Recorrido en profundidad: los tipos de arista

Grafos dirigidos y no dirigidos, ciclos con amplitud y un recorrido con caminos cerrados

Carlos A Delgado S

Pontificia Universidad Javeriana Cali
Sesiones de refuerzo de algoritmia

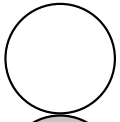
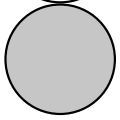
Contenido

1. Los tipos de arista en un grafo dirigido
2. El mismo recorrido en un grafo no dirigido
3. Ciclos con amplitud: los grados de entrada
4. Recorrer sin usar caminos cerrados

Contenido

1 Los tipos de arista en un grafo dirigido
- Concepto
- Ejemplo

Tres estados

	Estado	Su llamada `visitar(u)`
	blanco	no ha empezado
	gris	sigue abierta: u está en el camino actual
	negro	ya devolvió: todo lo que sale de u quedó visto

Lo que pasa en el recorrido

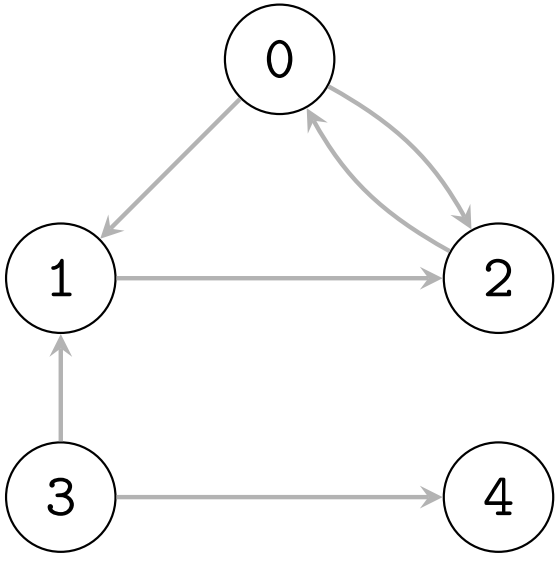
`visitar(u)` pinta u de gris, mira cada arista $u \to v$ y baja a v si está blanco. Cuando se le acaban las aristas, pinta u de negro.

Un reloj para dos tiempos

$u.d$ y $u.f$

Un solo contador, que arranca en 0, sube en uno cada vez que un nodo se pinta. $u.d$ es su valor cuando u se pinta de gris, y $u.f$ cuando se pinta de negro. Con V nodos salen los enteros de 1 a $2V$, y u está gris justo entre $u.d$ y $u.f$.

¿Y si v ya no está blanco?

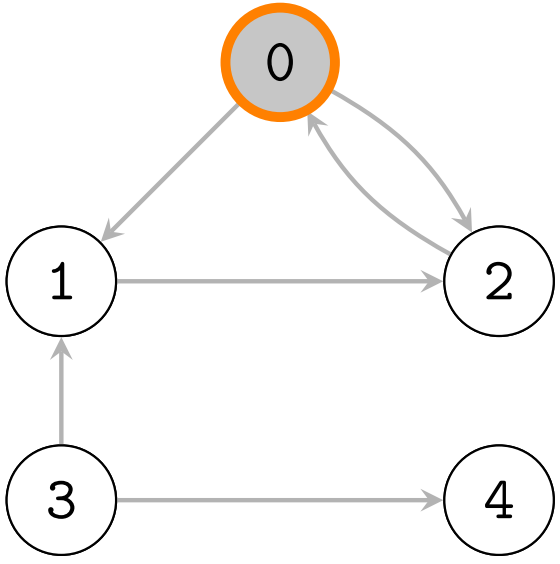


Nodo	Lista
0	[1, 2]
1	[2]
2	[0]
3	[1, 4]
4	[ ]

Pregunta

Al mirar $u \to v$ con v blanco, la profundidad baja. ¿Y si v está gris, o negro? Se recorre G_1 desde 0.

La traza sobre G_1

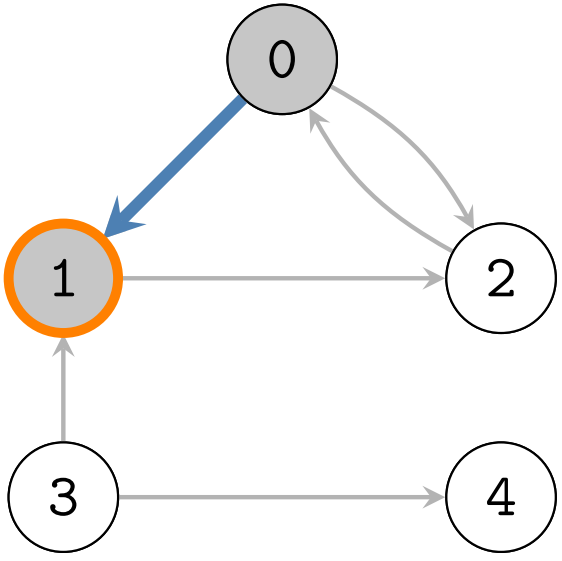


Arista	v está	Tipo

Nodo	0	1	2	3	4
Estado	gris	blanco	blanco	blanco	blanco
$u.d$	1				
$u.f$					

Arranca en 0: se pinta de gris y el reloj marca $0.d = 1$.

La traza sobre G_1

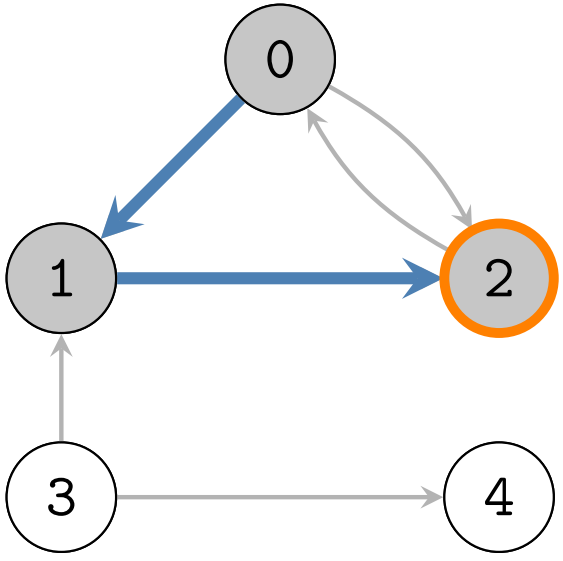


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**

Nodo	0	1	2	3	4
Estado	gris	gris	blanco	blanco	blanco
$u.d$	1	2			
$u.f$					

0 mira a 1, que está blanco: arista de **árbol**. Se baja a 1, gris, con $1.d = 2$.

La traza sobre G_1

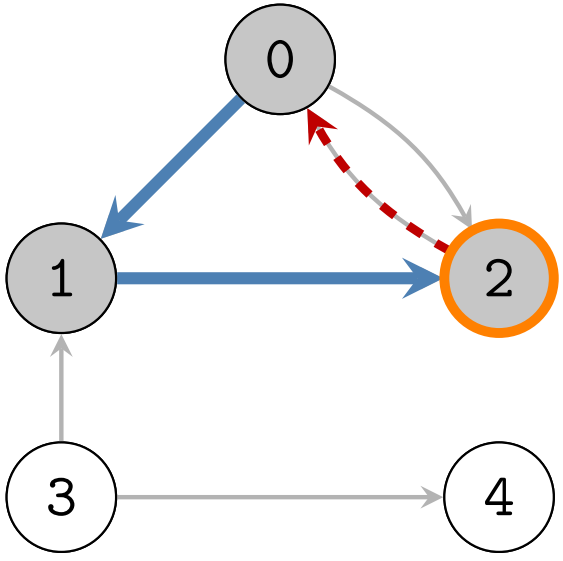


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**

Nodo	0	1	2	3	4
Estado	gris	gris	gris	blanco	blanco
$u.d$	1	2	3		
$u.f$					

1 mira a 2, blanco: otra de **árbol**. El nodo 2 queda gris con $2.d = 3$.

La traza sobre G_1

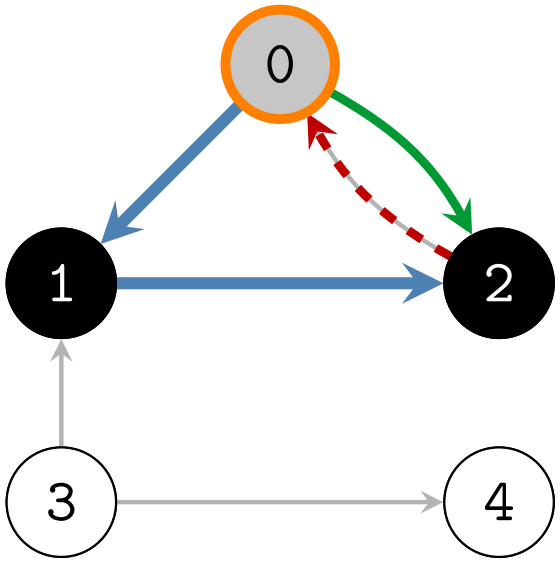


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**

Nodo	0	1	2	3	4
Estado	gris	gris	gris	blanco	blanco
$u.d$	1	2	3		
$u.f$					

2 mira a 0, que está gris: su llamada sigue abierta, así que 0 es ancestro de 2. Arista de **retroceso**.

