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?

Inconsistencia

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

$$A \to B$$

Si existe alguna secuencia inconsistente de requisitos, como

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

entonces no es posible realizar el proyecto. ¿Es la única condición?

En un 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?

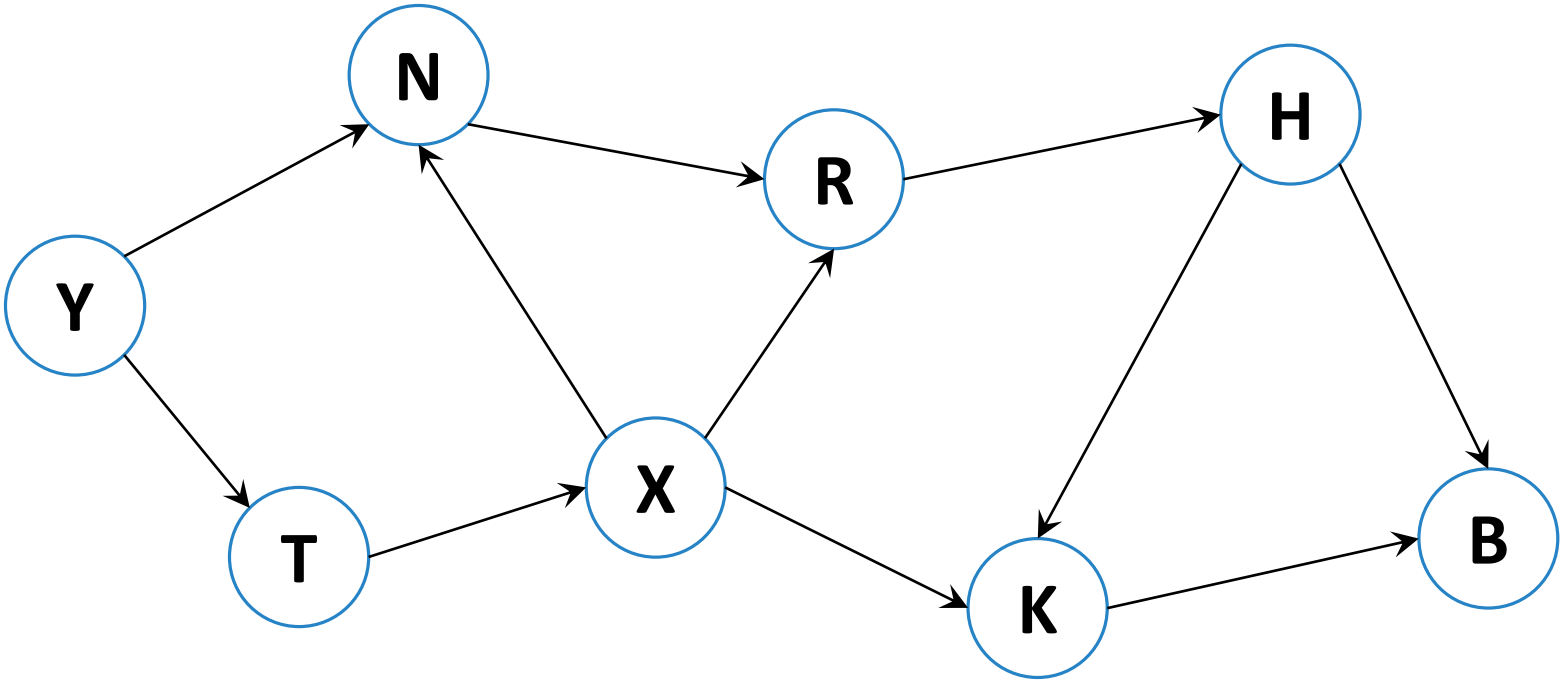
Grafo

Un **grafo** es un conjunto de **nodos** y **aristas** que los unen

Es simplemente una forma de **representar** un problema

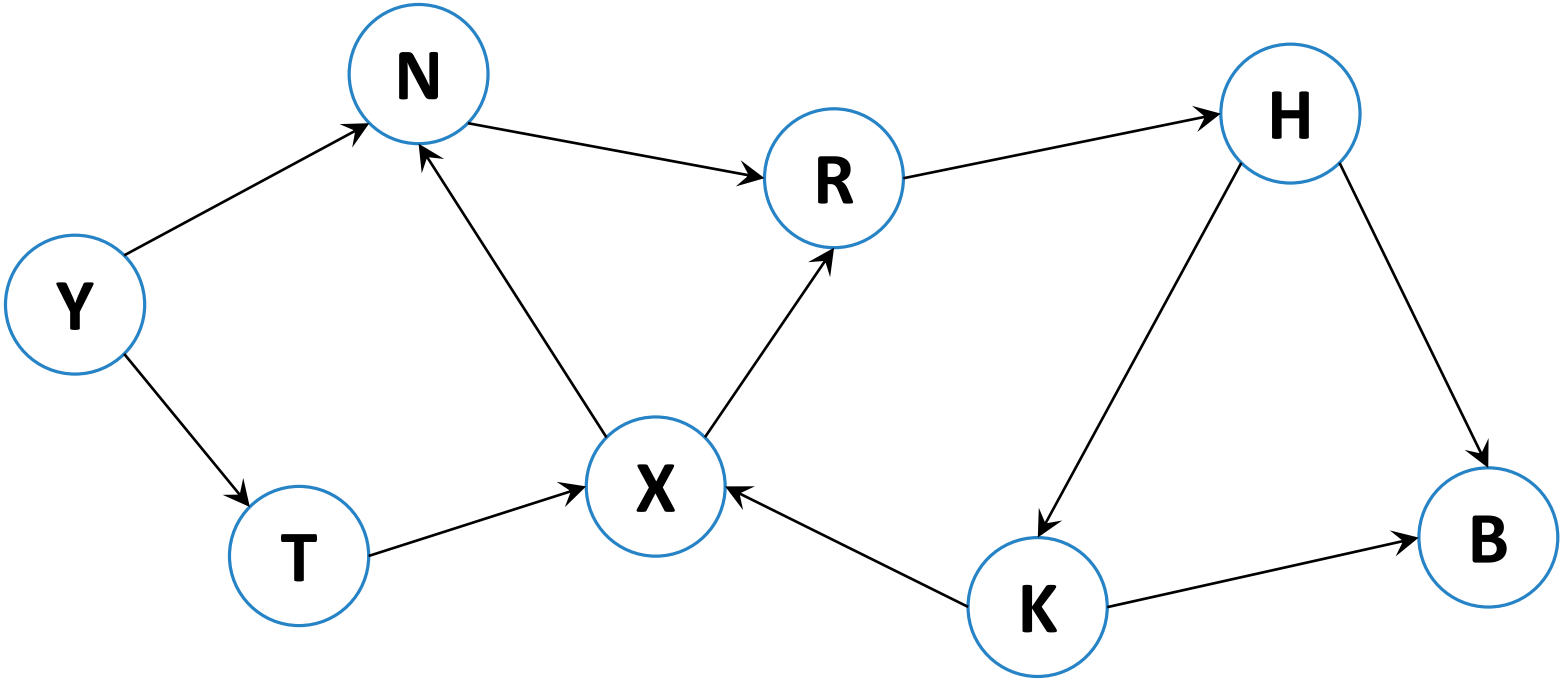
¿Cómo podríamos plantear el grafo de este problema?

El grafo de un problema



¿Algún problema con este proceso?

El grafo de otro problema



¿Y con este?

Ciclos

Si el grafo tiene un **ciclo**, entonces el proceso no es posible

¿Cómo podemos buscar ciclos de manera eficiente?

La relación *posterior a*

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

$$X \rightarrow Y$$

ó

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

Significa que X debe realizarse antes que Y

Posteriores y ciclos

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

¿Cómo podemos identificar las tareas posteriores a una tarea?

Pensemos en la definición de la propiedad

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

$\boldsymbol{P} \leftarrow \emptyset$

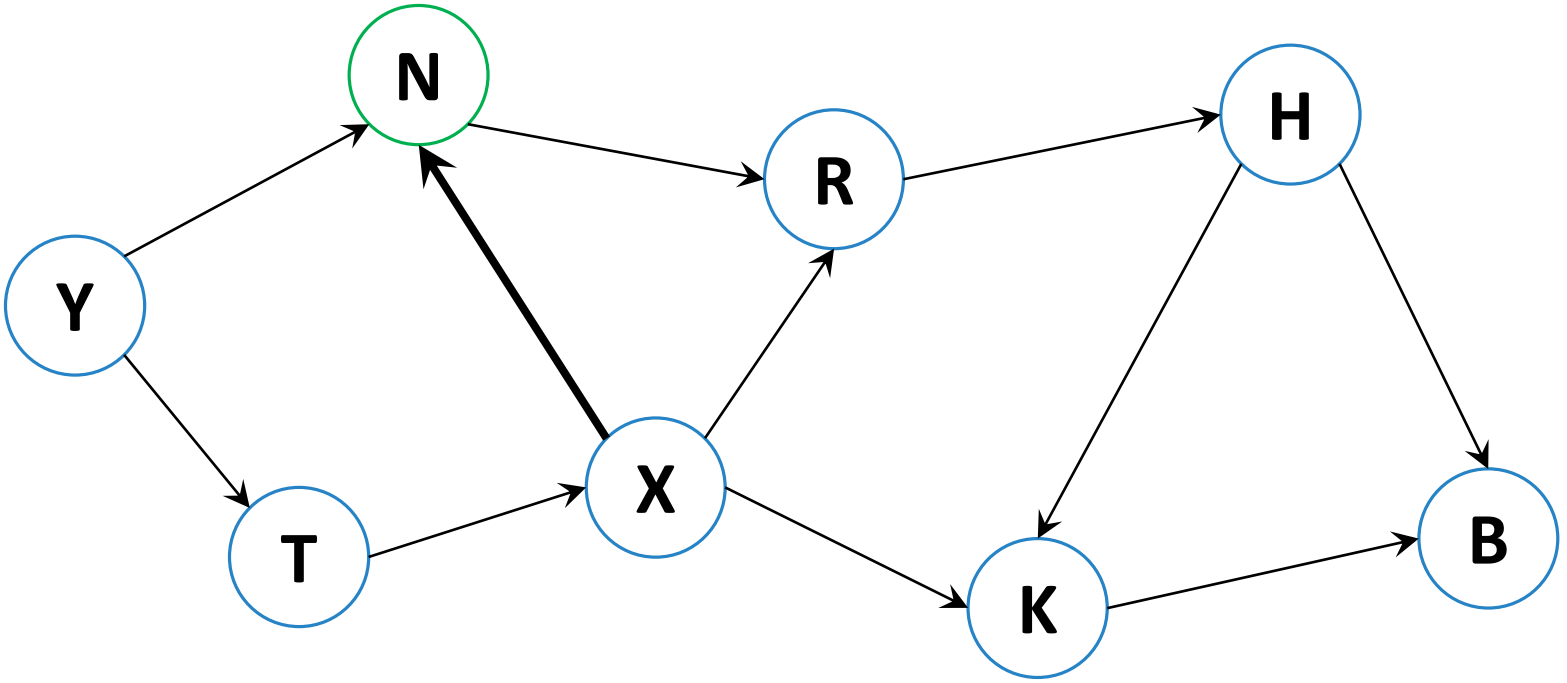
for $\boldsymbol{Y}$ tal que $\boldsymbol{X} \rightarrow \boldsymbol{Y}$:

