Árboles

Gregorio Hernández Peñalver

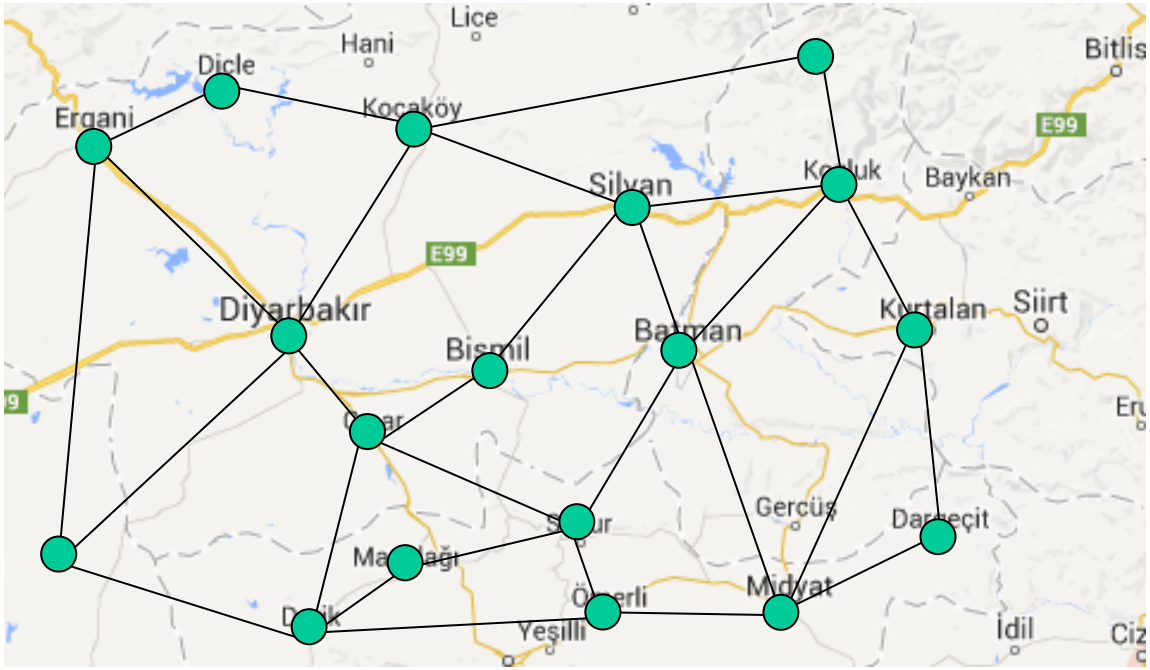
UPM

Matemática Discreta II

(MI)

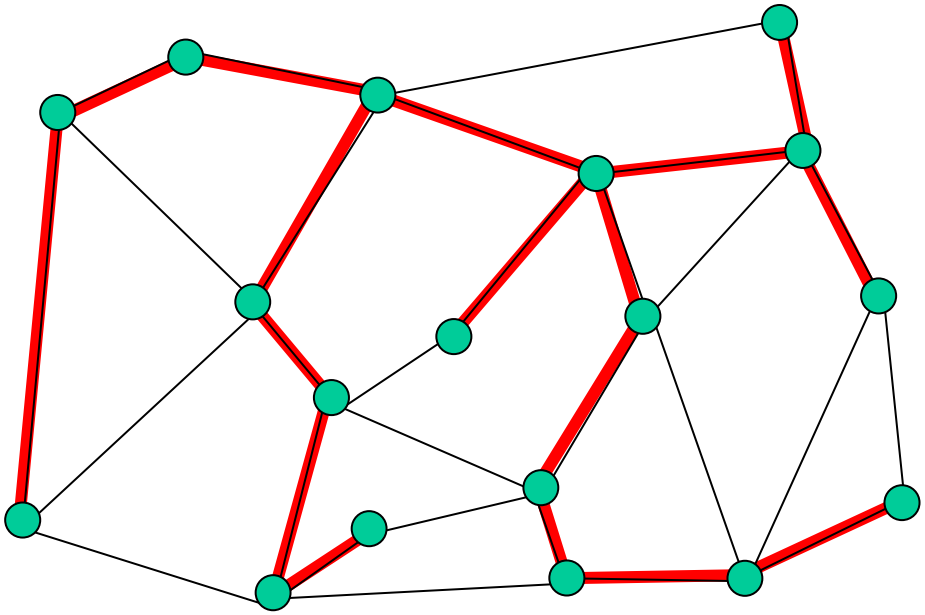
Árboles

Un terremoto destruye las carreteras de una comarca. ¿Cuáles se deben reparar para conseguir rápidamente que todos los pueblos sigan conectados?



Árboles

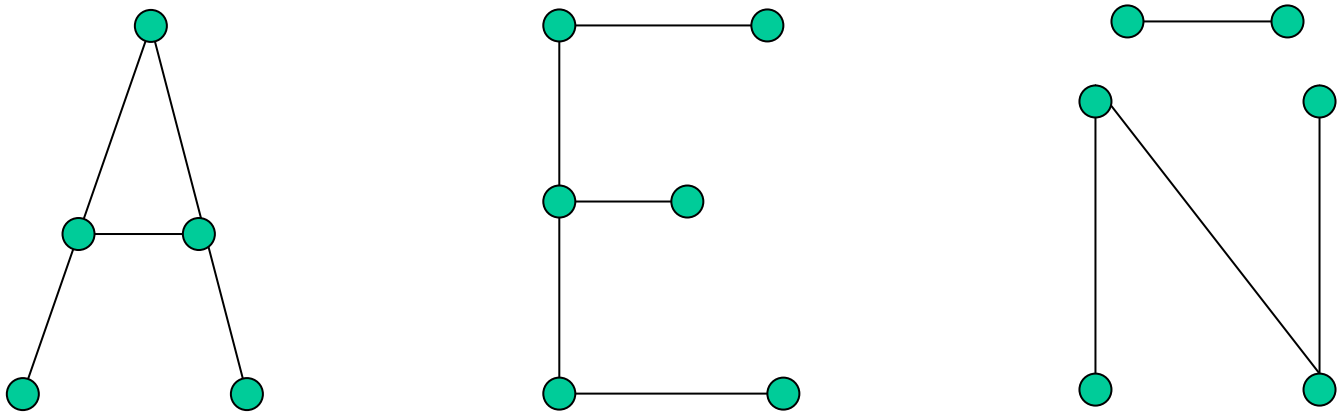
Un terremoto destruye las carreteras de una comarca. ¿Cuáles se deben reparar para conseguir rápidamente que todos los pueblos sigan conectados?



Grafo conexo
Sin ciclos

Árboles

G es un árbol si es un grafo conexo y acíclico.



Propiedades

- Entre cada par de vértices existe un camino único.
- Toda arista es puente.
- Un árbol de n vértices tiene n – 1 aristas
- Todo árbol no trivial tiene al menos dos hojas

Caracterizaciones

Un grafo simple es un árbol

(2) $\Leftrightarrow$ Entre cada par de vértices existe un camino único.
(3) $\Leftrightarrow$ G es conexo y toda arista es puente.
(4) $\Leftrightarrow$ G es acíclico maximal
(5) $\Leftrightarrow$ G es acíclico y $q = n - 1$.
(6) $\Leftrightarrow$ G es conexo y $q = n - 1$.

Probaremos que (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (4) $\Rightarrow$ (5) $\Rightarrow$ (6) $\Rightarrow$ (1)

Caracterizaciones de árbol

(2) $\Rightarrow$ (3) G es un grafo conexo porque existe un camino entre cada par de vértices de G. Si una arista **e**=**uv** no fuera puente, entonces G-e sería conexo y habría un camino P de **u** a **v** sin utilizar la arista **e**. Así habría dos caminos de **u** a **v** en G**,** el camino P y la arista **uv.**

(3) $\Rightarrow$ (4) Si G tiene un ciclo C y e es una arista de C entonces G–e sigue siendo conexo, luego la arista e no sería puente. Por tanto G es acíclico.
Sean u, v dos vértices cualesquiera no adyacentes de G. La adición de la arista uv crea el ciclo formado por la arista uv y el camino P (en G) que conecta los vértices u y v (por ser G conexo)

(4) $\Rightarrow$ (5) El grafo es acíclico por hipótesis. Si tuviera dos (o más) componentes conexas G_1 y G_2 la adición de una arista conectando un vértice de G_1 con un vértice de G_2 no crearía ningún ciclo.
Por tanto G es un bosque con **una** componente conexa que tiene n–1 aristas

(5) $\Rightarrow$ (6) Un grafo acíclico con k componentes tiene n – k aristas. Por tanto, si tiene n – 1 aristas será conexo

Caracterizaciones de árbol

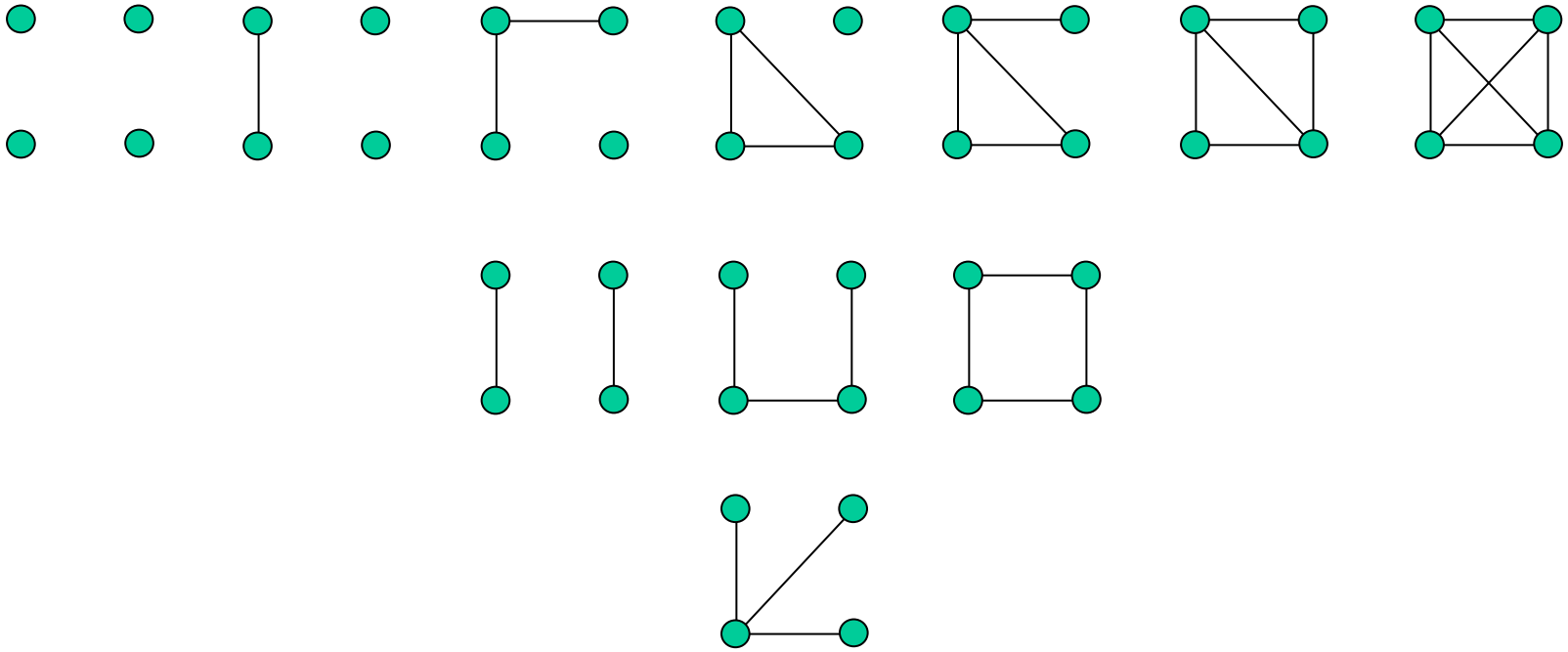
(6) $\Rightarrow$ (1) Falta probar que G no tiene ciclos. Renombrémosle como G_0, es decir, $G = G_0$. Supongamos que sí tiene un ciclo C y elijamos una arista $e \in C$.
El grafo $G_1 = G_0 - e$ sigue siendo conexo, tiene n vértices y n–2 aristas.
Si G_1 sigue teniendo algún ciclo se repite el proceso borrando una arista e_1 del ciclo. Así se obtiene el grafo $G_2 = G_1 - e_1$, conexo y con n–3 aristas. El proceso continúa hasta alcanzar un grafo G_p acíclico y conexo, es decir, árbol. G_p tiene, por la construcción efectuada, n vértices y n–(p+1) aristas. Pero un árbol de n vértices tiene n–1 aristas. Por tanto, p=0, es decir, el grafo inicial $G = G_0$ era acíclico.

Enumeración de grafos

¿Cuántos grafos simples hay con n vértices?