La traza sobre G_1

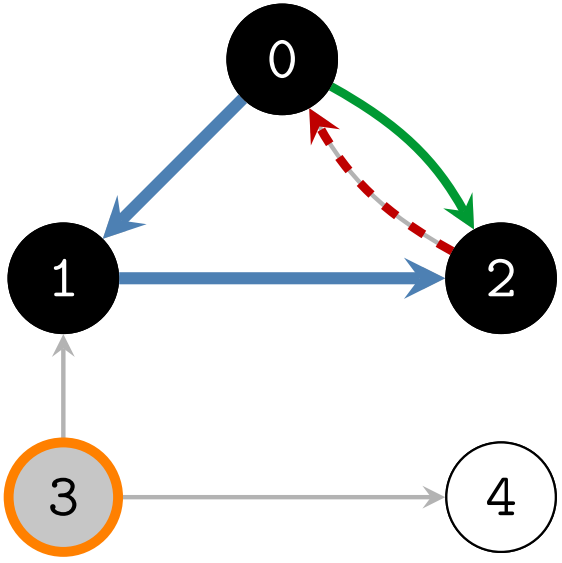


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**
$0 \to 2$	negro, $1 < 3$	**avance**

Nodo	0	1	2	3	4
Estado	gris	negro	negro	blanco	blanco
$u.d$	1	2	3		
$u.f$		5	4		

2 y 1 terminan ($2.f = 4$, $1.f = 5$). Vuelve 0 y mira a 2: negro, y $0.d = 1 < 3 = 2.d$. Arista de **avance**.

La traza sobre G_1

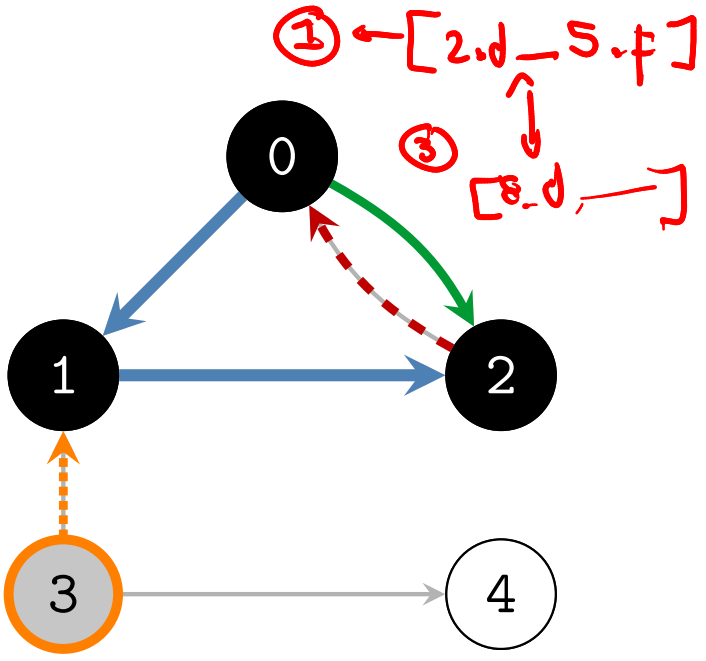


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**
$0 \to 2$	negro, $1 < 3$	**avance**

Nodo	0	1	2	3	4
Estado	negro	negro	negro	gris	blanco
$u.d$	1	2	3	7	
$u.f$	6	5	4		

0 termina con $0.f = 6$. El ciclo de afuera pasa por 1 y 2, negros, y arranca en 3, blanco: $3.d = 7$.

La traza sobre G_1

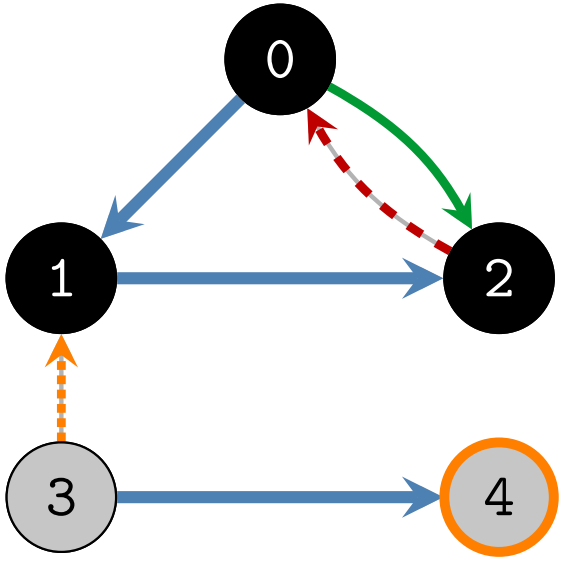


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**
$0 \to 2$	negro, $1 < 3$	**avance**
$3 \to 1$	negro, $7 > 2$	**cruzada**

Nodo	0	1	2	3	4
Estado	negro	negro	negro	gris	blanco
$u.d$	1	2	3	7	
$u.f$	6	5	4		

3 mira a 1: negro, y $3.d = 7 > 2 = 1.d$. 1 terminó antes de que 3 empezara: arista **cruzada**.

La traza sobre G_1

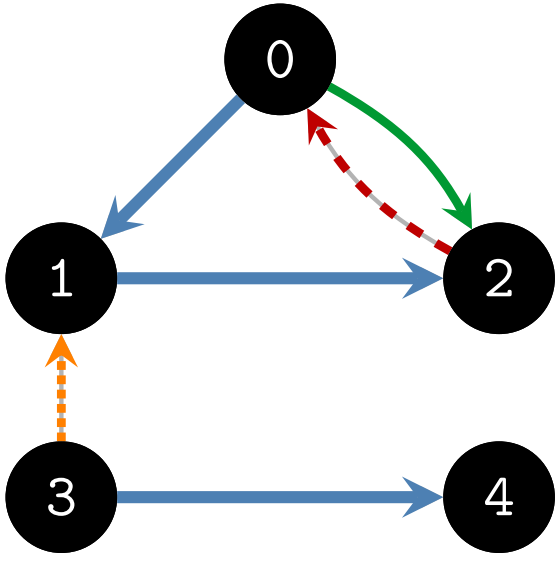


Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**
$0 \to 2$	negro, $1 < 3$	**avance**
$3 \to 1$	negro, $7 > 2$	**cruzada**
$3 \to 4$	blanco	**árbol**

Nodo	0	1	2	3	4
Estado	negro	negro	negro	gris	gris
$u.d$	1	2	3	7	8
$u.f$	6	5	4		

3 mira a 4, blanco: arista de **árbol**, y $4.d = 8$.

La traza sobre G_1



Arista	v está	Tipo
$0 \to 1$	blanco	**árbol**
$1 \to 2$	blanco	**árbol**
$2 \to 0$	gris	**retroceso**
$0 \to 2$	negro, $1 < 3$	**avance**
$3 \to 1$	negro, $7 > 2$	**cruzada**
$3 \to 4$	blanco	**árbol**

Nodo	0	1	2	3	4
Estado	negro	negro	negro	negro	negro
$u.d$	1	2	3	7	8
$u.f$	6	5	4	10	9

4 y 3 terminan ($4.f = 9$, $3.f = 10$). Todos quedan negros, y cada arista tiene su tipo.

La regla de los cuatro tipos

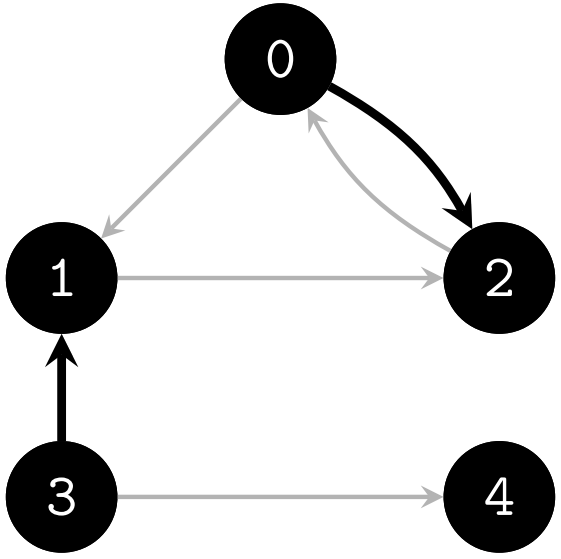
	Tipo	v está	Qué significa
———	**árbol**	blanco	v se descubre por esa arista
- - - - -	**retroceso**	gris	v es ancestro de u
———	**avance**	negro, $u.d < v.d$	v es descendiente de u, pero no su hijo
.....	**cruzada**	negro, $u.d > v.d$	ninguno es ancestro del otro

Dos preguntas en orden

Primero el estado de v. Solo si es negro se comparan los tiempos de descubrimiento.

