Matemática Discreta

TEORIA DE GRAFOS

Mercè Claverol
Ester Simó
Marisa Zaragozá

Departamento Matemática Aplicada IV
EPSEVG - UPC

Índice general

Capítulo 1

Grafos: Definiciones Básicas

1.1. Introducción

En este capítulo introducimos el concepto de grafo. Un grafo es una estructura matemática que consta de vértices y aristas que conectan estos vértices. Estas estructuras se utilizan para resolver problemas en muchos campos. Por ejemplo, se pueden utilizar para determinar si un circuito se puede implementar de forma plana, para distinguir entre dos componentes químicos con la misma fórmula molecular pero diferente estructura, etc. Veamos otras posibles aplicaciones de los grafos.

Planificación de proyectos

Una de las primeras aplicaciones por ordenador de los grafos estaba relacionada con la planificación de proyectos. Un grafo con aristas dirigidas es una forma natural de describir, representar y analizar proyectos complejos que consten de muchas actividades relacionadas entre sí.

Sistemas de tráfico

Una herramienta de uso frecuente por parte de quienes hacen diseños urbanísticos y de transportes es la simulación por ordenador de sistemas de tráfico. Los sistemas que se modelan van desde las redes de tráfico nacionales, a las calles de una ciudad, pasando por ciertas zonas urbanas y llegando, incluso, al tráfico existente en cierto puente o cruce de carreteras. Los modelos

se utilizan para poner de manifiesto puntos negros actuales o futuros, y para sugerir y probar cambios propuestos o nuevos sistemas.

Por ejemplo, en una ciudad, el sistema de calles se puede modelar como un grafo en el cual los cruces se representan como vértices, y los segmentos de calle existentes entre cruces son las aristas. Las calles de doble sentido se representan como aristas no dirigidas, mientras que las calles de dirección única se representan mediante aristas dirigidas.

Redes de ordenadores

Una aplicación más reciente de los grafos es el modelado de redes de ordenadores. En la representación de una red mediante un grafo, los ordenadores o dispositivos periféricos están representados por vértices y, las líneas de comunicación por las aristas. Los grafos son una herramienta importante que nos permitirá modelar estas redes con el objetivo de, por ejemplo, mejorar su fiabilidad, obtener redes más eficientes, etc.

1.2. Definiciones básicas

A continuación introduciremos parte de la terminología básica de teoría de grafos.

1.2.1. Grafos Simples

Un *grafo simple* G es una estructura matemática que consta de un par ordenado de conjuntos (V, E), siendo $V \neq \emptyset$. Los elementos de V se llaman *vértices* y los elementos de E se llaman *aristas*. Notemos que en un grafo simple, una arista es un par $\{x, y\}$ no ordenado de vértices diferentes.

Un ejemplo de un grafo $G = (V, E)$ viene dado por los conjuntos

$$V = \{1, 2, 3, 4\}$$
$$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$$

Llamaremos *orden* al cardinal del *conjunto de vértices* V y *tamaño* al cardinal del *conjunto de aristas* E. Por lo tanto, G es un grafo de orden 4 y tamaño 5.

Ni este ejemplo, ni la propia definición de grafo nos dice gran cosa, pero todo cambia si mostramos la *representación gráfica* de un grafo.

Gráficamente los vértices se representan por puntos y unimos dos puntos por una línea si el par correspondiente de vértices es una arista.

En la Figura 1.1 tenemos una representación gráfica del grafo dado en el ejemplo anterior.

Figura 1.1: *Representación gráfica de un grafo*

Acostumbraremos a usar la notación xy como una abreviatura de la arista $\{x, y\}$, aunque no debemos olvidar que una arista es un par no ordenado de vértices, de forma que xy e yx representan lo mismo.

La arista xy se dice que es *incidente* a los vértices x e y y, los vértices x e y se dice que son *adyacentes* o *vecinos*. La arista xy se dice que *conecta* x a y y, los vértices x e y se dice que son *extremos* de la arista xy.

1.2.2. Grafos no tan Simples

Veamos como hay muchas variantes de la definición de grafo, que difieren con respecto al tipo y el número de aristas que pueden conectar un par de vértices.

Multigrafos

Un *multigrafo* $G = (V, E)$ está formado por un conjunto V de vértices, un conjunto E de aristas, y una función f de E a $\{uv \,/\, u, v \in V,\, u \neq v\}$. Diremos que las aristas e_1 y e_2 son *múltiples* si $f(e_1) = f(e_2)$.

El grafo representado en la Figura 1.2(a) tiene dos aristas paralelas que conectan los vértices 1 y 2, y tres aristas paralelas que conectan los vértices 1 y 3. Notemos que mientras en un grafo simple como mucho hay una arista

entre cada par de vértices, en un *multigrafo* se permiten múltiples aristas entre el mismo par de vértices.

Pseudografos

Un *pseudografo* $G = (V, E)$ consta de un conjunto V de vértices, un conjunto E de aristas, y una función f de E a $\{uv \,/\, u, v \in V\}$. Diremos que una arista es un *bucle* si $f(e) = uu = \{u\}$ para algún $u \in V$.

Figura 1.2: *Multigrafo y Pseudografo*

La Figura 1.2(b) representa un pseudografo con dos bucles. Notemos que los pseudografos son el tipo más general de grafos no dirigidos ya que ellos pueden contener bucles y aristas múltiples.

Grafos ponderados

En algunas ocasiones resulta útil asociar un número con cada arista de un grafo.

Un *grafo ponderado* $G = (V, E, w)$ está formado por un grafo $G = (V, E)$ y una aplicación

$$\begin{aligned} w: \quad & E \to R^+ \\ & e \to w(e) \end{aligned}$$

que asocia un número, llamado *peso*, con cada arista del grafo.

La Figura 1.3 representa un grafo ponderado de orden 5.

Figura 1.3: *Grafo con pesos en sus aristas*

Grafos dirigidos

Un *grafo dirigido* o *digrafo* $D = (V, A)$ consta de un conjunto de vértices V y un conjunto de arcos A que son pares ordenados de elementos de V.

En un *grafo dirigido* los arcos están representados por flechas, las cuales indican la dirección de los mismos. En la Figura 1.4 tenemos una representación gráfica del digrafo $D = (V, A)$, donde

$$V = \{x, y, v, w\}$$
$$A = \{(v, w), (w, y), (y, x), (x, v)\}$$

Figura 1.4: *Digrafo de orden 4*

Notemos que la diferencia entre un grafo y un digrafo está en que en un grafo una arista es un par no ordenado de vértices, mientras que en un digrafo una arco (arista dirigida) es un par ordenado de vértices.

1.3. Clases Especiales de Grafos

Algunos grafos han adquirido nombres más o menos estándards porque aparecen de forma frecuente en muchas aplicaciones.

Grafo Completo

Si n es un entero positivo, el *grafo completo* de n vértices, denotado K_n, es un grafo simple de orden n, en el cual cada par de vértices diferentes está unido por una arista.

En la Figura 1.5 tenemos representados los grafos completos de órdenes 2, 3 y 4.

Figura 1.5: *Grafos completos K_2, K_3 y K_4*

Grafo Nulo

Si n es un entero positivo, el *grafo nulo* de orden n, denotado N_n, es el grafo que tiene por conjunto de vértices $\{v_1, v_2, \cdots, v_n\}$ y ninguna arista.

En la Figura 1.6 tenemos representados los grafos nulos de órdenes 2, 3 y 4. En cierto modo, los grafos nulos son opuestos a los grafos completos.

Ciclo

Si n es un entero mayor o igual que 3, el *ciclo* de orden n , denotado C_n, es un grafo con conjunto de vértices $V = \{v_1, v_2, \cdots, v_n\}$ y conjunto de aristas $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \cdots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.

Los ciclos, C_n, de ordenes 3, 4 y 5 están representados en la Figura 1.7.

Figura 1.6: *Grafos nulos* N_2, N_3 *y* N_4

Figura 1.7: *Grafos ciclo* C_3, C_4 *y* C_5

Grafo Rueda

Obtenemos el *grafo rueda* de orden $n+1$, denotado W_n, cuando añadimos un vértice adicional al ciclo C_n, para $n \geq 3$, y conectamos este nuevo vértice a cada uno de los vértices del ciclo C_n, mediante n nuevas aristas.

Los grafos rueda, W_n, de ordenes 3, 4 y 5 están representados en la Figura 1.8.

Hipercubo

El *hipercubo* de orden n, denotado Q_n, es el grafo que tiene 2^n vértices representando las 2^n cadenas binarias de longitud n, de modo que dos vértices son adyacentes si y sólo si las cadenas binarias que ellos representan difieren en exactamente una posición.

Los hipercubos, Q_n, de órdenes 2, 4 y 8 están representados en la Figura 1.9.

Figura 1.8: *Grafos rueda W_3, W_4 y W_5*

Figura 1.9: *Hipercubos Q_1, Q_2 y Q_3*

Grafo Bipartito

Un grafo simple se dice que es *bipartito* si su conjunto de vértices V se puede particionar en dos subconjuntos disjuntos no vacíos

$$V_1 = \{v_1, v_2, \cdots, v_{n_1}\} \text{ y } V_2 = \{w_1, w_2, \cdots, w_{n_2}\},$$

tal que dada una arista cualquiera del grafo tendrá un vértice extremo en V_1 y el otro en V_2.

La Figura 1.10 representa el hipercubo Q_3 el cual es un ejemplo de grafo bipartito.

Grafo Bipartito Completo

Un *grafo bipartito completo*, denotado K_{n_1,n_2}, es un grafo bipartito en el que cada vértice de $V_1 = \{v_1, v_2, \cdots, v_{n_1}\}$ está unido a cada uno de los

Figura 1.10: Q_3 *es un grafo bipartito*

vértices de $V_2 = \{w_1, w_2, \cdots, w_{n_2}\}$.

El hipercubo Q_2 es un ejemplo de grafo bipartito completo. En la Figura 1.11 lo hemos representado coloreando sus vértices de diferente color con el objeto de diferenciar a los dos subconjuntos de la partición de vértices.

Figura 1.11: Q_2 *es el grafo bipartito completo* $K_{2,2}$

1.3.1. Aplicación

Para finalizar la sección mostraremos cómo algunas de estas familias de grafos son utilizadas en la práctica para modelar redes de comunicación de datos.

Varios ordenadores en un edificio, así como aparatos periféricos tales como impresoras y plotters, pueden ser conectados utilizando una *red de área local*. Algunas de estas redes están basadas en una *topología estrella*, donde todos los aparatos que componen la red están conectados a un dispositivo de control central. Una red de área local basada en esta topología se puede representar

utilizando un grafo bipartito completo $K_{1,n}$ como se muestra en la Figura 1.12(a). Los mensajes son enviados de dispositivo a dispositivo a través del dispositivo de control central.

