Matemáticas discretas II: Teoría de Grafos III

Universidad del Valle
EISC

Septiembre 2018

1 Isomorfismos

2 Algoritmos sobre grafos
- Coloreo de grafos
- Algoritmo de Dijkstra
- Búsqueda en amplitud
- Búsqueda en profundidad

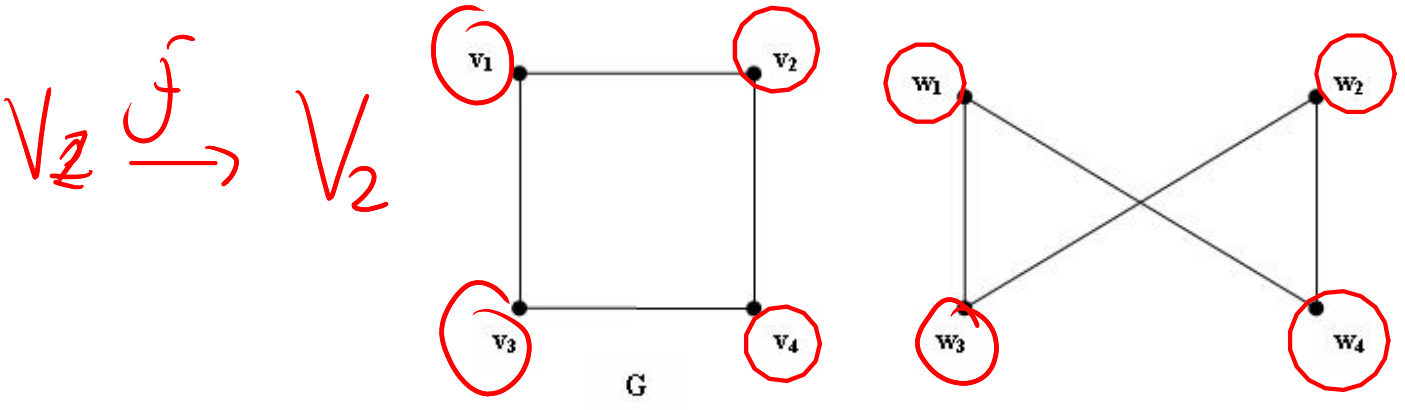
Contenido

Universidad del Valle

Isomorfismos de Grafos

Los grafos simples $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son `isomorfos` si hay una función biyectiva f de V_1 en V_2 con la propiedad de que, para cada par de vértices $u, v \in V_1$, u y v son adyacentes en G_1 si, y sólo si, $f(u)$ y $f(v)$ son adyacentes en G_2. Se dice que esta función f es un isomorfismo.

Ejemplo. Los grafos $G = (V, E)$ y $H = (W, F)$ en la siguiente figura son isomorfos.



La función f con $f(v_1) = w_1, f(v_2) = w_4, f(v_3) = w_3, f(v_4) = w_2$

Isomorfismos

Definición 2.

Sean $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ dos grafos. Una función $f : V_1 \longrightarrow V_2$ es un isomorfismo de grafos si:

1. f es inyectiva y sobreyectiva. → biyectiva
2. $\forall a, b \in V_1, \{a, b\} \in E_1$ si y sólo si $\{f(a), f(b)\} \in E_2$.

Cuando existe tal función f se dice que G_1 y G_2 son isomorfos.

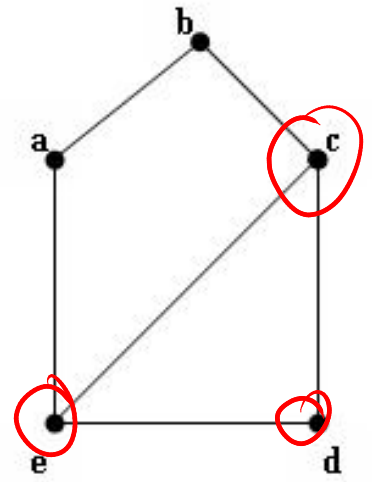
Ejemplo. Sean los grafos G y H son isomorfos?

$G = \{3, 3, 2, 2, 2\}$

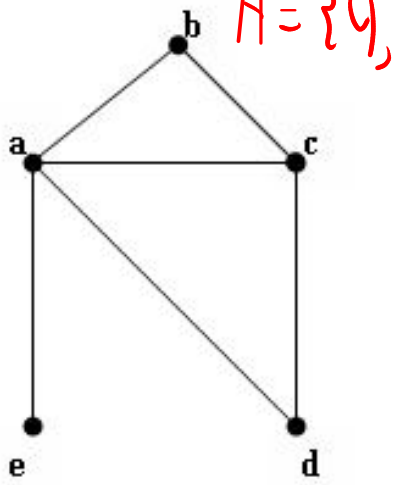
$H = \{4, 3, 2, 2, 1\}$

$V_2 \xrightarrow{f} V_2$

9 → 9
6 → 6
c → c
d → d
e → e



G



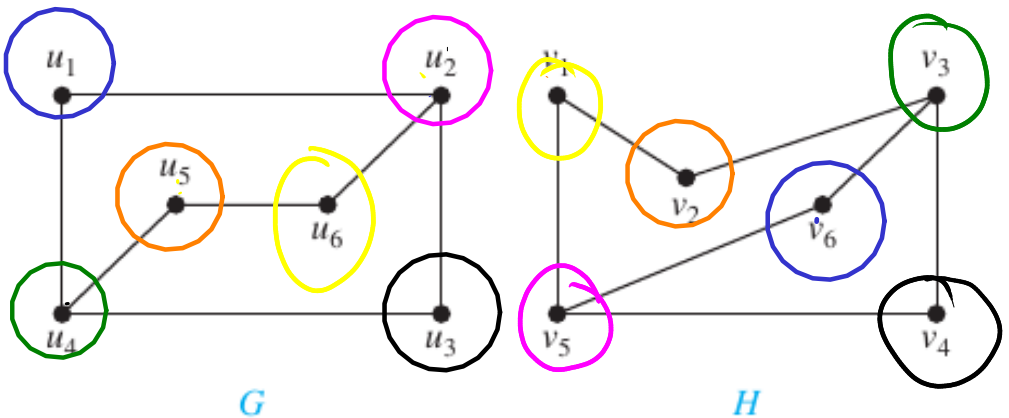
H

No son isomorfos por varias razones:

- H tiene el vértice e de grado 1, mientras que G no tiene.
- Si cualquiera de estas cantidades toma valores diferentes en dos grafos simples, dichos grafos simples no pueden ser isomorfos.
- Ahora el que estas cantidades sean iguales no garantiza que los grafos G y H sean isomorfos.