https://vjudge.net/problem/UVA-11080

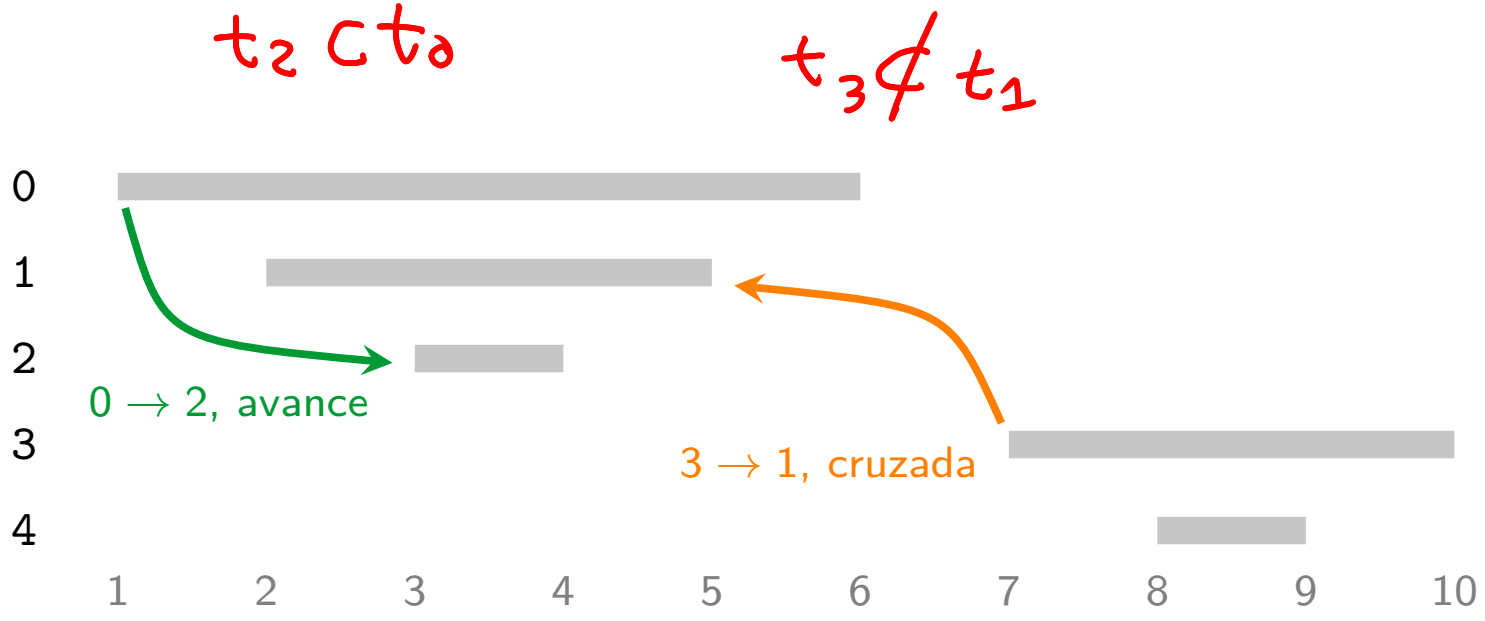
Dos aristas hacia un nodo negro



Pregunta

$0 \to 2$ y $3 \to 1$ llegan las dos a un nodo negro. ¿Qué tienen de distinto?

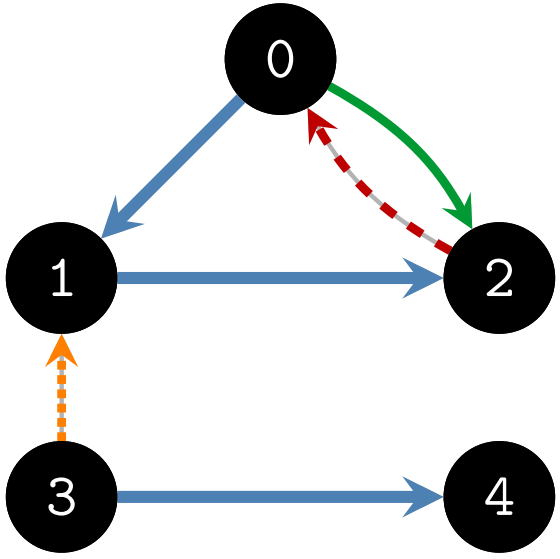
Los tiempos lo dicen



Respuesta

El intervalo de 2, de 3 a 4, cabe dentro del de 0, de 1 a 6: 2 es descendiente de 0. El de 1 acabó en 5, antes de que 3 empezara en 7.

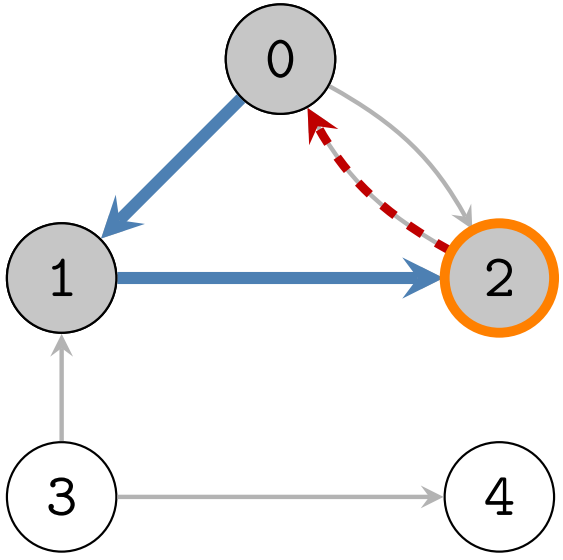
¿Cuál de los cuatro avisa un ciclo?



Pregunta

G_1 tiene un ciclo: 0, 1, 2. ¿Qué tipo de arista lo delata?

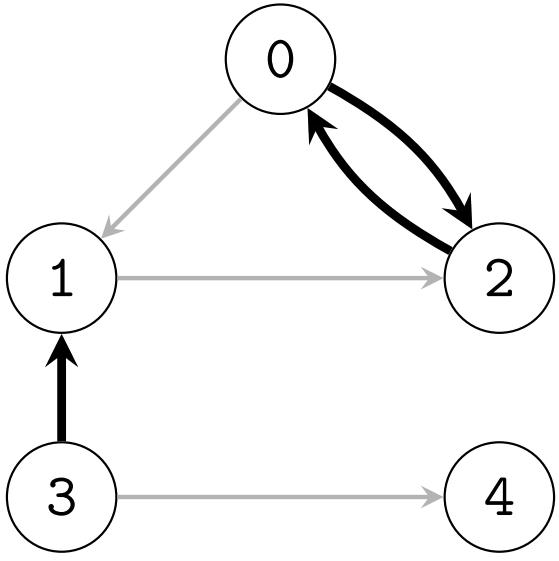
El de retroceso



Respuesta

Si v está gris al mirar $u \to v$, hay un camino de árbol de v hasta u, que es el camino actual, y la arista $u \to v$ lo cierra. Aquí: $0 \to 1 \to 2$ y luego $2 \to 0$. Un grafo dirigido tiene ciclo si y solo si el recorrido encuentra una arista de retroceso.

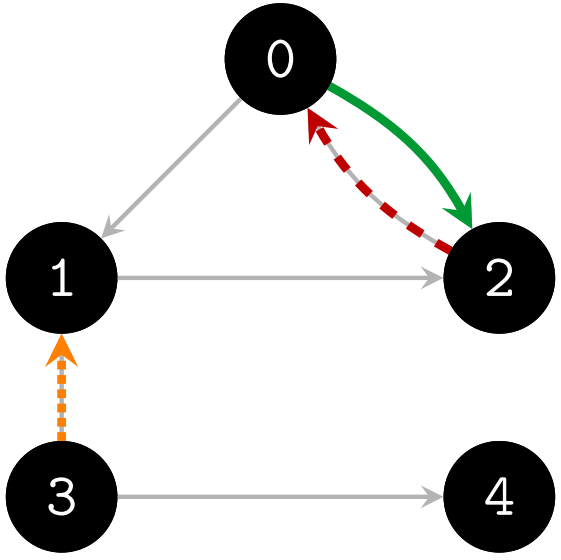
Con solo visitado



Pregunta

Si cada nodo solo se marca como visitado, y se declara ciclo cuando una arista llega a uno marcado, ¿cuántos ciclos encuentra el recorrido en G_1?

Tres avisos y un solo ciclo



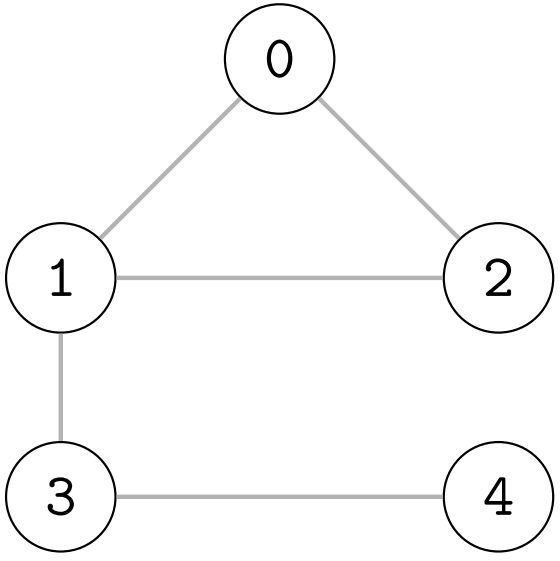
Respuesta

Tres aristas llegan a un visitado: $2 \to 0$, $0 \to 2$ y $3 \to 1$. Solo la primera cierra un ciclo. Visitado junta gris con negro, y por eso hacen falta los tres estados.

