden contener **ciclos**

Si el grafo direccional que representa a un proyecto tiene un **ciclo**, entonces el proyecto no puede llevarse a cabo

¿Cómo podemos buscar ciclos en un grafo de manera eficiente?

Definimos la relación *es posterior a*

Diremos que una tarea Y **es posterior a** una tarea X si:

$$X \rightarrow Y$$

o

Existe una tarea Z tal que $X \rightarrow Z$, y Y **es posterior a** Z

Significa que X debe realizarse antes que Y

¿Qué pasa si una tarea es posterior a sí misma?

Si una tarea **es posterior a** sí misma, entonces forma parte de un **ciclo**

¿Cómo podemos identificar las tareas posteriores a una tarea?

Pensemos en la definición de la propiedad

$\boldsymbol{posteriores}(\boldsymbol{X})$:

$\boldsymbol{P} \leftarrow \emptyset$

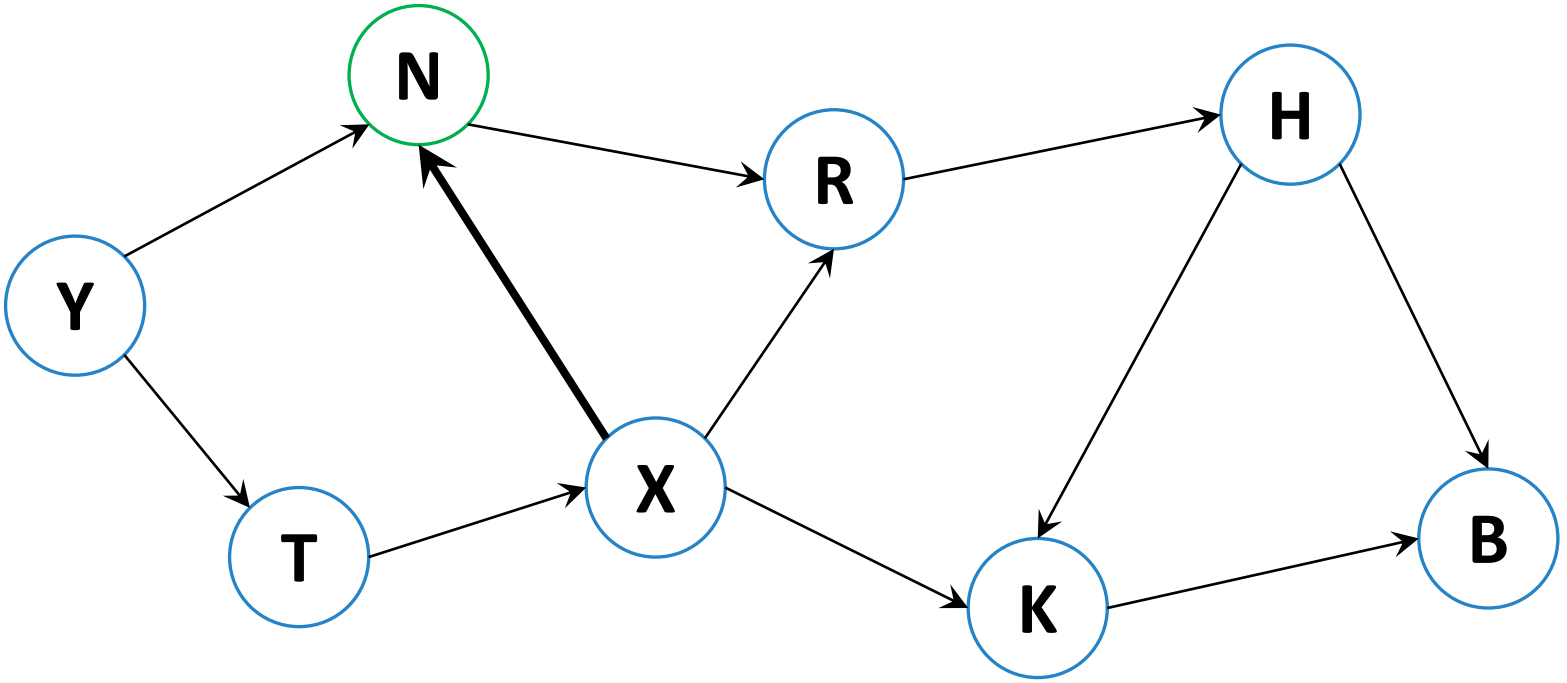
for $\boldsymbol{Y}$ tal que $\boldsymbol{X} \rightarrow \boldsymbol{Y}$:

$\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{\boldsymbol{Y}\}$

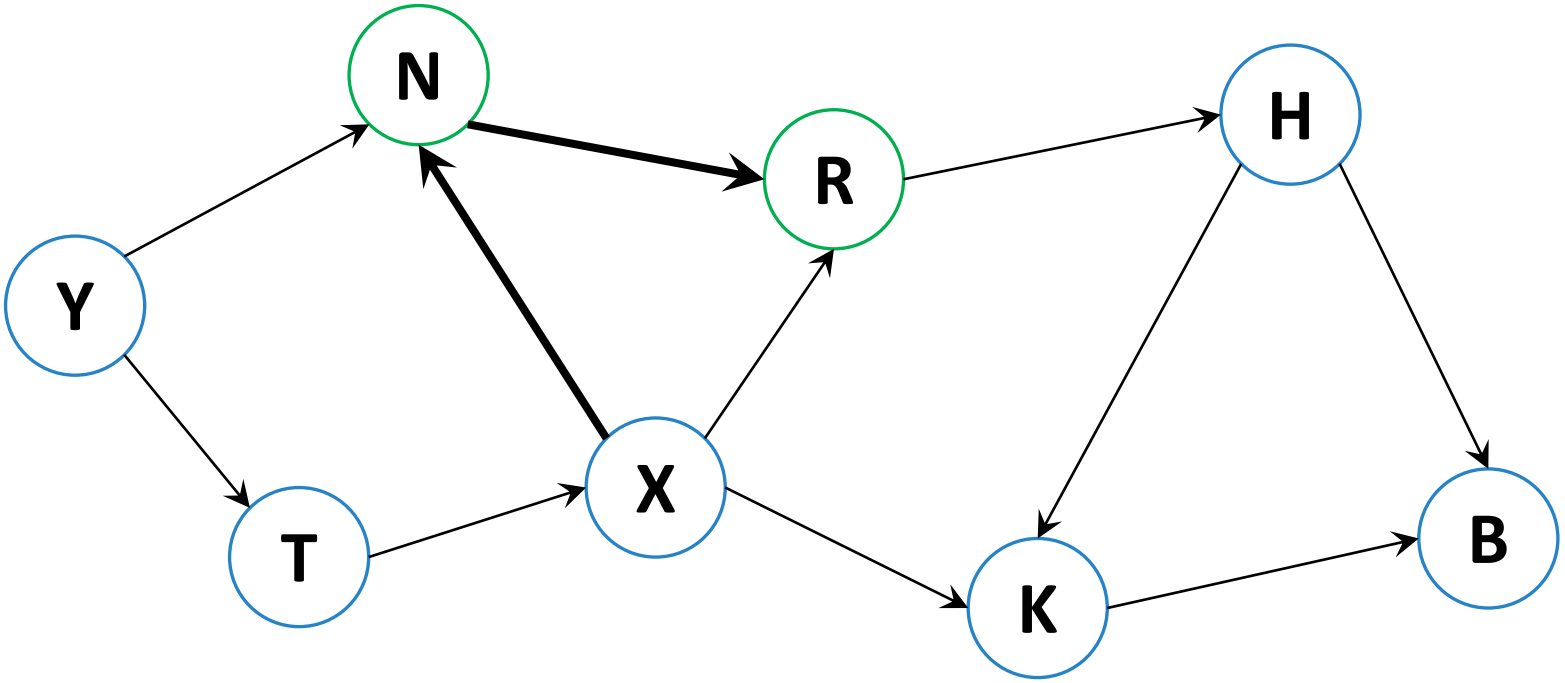
$\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \boldsymbol{posteriores}(\boldsymbol{Y})$

return $\boldsymbol{P}$

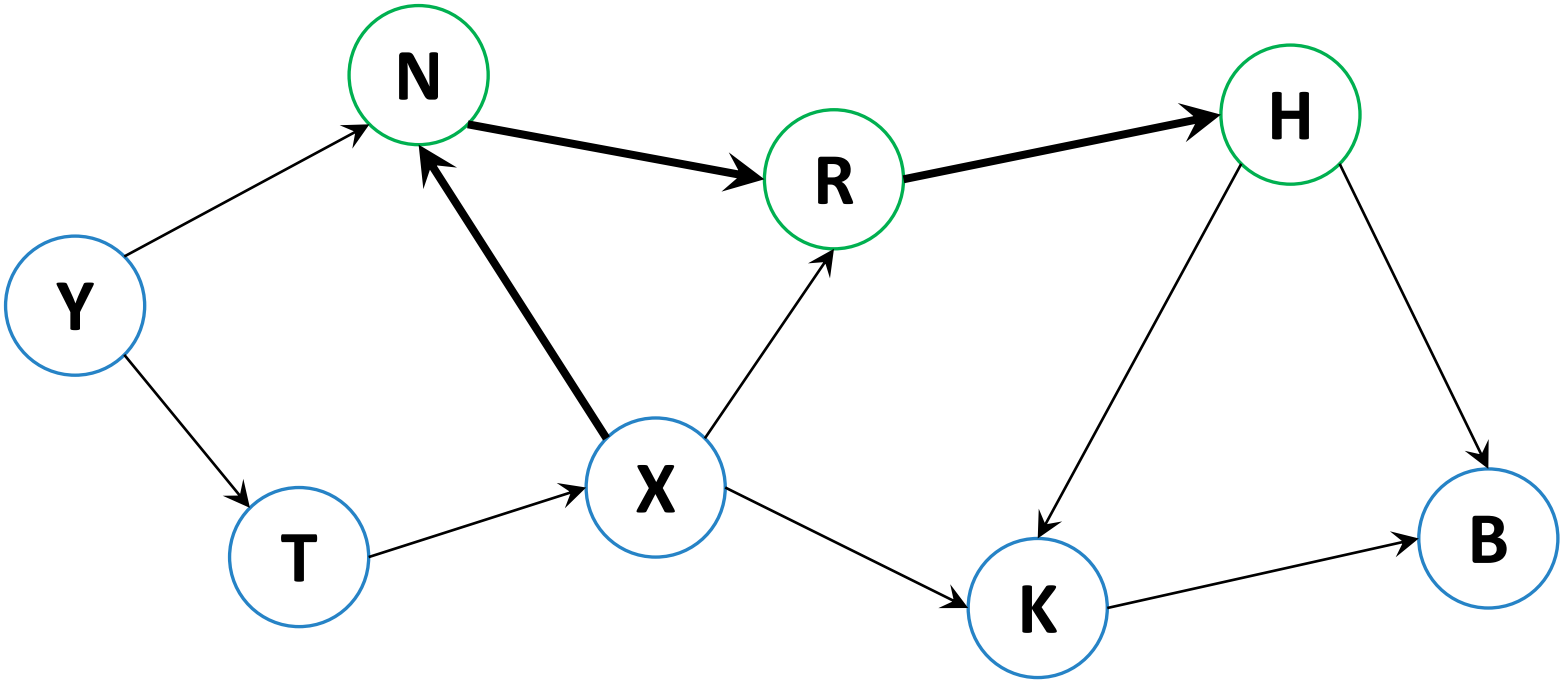
¿Cuáles tareas, o nodos, son posteriores a *X*?



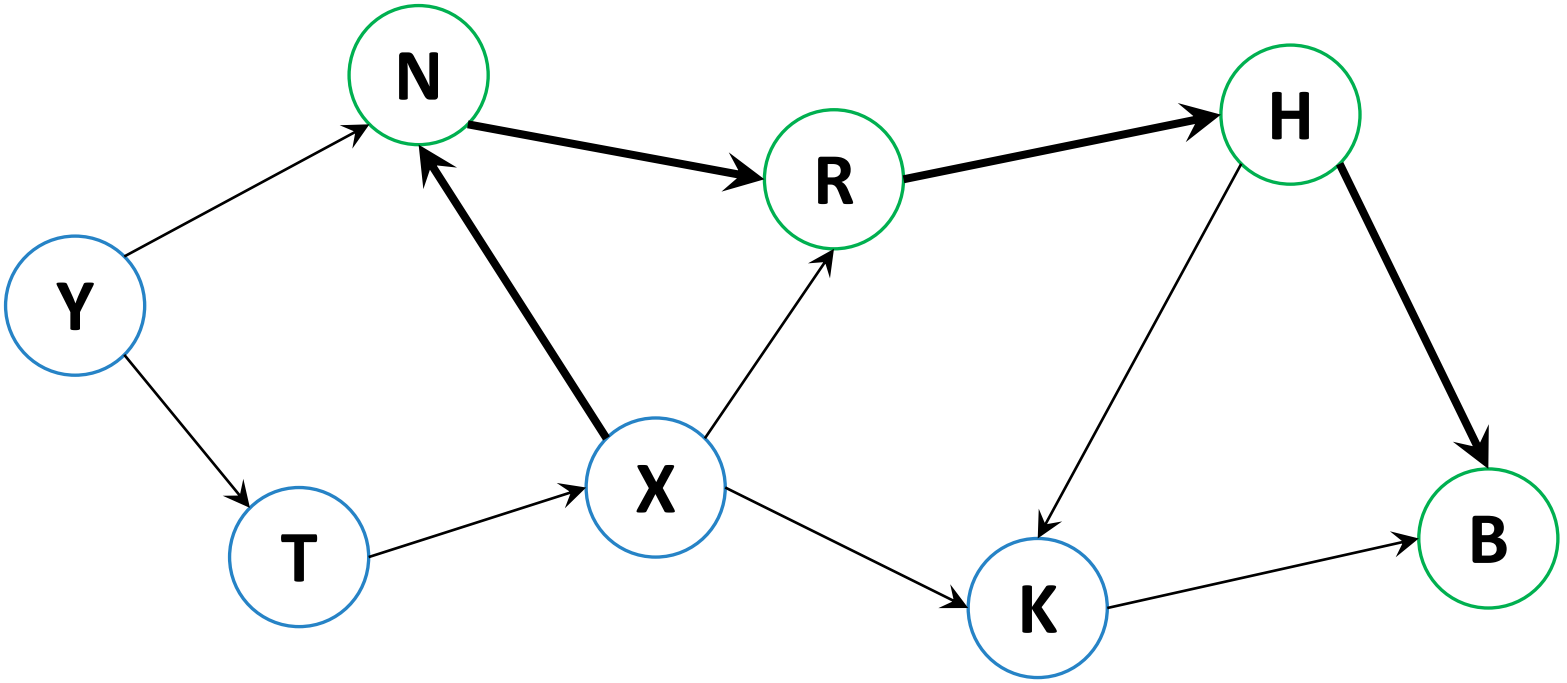
¿Cuáles nodos son posteriores a *N*?



