¿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, 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 recibo?

5: 2 3 6 —*la tarea 5 tiene como prerrequisitos las tareas 2, 3 y 6*

3: 0 1 —*la tarea 3 tiene a su vez como prerrequisitos las tareas 0 y 1*

6: 3 4 — *… y así sucesivamente*

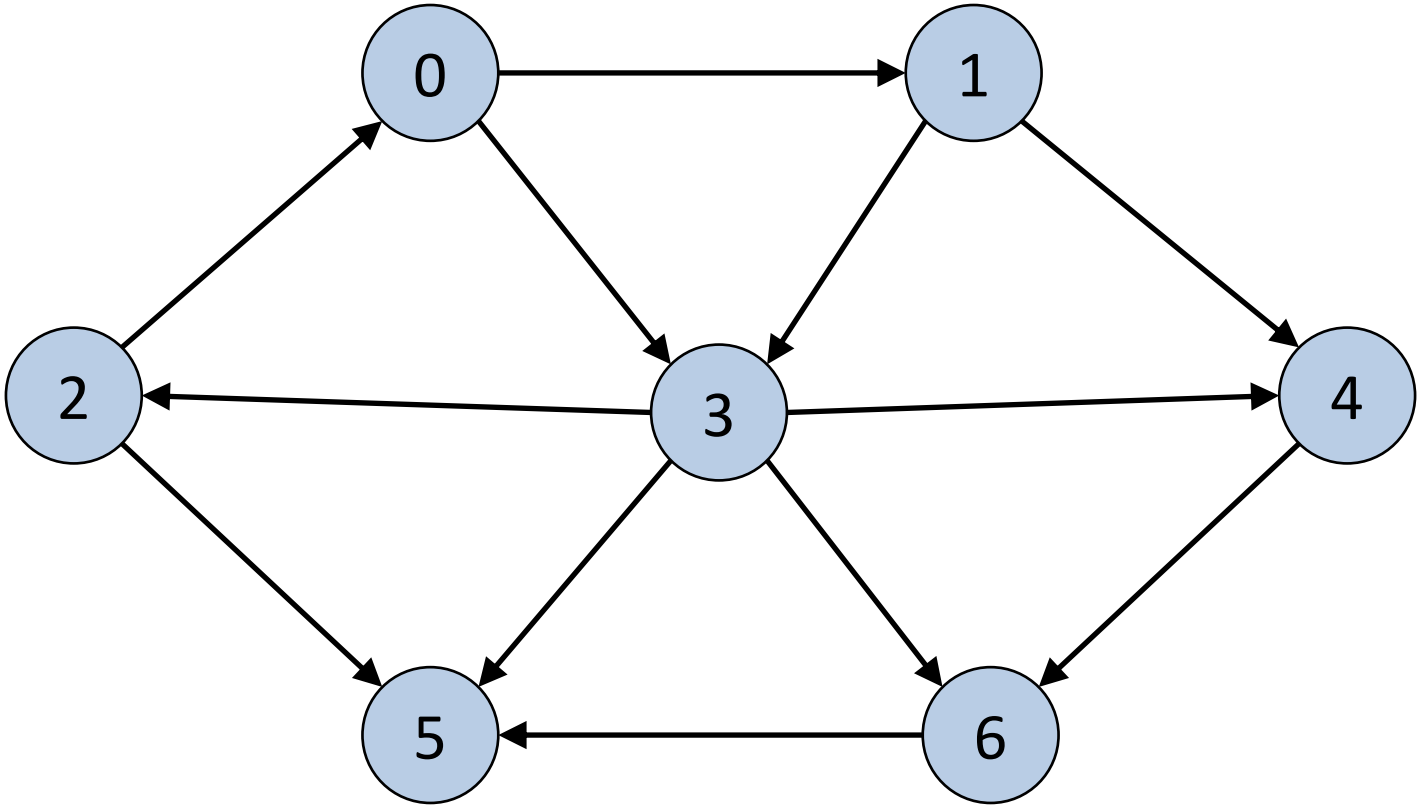
2: 3

0: 2

1: 0

4: 1 3

¿Cómo lo representó visualmente?



La estructura anterior es un **grafo**

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

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

Hay dos tipos principales de grafos :

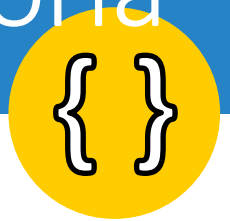
1. direccionales (como el del ejemplo)

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

2. no direccionales

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

Representación de grafos en memoria



Hay dos principales maneras de representar un grafo en la memoria del computador:

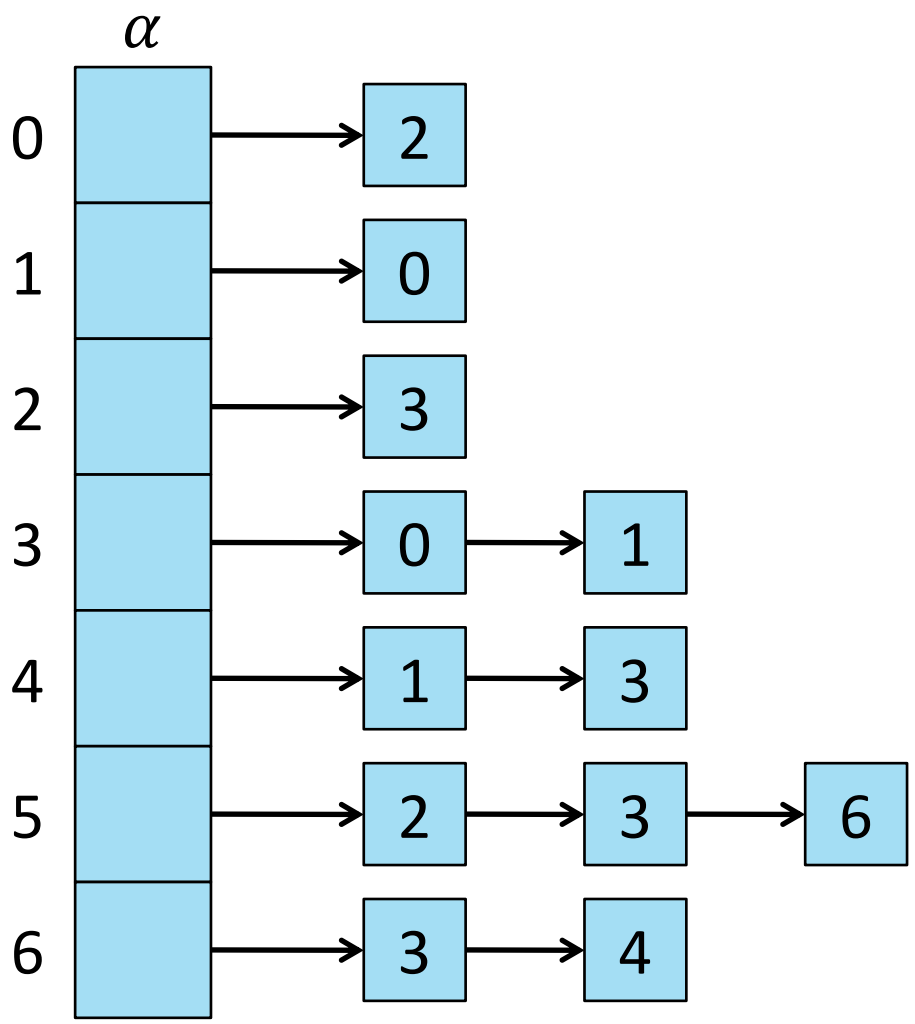
1. **Listas de adyacencias**

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

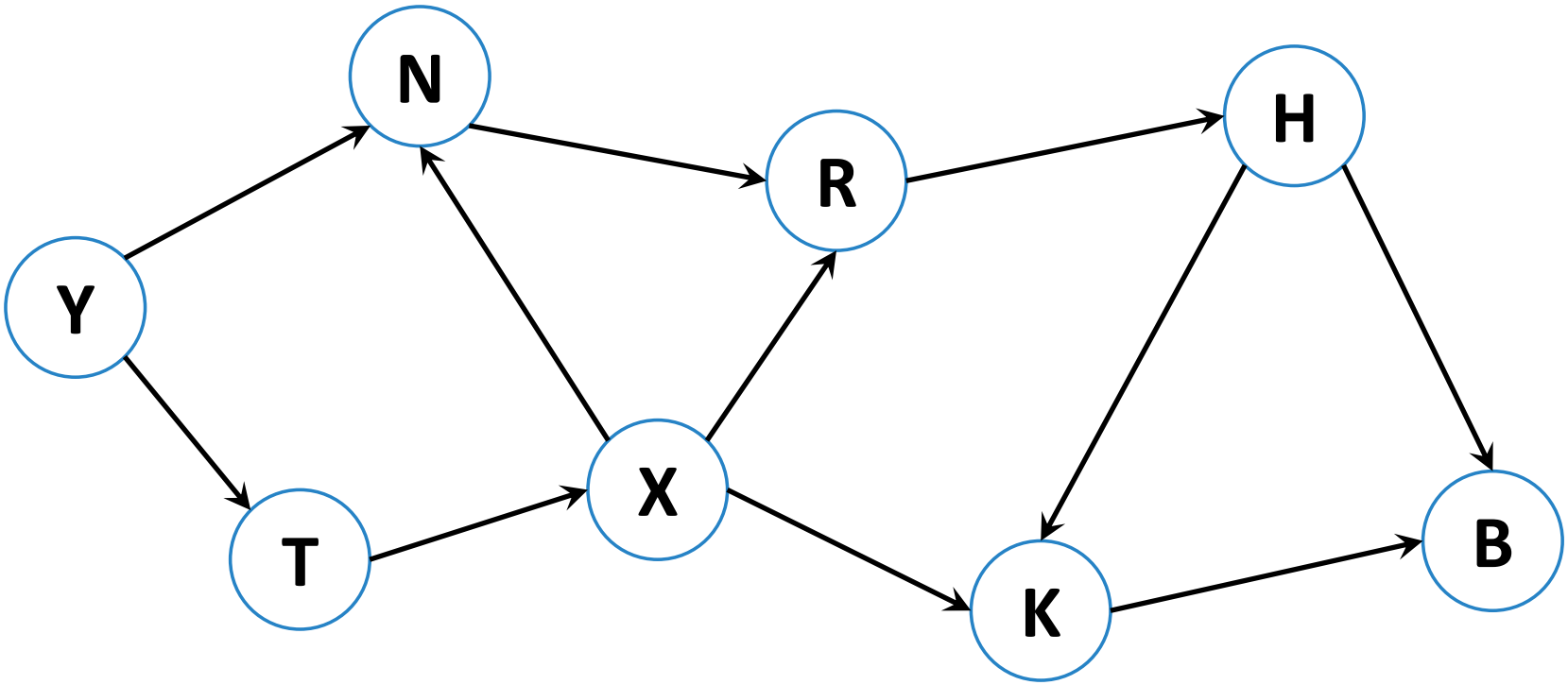
2. Matriz de adyacencias

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

El grafo direccional anterior es representado por 7 listas de adyacencias, $\alpha[i]$

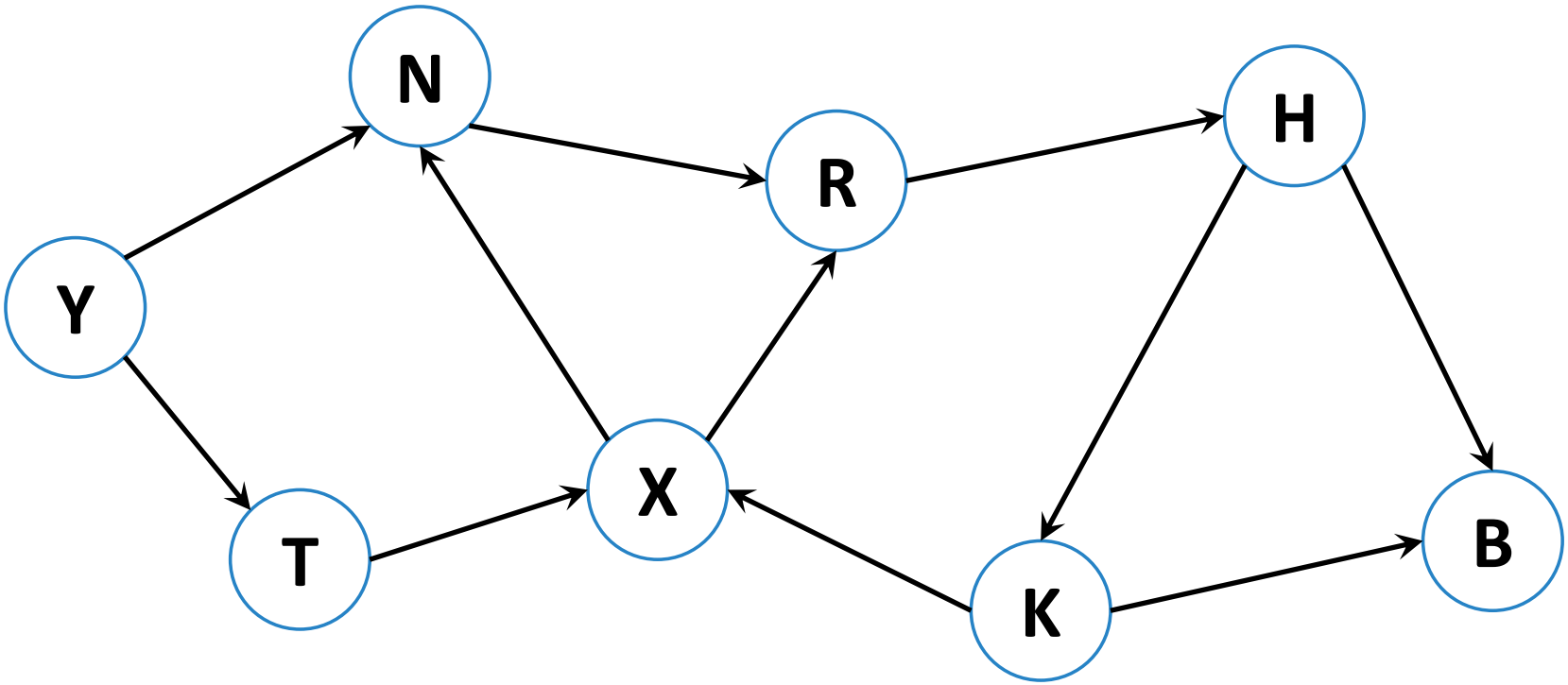


Dibujemos el grafo de un proyecto



¿Algún problema con este proyecto?

¿Qué pasa en este caso?



¿Algún problema con este proyecto?