Contenido

2 El mismo recorrido en un grafo no dirigido
- Concepto
- Ejemplo

El grafo G_2

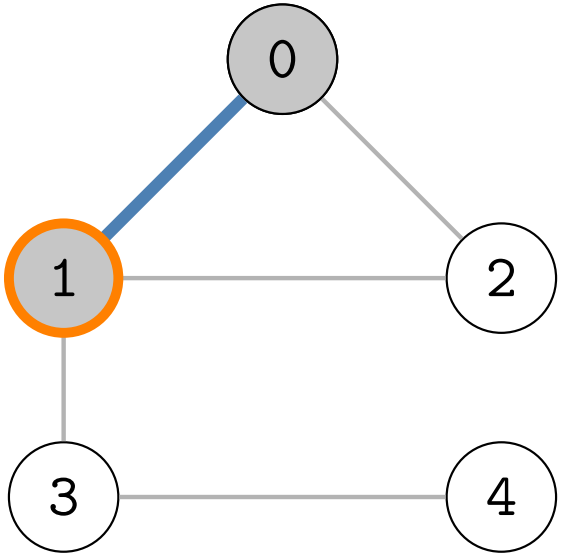


Nodo	Lista
0	[1, 2]
1	[0, 2, 3]
2	[0, 1]
3	[1, 4]
4	[3]

Cada arista, dos veces

Es el dibujo de G_1 sin flechas. La arista $1 - 3$ está en la lista de 1 y en la de 3, así que el recorrido la mira dos veces, una desde cada punta.

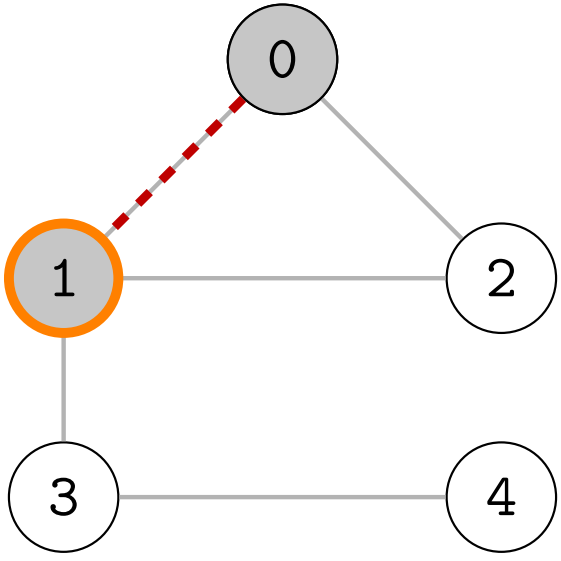
La regla del dirigido, sin cambios



Pregunta

Desde 0 se baja a 1. La lista de 1 empieza con 0, que está gris. Con la regla de antes, ¿qué tipo es $1 - 0$, y qué concluye el recorrido?

Un ciclo que no existe



Respuesta

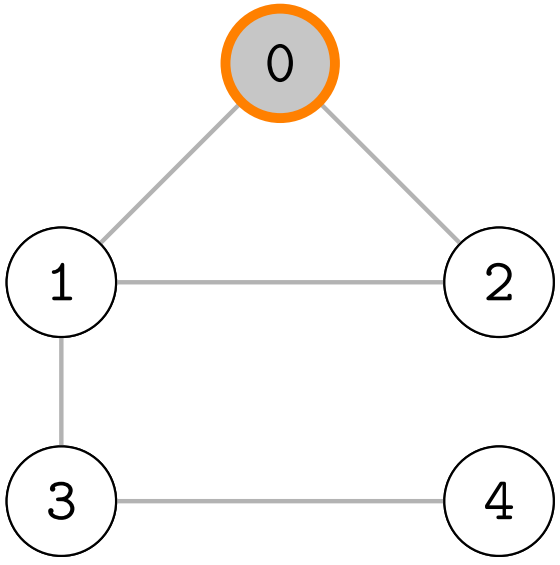
Saldría de retroceso, y el recorrido diría que hay ciclo. Pasa con toda arista de árbol: desde el hijo, el padre siempre está gris. Con esa regla, cualquier grafo no dirigido con una arista tendría ciclo.

El padre

`visitar(u, padre)`

Cada llamada recibe el nodo desde el que se bajó. Al mirar $u - v$ con v igual al padre, la arista se salta: es la de árbol, vista desde abajo. La raíz de cada arranque se llama con padre -1. Lo demás sigue igual: blanco es árbol y gris es retroceso.

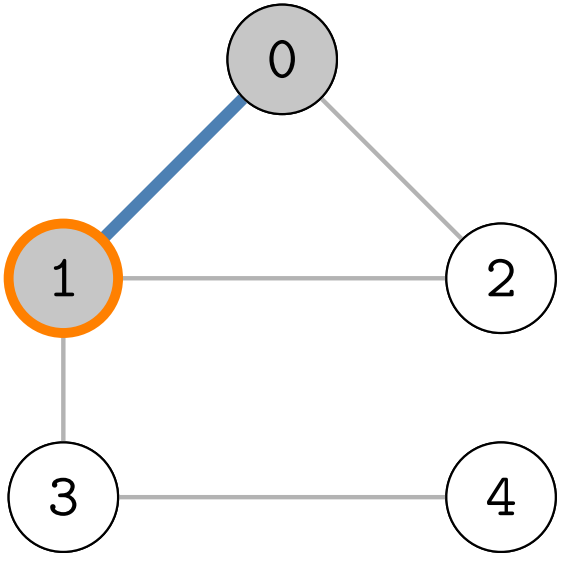
La traza sobre G_2



Arista	v está	Tipo

Arranca en 0, sin padre: se pinta de gris.

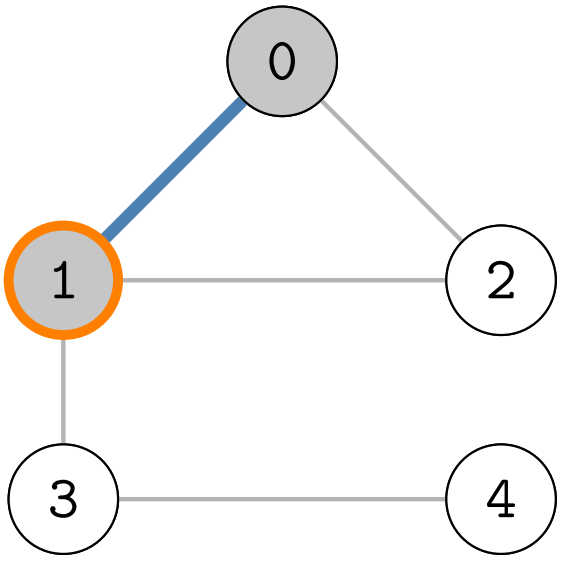
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**

0 mira a 1, blanco: **árbol**. Se baja a 1 con padre 0.

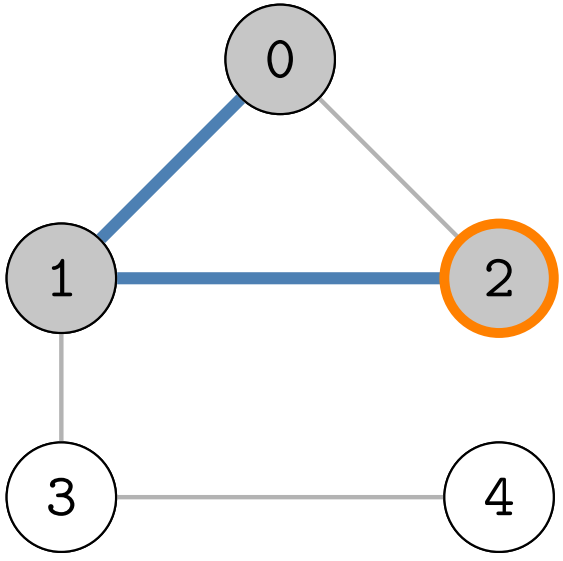
La traza sobre G_2



Arista	v está	Tipo
$0 - 1$	blanco	**árbol**
$1 - 0$	gris, padre	**padre**

1 mira a 0: es su padre, la misma arista por la que bajó. Se salta.

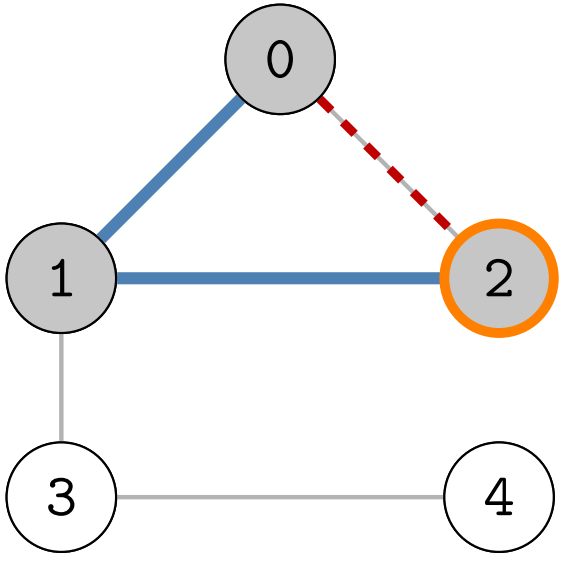
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**

1 mira a 2, blanco: **árbol**. Se baja a 2 con padre 1.