No etiquetados

n=4



¿Cuántos grafos simples hay con n vértices **etiquetados**? $2^{\binom{n}{2}}$

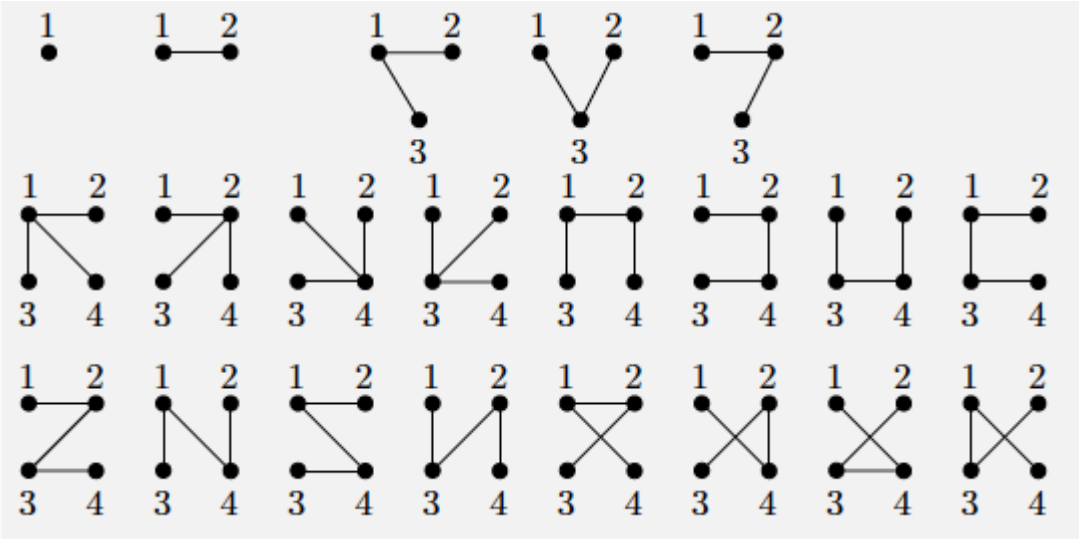
Enumeración de árboles

Recuento de isómeros químicos (Cayley)

Contemos árboles etiquetados:

$T_1 = 1$, $T_2 = 1$, $T_3 = 3$, $T_4 = 16$

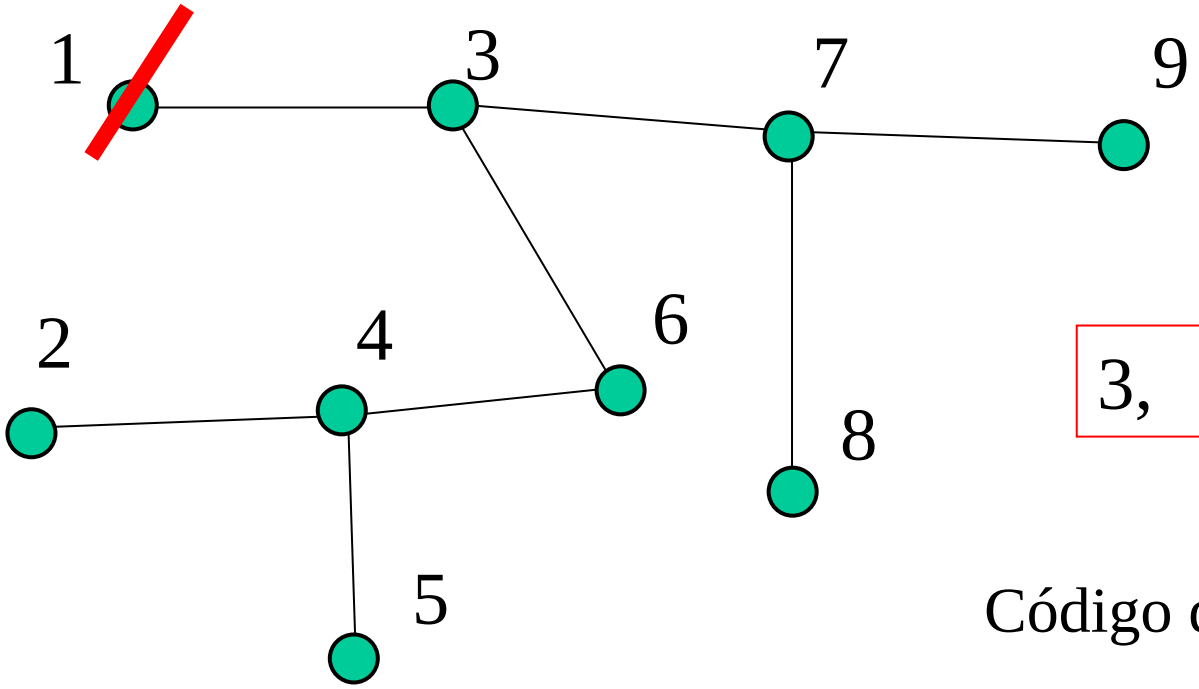
¿Y no etiquetados?



Fórmula de Cayley

El nº de árboles etiquetados de n vértices es n^{n-2}

Código de Prüfer de un árbol etiquetado

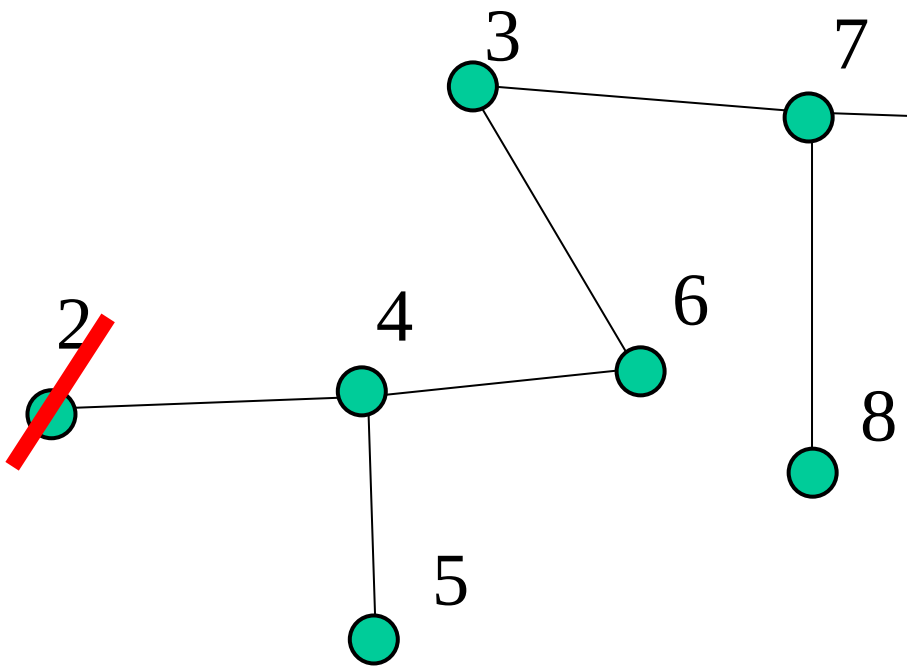


3,

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado

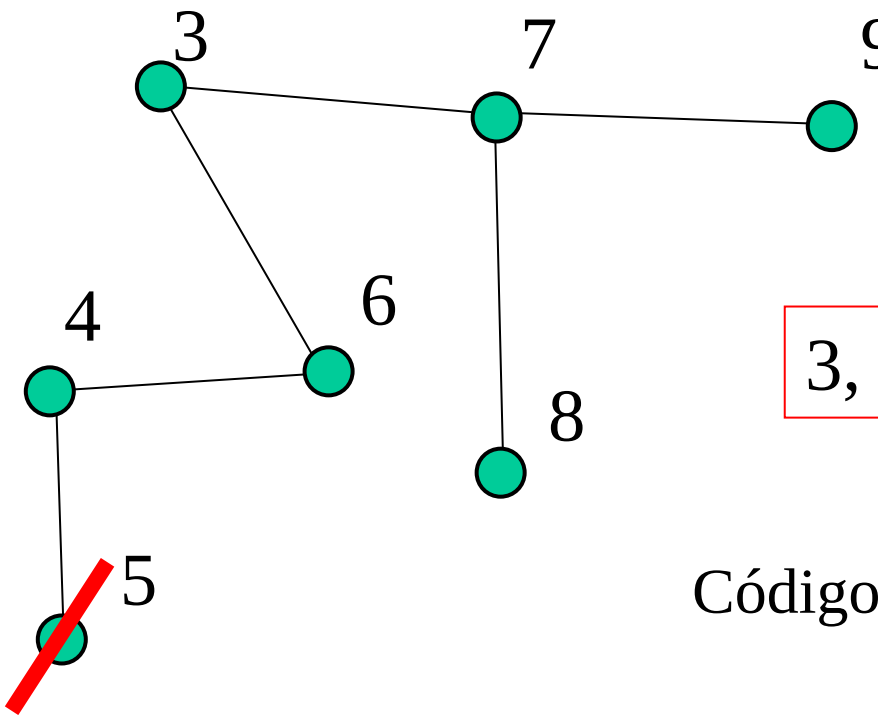


3, 4,

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado

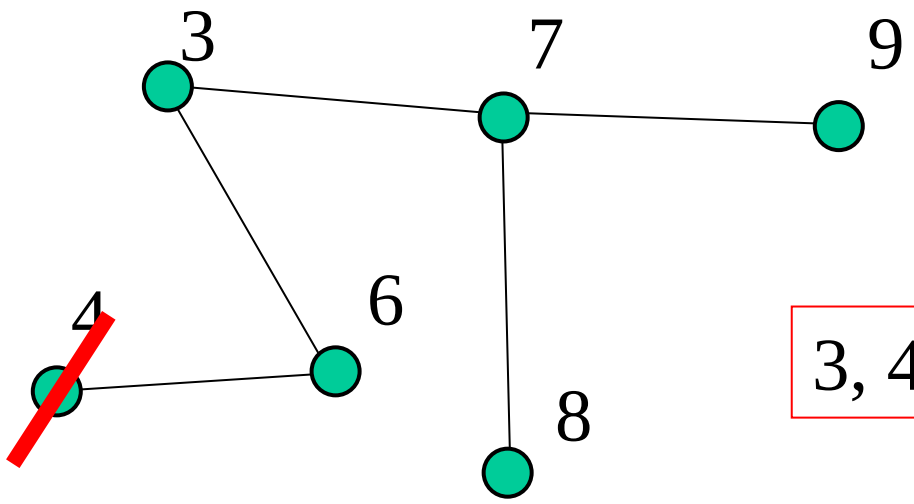


3, 4, 4,

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado

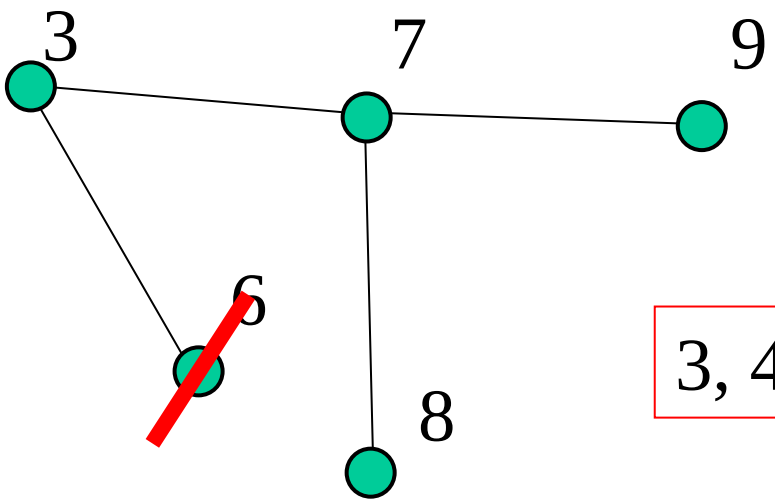


3, 4, 4, 6,

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado

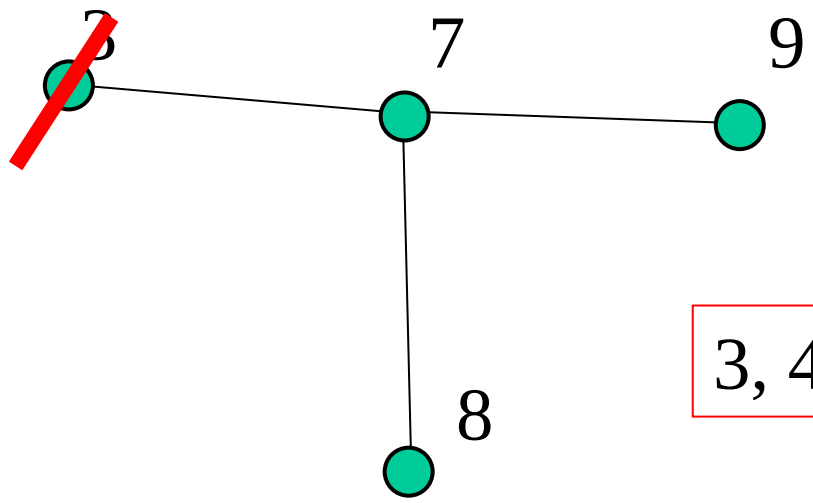


3, 4, 4, 6, 3,

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado

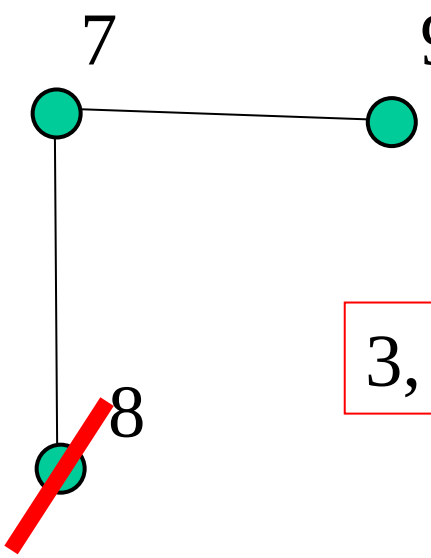


3, 4, 4, 6, 3, 7

Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Código de Prüfer de un árbol etiquetado