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

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

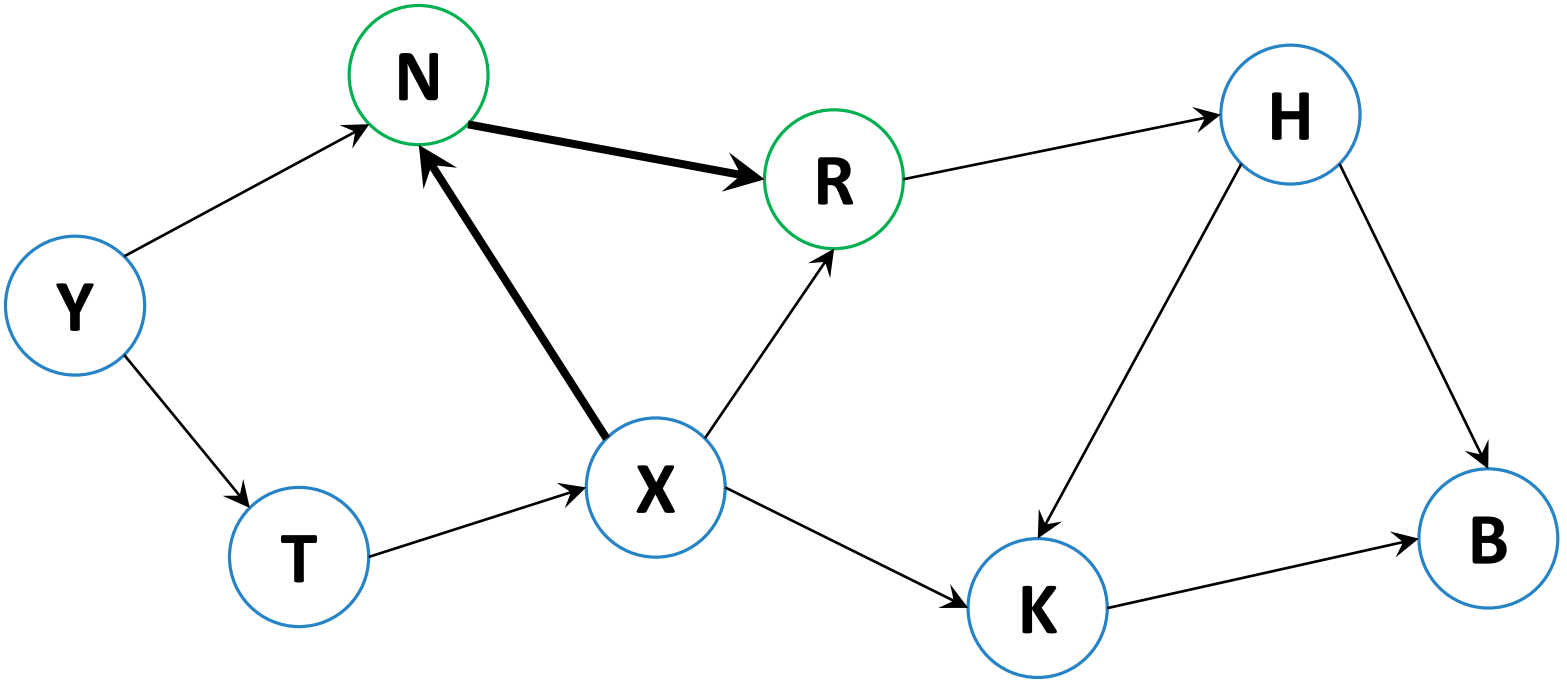
return $\boldsymbol{P}$

Posteriores



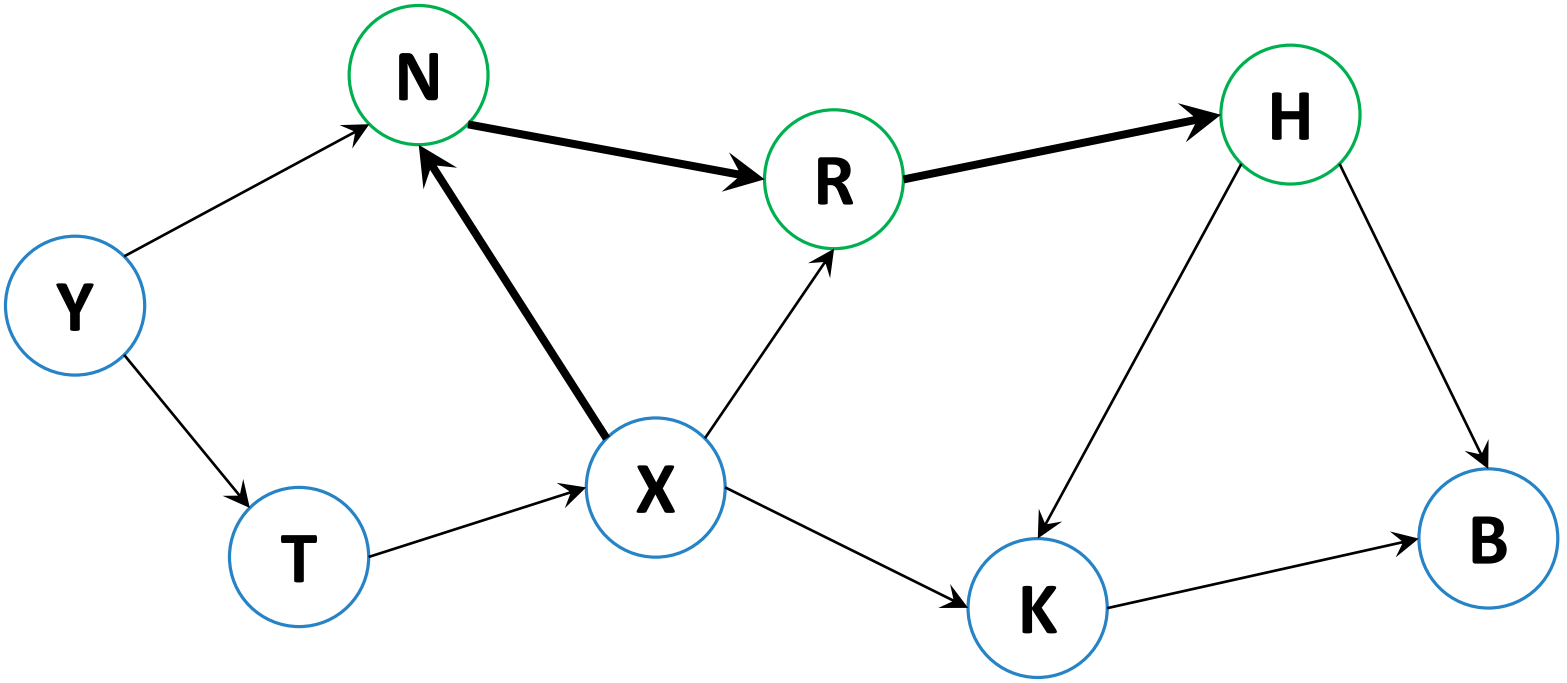
¿Cuáles nodos son posteriores a **X**?

Posteriores



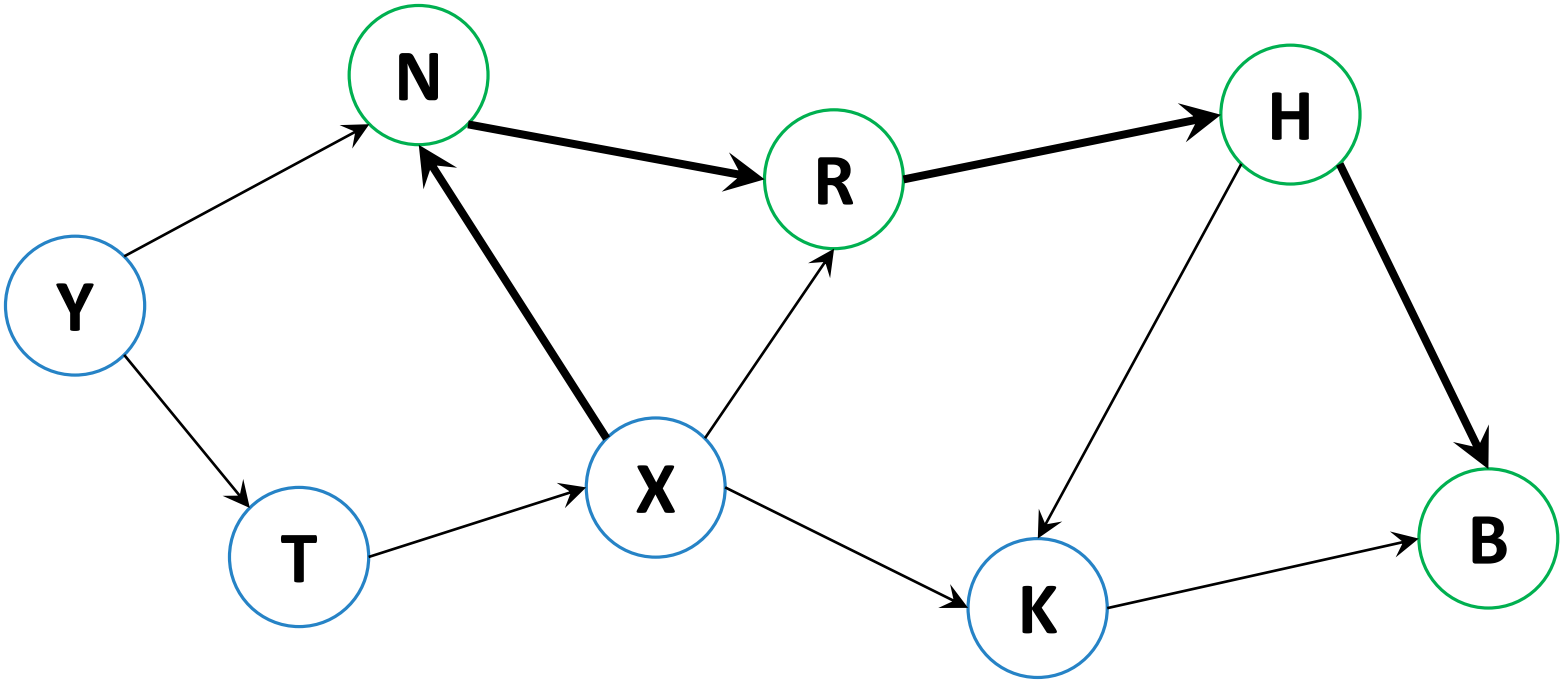
¿Cuáles nodos son posteriores a **N**?

Posteriores



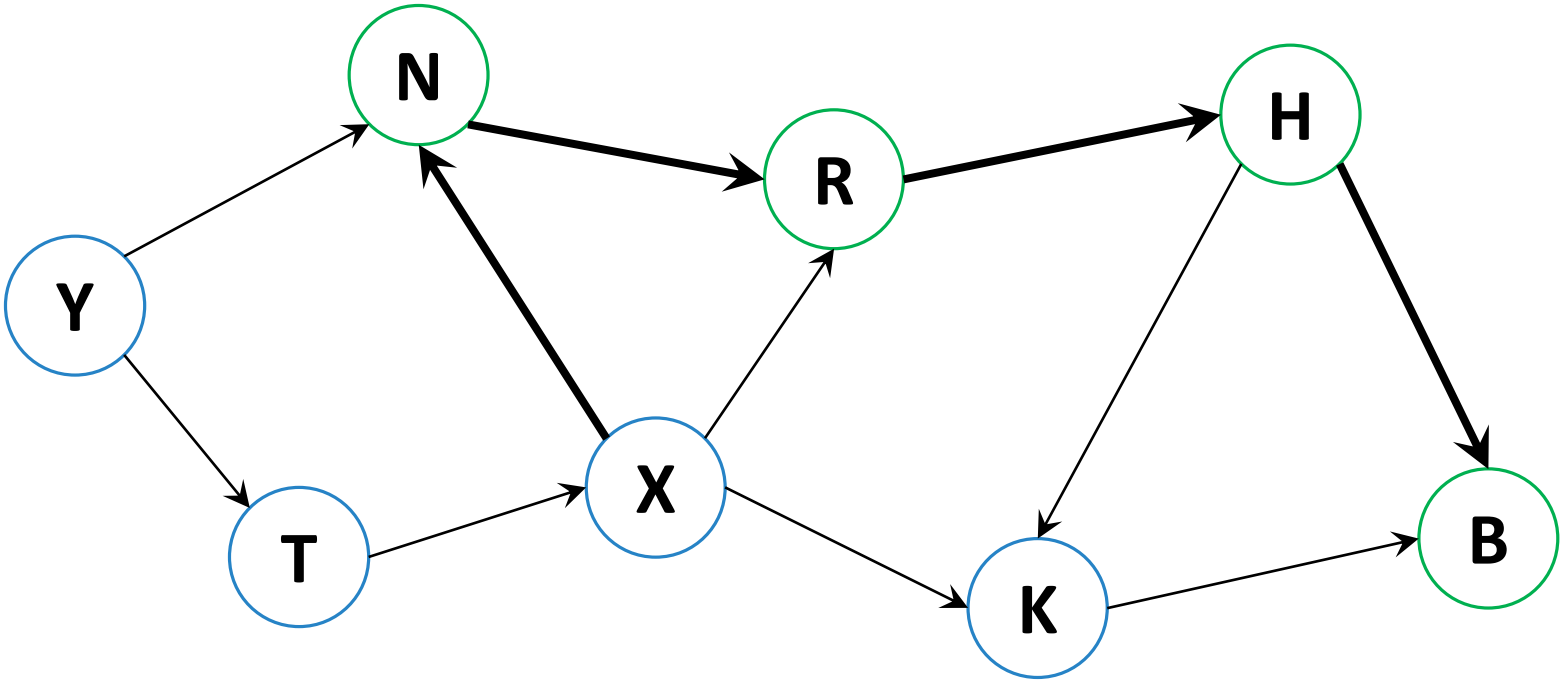
¿Cuáles nodos son posteriores a **R**?

Posteriores



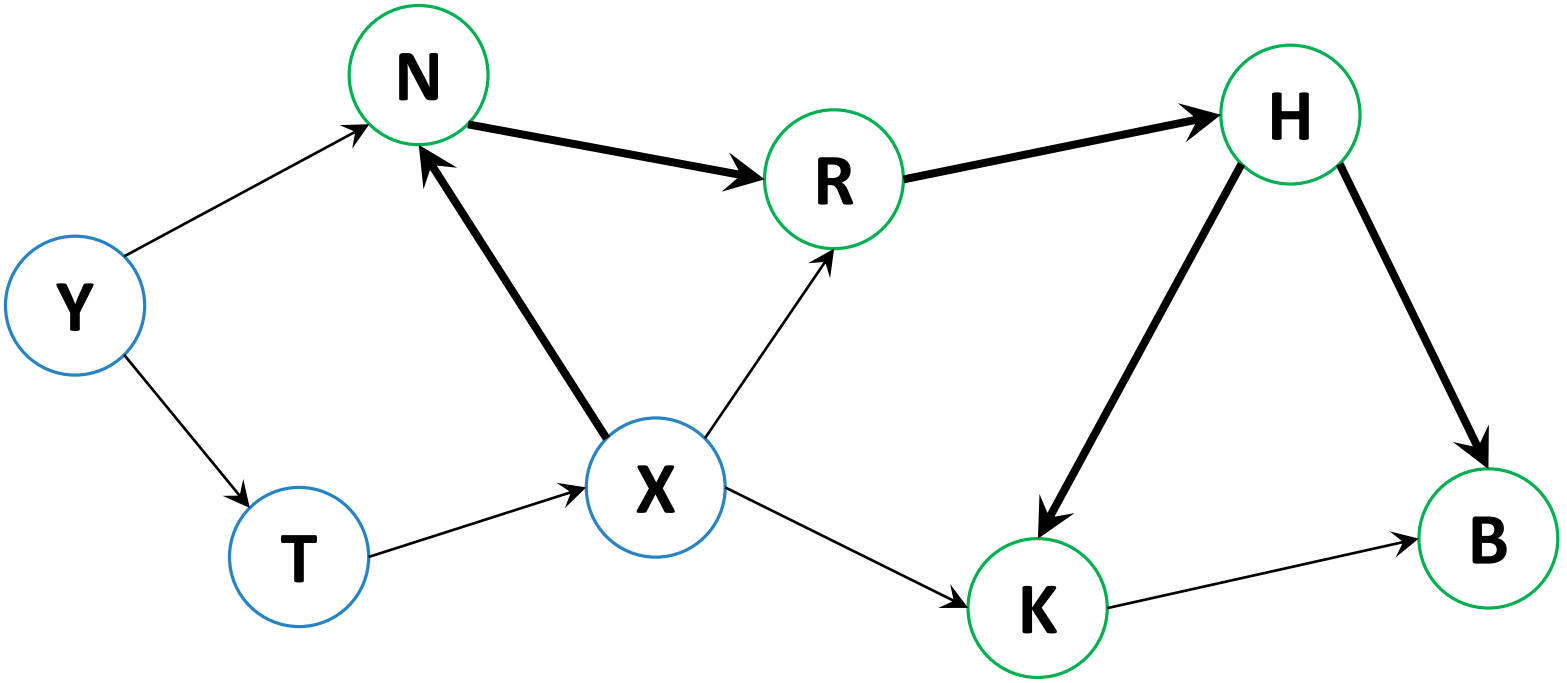
¿Cuáles nodos son posteriores a **H**?