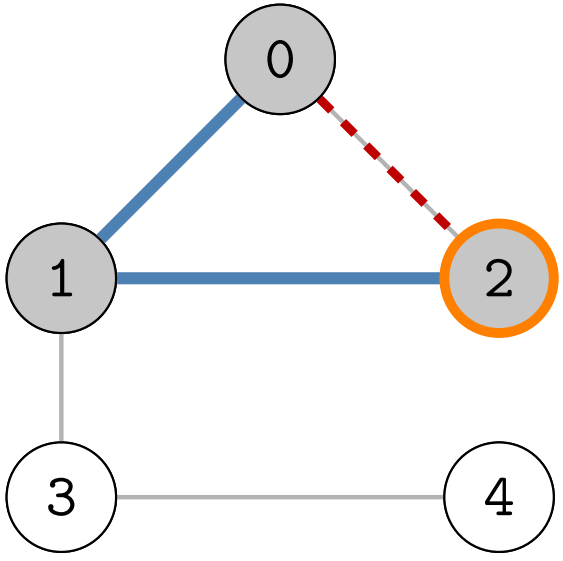
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**

2 mira a 0: gris y no es el padre. **retroceso**: el ciclo 0, 1, 2.

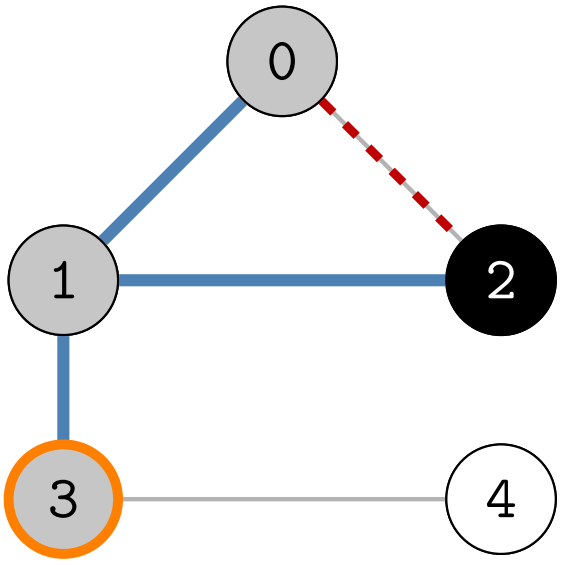
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**

2 mira a 1: es su padre. Se salta, y 2 termina.

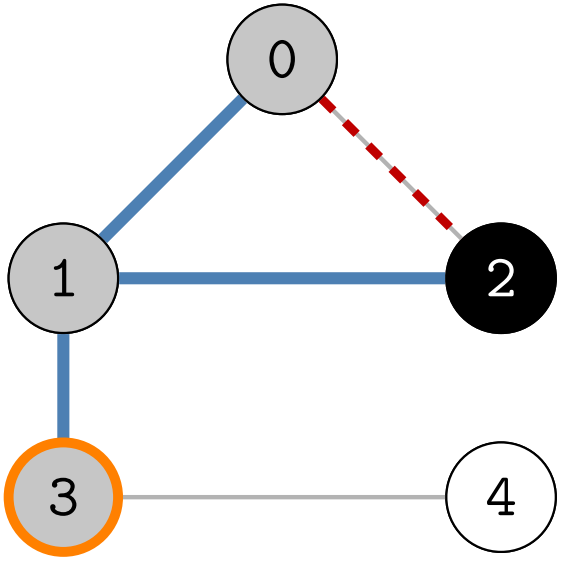
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**
$1-3$	blanco	**árbol**

1 mira a 3, blanco: **árbol**. Se baja a 3 con padre 1.

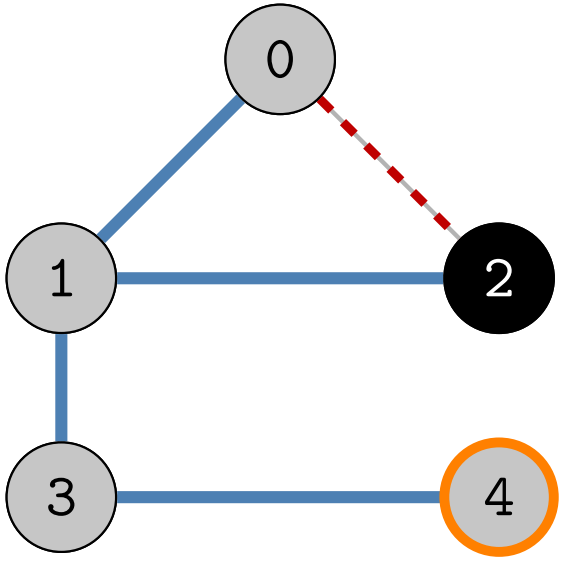
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**
$1-3$	blanco	**árbol**
$3-1$	gris, padre	**padre**

3 mira a 1: su padre. Se salta.

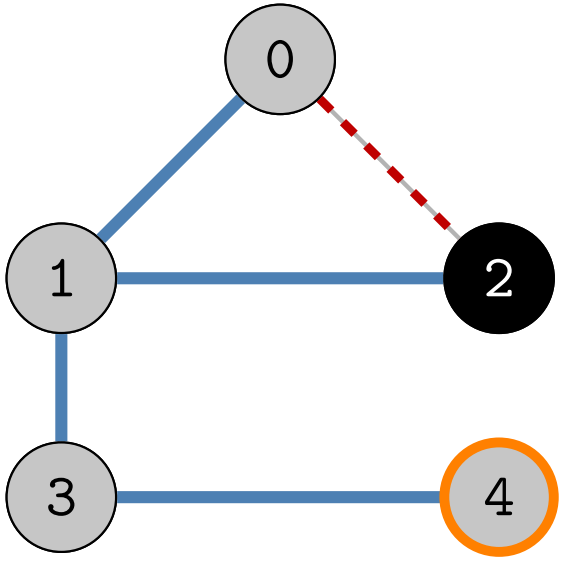
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**
$1-3$	blanco	**árbol**
$3-1$	gris, padre	**padre**
$3-4$	blanco	**árbol**

3 mira a 4, blanco: **árbol**. Se baja a 4 con padre 3.

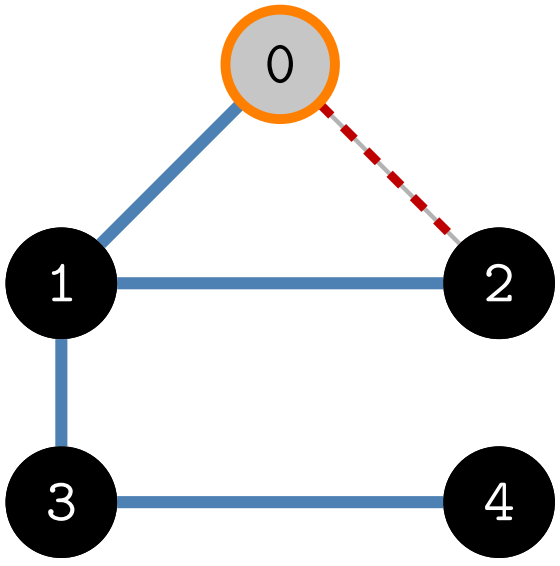
La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**
$1-3$	blanco	**árbol**
$3-1$	gris, padre	**padre**
$3-4$	blanco	**árbol**
$4-3$	gris, padre	**padre**

4 mira a 3: su padre. Se salta; terminan 4, 3 y 1.

La traza sobre G_2



Arista	v está	Tipo
$0-1$	blanco	**árbol**
$1-0$	gris, padre	**padre**
$1-2$	blanco	**árbol**
$2-0$	gris	**retroceso**
$2-1$	gris, padre	**padre**
$1-3$	blanco	**árbol**
$3-1$	gris, padre	**padre**
$3-4$	blanco	**árbol**
$4-3$	gris, padre	**padre**
$0-2$	negro	**repetida**

0 mira a 2: negro. Es la arista $2-0$, que ya salió como **retroceso** desde el otro lado: **repetida**. Termina 0.

¿Y el avance y la cruzada?

Pregunta

En G_2 no salió ninguna arista de avance ni cruzada, y $0 - 2$ solo repitió una de retroceso. ¿Puede salir una cruzada en algún grafo no dirigido?