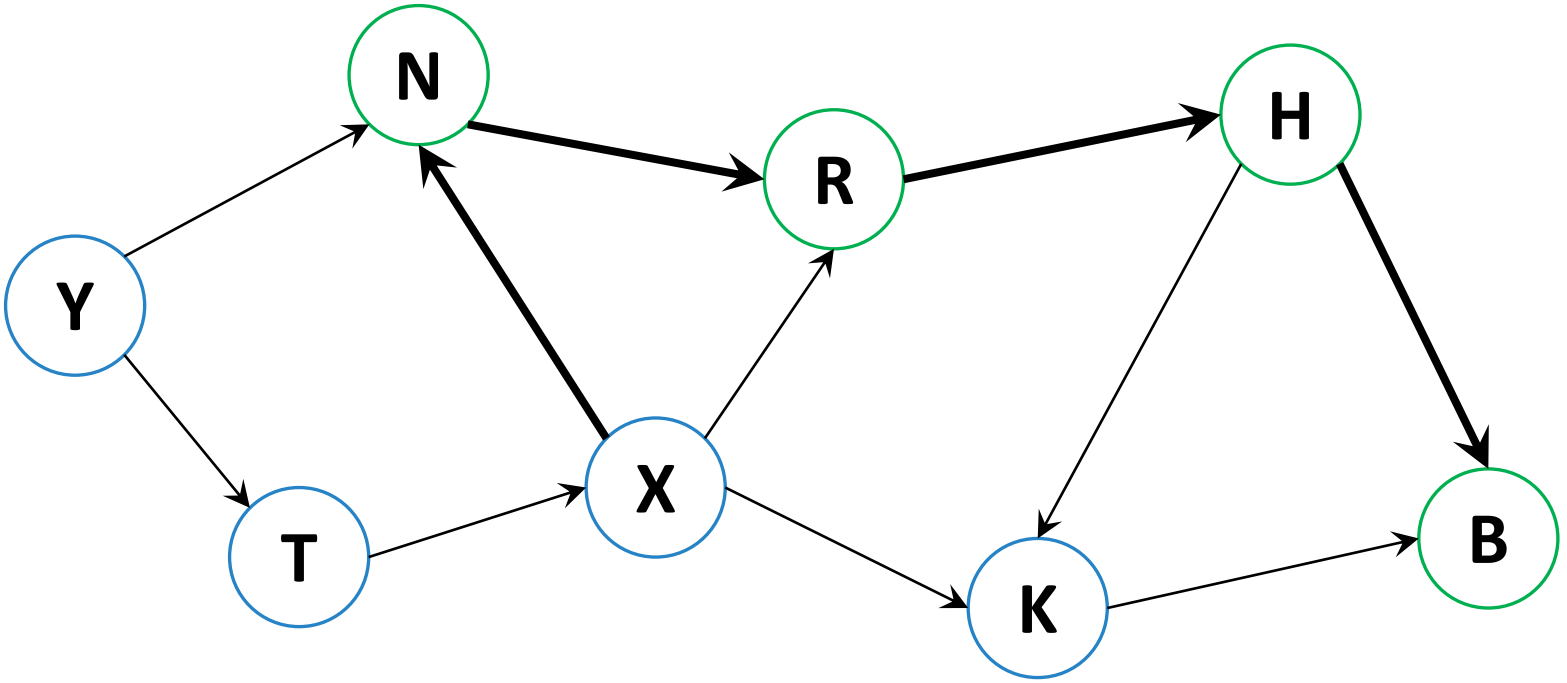
¿Cuáles nodos son posteriores a *R*?



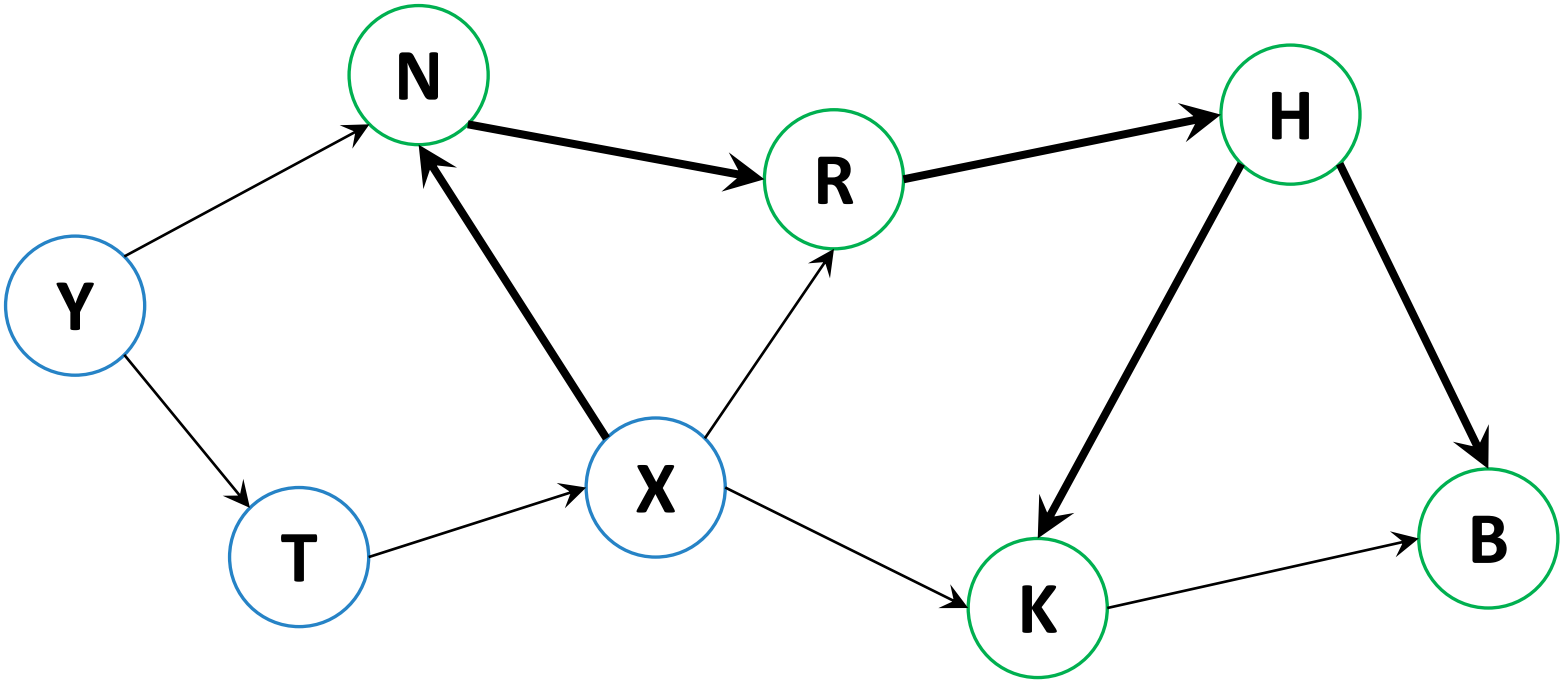
¿Cuáles nodos son posteriores a *H*?



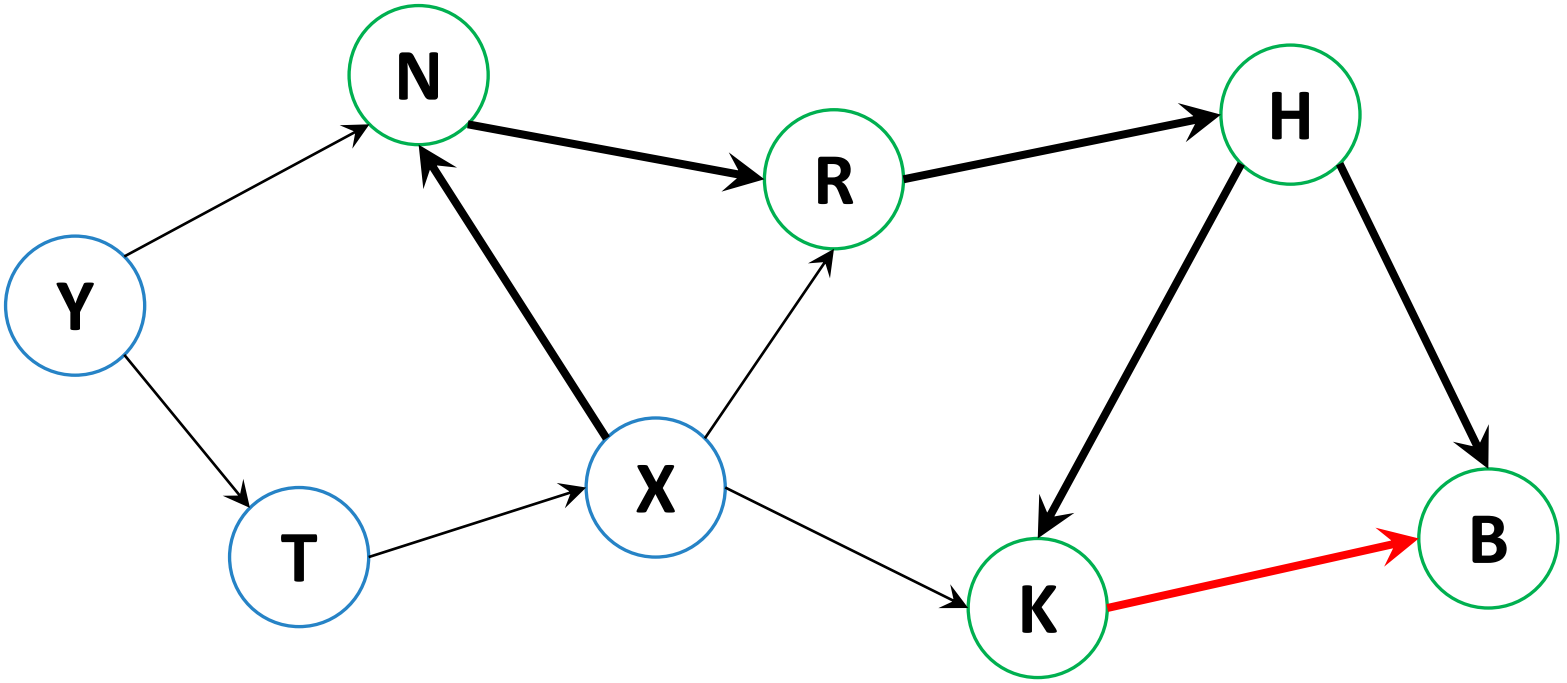
¿Cuáles nodos son posteriores a *B*?



¿Cuáles nodos son posteriores a *K*?



¿Cuáles nodos son posteriores a *B*?



Espera … ya preguntamos por los nodos posteriores a *B*

Nodos por los que ya pasamos

Estamos haciendo llamadas a $\boldsymbol{posteriores}()$ repetidas: con el mismo nodo como parámetro

Es más, si el nodo forma parte de un ciclo, entonces el algoritmo no termina

¿Cómo se soluciona esto?

$\boldsymbol{posteriores}(\boldsymbol{X})$:

    if $\boldsymbol{X}$ está pintado: ***return*** $\emptyset$

    pintar $\boldsymbol{X}$

    $\boldsymbol{P} \leftarrow \emptyset$

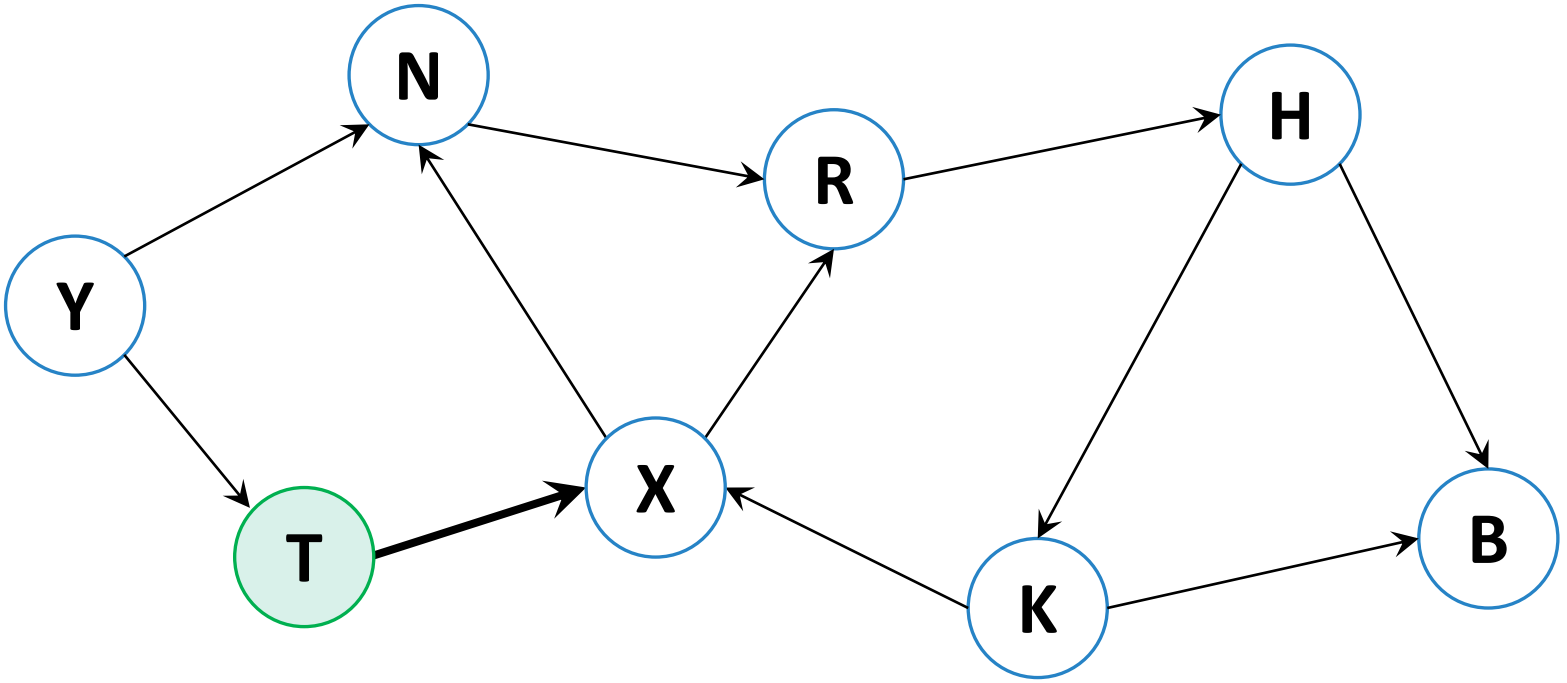
    for $\boldsymbol{Y}$ tal que $\boldsymbol{X} \rightarrow \boldsymbol{Y}$:

        $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \{\boldsymbol{Y}\}$

        $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup \boldsymbol{posteriores}(\boldsymbol{Y})$

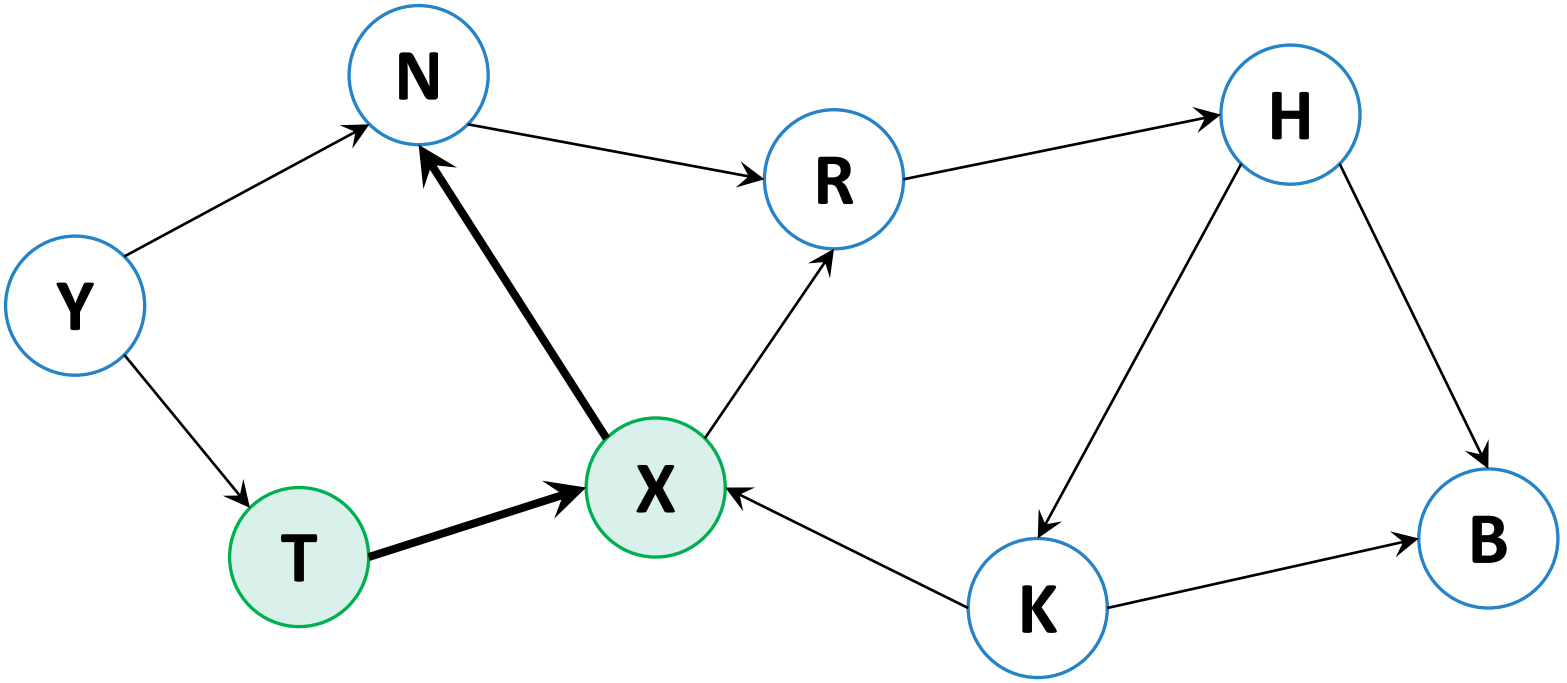
    return $\boldsymbol{P}$

Veamos ahora:
¿cuáles son los nodos posteriores a *T*?