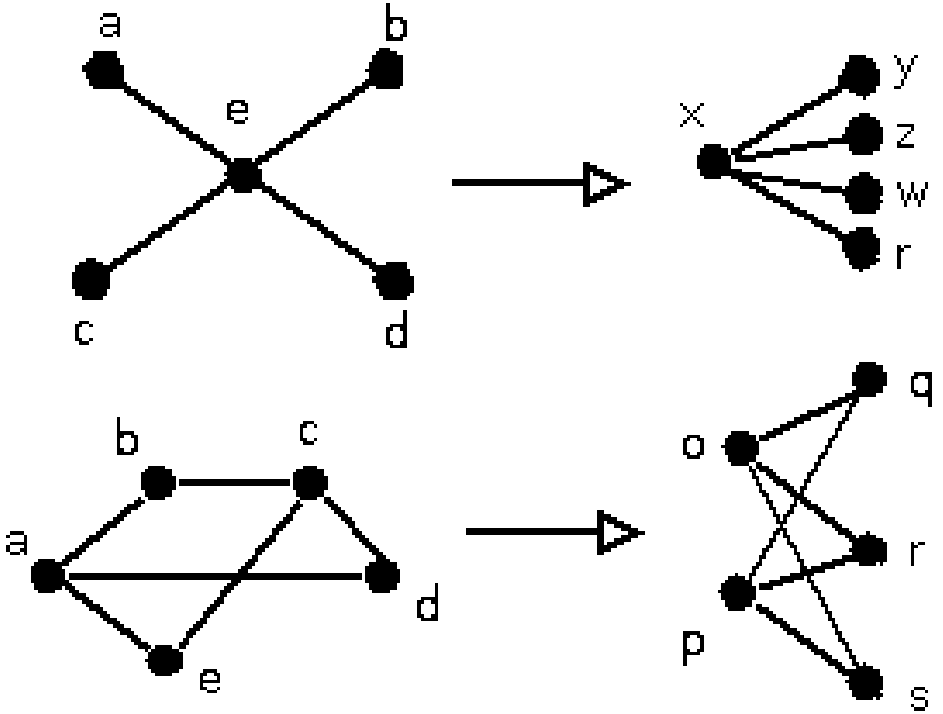
Isomorfismos

Ejemplo. Determinar si H y G son isomorfos.



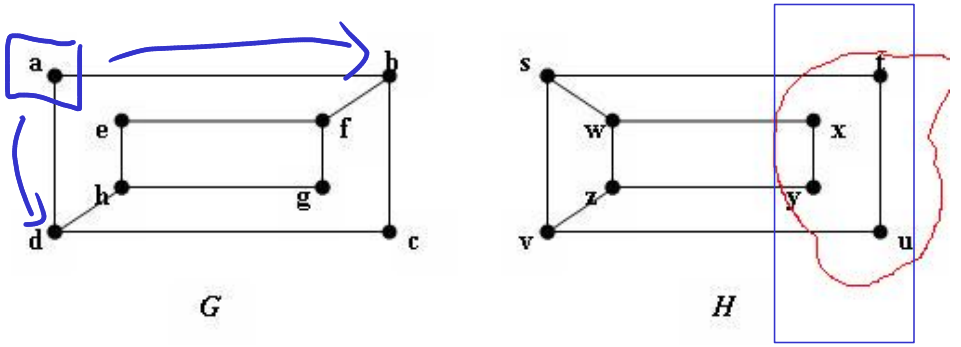
$$\mathbf{A}_G = \begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}, \qquad \mathbf{A}_H = \begin{array}{c} \\ v_6 \\ v_3 \\ v_4 \\ v_5 \\ v_1 \\ v_2 \end{array} \begin{array}{c} \begin{array}{cccccc} v_6 & v_3 & v_4 & v_5 & v_1 & v_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Isomorfismos



Isomorfismos

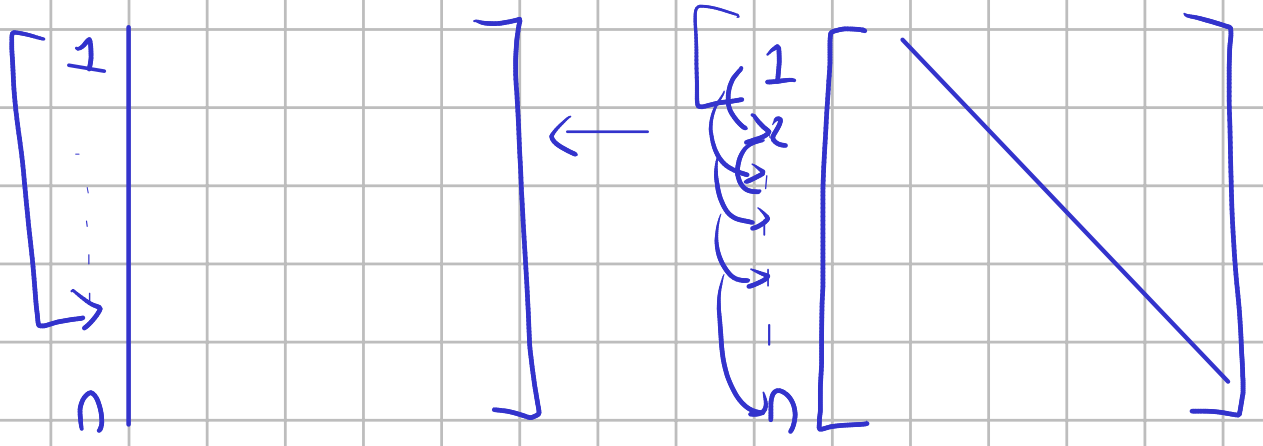
Ejemplo. Sean los grafos G y H son isomorfos?