Otras redes de área local están basadas en la *topología anillo*, donde cada dispositivo está conectado a exactamente otros dos. Las redes de área local con topología anillo están modeladas utilizando ciclos de orden n, C_n, como se muestra en la Figura 1.12(b). Los mensajes se envían de aparato a aparato alrededor del ciclo hasta que el receptor deseado de un mensaje sea alcanzado.

Finalmente, algunas redes de área local utilizan un híbrido de estas dos topologías. Los mensajes se pueden enviar alrededor del anillo, o a través de un dispositivo central. Esta redundancia hace que la red sea más fiable. Las redes de área local con esta redundancia se pueden modelar utilizando grafos rueda, W_n, como se muestran en la Figura 1.12(c).

Figura 1.12: *Topologías estrella, anillo e híbrida para redes de área local*

1.4. Representando Grafos

Hemos definido un grafo en términos de un conjunto de vértices y un conjunto de aristas. En esta sección veremos que existen otras formas de representar grafos, que nos proporcionan la información relevante de éstos, y que por su estructura resultan más adecuadas si queremos que sean procesadas por un ordenador.

Matriz de adyacencia

Supongamos que $G = (V, E)$ es un grafo simple de orden n, con conjunto de vértices $V = \{v_1, v_2, \cdots, v_n\}$. La *matriz de adyacencia* de G, denotada $A(G) = [a_{ij}]$, es una matriz cuadrada de orden $n \times n$ definida como:

$$a_{ij} = \begin{cases} 1 & \text{si } \{v_i, v_j\} \text{ es una arista de } G, \\ 0 & \text{en caso contrario.} \end{cases}$$

La matriz de adyacencia del grafo de orden 4 representado en la Figura 1.13 viene dada por

$$A[G] = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Figura 1.13: *Un grafo simple de orden 4*

Notemos que la matriz de adyacencia de un grafo simple es simétrica respecto a su diagonal principal, es decir, $a_{ij} = a_{ji}$, y dado que un grafo simple no tiene bucles, los elementos de la diagonal principal $a_{i,i}$, para $i = 1, 2, \cdots, n$, son cero. Además el número de unos de cada una de sus filas coincide con el grado del vértice correspondiente.

Matriz de incidencia

Otra forma común de representar grafos es mediante las matrices de incidencia. Mientras la matriz de adyacencia involucra las adyacencias de vértices, la matriz de incidencia involucra la incidencia de vértices y aristas.

Supongamos que $G = (V, E)$ es un grafo simple con conjunto de vértices $V = \{v_1, v_2, \cdots, v_n\}$ y conjunto de aristas $E = \{e_1, e_2, \cdots, e_m\}$. La *matriz de incidencia* de G, denotada $I(G) = [b_{ij}]$, es una matriz de orden $n \times m$ definida como:

$$m_{ij} = \begin{cases} 1 & \text{cuando la arista } e_j \text{ es incidente con el vértice } v_i, \\ 0 & \text{en caso contrario.} \end{cases}$$

La matriz de incidencia

$$I[G] = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

representa el grafo de la Figura 1.14.

Figura 1.14: *Un grafo simple de orden 4*

Notemos que el número de unos de cada una de las filas de la matriz de incidencia de un grafo es igual al grado del vértice correspondiente, mientras que el número de unos de cada una de sus columnas es 2.

1.5. Subgrafos

Un grafo $H = (V(H), E(H))$ es un *subgrafo* de un grafo $G = (V(G), E(G))$ si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$.

La Figura 1.15 nos muestra un grafo G y dos de sus subgrafos, H_1 y H_2.

A continuación presentaremos algunos tipos especiales de subgrafos:

Figura 1.15: *Un grafo simple de orden 5 y dos de sus subgrafos*

Dado un grafo $G = (V(G), E(G))$ y $H = (V(H), E(H))$ un subgrafo de G, diremos que H es un *subgrafo generador* de G si $V(H) = V(G)$.

En la Figura 1.16 tenemos representado un grafo G y dos de sus subgrafos generadores. Éstos son dos de los $2^4 = 16$ posibles subgrafos generadores que existen del grafo G.

Figura 1.16: *Un grafo simple de orden 5 y dos de sus subgrafos generadores*

Sea $G = (V, E)$ un grafo y sea U un subconjunto no vacío de vértices del grafo. Llamaremos *subgrafo inducido* por U, y lo denotaremos $\langle U \rangle$, a aquel subgrafo de G cuyo conjunto de vértices es U y que contiene todas las aristas del grafo G de la forma xy con $x, y \in U$.

Para los subgrafos de la Figura 1.15, vemos que H_2 es un subgrafo inducido de G pero el subgrafo H_1 no lo es por no aparecer las aristas $\{1, 2\}$ ni la $\{1, 3\}$.

Otros tipos especiales de subgrafos se obtienen al eliminar del grafo cierto

vértice o arista. Hemos formalizado esta idea en las siguientes definiciones.

Sea v un vértice de un grafo $G = (V(G), E(G))$. El subgrafo $G - v$ de G es aquel grafo que tiene por conjunto de vértices $V(G) - \{v\}$ y por conjunto de aristas $E(G - v)$ a todas las aristas del grafo G excepto las incidentes con el vértice v. Por lo tanto, $G - v$ es el subgrafo de G inducido por $V(G) - \{v\}$.

En la Figura 1.17 podemos comprobar gráficamente cuál es el resultado de eliminar el vértice v del grafo G.

Figura 1.17: *Grafos G y $G - v$*

De forma análoga, si e es una arista de un grafo $G = (V, E)$, el subgrafo $G - e$ de G es aquel grafo que tiene por conjunto de aristas $E(G - e) = E(G) - \{e\}$ y por conjunto de vértices $V(G - e) = V(G)$. Por lo tanto, $G - e$ es un subgrafo generador de G.

En la Figura 2.4 podemos comprobar gráficamente cuál es el resultado de eliminar la arista uv del grafo G.

La idea de subgrafo nos ofrece una forma sencilla de desarrollar el concepto de complementario de un grafo.

Sea G un grafo simple de orden n. El *complementario* de G, que denotamos por $\overline{G}$, es el subgrafo de K_n formado por los n vértices de G y todas las aristas del grafo completo K_n que no están en G.

Así, si dos vértices están conectados en G, no lo están en $\overline{G}$; y viceversa. En la Figura 1.19 tenemos representados un grafo de orden 4 y su grafo complementario.

Figura 1.18: *Grafos G y $G - uv$*

Figura 1.19: *Grafos K_4, G y $\overline{G}$*

1.6. Propiedades

1.6.1. Grado de un vértice

El *grado* de un vértice v de un grafo G no dirigido, que denotaremos $deg(v)$, es el número de aristas en G que son incidentes con él. En este caso, un bucle en un vértice v se considera como dos aristas que inciden en v.

Por ejemplo, el grado de los vértices del grafo representado en la Figura 1.20 es

$$d(x) = 2,\ d(y) = 4,\ d(u) = 3,\ d(v) = 0,\ d(w) = 4 \text{ y } d(t) = 5$$

¿Qué conseguimos cuando sumamos todos los grados de los vértices de un grafo $G = (V, E)$? Cada arista del grafo contribuye en dos unidades a la suma

Figura 1.20: *El grafo no dirigido G*

de los grados de los vértices ya que cada arista es incidente con exactamente dos (posiblemente iguales) vértices del grafo. Esto significa que la suma de los grados de los vértices es dos veces el número de aristas del grafo.

Teorema 1.1 *Sea $G = (V, E)$ un grafo no dirigido de tamaño m, entonces*

$$\sum_{v \in V} d(v) = 2m$$

Este teorema proporciona un indicio sobre el número de vértices de grado impar que pueden existir en un grafo.

Corolario 1.2 *Un grafo no dirigido tiene un número par de vértices de grado impar*

Dem.

Sean V_1 y V_2 el conjunto de vértices de grado par y el conjunto de vértices de grado impar, respctivamente, en un grafo $G = (V, E)$. Entonces

$$2m = \sum_{v \in V} d(v) = \sum_{v \in V_1} d(v) + \sum_{v \in V_2} d(v)$$

Dado que $d(v)$ es par para $v \in V_1$, el primer término del lado derecho de la igualdad es par. Además la suma de estos dos términos es par, ya que vale $2m$. De aquí, el segundo término de la suma también será par. Y dado que todos los términos de este segundo sumatorio son impares, para que la suma sea par deberemos sumar un número par de términos. Así, hay un número par de vértices de grado impar. □

En el siguiente ejemplo aplicamos el Teorema 1.1

Ejemplo 1.1

Diremos que un grafo es *regular* si todos sus vértices tienen el mismo grado. Si $deg(v) = k$ para todos los vértices del grafo, entonces el grafo se dice que es $k - regular$. ¿Es posible tener un grafo 4-regular con 10 aristas?

Por el Teorema 1.1, $2|E| = 20 = 4|V|$, por lo que tenemos 5 vértices de grado 4. La Figura 1.21 proporciona un ejemplo de un grafo que satisface lo solicitado.

Figura 1.21: *El grafo completo* K_5

1.6.2. Isomorfismo de grafos

Es evidente que lo importante de un grafo no son los nombres de los vértices ni su representación gráfica, o cualquier otra representación. La propiedad característica de un grafo es la manera en que los vértices están unidos por las aristas. Esto motiva la siguiente definición.

Los grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son *isomorfos* si existe una aplicación biyectiva $f : V_1 \to V_2$ que conserva las adyacencias, es decir,

$$\{v, w\} \in E_1 \text{ si y sólo si } \{f(v), f(w)\} \in E_2$$

A tal función se le llamará *isomorfismo de grafos*.

Los grafos representados en la Figura 1.22 son isomorfos. Notemos que, cuando dos grafos son isomorfos pensamos en ellos como si fueran el mismo grafo. De hecho, estos dos grafos son dos posibles representaciones gráficas del grafo hipercubo Q_3.

Figura 1.22: *Ejemplo de grafos isomorfos*

Por lo general, es difícil determinar si dos grafos simples son isomorfos, dado que hay $n!$ aplicaciones biyectivas entre los conjuntos de vértices de dos grafos de orden n, e ir comprobando todas las biyecciones hasta encontrar una que conserve las adyacencias no es viable si el tamaño de n es grande.

Sin embargo, podemos demostrar que dos grafos simples no son isomorfos comprobando que ellos no comparten una propiedad que los grafos isomorfos simples deberían tener en común. Por ejemplo:

1. Dos grafos isomorfos deben tener el mismo número de vértices, ya que hay una aplicación biyectiva entre los conjuntos de vértices de los grafos.

2. Dos grafos isomorfos deben tener el mismo número de aristas, porque la aplicación biyectiva entre vértices establece una biyección entre aristas.

3. Los grados de los vértices en grafos simples isomorfos deben ser los mismos. Es decir, un vértice v de grado d en G_1 se corresponde con un vértice $f(v)$ de grado d en G_2, ya que un vértice w en G_1 es adyacente a v sí y sólo si $f(v)$ y $f(w)$ son adyacentes en G_2.