3, 4, 4, 6, 3, 7, 7

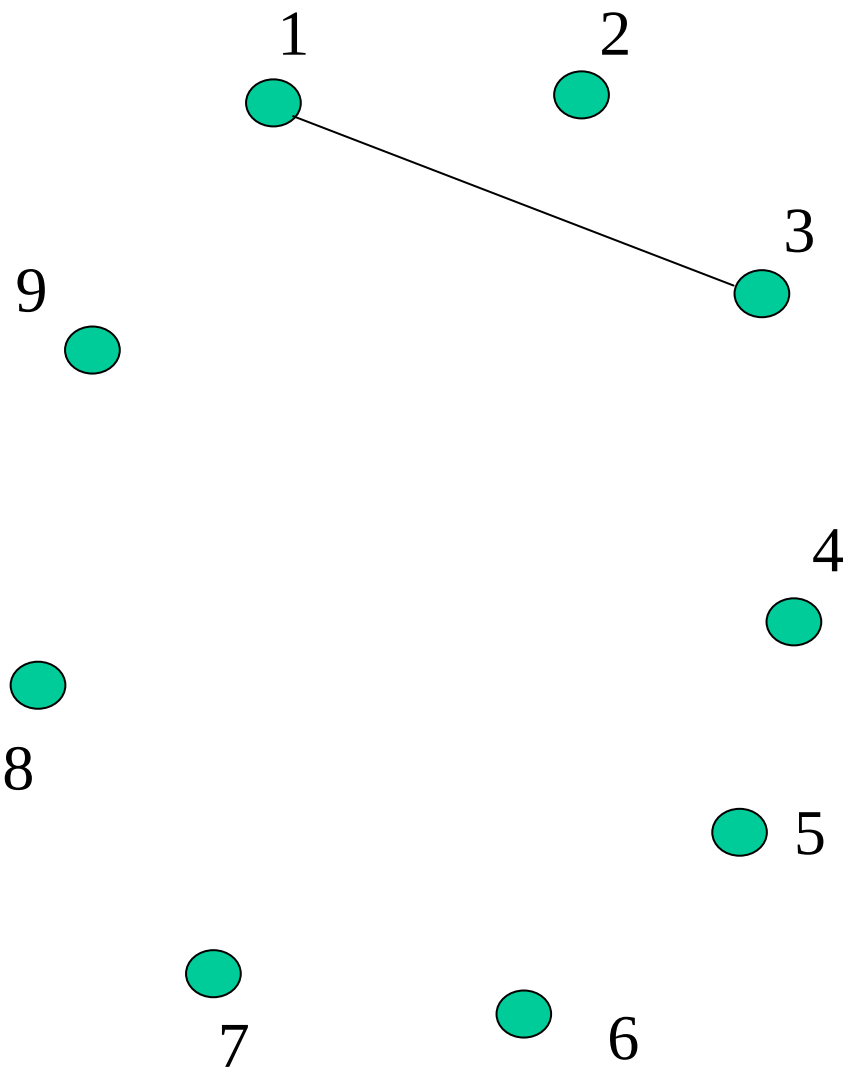
Código de longitud n – 2

Clave: vecino de la hoja con menor etiqueta
hasta que restan sólo dos vértices

Del código al árbol etiquetado

C 3 4 4 6 3 7 7

L ={1, 2, 3, 4, 5, 6, 7, 8, 9}



Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}

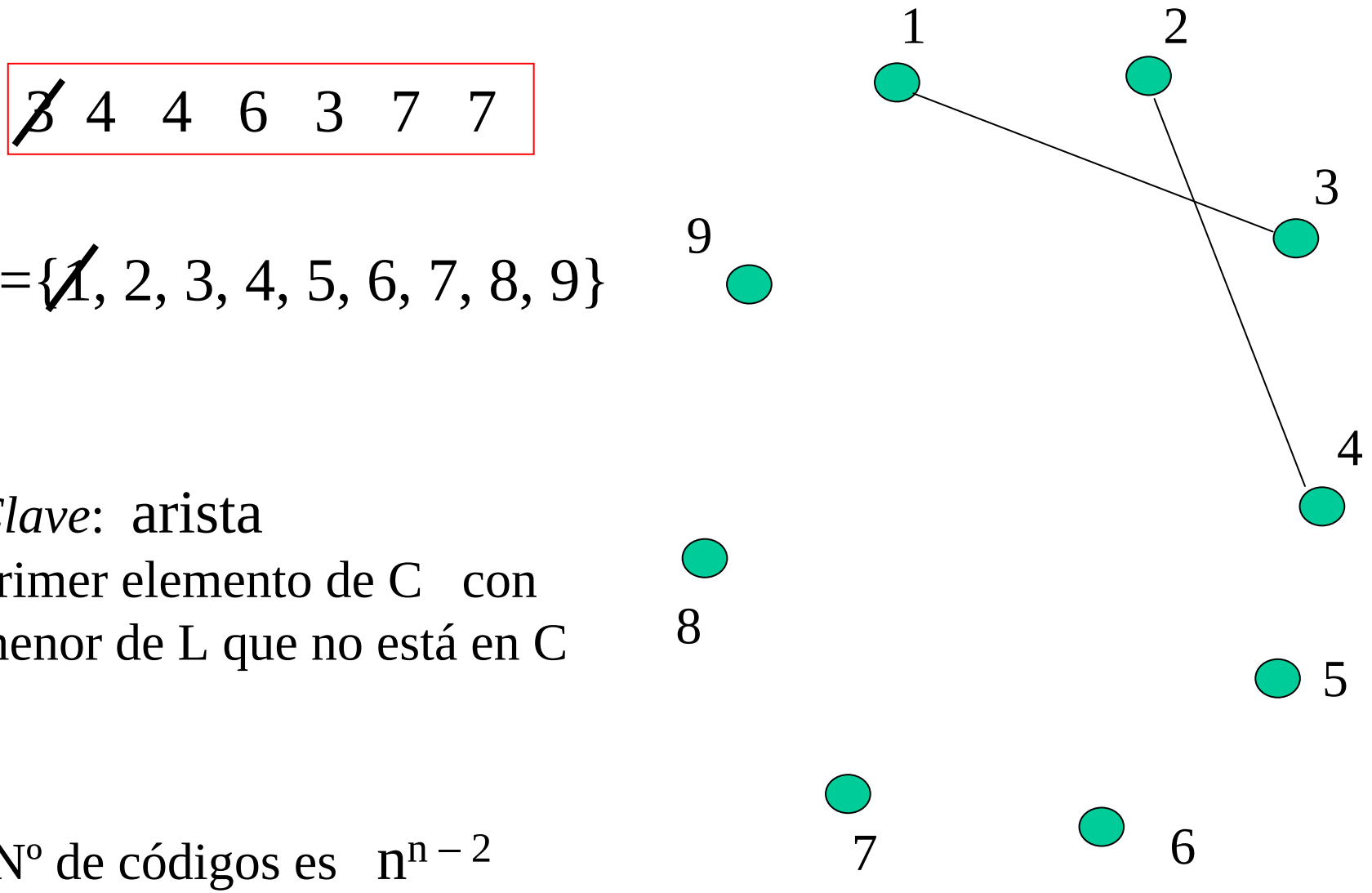
Del código al árbol etiquetado

C ~~3~~ 4 4 6 3 7 7

L = {~~1~~, 2, 3, 4, 5, 6, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



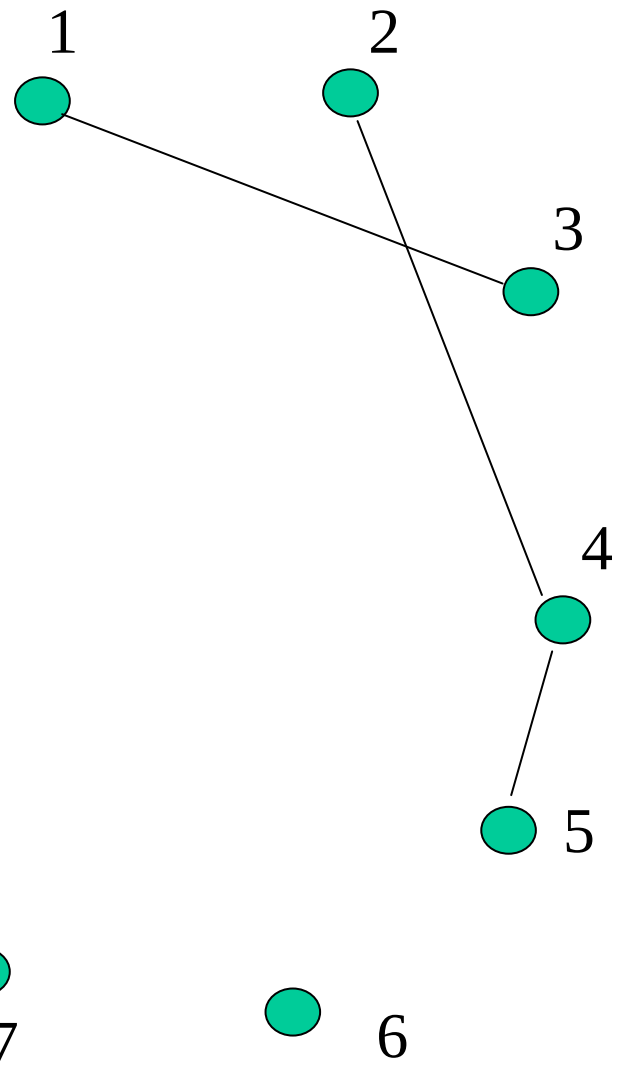
Del código al árbol etiquetado

C 3 4 4 6 3 7 7

L = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



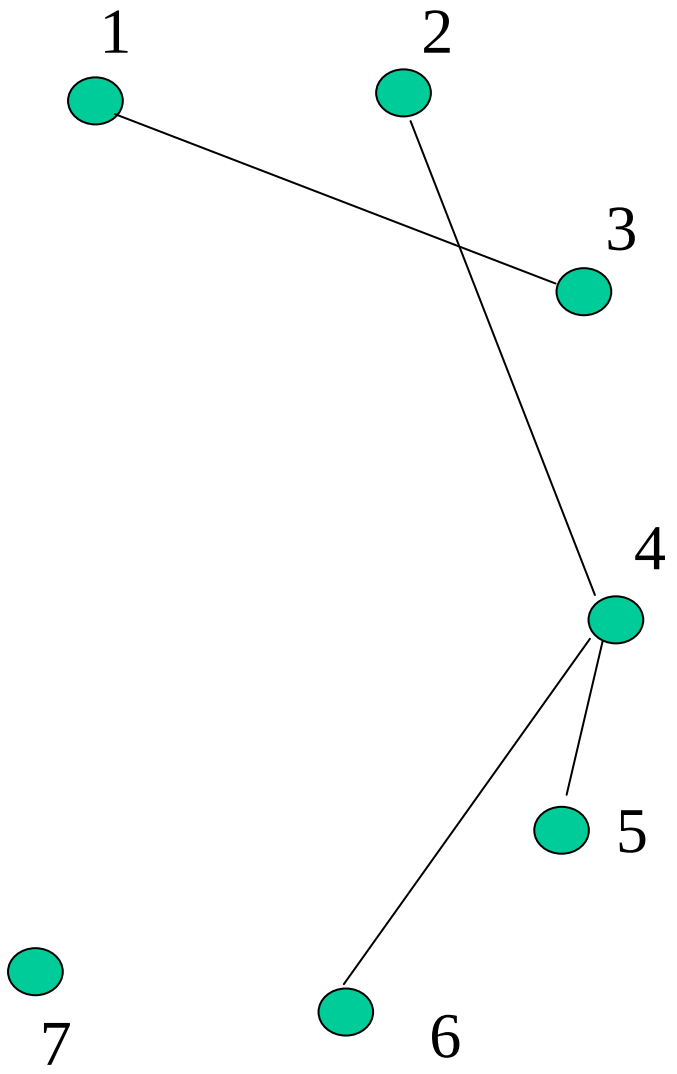
Del código al árbol etiquetado

C ~~3~~ ~~4~~ ~~4~~ 6 3 7 7

L = {~~1~~, ~~2~~, 3, 4, ~~5~~, 6, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



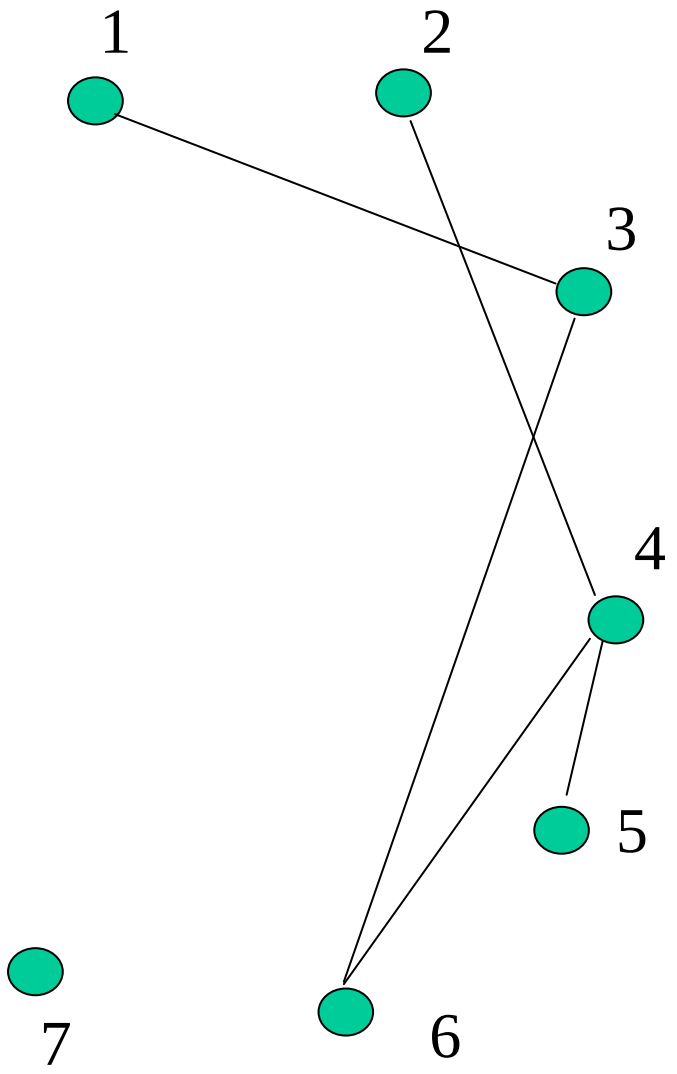
Del código al árbol etiquetado

C ~~3~~ ~~4~~ ~~4~~ ~~6~~ 3 7 7

L = {~~1~~, ~~2~~, 3, ~~4~~, ~~5~~, 6, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