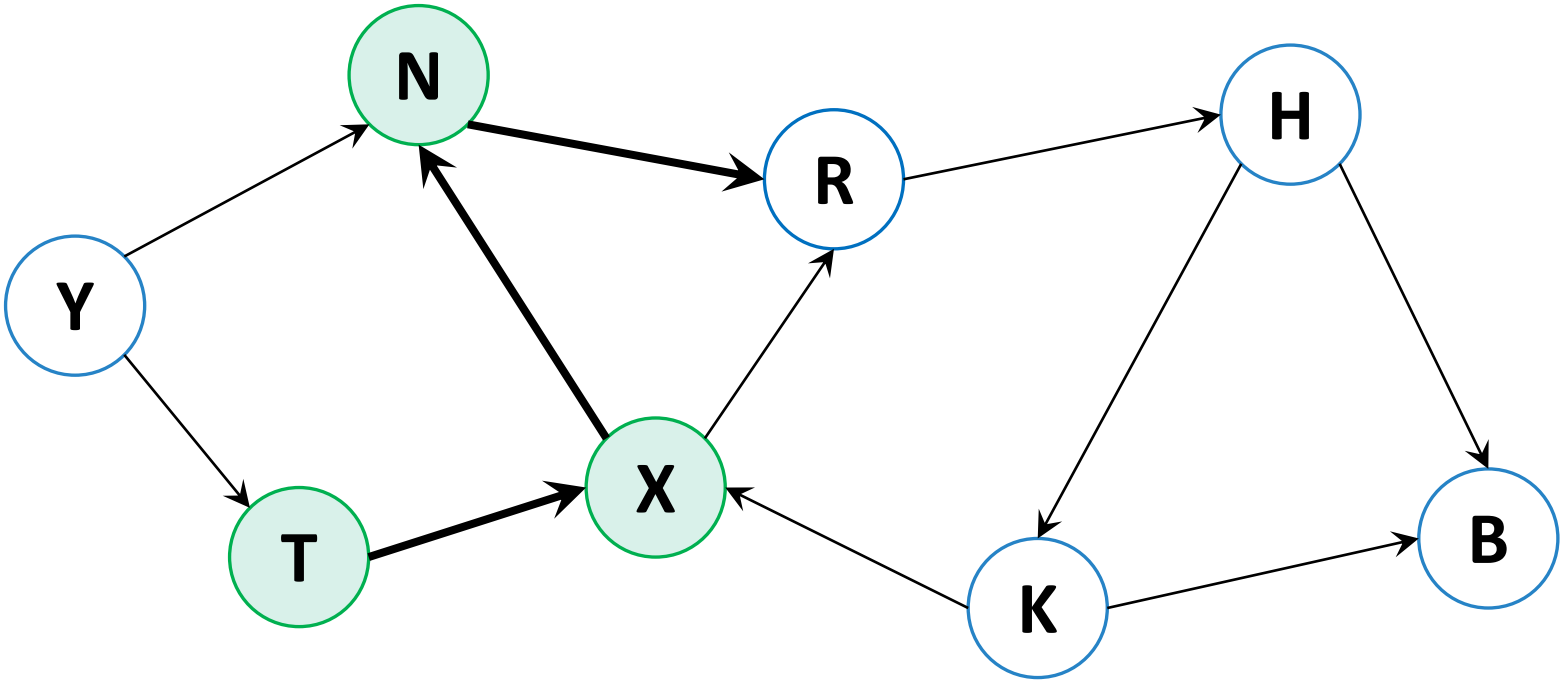
T no está pintado, lo pintamos;
hay una sola arista que sale de *T*,
la que va a *X* … sigámosla

Pintamos *X*, y buscamos los nodos posteriores a *X*



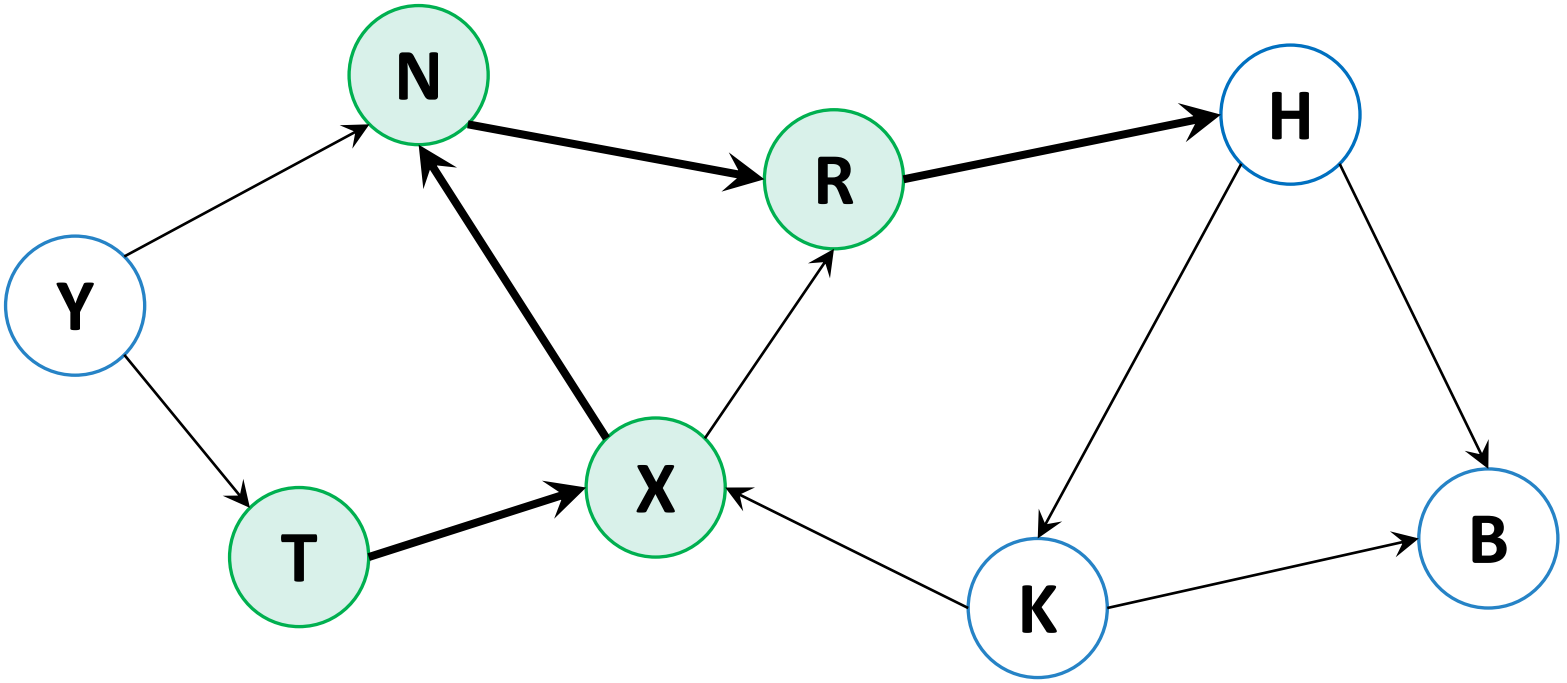
Hay dos aristas que salen de *X*;
seguimos la que va a *N* … y dejamos pendiente la que va a *R*

Pintamos *N*, y buscamos los nodos posteriores a *N*



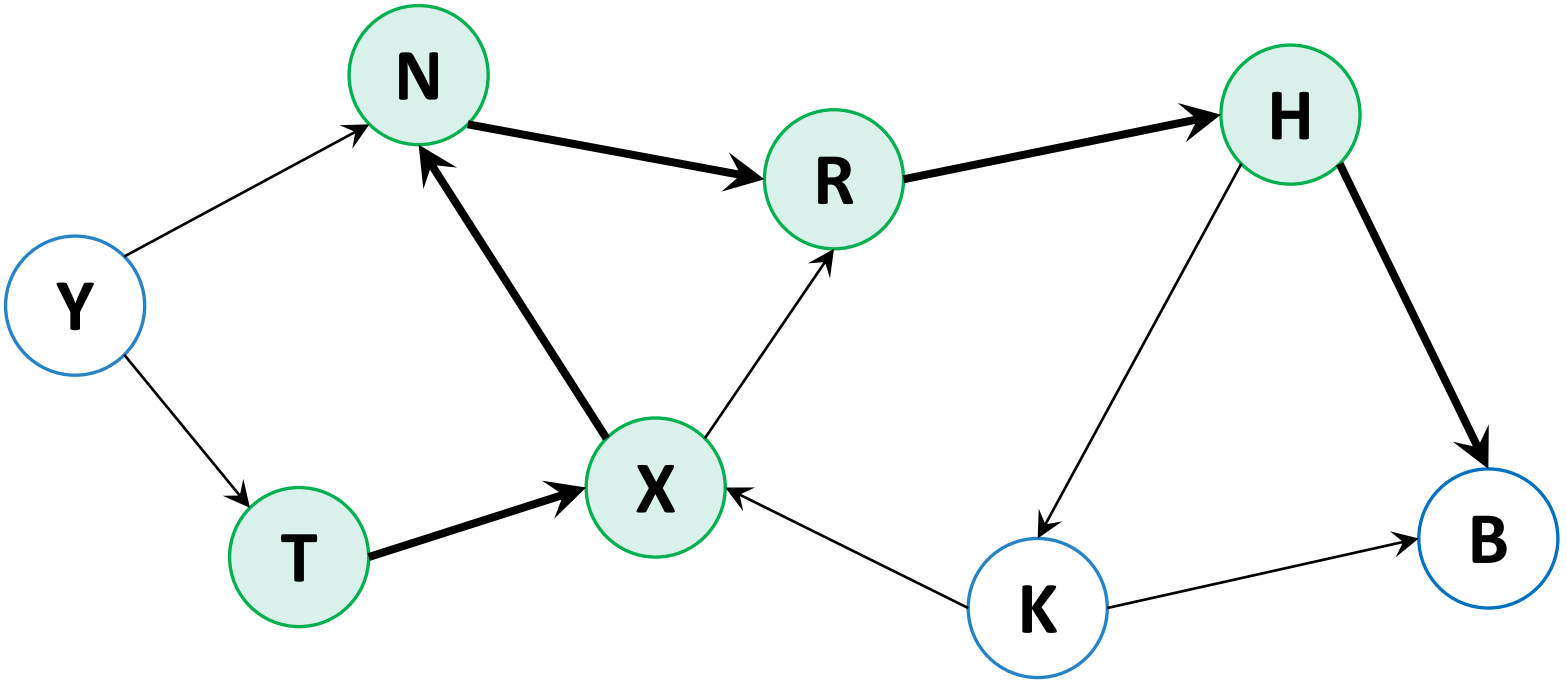
Hay una sola arista que sale de *N*;
la seguimos

Pintamos *R*, y buscamos los nodos posteriores a *R*



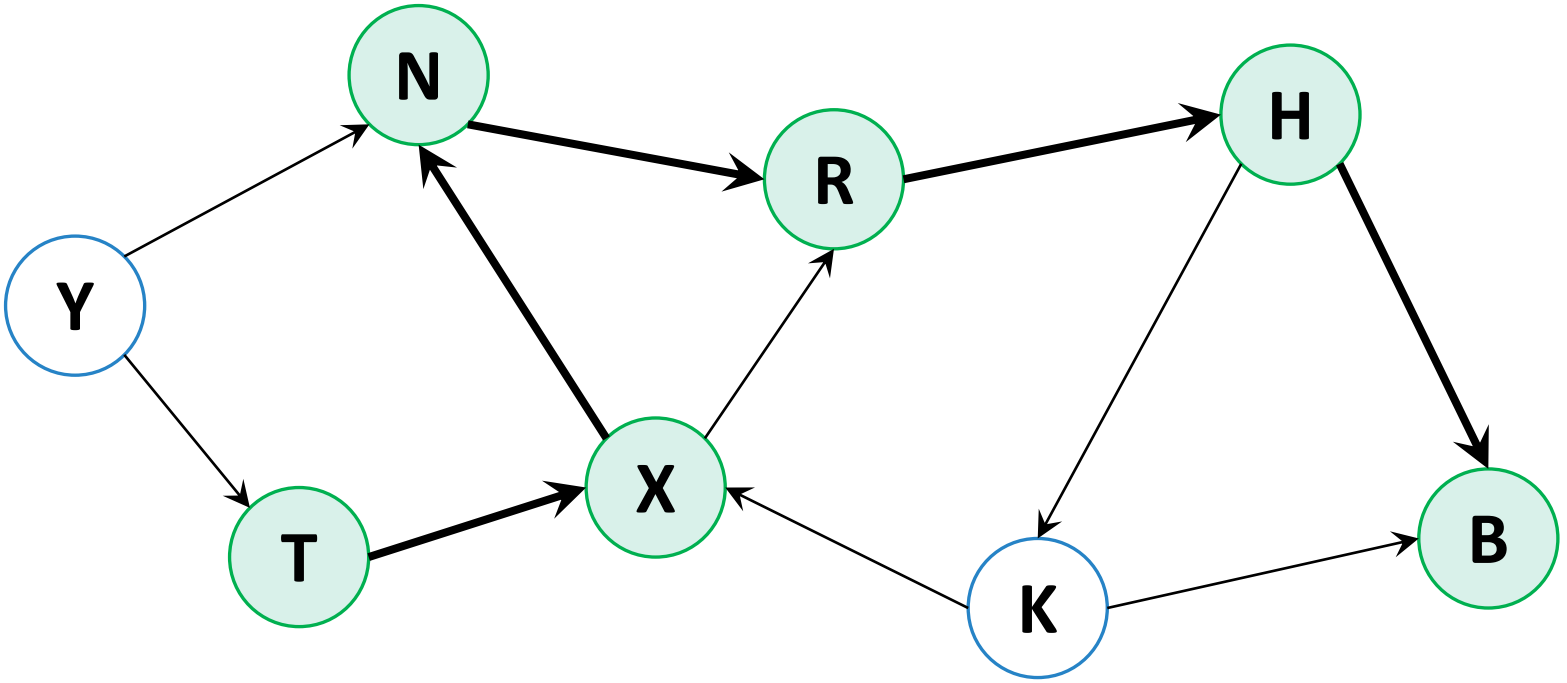
Hay una sola arista que sale de *R*;
la seguimos