Si cualquiera de las cantidades anteriores difieren en dos grafos simples, éstos no pueden ser isomorfos. Sin embargo, notemos que si estas cantidades coinciden para los dos grafos, ello no necesariamente significa que los grafos sean isomorfos.

Ejemplo 1.2

Los grafos de la Figura 1.23 tienen seis vértices y nueve aristas cada uno. Por tanto, es razonable preguntarse si son isomorfos.

Notemos que la secuencia de grados en el grafo G_1 es $\{2, 2, 3, 3, 4, \}$, mientras que en el grafo G_2 tenemos la secuencia $\{2, 2, 2, 4, 4, 4, \}$, con lo cual es imposible establecer una biyección entre los conjuntos de vértices de los grafos que conserve las adyacencias.

Figura 1.23: *Grafos no isomorfos*

Capítulo 2

Grafos: Caminos y Conexión

Muchos problemas, como por ejemplo, determinar si un mensaje se puede enviar entre dos ordenadores utilizando enlaces intermedios o planificar las rutas del camión de recogida de basura en una ciudad, se pueden resolver modelando la red de ordenadores o la ciudad mediante un grafo y viajando a lo largo de las aristas del grafo. Empezaremos el tema presentando la terminología básica que trata estos conceptos.

2.1. Caminos

Sean x e y dos vértices (no necesariamente distintos) de un grafo $G = (V, E)$ no dirigido. Un $x - y$ *recorrido* es una secuencia alternada de vértices y aristas

$$x = x_0, e_1, x_1, e_2, x_2, \cdots, e_{n-1}, x_{n-1}, e_n, x_n = y$$

que comienza en el vértice x (*vértice inicial* del recorrido) y termina en el vértice y (*vértice final* del recorrido) y que contiene las n aristas

$$e_i = \{x_{i-1}, x_i\}, \text{ donde } 1 \leq i \leq n.$$

Diremos que un $x - y$ recorrido es *cerrado* si $x = y$ y $n > 1$.

La *longitud* de un recorrido es n, el número de aristas que hay en el mismo. Notemos que un recorrido puede repertir vértices y aristas, por lo tanto al calcular la longitud de un recorrido que repite aristas, éstas se cuentan tantas veces como son atravesadas por el mismo.

Ejemplo 2.1

Para el grafo de la Figura 2.1 tenemos, por ejemplo, los siguientes cuatro recorridos. Notemos que también podemos enumerar un recorrido citando sólo las aristas o sólo los vértices que intervienen en el mismo.

1. ab, bc, cd, db, bc, cf es un $a - f$ recorrido de longitud 6, en el que se repite la arista bc, y los vértices b y c.

2. a, b, c, f es un $a - f$ recorrido de longitud 3, en el cual no se repiten ni vértices ni aristas.

3. d, b, c, e, d, c es un $d - c$ recorrido de longitud 5, en el cual se repiten los vértices b y c, pero no se repiten aristas.

4. $bd, de, ec, cb, bd, de, ec, cb$ es un $b - b$ recorrido cerrado de longitud 8, en el cual se repiten todos los vértices y todas las aristas que intervienen en el mismo.

Figura 2.1: *Tipos de recorrido*

Examinemos algunos tipos especiales de recorridos. Llamaremos:

- $x - y$ *Cadena* a un recorrido $x - y$ que no repite aristas.
- $x - y$ *Camino* a un recorrido $x - y$ que no repite ningún vértice interno.
- *Circuito* a una $x - x$ cadena cerrada.

- *Ciclo* a un $x - x$ camino cerrado.

Ejemplo 2.2

Para el grafo de la Figura 2.2 tenemos:

1. a, e, j, k, d, e es una $a - e$ cadena de longitud 5.
2. a, e, j, i, h, g es un $a - g$ camino de longitud 5.
3. a, e, f, i, j, e, d, a es un $a - a$ circuito de longitud 7.
4. d, e, j, k, d es un $d - d$ ciclo de longitud 4.

Figura 2.2: *Recorridos especiales*

Y para finalizar la sección introduciremos el concepto de distancia.

Dados x e y vértices de un grafo G, definimos la distancia de x a y en G, y la denotaremos $d(x, y)$, como la longitud del $x - y$ camino más corto. Convenimos en decir que si no existe un $x-y$ camino, entonces $d(x, y) = +\infty$.

Esta distancia es una *métrica* en el conjunto de vértices $V(G)$ ya que satisface las propiedades siguientes:

1. $d(x, y) \geq 0$ y $d(x, y) = 0$ si y sólo si $x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, y) \leq d(x, v) + d(v, y)$ (desigualdad triangular).

Ejemplo 2.3

Dado el grafo G, representado en la Figura 2.3, tenemos que la distancia del vértice 1 a los demás es:

Figura 2.3: *Cálculo de distancias*

$$d(1,2) = 1, \quad d(1,5) = 1, \quad d(1,3) = 1, \quad d(1,4) = 1,$$
$$d(1,6) = 2, \quad d(1,7) = 3, \quad d(1,8) = 3.$$

A partir de la noción de distancia podemos definir un parámetro relevante de un grafo, el diámetro.

El *diámetro* de un grafo $G = (V, E)$, denotado $D(G)$, es la máxima de las distancias entre vértices de G, es decir,

$$D(G) = max\{d(x, y) \ / \ x, y \in V\}$$

Notemos que el diámetro del grafo representado en la Figura 2.3 es $D(G) = 3$, pues no hay ningún par de vértices que estén a una distancia mayor que 3.

2.2. Grafos Conexos

Supongamos que nos hacen la siguiente pregunta: ¿Cuándo una red de área local tiene la propiedad de que cualquier par de ordenadores de la misma

puede compartir información? Para resolverla la formulamos en términos de Teoría de Grafos: Suponiendo que la red está modelada por un grafo, ¿hay siempre un camino entre cualquier par de vértices del grafo?

Grafo conexo

Diremos que un grafo $G = (V, E)$ es *conexo* si y sólo si hay un camino entre cualquier par de vértices diferentes del grafo.

Así, cualquier par de dispositivos de la red se pueden comunicar si el grafo que modela esta red es conexo.

Ejemplo 2.4

Figura 2.4: *Eliminación de una arista*

En la Figura 2.4 podemos ver que mientras G es un grafo conexo, el grafo $G - uv$ no lo es, ya que hay pares de vértices para los que no existe camino que los conecte, por ejemplo los vértices u y v.

Diremos que dos vértices x e y de un grafo $G = (V, E)$ *están conectados* en G si y sólo si existe un $x - y$ camino en G.

Evidentemente, la relación *estar conectados* es de equivalencia en V. Como cualquier relación de equivalencia induce una partición del conjunto de vértices en clases de equivalencia $V_1, V_2, \cdots, V_k$.

A los subgrafos inducidos por estas clases de equivalencia:

$$\langle V_1 \rangle, \langle V_2 \rangle, \cdots, \langle V_k \rangle$$

se les llama *componentes conexas* de G.

Dado un grafo G, denotaremos por $w(G)$ al número de componentes conexas del grafo. Notemos que si G es conexo, $w(G) = 1$. Un grafo con dos componentes conexas está representado en la Figura 2.5.

Figura 2.5: *Grafo no conexo*

2.3. Conectividad

Cuando se diseñan redes de telecomunicación, sistemas de carreteras, y otros sistemas que están interconectados de algún modo, una cuestión a tener en cuenta sobre el grafo que modela el sistema es ¿hasta cuántas aristas o cuántos vértices se pueden eliminar del grafo conexo para que no se desconecte? Por ejemplo, en redes de área local es esencial que la red siga siendo operable a pesar de que algunos enlaces o dispositivos de la red se hayan dañado. Con este objetivo se diseñan las redes de modo que hayan diferentes caminos entre los diferentes dispositivos que componen la red. Frente a tal situación es importante conocer hasta cuántos enlaces (aristas) o cuántos dispositivos (vértices) se pueden estropear en la red sin impedir que falle la comunicación.

Con el objetivo de responder a cuestiones de este tipo, necesitamos estudiar los grafos conexos un poco más.

2.3.1. Aristo-Conectividad

Consideremos los cuatro grafos conexos representados en la Figura 2.6, así como las siguientes situaciones:

- Si eliminamos una de las aristas vw o vx del grafo de la Figura 2.6(a), éste deja de ser conexo, pasando a tener dos componentes conexas.
- El grafo de la Figura 2.6(b) se desconecta si eliminamos la arista vw, pasando a tener dos componentes conexas.
- El grafo de la Figura 2.6(c) no se desconecta si eliminamos una única arista, mientras que si eliminamos dos aristas (por ejemplo las aristas uw y vw) lo desconectamos.
- Análogamente, el grafo de la Figura 2.6(d) sólo se desconecta si eliminamos dos aristas (por ejemplo, las aristas uw y wx).

Figura 2.6: *Grafos y pseudografos*

A partir de estos ejemplos definimos:

Una *arista puente* como aquella arista cuya eliminación desconecta un grafo.

Por tanto, aristas puente son vw o vx del grafo representado en la Figura 2.6 o la arista vw del grafo representado en la Figura 2.6(b). Veamos a continuación un resultado que caracteriza a una arista puente.

Teorema 2.1 *Si e es una arista de un grafo conexo G, tenemos que e es arista puente si y sólo si no se encuentra contenida en ningún ciclo.*

Dem.

1. Supongamos que $e = \{x, y\} \in E$ es una arista puente del grafo G, y veamos que no puede estar contenida en ningún ciclo del grafo G.

   Para ello procederemos por reducción al absurdo y supondremos que está contenida en un ciclo C del grafo G. Por definición de arista puente tenemos: $w(G-e) > w(G)$. Supongamos que u y v son dos vértices del grafo que están conectados en G pero no lo están en $G - e$; entonces, existirá un $u-v$ camino en G que atraviesa la arista puente. Si la arista puente está contenida en un ciclo, en $G - e$ tenemos un $u - v$ camino alternativo, por lo que u y v no están desconectados en $G - e$. Lo cual es una contradicción.

2. Supongamos que $e = \{x, y\}$ es una arista que no se encuentra contenida en ningún ciclo y veamos que se trata de una arista puente.

   Para probarlo procederemos por reducción al absurdo, y supondremos que e no es una arista puente. Entonces $w(G - e) = w(G)$, por lo que los vértices extremos de la arista se encuentran en $G - e$ en la misma componente conexa. Entonces existirá un $x - y$ camino en $G - e$, y si añadimos la arista e tendremos un ciclo en el grafo G, lo cual es una contradicción. □

La *aristo-conectividad* de un grafo conexo G, que denotaremos por $\lambda(G)$, es el menor número de aristas que hay que suprimir de G para que el grafo resultante pase a ser desconexo.