Posteriores



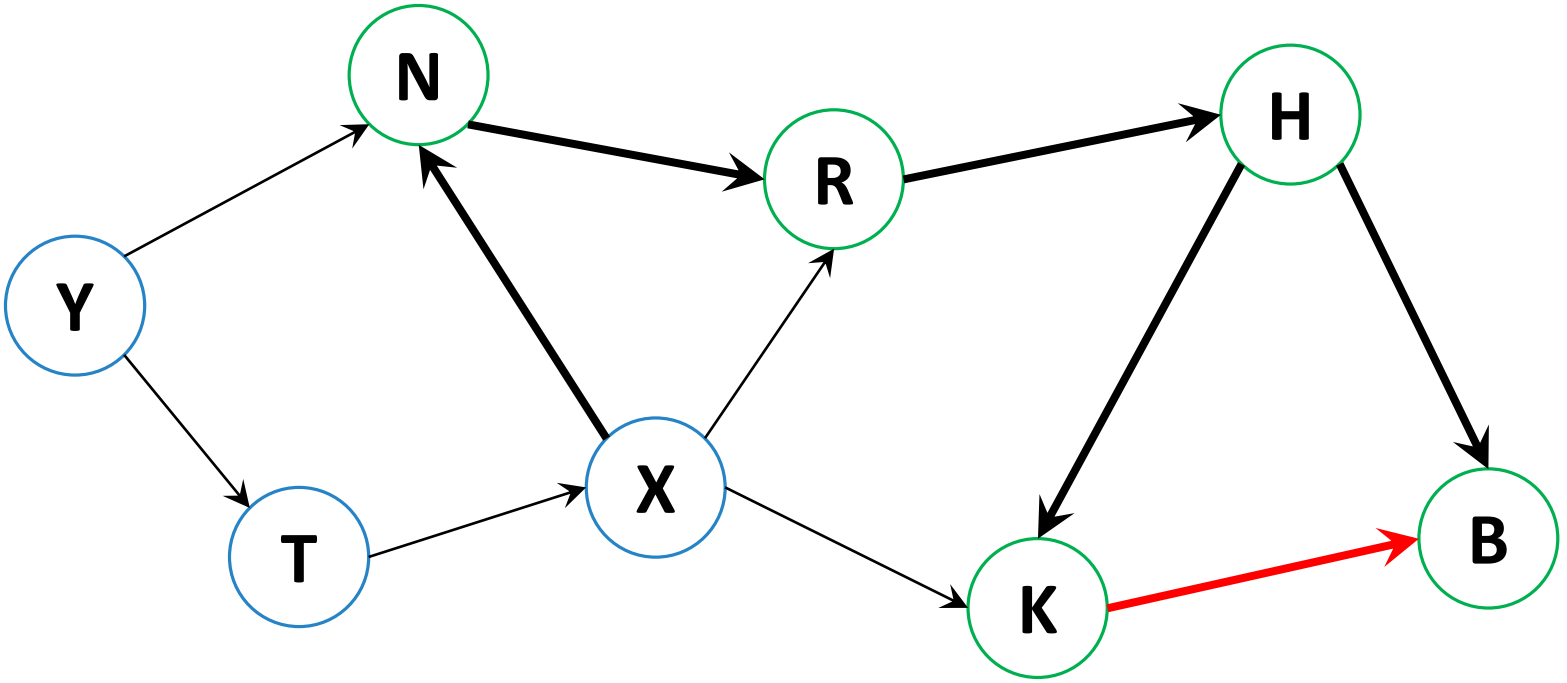
¿Cuáles nodos son posteriores a **B**?

Posteriores



¿Cuáles nodos son posteriores a **K**?

Posteriores



¿Cuáles nodos son posteriores a **B**? Espera…

Repeticiones

Estamos haciendo llamadas repetidas

Y es más, si pasamos por un ciclo el algoritmo no termina

¿Cómo se soluciona esto?

posteriores(X):

if X está pintado: ***return*** $\emptyset$

Pintar X

$P \leftarrow \emptyset$

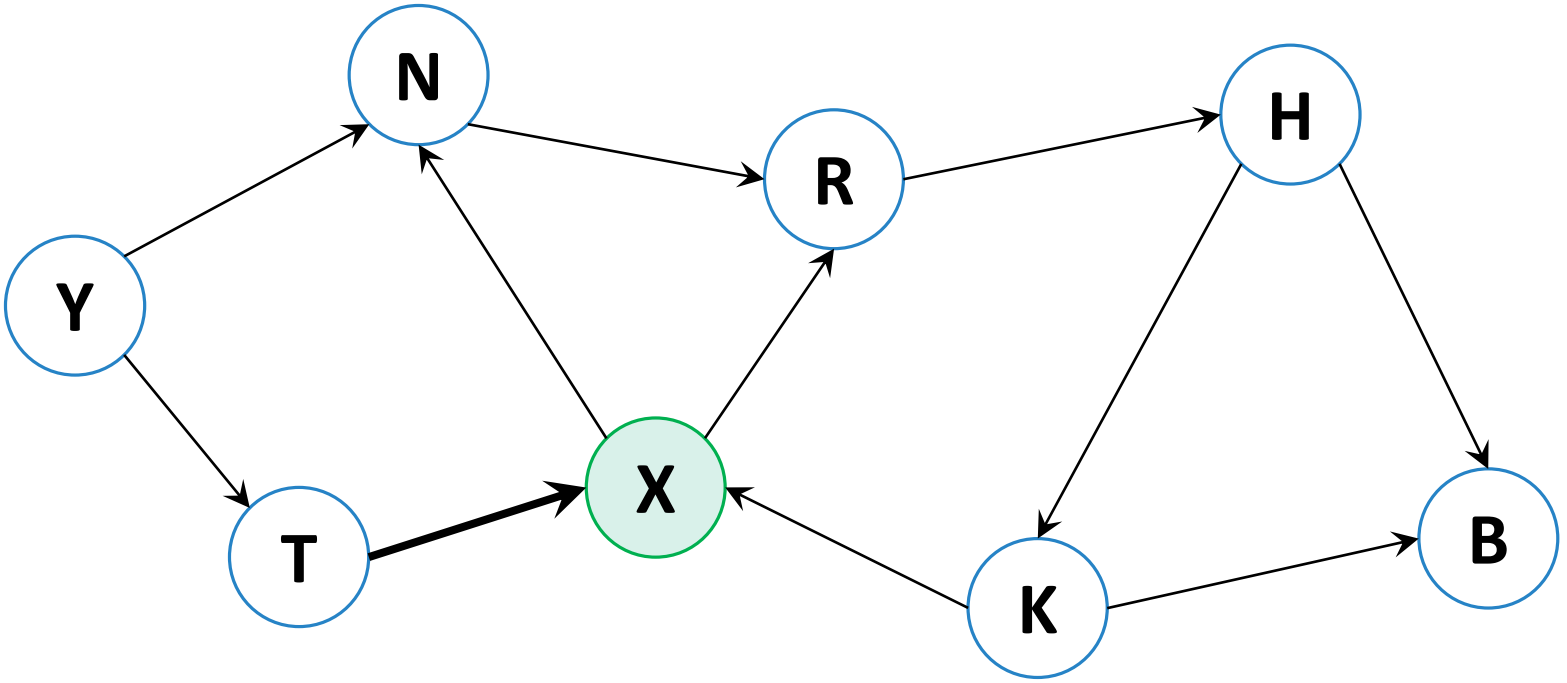
for Y tal que $X \rightarrow Y$:

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

$P \leftarrow P \cup$ ***posteriores***(Y)

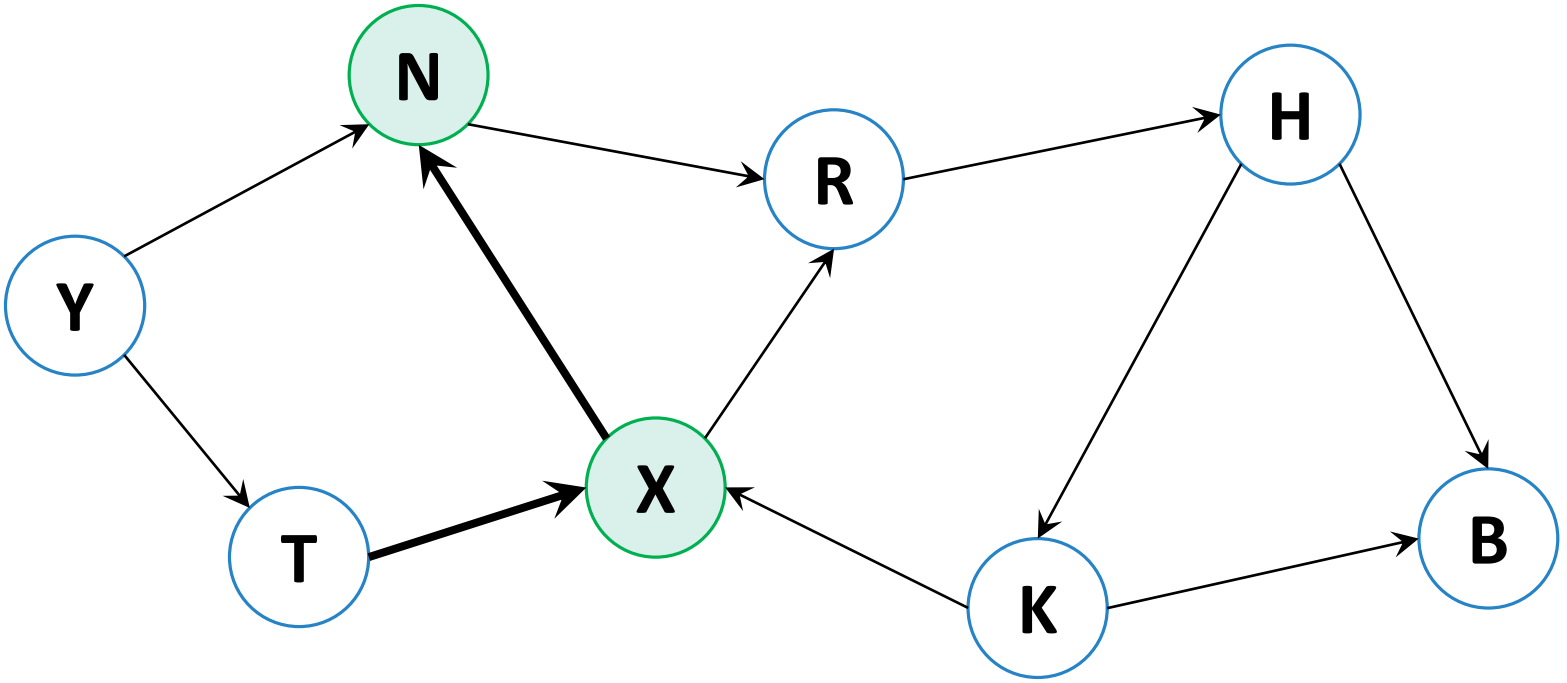
return P

Posteriores



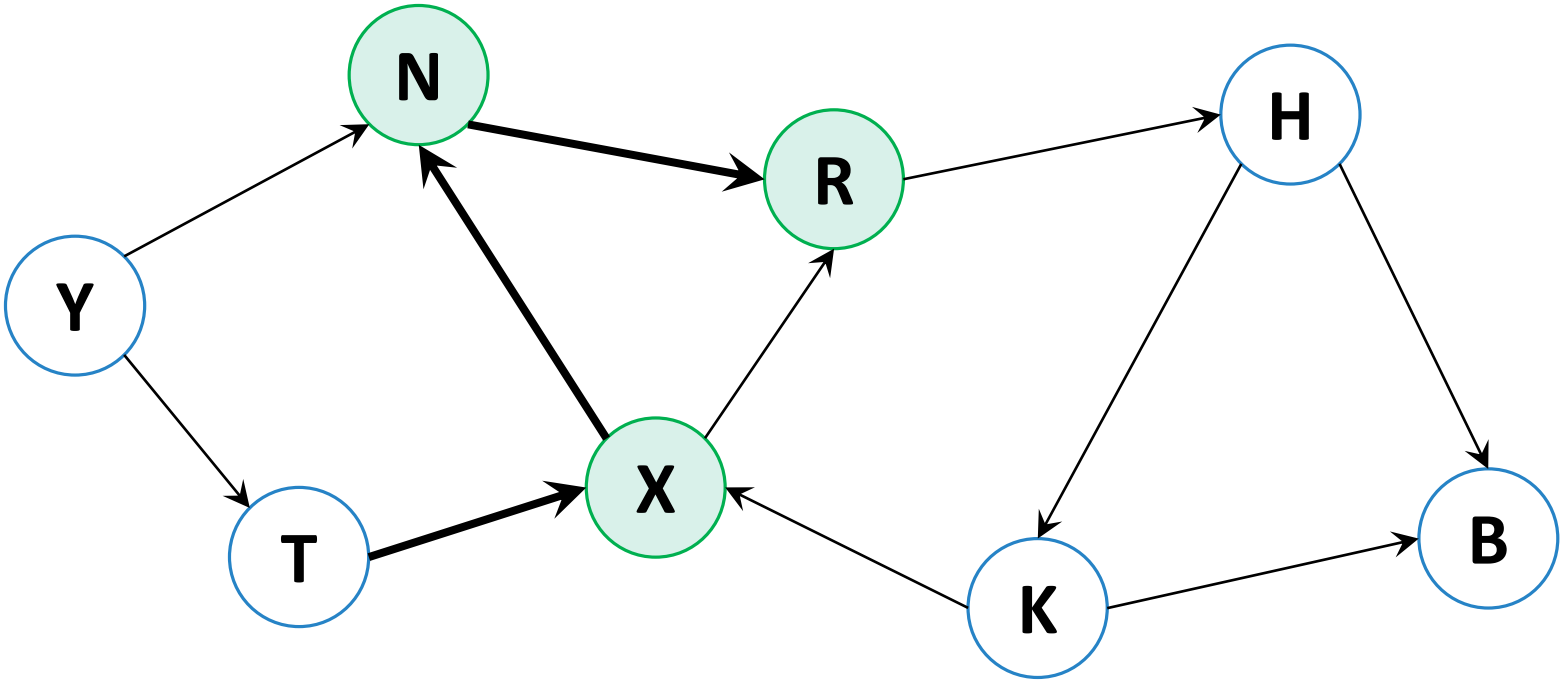
¿Cuáles nodos son posteriores a **T**?