Ciclos

Si el grafo tiene un **ciclo**, entonces el proyecto (representado por el grafo) no puede llevarse a cabo

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

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

Tareas posteriores a una tarea

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

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

$\boldsymbol{P} = \emptyset$

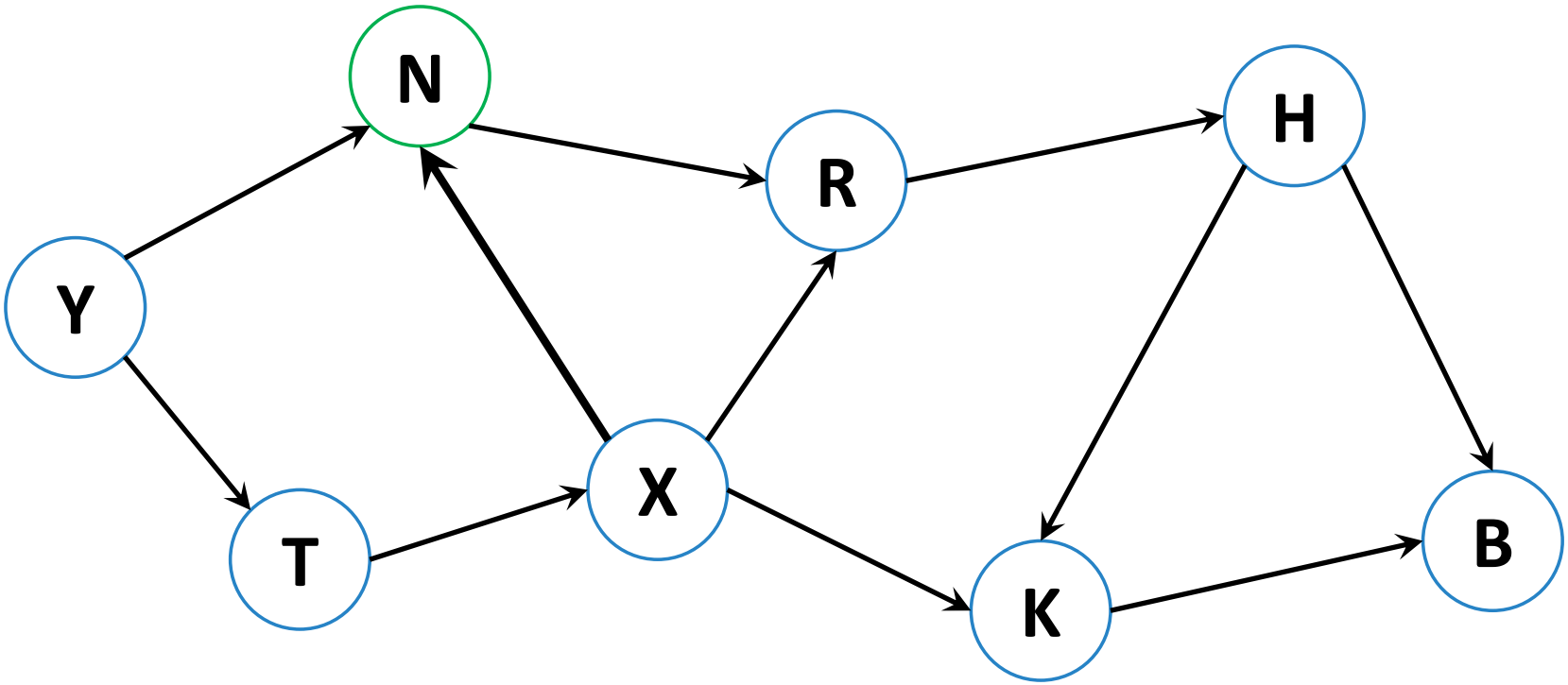
$\boldsymbol{for}\ \boldsymbol{Y}$ tal que $\boldsymbol{X} \rightarrow \boldsymbol{Y}$:

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

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

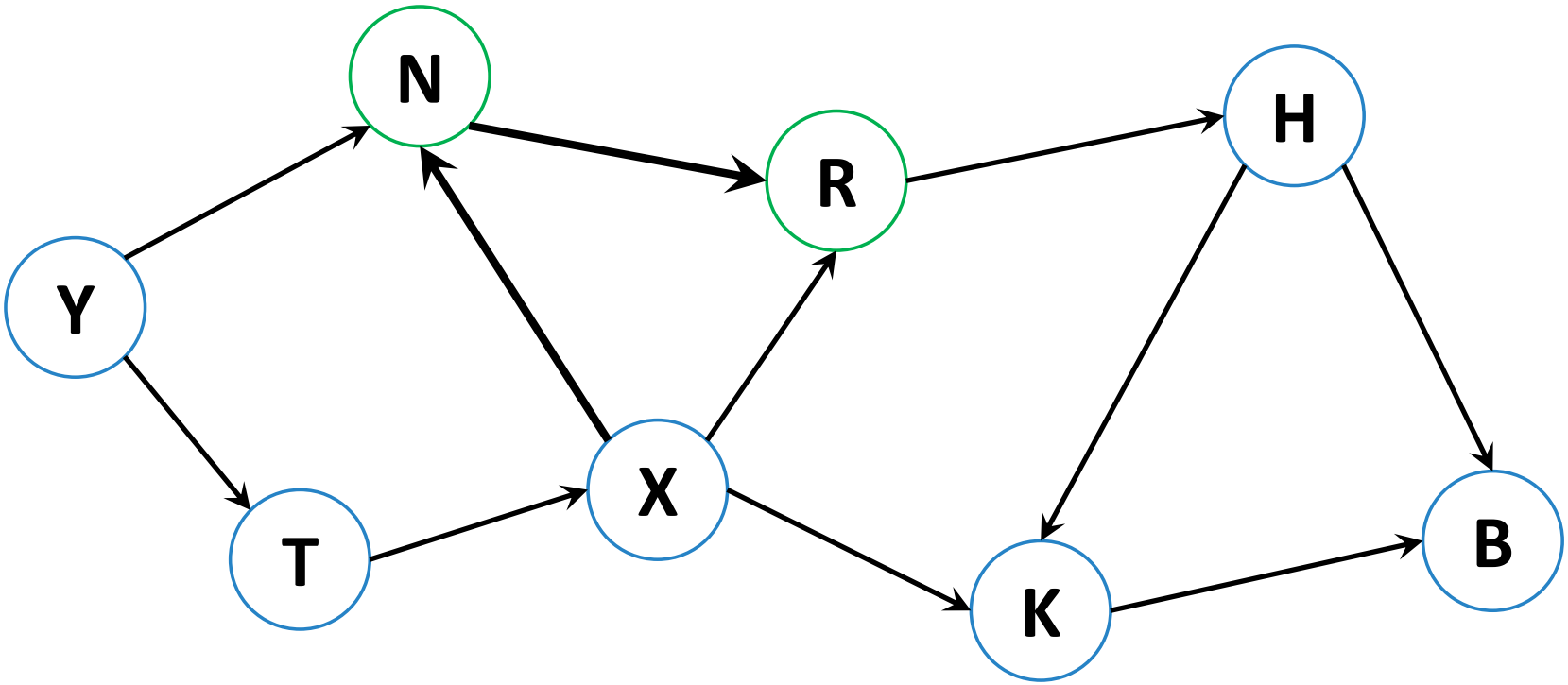
$\boldsymbol{return}\ \boldsymbol{P}$

Tareas, o nodos, posteriores



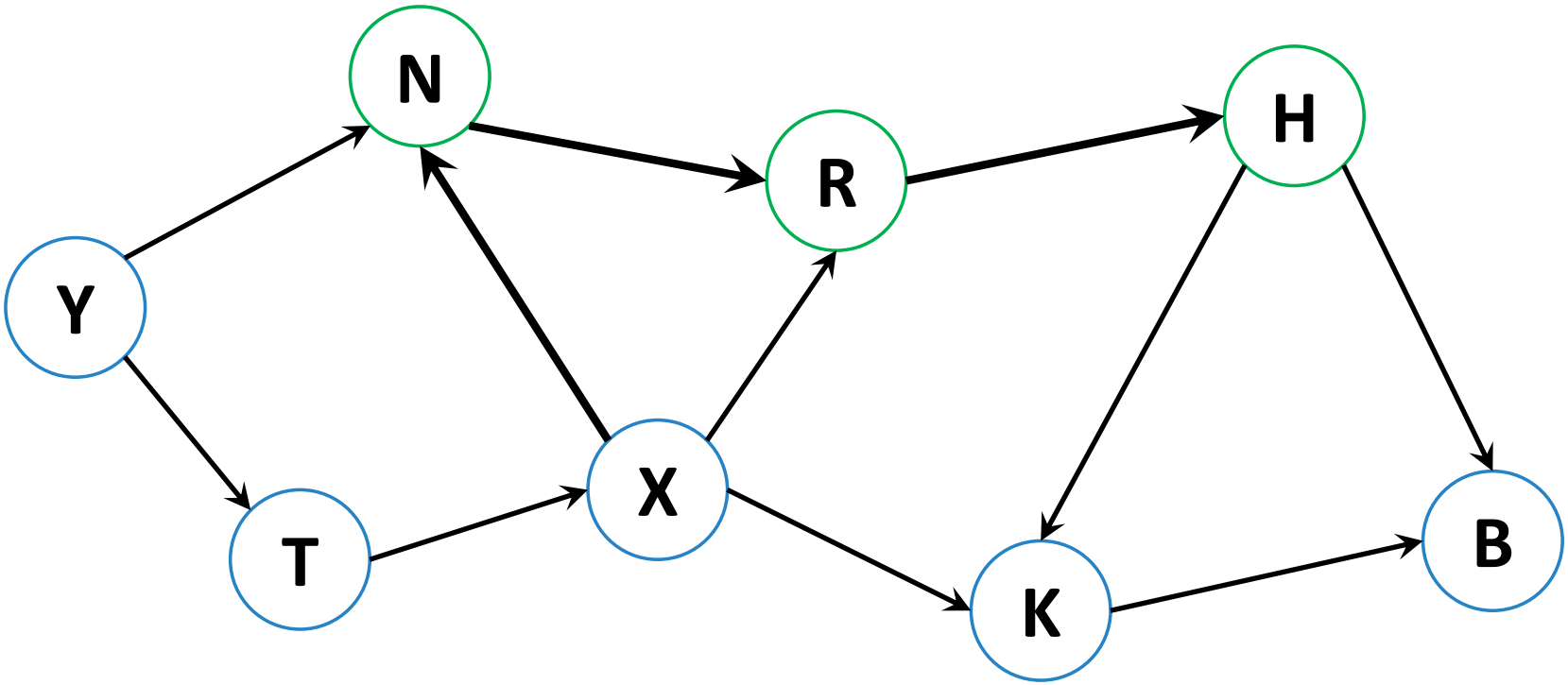
¿Cuáles nodos son posteriores a **X**?

Tareas, o nodos, posteriores



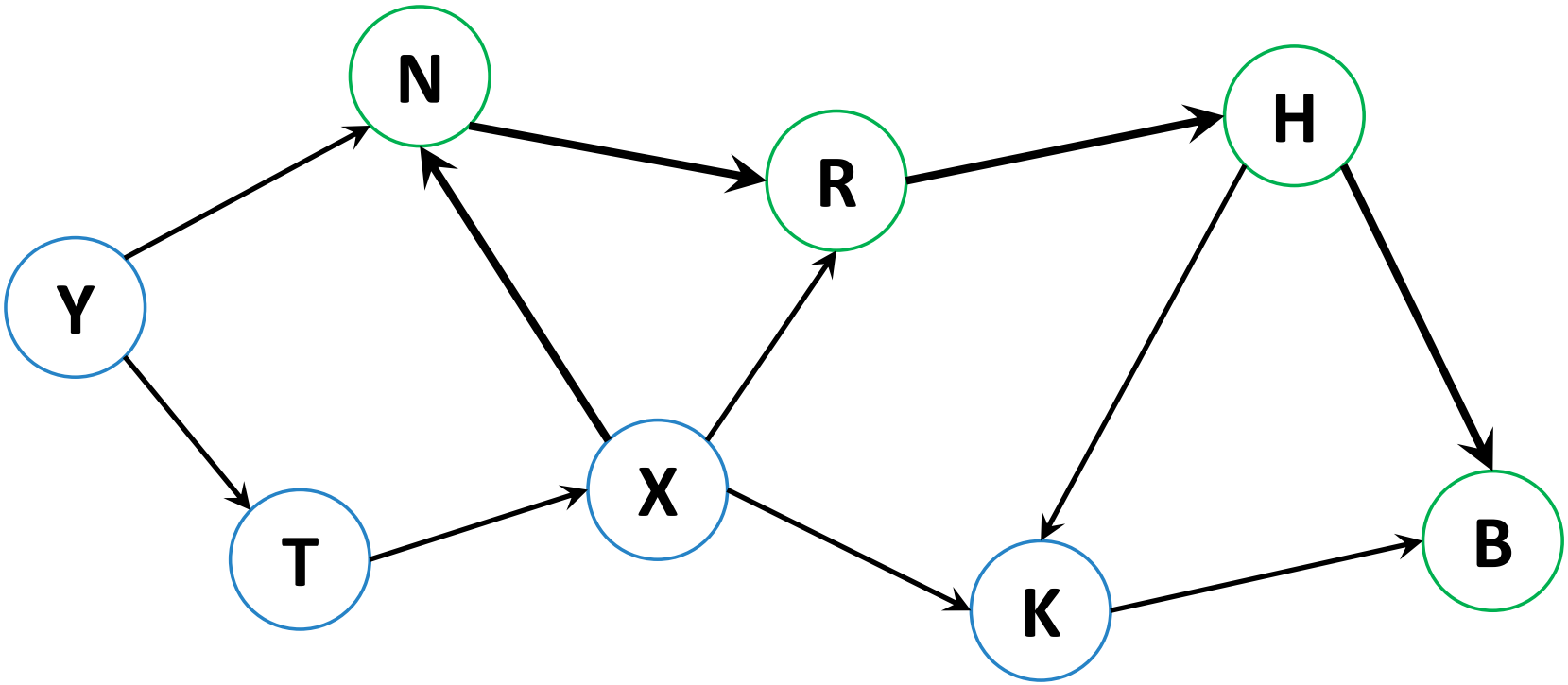
¿Cuáles nodos son posteriores a **N**?