Así los grafos representados en la Figura 2.6(a) y (b) tienen aristo-conectividad 1, mientras que los grafos (c) y (d) tienen aristo-conectividad 2.

2.3.2. Vértice-Conectividad

También podemos pensar la conectividad en términos del menor número de vértices que se necesita eliminar para desconectar el grafo. Recordemos que al eliminar un vértice, debemos eliminar también las aristas incidentes con él.

Consideremos de nuevo los grafos de la Figura 2.6. Notemos que los grafos (a) y (b) se pueden desconectar al eliminar un vértice de dos posibles (o v o w); el grafo (c) también se desconecta al eliminar un único vértice (el vértice w); mientras que el grafo (d) no se desconecta si eliminamos un único

vértice, pero la eliminación de dos vértices no adyacentes (por ejemplo u y x) lo desconecta. Vistos estos ejemplos definimos:

Un *vértice de corte* es un vértice cuya eliminación incrementa el número de componentes conexas del grafo.

Ejemplos de vértices de corte son los vértices v o w de los grafos de la Figura 2.6(a) y (b), o el vértice w del grafo (c). Presentemos a continuación un resultado que caracteriza a un vértice de corte.

Teorema 2.2 *Si v es un vértice de un grafo conexo G, entonces v es un vértice de corte si y sólo si existen dos vértices x e y $(x, y \neq v)$, tal que v está en cualquier $x - y$ camino en G.*

Dem.

1. Veamos que si v es un vértice de corte, entonces existirán dos vértices de modo que v está en cualquier $x - y$ camino.

   Si v es un vértice de corte de G, entonces $G - v$ es desconexo. Supongamos que $w(G - v) = k$, y sean

   $$\langle V_1 \rangle, \langle V_2 \rangle, \cdots, \langle V_k \rangle$$

   las k componentes conexas del grafo $G - v$.

   Tomemos $x \in V_1$ e $y \in V_2 \cup \cdots \cup V_k$. Notemos que en $G - v$ no hay un $x - y$ camino, mientras que en G si que los hay (por ser G conexo). Por lo tanto, v está en todos los $x - y$ caminos en G.

2. Veamos que si existen dos vértices x e y $(x, y \neq v)$ tal que v está en todos los $x - y$ caminos, entonces v es vértice de corte.

   Por hipótesis el grafo G es conexo. Si existen dos vértices x e y $(x \neq y)$ tal que v está en cualquier $x - y$ camino en G, entonces en $G - v$ no hay un $x - y$ camino, por lo que $G - v$ no es conexo. Por lo tanto v es un vértice de corte. □

La *conectividad* (o *vértice-conectividad*) de un grafo conexo G, que denotaremos por $\kappa(G)$, es el menor número de vértices que hay que suprimir del grafo para que el grafo resultante pase a ser desconexo o pase a ser el grafo trivial.

Así los grafos representados en la Figura 2.6(a), (b) y (c) tienen vértice-conectividad 1, mientras que el grafo (d) tiene vértice-conectividad 2.

Notemos que la conectividad $\kappa(G)$ no excede a la aristo-conectividad $\lambda(G)$ en ninguno de los grafos representados en la Figura 2.6. Veamos cómo esta desigualdad se tiene para todos los grafos conexos.

Teorema 2.3 *Para cualquier grafo conexo* $G = (V, E)$*,*

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

donde $\delta(G)$ *es el mínimo de los grados de los vértices del grafo* G*.*

Dem.

Si v es un vértice de grado $\delta(G)$, entonces G se puede desconectar eliminando todas las aristas incidentes con v. De esto se sigue que $\lambda(G)$, el número mínimo de aristas cuya eliminación desconecta G, no puede exceder $\delta(G)$. De aquí tenemos $\lambda(G) \leq \delta(G)$.

Veamos que $\kappa(G) \leq \lambda(G)$. Sea G un grafo con aristo-conectividad λ. Entonces, al menos hay un conjunto de λ aristas cuya eliminación desconecta G en al menos dos componentes conexas G_1 y G_2, como se ilustra en la Figura 2.7. Sin embargo, podemos también eliminar estas aristas eliminando como mucho λ vértices extremos de estas aristas (eligiendo un vértice extremo de cada una de las aristas). De aquí se sigue que el número mínimo de vértices que hay que eliminar del grafo para desconectarlo no puede exceder λ, o lo que es lo mismo $\kappa(G) \leq \lambda(G)$. □

Figura 2.7: *Ilustración de la desigualdad de Whitney*

2.4. Circuitos Eulerianos

El problema de los puentes de Königsberg

Durante el siglo XVIII, la ciudad de Königsberg, en Prusia Oriental (ahora llamada Kaliningrado y parte de la República de Rusia), estaba dividida en cuatro zonas (incluida la isla de Kneiphof) por el río Pregel. Siete puentes comunicaban estas regiones, como se muestra en la Figura 2.8. Se decía que los habitantes hacían paseos dominicales tratando de encontrar una forma de caminar por la ciudad cruzando cada puente exactamente una vez y regresando al punto donde se había iniciado el paseo.

El matemático suizo Leonhard Euler resolvió este problema. Su solución, publicada en 1736, es el primer trabajo conocido de Teoría de Grafos. Euler estudió este problema utilizando el multigrafo obtenido cuando las cuatro regiones se representan por medio de vértices y los puentes por aristas. El multigrafo resultante está representado en la Figura 2.9.

El problema de los puentes de Königsberg en términos de grafos queda formulado como: ¿Hay algún circuito en este multigrafo que contenga todas las aristas?

Figura 2.8: *Situación de los puentes de Königsberg*

2.4.1. Circuitos y Cadenas de Euler

Un *circuito de Euler* en un grafo G es una cadena cerrada que contiene todas las aristas del grafo, mientras que una *cadena de Euler* en G es aquella cadena que contiene todas las aristas del grafo. Si un grafo contiene un circuito de Euler le llamamos *Euleriano*.

Figura 2.9: *Multigrafo que modela el problema de los puentes de Königsberg*

Ejemplo 2.5

El grafo representado en la Figura 2.10(b) tiene varios circuitos Eulerianos, uno de ellos es 12345142531 y no tiene ninguna cadena Euleriana, mientras que el grafo representado en la Figura 2.10(a) presenta varias cadenas Eulerianas, por ejemplo DCABC o CBACD, pero no presenta ningún circuito Euleriano.

Figura 2.10: *Cadenas y circuitos eulerianos*

A continuación vamos a presentar una caracterización de los grafos que contienen un circuito Euleriano.

Teorema 2.4 *Un multigrafo conexo G tiene un circuito de Euler si y sólo si sus vértices tienen grado par.*

Dem.

- Veamos que si G es un multigrafo conexo Euleriano, entonces todos los vértices de G tienen grado par.

  Sea

  $$x_0, e_1, x_1, e_2, x_2, \cdots, x_{m-1}, e_m, x_m = x_0, \quad \text{con } e_i \neq e_j, \, \forall i \neq j$$

  un circuito de Euler contenido en G.

  Notemos que cada vez que la ruta se encuentra con un vértice interno del circuito, x_i, $\forall i \in \{1, 2, \cdots, m-1\}$, hay dos aristas que inciden sobre él: con una *entramos* a x_i y con la otra *salimos*. Por lo tanto, contribuimos en dos unidades al grado de x_i. Esto también se satisface para el vértice extremo x_0, ya que *salimos* y *entramos* al mismo vértice al principio y final de la ruta, respectivamente. Con este argumento queda demostrado que el grado de los vértices de un multigrafo conexo Euleriano es par.

- Veamos que si G es un multigrafo conexo con todos sus vértices de grado par, entonces G es Euleriano.

  Sea G un multigrafo conexo tal que todos los vértices tienen grado par. Si el número de aristas de G es dos, entonces G debe ser como el multigrafo de la Figura 2.11. El circuito Euleriano es inmediato en este caso. Ahora procederemos por inducción y supondremos que el resultado es válido para todas las situaciones con menos de m aristas. Si G tiene m aristas, seleccionamos un vértice v en G como punto inicial para construir un circuito Euleriano. Como el grafo (o multigrafo) G es conexo y cada vértice tiene grado par, podemos construir al menos un circuito C que contenga a v. Si el circuito contiene todas las aristas de G, hemos terminado. Si no, eliminamos las aristas del circuito de G, asegurándonos de eliminar cualquier vértice que haya quedado aislado. El subgrafo restante H tiene todos los vértices de grado par, pero puede no ser conexo. Sin embargo, cada componente conexa de H es conexa y tendrá un circuito Euleriano. (¿Por qué?, pues por hipótesis de inducción: por ser grafos de tamaño menor que m con todos sus vértices de grado par). Además cada uno de estos circuitos Eulerianos tiene un vértice que está en C. En consecuencia, podemos partir de v y recorrer C hasta llegar al vértice v_1 que está en el circuito Euleriano

de una componente conexa C_1 de H. Entonces se recorre este circuito Euleriano y al regresar a v_1 continuamos en C hasta llegar a un vértice v_2 que está en el circuito Euleriano de la componente C_2 de H. Como el grafo es finito, podemos continuar este proceso hasta construir un circuito Euleriano para G.

Figura 2.11: *Multigrafo*

A continuación presentamos un procedimiento (que establecemos sin demostrar) que nos permite construir un circuito Euleriano en un grafo Euleriano dado.

Algoritmo de Fleury

Si G es un grafo Euleriano, entonces llevando a cabo los siguientes pasos se obtendrá un circuito Euleriano en G:

- *Paso 1: Elegir un v vértice inicial (de forma arbitraria).*
- *Paso 2: En cada etapa, recorrer cualquier arista disponible, eligiendo una arista puente únicamente cuando no queden más alternativas.*
- *Paso 3: Después de recorrer una arista, eliminarla del grafo (al igual que los vértices de grado 0 que resulten de eliminar aristas), y entonces elegir otra arista disponible en el vértice extremo donde finalizó la última arista recorrida.*
- *Paso 4: Parar cuando no queden aristas en el grafo.*

Este algoritmo es muy fácil de aplicar. En cada etapa, elegimos una arista incidente con el vértice final de la arista recorrida en la etapa anterior. Se eligirá una arista puente *únicamente como último recurso*; este requisito es necesario, ya que una vez que hemos atravesado una arista puente, no podemos volver a la parte del grafo que hemos abandonado.

Ilustremos el algoritmo de Fleury aplicándolo al grafo representado en la Figura 2.12(a):

- Paso 1: Elegimos el vértice u para iniciar el algoritmo.
- Pasos 2-3: De todas las aristas posibles recorremos en primer lugar la arista ua, y a continuación atravesamos la arista ab.

  Eliminamos del grafo las dos aristas que hemos atravesado (y el vértice a) dándonos como resultado el grafo de la Figura 2.12(b).

  Seguimos por la arista bc (descartamos de momento bu por ser una arista puente), y a continuación cd y db.