Pintamos *H*, y buscamos los nodos posteriores a *H*



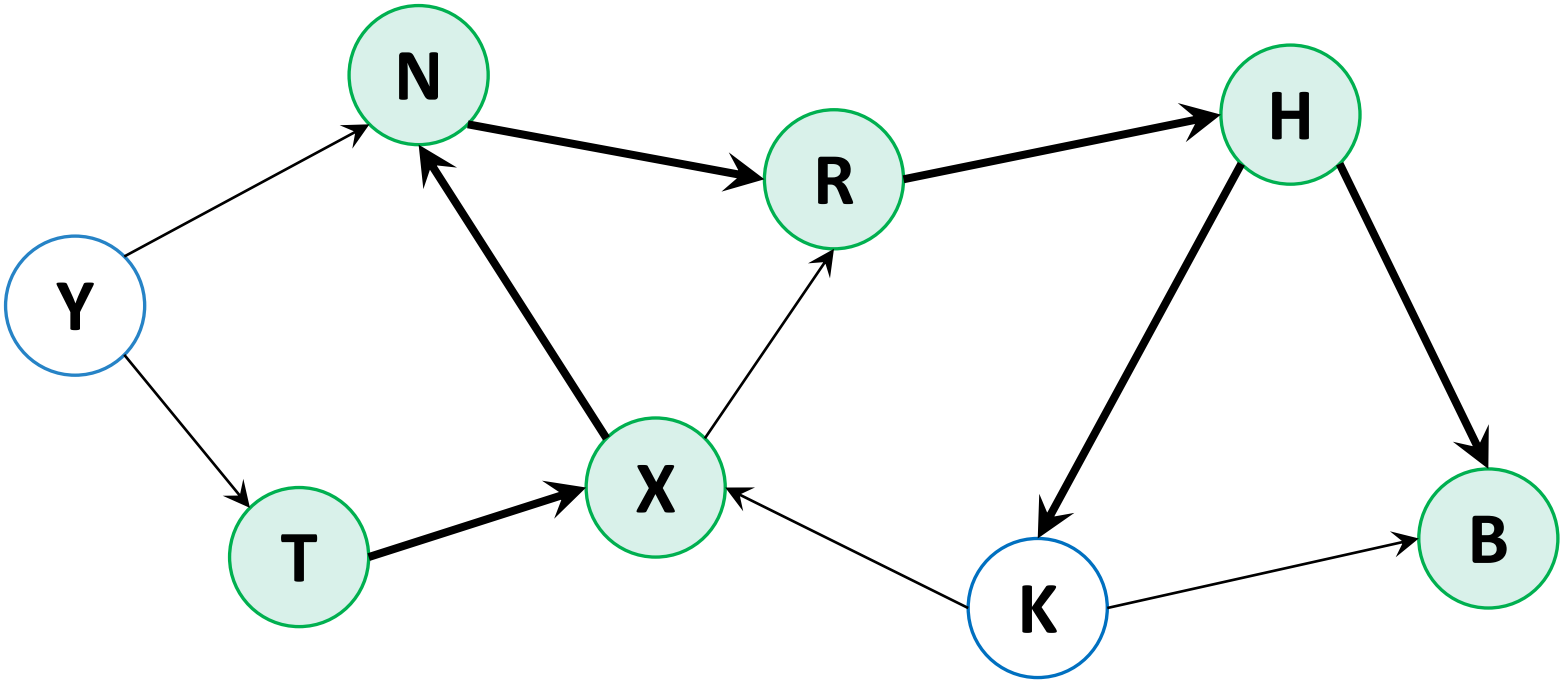
Hay dos aristas que salen de *H*; seguimos la que va a *B* … y dejamos pendiente la que va a *K*

Pintamos *B*, y buscamos los nodos posteriores a *B*



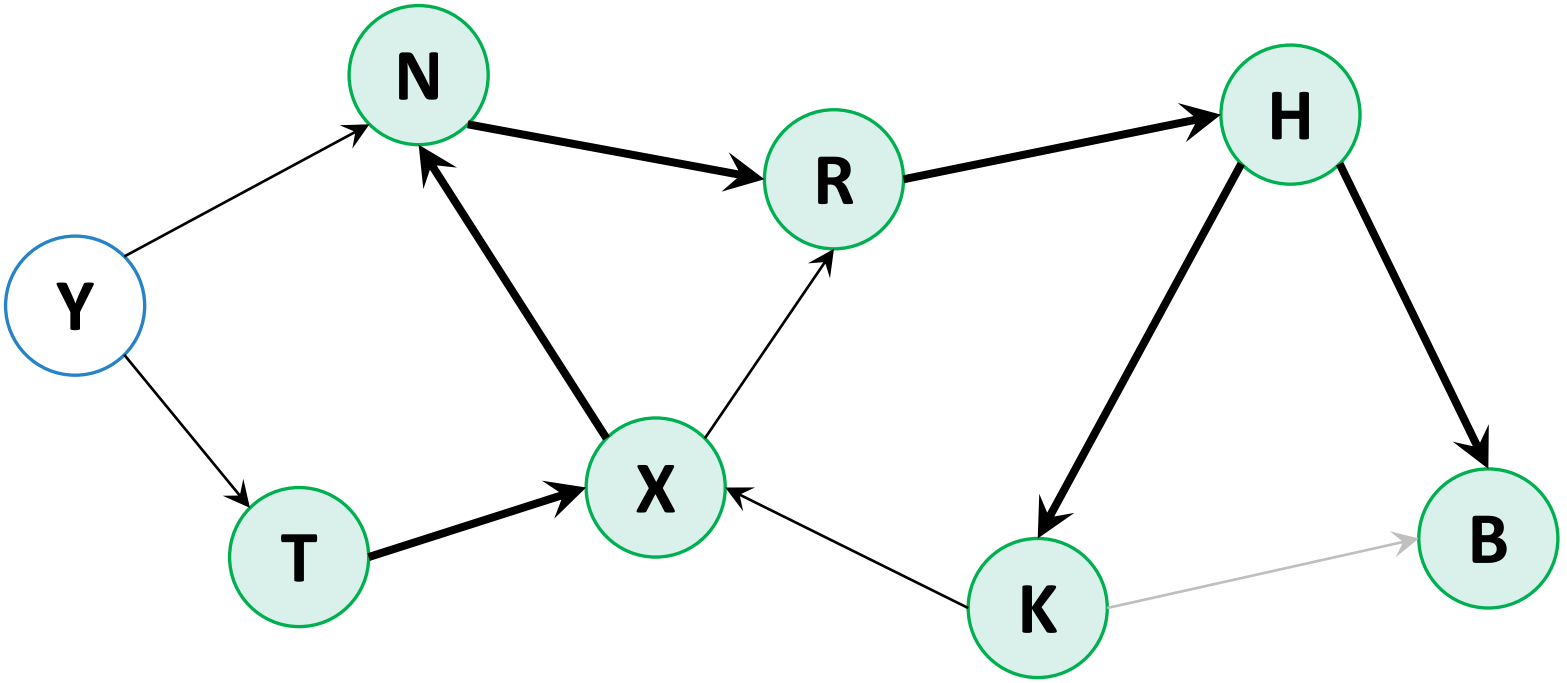
No hay aristas que salgan de *B*;
volvemos a *H*

Seguimos buscando nodos posteriores a *H*



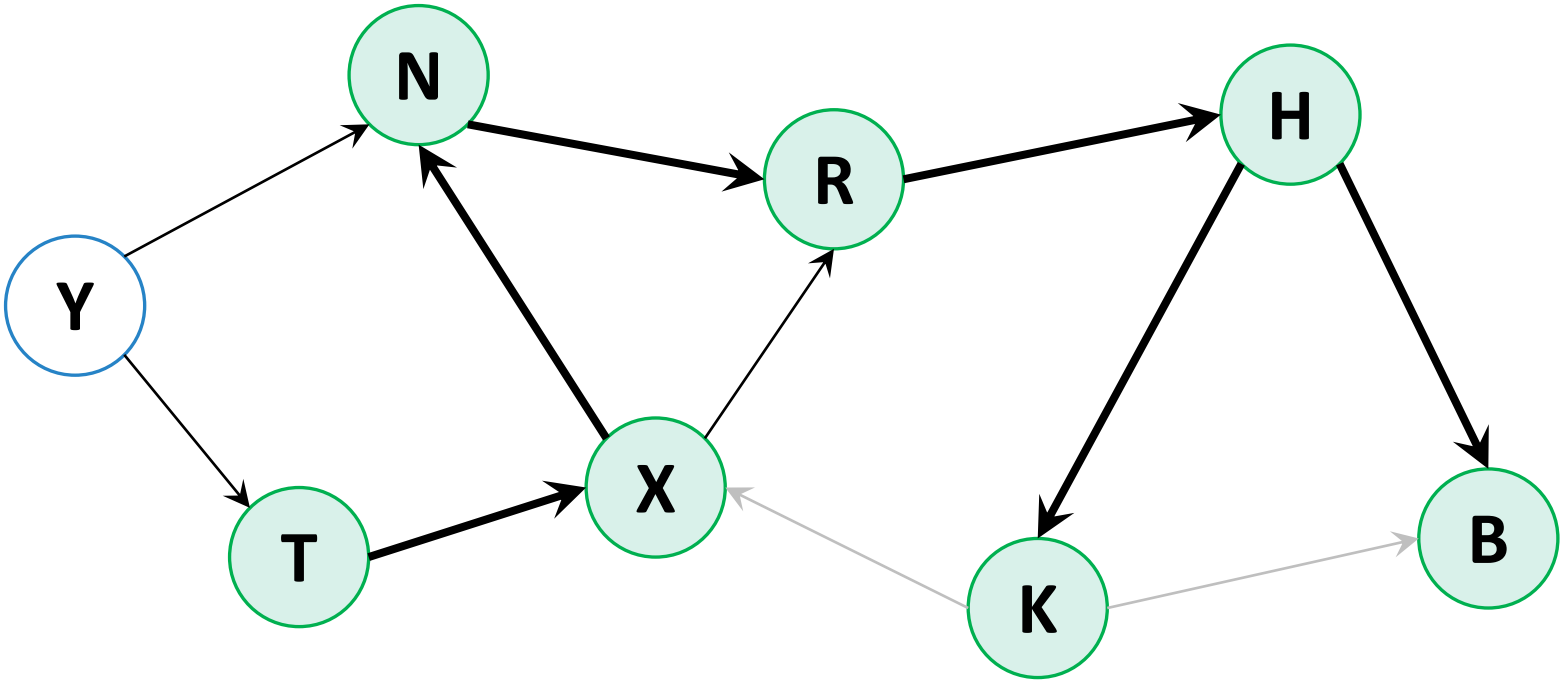
Seguimos la segunda arista que sale de *H*; la que va a *K*

Pintamos *K*, y buscamos los nodos posteriores a *K*



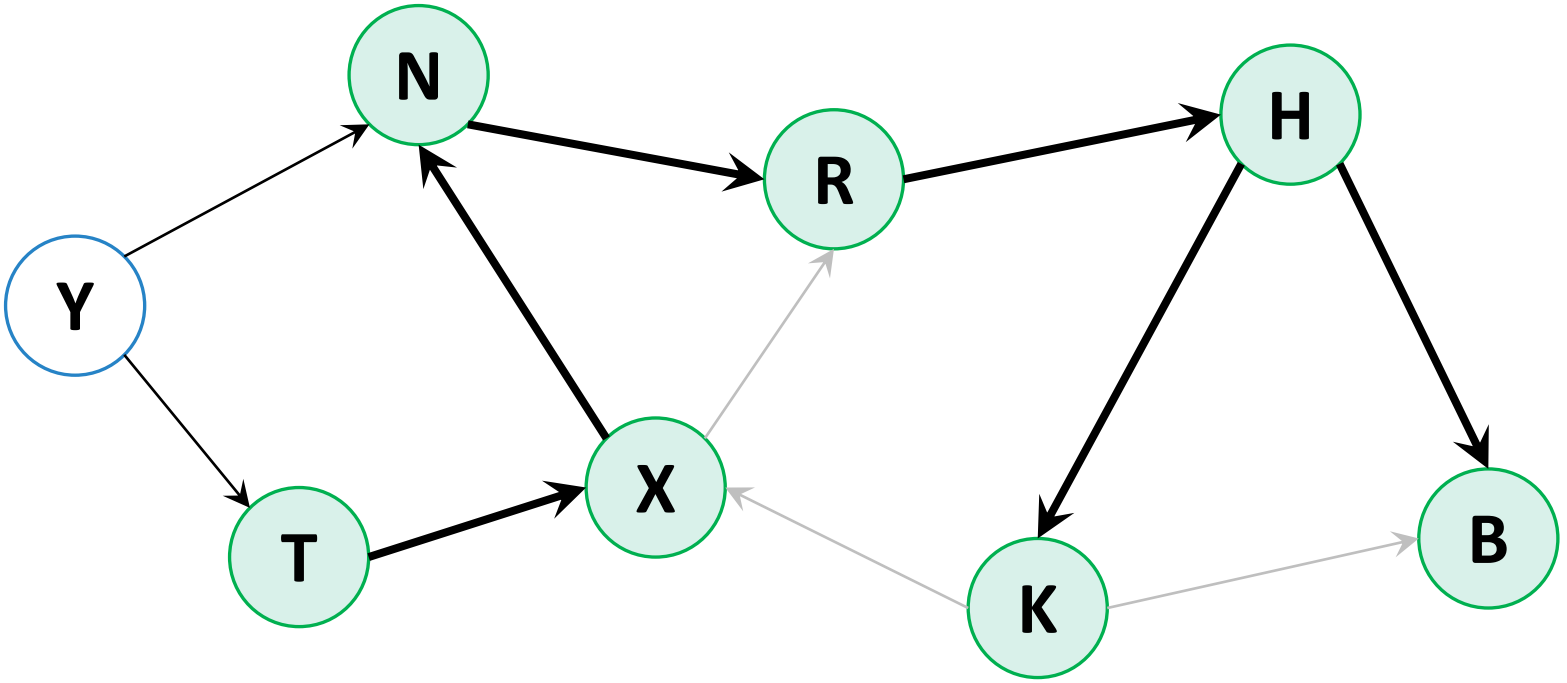
Hay dos aristas que salen de *K*;
tratamos de seguir la que va a *B*, pero
notamos que *B* ya fue visitado … volvemos a *K*

Seguimos buscando nodos posteriores a *K*



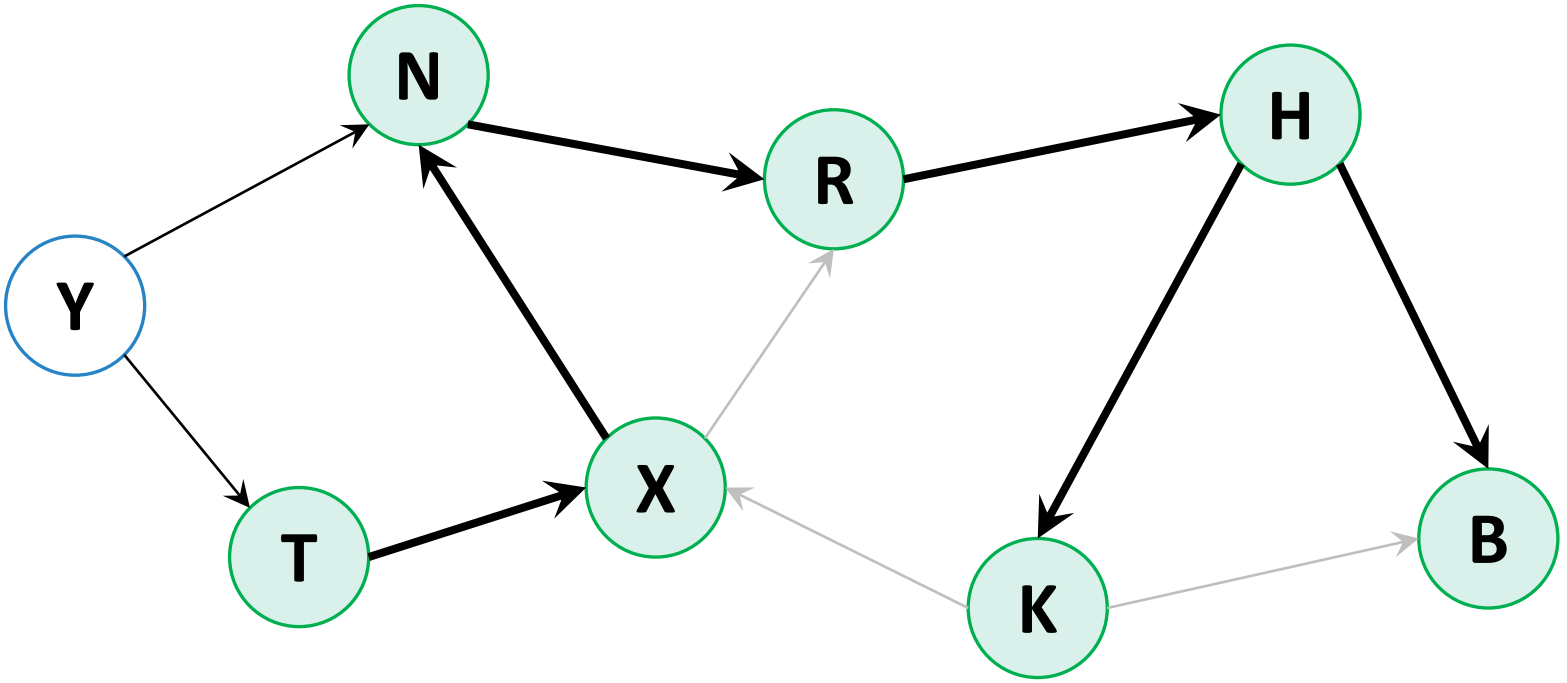
Ahora tratamos de seguir la que va a *X*, pero notamos que *X* ya fue visitado … volvemos a *K*, y de ahí a *H*, a *R*, a *N*, y a *X*

Seguimos buscando nodos posteriores a *X*



En *X,* habíamos dejado pendiente la arista que va a *R*; ahora tratamos de seguirla, pero notamos que *R* ya fue visitado … volvemos a *T*

¡ Listo !