Posteriores



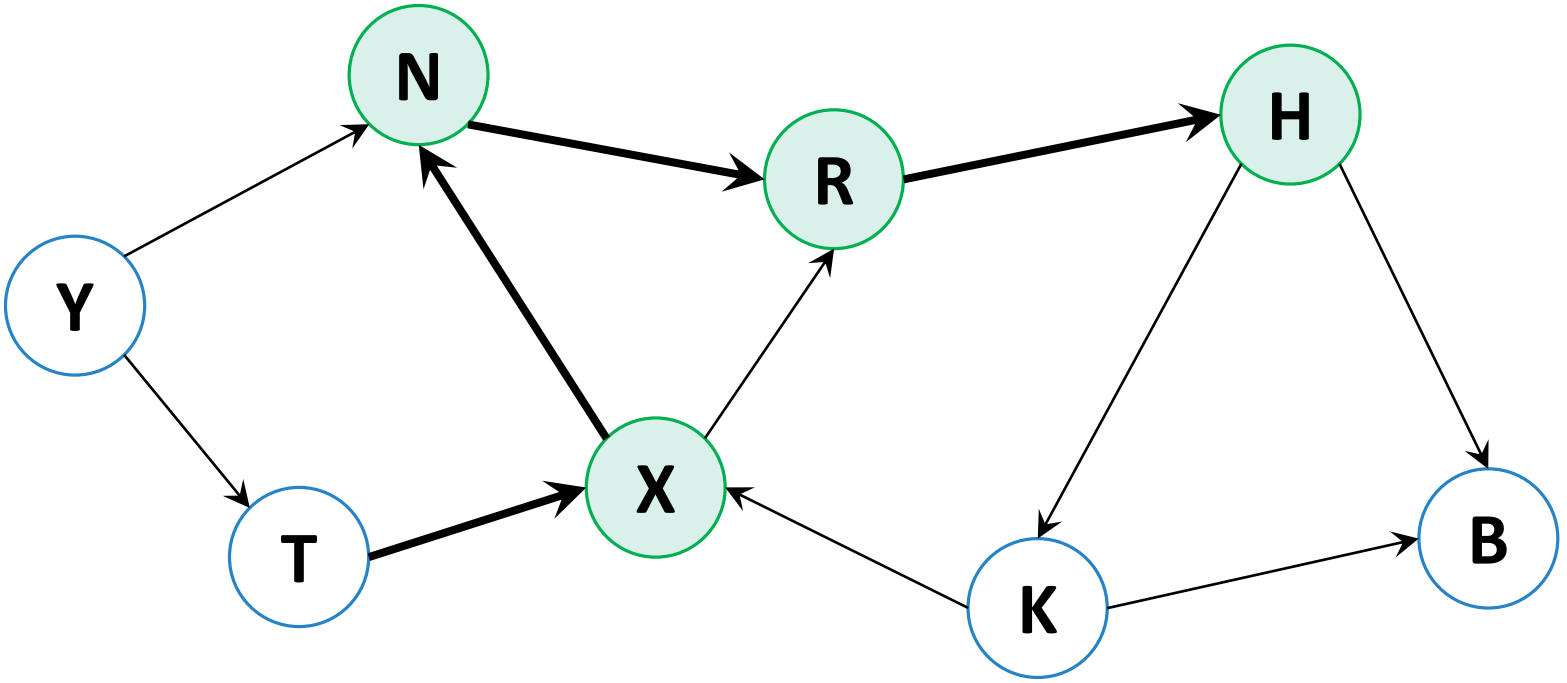
¿Cuáles nodos son posteriores a **X**?

Posteriores



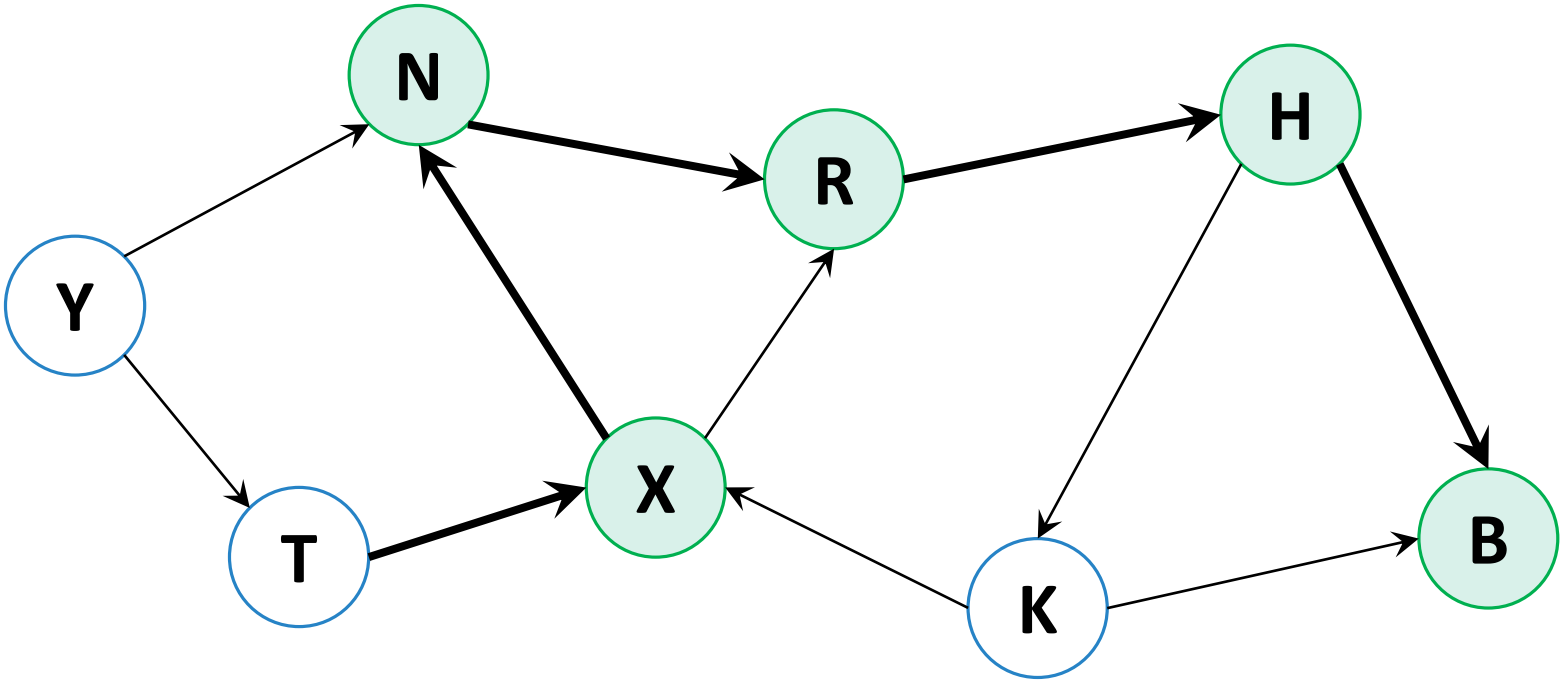
¿Cuáles nodos son posteriores a **N**?

Posteriores



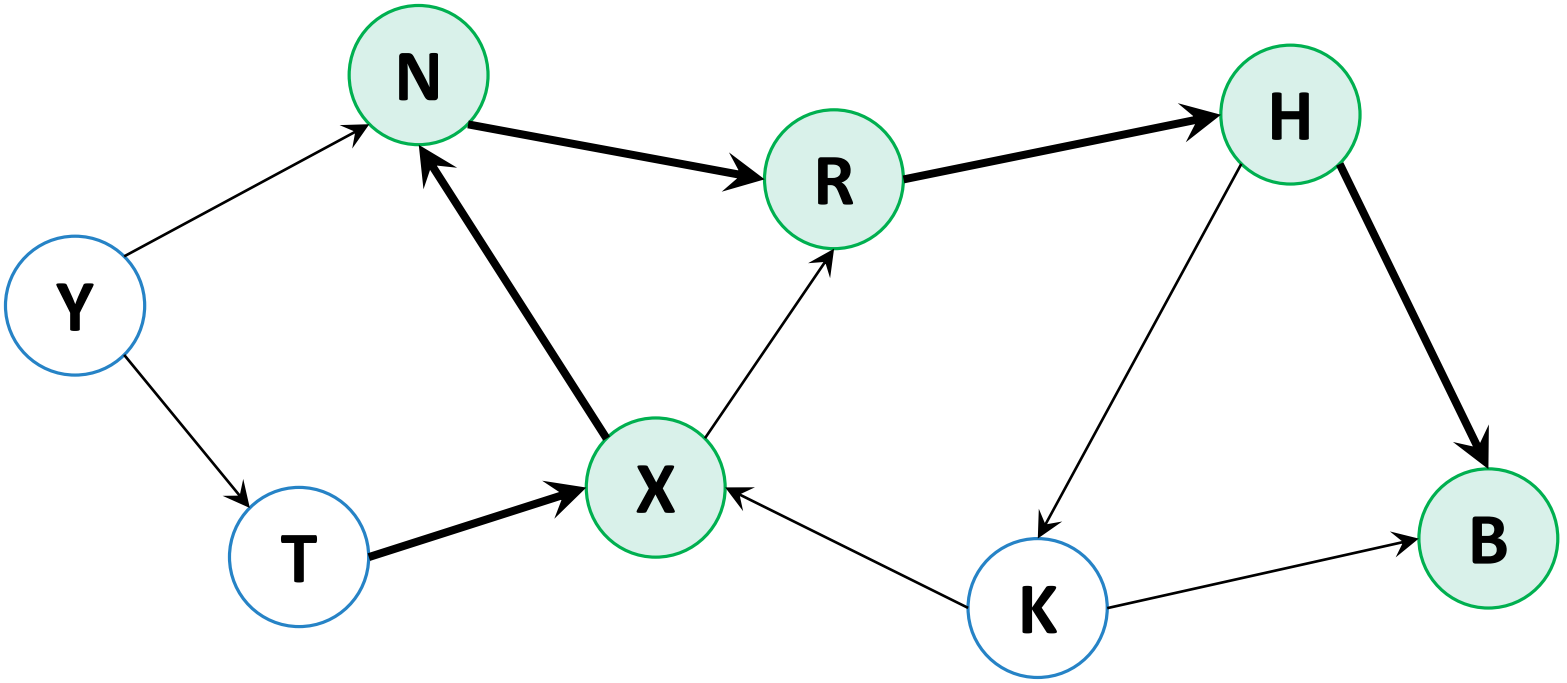
¿Cuáles nodos son posteriores a **R**?

Posteriores



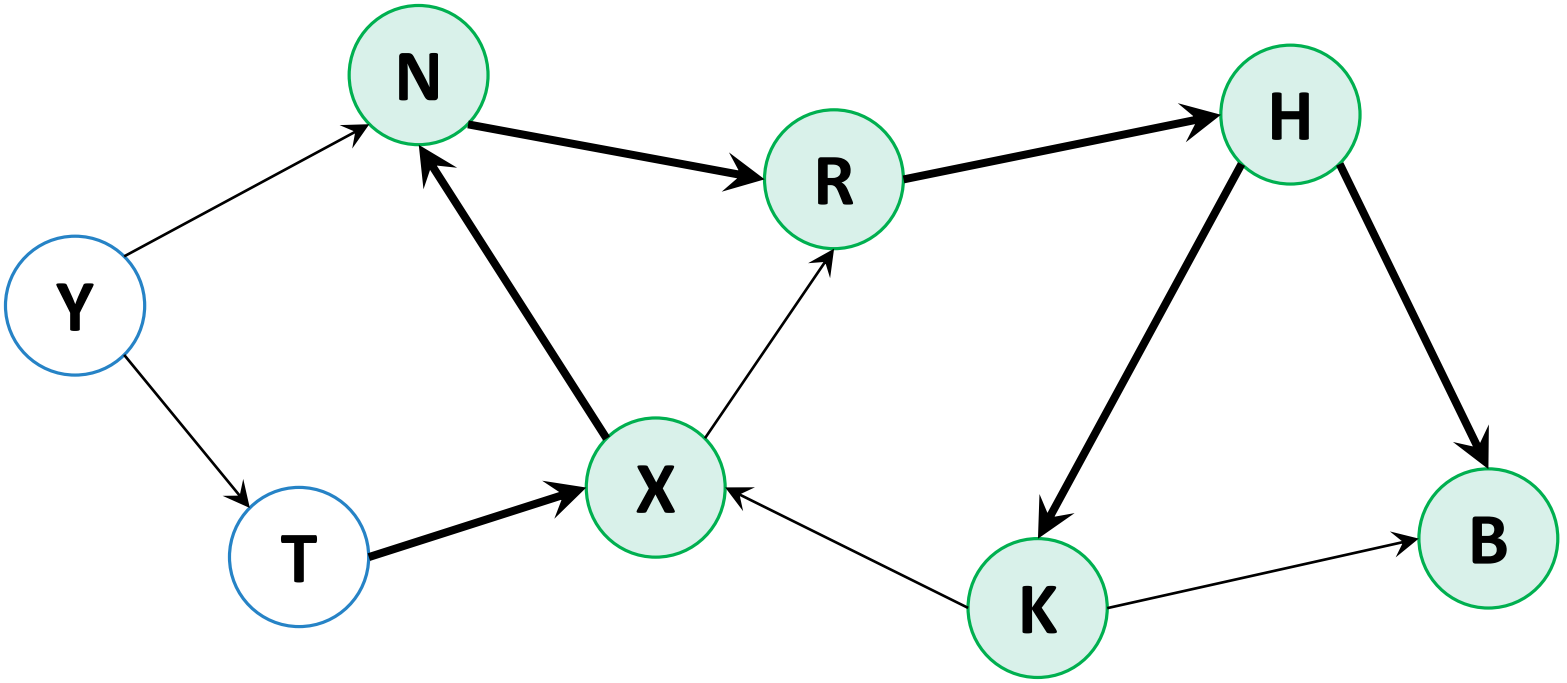
¿Cuáles nodos son posteriores a **H**?