- No son isomorfos por que por ejemplo $\delta(a) = 2$ en G y si observamos a debería corresponder con los vértices $t, u, x,$ o y de H, ya que estos son los vértices de grado dos de H. Sin embargo, cada uno de estos cuatro vértices de H son adyacentes a otros de grado dos, lo que no es cierto en a. Pasa lo mismo con c, g, e en G
- Los mejores algoritmos para determinar si dos grafos son isomorfos tienen complejidad exponencial en el peor caso.
- El mejor algoritmmo capaz de determinar si dos grafos son isomormos con 100 vértices es el NAUTY y se consigue en internet.

Universidad del Valle

$|V| = n$ $\quad O(n^2)$

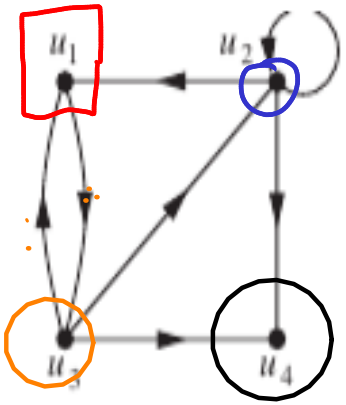


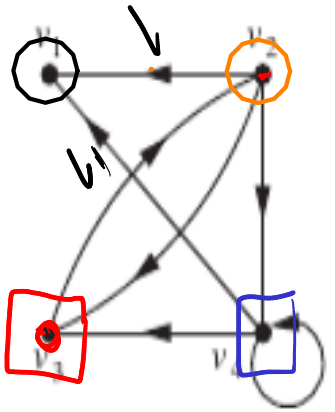
$n-1 \times n-2 \times n-3 \times \quad \times 1 = O(n!)$

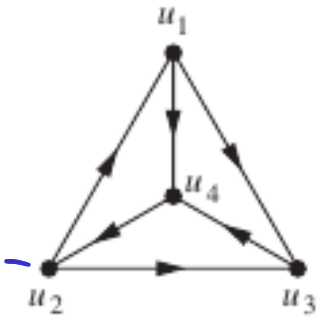
1- 8 2- 5 3- 6

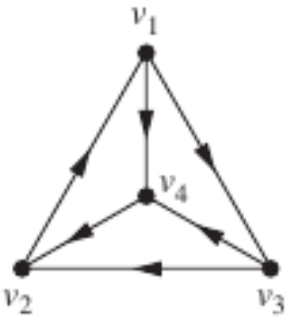
Isomorfismos

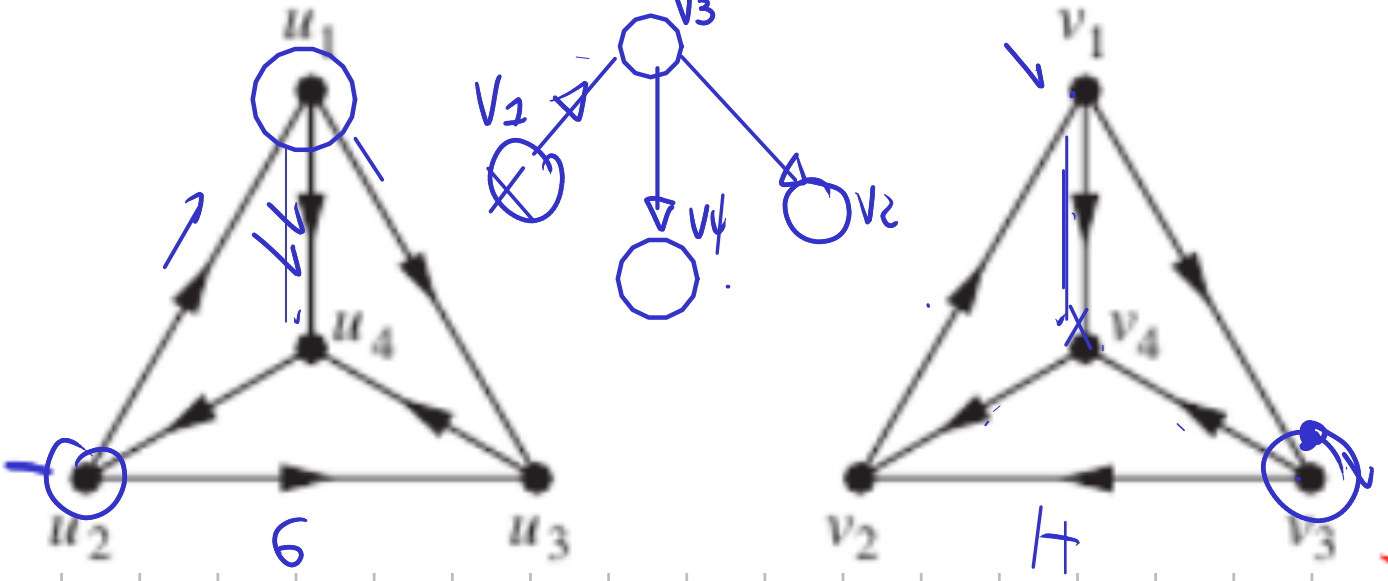
Determine si los siguientes par de grafos son isomorfismos, si lo son encuentre una función de correspondencia entre los vértices de cada uno de los grafos.