Desde *T* no hay nada más que hacer
… terminamos

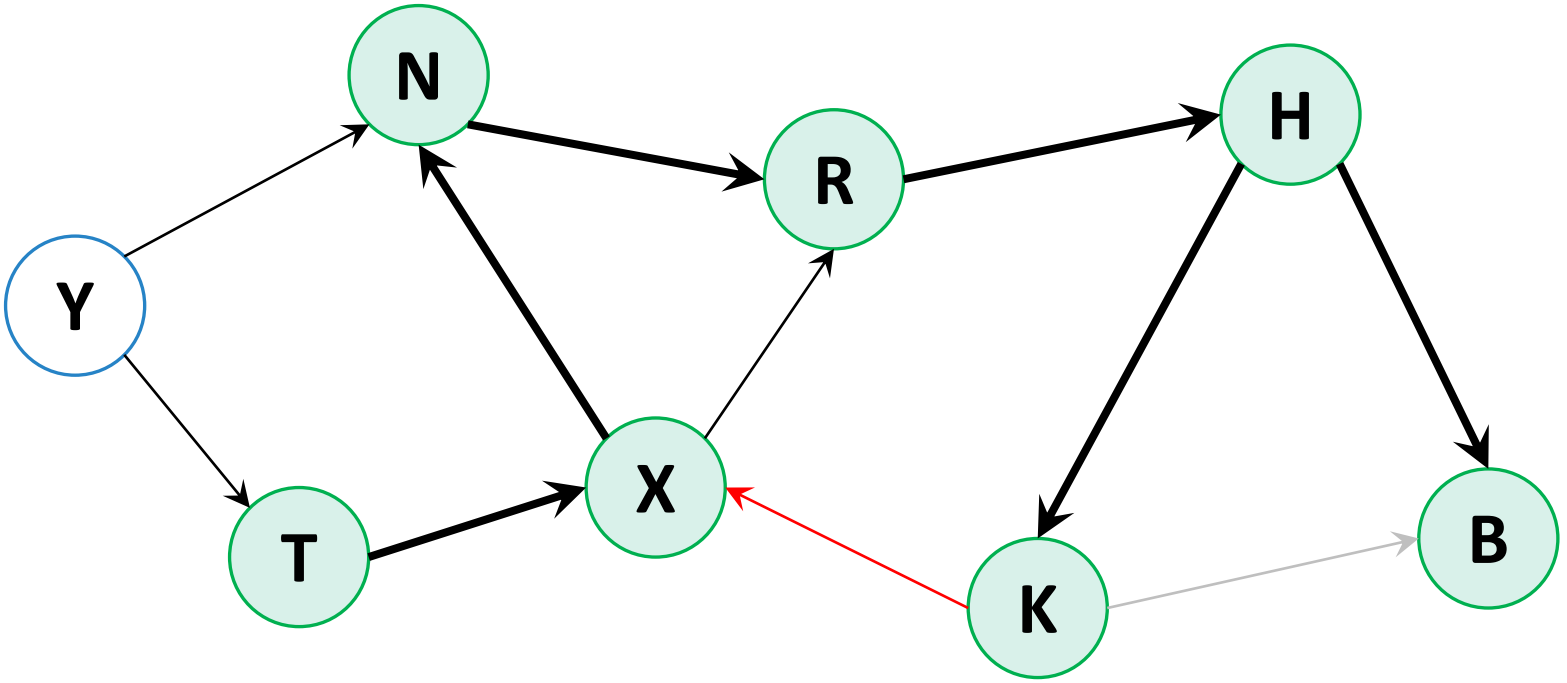
¿Cómo identificamos ciclos?

Ok, podemos identificar las tareas posteriores a una tarea

Ahora, viendo este algoritmo, ¿se nos ocurre algo?

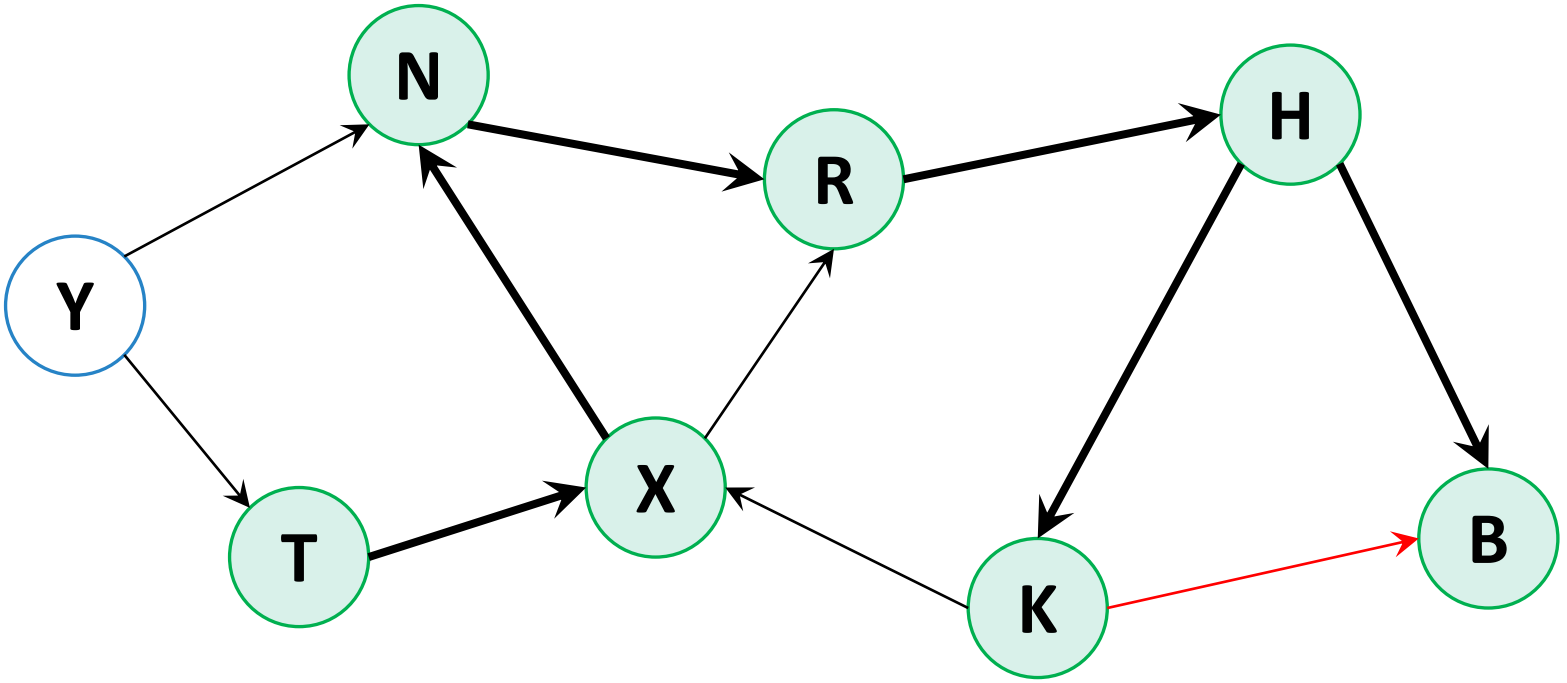
¿Podemos usar este enfoque para identificar ciclos?

Algo así como esto



¡Algoritmo, date cuenta de que esto es un ciclo!

… ¿y como esto?



¡Y que esto otro, no!

Regla de los ciclos

Si el nodo que vamos a visitar, Y, está pintado, entonces (no lo visitamos y) hay dos posibilidades:

- si estamos en un nodo posterior a Y, entonces **hay un ciclo**
- si no estamos en un nodo posterior a Y, entonces no hay un ciclo (al menos por ahora)

Hasta que $\boldsymbol{posteriores}(X)$ retorne, todos los nodos visitados (que van siendo pintados) son posteriores a X

hay ciclo luego de (X):

if X está pintado de gris: ***return true***

if X está pintado de negro: ***return false***

Pintar X de gris

for Y tal que $X \rightarrow Y$:

if hay ciclo luego de (Y), ***return true***

Pintar X de negro

return false

hay ciclo en $\big(\boldsymbol{G(V,E)}\big)$**:**

for $\boldsymbol{X} \in \boldsymbol{V}$:

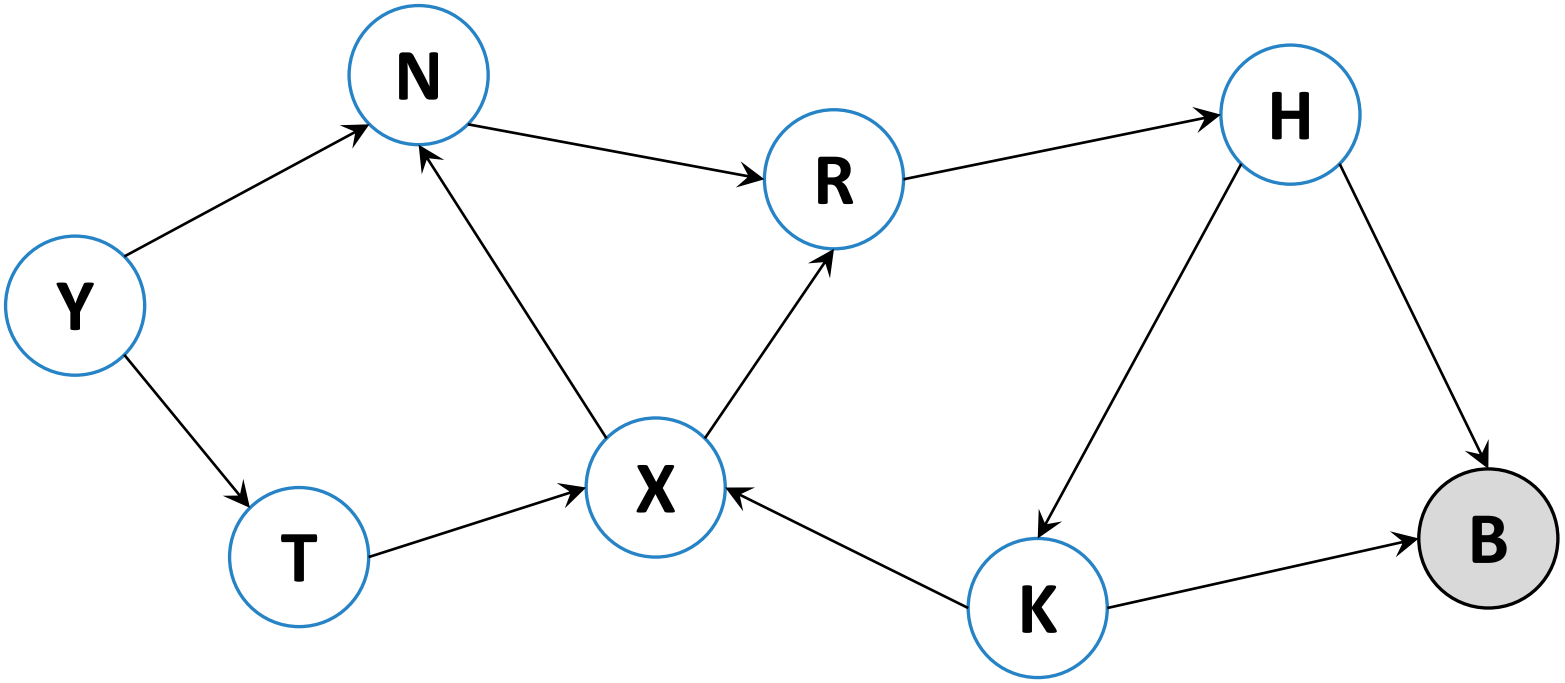
if $\boldsymbol{X}$ está pintado, ***continue***

if hay ciclo luego de $(\boldsymbol{X})$:

return true

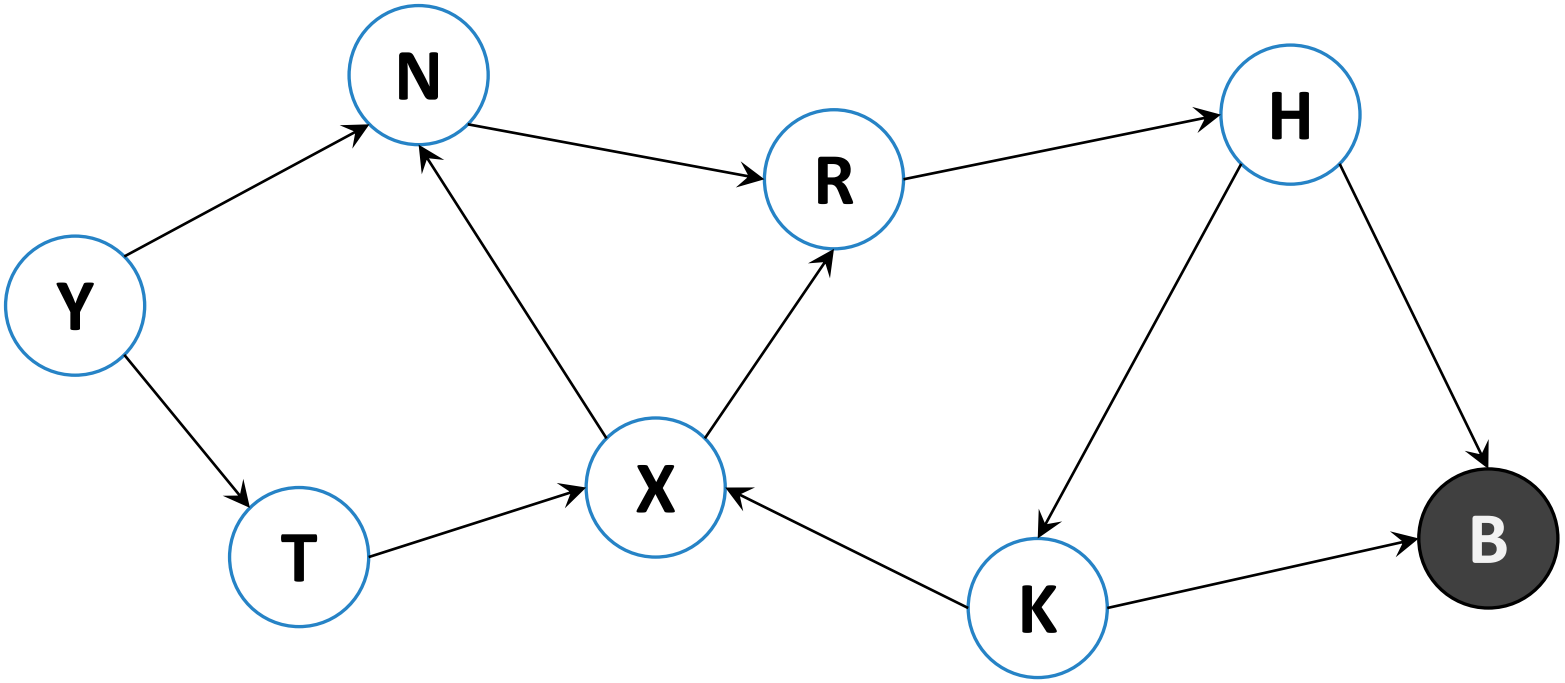
return false

El algoritmo *hay ciclo en* en acción



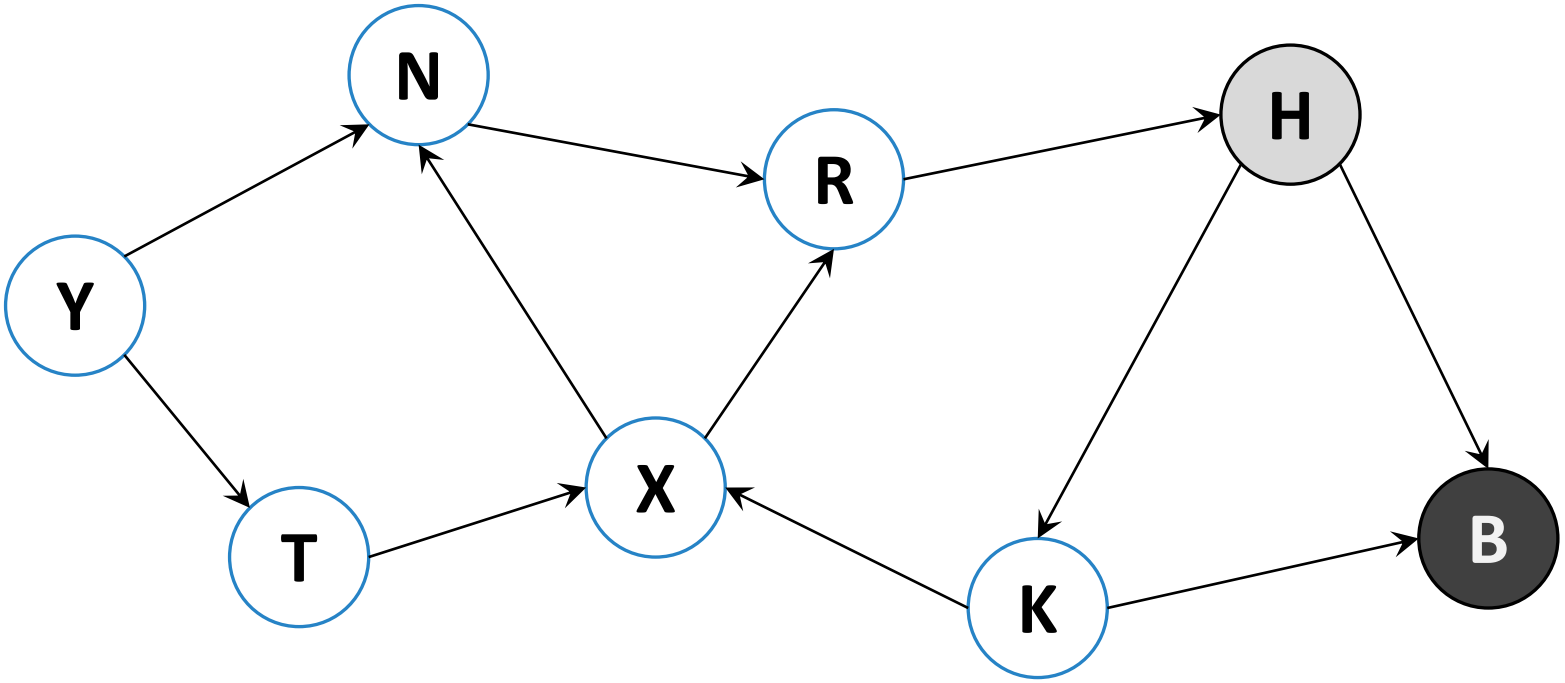
¿Hay un ciclo luego de **B**?

El algoritmo en acción



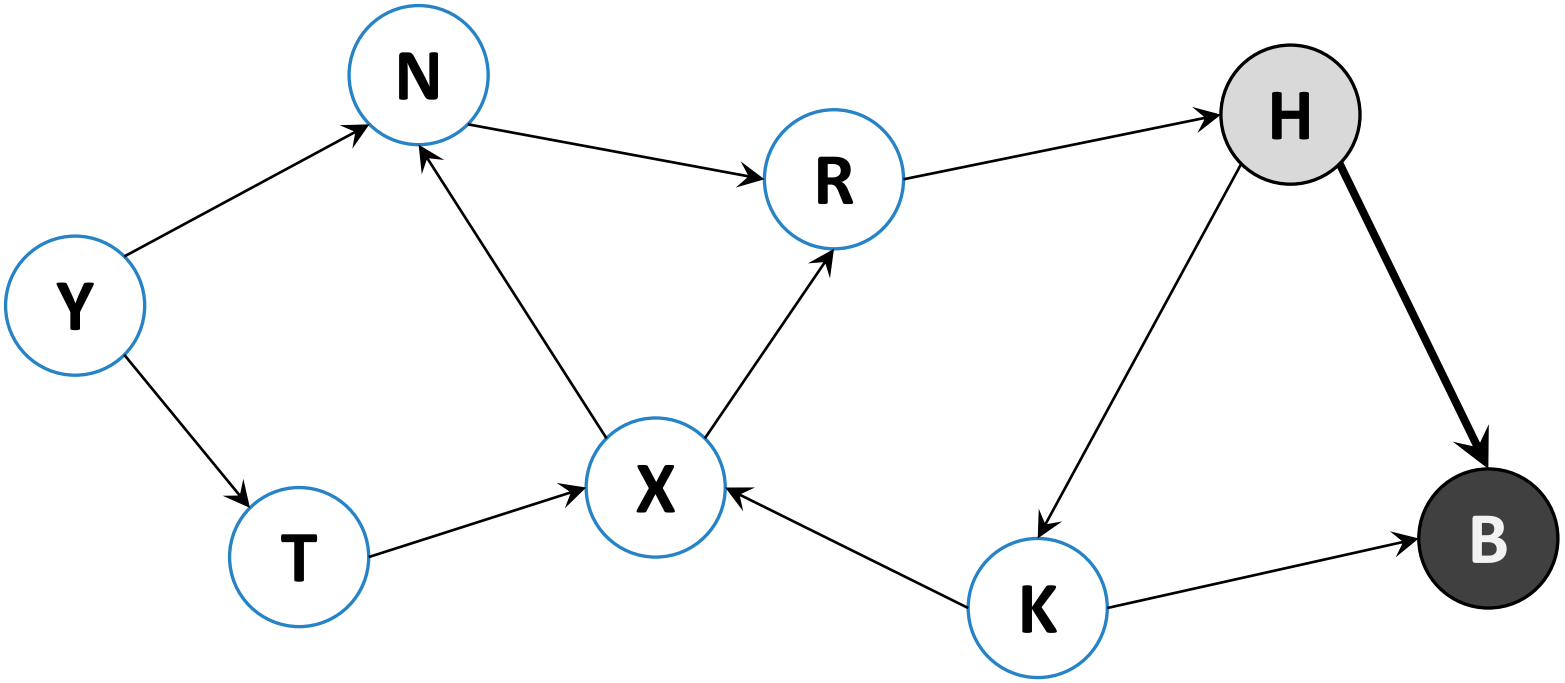
No

El algoritmo en acción



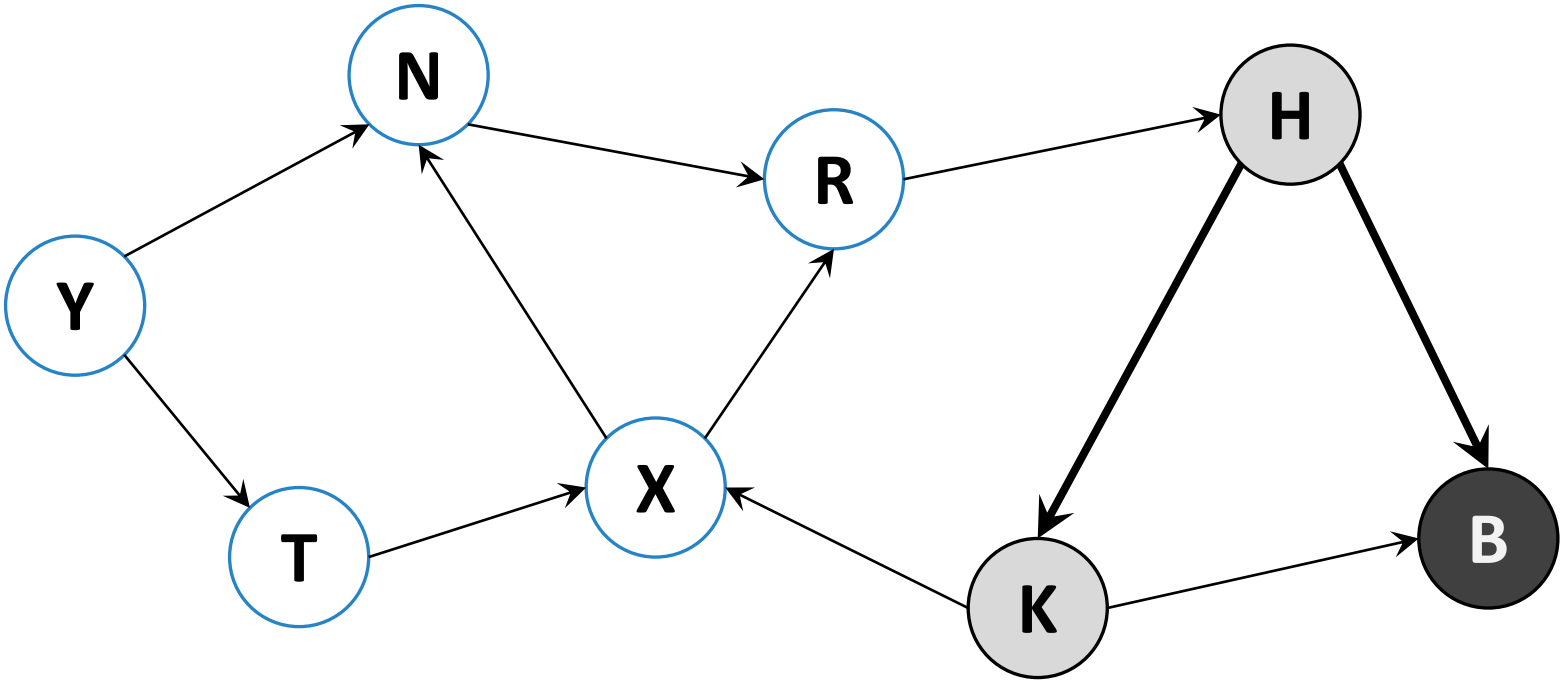
OK, ¿hay un ciclo luego de **H**?

El algoritmo en acción



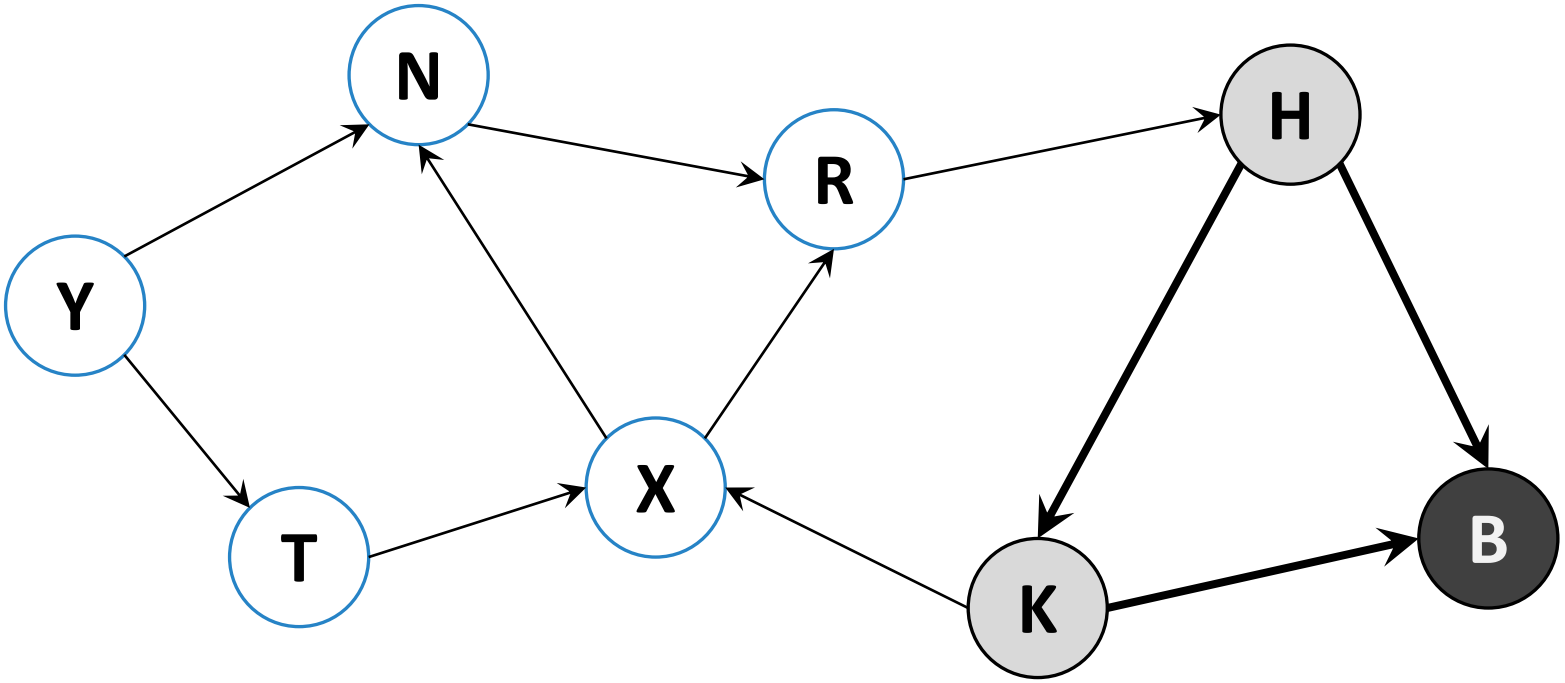
¿Hay un ciclo luego de **H**?