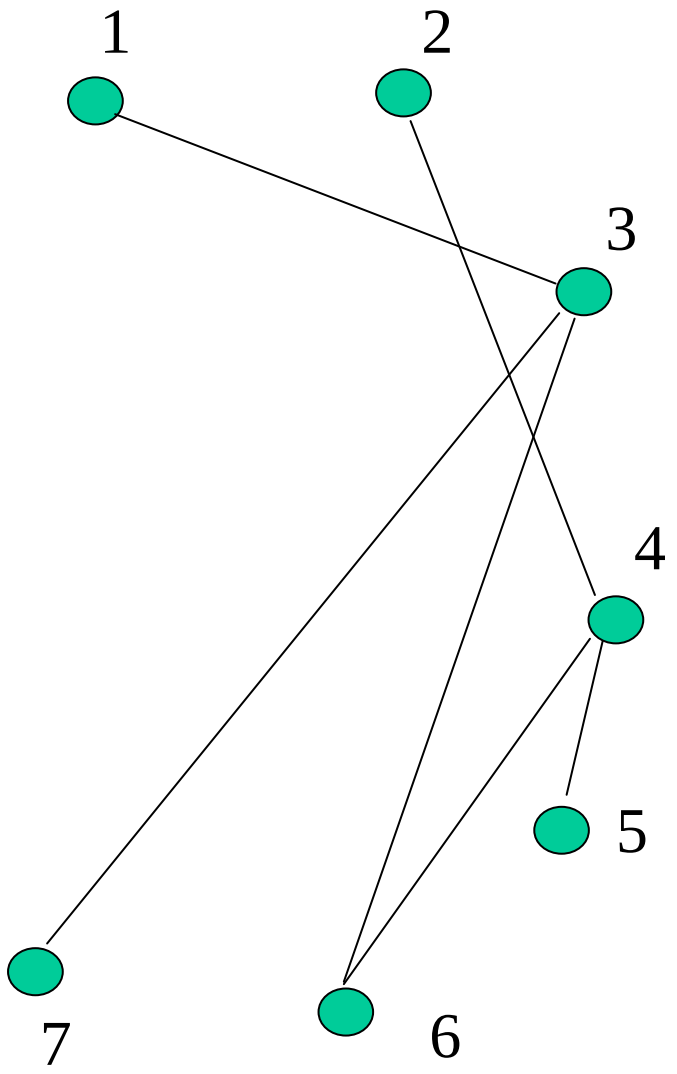
Del código al árbol etiquetado

C ~~3~~ ~~4~~ ~~4~~ ~~6~~ ~~3~~ 7 7

L = {~~1~~, ~~2~~, 3, ~~4~~, ~~5~~, ~~6~~, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



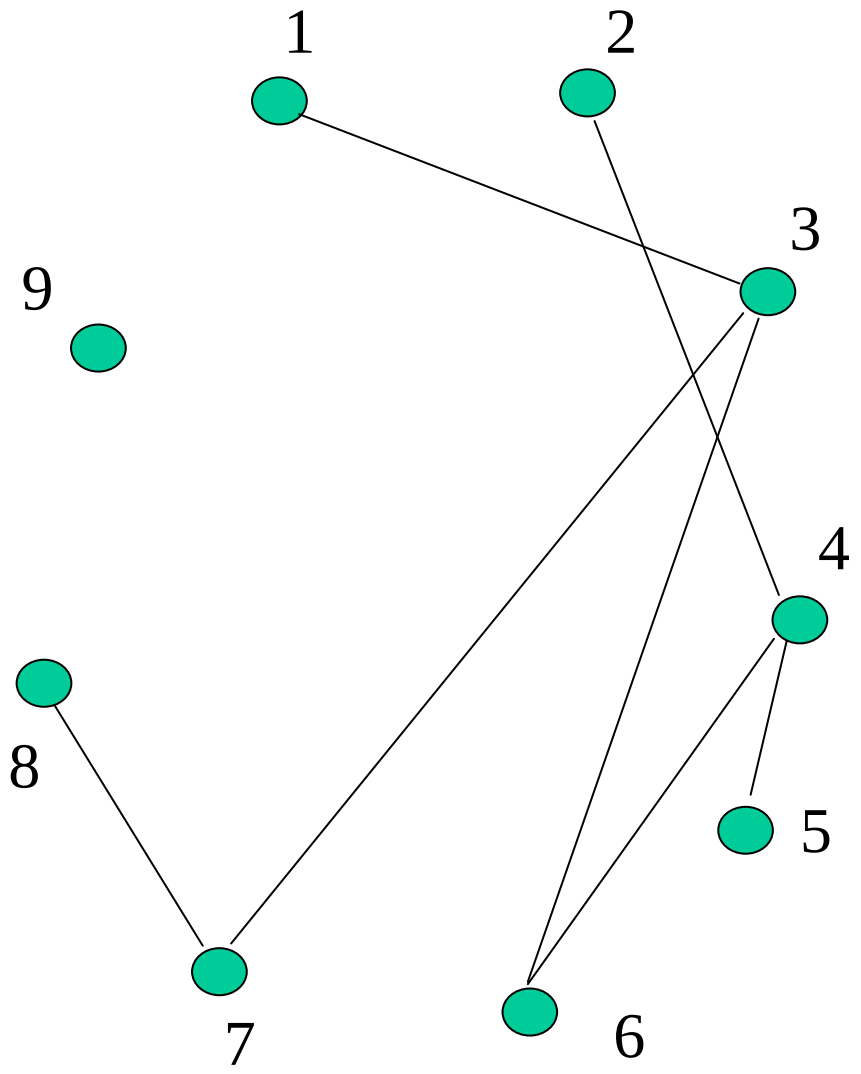
Del código al árbol etiquetado

C ~~3~~ ~~4~~ ~~4~~ ~~6~~ ~~3~~ ~~7~~ 7

L = {~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, 7, 8, 9}

Clave: arista
primer elemento de C con
menor de L que no está en C

Nº de códigos es n^{n-2}



Del código al árbol etiquetado

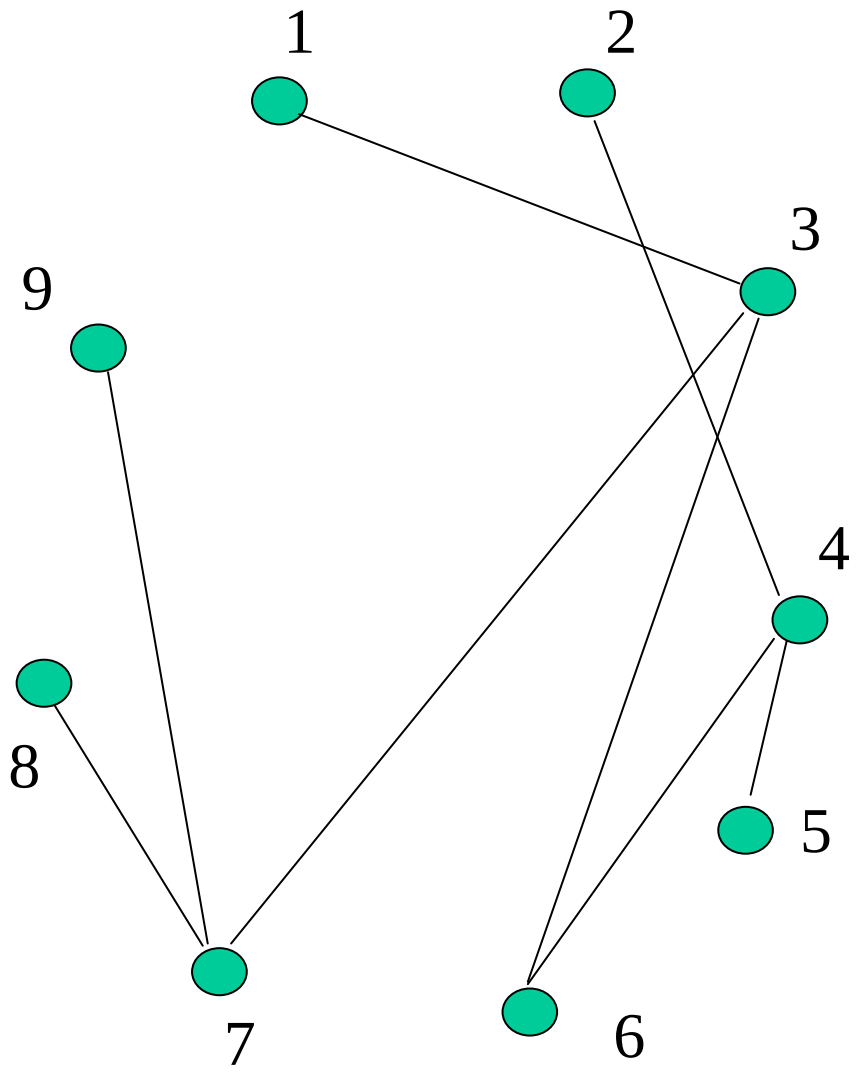
C ~~3 4 4 6 3 7 7~~

L = {~~1, 2, 3, 4, 5, 6~~, 7, ~~8~~, 9}

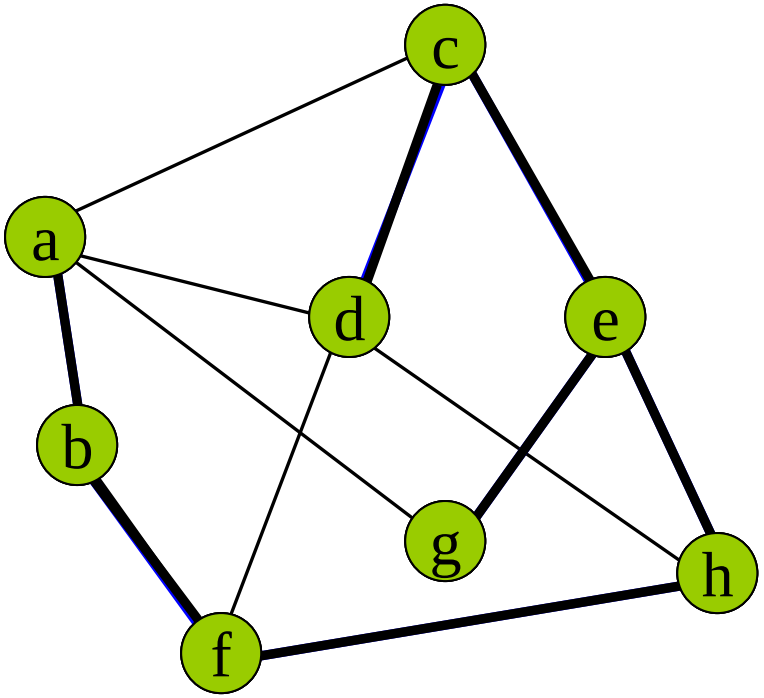
Clave: arista
primer elemento de C con
menor de L que no está en C