No: la arista se mira desde las dos puntas

Respuesta

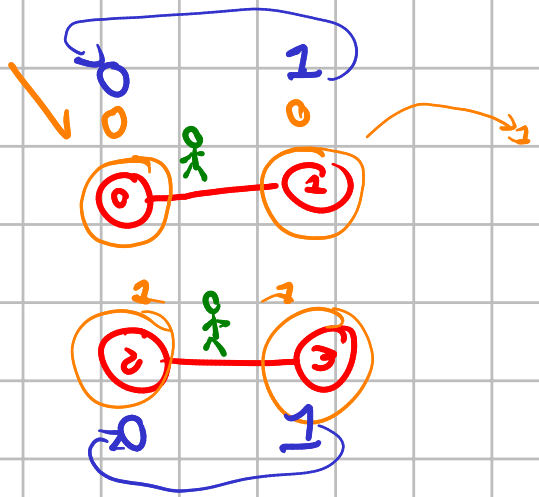
Suponga $u - v$ con v negro sin ser descendiente de u. Entonces v terminó antes de que u empezara, con u en blanco. Pero u está en la lista de v: al mirarla, v habría bajado a u. Así que v negro solo puede ser descendiente, y esa arista ya salió de retroceso desde v.

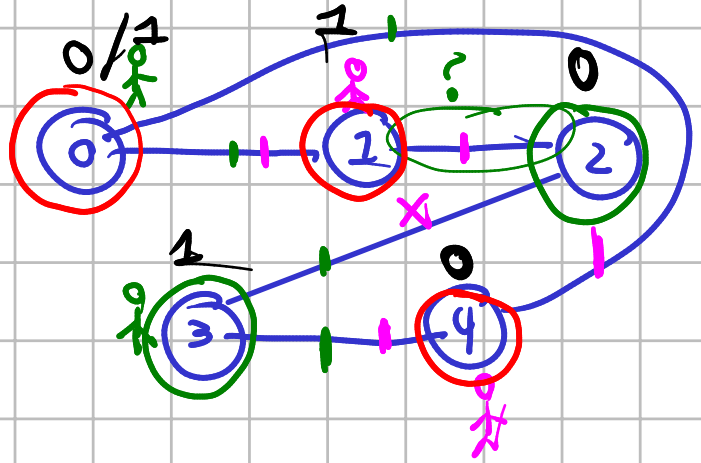
Dirigido y no dirigido

	Dirigido	No dirigido
Tipos que aparecen	árbol, retroceso, avance y cruzada	árbol y retroceso
Cada arista se mira	una vez	dos veces
Hay ciclo si	v está gris	v está gris y no es el padre
Lo que lleva la llamada	u	u y su padre

2

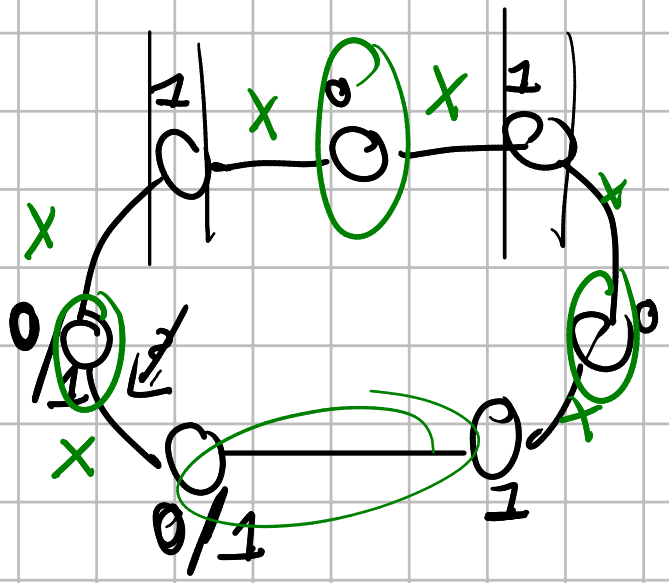
4	2
0	1
2	3
5	5
0	1
1	2
2	3
0	4
3	4

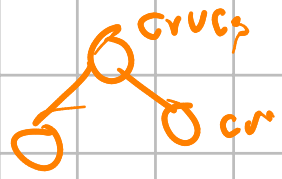




2

-1

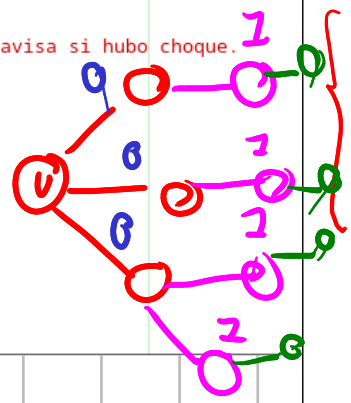




[-1, -1, -1, -1]

```python
import sys

def pintar(adj, color, u, c, cuenta):
    # Pinta u con c, cuenta cuántos quedan de cada color y avisa si hubo choque.
    color[u] = c
    cuenta[c] = cuenta[c] + 1
    sin_choque = True
    for v in adj[u]:
        if color[v] == -1:
            if not pintar(adj, color, v, 1 - c, cuenta):
                sin_choque = False
        elif color[v] == c:
            sin_choque = False
    return sin_choque
```




```python
def guardias(adj):
    # Suma, pedazo por pedazo, el color menos usado; -1 si un pedazo choca.
    color = [-1] * len(adj)
    total = 0
    posible = True
    for u in range(len(adj)):
        if color[u] == -1:
            cuenta = [0, 0]
            if not pintar(adj, color, u, 0, cuenta):
                posible = False
            elif cuenta[1] == 0:
                total = total + 1
            else:
                total = total + min(cuenta[0], cuenta[1])
    respuesta = -1
    if posible:
        respuesta = total
    return respuesta
```

VERTICES

```python
def main():
    sys.setrecursionlimit(10000)
    datos = sys.stdin.read().split()
    pos = 0
    casos = int(datos[pos])
    pos = pos + 1
    salida = []
    caso = 0
    while caso < casos:
        v = int(datos[pos])
        e = int(datos[pos + 1])
        pos = pos + 2
        adj = [[] for _ in range(v)]
        i = 0
        while i < e:
            f = int(datos[pos])
            t = int(datos[pos + 1])
            pos = pos + 2
            adj[f].append(t)
            adj[t].append(f)
            i = i + 1
        salida.append(str(guardias(adj)))
        caso = caso + 1
    print('\n'.join(salida))

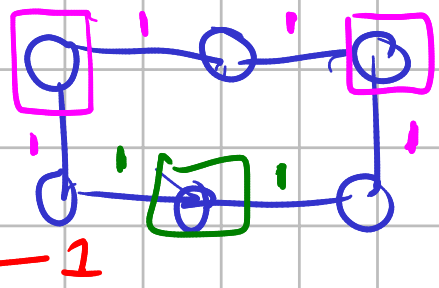

main()
```

Adyacencia

Caso.

#V #A

#V #A



0 — 1

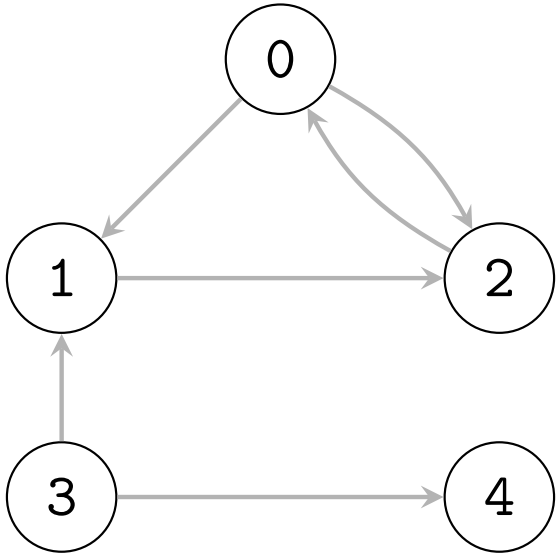
0: [1]

1: [0]

Contenido

3 Ciclos con amplitud: los grados de entrada
- Concepto
- Ejemplo

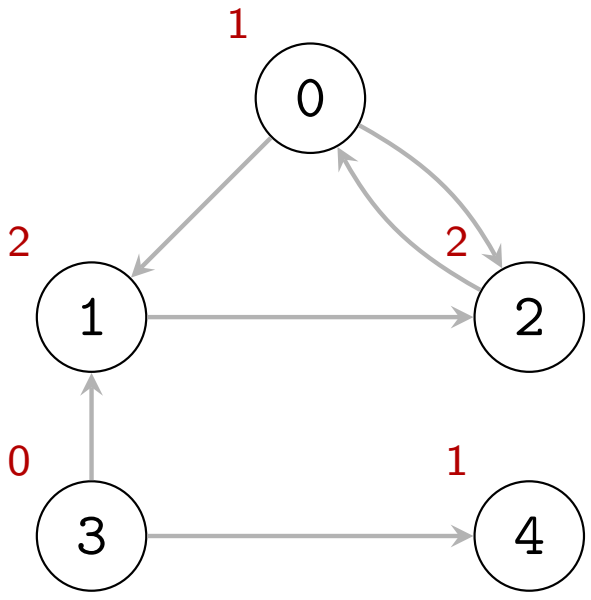
¿Quién puede ir primero?



Pregunta

Lea cada arista $u \to v$ como "u va antes que v". En G_1, ¿qué nodo puede ir primero?