El algoritmo en acción



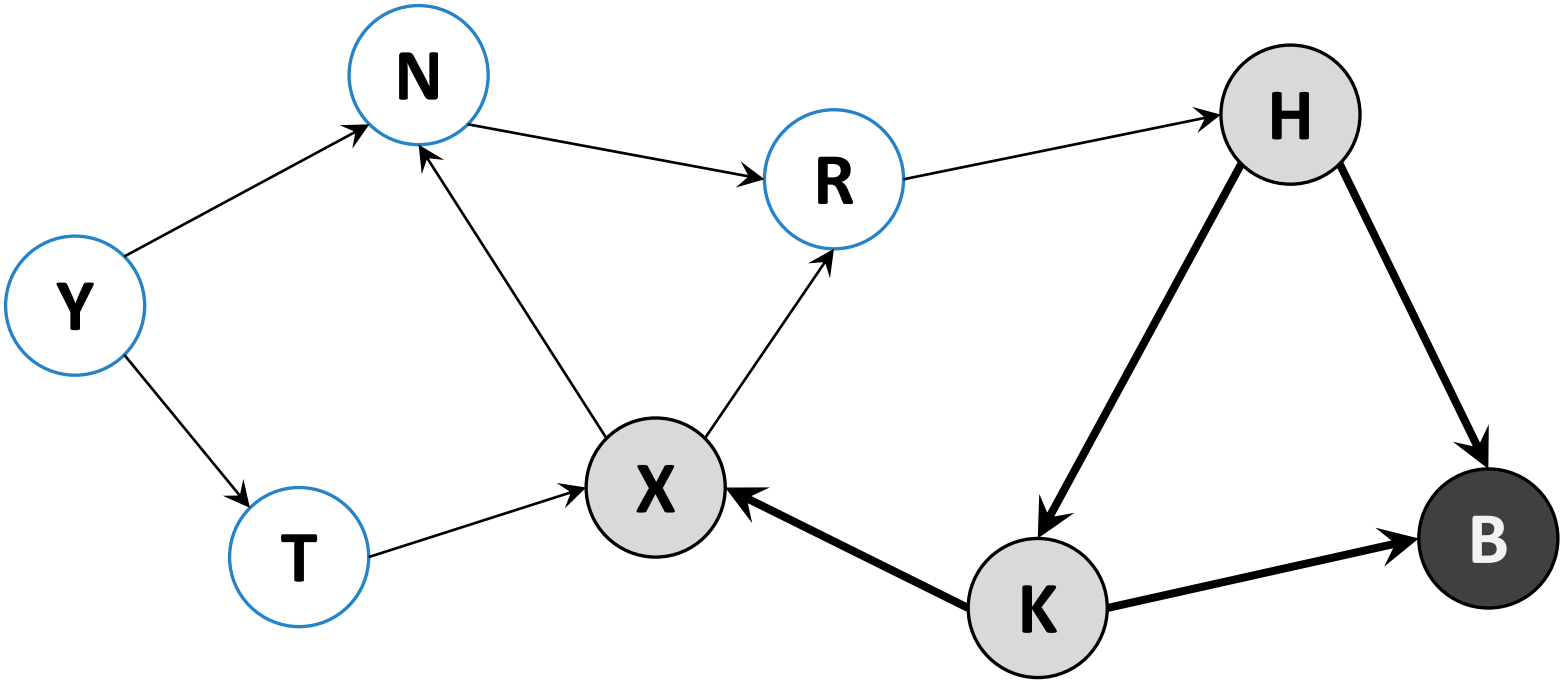
¿Hay un ciclo luego de **K**?

El algoritmo en acción



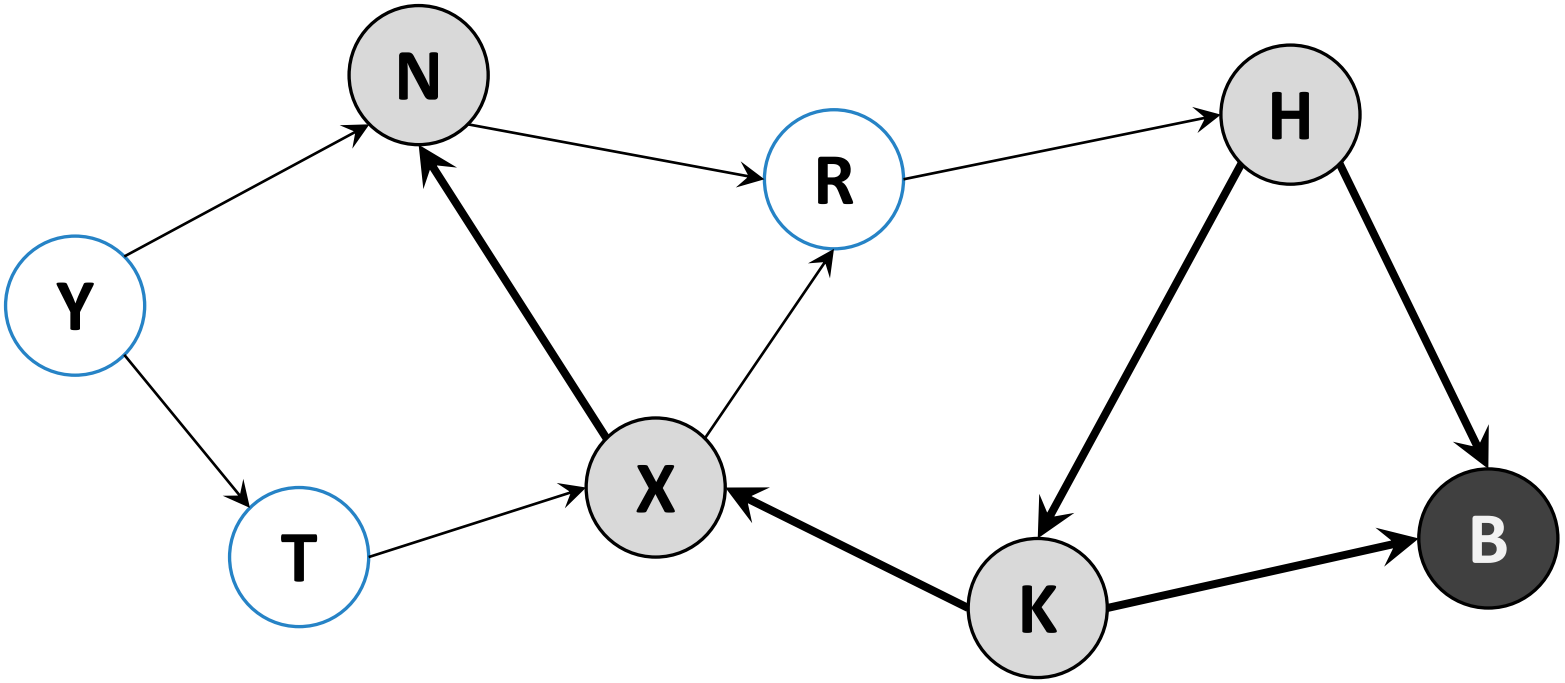
¿Hay un ciclo luego de **K**?

El algoritmo en acción



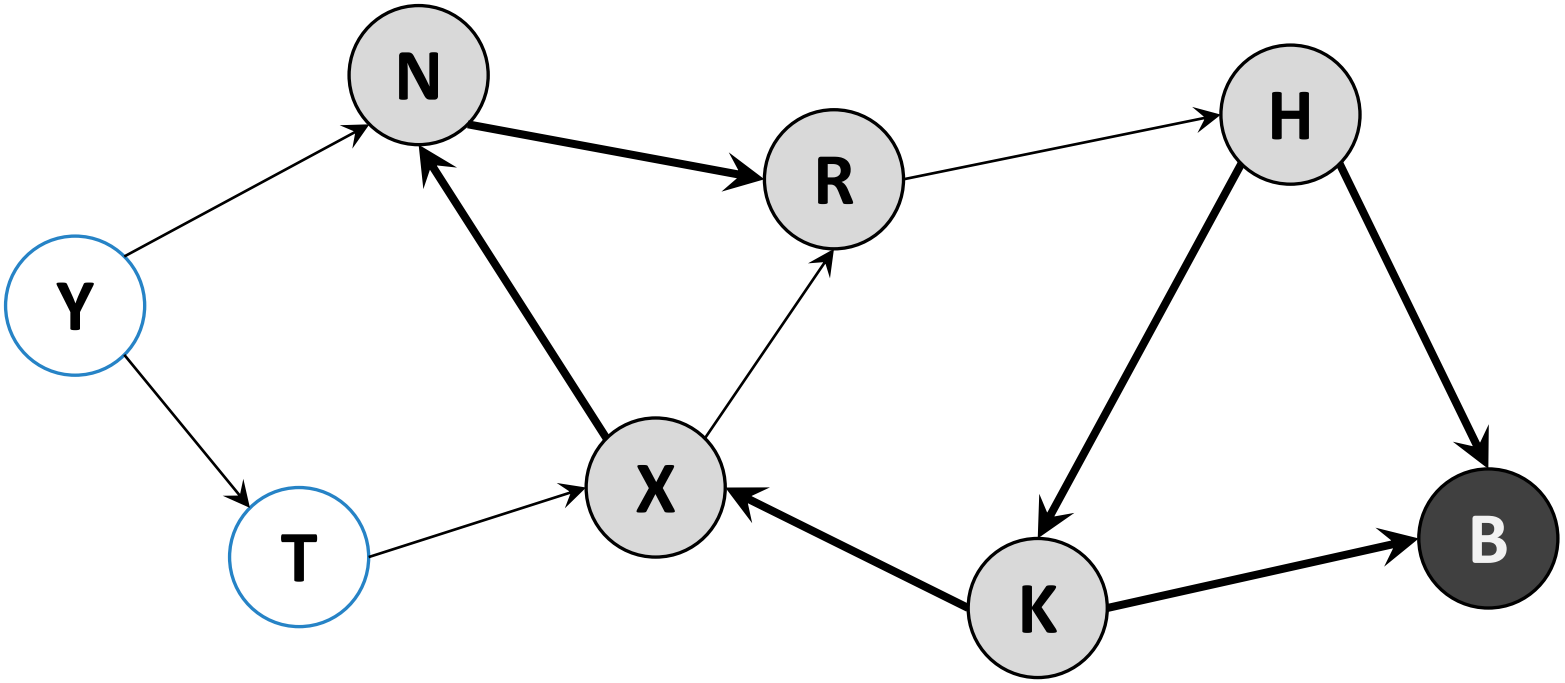
¿Hay un ciclo luego de **X**?

El algoritmo en acción



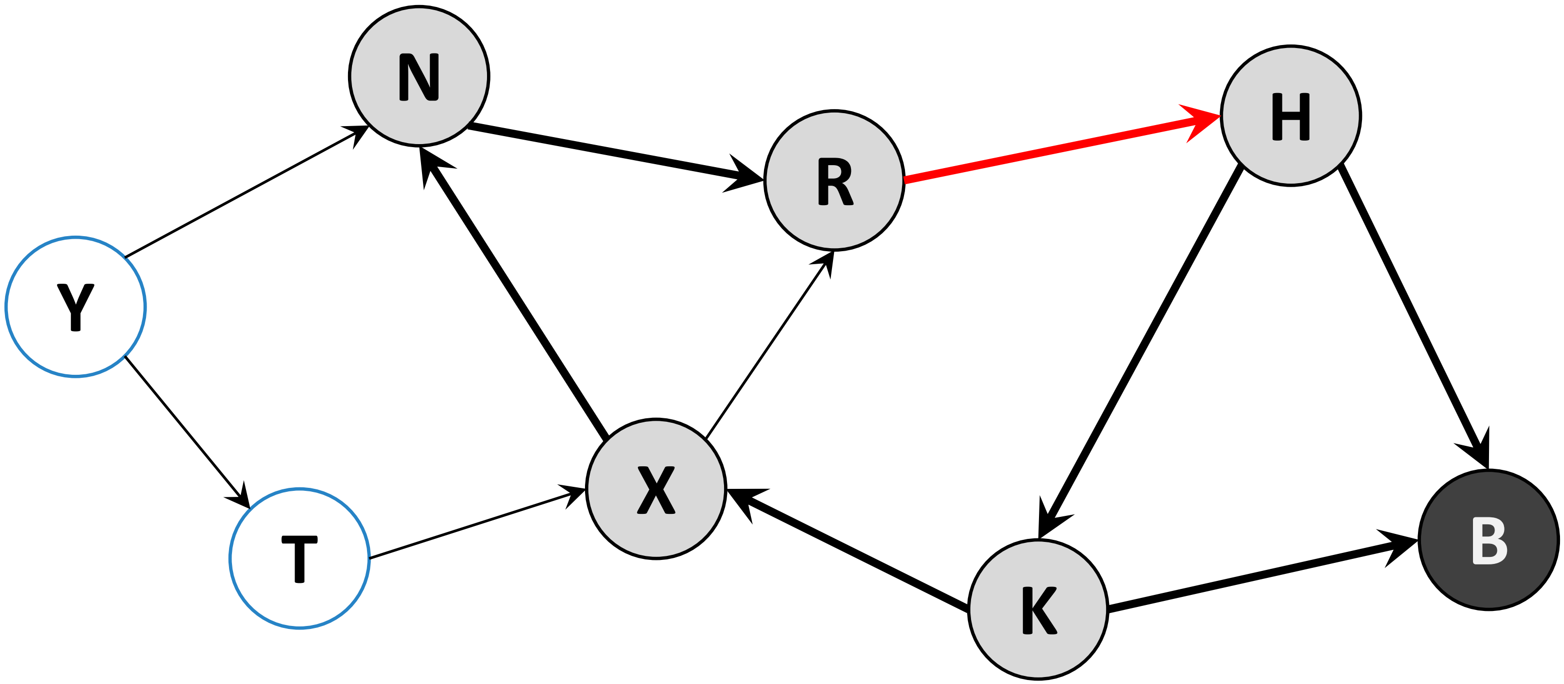
¿Hay un ciclo luego de **N**?

El algoritmo en acción



¿Hay un ciclo luego de **R**?

El algoritmo en acción



!!!

Algortimos de este tipo se llaman de “búsqueda en profundidad”

Los algoritmos de **búsqueda en profundidad** (o ***DFS***) llegan hasta el final de una “rama” (una secuencia de aristas) antes de empezar a explorar otra

¿Cuál es la complejidad de estos algoritmos?

El algoritmo DFS: *depth-first search* o exploración (primero) en profundidad

Las aristas son exploradas a partir del vértice *v* descubierto más recientemente

… que aún tiene aristas no exploradas que salen de él:

- cuando todas las aristas de *v* han sido exploradas, la exploración retrocede para explorar aristas que salen del vértice a partir del cual *v* fue descubierto
- busca más profundamente en *G* mientras sea posible

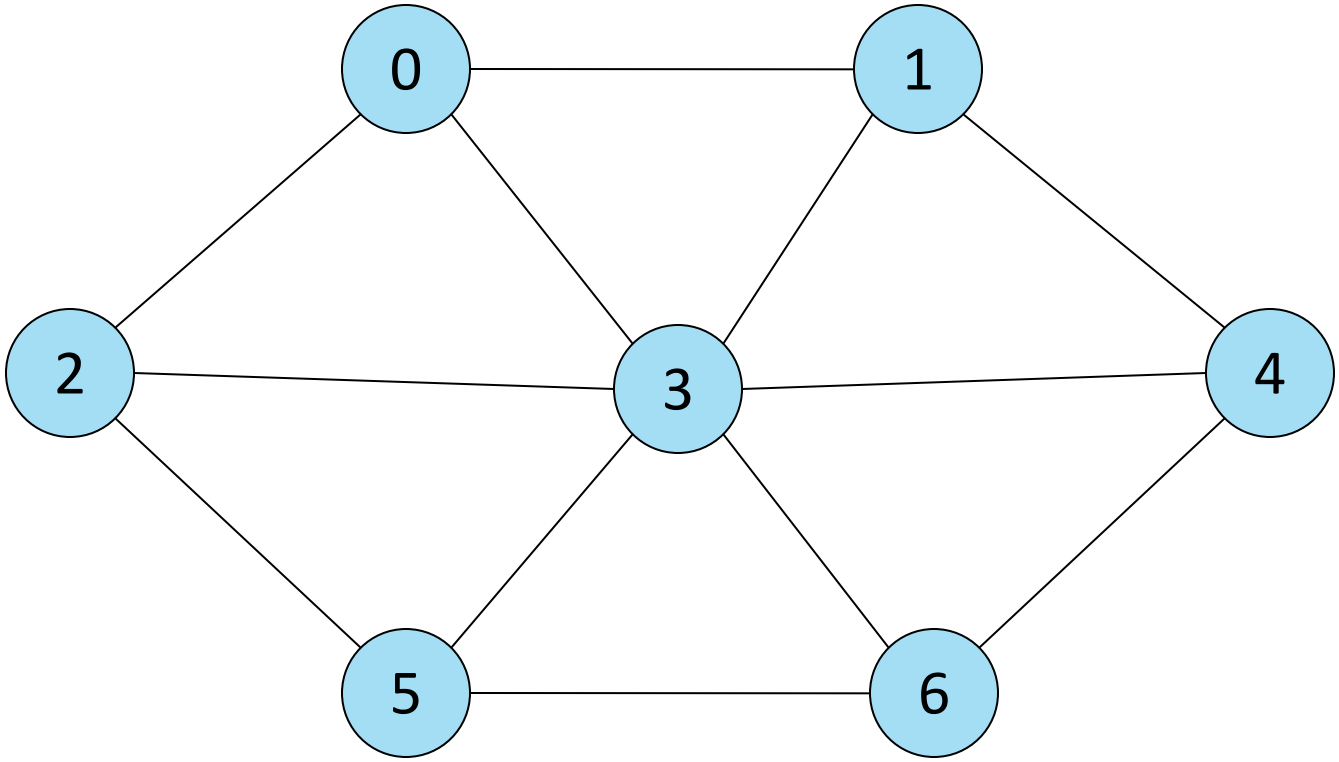
DFS pinta los vertices blancos, grises o negros