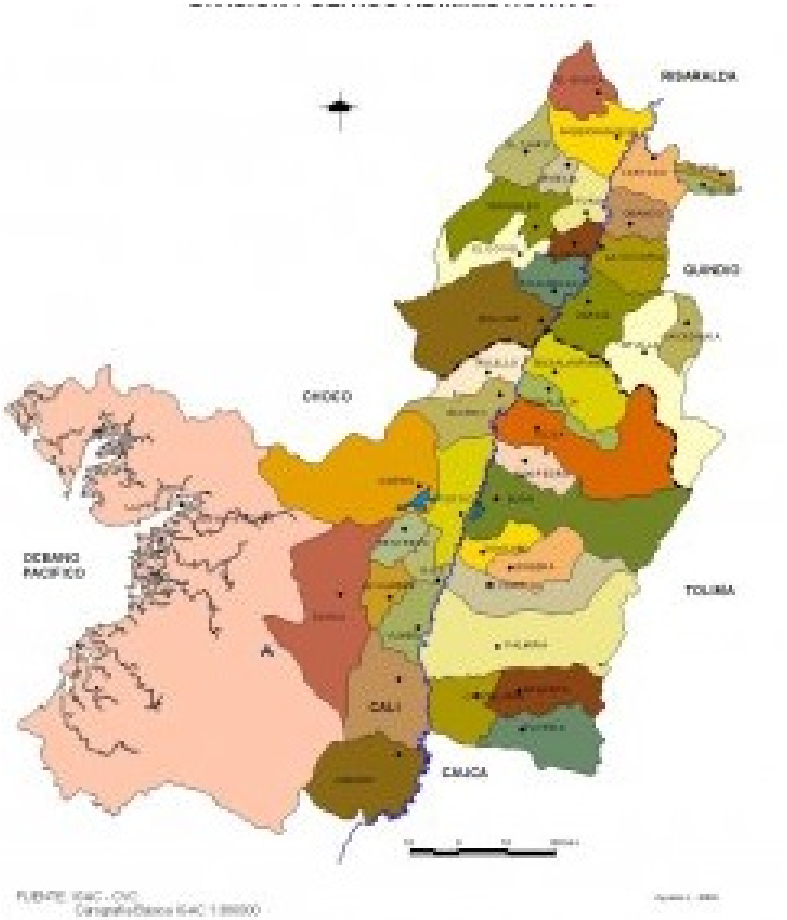



u_1
u_2
u_3
u_4
v_1
v_2
v_3
v_4

Contenido

Coloreado de grafos

Dado un mapa,¿ cual es el **menor numero** de colores que deben utilizarse para colorearlo, de modo que dos regiones adyacentes no tengan nunca el mismo color ?.



Coloreado de grafos

El **grafo dual** de un mapa corresponde a su representación como un grafo, en la cual:

- Cada región del mapa es un vértice
- Si existe una frontera entre cada par de regiones (vértices), existirá una arista entre ellos.

Dos regiones que se tocan en un solo punto no se consideran adyacentes

Coloreado de grafos

Ejemplo de mapa y grafo dual

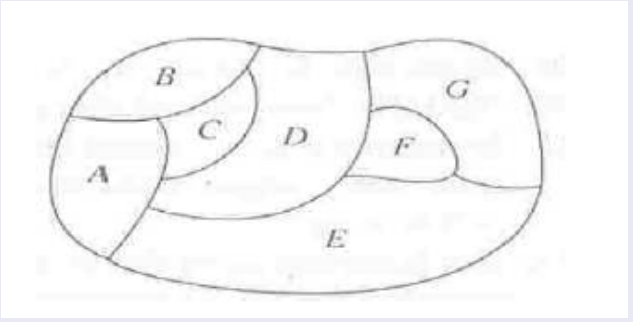


Figure: Mapa

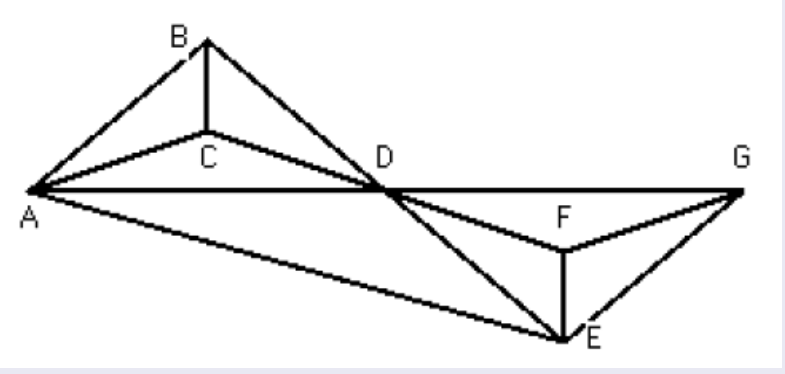


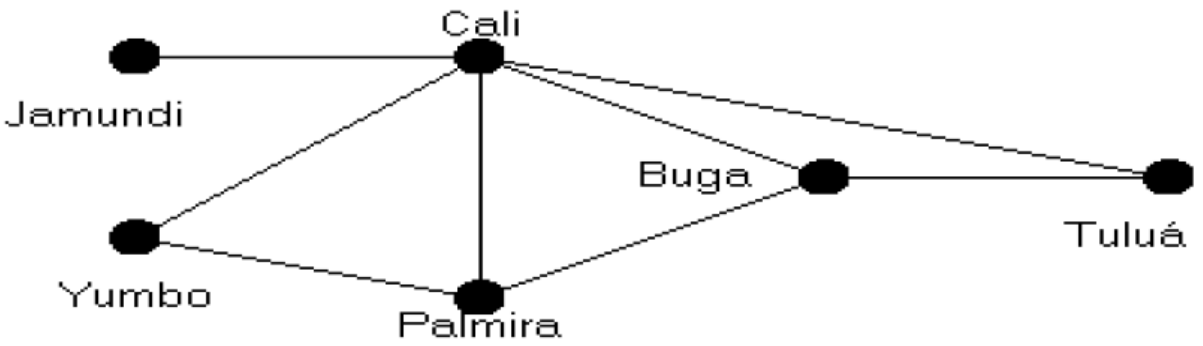
Figure: Grafo dual

Coloración de un grafo

Una coloración de un grafo simple consiste en asignarle un color a cada vértice del grafo de manera que a cada dos vértices adyacentes se les asignan colores distintos.

Número cromático

El número cromático de un grafo es el número mínimo de colores que se requieren para su coloración.



¿Cual es el número cromático de este grafo?

Teorema de los cuatro colores

El número cromático de un grafo plano es menor o igual que 4.

Número cromático

El número cromático de un grafo es el número mínimo de colores que se requieren para su coloración.
Propuesto inicialmente en 1850 y finalmente demostrado por los matemáticos estadounidenses Kenneth Appel y Wolfgang Haken en 1976.

Demostración

Si el teorema era falso, debería existir un contraejemplo, en una lista de aproximadamente 2000 candidatos. Empleando mas de 1000 horas de tiempo de calculo de un ordenador, no encontraron dicho contraejemplo.
Haken en 1976.

Indicar los números cromáticos de:

1. K_n: Grafo completo. R/ n
2. $K_{m.n}$ Grafo bipartito completo. R/ 2
3. C_n Ciclo: R/ 2 si n es par, 3 si n es impar.