Tareas, o nodos, posteriores



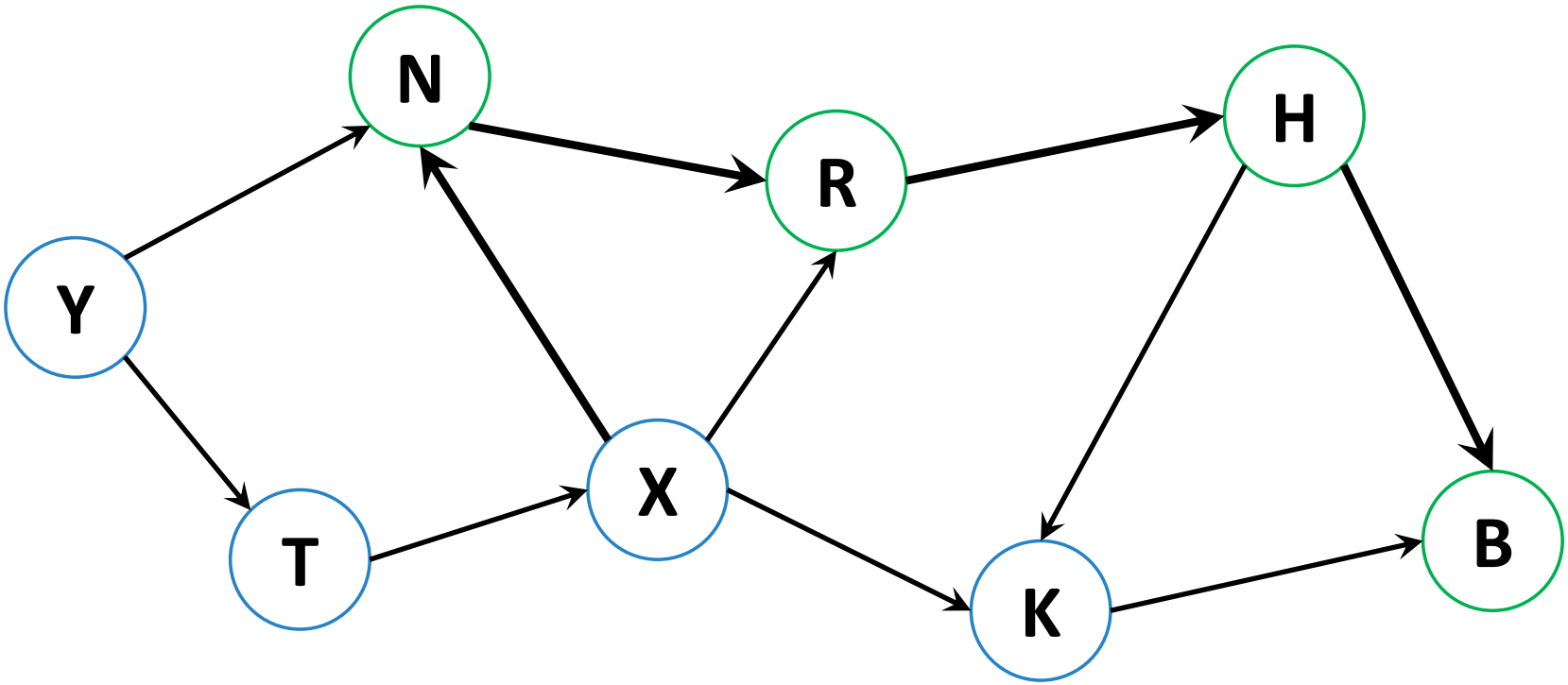
¿Cuáles nodos son posteriores a **R**?

Tareas, o nodos, posteriores



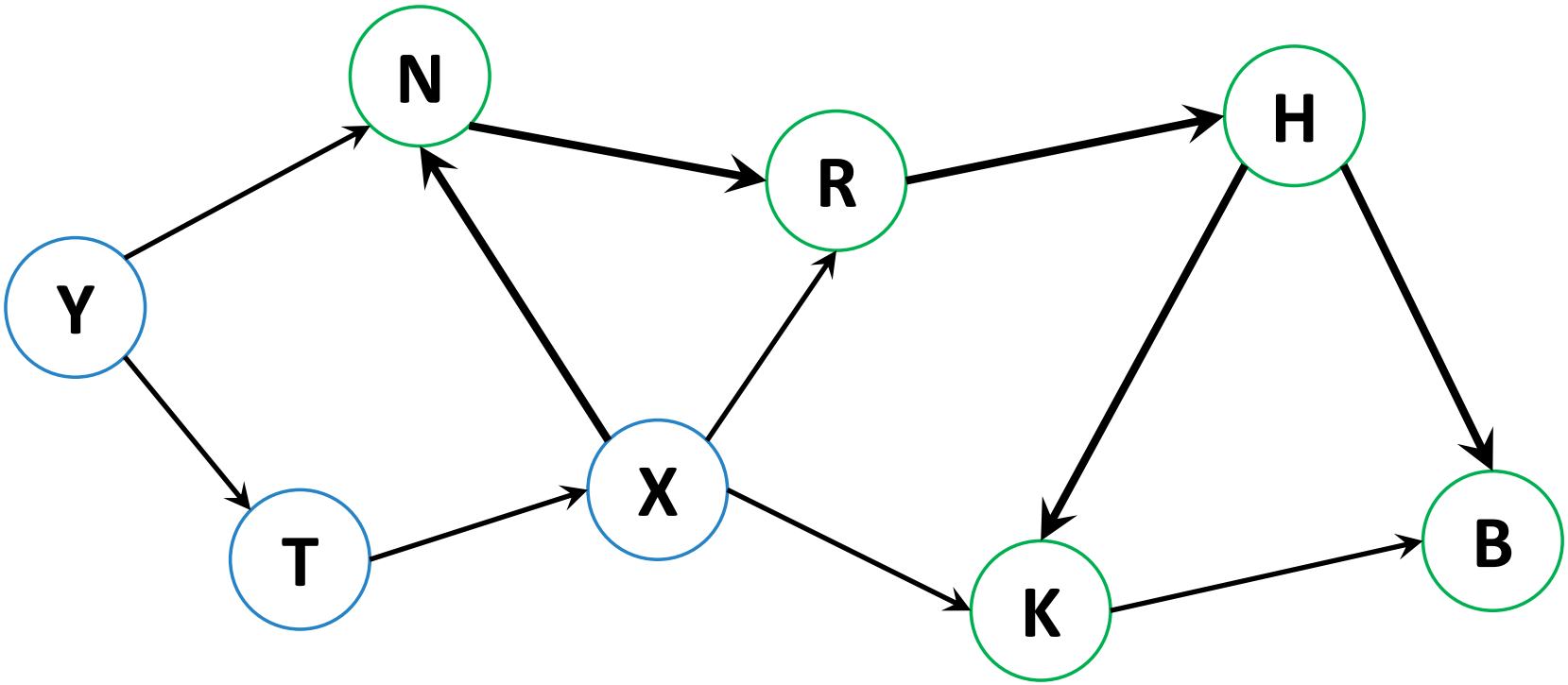
¿Cuáles nodos son posteriores a **H**?

Tareas, o nodos, posteriores



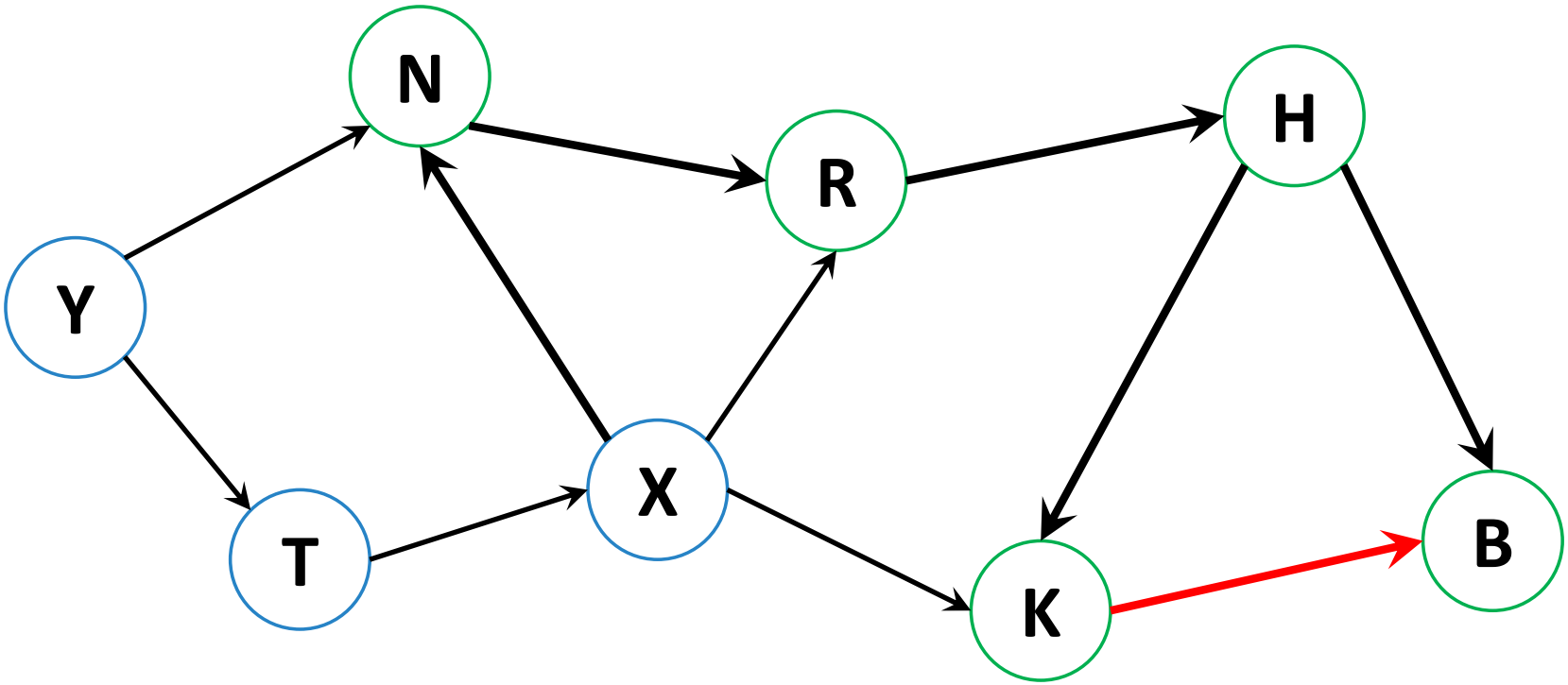
¿Cuáles nodos son posteriores a **B**?

Tareas, o nodos, posteriores



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

Tareas, o nodos, posteriores



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

Nodos por los que ya pasamos

Estamos haciendo llamadas repetidas

Es más, si pasamos por un ciclo, entonces el algoritmo no termina

¿Cómo se soluciona esto?

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

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

pintar $\boldsymbol{X}$

$\boldsymbol{P} = \emptyset$

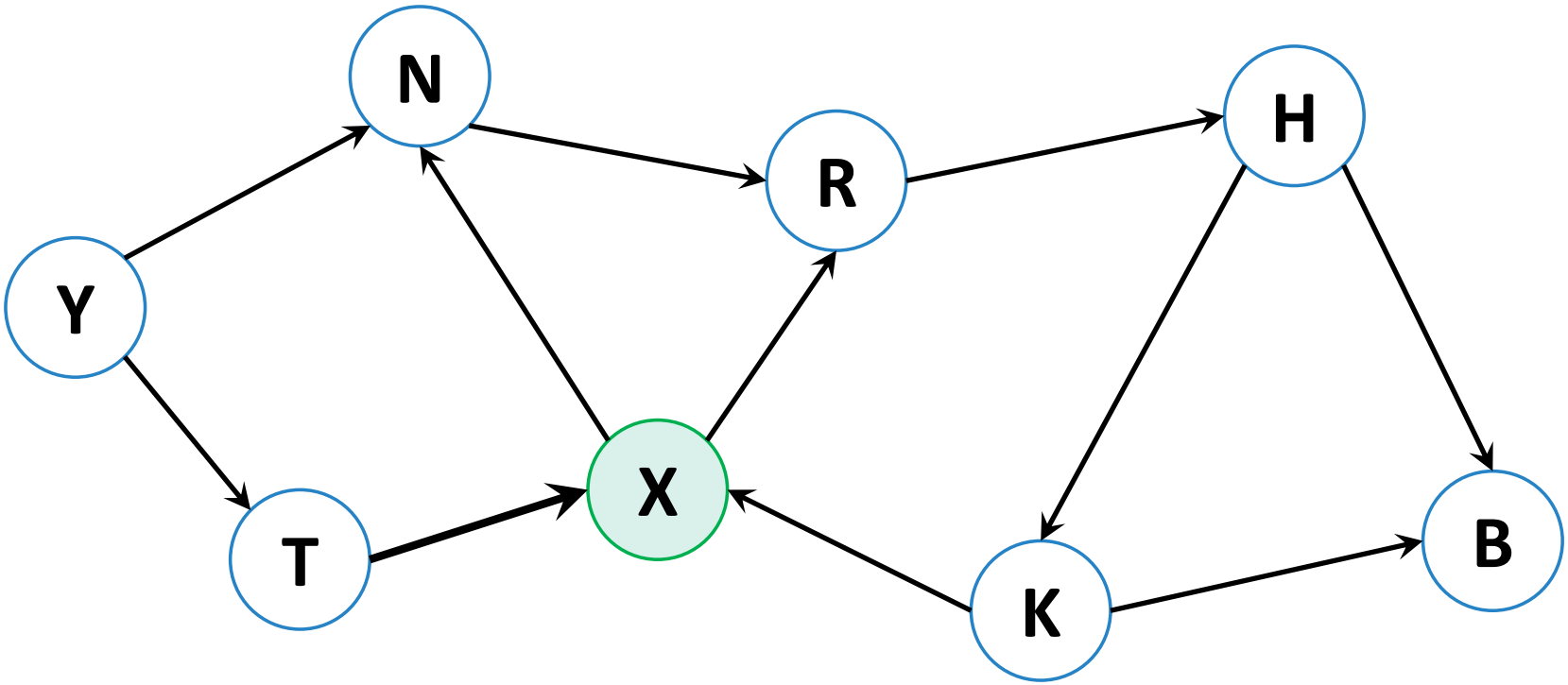
$\boldsymbol{for}$ $\boldsymbol{Y}$ tal que $\boldsymbol{X} \rightarrow \boldsymbol{Y}$:

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

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

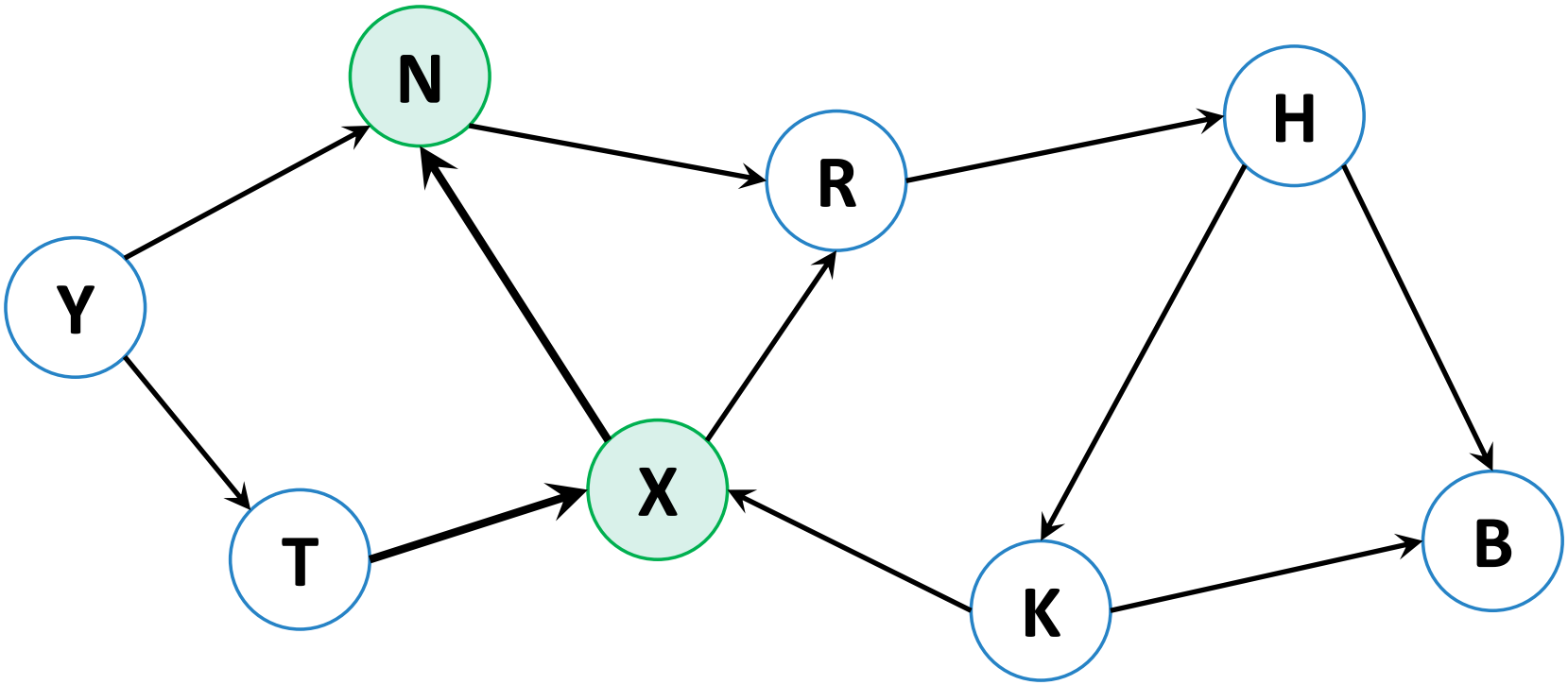
$\boldsymbol{return}$ $\boldsymbol{P}$

Nodos posteriores



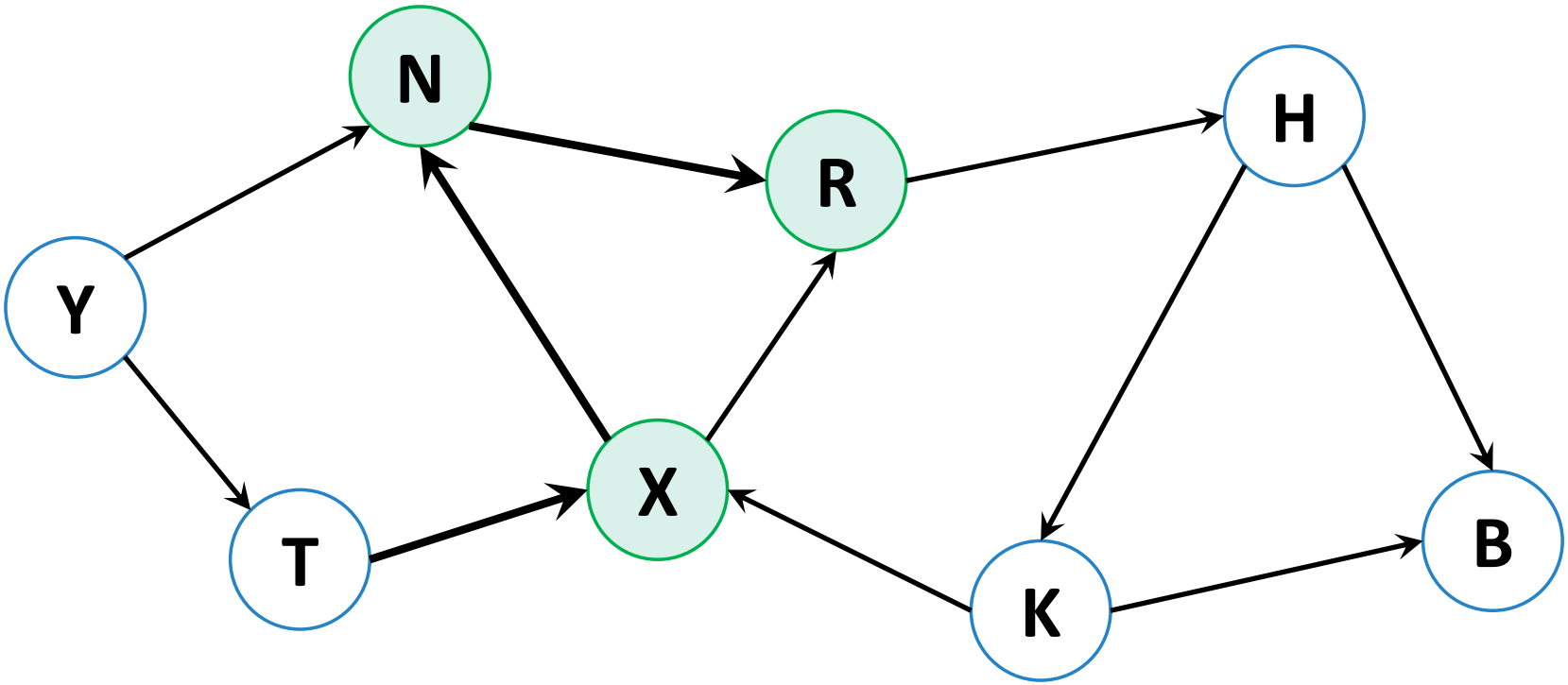
¿Cuáles nodos son posteriores a **T**?

Nodos posteriores



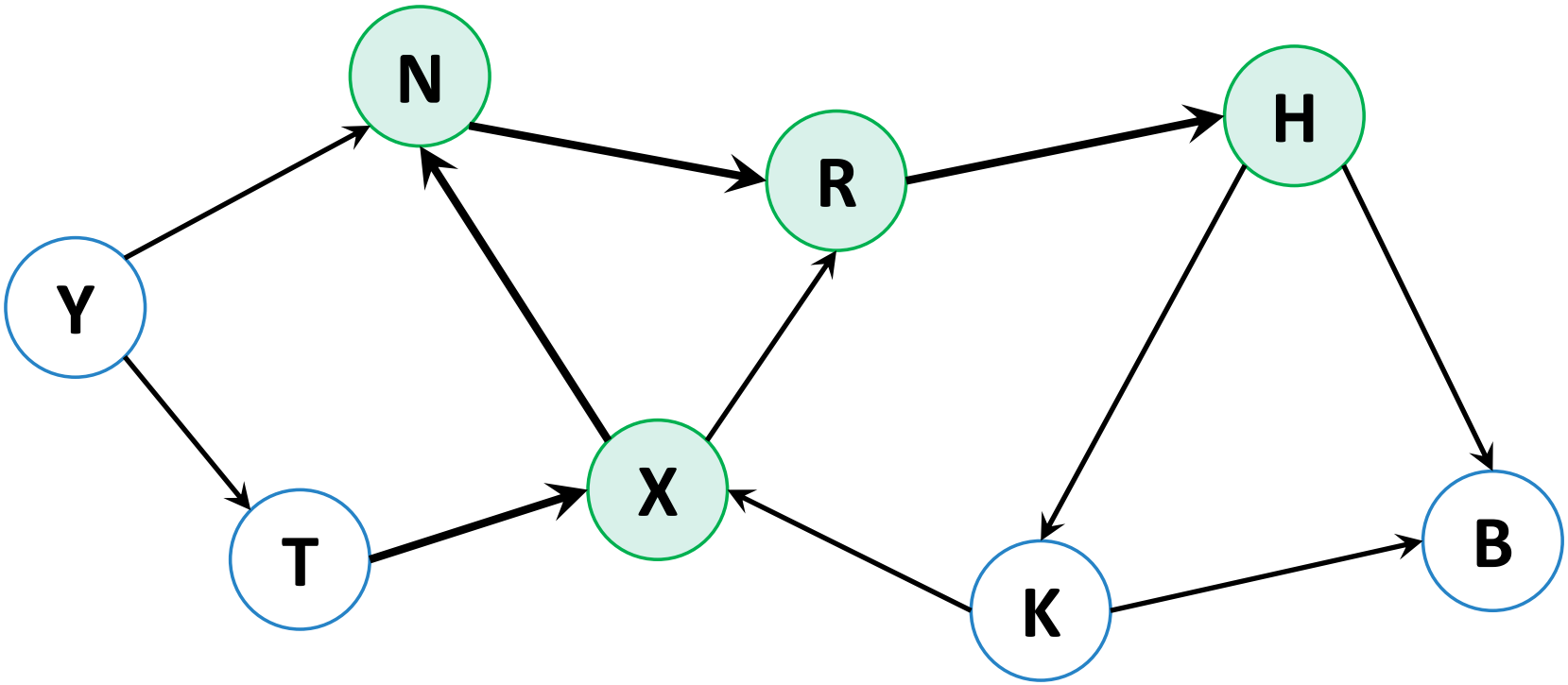
¿Cuáles nodos son posteriores a **X**?

Nodos posteriores



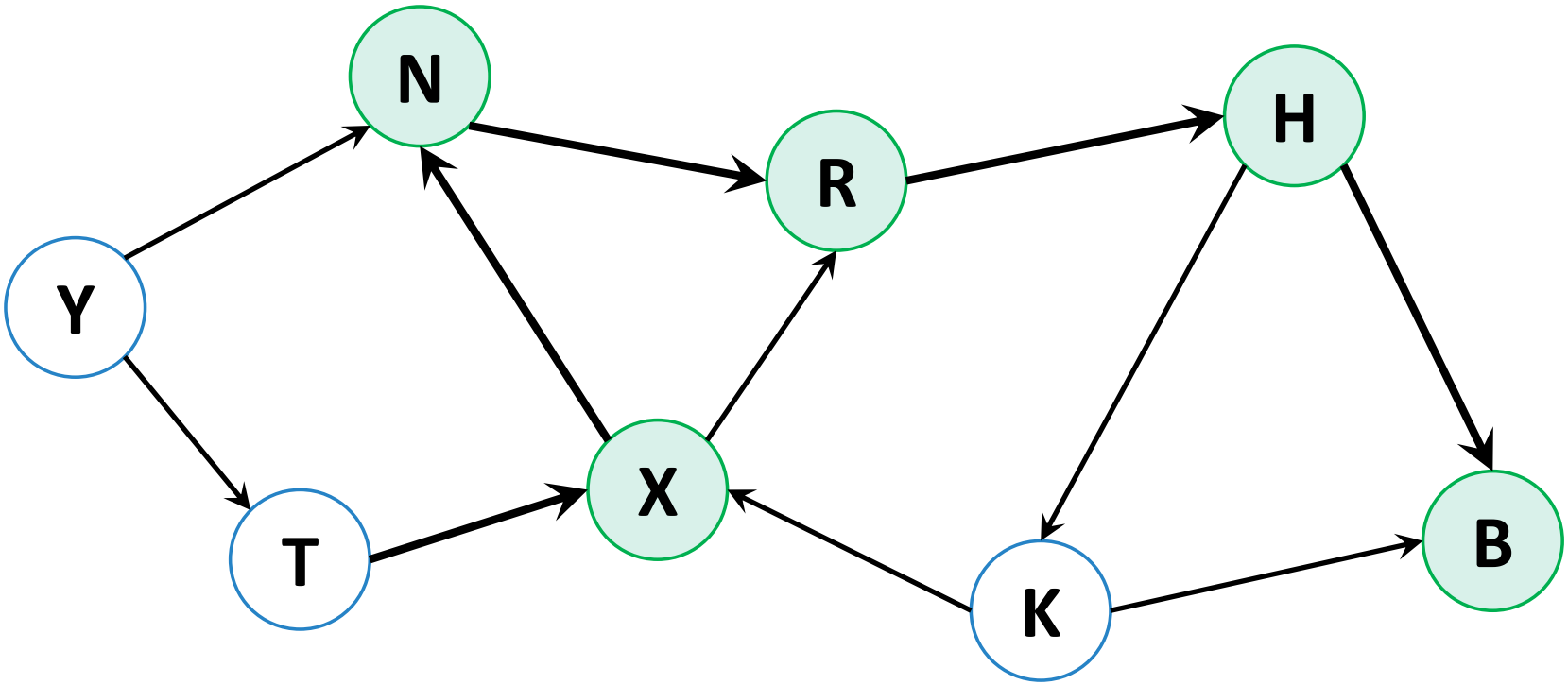
¿Cuáles nodos son posteriores a **N**?

Nodos posteriores



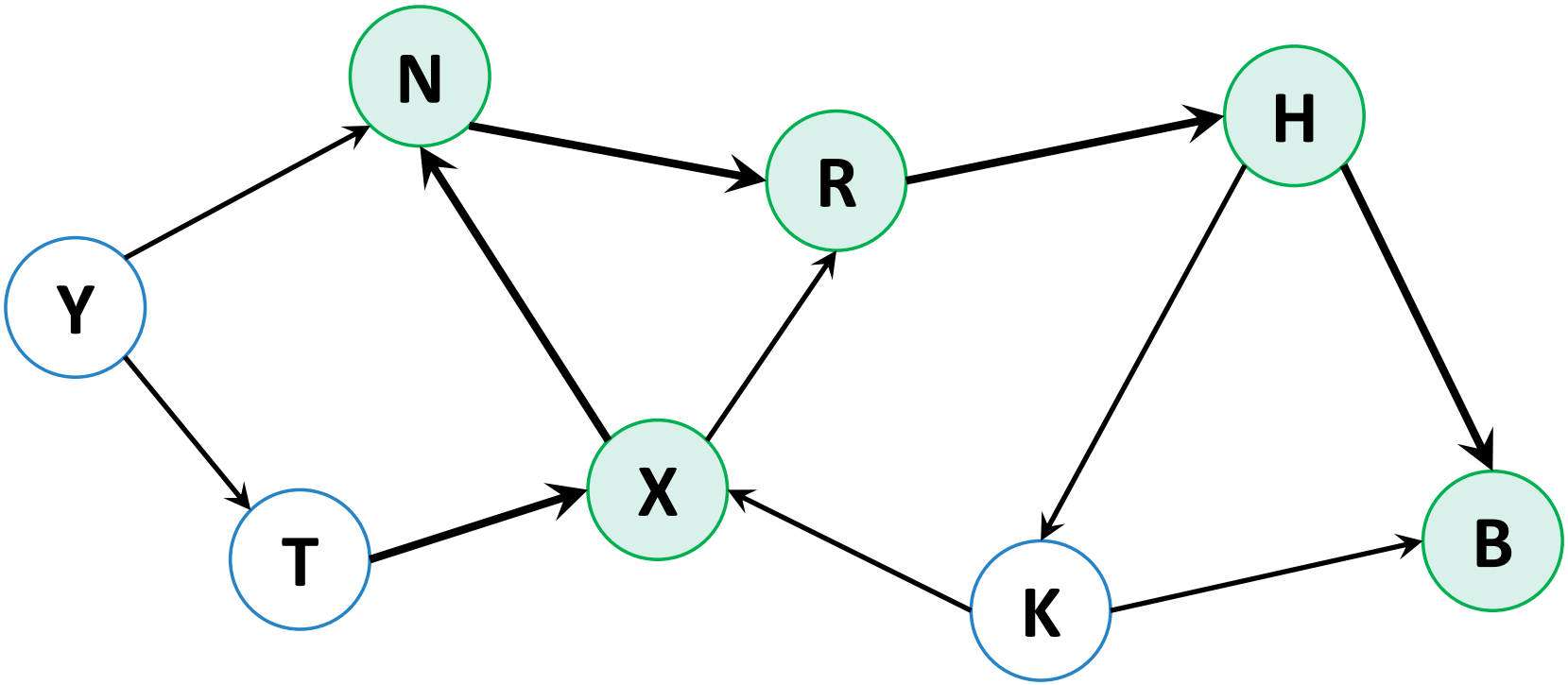
¿Cuáles nodos son posteriores a **R**?