Últimos elementos en L forman
la última arista

Nº de códigos es n^{n-2}



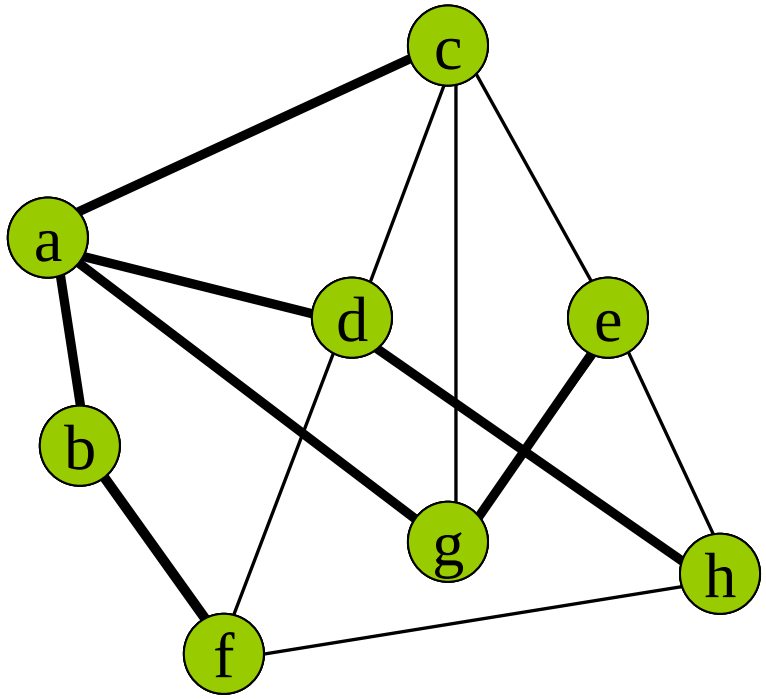
Recorrido en profundidad



Árbol de búsqueda T

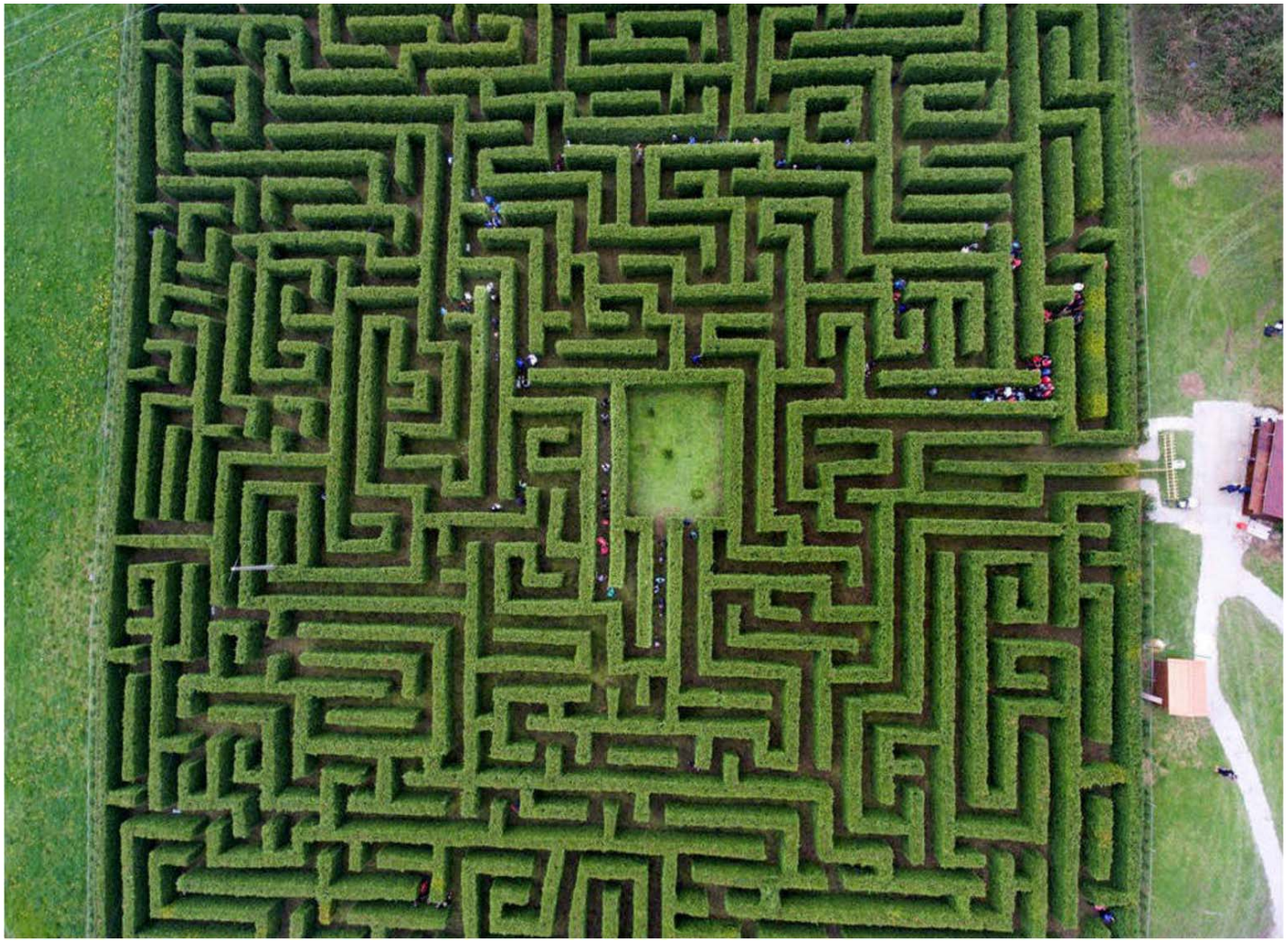
Camino: a
Camino: a,b
Camino: a,b,f
Camino: a,b,f,h
Camino: a,b,f,h,e
Camino: a,b,f,h,e,c
Camino: a,b,f,h,e,c,d
Camino: a,b,f,h,e,c
Camino: a,b,f,h,e
Camino: a,b,f,h,e,g

Recorrido en anchura



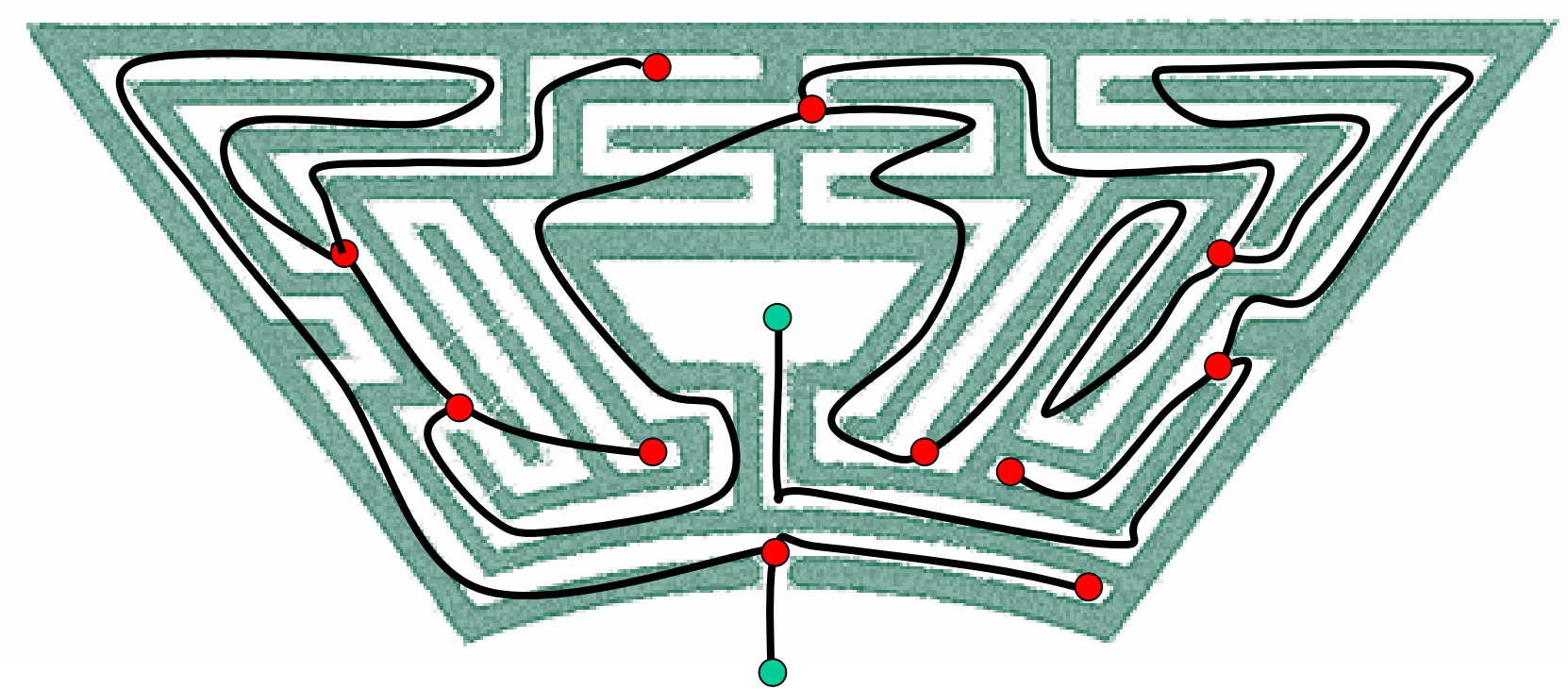
Frente: a
Frente: b,g,d,c
Frente: g,d,c,f
Frente: d,c,f,e
Frente: c,f,e,h
Frente: f,e,h
Frente: e,h
Frente: h

Laberinto de Villapresente

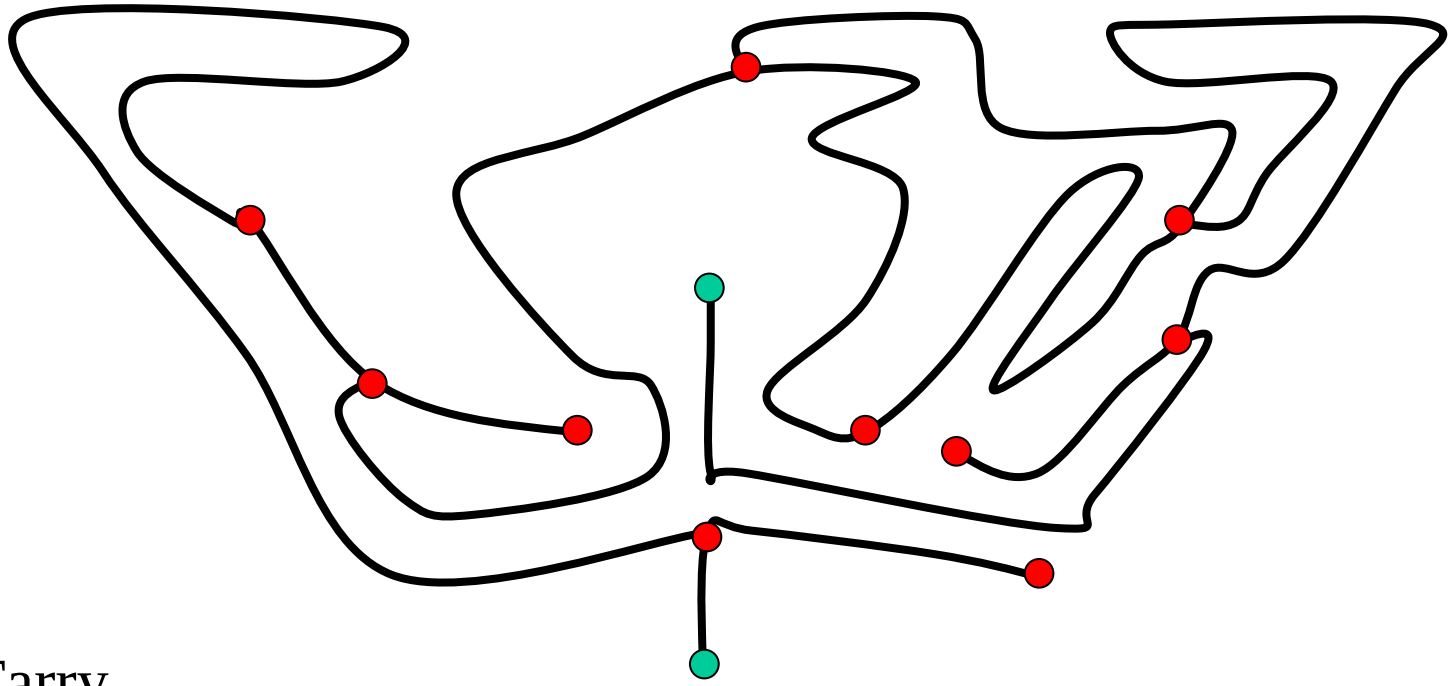


Laberinto de Hampton Court

Laberinto de Hampton Court



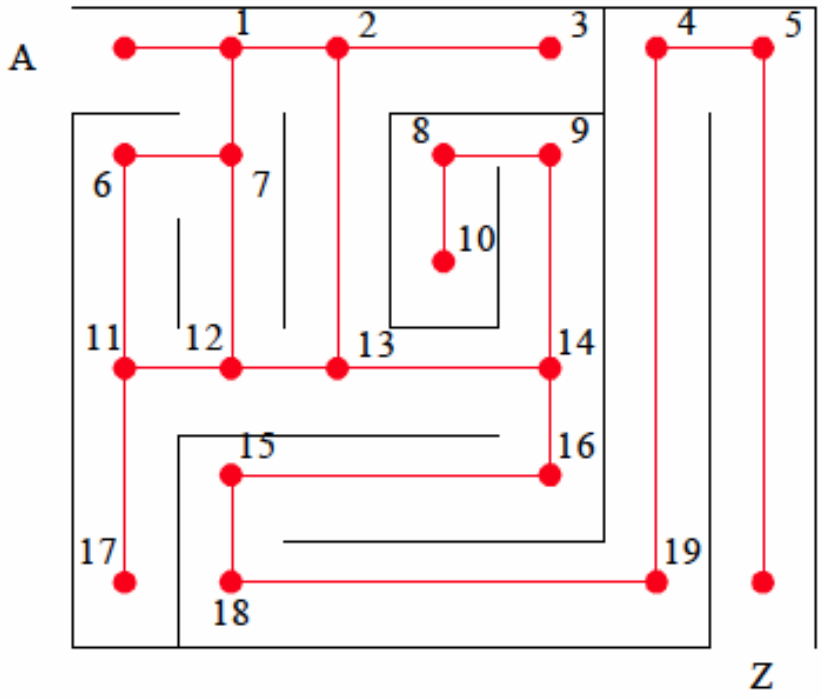
Laberinto de Hampton Court

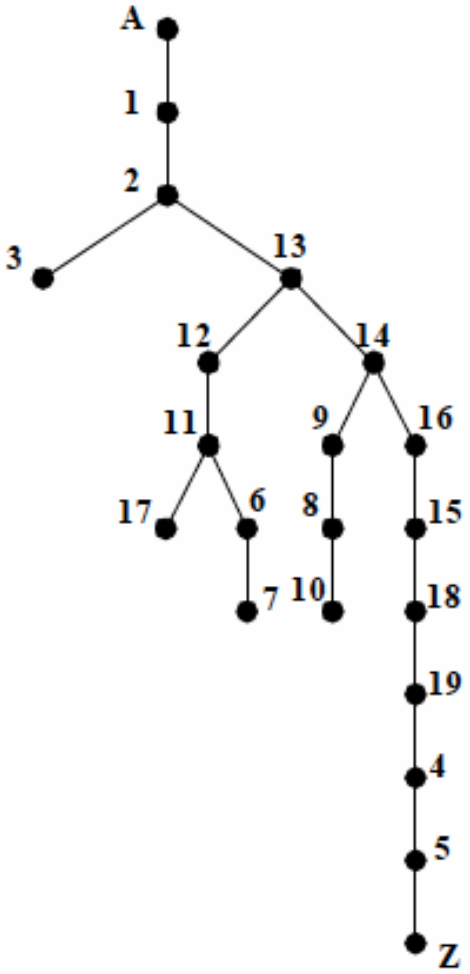


G. Tarry

“Cada vez que entramos en un pasadizo, dejamos dos marcas al comienzo. Cuando llegamos a una bifurcación nueva dejamos una marca en el pasadizo que nos llevó hasta allí; en cambio, si es una bifurcación que ya habíamos visitado antes, dejamos tres marcas. Cuando lleguemos a una bifurcación tenemos que continuar por los pasadizos que no tengan ninguna marca o los que tengan una sola”.