Problema computacionalmente difícil

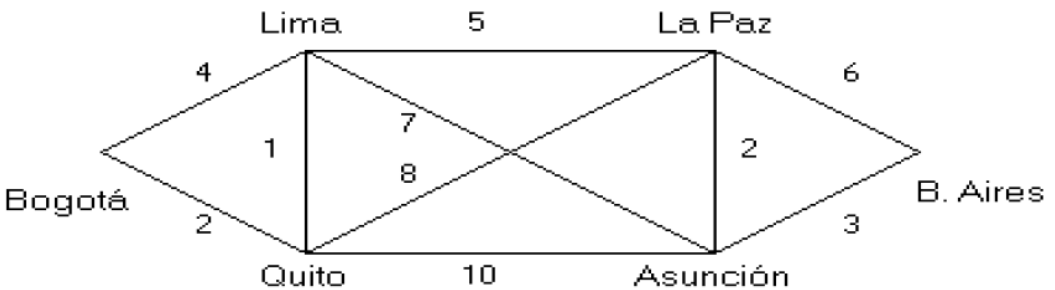
Los mejores algoritmos conocidos para calcular el número cromático de un grafo tienen **complejidad exponencial** (en base al número de vértices)

Aplicaciones

1. **Programación de asignaturas:** Programar las asignaturas sin que ningún estudiante tenga dos asignaturas al mismo tiempo.
   Idea: Los vértices son asignaturas y existe una arista entre un par de vértices, si hay un estudiante matriculado en ambas.
2. **Asignación de frecuencias:** Evitar interferencias en sistemas de comunicaciones móviles.
   Idea: Los vértices son operadores de comunicaciones y hay un par de vértices entre ellos si prestan el mismo servicio

Grafo ponderado

1. A los grafos se les puede asignar un peso a sus aristas
2. A estos grafos se les conoce como **grafos ponderados**
3. La longitud de un camino de un grafo ponderado corresponde a la suma de los pesos en ese camino



Algoritmo de Dijkstra

1. Algoritmo descubierto por el matemático holandés Edsger Dijkstra en 1959.
2. Describe como solucionar el problema de camino de longitud mínima desde a hasta z en grafos ponderados no dirigidos, donde todos los pesos son positivos.
3. Es fácil adaptarlo para solucionar el problema de camino de longitud mínima en grafos dirigidos.

Procedimiento *Dijkstra* (G: grafo ponderado conexo
y todos los pesos positivos)

Para $i = 1$ **Hasta** n

$L(v_i) = \infty$

$L(a) = 0$

$S = \emptyset$

Mientras $z \notin S$

Inicio

u = vertice con $L(u)$ minima entre los vertices
que no estan en S

$S = S \bigcup \{u\}$

Para todos los vertices v que no estan en S

Inicio

Si $L(u) + w(u, v) < L(v)$

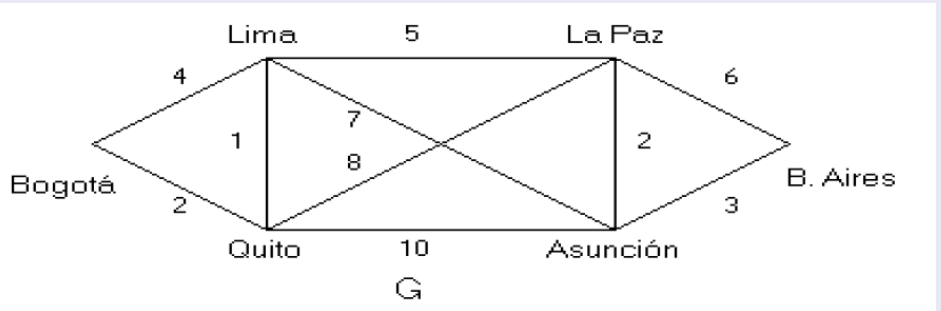
$L(v) = L(u) + w(u, v)$

Fin

Fin (L(z)= longitud del camino mas corto entre a y z)

Algoritmo de Dijkstra

Aplica el algoritmo de Dijkstra para encontrar el camino más corto entre Bogotá y Buenos Aires:



Algoritmo de Dijkstra

Bogotá	Lima	Quito	Paz	Asunción	B. Aires
0	∞	∞	∞	∞	∞
0	4^{Bog}	2^{Bog}	∞	∞	∞
0	3^{Quito}	2^{Bog}	10^{Quito}	12^{Quito}	∞
0	3^{Quito}	2^{Bog}	8^{Lima}	10^{Lima}	14^{Paz}
0	3^{Quito}	2^{Bog}	8^{Lima}	10^{Lima}	13^{Asunc}

El camino más corto está dado por:
$Bogota \rightarrow Quito \rightarrow Lima \rightarrow Asunc \rightarrow B.Aires$ y tiene longitud 13

Teoremas

1. El algoritmo de Dijkstra determina la longitud del **camino más corto** entre dos vértices, en un **grafo ponderado simple, conexo y no dirigido**
2. El algoritmo tiene complejidad $O(v^2 + e)$ donde v es el número de vértices y e de aristas

Introducción

Existen dos formas de recorrer un grafo. Búsqueda por amplitud (**BFS**) y búsqueda por profundidad (**DFS**). **BFS** se puede utilizar para hallar la distancia más corta entre algún nodo inicial y los nodos restantes de un grafo. La distancia más corta es el mínimo número de aristas que hay que recorrer entre un par vértices.

Búsqueda en amplitud

Introducción

La idea de esta búsqueda es iniciar un nodo concreto del grafo e ir examinando todos los nodos adyacentes a este, repitiendo este proceso hasta encontrar el nodo deseado. Para mejorar esta búsqueda se pueden marcar como visitados los nodos ya recorridos, para evitar tener que recorrerlos de nuevo.