Posteriores



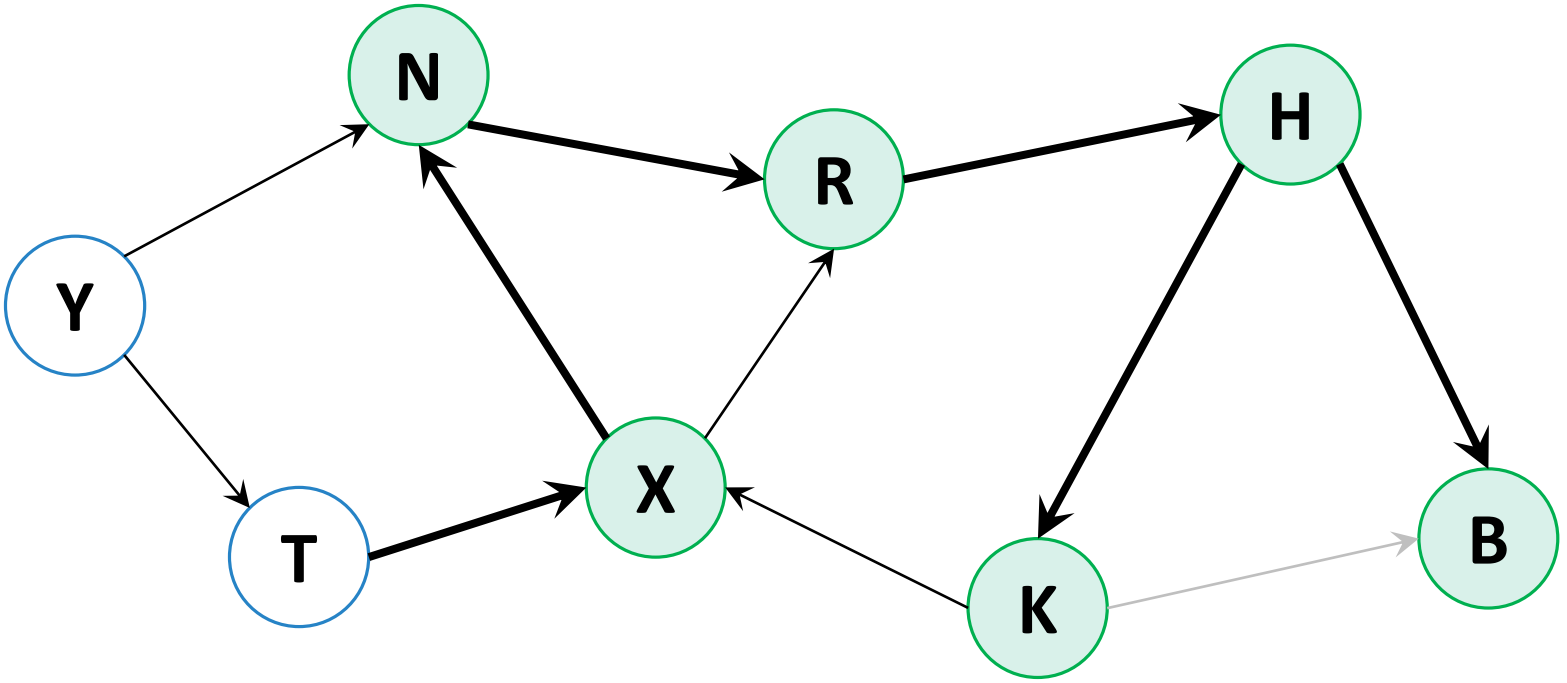
¿Cuáles nodos son posteriores a **B**?

Posteriores



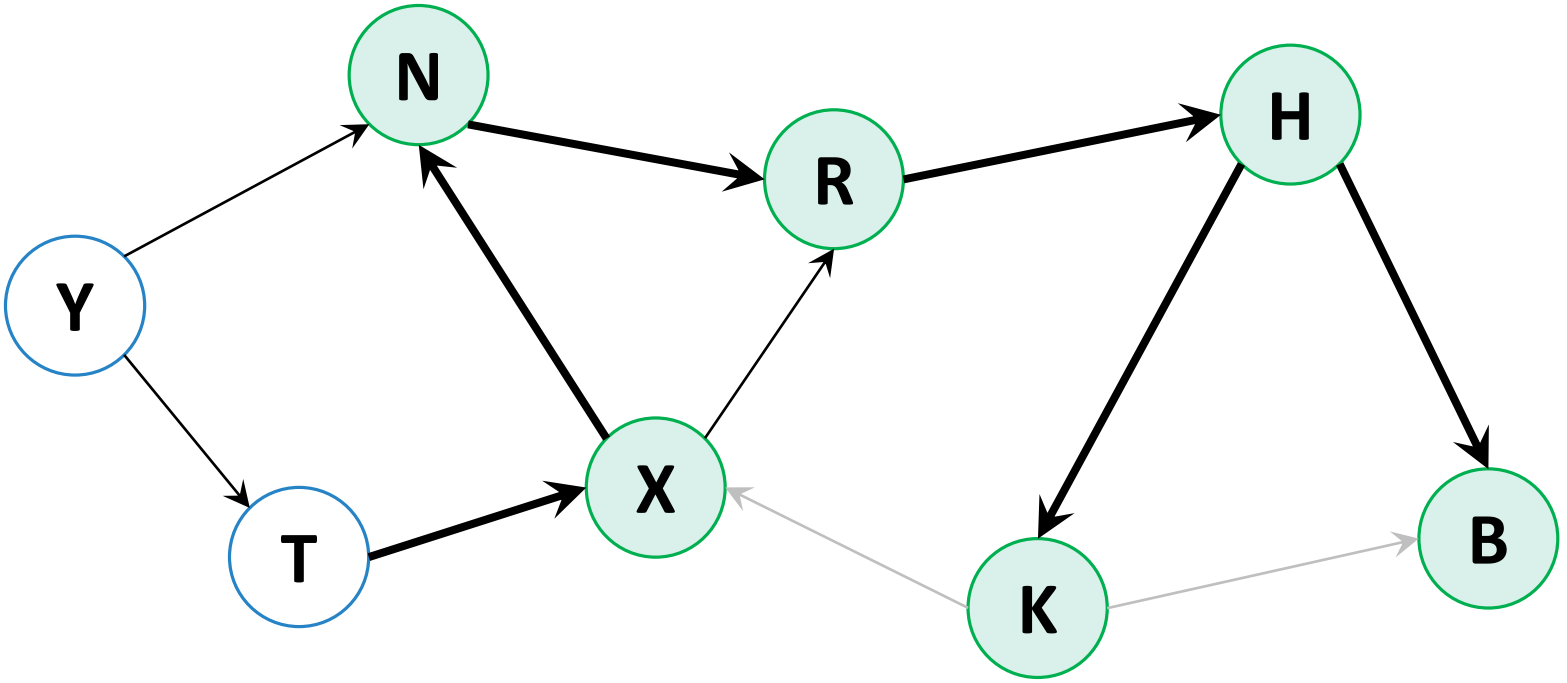
¿Cuáles nodos son posteriores a **H**?

Posteriores



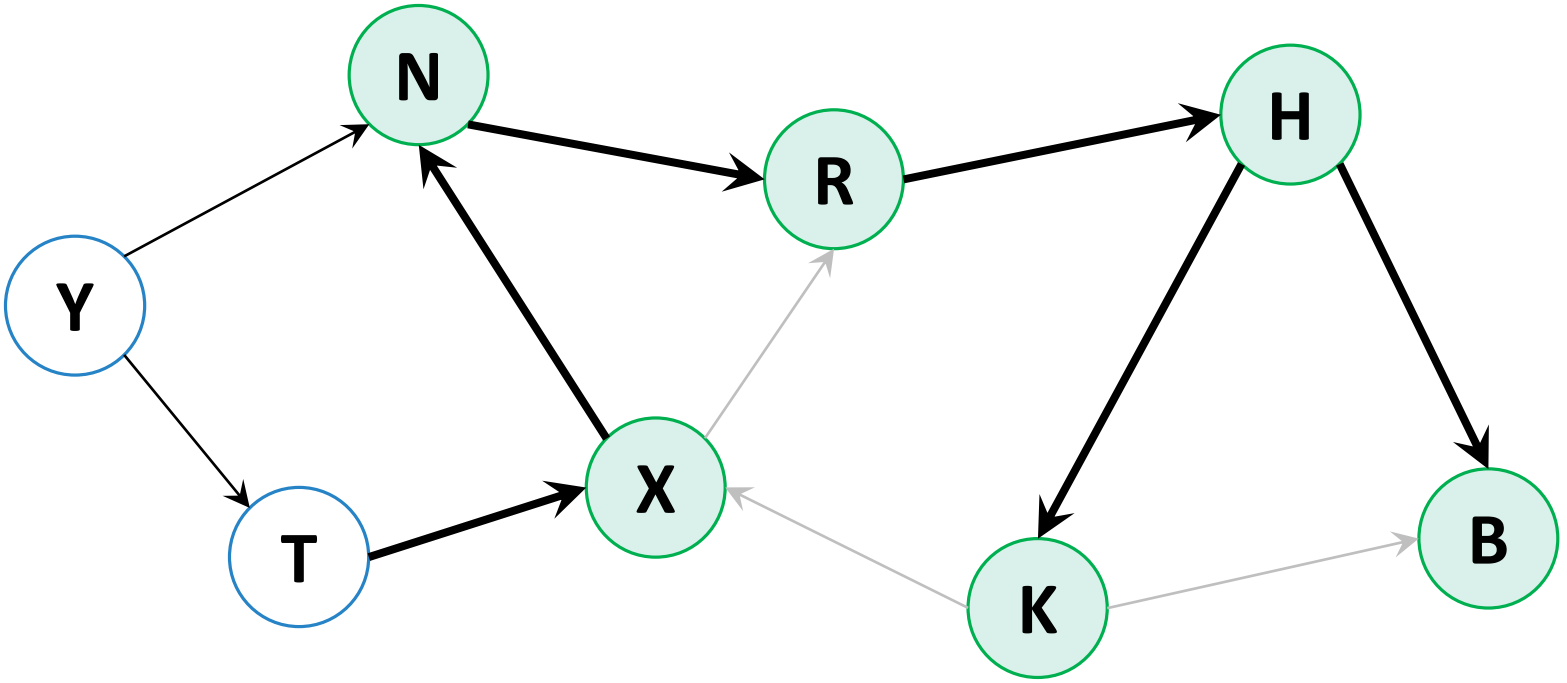
¿Cuáles nodos son posteriores a **K**?

Posteriores



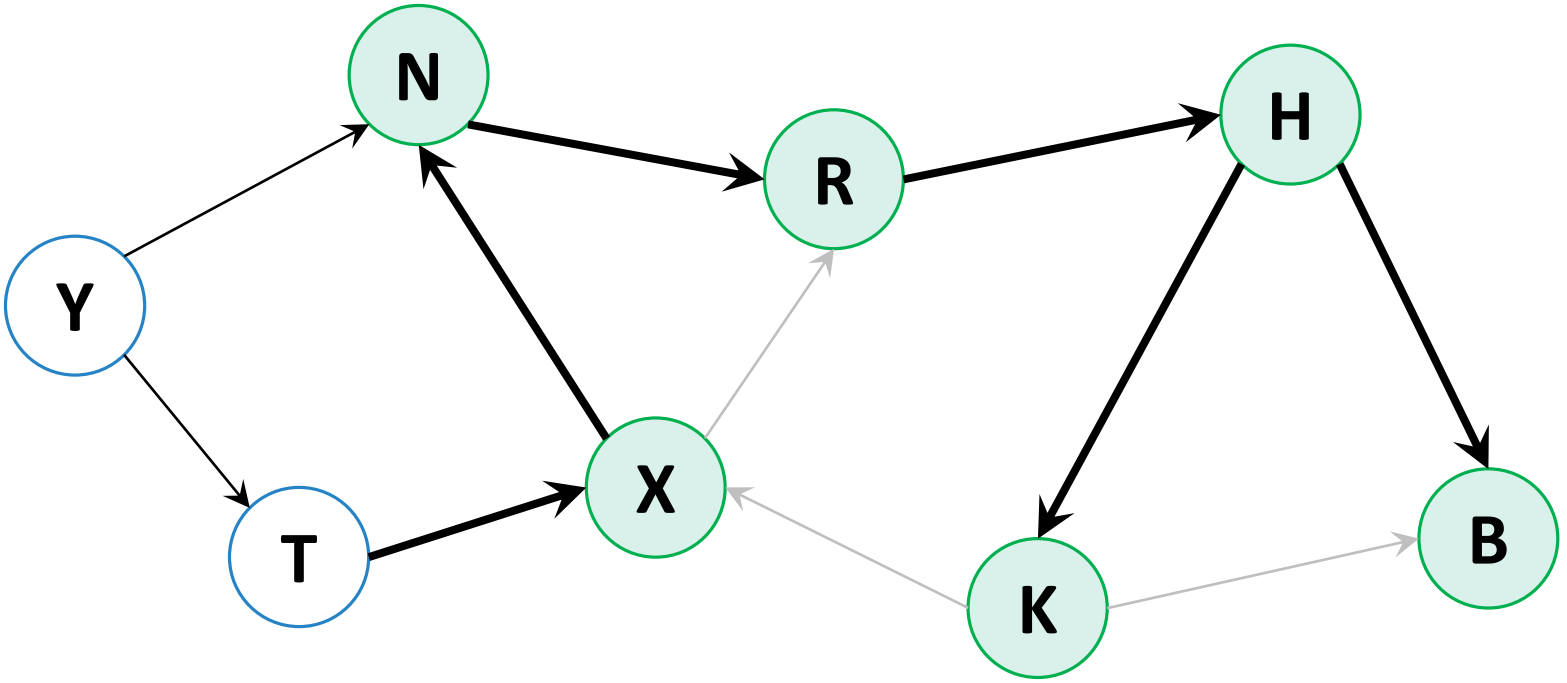
¿Cuáles nodos son posteriores a **K**?

Posteriores



¿Cuáles nodos son posteriores a **X**?

Posteriores



¡Listo!