  Eliminamos estas aristas (y los vértices c y d), obteniendo como resultado el grafo de la Figura 2.12(c).

  Ahora como no hay más alternativas atravesamos la arista puente bu, y a continuación recorremos el ciclo $uefu$.
- Paso 4: Paramos porque no quedan más aristas por recorrer. Por tanto, el circuito Euleriano que hemos obtenido al realizar el procedimiento es $uabcdbuefu$

Figura 2.12: *Algoritmo de Fleury*

Y para finalizar la sección presentamos una caracterización de los multigrafos que contienen una cadena euleriana.

Teorema 2.5 *Si G es un multigrafo conexo, podemos construir una cadena euleriana en G si y sólo si G tiene exactamente dos vértices de grado impar.*

Dem.

- Supongamos que G es un multigrafo conexo que tiene una cadena euleriana de a a b, pero no un circuito Euleriano, y veamos que G tiene exactamente 2 vértices de grado impar.

  Sea

  $$a = x_0, e_1, x_1, e_2, x_2, \cdots, x_{m-1}, e_m, x_m = b, \quad \text{con } e_i \neq e_j, \forall i \neq j$$

  un cadena de Euler contenido en G.

  Notemos que cada vez que la ruta se encuentra con un vértice interno de la cadena, x_i, $\forall i \in \{1, 2, \cdots, m-1\}$, hay dos aristas que inciden sobre él: con una *entramos* a x_i y con la otra *salimos*. Por lo tanto, contribuimos en dos unidades al grado de x_i. Mientras que la primera arista de la cadena contribuye en una unidad al grado del vértice a, al igual que la última arista sobre el grado del vértice b. Con este argumento demostramos que el grado de los vértices del grafo es par a excepción de a y b que tienen grado impar.

- Supongamos que G es un multigrafo conexo que tiene exactamente dos vértices de grado impar, llamémosles a y b, y veamos que G tiene una cadena Euleriana.

  Consideremos un nuevo grafo G_1 que es el resultado de añadir la arista ab al grafo G. Ahora tenemos un multigrafo G_1 conexo con todos sus vértices de grado par. Por lo tanto, G_1 tiene un circuito Euleriano C; cuando eliminamos la arista ab de C, obtenemos una cadena euleriana para G. (Así, la cadena euleriana comienza en uno de los vértices de grado impar y término en el otro vértice de grado impar.)

2.5. Ciclos Hamiltonianos

Un viaje alrededor del mundo

En 1859, el matemático irlandés Sir William Rowan Hamilton desarrolló un juego que vendió a un fabricante de juguetes de Dublín. El juego era un dodecaedro regular de madera con 20 esquinas (vértices) en las que aparecían inscritos los nombres de ciudades importantes (ver figura inferior). El objetivo del juego era encontrar una ruta a lo largo de las aristas del sólido de modo que pasara a través de cada ciudad exactamente una vez y que regresara al punto de partida.

2.5.1. Ciclos y Caminos Hamiltonianos

Un *ciclo Hamiltoniano* en un grafo G es un ciclo que contiene todos los vértices del grafo, mientras que un *camino Hamiltoniano* en G es un camino no cerrado que contiene todos los vértices de G. Si un grafo contiene un ciclo de Hamilton le llamamos *Hamiltoniano*.

Ejemplo 2.6

El grafo de la Figura 2.13(a) tiene varios caminos Hamiltonianos, dos de los cuales son 12435 y 34521, y ningún ciclo Hamiltoniano. Mientras que el grafo de la Figura 2.13(b) tiene varios ciclos Hamiltonianos, dos de los cuales son 1234567891 y 7654321987, y también tiene varios caminos Hamiltonianos, por ejemplo 123456789 y 765432198, los cuales se obtienen de los ciclos Hamiltonianos al eliminar el vértice final.

Figura 2.13: *Caminos y ciclos hamiltonianos*

A primera vista, el problema de decidir si un grafo dado es o no Hamiltoniano parece muy similar al problema de decidir si un grafo es o no Euleriano, y podríamos esperar hallar una condición necesaria y suficiente para que un grafo sea Hamiltoniano, análoga a la dada en el Teorema 1.2 para grafos Eulerianos. Sorprendentemente, tal condición no se conoce. Sin embargo, se conocen muchos teoremas que dan condiciones suficientes para la existencia de ciclos Hamiltonianos. Así como, ciertas propiedades que pueden ser utilizadas para probar si un grafo tiene ciclos Hamiltonianos. Por ejemplo:

- Un grafo con un vértice de grado 1 no puede ser Hamiltoniano, ya que en un ciclo Hamiltoniano cada vértice es incidente con dos aristas del ciclo.

- Si $G = (V, E)$ tiene un ciclo Hamiltoniano, entonces para $v \in V$, $d(v) \geq 2$.

- Si $v \in V$ y $d(v) = 2$, entonces las dos aristas incidentes con el vértice v deben estar incluidas en cualquier ciclo Hamiltoniano de G.

- Si $v \in V$ y $d(v) > 2$, cuando tratamos de construir un ciclo Hamiltoniano, una vez que hemos pasado por el vértice v, dejamos de tener en cuenta las aristas no utilizadas incidentes con v.

- Al construir un ciclo Hamiltoniano para G, no podemos obtener un ciclo para un subgrafo de G a menos que contenga todos los vértices de G.

- Si al quitar k vértices del grafo G se producen más de k componentes conexas, entonces G no es Hamiltoniano. Se deduce del hecho de que en un ciclo la anterior situación es imposible.

Ejemplo 2.7

¿Tiene el grafo representado en la Figura 2.14(a) un ciclo Hamiltoniano? ¿Y el grafo representado en la Figura 2.14(b)?

En el primer caso, la respuesta es que no por tener el grafo un vértice de grado 1. Consideremos ahora el grafo representado en la Figura 2.14(b). Dado que el grado de los vértices a, b, d y e es 2, cada arista incidente con estos vértices debe formar parte de cualquier ciclo Hamiltoniano. Ahora es fácil ver que este grafo no puede ser Hamiltoniano, porque de existir un ciclo Hamiltoniano este debería contener las cuatro aristas incidentes con el vértice c, lo cual es imposible.

A continuación presentamos dos de los más importantes resultados que establecen condiciones suficientes para la existencia de ciclos Hamiltonianos.

Figura 2.14: *Estudio de ciclos Hamiltonianos*

Teorema 2.6 (Teorema de Dirac) *Sea G un grafo simple con n vértices, siendo $n \geq 3$. Si $d(v) \geq \frac{n}{2}$ para cada vértice v del grafo, entonces G es Hamiltoniano.*

Teorema 2.7 (Teorema de Ore) *Sea G un grafo simple con n vértices, siendo $n \geq 3$. Si $d(v) + d(w) \geq n$, para cualquier par de vértices v y w del grafo, entonces G es Hamiltoniano.*

Notemos que si $d(v) \geq \frac{n}{2}$ para cada vértice v del grafo, entonces $d(v)+d(w) \geq n$ para cada par de vértices v y w no adyacentes. Por lo tanto, el teorema de Dirac se puede deducir a partir del teorema de Ore, por esta razón únicamente probaremos el teorema de Ore.

Dem.

Procederemos por reducción al absurdo, y supondremos que existe un grafo no Hamiltoniano en el cual $d(v) + d(w) \geq n$ para cualquier par de vértices v y w no adyacentes del grafo.

Añadimos aristas a G hasta obtener un subgrafo H de K_n tal que H no contenga un ciclo Hamiltoniano, pero que para cualquier arista e, del grafo completo K_n, que no está en H, $H + e$ sí tiene un ciclo Hamiltoniano.

Como $H \neq K_n$, existen vértices a, $b \in V$ tales que ab no es arista de H pero tal que $H + ab$ tiene un ciclo Hamiltoniano C. El grafo H no tiene dicho ciclo, por lo que la arista ab forma parte del ciclo C. Enumeraremos los vértices de H (y de G) sobre el ciclo C como sigue:

$$a(= v_1) \to b(= v_2) \to v_3 \to v_4 \cdots \to v_{n-1} \to v_n \to a$$

Para cualquier i, siendo $3 \leq i \leq n$, si la arista bv_i está en el grafo H, entonces afirmamos que la arista av_{i-1} no puede ser una arista de H, ya que

si ambas aristas están en H, para algún i, con $3 \leq i \leq n$, obtenemos el ciclo Hamiltoniano

$$\mathbf{b} \to \mathbf{v_i} \to v_{i+1} \to \cdots \to v_{n-1} \to v_n \to \mathbf{a} \to \mathbf{v_{i-1}} \to v_{i-2} \cdots v_4 \to v_3 \to b$$

en el grafo H (que no tiene ciclos Hamiltonianos). Por lo tanto, para cada $3 \leq i \leq n$, como máximo una de las aristas bv_i, av_{i-1} está en H. En consecuencia,

$$d_H(a) + d_H(b) < n,$$

donde $d_H(v)$ denota el grado del vértice v en H. Para cualquier $v \in V(G)$, $d_H(v) \geq d_G(v) = d(v)$, por lo que tenemos los vértices a, b no adyacentes (en G) que cumplen

$$d(a) + d(b) < n.$$

Esto contradice la hipótesis $d(v) + d(w) \geq n$ para cualquier par de vértices no adyacentes, por lo que rechazamos nuestra suposición y vemos que G contiene un ciclo Hamiltoniano.

Ejemplo 2.8

Ilustremos el uso de estos teoremas considerando los grafos de la Figura 2.15.

Para el grafo de la Figura 2.15(a), $n = 6$ y $d(v) = 3$ para todos los vértices del grafo, por lo tanto, aplicando el teorema de Dirac, este grafo es Hamiltoniano.

Para el grafo de la Figura 2.15(b), $n = 5$ pero $d(u) = 2$, por lo que no podemos aplicar el teorema de Dirac. Sin embargo, $d(v) + d(w) \geq 5$ para todos los pares de vértices no adyacentes v y w, por lo tanto, aplicando el teorema de Ore podemos concluir que el grafo es Hamiltoniano.

Figura 2.15: *Teoremas de Dirac y Ore*

Capítulo 3

Grafos: Árboles

3.1. Árboles

Hay un tipo especial de grafo, denominado *árbol*, que se presenta en múltiples aplicaciones. En ciencias de la computación los árboles son particularmente útiles. Por ejemplo se utilizan para organizar información de tal modo que sea posible efectuar eficientemente operaciones que atañan a esa información. Para construir algoritmos eficientes para localizar artículos en una lista. Para construir códigos eficientes para almacenar y transmitir datos. Para modelar procedimientos que son llevados a cabo al utilizar una secuencia de decisiones. Introduzcamos pués el concepto de árbol.

Árbol

Un *árbol* es un grafo $T = (V, E)$ acíclico y conexo

Dado que un árbol no puede tener ciclos, no podrá contener ni aristas múltiples ni bucles, por lo tanto cualquier árbol debe ser un grafo simple.