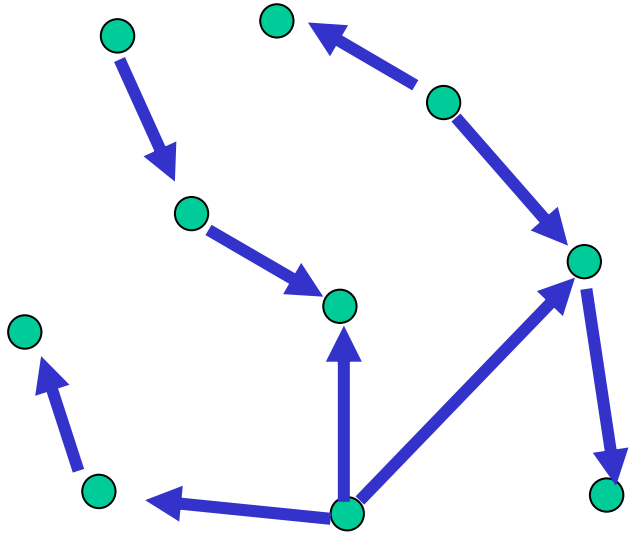
DFS y Laberintos





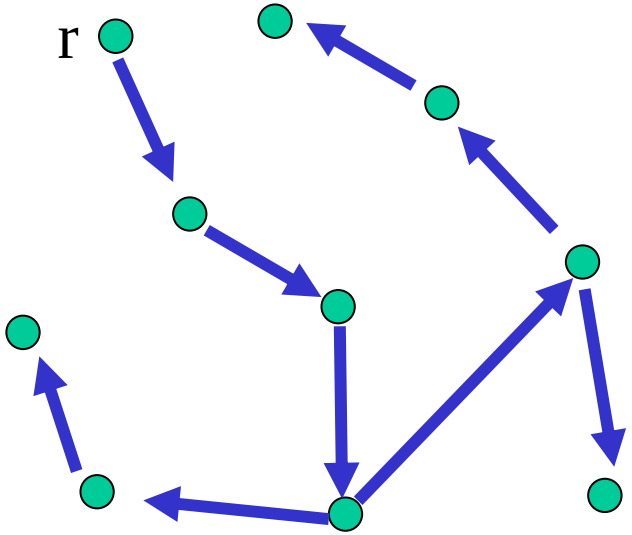
ÁRBOL DIRIGIDO

El grafo subyacente es un árbol

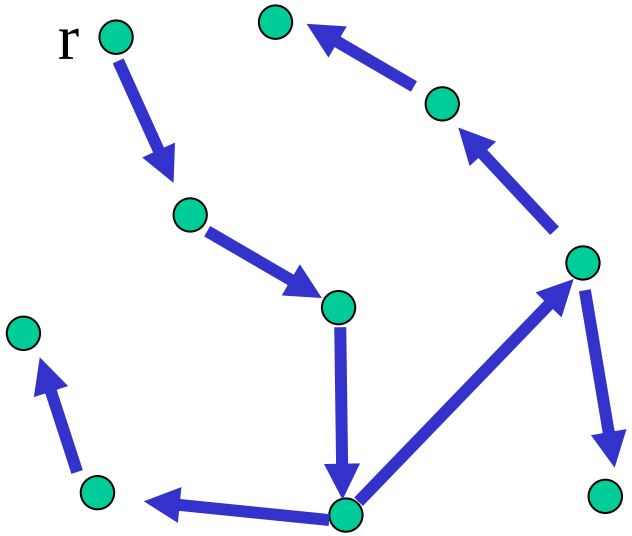


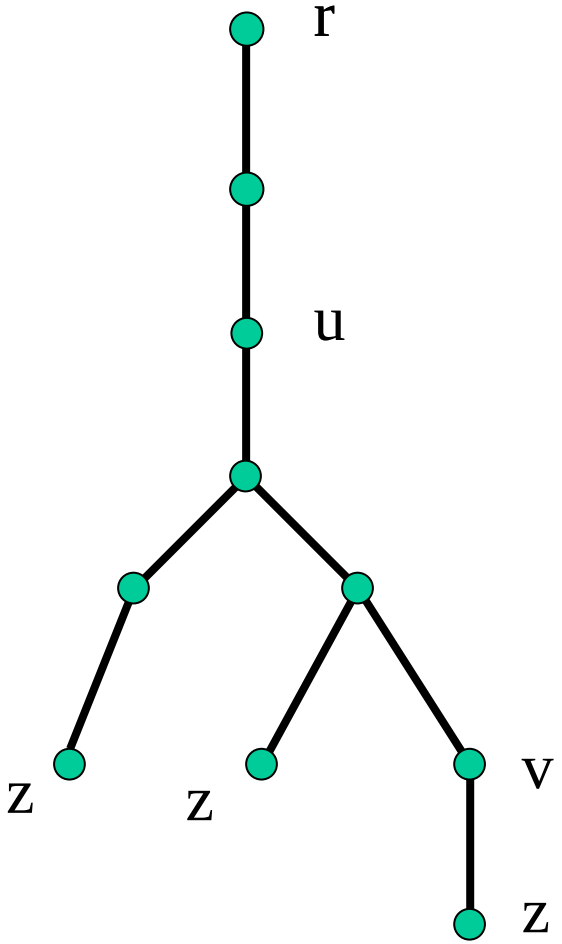
ÁRBOL CON RAÍZ

Existe r tal que para todo v hay un camino rv



ÁRBOL CON RAÍZ





Altura (T) = 6

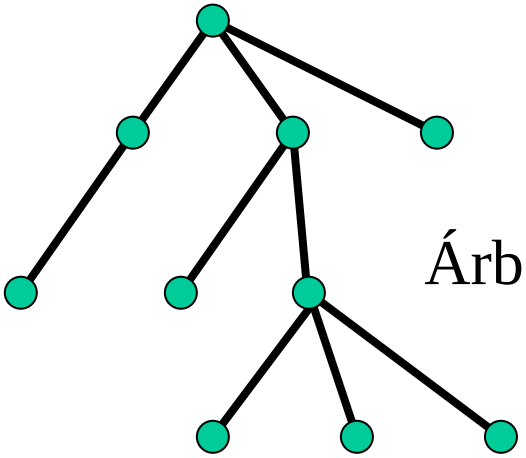
u, v vértices interiores

z hoja

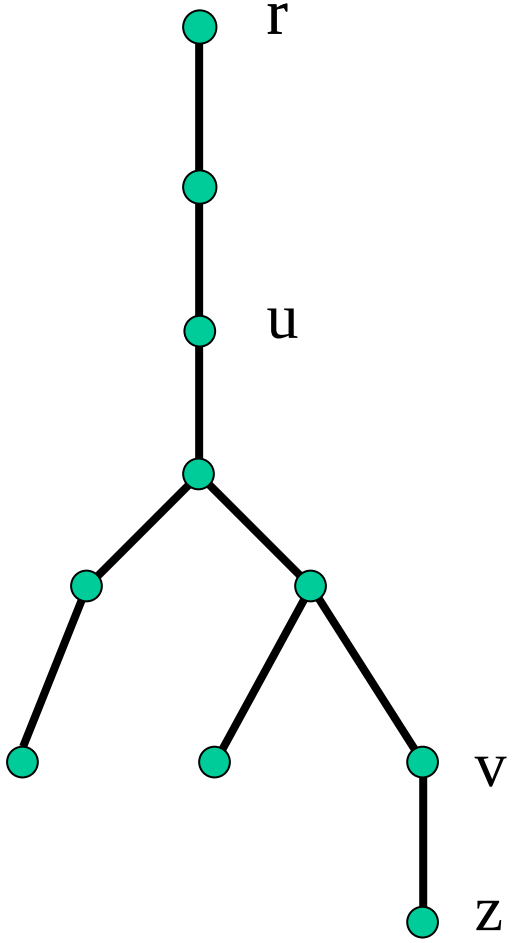
ÁRBOL CON RAÍZ

Árbol binario

Árbol m-ario
Cada padre a lo más m hijos



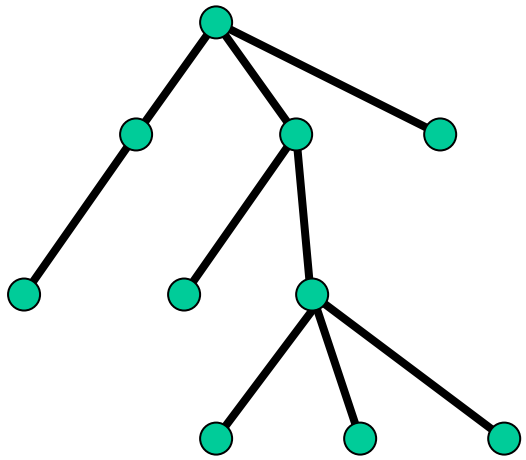
Árbol ternario



T es **equilibrado** si hojas en nivel h ó h – 1 (h=altura(T))

ÁRBOL CON RAÍZ

Árbol m-ario
Cada padre a lo más m hijos



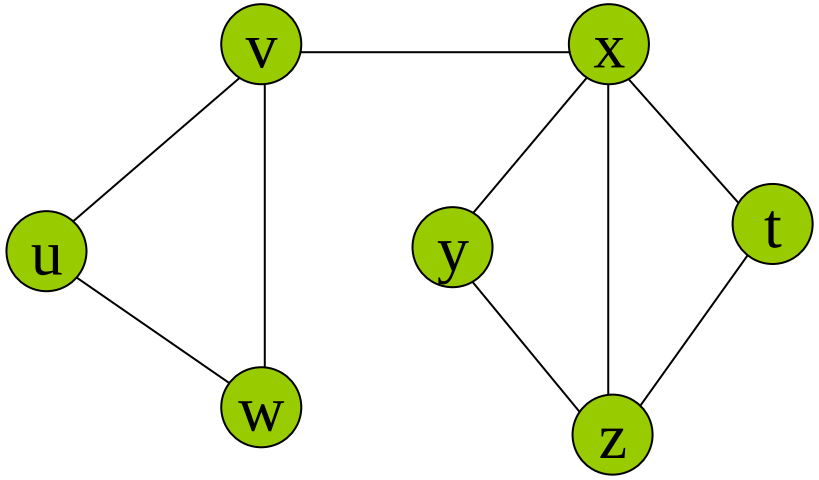
El número de hojas de un árbol ternario de altura h es, a lo más, 3^h

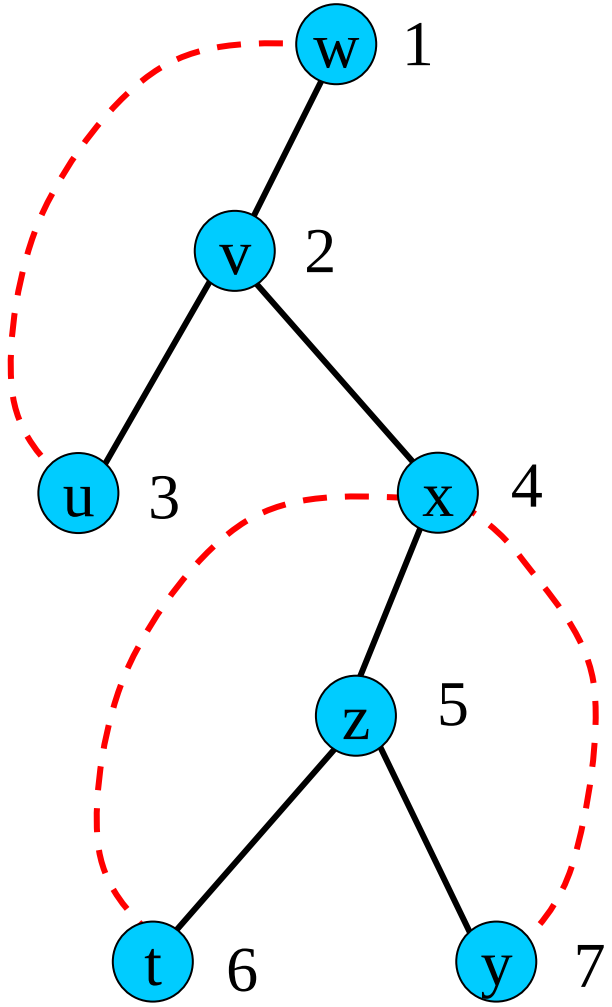
El número de hojas de un árbol m-ario de altura h es, a lo más, m^h

La altura de un árbol m-ario de ℓ hojas es, al menos, $\log_m \ell$

$$h \geq \log_m \ell$$

Recorrido en profundidad





Índice de búsqueda df(v)