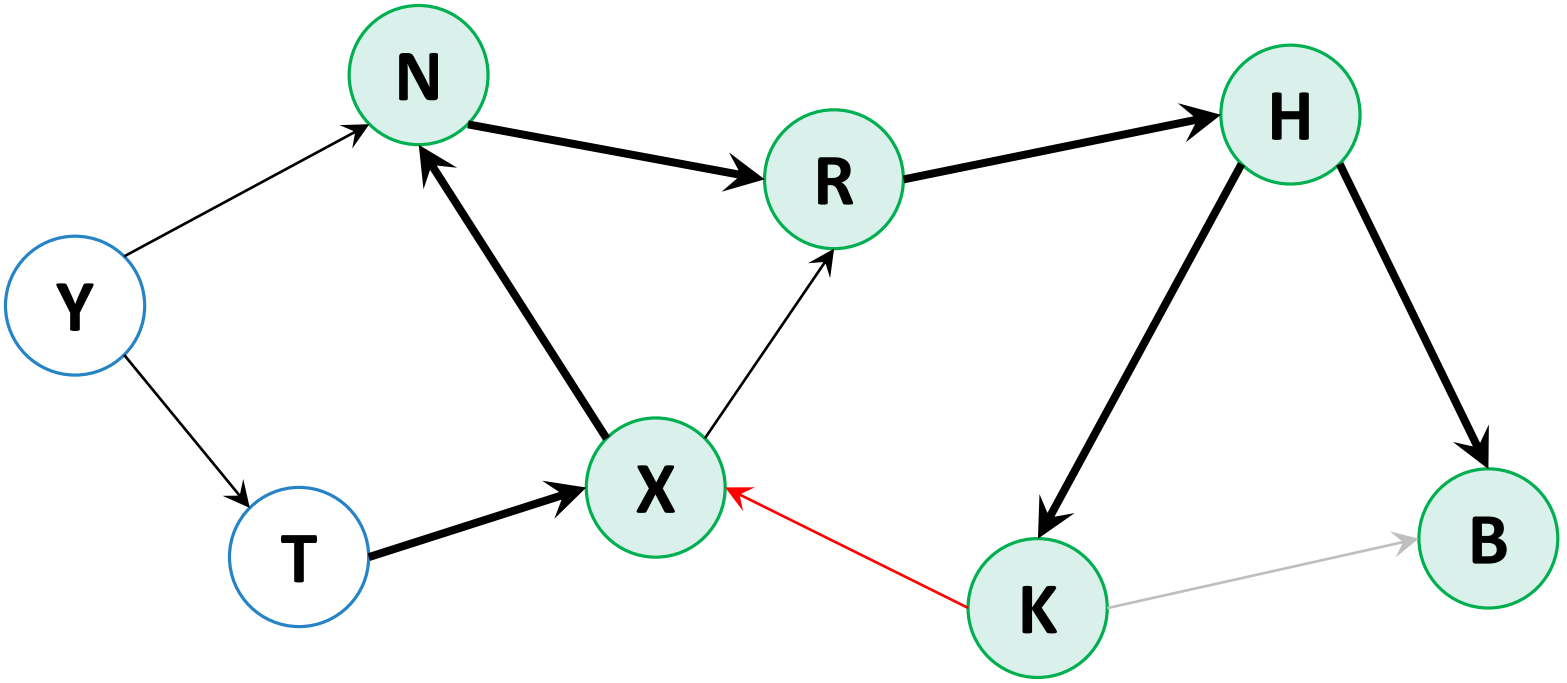
Identificar Ciclos

Ok, podemos identificar las tareas posteriores a una tarea

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

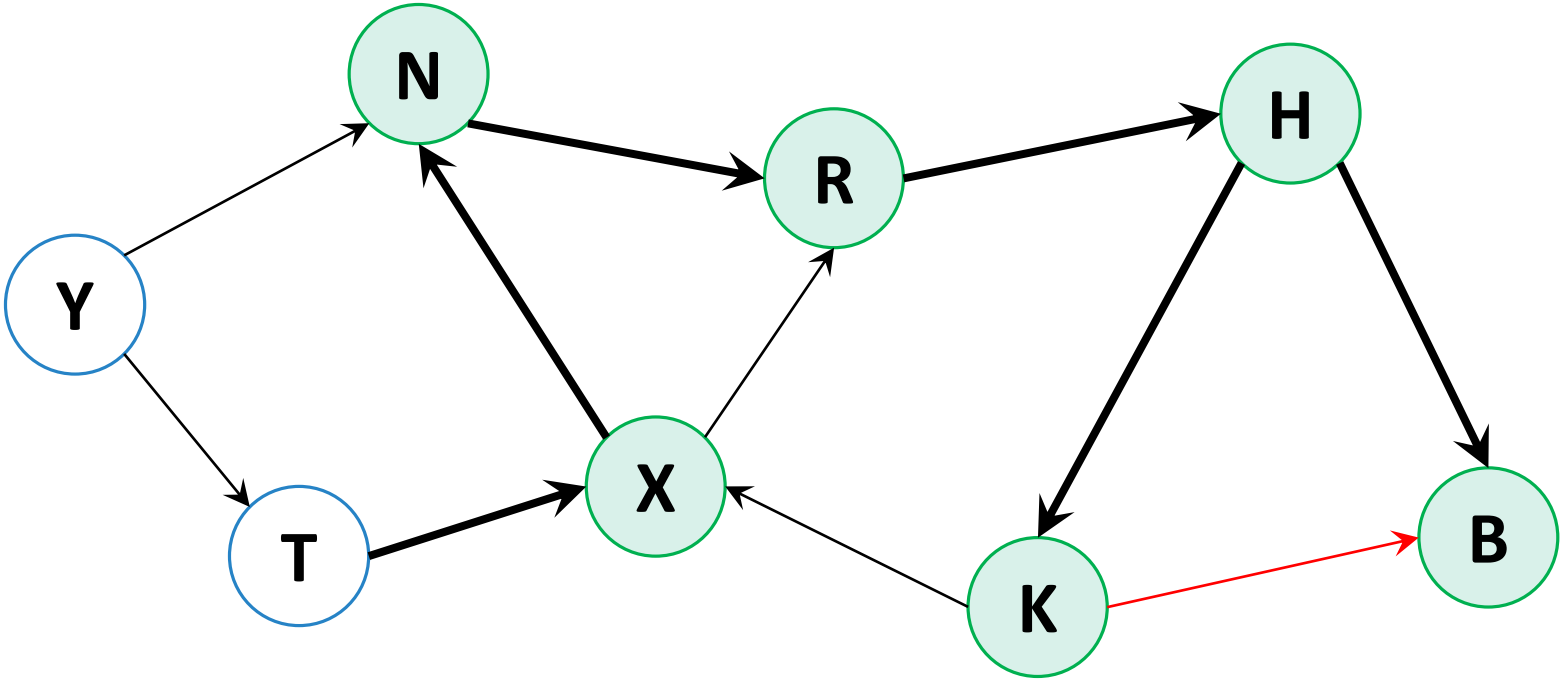
¿Podemos usar este *approach* para identificar ciclos?

Algo así como esto



¡Algoritmo date cuenta que eso es un ciclo!

Algo así como esto



¡Y que eso no!

Observación

Si el nodo recién descubierto Y está pintado, hay dos casos:

- Si lo descubrió un nodo posterior a Y, hay ciclo
- Si lo descubrió un nodo anterior a Y, no hay problema

Hasta que $\boldsymbol{posteriores}(X)$ retorne, todos los nodos explorados 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 dentro de $\big(G(V, E)\big)$**:**

for $X \in V$:

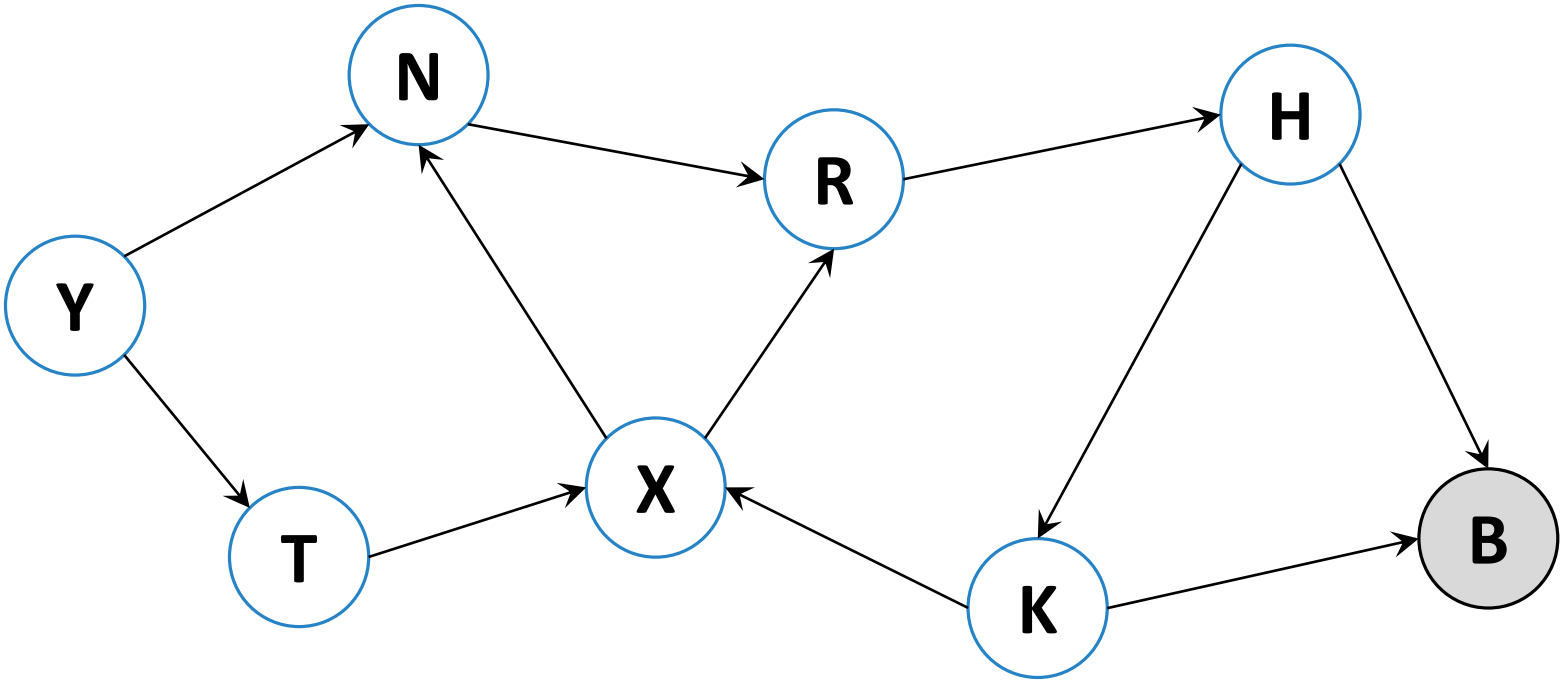
if X está pintado, ***continue***

if hay ciclo luego de (X):