Ejemplo 3.1

En la Figura 3.1 tenemos representados todos los árboles de orden 6 no isomorfos.

Notemos que un árbol es un grafo minimal respecto a la propiedad de ser conexo (al eliminar una arista cualquiera del grafo, éste deja de ser conexo) y máximal respecto a la propiedad de ser acíclico (al añadir una arista cualquiera del grafo, formamos un ciclo).

Figura 3.1: *Árboles de orden* 6 *no isomorfos*

Empezaremos nuestro estudio presentando algunas propiedades que nos permitirán caracterizar a los árboles.

Teorema 3.1 *Si* $T = (V, E)$ *es un grafo de orden* n*, son equivalentes*

1. T *es un árbol.*
2. *Entre cualquier par de vértices de* T *existe un único camino.*
3. T *es conexo y toda arista de* T *es una arista puente.*
4. T *es acíclico y maximal respecto a esta propiedad.*
5. T *es conexo y tiene tamaño* $n - 1$*.*
6. T *es acíclico y tiene tamaño* $n - 1$*.*

Dem.

- *1* $\leftrightarrow$ *2*

  T es árbol $\leftrightarrow$ T es un grafo conexo y acíclico $\leftrightarrow$ $\forall\, u, v \in V$, existe al menos un $u - v$ camino sin ciclos $\leftrightarrow$ existe un único $u - v$ camino (ya que si existieran dos $u - v$ caminos, llamémosles C_1 y C_2, su unión $C_1 \cup C_2$ sería un recorrido cerrado que contendría un ciclo)

- *1* $\leftrightarrow$ *3*

  T es árbol $\leftrightarrow$ T es conexo y acíclico $\leftrightarrow$ T es conexo y $\forall\, e \in E$, e no pertenece a ningún ciclo $\leftrightarrow$ T es conexo y cualquier arista del grafo es arista puente.

- *1* $\leftrightarrow$ *4*

  Veamos primero la demostración de *1*$\rightarrow$ *4*:

  Si T es árbol y $u, v \in V$ no adyacentes $\rightarrow$ existe un único $u - v$ camino $\rightarrow T + uv$ contiene exactamente un ciclo.

  Veamos a continuación *4* $\rightarrow$ *1*:

  Para demostrar que T es árbol, sólo nos falta probar que es conexo.

  Consideremos $u, v \in V(T)$.

  - Si u y v son adyacentes, entonces en T tenemos un $u - v$ camino de longitud 1.
  - Si u y v no son adyacentes en $T \rightarrow T + uv$ contiene exactamente un ciclo $\rightarrow$ eliminando la arista uv tenemos un único $u - v$ camino en T.

- *1* $\leftrightarrow$ *5*

  Veamos primero la demostración de *1*$\rightarrow$ *5*:

  Por ser T árbol $\rightarrow T$ es conexo. Para probar que $|E| = n - 1$, procederemos por inducción sobre n.

  Si $n = 1$ se satisface la propiedad, dado que el árbol de orden 1 tiene $1 - 1 = 0$ aristas.

  Supongamos que la propiedad es cierta para todos los árboles de orden $k < n$. Hipótesis de inducción: Si $|V(T)| = k \leq n \rightarrow |E(T)| = k - 1$. Y veamos que la propiedad se cumple para los árboles de orden $n + 1$.

  Sea $e = uv \in E(T) \rightarrow e$ es el único $u - v$ camino que une u y v en T $\rightarrow T - e$ tiene exactamente dos componentes conexas, llamémosles T_u y T_v, cada una de las cuales es árbol por ser subgrafos de T.

  Como $|V(T_u)| = n_u \leq n$ y $|V(T_v)| = n_v \leq n \rightarrow$ aplicando la hipótesis de inducción, tendremos que $|E(T_u)| = n_u - 1$ y $|E(T_v)| = n_v - 1$. Por lo tanto en T tendremos

  $$|E(T)| = n_u - 1 + n_v - 1 + 1 = n_u + n_v - 1 = (n + 1) - 1 = n$$

  Por lo tanto, aplicando el Principio de Inducción, la propiedad es cierta para todos los árboles de orden $n \leq 1$.

  Veamos a continuación *5* $\rightarrow$ *1*:

T es conexo y tiene tamaño $n-1 \rightarrow$ para probar que T es árbol, nos falta ver que T es acíclico. Sabemos que el tamaño de un grafo conexo de orden n es $\leq n-1$, por lo que T no contiene ningún ciclo.

- *1* $\leftrightarrow$ *6*

  La demostración de *1*$\rightarrow$ *6* es análoga a la realizada en el apartado anterior. Veamos a continuación *6*$\rightarrow$ *1*:

  T es acíclico y $|E| = n-1 \rightarrow$ para probar que T es árbol, nos falta ver que T es conexo. Demostrémoslo procediendo por reducción al absurdo. Supongamos que T no es conexo $\rightarrow$ existirán $T_1, T_2, \cdots T_k$ k componentes conexas que son árboles por ser conexas y acíclicas $\rightarrow$ $|V(T_i)| = n_i$ y $|E(T_i)| = n_i - 1$ $\forall i$, siendo $\sum_{i=1}^{k} n_i = n \rightarrow |E(T)| = \sum_{i=1}^{n} n_i - 1 = n - k \neq n - 1$ contradicción $\rightarrow T$ es conexo. $\square$

3.2. Árboles Generadores

Un problema importante que está asociado a una red representada por un grafo consiste en obtener un árbol que contenga todos los vértices del grafo y algunas de sus aristas para asegurar la conectividad, o lo que es lo mismo, un *árbol generador* del grafo.

Árbol Generador

Un *Árbol generador* (*spanning tree*) de un grafo G es un subgrafo generador de G que además es un árbol.

Ejemplo 3.2

En la Figura 3.2 tenemos representados dos árboles generadores del grafo G de orden 5. Notemos que los dos árboles tienen orden 5 (por ser generadores) y tamaño 4.

Teorema 3.2 *Todo grafo conexo G contiene un árbol generador.*

Dem.
Sea G un grafo conexo de orden n y tamaño m, veamos que G contiene un árbol generador. Para ello procederemos por inducción sobre m.

Figura 3.2: *Arboles generadores*

- Si $m = n - 1$ tenemos que G es un grafo conexo de tamaño $n - 1$, y aplicando el Teorema 1.1 de caracterización, tenemos que G es árbol.
- Hipótesis de inducción: Supongamos que todo grafo conexo de tamaño $m \leq k$ contiene un árbol generador.
- Veamos a continuación si la propiedad se cumple para los grafos conexos de tamaño $k + 1$.

  Sea G un grafo conexo de tamaño de tamaño $k + 1 > n - 1 \rightarrow G$ no es acíclico, pues si lo fuera, G sería árbol (por ser acíclico y conexo), con lo que $|E(G)| = n - 1$, lo cual es absurdo.

  Sea $e \in E$ una arista del grafo contenida en un ciclo $\rightarrow e$ no es arista puente, por lo que $G - e$ es un grafo conexo, de tamaño

  $$|E(G - e)| = (k + 1) - 1 = k.$$

  Aplicando la hipótesis de inducción a $G - e$, tendremos que este grafo tiene un árbol generador T, el cual también es árbol generador de G, dado que $V(T) = V(G)$ y $E(T) \subseteq E - \{e\} \subset E$. □

Hay varias técnicas para obtener un árbol generador correspondiente a un grafo G dado. Un enfoque consiste en eliminar las aristas que pertenezcan a ciclos, una tras otra hasta que no queden ciclos en el grafo.

Método Cutting-down

Empezaremos por elegir cualquier ciclo de un grafo G conexo y eliminar una de sus aristas. (Si no hubieran ciclos, el grafo G por símismo sería un árbol generador). Ya que no se puede desconectar un grafo al eliminar una arista

e de un ciclo, el grafo resultante $G - e$ sigue siendo conexo. A continuación repetimos este proceso hasta que no queden ciclos en el grafo resultante. Este procedimiento dará lugar al árbol generador T que andamos buscando.

Ejemplo 3.3

Del grafo representado en la Figura 3.3(a), podemos eliminar las aristas
vy (destruimos el ciclo v, w, y, v),
yz (destruimos el ciclo v, w, y, z, v),
xy (destruimos el ciclo w, x, y, w),
y asíobtener el árbol generador representado en la Figura 3.3(b).

Figura 3.3: *Método Cutting-down*

Un enfoque alternativo (y posiblemente más eficiente) para la obtención de un árbol generador es la selección de una sucesión de $n - 1$ aristas, una a una, tales que en cada paso el subgrafo actual sea acíclico.

Método Building-up
Seleccionamos aristas del grafo G de una en una, de tal modo que en cada selección no se cree ningún ciclo, y repetimos este procedimiento hasta que todos los vértices del grafo se hayan incluido.

Ejemplo 3.4

Para el grafo de la Figura 3.3(a), podemos elegir las aristas vz, wx, xy e yz, no hemos creado ningún ciclo y obtenemos el árbol generador de la Figura 3.3(c).

Notemos que tanto la búsqueda en profundidad como la búsqueda en anchura utilizan este último enfoque.

3.3. Búsqueda en árboles

Un problema que surge frecuentemente en la práctica es el de buscar sistemáticamente a través de alguna estructura de árbol. Por ejemplo, un directorio en un ordenador puede estar organizado como una estructura de datos tipo árbol, y será necesaria una búsqueda sistemática en el árbol siempre que se requiera una información particular. En la práctica, ésto implica examinar cada parte del árbol hasta que el vértice o la arista deseada sea encontrada. En orden a evitar un gasto innecesario de tiempo y recursos de procesado, necesitamos una técnica de búsqueda que nos garantice visitar todas las partes del árbol, sin visitar demasiadas veces alguna de estas partes. Esencialmente, podemos emplear dos estrategias distintas. Podríamos profundizar, moviéndonos a un nuevo vértice siempre que fuera posible (*búsqueda en profundidad (DFS)*), o podríamos desplegarnos, comprobando todos los vértices en un nivel antes de pasar al siguiente (*búsqueda en anchura (BFS)*). En el siguiente apartado presentaremos la primera estrategia.

3.3.1. Búsqueda en Profundidad

La idea básica de la búsqueda en profundidad es penetrar tan profundamente como sea posible dentro de un árbol antes de desplegarse a otros vértices. Esto se consigue al tomar el nuevo vértice adyacente al último de los posibles vértices anteriores.