Nodos posteriores



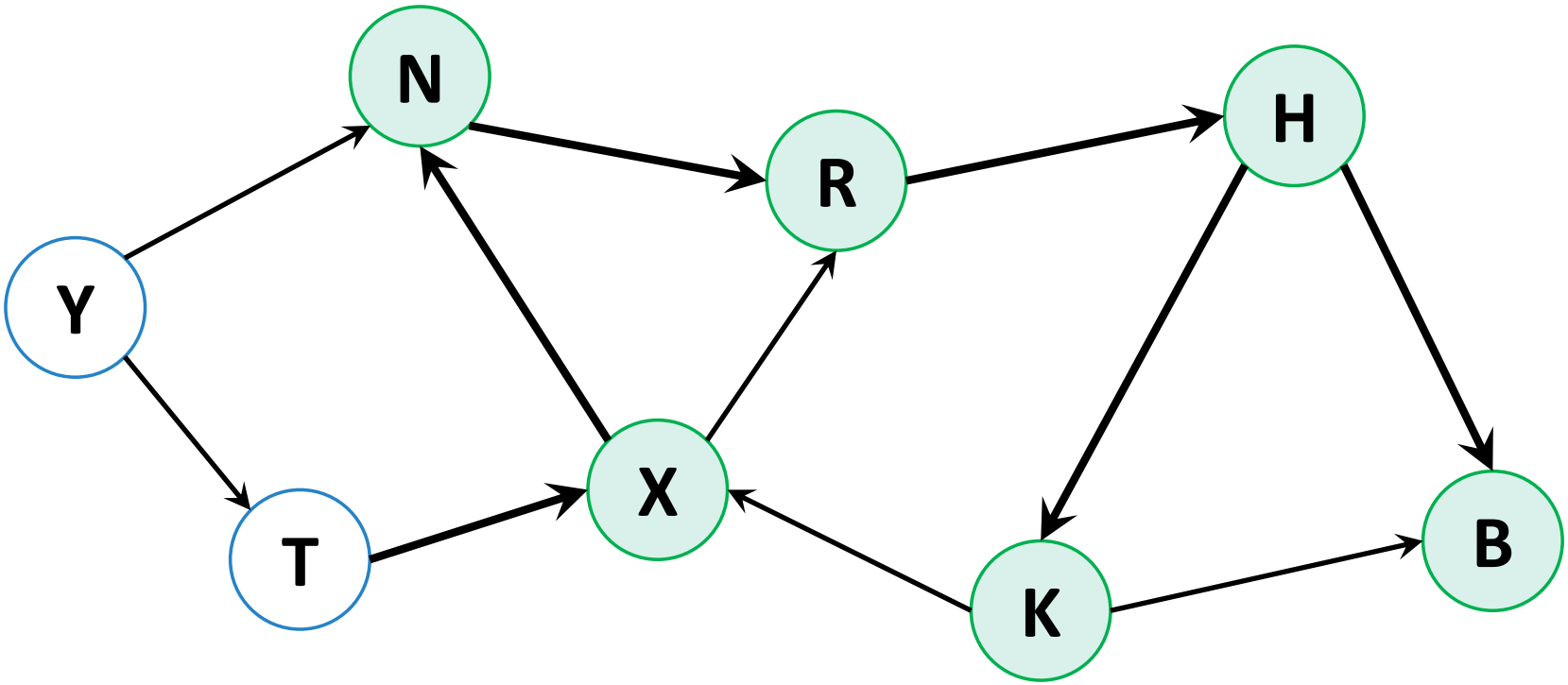
¿Cuáles nodos son posteriores a **H**?

Nodos posteriores



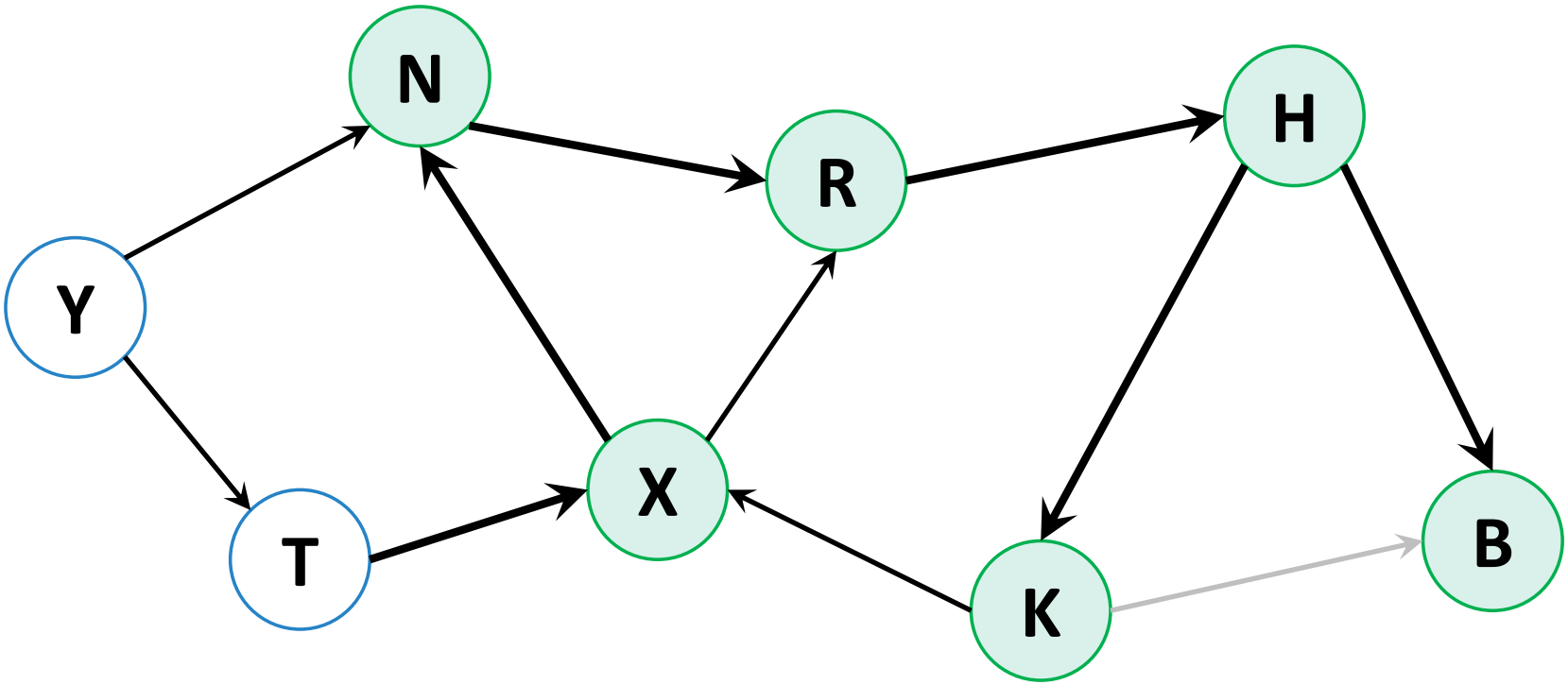
¿Cuáles nodos son posteriores a **B**?

Nodos posteriores



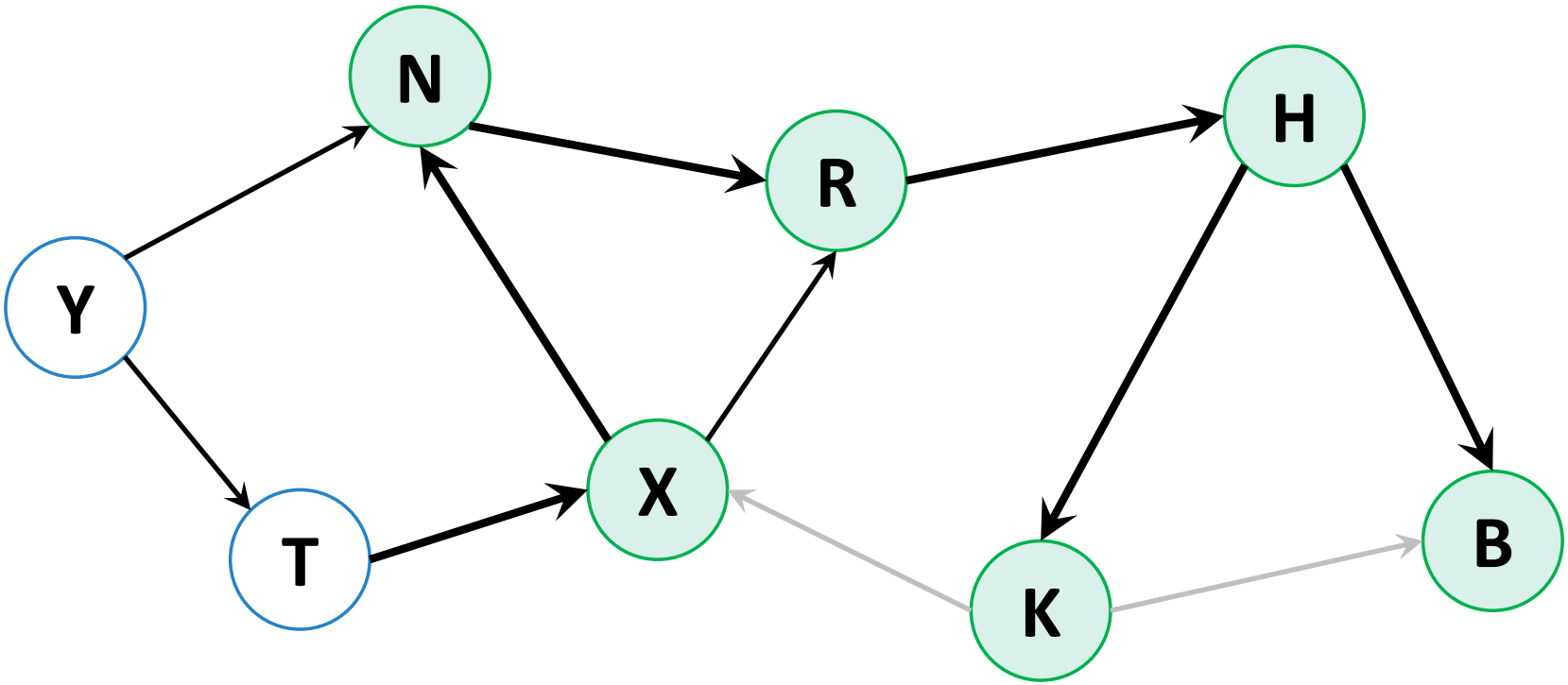
¿Cuáles nodos son posteriores a **H**?

Nodos posteriores



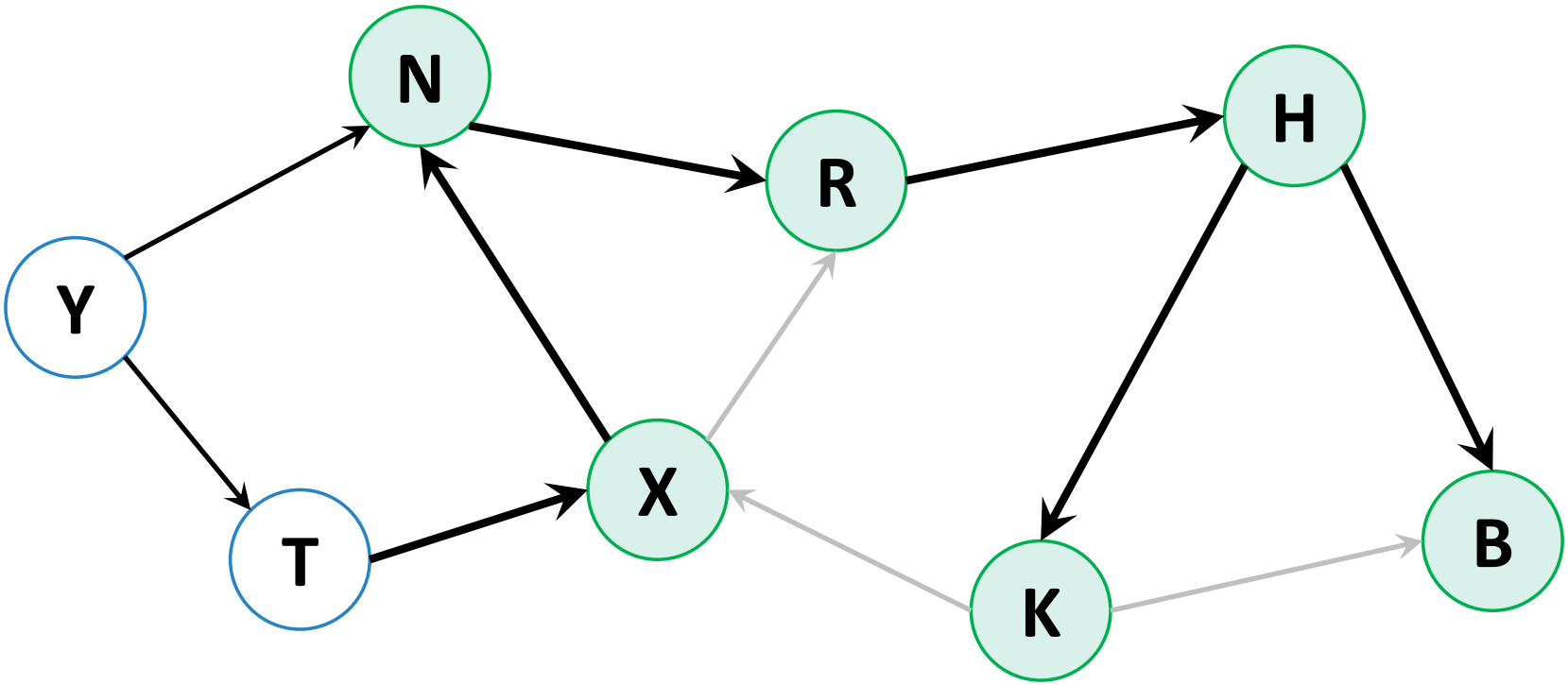
¿Cuáles nodos son posteriores a **K**?