Búsqueda en amplitud

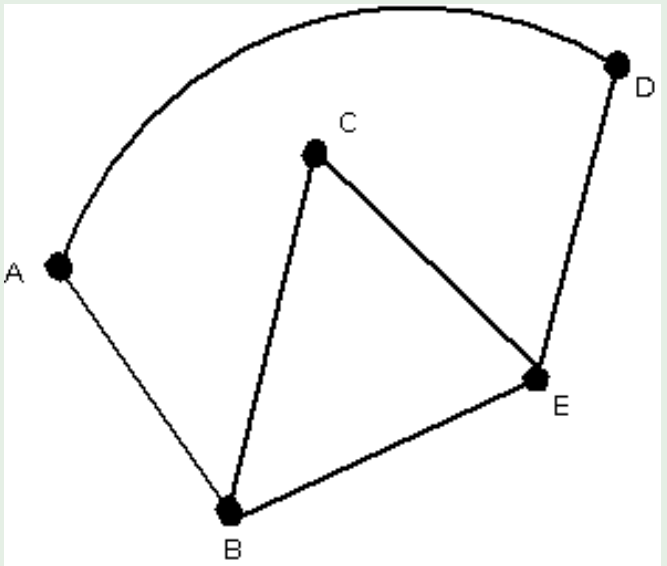
Introducción

El enfoque general de **BFS** es:

1. Inicie en un nodo s
2. Observe todas las aristas incidentes
3. Para a todos los nodos adyacentes w repitiéndose el proceso
4. Almacene los caminos encontrados desde nodo inicial s al resto de nodos

Ejemplo

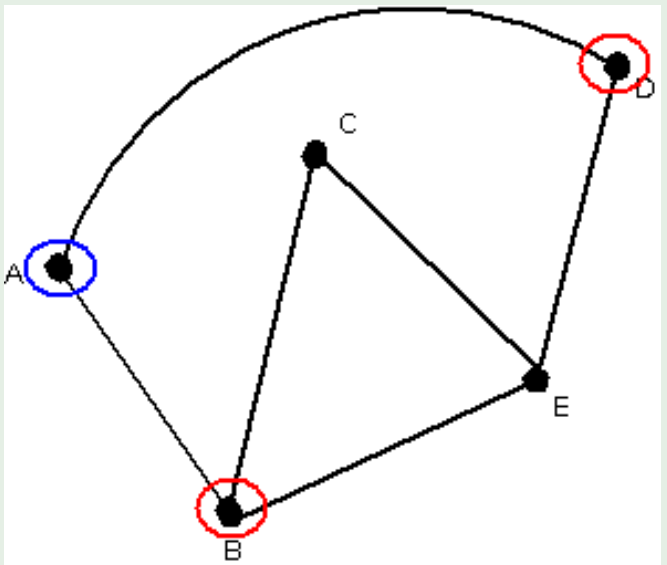
Tomemos como ejemplo el siguiente grafo:



Vamos a mirar cual es la distancia más corta desde A a cualquier nodo.

Ejemplo

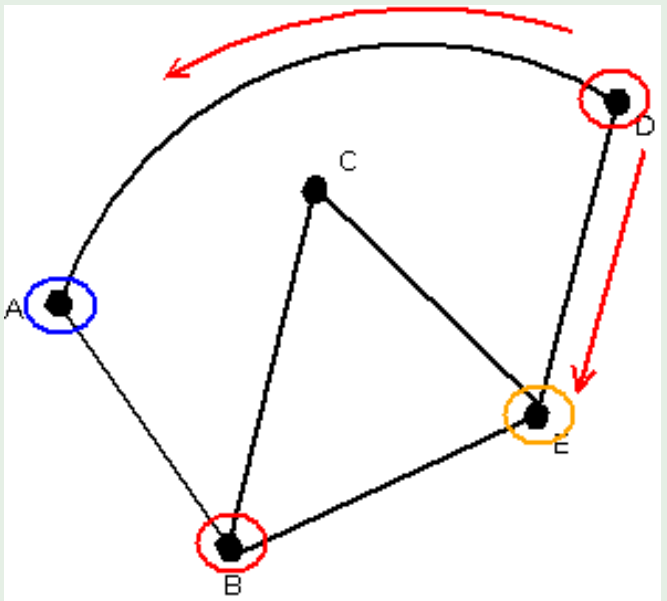
Empezamos por A y miramos los nodos que son vecinos.



Los caminos encontrados son: $\{A, D\}$, $\{A, B\}$.

Ejemplo

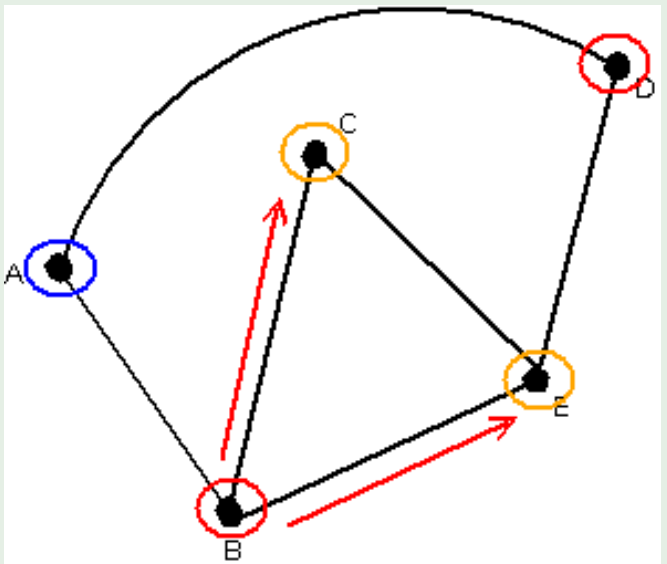
Ahora el algoritmo se repite desde D, evitamos recorrer de nuevo A (ya que lo recorrimos)



Los caminos encontrados son: $\{A, D\}$, $\{A, B\}$, $\{A, D, E\}$.

Ejemplo

Ahora el algoritmo se repite desde B, evitamos recorrer de nuevo A (ya que lo recorrimos)



Los caminos encontrados son: $\{A, D\}$, $\{A, B\}, \{A, D, E\}, \{A, B, C\}, \{A, B, E\}$.