Algoritmo DFS

Entrada	un grafo G y un vértice v de G,
Salida	Un árbol generador de la componente conexa de v.
Inciar	$L := \{v\}, W := \{v\}, B := \emptyset$;
Mientras	$L \neq \emptyset$ hacer
	x:=último de L;
	Si x es adyacente a algún y, $y \notin W$,
	entonces $L := L \cup \{y\}$, $W := W \cup \{y\}$, $B := B \cup \{xy\}$;
	en otro caso $L := L - \{x\}$
	FinalSi
FinalMientras	
Escribir	(W, B)

Teorema 3.3 *Sea v un vértice del grafo G y $T = (W, B)$ el grafo construido mediante el método de búsqueda en profundidad (DFS). Entonces T es un árbol generador de la componente conexa de G que contiene al vértice v.*

Dem.

Que T es un árbol es una consecuencia directa de las reglas de la construcción. Efectivamente:

Cada adición de un vértice y a W, implica la adición de una arista xy a B con $x \in W$. Por tanto, el grafo dado por el par ordenado (W, B) es conexo en cada fase del algoritmo.

Cada arista xy que se añade a B conecta un vértice $x \in W$ y un vértice $y \notin W$, de manera que no se creen ciclos. AsíT es un árbol.

Para probar que el árbol es generador procederemos por reducción al absurdo.

Sea z un vértice en la misma componente que v, de forma que existe un camino en G

$$v = v_0, v_1, \cdots, v_k = z.$$

Si z no es de T, entonces (ya que v es de T) ha de existir un par v_i, v_{i+1} tal que v_i sea vértice de T y v_{i+1} no lo sea. Al ser v_i de T, será el vértice activo x (último de L) al menos una vez. Siempre que avancemos de v_i hacia un nuevo vértice debemos volver eventualmente a v_i, ya que se puede retroceder a través de las aristas del árbol. No podemos abandonar v_i hasta que todos sus vecinos hayan sido visitados; en particular, si v_{i+1} no es de T, no puede ser vecino de v_{i+1}, contrariamente a la hipótesis. Asípues, z debe estar en T. □

Ejemplo 3.5

Apliquemos el algoritmo DFS para hallar un árbol generador del grafo representado en la Figura 3.4. La siguiente tabla presenta los pasos seguidos hasta obtener el árbol generador representado en la Figura 3.5

L	W	B	y
$\{a\}$	$\{a\}$	$\emptyset$	b
$\{a, b\}$	$\{a, b\}$	$\{ab\}$	c
$\{a, b, c\}$	$\{a, b, c\}$	$\{ab, bc\}$	d
$\{a, b, c, d\}$	$\{a, b, c, d\}$	$\{ab, bc, cd\}$	–
$\{a, b, c\}$	$\{a, b, c, d\}$	$\{ab, bc, cd\}$	e
$\{a, b, c, e\}$	$\{a, b, c, d, e\}$	$\{ab, bc, cd, ce\}$	–
$\{a, b, c\}$	$\{a, b, c, d, e\}$	$\{ab, bc, cd, ce\}$	–
$\{a, b\}$	$\{a, b, c, d, e\}$	$\{ab, bc, cd, ce\}$	–
$\{a\}$	$\{a, b, c, d, e\}$	$\{ab, bc, cd, ce\}$	–
$\emptyset$	$\{a, b, c, d, e\}$	$\{ab, bc, cd, ce\}$	–

Figura 3.4: *Árbol generador*

Figura 3.5: *Árbol generador obtenido aplicando el algoritmo DFS*

3.3.2. Búsqueda en Anchura

La idea básica de la búsqueda en anchura es desplegarse a tantos vértices como sea posible antes de penetrar en profundidad dentro de un árbol. Esto significa que visitaremos todos los vértices adyacentes a uno dado antes de cambiar de nivel.

Algoritmo BFS

Entrada	un grafo G y un vértice v de G,
Salida	Un árbol generador de la componente conexa de v.
Inciar	$L := \{v\}, W := \{v\}, B := \emptyset$;
Mientras	$L \neq \emptyset$ hacer
	x:=primero de L;
	Si x es adyacente a algún y, $y \notin W$,
	entonces $L := L \cup \{y\}$, $W := W \cup \{y\}$, $B := B \cup \{xy\}$;
	en otro caso $L := L - \{x\}$
	FinalSi
FinalMientras	
Escribir	(W, B)

Notemos que un vértice no se elimina del conjunto L hasta que no se han incorporado todos sus vértices adyacentes.

La demostración del siguiente teorema es muy parecida a la del teorema 1.3 y la dejamos como ejercicio.

Teorema 3.4 *Sea v un vértice del grafo G y $T = (W, B)$ el grafo construido mediante el método de búsqueda en anchura (BFS). Entonces T es un árbol generador de la componente conexa de G que contiene al vértice v.*

Ejemplo 3.6

Apliquemos el algoritmo BFS para hallar un árbol generador del grafo representado en la Figura 3.6.

La siguiente tabla presenta los pasos seguidos hasta obtener el árbol generador representado en la Figura 3.7

Figura 3.6: *Árbol generador*

L	W	B	y
$\{a\}$	$\{a\}$	$\emptyset$	b
$\{a, b\}$	$\{a, b\}$	$\{ab\}$	c
$\{a, b, c\}$	$\{a, b, c\}$	$\{ab, ac\}$	e
$\{a, b, c, e\}$	$\{a, b, c, e\}$	$\{ab, ac, ae\}$	–
$\{b, c, e\}$	$\{a, b, c, e\}$	$\{ab, ac, ae\}$	d
$\{b, c, e, d\}$	$\{a, b, c, d, e\}$	$\{ab, ac, ae, bd\}$	–
$\{c, e, d\}$	$\{a, b, c, d, e\}$	$\{ab, ac, ae, bd\}$	–
$\{e, d\}$	$\{a, b, c, d, e\}$	$\{ab, ac, ae, bd\}$	–
$\{d\}$	$\{a, b, c, d, e\}$	$\{ab, ac, ae, bd\}$	–
$\emptyset$	$\{a, b, c, d, e\}$	$\{ab, ac, ae, bd\}$	–

Figura 3.7: *Árbol generador obtenido aplicando el algoritmo BFS*

Notemos que la búsqueda en amplitud también se puede utilizar para hallar la distancia entre algún vértice inicial y los restantes vértices del grafo.

Esta distancia es el número mínimo de aristas que hay que recorrer para pasar desde el vértice inicial hasta el vértice concreto que se esté examinando.

3.4. El Problema del Árbol Generador Mínimo

Los árboles generadores son útiles en varios contextos. Por ejemplo, supongamos que una compañía proyecta construir una red de comunicación que conecte sus cinco delegaciones. Cualquier par de estos centros se puede enlazar mediante una línea telefónica alquilada. ¿Qué enlaces se deberían realizar para asegurarnos que todas las delegaciones están conectadas de forma que el coste total de la red sea mínimo?

Podemos modelar este problema utilizando el grafo ponderado de la Figura 3.8, donde los vértices representan las delegaciones, las aristas representan posibles líneas telefónicas en alquiler, y los pesos sobre cada arista son la tarifa mensual de arrendamiento de la línea representada por la arista. Podemos resolver este teorema encontrando un árbol generador de forma que la suma de los pesos de las aristas del árbol sea mínima. A tal árbol generador se le conoce como *árbol generador mínimo*.

Un *árbol generador mínimo* en un grafo ponderado $G = (V, E, w)$ conexo es un árbol generador que tiene mínima la suma de los pesos de sus aristas, que denotaremos por $w(T)$.

Figura 3.8: *Grafo ponderado que representa el alquiler mensual (en miles de pesetas) para líneas telefónicas entre delegaciones*

Como los valores w de los pesos de las aristas son enteros positivos, es evidente que el problema tiene solución, ya que hay un número finito de árboles generadores T de G y cada uno de ellos proporciona un valor positivo de $w(T)$. En otras palabras, existe un árbol generador minimal T_0 tal que

$$w(T_0) \leq w(T)$$

para todo árbol generador T. Notemos, sin embargo, que pueden existir varios árboles distintos con la misma propiedad.

Presentamos a continuación dos algoritmos para construir árboles generadores mínimos. Ambos proceden por añadir sucesivamente aristas de peso más pequeño de entre aquellas con una propiedad específica que no hayan sido utilizadas antes.

Algoritmo de Kruskal

Para llevar a cabo el algoritmo de Kruskal, elegimos una arista del grafo con peso mínimo. Y en pasos sucesivos añadimos aristas con peso mínimo que no formen un ciclo con aquellas aristas que ya han sido elegidas, parando después de haber elegido $n-1$ aristas.

Entrada	un grafo ponderado $G = (V, E, w)$ conexo de orden n,
Salida	Un árbol generador minimal T.
Inciar	$i := 1, E_0 := \emptyset$;
Mientras	$i < n$ hacer
	elegir $e \in E - E_{i-1}$ de peso mínimo tal que
	$T_i = (V, E_{i-1} \cup \{e\})$ es un grafo acíclico;
	$E_i := E_{i-1} \cup \{e\}$, $i := i + 1$
FinalMientras	
Escribir	$T = (V, E_{n-1})$

Ejemplo 3.7

Utilicemos el algoritmo de Kruskal para diseñar una red de coste mínimo que conecte todas las delegaciones representadas por el grafo de la Figura 3.8.

Elección	Arista	Coste
1	{ Bilbao, Madrid}	50
2	{ Madrid, Valencia }	50
3	{ Barcelona, Valencia}	60
		Coste Total 160

El árbol generador minimal resultante está representado en la Figura 3.9

Figura 3.9: *Red de coste mínimo obtenida mediante el algoritmo de Kruskal*

Ejemplo 3.8

Utilicemos el algoritmo de Kruskal para obtener un árbol generador mínimo para el grafo ponderado representado en la Figura 3.10.

Elección	Arista	Coste
1	cd	1
2	bf	1
3	kl	1
4	ab	2
5	fj	2
6	dh	2
7	bc	3
8	fg	3
9	gk	3
10	ij	3
11	ae	3
		Coste Total 24

Figura 3.10: *Un grafo ponderado*

Figura 3.11: *Árbol generador minimal obtenido mediante el algoritmo de Kruskal*

El árbol generador minimal resultante está representado en la Figura 3.11

Teorema 3.5 *El algoritmo de Kruskal da lugar a un árbol generador minimal para todo grafo ponderado conexo.*

Dem.

Por construcción, T es un árbol generador. Para demostrar que T es minimal procederemos por reducción al absurdo, y supondremos que T no es minimal.

Sean $e_1, e_2, \cdots, e_{n-1}$ el conjunto de aristas de T.

De todos los árboles generadores minimales de G, sea H uno de los que tiene el máximo número de aristas en común con T. Como que $H \neq T$, entonces existirá $e_k \in V(T)$ la primera arista de T que no está en H. Consideremos el grafo $H + e_k$ el cual contiene un ciclo, llamémosle C.

Dado que T es acíclico, entonces existirá una arista e en C la cual no es una arista de T. Entonces el grafo $H' = (H + e_k) - e$ es conexo y de tamaño $n-1$, por lo que H' es un árbol generador de G.