El que no tiene flechas que le lleguen



Respuesta: el 3

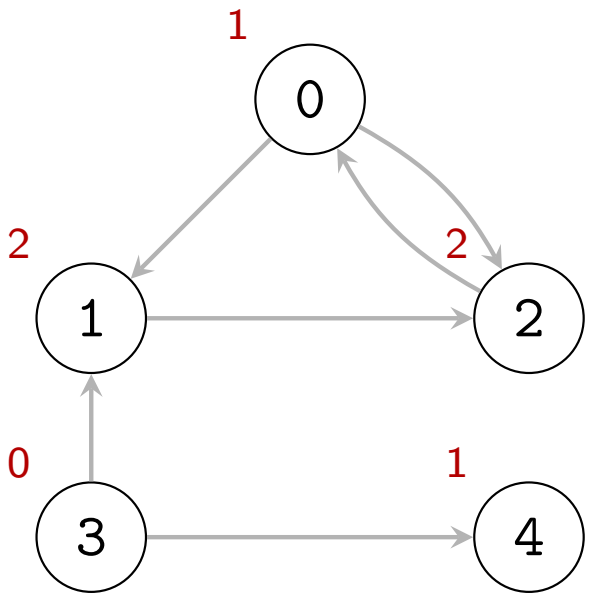
El grado de entrada de v es cuántas aristas llegan a v. Solo 3 tiene grado cero: nadie tiene que ir antes que él.

El algoritmo de Kahn

Una cola con los de grado cero

Se cuentan los grados de entrada y entran a la cola los de grado cero. Al sacar u, cada arista $u \to v$ baja en uno el grado de v, y si llega a cero, v entra. Si al vaciarse la cola no salieron todos, hay ciclo: los que quedan son el ciclo y lo que depende de él. Cuesta $\Theta(V + E)$.

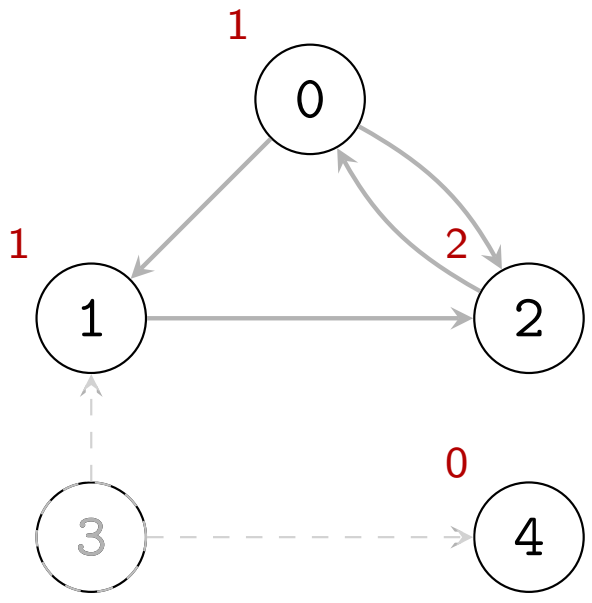
Kahn sobre G_1



Paso	Grado de entrada 0	1	2	3	4	Cola
inicio	1	2	2	0	1	[3]

Los grados son 1, 2, 2, 0 y 1. Solo 3 entra a la cola.

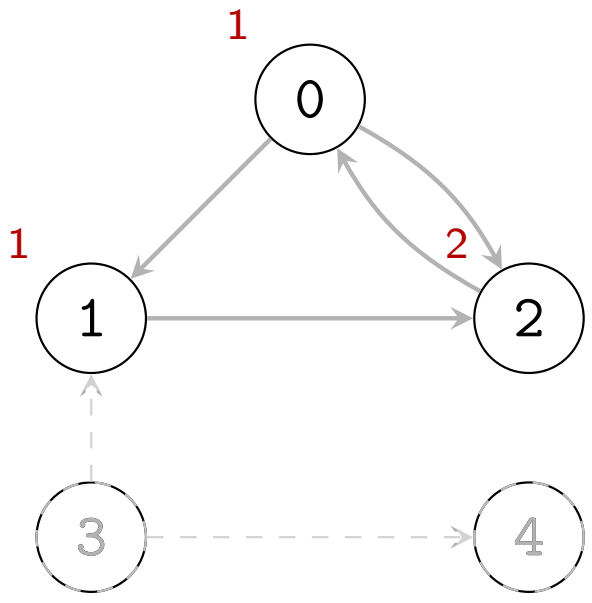
Kahn sobre G_1



Paso	Grado de entrada 0	1	2	3	4	Cola
inicio	1	2	2	0	1	[3]
sale 3	1	1	2		0	[4]

Sale 3. El grado de 1 baja a 1 y el de 4 a 0: entra 4.

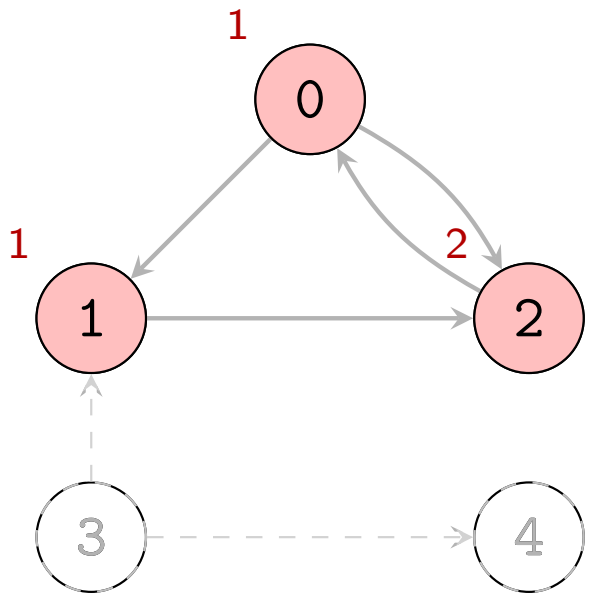
Kahn sobre G_1



Paso	Grado de entrada 0	1	2	3	4	Cola
inicio	1	2	2	0	1	[3]
sale 3	1	1	2		0	[4]
sale 4	1	1	2			[ ]

Sale 4, que no tiene aristas. La cola queda vacía.

Kahn sobre G_1



Paso	Grado de entrada 0	1	2	3	4	Cola
inicio	1	2	2	0	1	[3]
sale 3	1	1	2		0	[4]
sale 4	1	1	2			[ ]

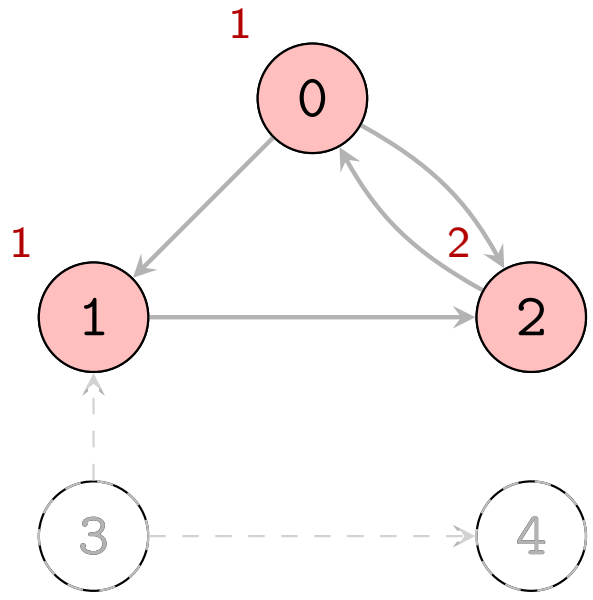
Salieron 2 de 5: hay ciclo. Quedan 0, 1 y 2, y ninguno tiene grado cero.

¿Por qué se atascan?

Pregunta

En el ciclo $0 \to 1 \to 2 \to 0$, ¿por qué ninguno de los tres llega nunca a grado cero?

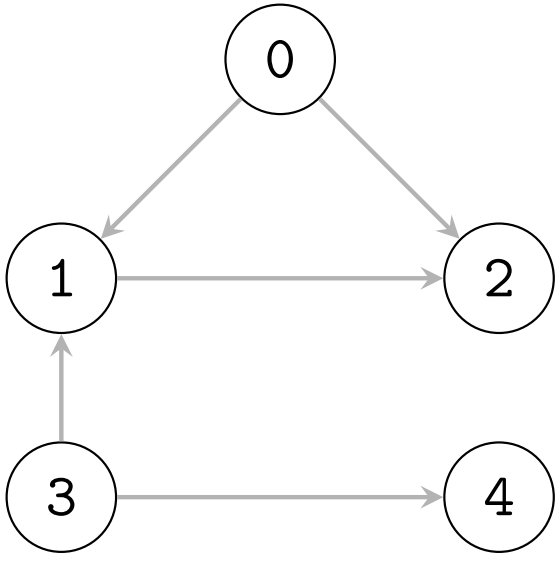
Cada uno espera al anterior



Respuesta

Un grado baja solo cuando sale el nodo de donde viene la arista. 0 espera a 2, 2 espera a 1 y 1 espera a 0: el primero que saliera tendría que haber salido después de otro del ciclo.

Sin la arista $2 \to 0$



	Grado de entrada					
Paso	0	1	2	3	4	Cola
inicio	0	2	2	0	1	[0, 3]
sale 0		1	1	0	1	[3]
sale 3		0	1		0	[1, 4]
sale 1			0		0	[4, 2]
sale 4			0			[2]
sale 2						[ ]

Salen los cinco, en el orden 0, 3, 1, 4, 2: no hay ciclo, y ese orden respeta todas las flechas.