Nodos posteriores



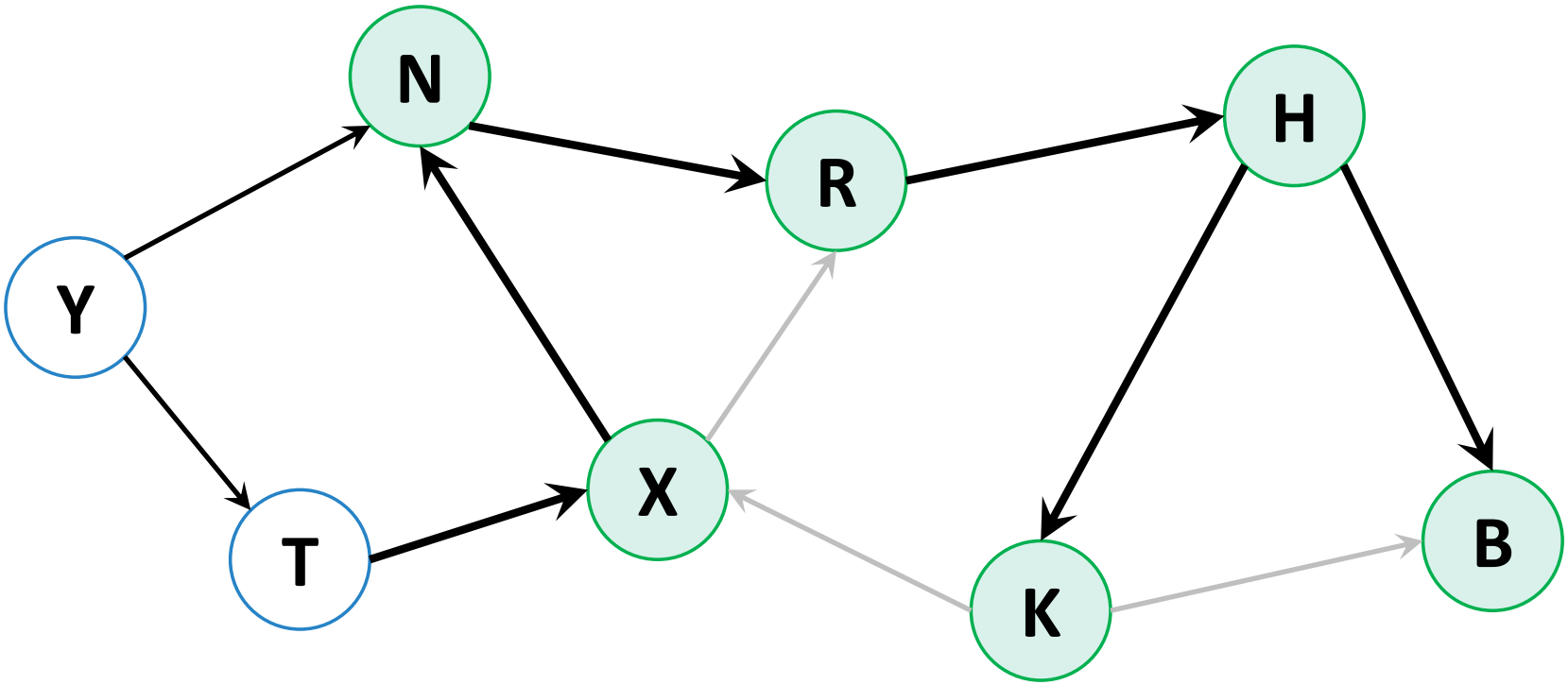
¿Cuáles nodos son posteriores a **K**?

Nodos posteriores



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

Nodos posteriores



¡Listo!

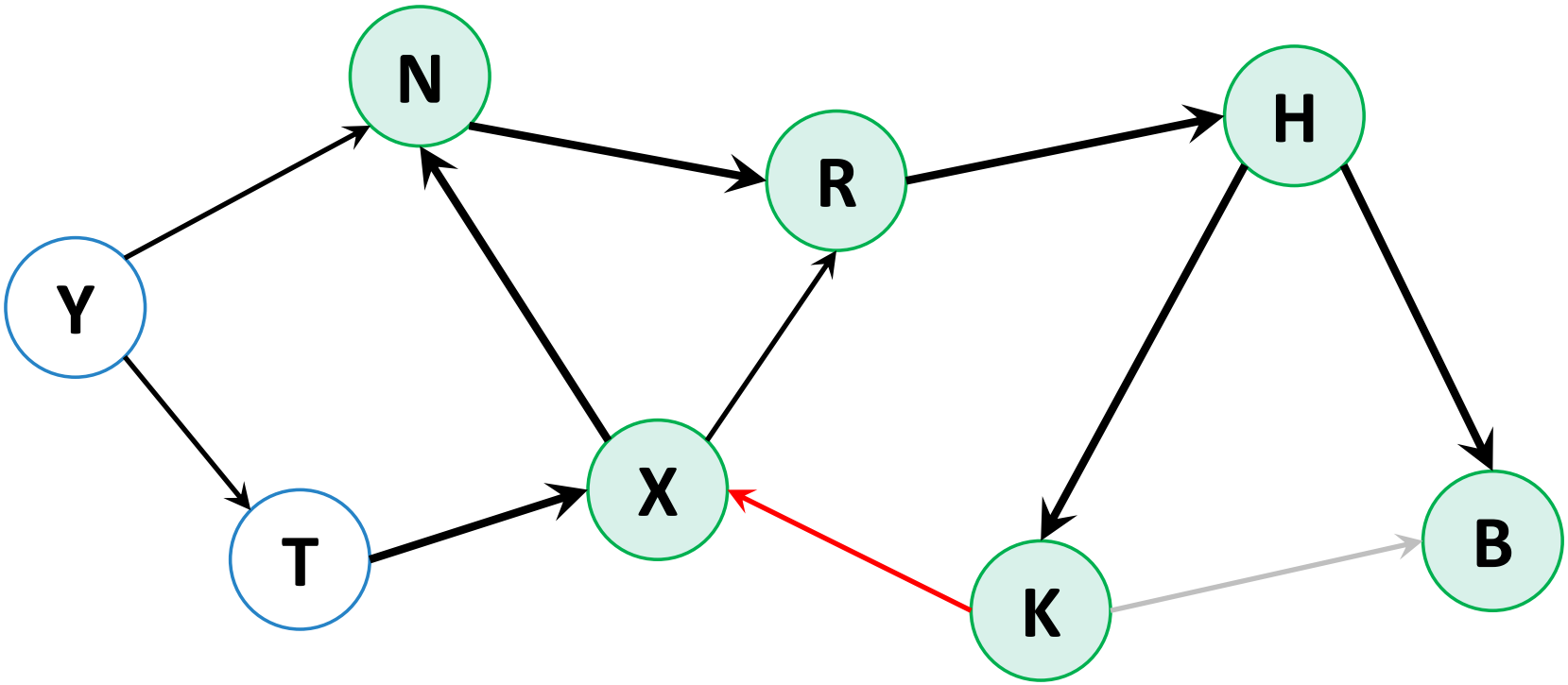
¿Cómo identificamos ciclos?

Ok, podemos identificar las tareas posteriores a una tarea

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

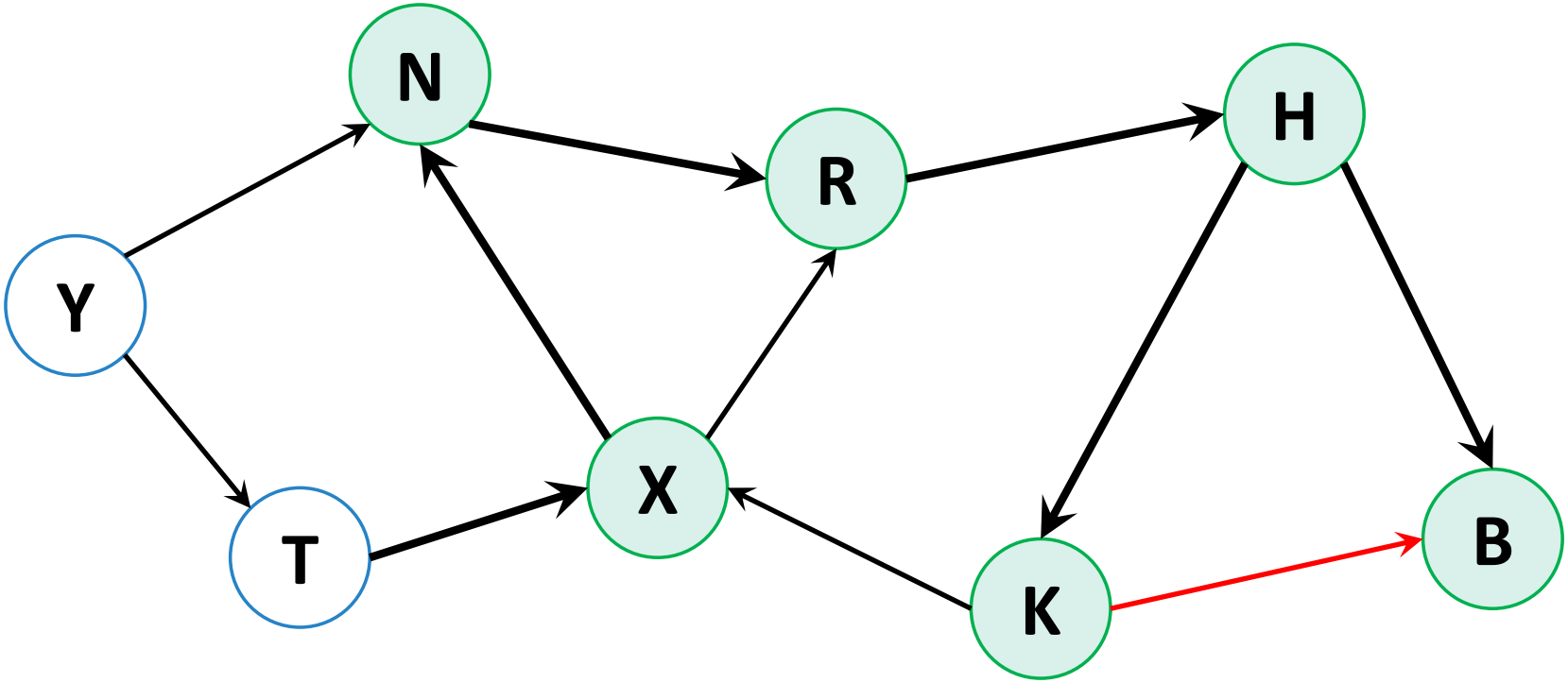
¿Podemos usar este enfoque para identificar ciclos?

Algo así como esto



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

… ¿y como esto?



¡Y que esto otro no!

Observación

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

- Si lo descubrió un nodo posterior a Y, hay ciclo
- Si lo descubrió un nodo anterior a Y, no hay ciclo (aún)

Hasta que $\boldsymbol{posteriores}(X)$ retorne, todos los nodos explorados son posteriores a X

hay ciclo (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 $X \rightarrow Y$:

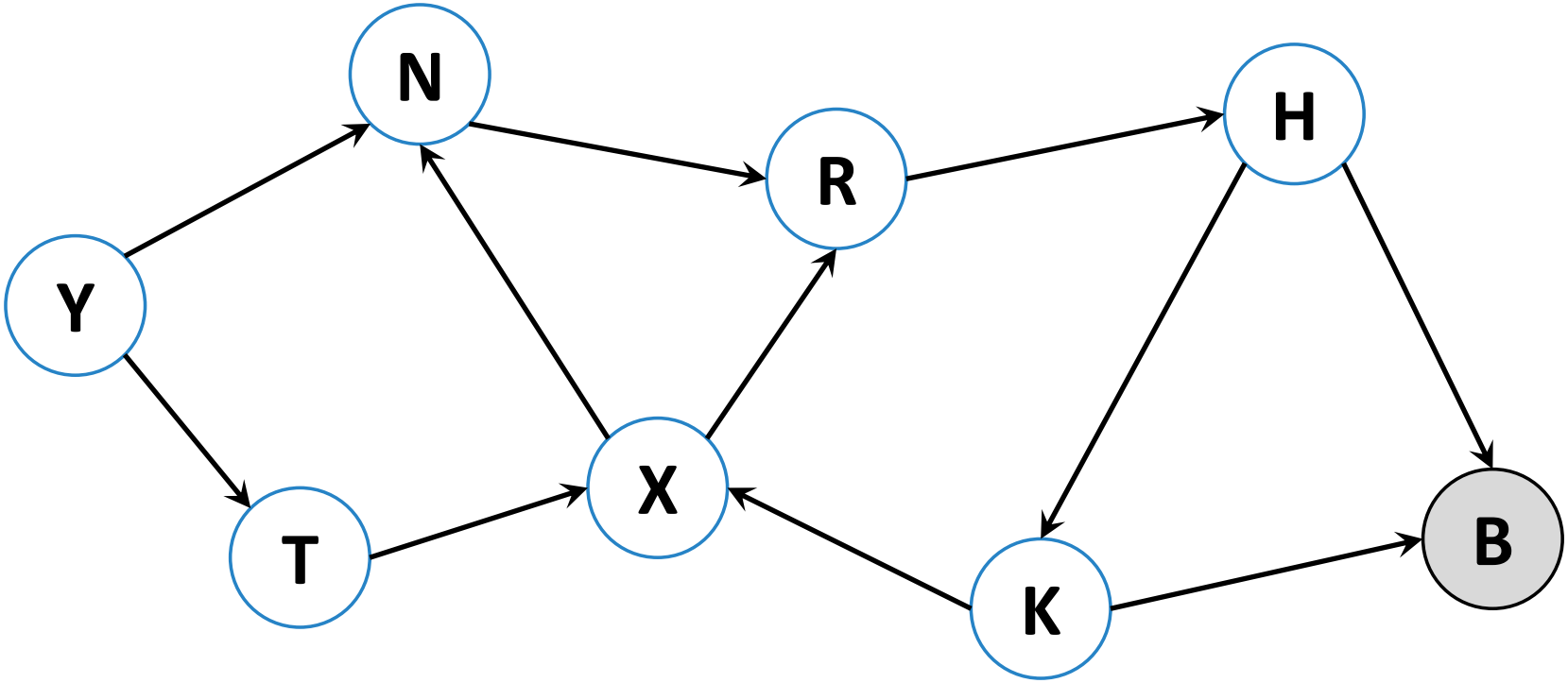
if hay ciclo (Y), ***return true***

Pintar X de negro

return false

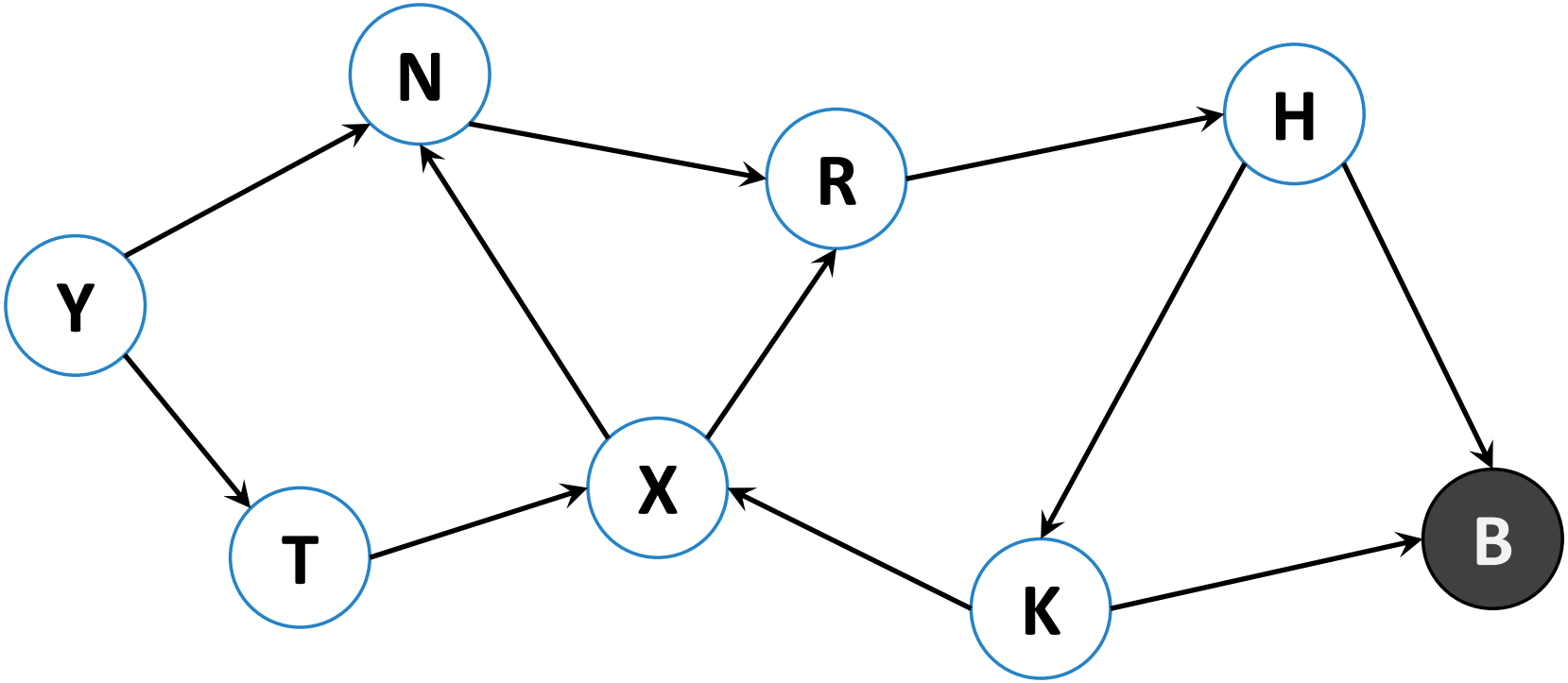
```
hay ciclo en(G(V, E)):
    for X ∈ V:
        if X está pintado, continue
        if hay ciclo (X):
            return true
    return false
```