return true

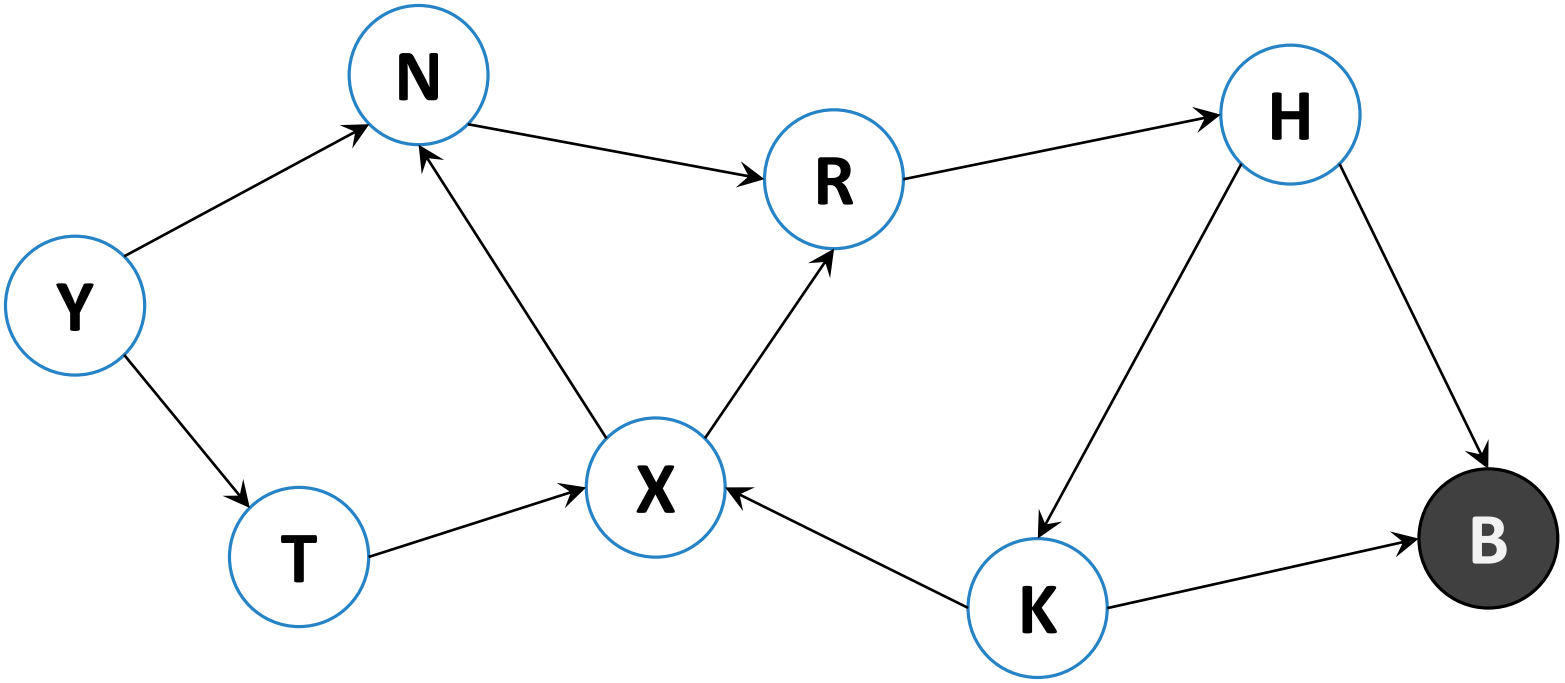
return false

En acción



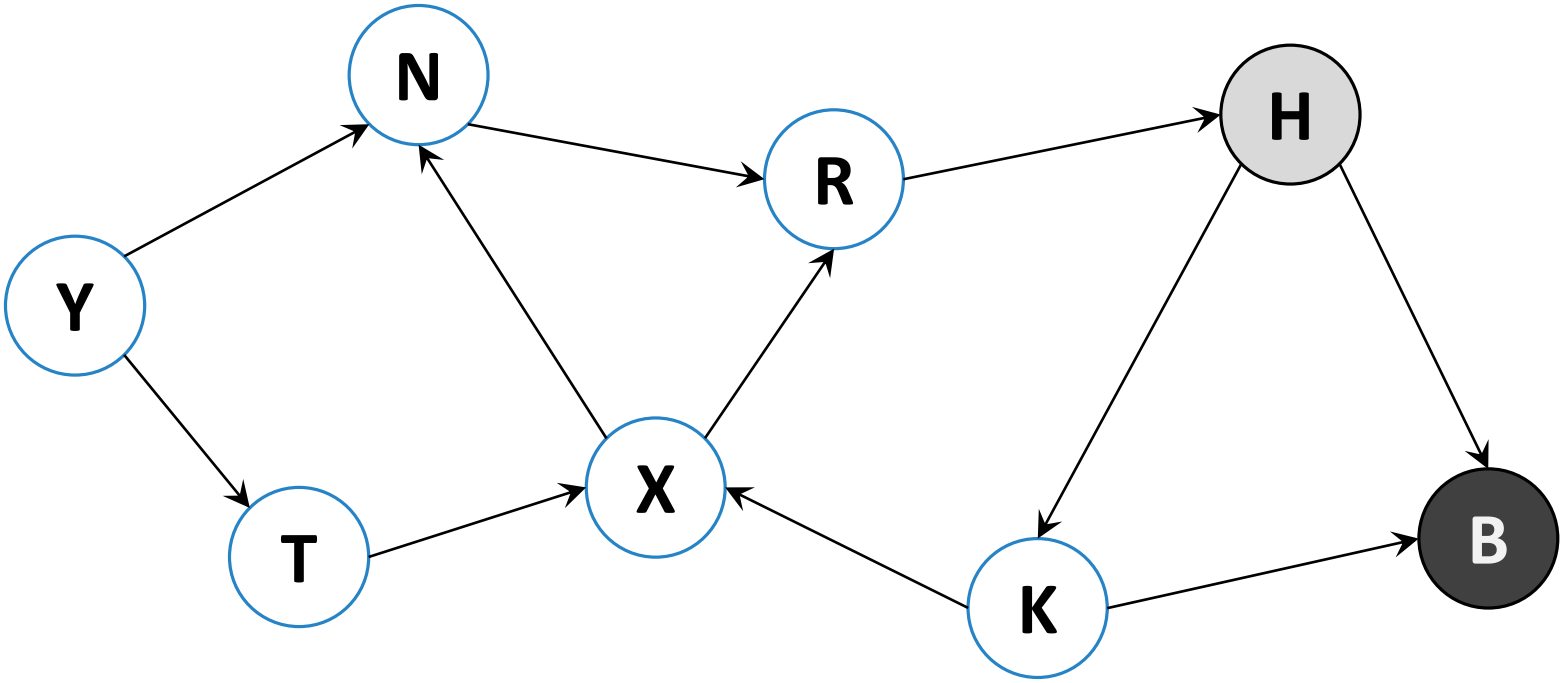
¿Hay un ciclo luego de **B**?

En acción



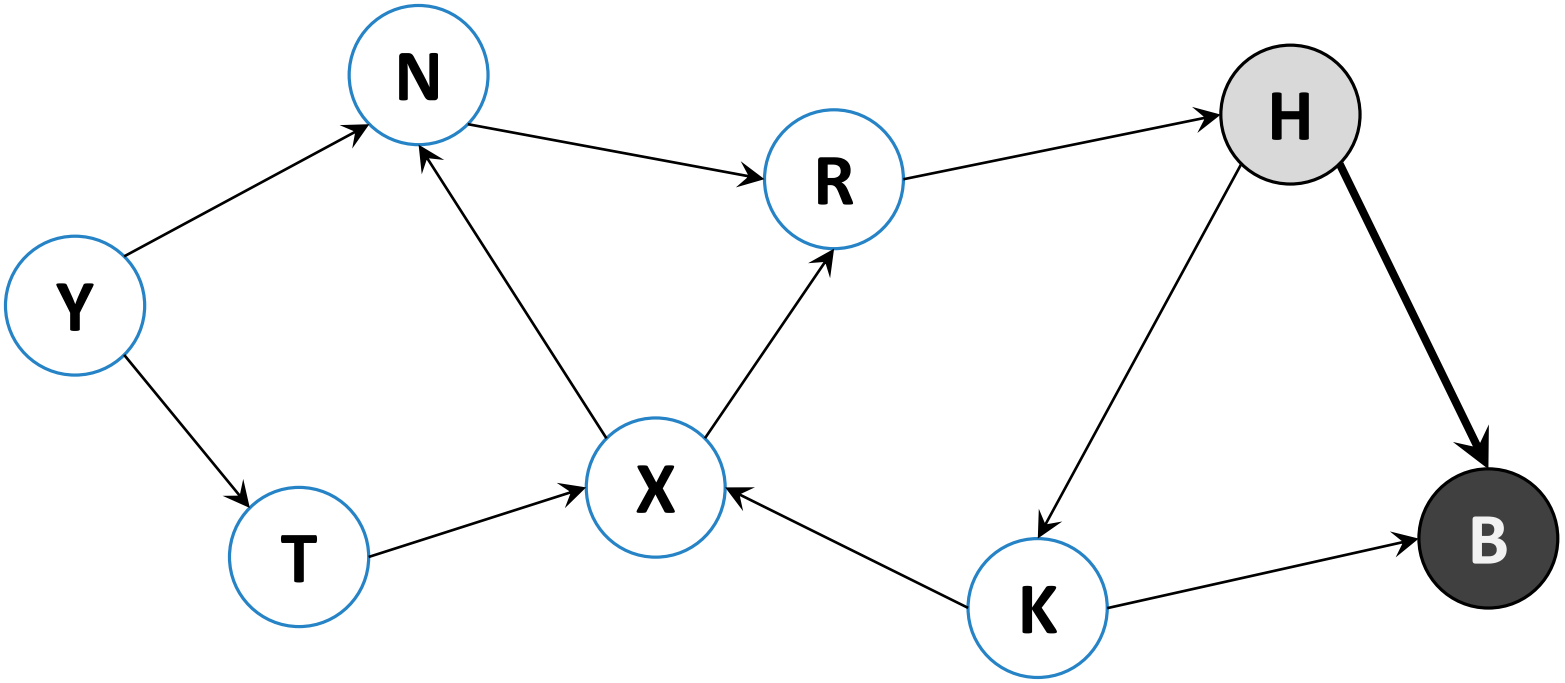
No

En acción



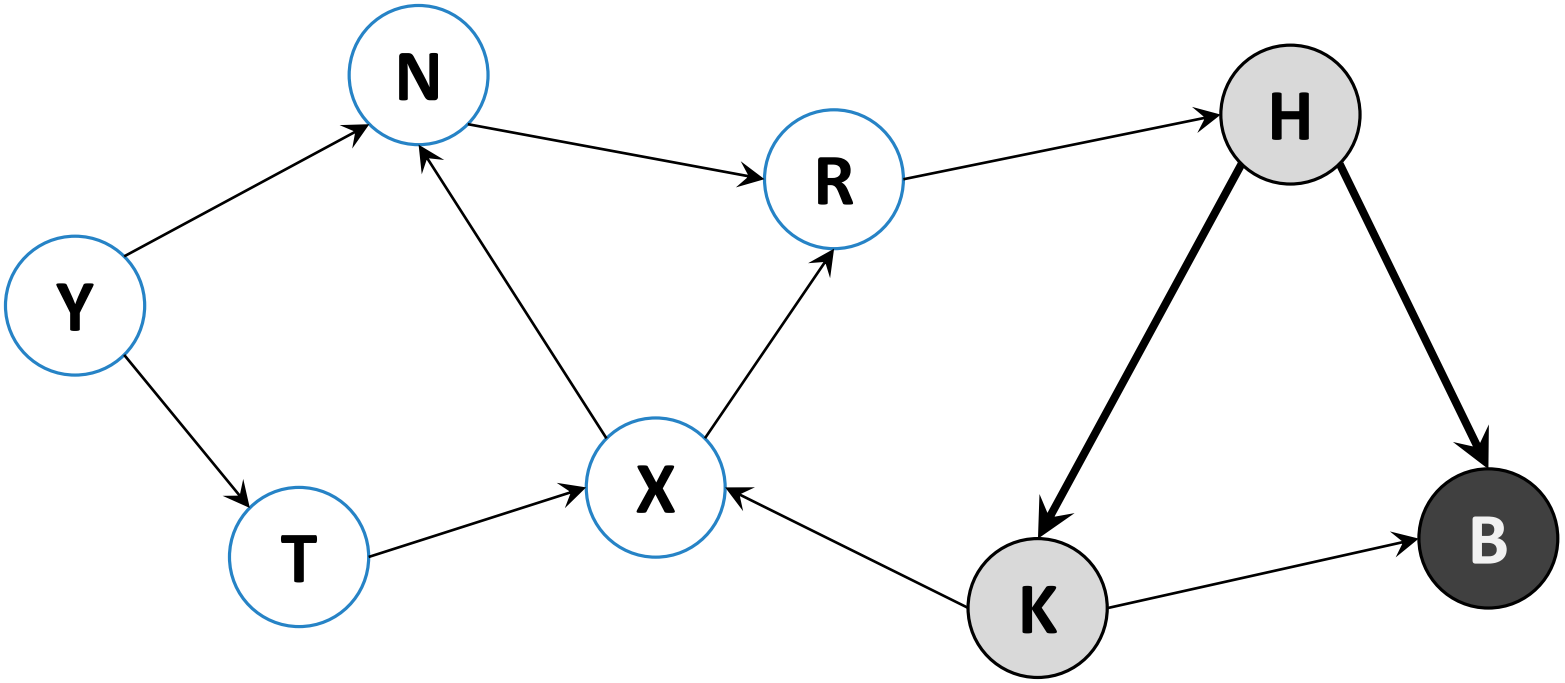
Ok, ¿Hay un ciclo luego de **H**?

En acción



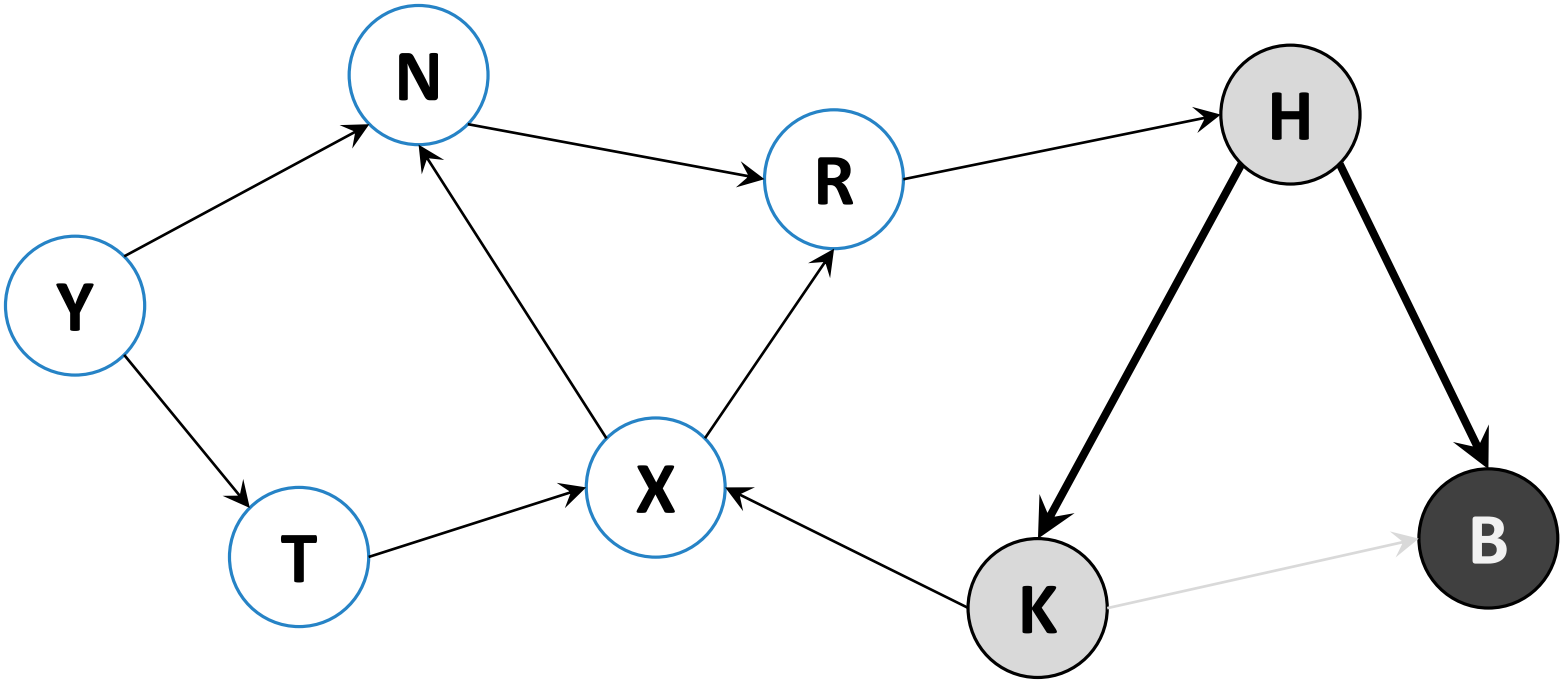
¿Hay un ciclo luego de **H**?