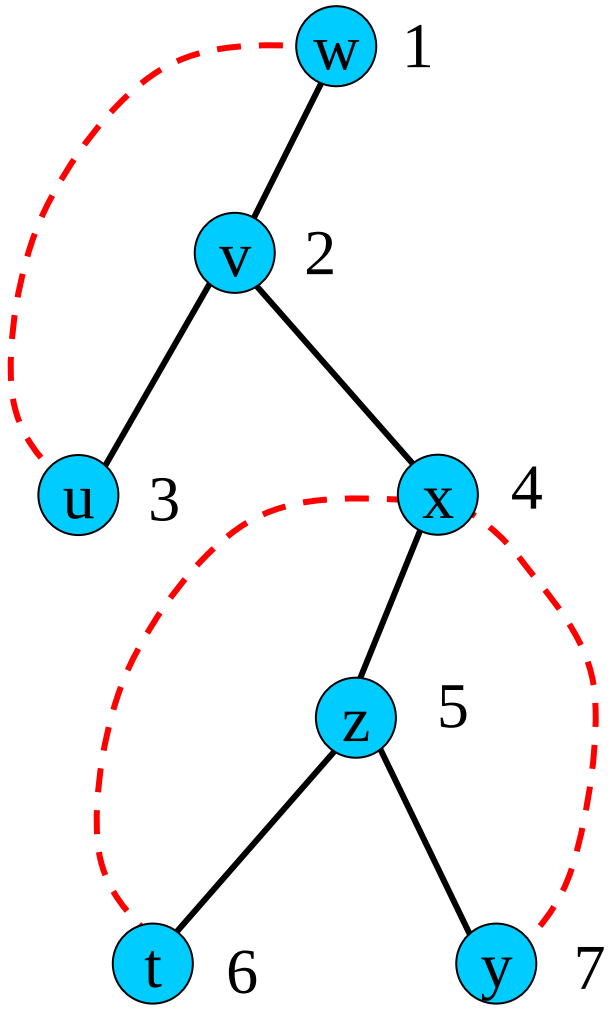
Aristas en T ——————

Aristas no en T - - - - - -
(de retroceso)

Recorrido en profundidad

Proposición 1

Si e=ab no es arista de T, df(a)< df(b) entonces a es ascendiente de b en el árbol T

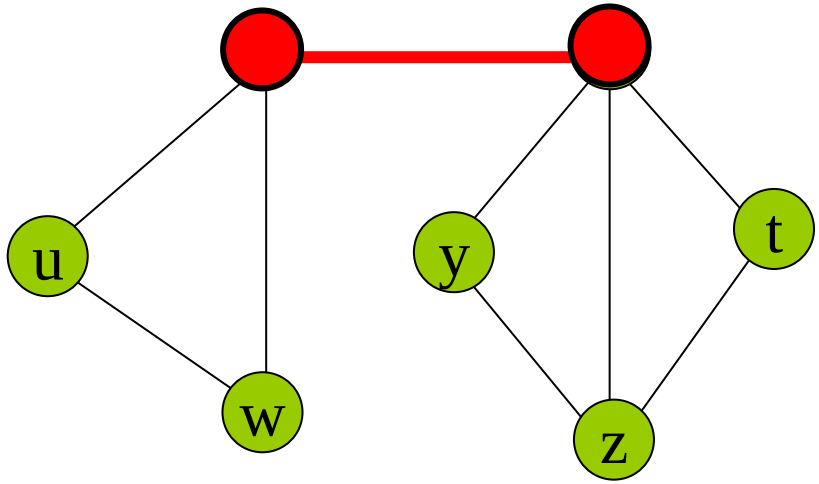


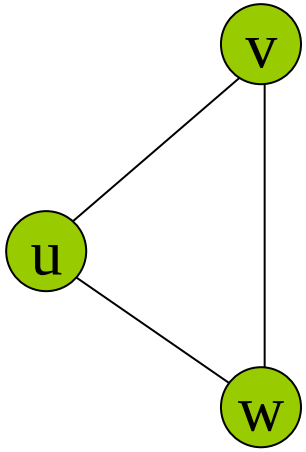
Cuando la búsqueda alcanza **a** la arista **e** está disponible. Como **e** $\notin$ T la búsqueda debe visitar **b** antes de retroceder más allá de **a**

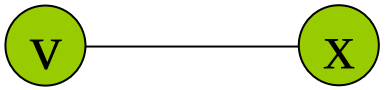
Recorrido en profundidad

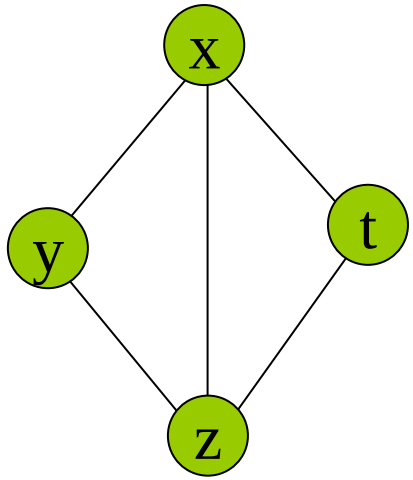
Aplicaciones

1. Componentes conexas
2. Vértices-corte
3. Aristas puente
4. Bloques





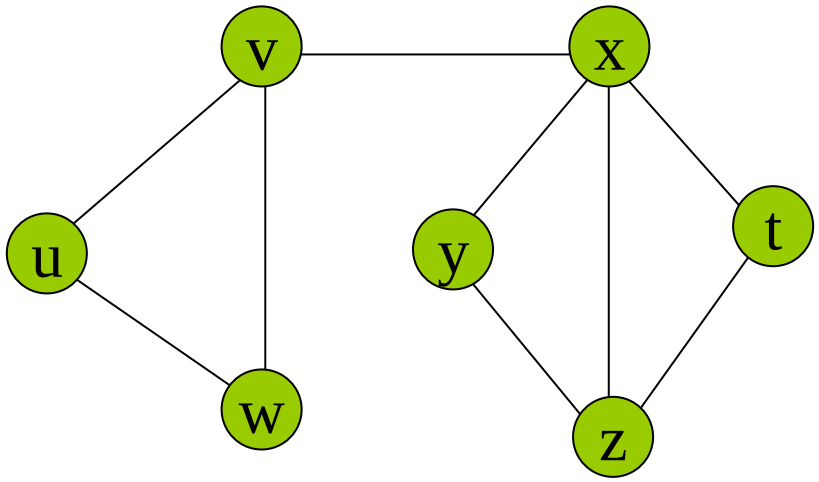




Vértices corte en un grafo conexo

Proposición 2

v es vértice-corte $\Leftrightarrow$ $\exists$ **u**, **x** tales que **v** está en todos los caminos de **u** a **x**

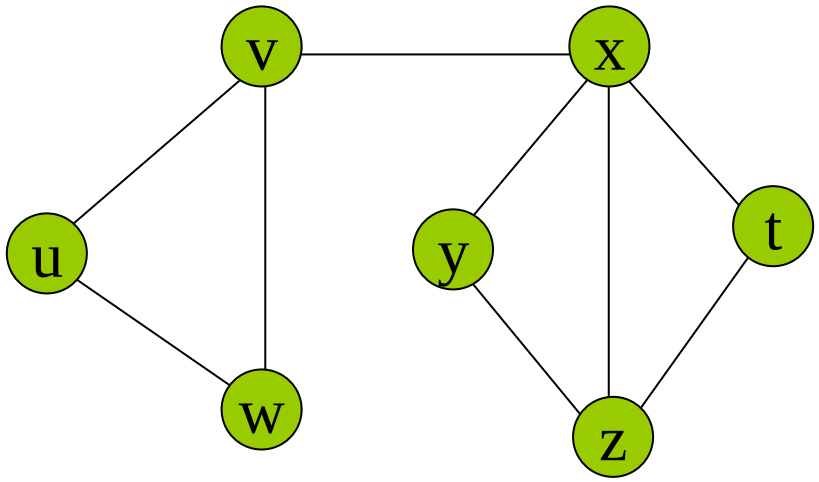


Vértices corte en un grafo conexo

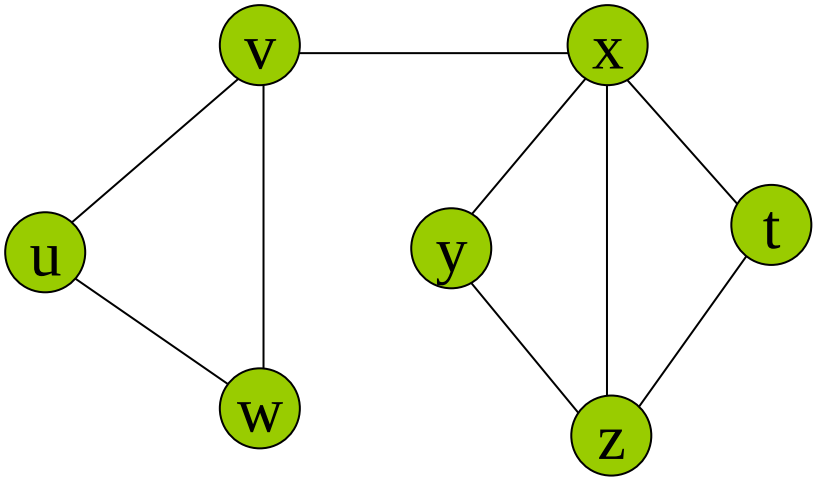
Proposición 3

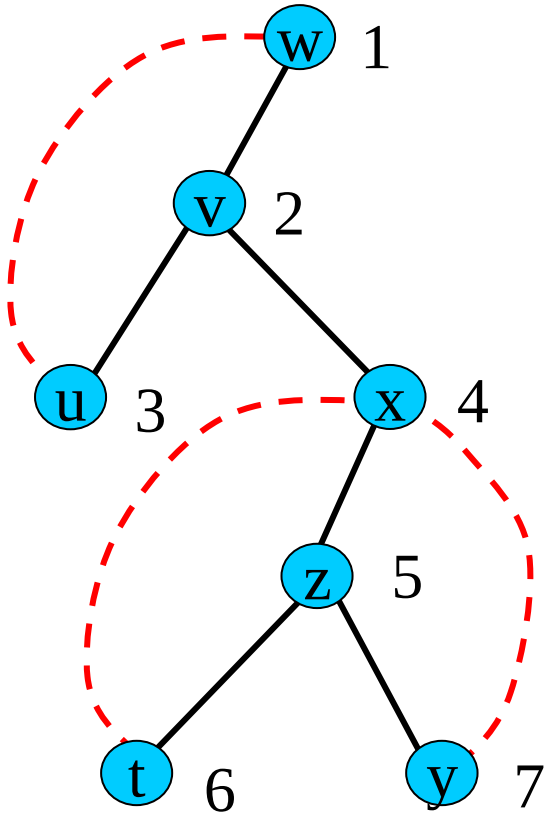
Sea T árbol DFS de G conexo, **r** la raíz del árbol

r es vértice-corte ⇔ **r** tiene más de un hijo en T



Vértices corte en un grafo conexo

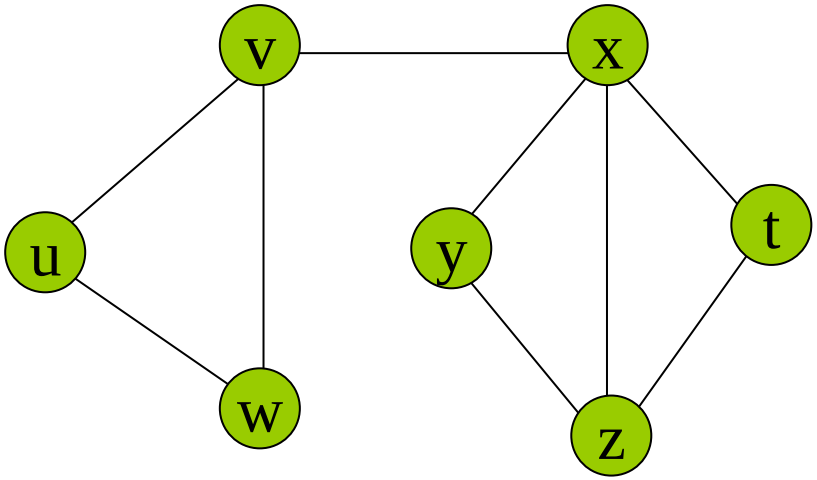


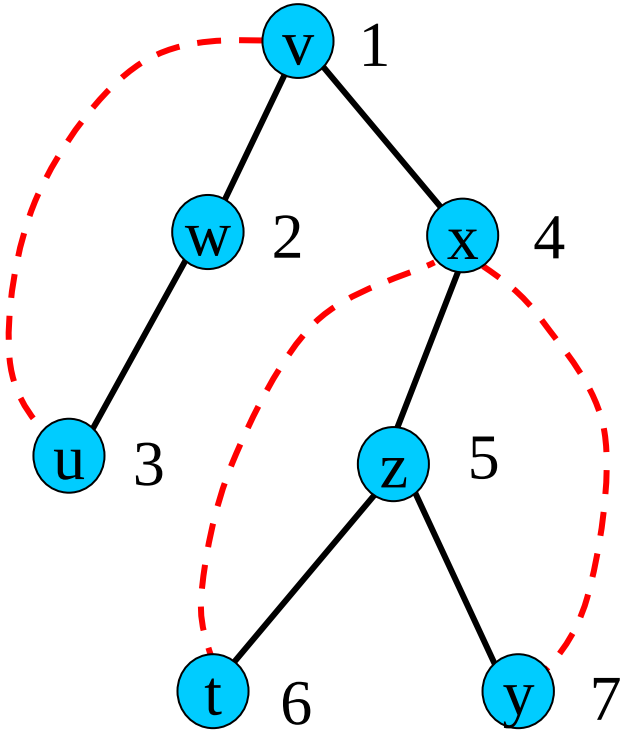


Dem.