Dos formas de encontrar un ciclo dirigido

	Profundidad	Amplitud con grados
Lo que guarda	estado de cada nodo	grado de cada nodo y una cola
Avisa el ciclo con	una arista de retroceso	los que no salen
Sin ciclo deja	tiempos $u.f$	un orden de los nodos
Cuesta	$\Theta(V + E)$	$\Theta(V + E)$

Sample Input

5 4

1 2

2 3

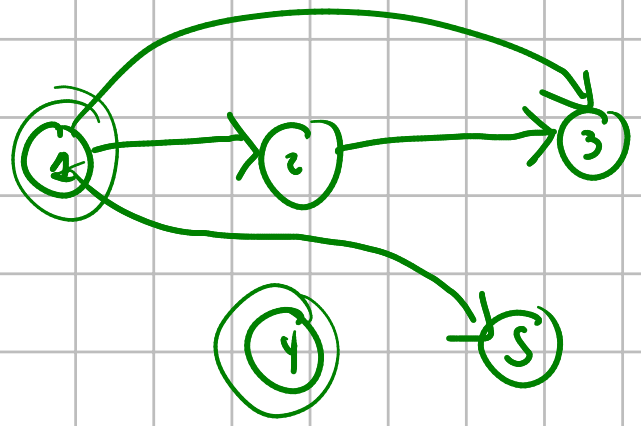
1 3

1 5

0 0

Sample Output

1 4 2 5 3



1 4 2 5 3

1) Grado

2) Valor nodo

Contenido

4 Recorrer sin usar caminos cerrados
- El enunciado
- Qué entra y qué sale
- Por tanteo
- La idea

Caminos cerrados

El problema

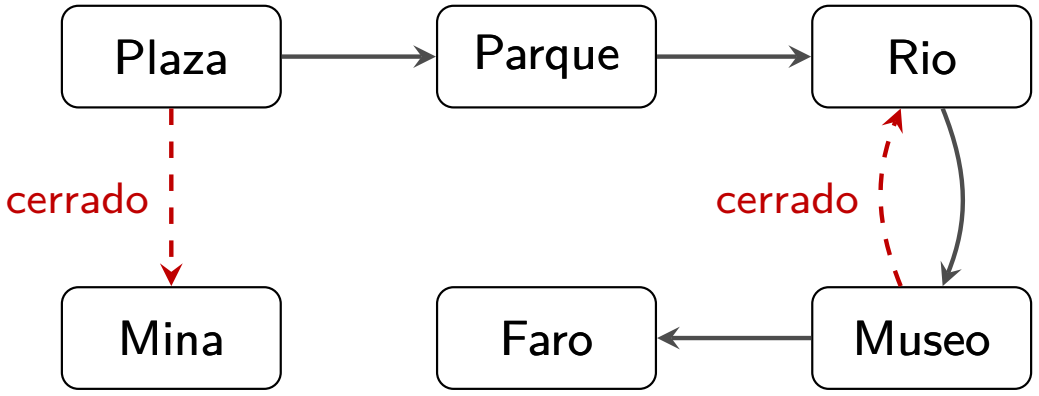
Una ciudad tiene lugares con nombre y caminos de un solo sentido entre ellos. Algunos caminos están cerrados. Desde un lugar de partida, se piden todos los lugares a los que se puede llegar sin usar un camino cerrado, cada uno una sola vez.

Qué entra y qué sale

`alcanzables(lugares, caminos, cerrados, inicio)`

- `lugares`: la lista de nombres, cadenas.
- `caminos`: pares (origen, destino), cada uno de un sentido.
- `cerrados`: pares (origen, destino) que no se pueden usar.
- `inicio`: el nombre del lugar de partida.
- Sale la lista de lugares alcanzables, `inicio` incluido, sin repetir.

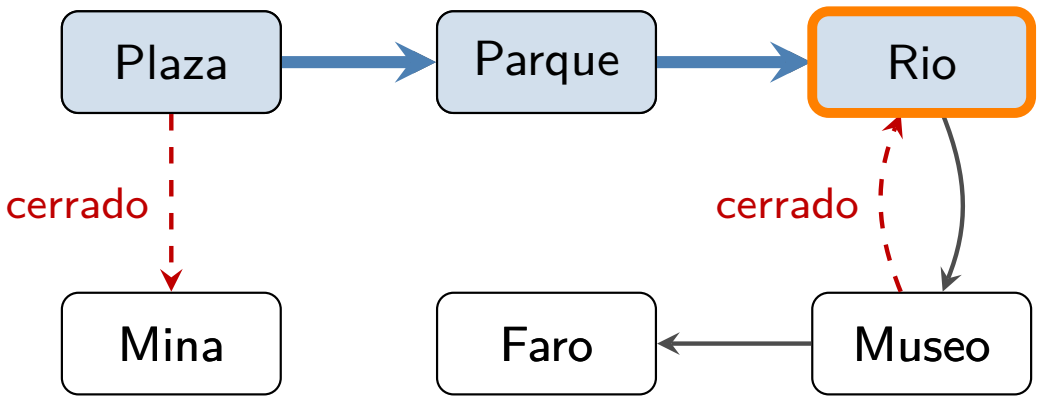
Un ejemplo chico



Desde Plaza

Seis lugares y seis caminos, dos de ellos cerrados: Plaza a Mina y Museo a Rio. Sale [Plaza, Parque, Rio, Museo, Faro]. A Mina solo se llega por el camino cerrado.

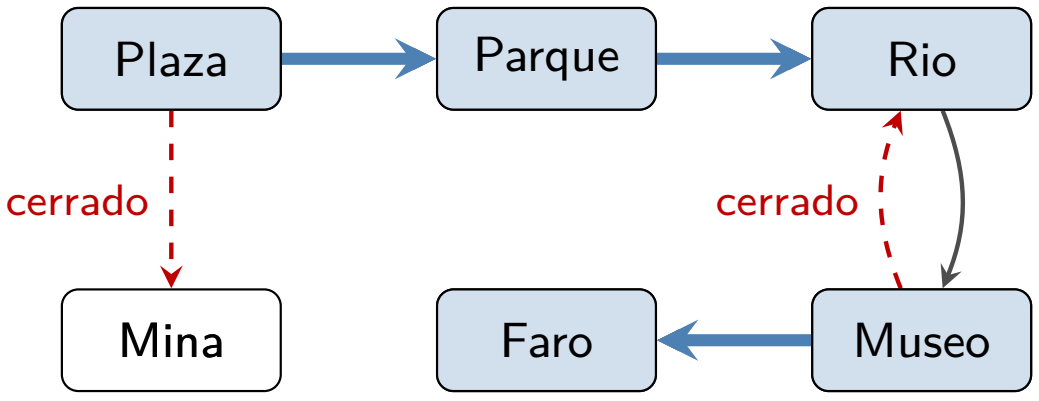
Por tanteo



Pregunta

Desde Plaza se llega a Parque y a Rio. El camino de Museo a Rio está cerrado. ¿Se puede ir de Rio a Museo?

Cerrar uno no cierra el otro



Respuesta: sí

Rio a Museo es otro camino, y está abierto. Quien cierra los dos sentidos responde [Plaza, Parque, Rio] y pierde a Museo y a Faro.

La forma directa

Pregunta

Antes de tomar un camino se busca en la lista de cerrados. Con E caminos y C cerrados, ¿cuánto cuesta el recorrido? ¿Y con $E = C = 10^5$?

La lista no alcanza

Respuesta

Cada camino se mira una vez, y cada mirada recorre los C cerrados: $\Theta(V + E \cdot C)$. Con $E = C = 10^5$ son 10^{10} comparaciones.

La idea: un conjunto de pares

Preguntar sin recorrer la lista

Los cerrados se guardan como pares (origen, destino) en un conjunto: `set` en Python, `std::set` en C++. La profundidad toma $u \to v$ solo si (u, v) no está en el conjunto y v no se ha visto. El par va en orden, así que (Museo, Rio) no cierra (Rio, Museo).

La idea: los nombres

Un diccionario de listas

La lista de adyacencia se indexa por el nombre: `adj["Plaza"]` es la lista de destinos de Plaza. Otra opción es pasar cada nombre a un índice con un diccionario y trabajar con vectores, como en G_1.

Lo que cuesta

De qué depende

Armar la lista cuesta $V + E$ y el conjunto, C. Cada lugar se visita una vez y cada camino se mira una vez. En Python, con diccionario y conjunto, el total es $\Theta(V + E + C)$ en promedio. En C++, `map` y `set` suman un factor logarítmico a cada consulta.

A resolver en el computador

El resultado

La función `alcanzables`, en Python o en C++, que con el ejemplo devuelve [Plaza, Parque, Rio, Museo, Faro]. Pruébela también desde Faro, que no tiene salidas, y con un cerrado que no está entre los caminos.

Referencias

- T. Cormen, C. Leiserson, R. Rivest y C. Stein. *Introduction to Algorithms*, tercera edición, 2009. Sección 22.3: recorrido en profundidad, tiempos y clasificación de aristas. Sección 22.4: orden topológico.