El algoritmo *hay ciclo en* en acción



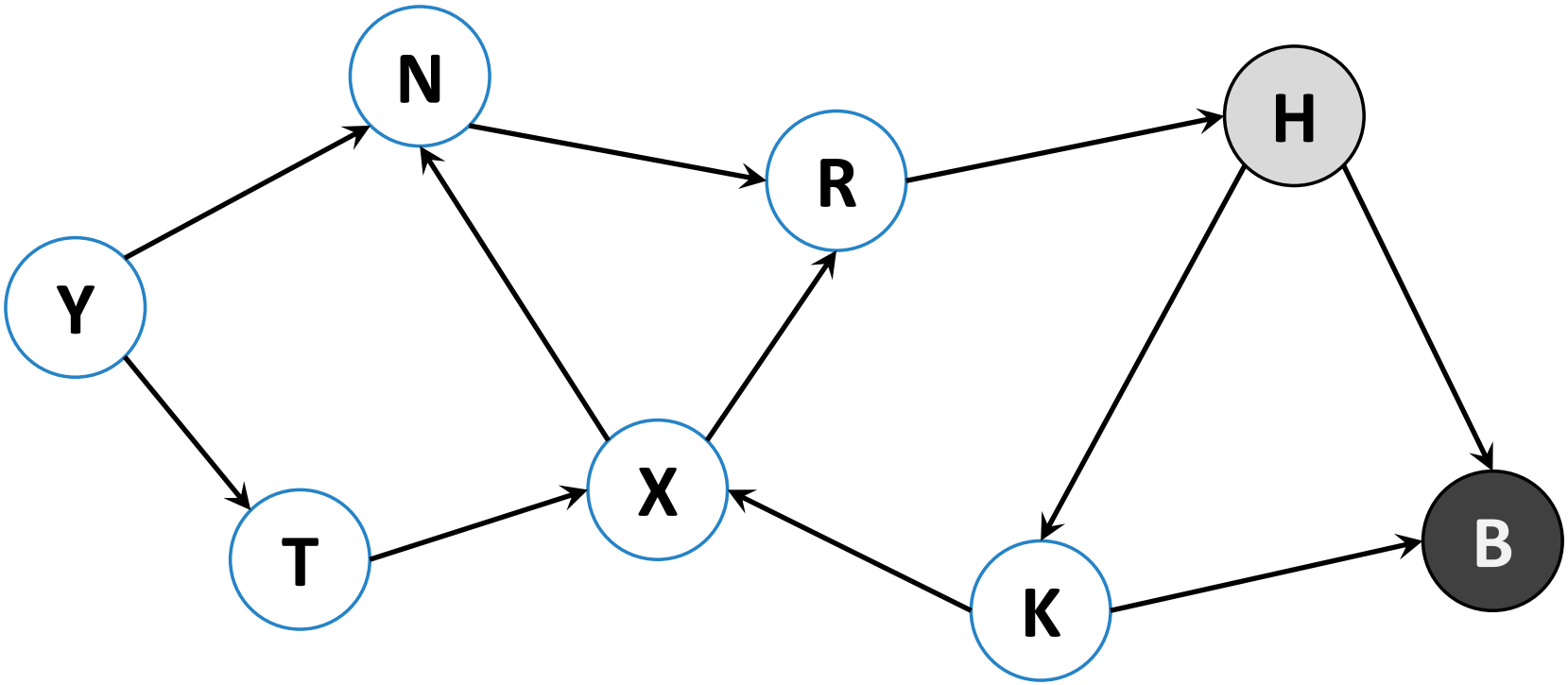
¿Hay un ciclo luego de **B**?

El algoritmo en acción



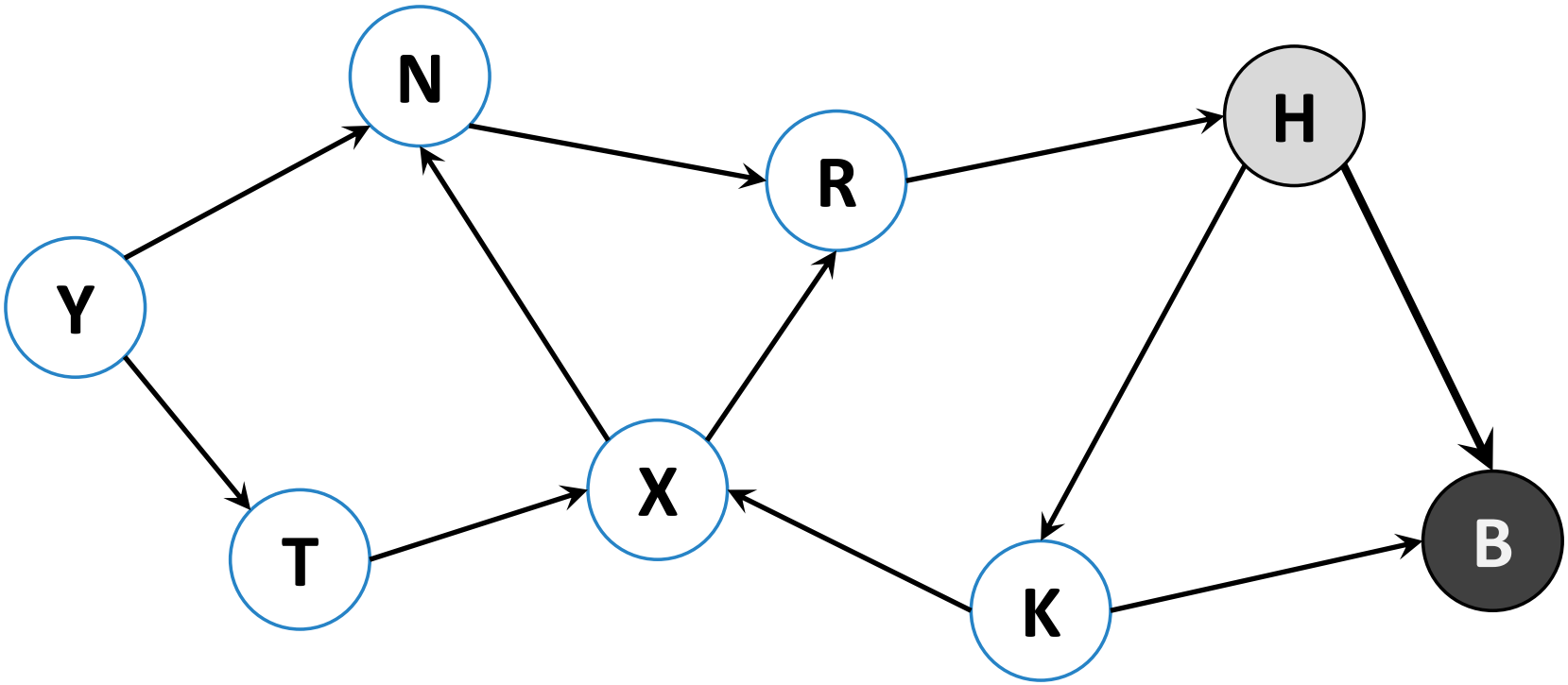
No

El algoritmo en acción



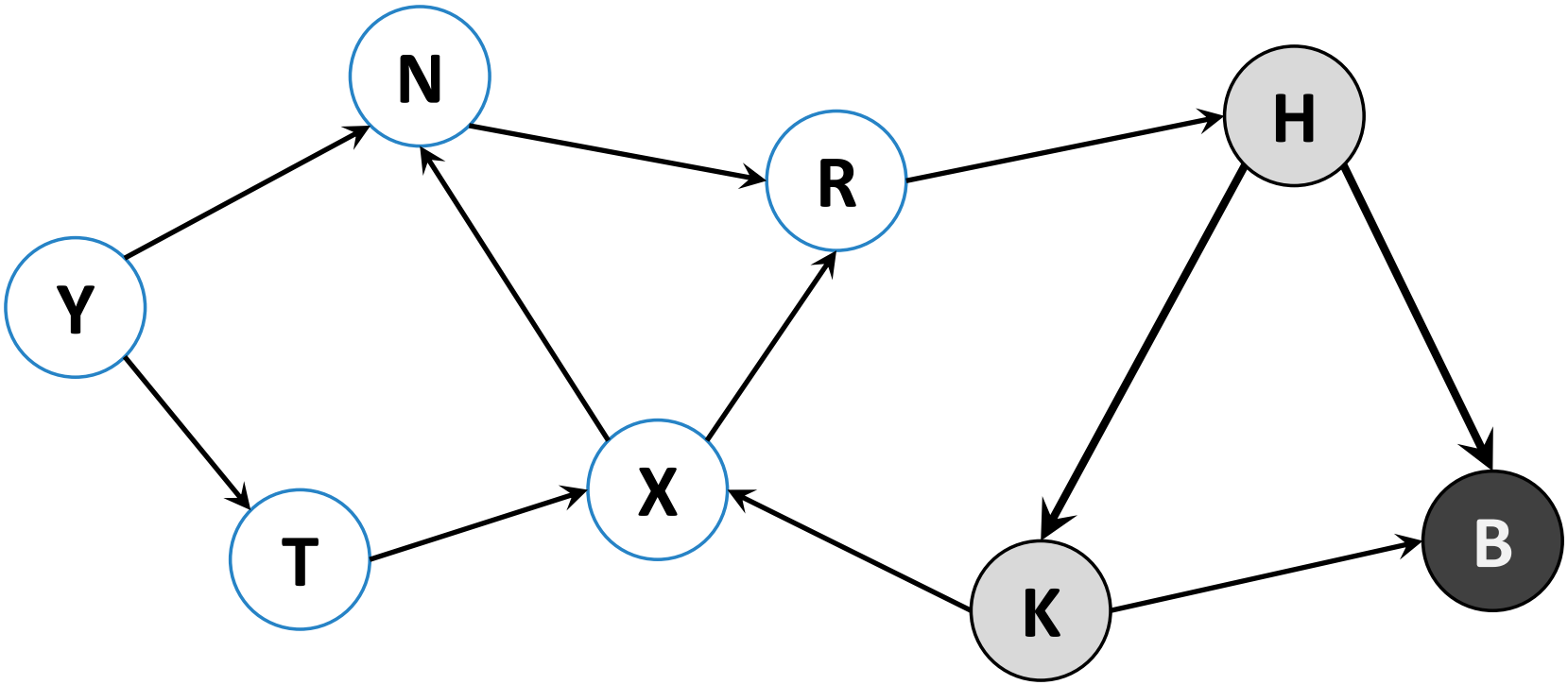
OK, ¿hay un ciclo luego de **H**?

El algoritmo en acción



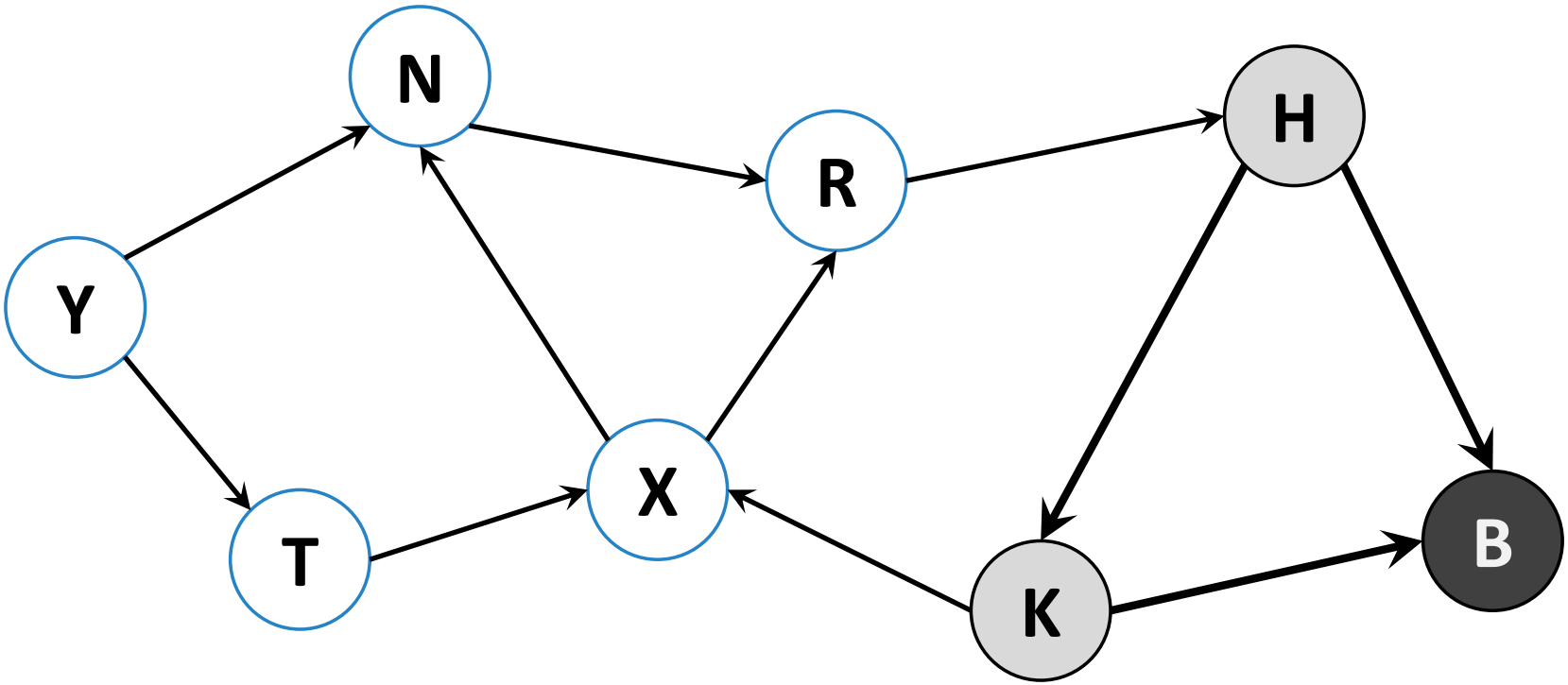
¿Hay un ciclo luego de **H**?