Todos los vértices son inicialmente *blancos*

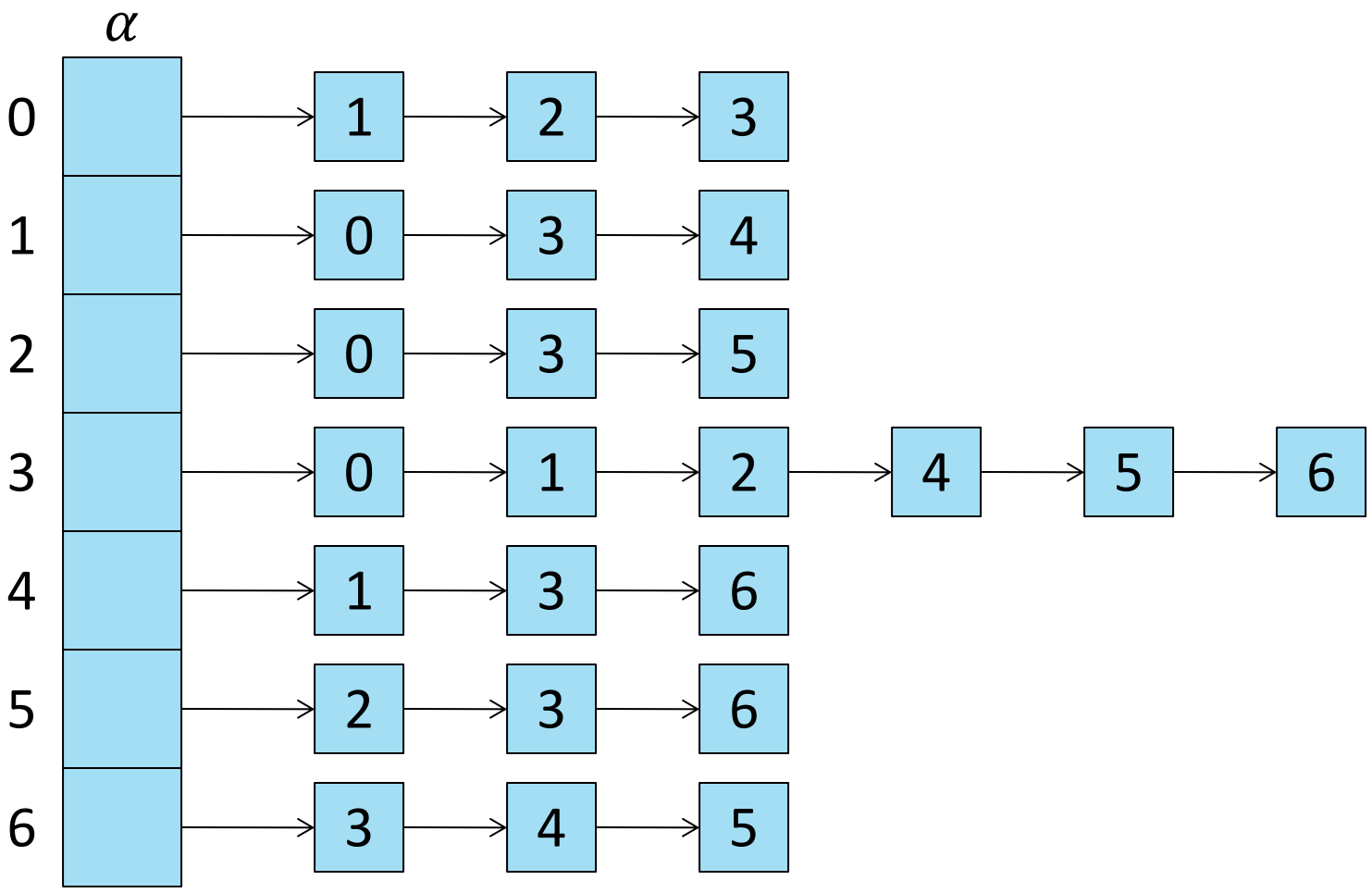
Un vértice se pinta de *gris* cuando es descubierto

Un vértice se pinta de *negro* cuando su lista de adyacencias ha sido examinada exhaustivamente

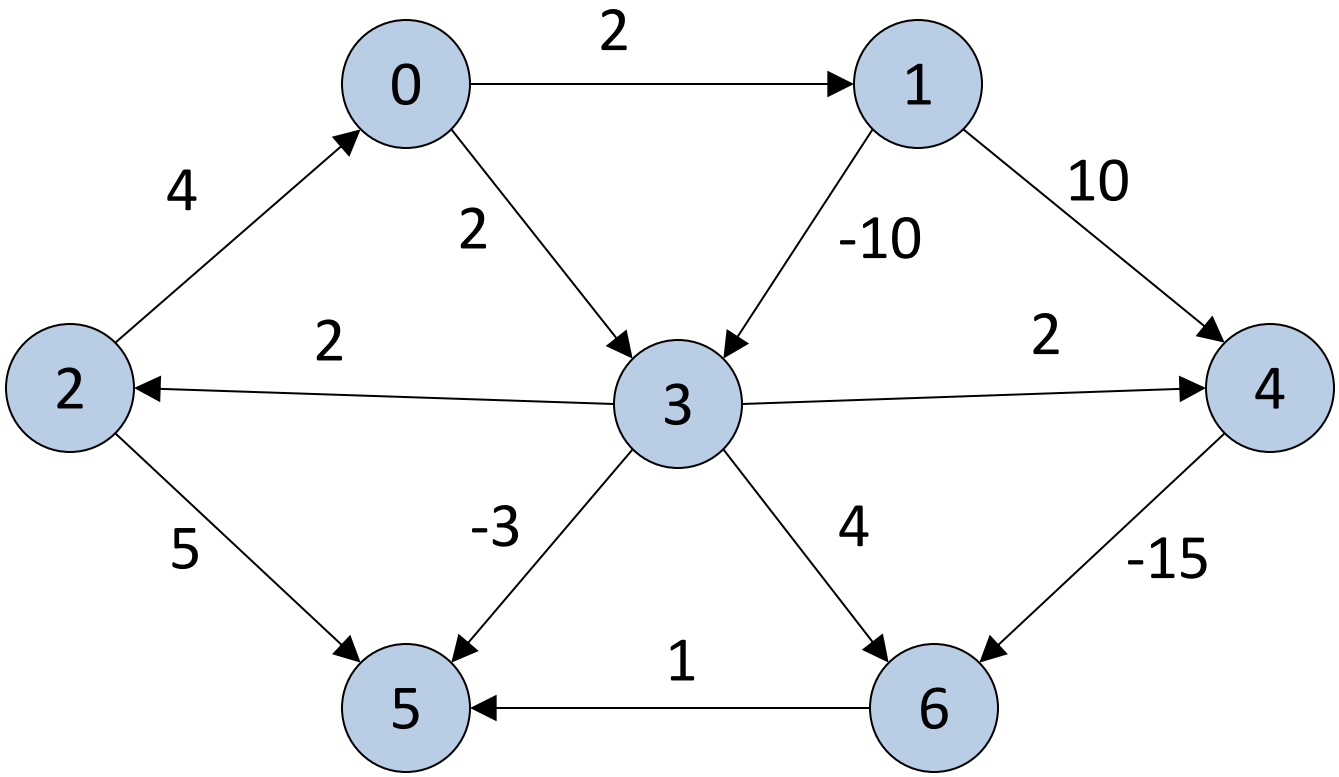
Un grafo no direccional sin costos



El grafo anterior representado por $|V|$ listas de adyacencias, α



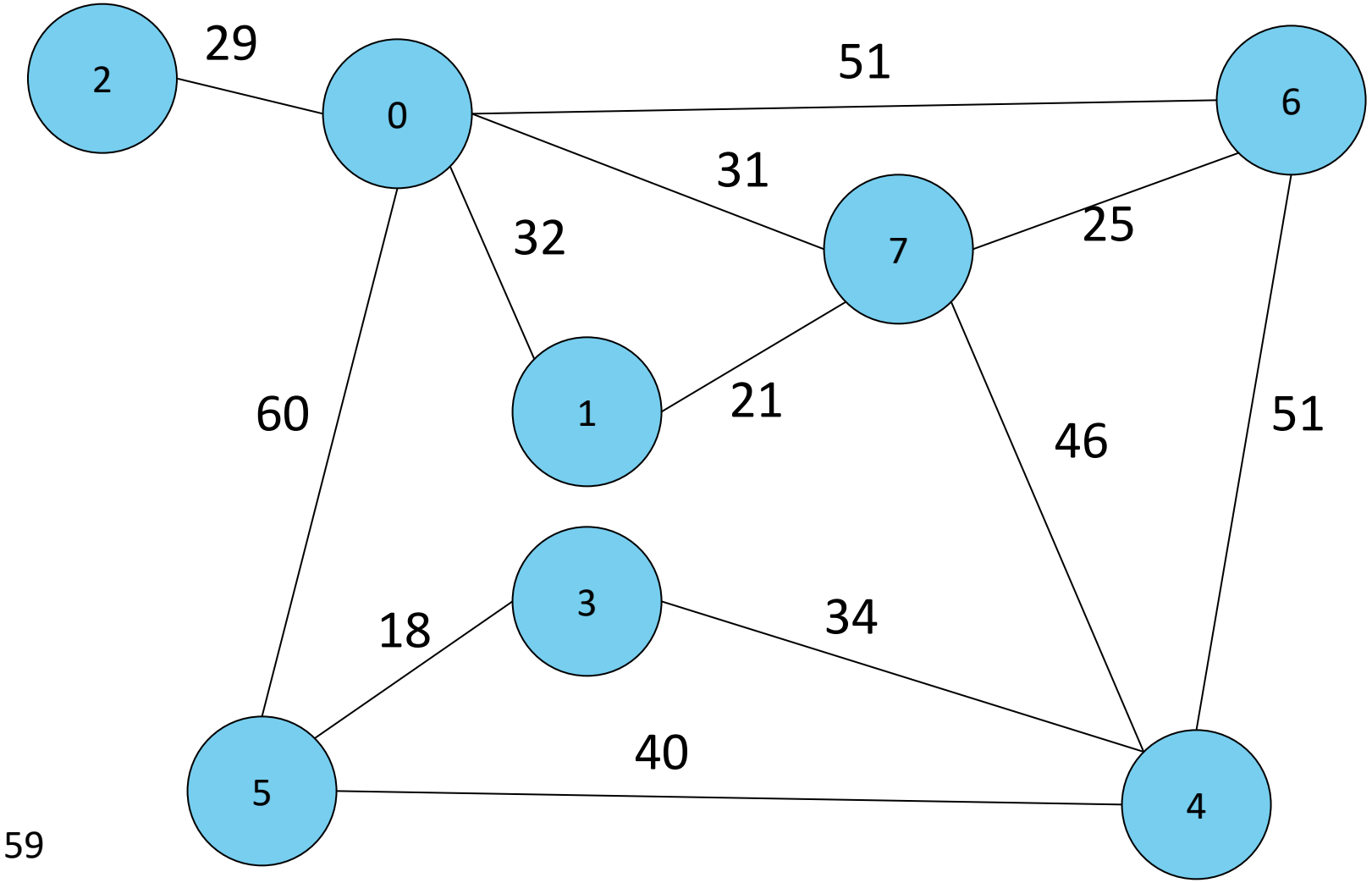
Un grafo direccional con costos



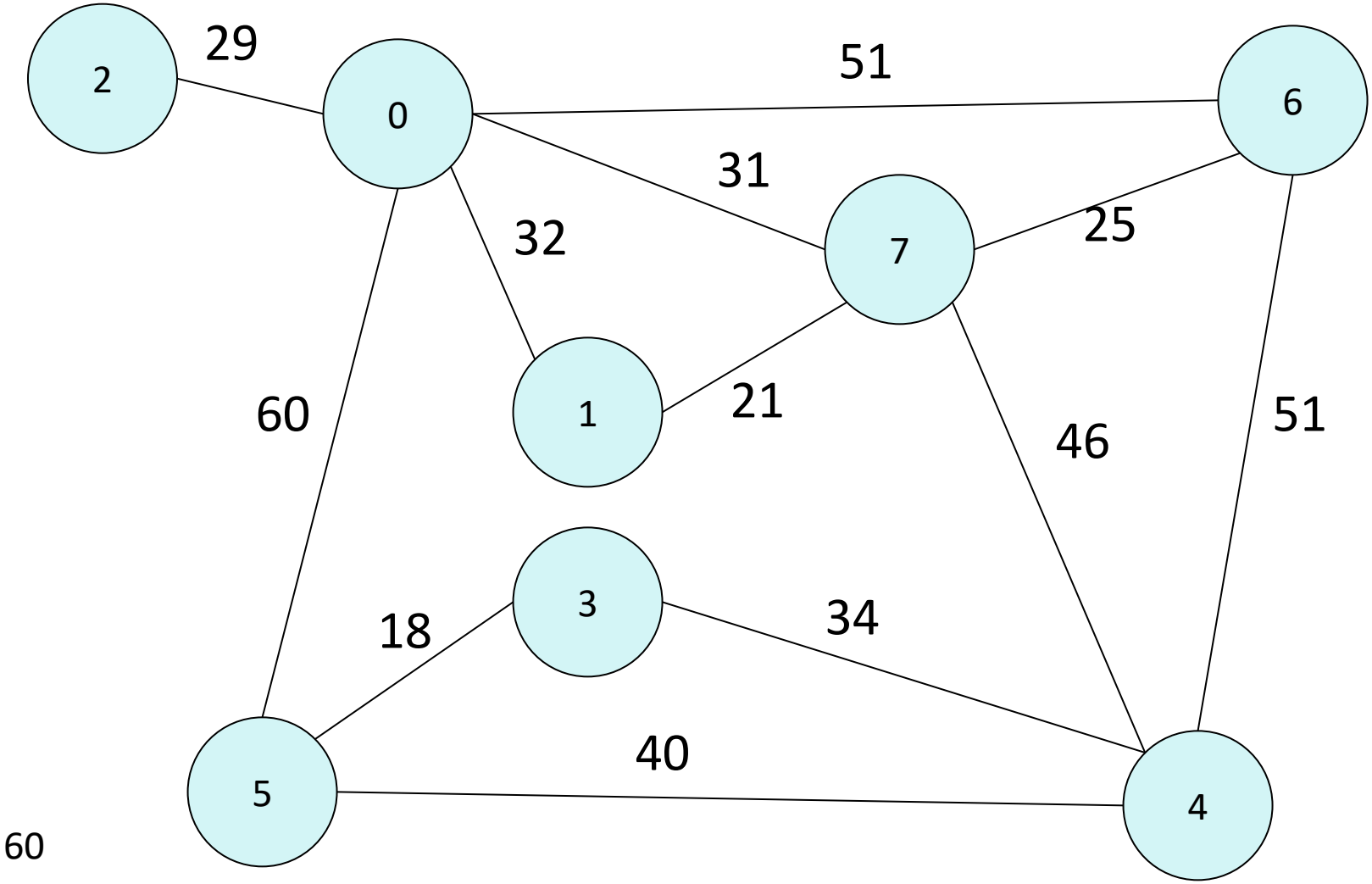
El grafo anterior representado con una matriz de adyacencia

	0	1	2	3	4	5	6
0		2		2			
1				-10	10		
2	4					5	
3			2		2	-3	4
4							-15
5							
6						1	

Un grafo no direccional con costos



Un grafo no direccional con costos



El grafo anterior representadopor $|V|$ listas de adyacencias, α