Ejemplo

- Si tomamos los caminos desde E, se debe tomar en cuenta que A, B y D ya ha sido visitado, por lo que obtenemos: $\{A, D\}$, $\{A, B\}, \{A, D, E\}, \{A, B, C\}, \{A, B, E\}, \{A, D, E, C\}, \{A, B, E, C\}$. De los caminos A hasta C encontramos ya uno de distancia 3, por lo que descartamos los de distancia 4, quedando los caminos así. $\{A, D\}$, $\{A, B\}, \{A, D, E\}, \{A, B, C\}, \{A, B, E\}$
- Finalmente si iniciamos en C, tanto B como E están marcados como visitados, por lo que, no es posible recorrer más.
- El algoritmo arroja que los caminos más cortos desde A a cualquier nodo son: $\{A, D\}$, $\{A, B\}, \{A, D, E\}, \{A, B, C\}, \{A, B, E\}$

Búsqueda en amplitud

Procedimiento

El algoritmo de **BFS** lo podemos implementar utilizando una cola:

1. Se marcan todos los nodos como no visitados
2. Se marca y se visita s (nodo inicial)
3. Se pone s en la cola
4. Mientras la cola no esté vacía:
    1. Se quita el primero de la cola y se toma como nodo actual
    2. Para cada vecino w del nodo actual: Si w no está marcado entonces visitamos el vecino, lo marcamos como visitado y lo agregamos en la cola actual

Búsqueda en amplitud

Para el ejemplo anterior, utilizando una cola M:

- Iniciamos en A y lo agregamos a la cola $M = [A]$
- Agregamos a los vecinos B y D, marcamos A como visitado $M = [B, D]$. Los caminos encontrados son $\{A, D\}$, $\{A, B\}$
- Tomamos B, agregamos a los adyacentes no recorridos y lo sacamos de la cola: $M = [D, E, C]$. Los caminos encontrados son $\{A, D\}$, $\{A, B\}\{A, B, C\}, \{A, B, E\}$
- Seguimos con D, $M = [E, C, E]$. Los caminos encontrados son $\{A, D\}$, $\{A, B\}\{A, B, C\}, \{A, B, E\}, \{A, D, E\}$
- Seguimos con E. $M = [C, E]$. Los caminos encontrados son $\{A, D\}$, $\{A, B\}\{A, B, C\}, \{A, B, E\}, \{A, D, E\}, \{A, B, E, C\}, \{A, D, E, C\}$. Descartamos los caminos de longitud 4 de $A - C$ porque ya tenemos mejores.
- Seguimos con C, pero ya no tenemos nodos que visitar. $M = [E]$.
- Seguimos con E, pero ya no tenemos nodos que visitar. $M = []$.
- Al estar vacía la cola, termina el algoritmo.

Búsqueda en profundidad

Introducción

Suponga que una persona se encuentra en un sistema de cuevas interconectadas y quiere encontrar la salida. Una estrategia es comenzar a explorar la cueva más a la izquierda, hasta encontrar una intersección. En esta seleccionamos la cueva más a la izquierda y a así sucesivamente hasta encontrar la salida. Si en un camino dado situado más a la izquierda no tiene éxito, ya que es un camino sin salida, se devuelve hasta la ultima intersección y toma el camino situado al lado de la ultima elección (camino más a la izquierda). Tambien puede aplicarse a la derecha de forma análoga. El sentido es: izquierda (contra manecillas del reloj)

Aclaración

A nivel computacional los grafos no existen en un espacio (no hay derecha ni izquierda), por lo que siempre se escoge el nodo que es el último al expandir los adyacentes, ya que se utiliza una pila como estructura para este algoritmo.

Universidad del Valle

Introducción

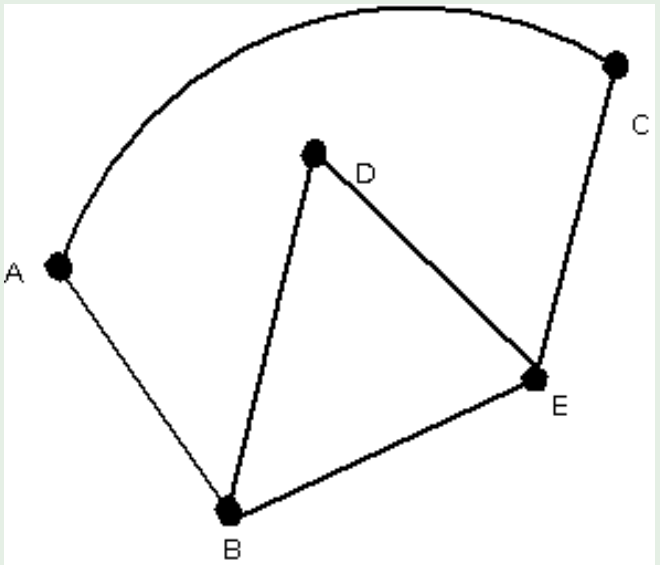
La rutina asociada una búsqueda DFS es:

1. Marcar todos los nodos como no visitados
2. Iniciamos en s y lo marcamos
3. Para cada vecino w de s que no esté marcado volvemos al punto 2, con $s = w$
4. El algoritmo se termina cuando encontramos el nodo buscado

Este algoritmo requiere **que se marquen los nodos como visitados** de lo contrario se caerá en ciclos. Así mismo, **no se garantiza encontrar el camino más corto**

Ejemplo

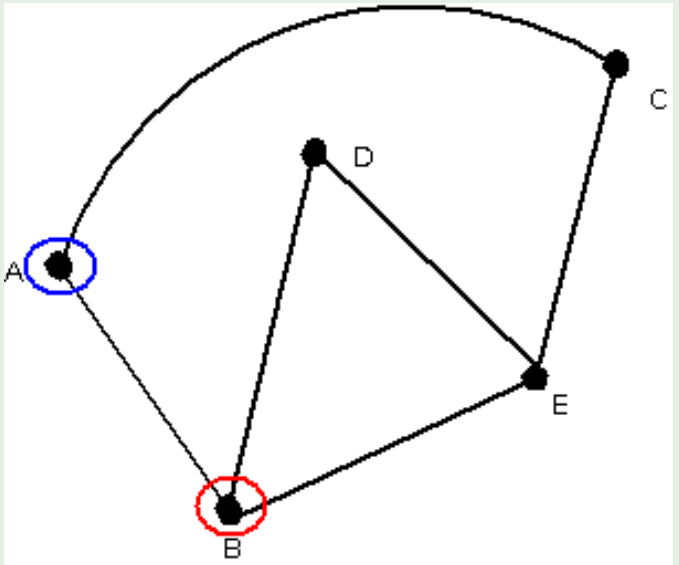
Tomemos como ejemplo el siguiente grafo:



Vamos a encontrar los caminos desde A a C

Ejemplo

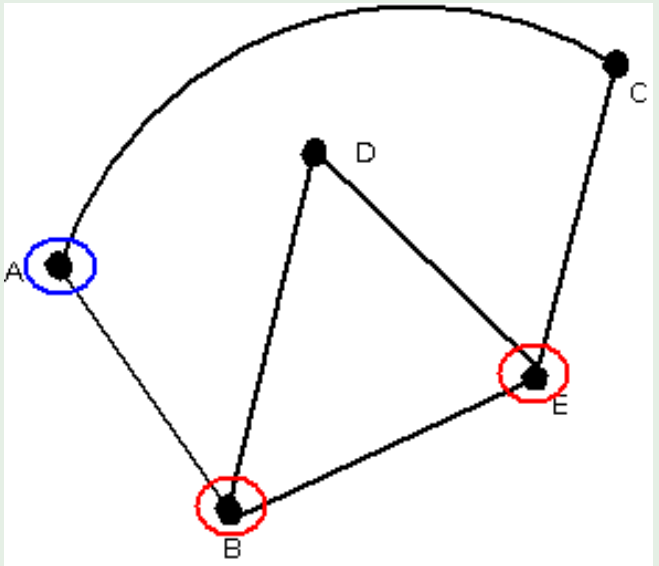
Los nodos adyacentes de A son B y D. Escogemos B ya que está a la izquierda



El camino explorado es $\{A, B\}$

Ejemplo

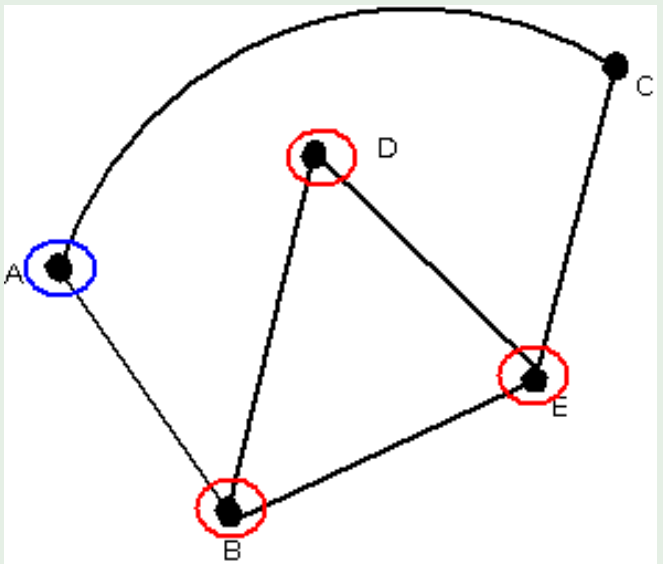
Los nodos adyacentes de B son A, E y C. Descartamos A, ya que está explorado. Escogemos E ya que está a la izquierda



El camino explorado es $\{A, B, E\}$

Ejemplo

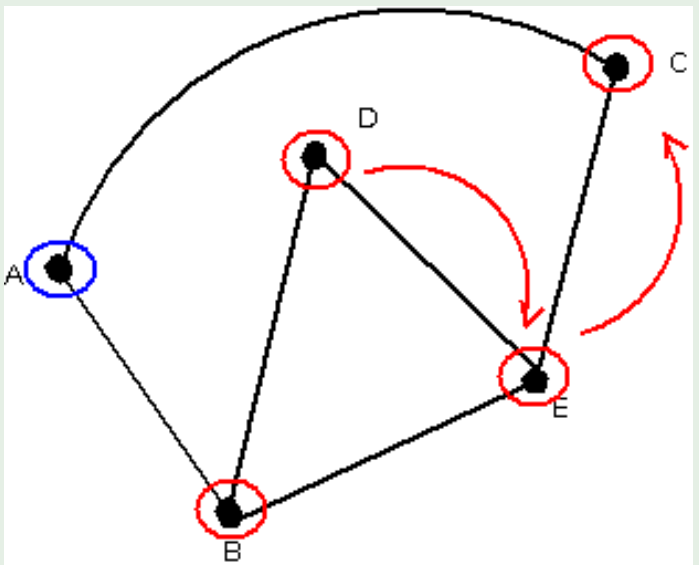
Los nodos adyacentes de E son B, C y D. Descartamos B, ya que está explorado. Escogemos D, ya que está más a la izquierda



El camino explorado es $\{A, B, E, D\}$

Ejemplo

Los nodos adyacentes de D son B y E. Debemos descartar ambos, ya que están marcados. Nos devolvemos a E, y en orden debemos explorar C. Como este es el nodo que buscamos, se termina el algoritmo.



El camino explorado es $\{A, B, E, C\}$. Como se puede observar, **no es el camino más corto (óptimo)**.

Búsqueda en profundidad

Este algoritmo lo podemos implementar con una pila M, con un tope de pila $top(M)$, un nodo inicial s y un nodo destino f

- Iniciamos en s y lo agregamos a la pila $M = [s]$
- Mientras q $top(M) \neq f$:
  1. Colocar en la pila (de izquierda a derecha) los nodos adyacentes al del tope de la pila
  2. Retirar el tope de la pila
  3. Si la pila está vacía no se puede encontrar el nodo

Búsqueda en profundidad

Para el caso anterior, de A hasta C

1. Agregamos A a la pila $M = [A]$
2. Como A no es lo que buscamos, lo borramos y agregamos los nodos adyacentes de izquierda a derecha. 1) $M = [C]$, 2) $M = [B, C]$
3. Como B no es lo que buscamos, borramos y agregamos a D y a E. $M = [E, D, C]$
4. Como E no es lo que buscamos, borramos y agregamos a D y a C. $M = [D, C, D, C]$
5. Como D no es lo que buscamos, borramos y no agregamos ya que los nodos están explorados. $M = [D, C, D, C]$
6. Como C es el tope de la pila, terminamos. $M = [C, D, C]$. El camino encontrado es $\{A, B, E, C\}$

Referencias

Kenneth H. Rosen.
Discrete Mathematics and Its Applications.
McGraw-Hill Higher Education, 7th edition, 2011.
Chapter 10. Graphs.

Universidad del Valle

Gracias

Próximo tema:
Árboles