El algoritmo en acción



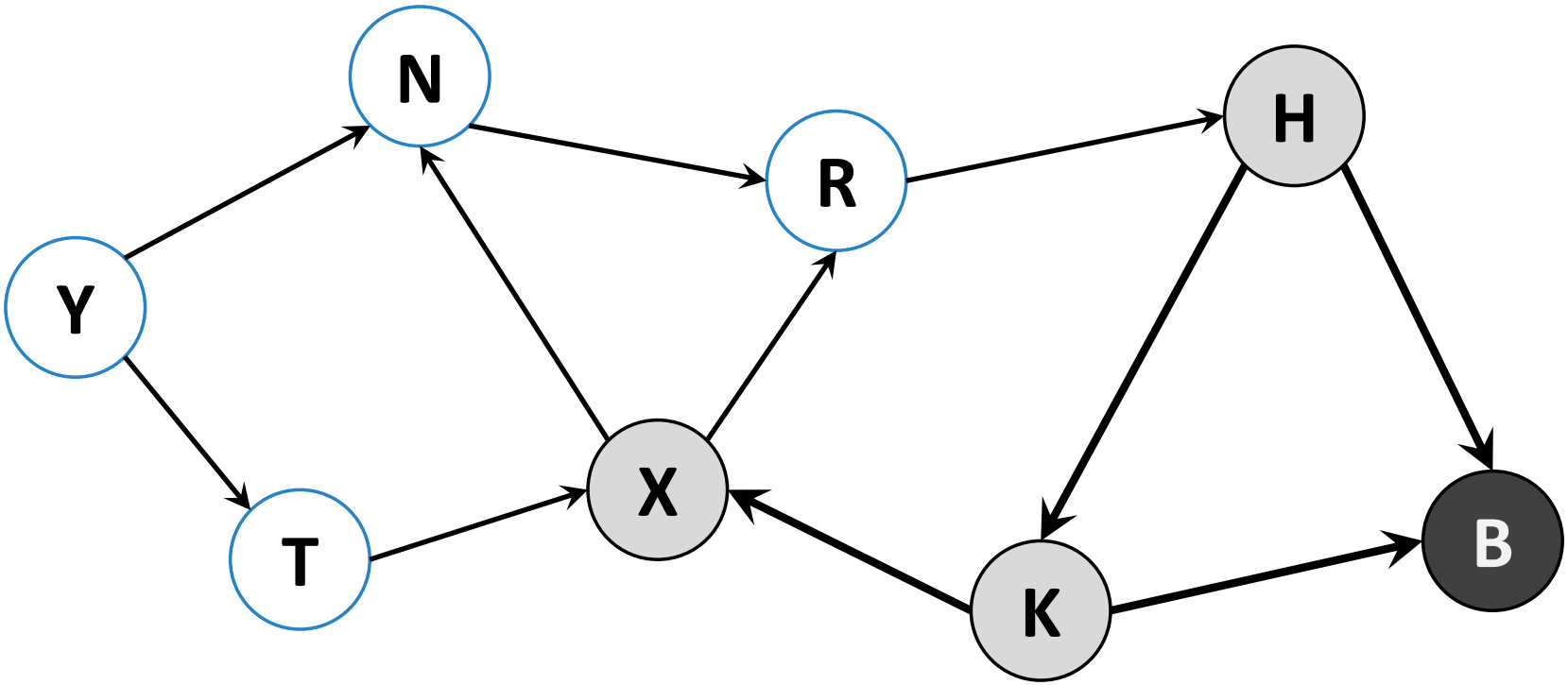
¿Hay un ciclo luego de **K**?

El algoritmo en acción



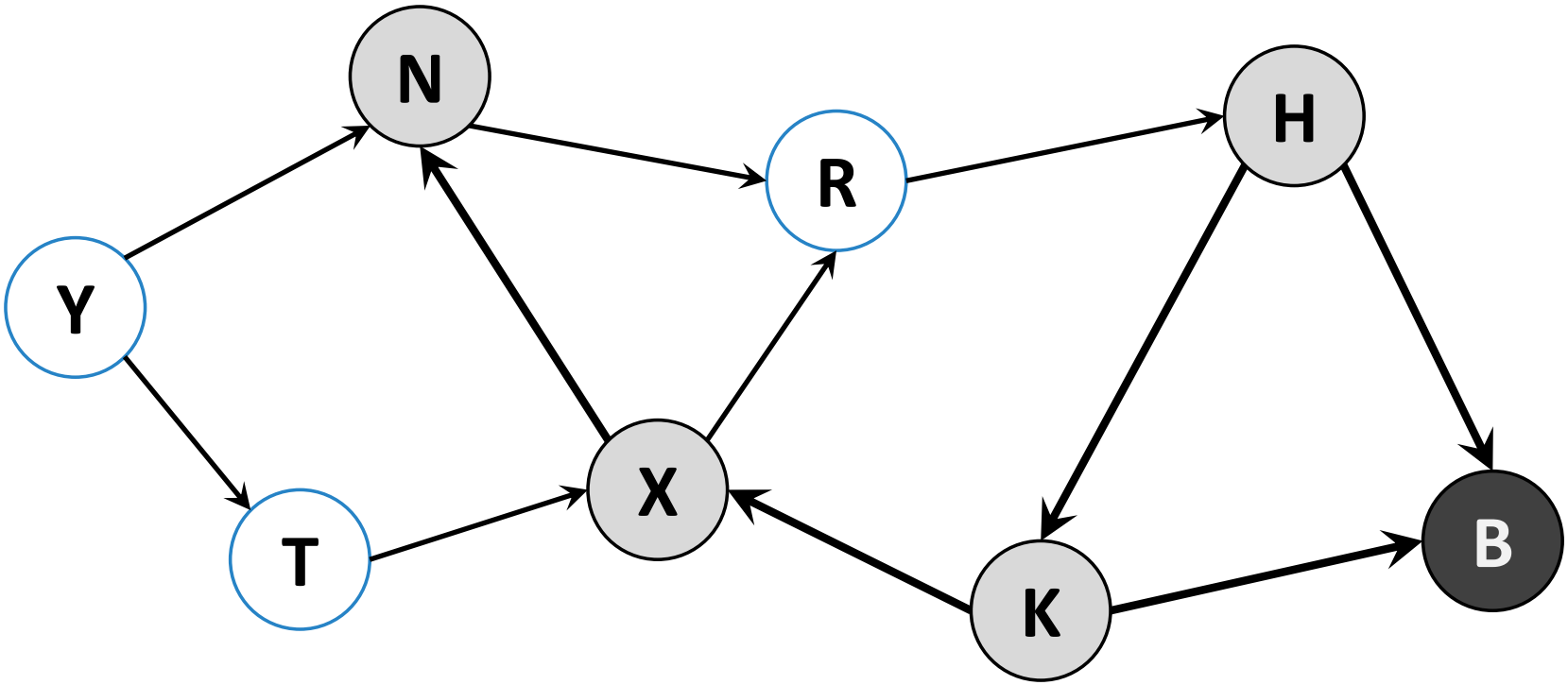
¿Hay un ciclo luego de **K**?

El algoritmo en acción



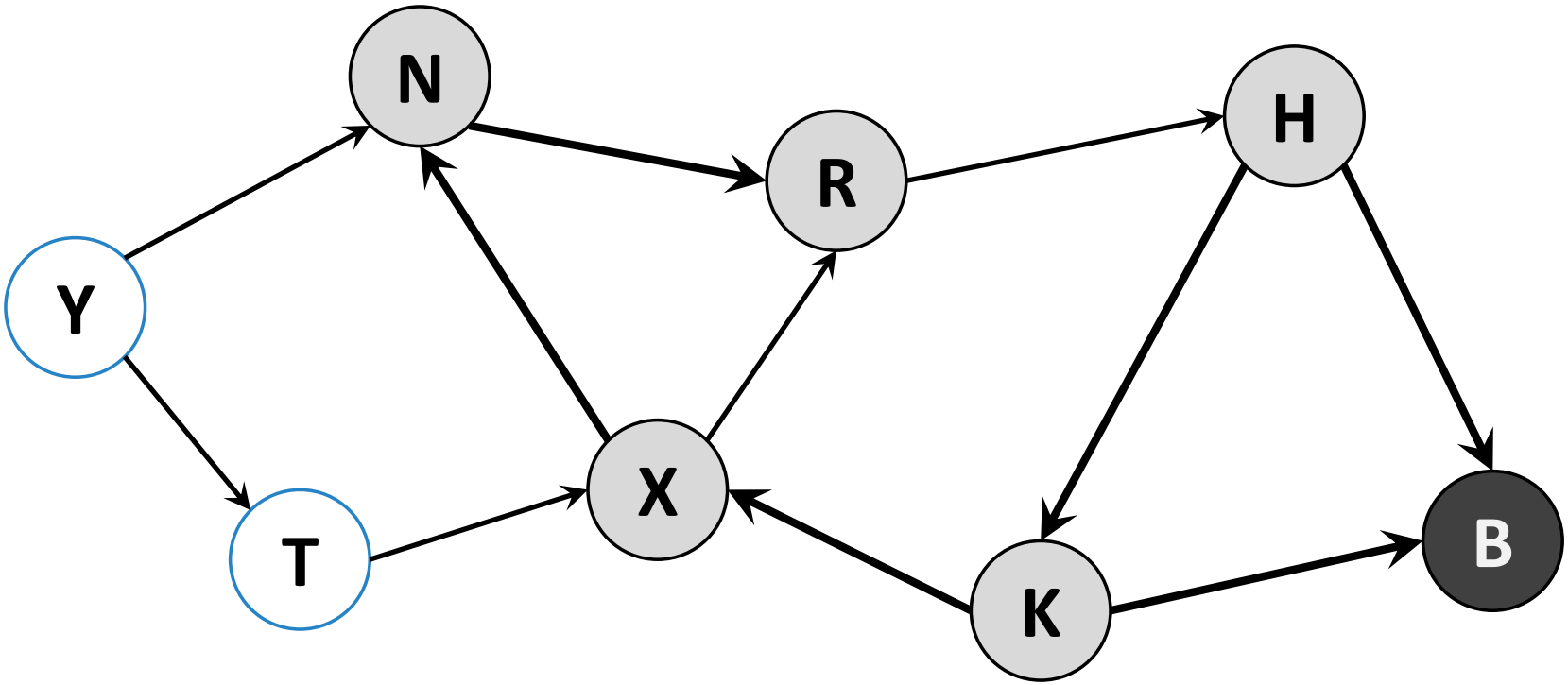
¿Hay un ciclo luego de **X**?

El algoritmo en acción



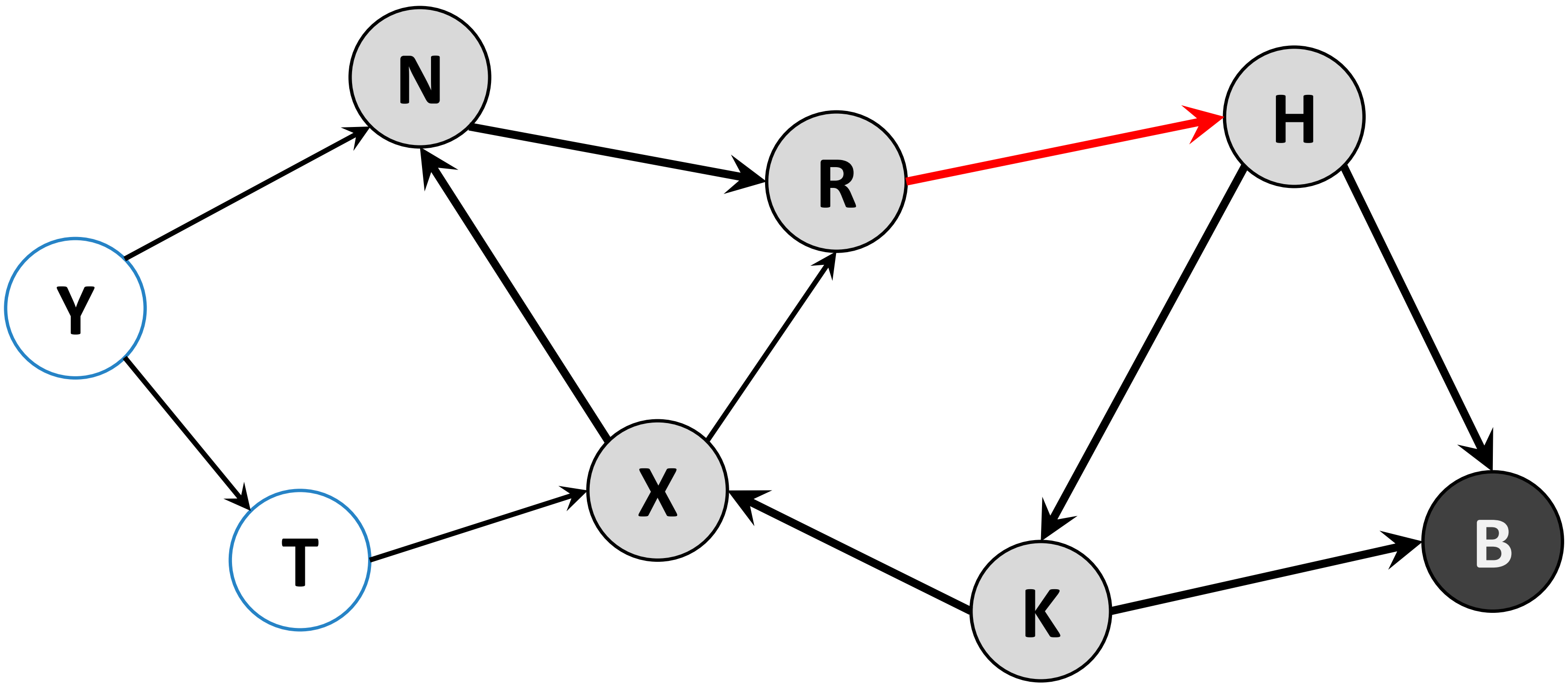
¿Hay un ciclo luego de **N**?

El algoritmo en acción



¿Hay un ciclo luego de **R**?

El algoritmo en acción



!!!

Depth First Search (DFS)

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

Llegan hasta el final de una rama antes de empezar a explorar otra

¿Cuál es la complejidad de estos algoritmos?

DFS: Exploración 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