En acción



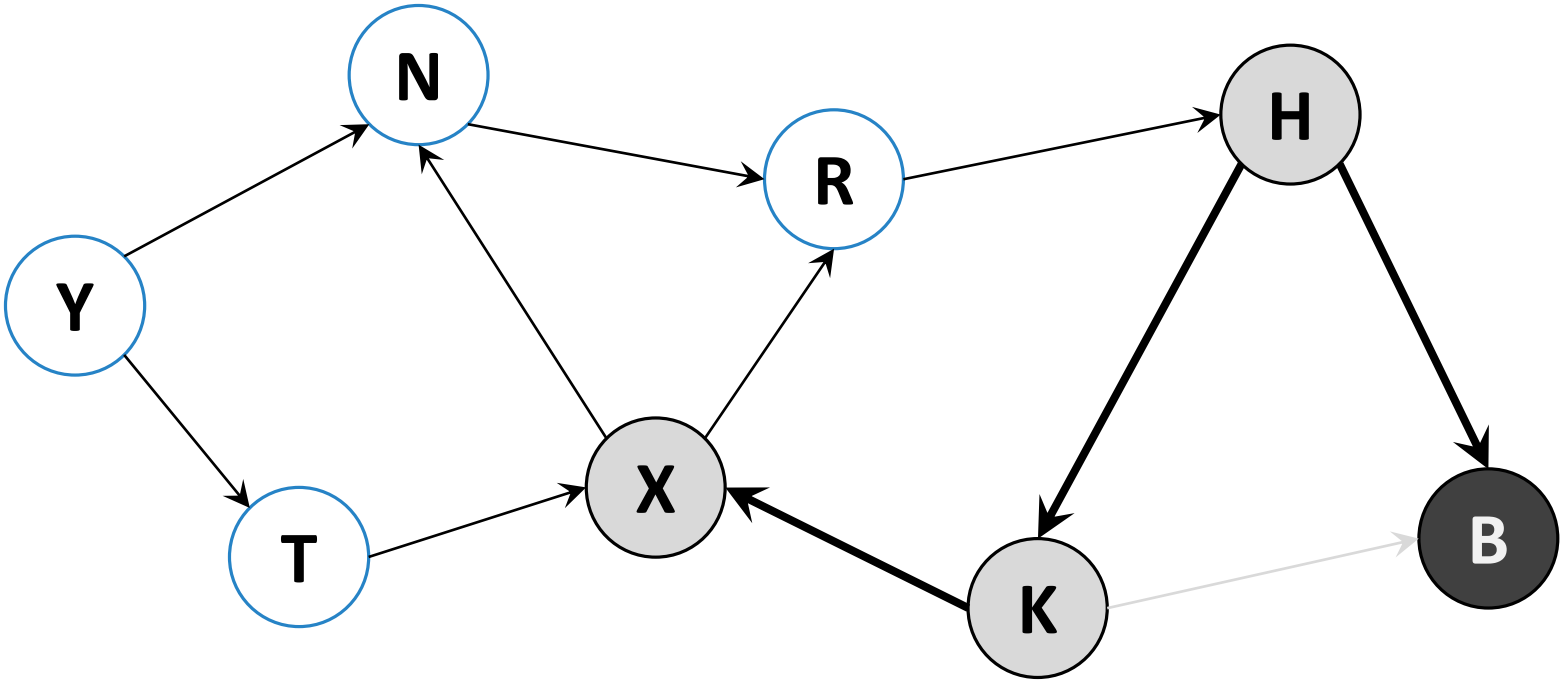
¿Hay un ciclo luego de **K**?

En acción



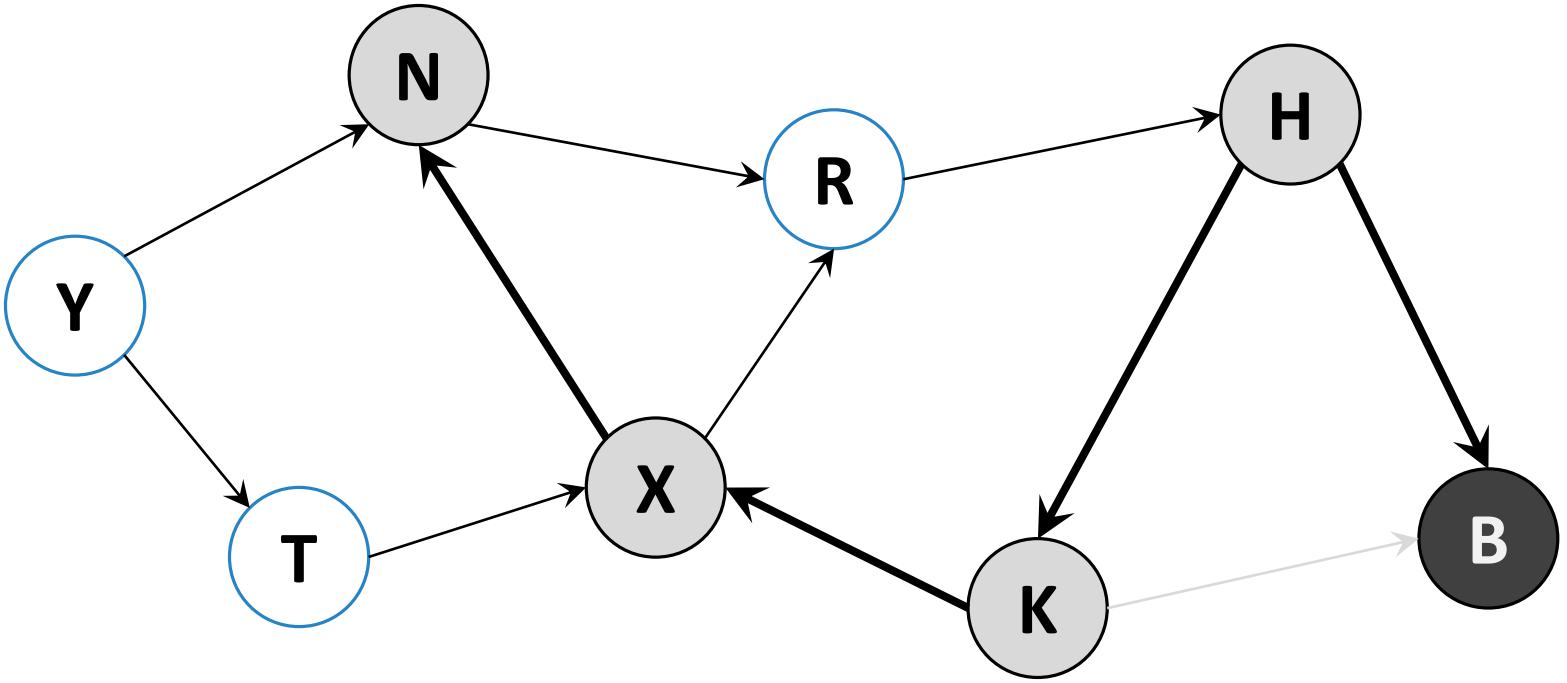
¿Hay un ciclo luego de **K**?

En acción



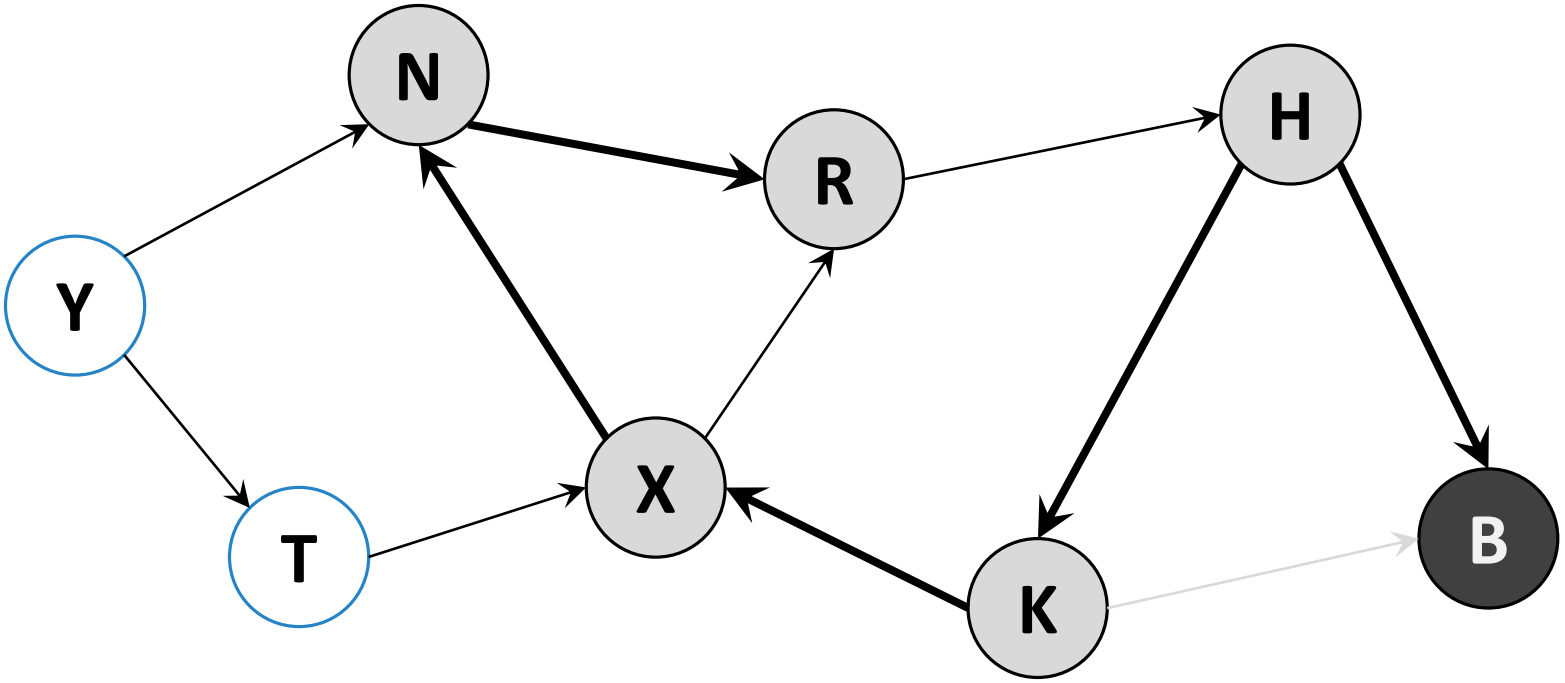
¿Hay un ciclo luego de **X**?

En acción



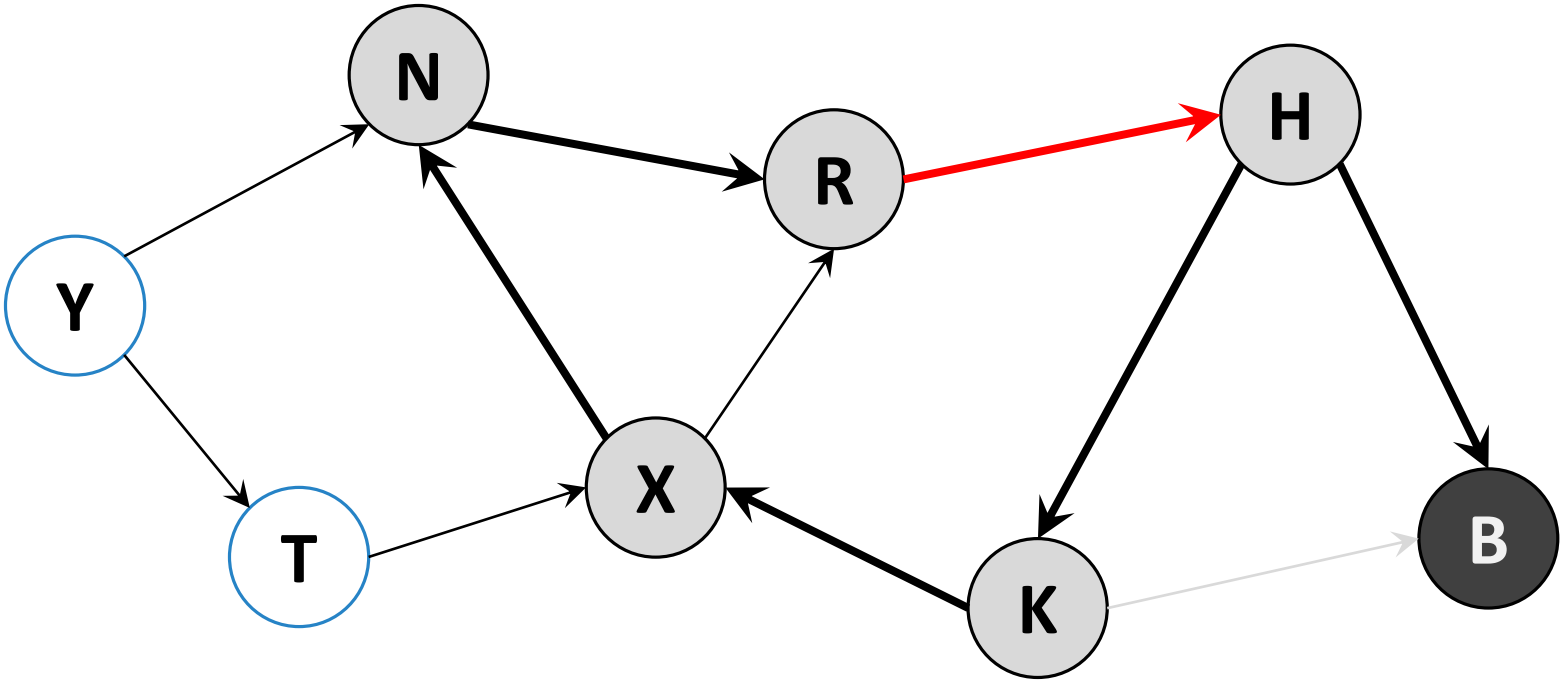
¿Hay un ciclo luego de **N**?

En acción



¿Hay un ciclo luego de **R**?

En acción



!!!!!!!!!!!!!!!!!!!!!!!!!!!

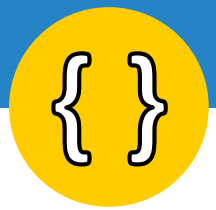
Depth First Search (DFS)

Algoritmos como estos se llaman de **búsqueda en profundidad**

Llegan al final de cada rama antes de seguir con la siguiente

¿Cuál es la complejidad de estos algoritmos?

Grafos en memoria



Hay dos principales maneras de implementar un grafo

1. Listas de Adyacencia

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

2. Matriz de adyacencia

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