$\Rightarrow$) Si r tiene sólo un hijo v el subárbol T_v es generador de G-r, luego G-r es conexo

Vértices corte en un grafo conexo



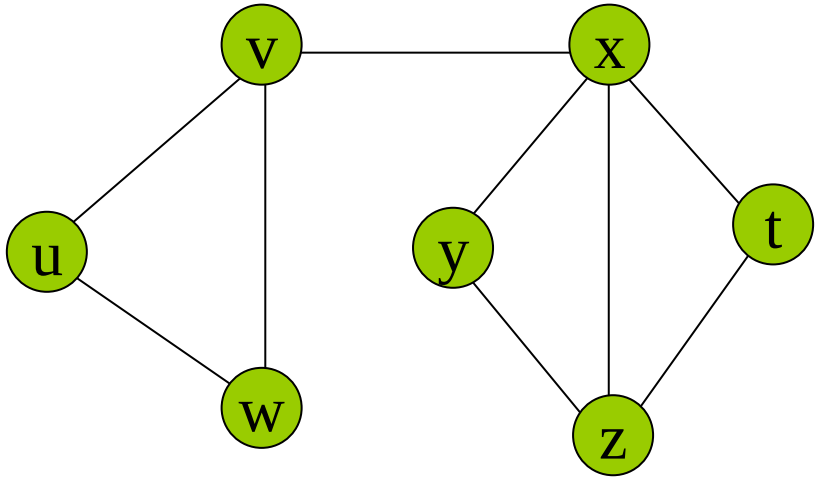


⇐) Si r tiene dos hijos x,w entonces todos los caminos de x a w pasan por r luego por (Prop. 2) r es un vértice corte

Vértices corte en un grafo conexo

Sea T árbol DFS de G conexo, **v** vértice no raíz del árbol

v es vértice-corte ⇔ **v** tiene un hijo z tal que ningún descendiente de z se une a un antecesor de v por una arista que no está en T



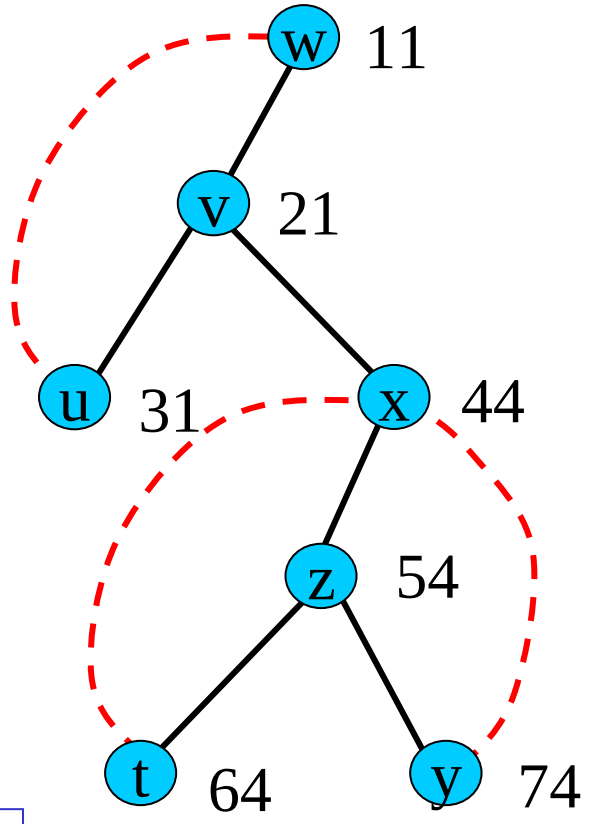
Vértices corte en un grafo conexo

Pongamos otra etiqueta $l(a)$
en cada vértice a, además de df(a)

Consideramos todos los vértices b accesibles desde a por un camino (puede ser vacío) formado por un camino dirigido en T seguido por a lo sumo una arista de retroceso.

$l(a)$=min{df(b) para todo vértice b de T que se alcanza desde z con un camino z--t que termina con una arista no de T}

Siempre es $l(a) \leq$ df(a)



a no raíz es vértice-corte $\Leftrightarrow$
existe una arista en T ***ab*** tal que
$l(\boldsymbol{b}) \geq$ df($\boldsymbol{a}$)

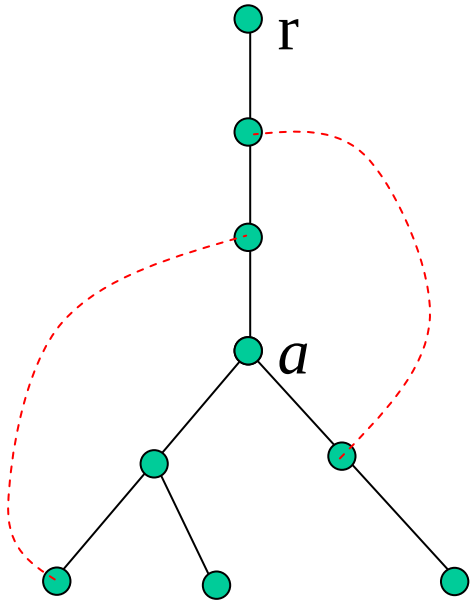
Proposición 4

Vértices corte en un grafo conexo

a no raíz es vértice-corte $\Leftrightarrow$ existe una arista en T ***ab*** tal que $l(\boldsymbol{b}) \geq \mathrm{df}(\boldsymbol{a})$

Dem.

$\Rightarrow$) Supongamos que $l(b) < \mathrm{df}(a)$ para todo hijo b de a



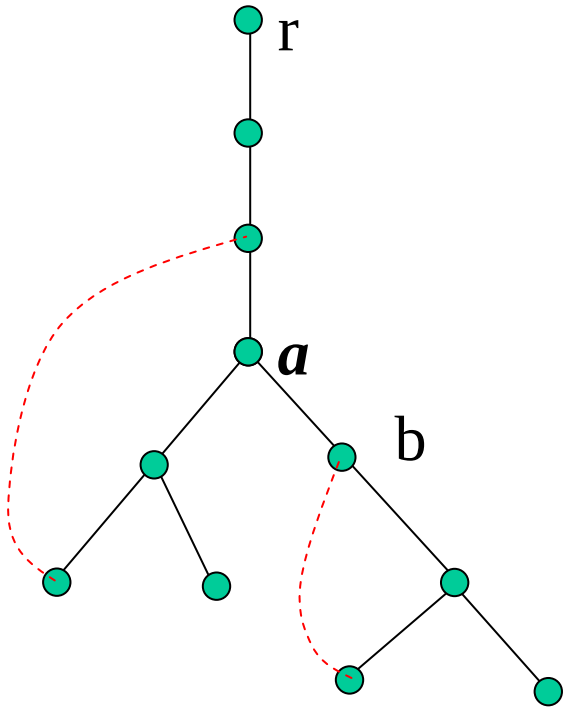
Dos vértices cualesquiera de T_a se pueden unir por un camino que no pasa por v. así por (Prop 2) v no es vértice-corte

Vértices corte en un grafo conexo

a no raíz es vértice-corte $\Leftrightarrow$ existe una arista en T ***ab*** tal que $l(\boldsymbol{b}) \geq \mathrm{df}(\boldsymbol{a})$

Dem.

$\Leftarrow$) Si ***a*** tiene un hijo **b** tal que $l(\mathbf{b}) \geq \mathrm{df}(\boldsymbol{a})$



entonces no hay ninguna arista roja de retroceso que enlace los descendientes de b con los ascendientes de *a*.
En particular TODOS los caminos de **b** a **r** pasan por el vértice ***a***. Luego ***a*** es vértice-corte

Vértices corte en un grafo conexo

ALGORITMO

Entrada: Un grafo G conexo
Salida: K, conjunto de vértices-corte
Inicializar K=∅
Elegir **r** vértice de G y construir T árbol de búsqueda con raíz **r**
    Si **r** tiene más de un hijo en T, añadir **r** a T
    Para cada **z** vértice de T, calcular $l(\mathbf{z})$
    Para cada **v** no raíz
        si **v** tiene un hijo **z** con $l(\mathbf{z}) \geq df(\mathbf{v})$, añadir v a K
Devolver K

Vértices corte en un grafo conexo

ALGORITMO

Cálculo de l(z)

Para todo vértice v de T, *l*(v) es el menor de los siguientes valores:

- df(v)
- df(z) para todo z unido a v por una arista de retroceso
- *l*(y) para todo hijo y de v

Esto permite calcular las etiquetas *l* al tiempo que se construye T

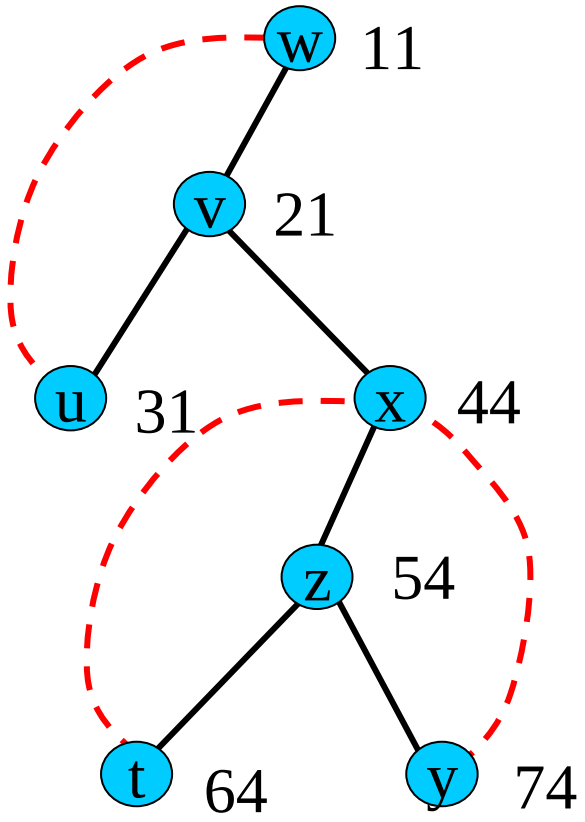
Complejidad O(q)

Aristas puente en un grafo conexo

Proposición 5

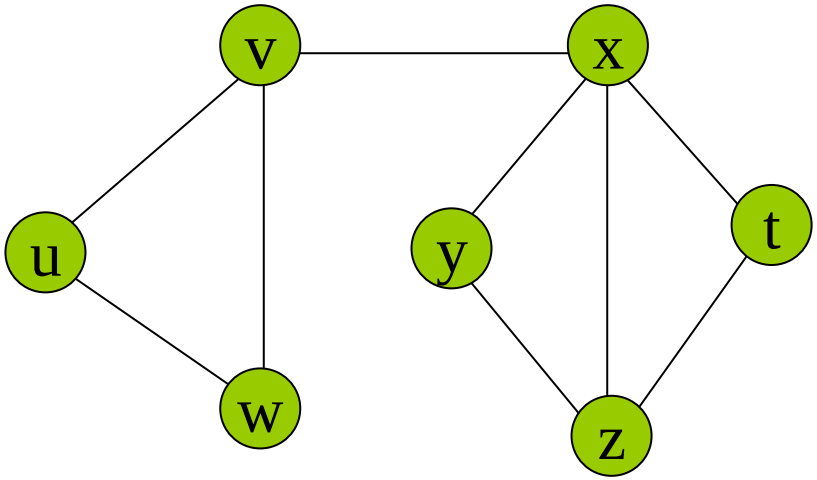
Sea T árbol DFS de G conexo, **uv** arista de G con df(u)<df(v)

uv es puente $\Leftrightarrow$ **uv** $\in$ T y l(v)>df(u)

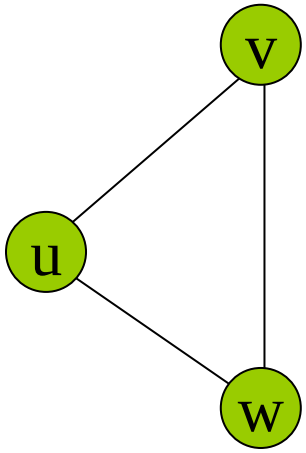


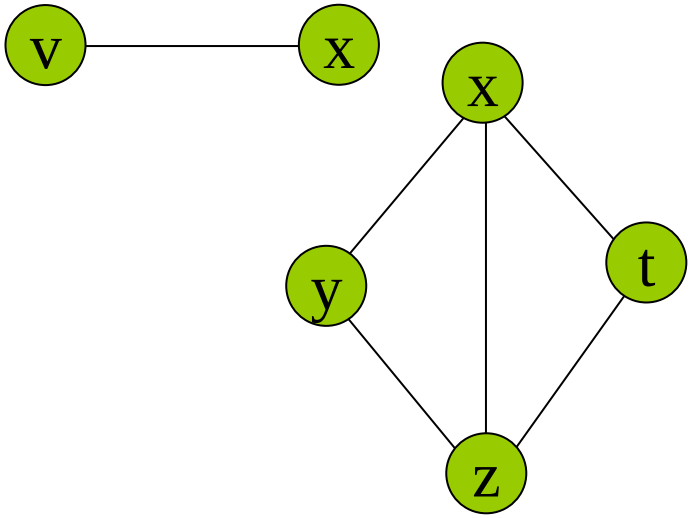
Bloques

Sea G un grafo. Un bloque de G es un subgrafo conexo maximal tal que ninguno de sus vértices es vértice-corte



Un grafo G y sus bloques





En la aplicación **IAGraph** están implementadas las búsquedas en profundidad y en anchura así como el doble etiquetado para la detección de vértices-corte y aristas puente

http://www.dma.fi.upm.es/personal/gregorio/grafos/web/iagraph/