Como H es árbol generador minimal de G, entonces

$$w(H) \leq w(H') = w(H) + w(e_k) - w(e)$$

por lo que $w(e_k) \geq w(e)$.

Según el algoritmo de Kruskal, e_k es la arista de peso mínimo tal que $(V, \{e_1, e_2, \cdots, e_{n-1}\})$ es acíclico, por lo que $w(e_k \leq w(e))$, (sinó el algoritmo habría elegido e antes que e_k).

Así$w(e_k) = w(e)$ y $w(H) = w(H')$, entonces H' es un árbol generador minimal de G con más aristas en común con T que H. Lo cual es absurdo. □

Algoritmo de Prim

Para llevar a cabo el algoritmo de Prim, elegimos una arista del grafo con peso mínimo. En pasos sucesivos añadimos aristas con peso mínimo que son adyacentes a aristas que ya han sido elegidas en pasos previos del algoritmo, y que no formen un ciclo con éstas. Paramos después de haber elegido $n-1$ aristas.

Entrada	Un grafo ponderado $G = (V, E, w)$ conexo de orden n,
Salida	Un árbol generador minimal T.
Inciar	$e := uv$ una arista de peso mínimo,
	$V_1 := \{u, v\}$, $E_1 := \{e\}$,
	$T = (V_1, E_1)$;
Mientras	$i < n$ hacer
	$i := i + 1$;
	$e := uv$ una arista de peso mínimo incidente a un vértice
	u en T que no forme un ciclo si es añadida a T;
	$V_i := V_{i-1} \cup \{v\}$, $E_i := E_{i-1} \cup \{e\}$
	$T := (V_i, E_i)$
FinalMientras	
Escribir	$T = (V, E_{n-1})$

Ejemplo 3.9

Utilicemos el algoritmo de Prim para obtener un árbol generador mínimal para el grafo ponderado representado en la Figura 3.12.

Figura 3.12: *Un grafo ponderado*

Elección	Arista	Coste
1	cd	1
2	dh	2
3	hg	3
4	gf	3
5	fb	1
6	ab	2
7	fj	2
8	gk	3
9	kl	1
10	ae	3
11	ij	3
		Coste Total 24

El árbol generador minimal resultante está representado en la Figura 3.13

Teorema 3.6 *El algoritmo de Prim da lugar a un árbol generador minimal para todo grafo ponderado conexo.*

Dem.

Sea $G = (V, E, w)$ un grafo ponderado conexo. Supongamos que las sucesivas aristas elegidas por el algoritmo de Prim son $e_1, e_2, \cdots, e_{n-1}$. Sea S el árbol que tiene $e_1, e_2, \cdots, e_{n-1}$ como sus aristas, y sea S_k el árbol con aristas $e_1, e_2, \cdots, e_k$. Sea T un árbol generador de G que contiene las aristas

Figura 3.13: *Árbol generador minimal obtenido mediante el algoritmo de Prim*

$e_1, e_2, \cdots, e_k$, siendo k el mayor entero para el cual existe un árbol generador que contenga las primeras k aristas elegidas por el algoritmo de Prim. El teorema queda demostrado si podemos demostrar que $T = S$.

Procedamos por reducción al absurdo y supongamos que $S \neq T$, asíque $k < n - 1$. Consecuentemente, T contiene $e_1, e_2, \cdots, e_k$ pero no e_{k+1}. Consideremos el grafo que se obtiene de añadir a T la arista e_{k+1}. Dado que este grafo es conexo y tiene n aristas, demasiadas aristas para ser un árbol, debe contener un ciclo. Este ciclo contiene la arista e_{k+1} ya que no habían ciclos en T. Además, debe haber una arista en el ciclo que no pertenezca a S_{k+1} ya que S_{k+1} es un árbol. Empezando en un vértice extremo de e_{k+1}, y siguiendo el ciclo, podemos encontrar una arista e que no esté en S_{k+1} que tiene un vértice extremo que también es vértice extremo de una de las aristas $e_1, e_2, \cdots, e_k$. Eliminando e de T y añadiendo e_{k+1}, obtenemos un árbol T' con $n-1$ aristas (es un árbol ya que es acíclico y de tamaño $n - 1$). Notemos que el árbol T' contiene las aristas $e_1, e_2, \cdots, e_k, e_{k+1}$. Además, puesto que e_{k+1} fue elegido por el algoritmo de Prim en el paso $k+1$, y la arista e también era disponible en ese paso, el peso de e_{k+1} es menor o igual que el de e. De esto se sigue que T' es también un árbol generador minimal, ya que la suma de los pesos de sus aristas no excede la suma de los pesos de las aristas de T. Esto contradice la elección de k como el entero más grande tal que exista un árbol generador minimal que contenga $e_1, e_2, \cdots, e_k$. De aquí, $k = n - 1$, y $S = T$. Por lo tanto, el algoritmo de Prim produce un árbol generador minimal. □

3.5. Árboles con raíz

Los árboles aparecen en diversos contextos y con frecuencia uno de los vértices del árbol queda señalado de alguna manera especial. En general, nos referiremos al vértice distinguido como la *raíz* y un árbol con una raíz señalada será un *árbol con raíz*.

Para estudiar los árboles con raíz, es natural disponer los vértices por niveles. Decimos que el vértice raíz r está en el *nivel* 0 y que los vecinos de r están en el *nivel 1*. Para cada $k \geq 2$, el *nivel* k contiene los vértices adyacentes a los vértices del nivel $k-1$, salvo los que ya han sido asignados al nivel $k-2$. El árbol con raíz que se muestra a la izquierda de la Figura 3.14 puede redibujarse para hacer visible la posición de los vértices en los distintos niveles.

Figura 3.14: *Árbol con raíz y niveles.*

Un vértice de un árbol con raíz es una *hoja* si está en el nivel i y no es adyacente a ningún otro vértice del nivel $i+1$. Un vértice que no es una hoja es un vértice *interno*. La *altura* de un árbol con raíz es el máximo valor de k para el cual el nivel k no es vacío. Por ejemplo, el árbol de la Figura 3.14 tiene seis hojas, cuatro vértices internos y su altura es 3.

Como un árbol es un grafo conexo, cada vértice v en el nivel i ($i > 0$) es adyacente a exactamente un vértice u en el nivel $i-1$. A veces se indica esta situación refiriéndonos a u como el *padre* de v y a v como el hijo de u. Cada vértice, salvo la raíz, tiene un único padre, pero un vértice puede tener cualquier número de hijos (incluso cero).

En las aplicaciones suele ocurrir que cada vértice interno tenga el mismo número de hijos. Si cada padre tiene m hijos, tenemos un árbol $m-ario$

con raíz; en particular, si $m = 2$ utilizamos la palabra *binario* y si $m = 3$, la palabra *ternario*.

Teorema 3.7 *La altura de un árbol con raíz m-ario con l hojas es al menos* $\log_m l$.

Dem.

Dado que

$$h \geq log_m l \quad \leftrightarrow \quad m^h \geq l,$$

es suficiente demostrar la afirmación equivalente de que un árbol con raíz m-ario de altura h posee como máximo m^h hojas. La demostración es por inducción sobre h.

La demostración es cierta para $h = 0$, ya que en este caso el árbol tiene sólo un vértice, la raíz, que es una hoja. Supongámosla cierta para $0 \leq h \leq h_0$ y sea T un árbol con raíz m-ario de altura $h_0 + 1$. Si suprimimos de T la raíz y las aristas que inciden sobre la raíz, obtenemos m árboles $T_1, \cdots T_m$; señalamos sus raíces como los vértices de T en el nivel 1. Cada T_i es un árbol con raíz de altura no superior a h_0 y, por hipótesis de inducción, tiene a los sumo m^{h_0} hojas. Pero las hojas de T son precisamente las hojas de los árboles $T_1, \cdots, T_m$, con lo que el número de hojas de T es menor o igual que $m \cdot m^{h_0} = m^{h_0+1}$.

Luego aplicando el principio de inducción el resultado es cierto para todo $h \geq 0$. □

Dado que $log_m l$ no acostumbra a ser un entero mientras que h sílo es, podemos mejorar ligeramente el enunciado del teorema anterior. Por ejemplo, si $m = 3$ y $l = 10$, entonces la desigualdad

$$h \geq log_m l = 2{,}0959 \cdots$$

implica que $h \geq 3$. En general, podemos decir que

$$h \geq \lceil log_m l \rceil,$$

donde $\lceil x \rceil$ es el menor entero z tal que $z \geq x$.

3.5.1. Árboles de decisión

Un aplicación frecuente del teorema 3.7 es al estudio de los *árboles de decisión*. Cada vértice interno de un árbol de decisión representa una decisión, y los posibles resultados de la decisión, se representan por las aristas que conducen al siguiente nivel. El resultado final del proceso está representado por las hojas del árbol. Si el resultado de cada decisión es simplemente que una afirmación es cierta o es falsa, tenemos un árbol binario. El ejemplo siguiente se refiere a un árbol ternario.

Ejemplo 3.10

El problema de la moneda falsa. Supongamos que tenemos una moneda auténtica, con la etiqueta 0, y otras r monedas, indistinguibles de 0 por su apariencia, salvo en que llevan las etiquetas 1,2, $\cdots$, r. Se sospecha que una de las monedas puede ser falsa -demasiado ligera o demasiado pesada. Demostrar que se necesitan al menos $\lceil log_3(2r+1) \rceil$ pesadas en una balanza para decidir qué moneda (si es que hay alguna) es falsa y si es más ligera o más pesada. Y diseñar un procedimiento que utilice exactamente este número de pesadas en el caso $r = 4$.

Hay $2r + 1$ resultados posibles, u hojas en el árbol de decisión,

$$B, 1P, 1L, \cdots, rP, rL,$$

que significan que todas las monedas son buenas, la moneda 1 es pesada, la moneda 1 es ligera, etc. El árbol de decisión es ternario, ya que para cada pesada hay tres resultados posibles (al pesar un montón de monedas contra otro), que son los siguientes:

$<$: el izquierdo es menos pesado,
$=$: los dos pesan igual,
$>$: el izquierdo es más pesado.

Asípues, la altura del árbol de decisión es al menos $\lceil log_3(2r+1) \rceil$.

Si $r = 4$, entonces $\lceil log_3(2r+1) \rceil = 2$ y una solución con dos pesadas se muestra en la Figura 3.15.

Figura 3.15: *Solución al problema de la moneda falsa cuando* $r = 4$.

Árboles y algoritmos de ordenación

A principio de curso presentamos el *algoritmo de la burbuja*, un procedimiento para disponer una lista $x_1, x_2, \cdots, x_n$ de enteros distintos en orden creciente. En el algortimo intervenían comparaciones entre números enteros y movimientos de los datos; este proceso puede representarse en forma de un árbol de decisión.

Cada vértice del árbol de decisión representa una comparación de dos enteros, por ejemplo los que en ese momento tienen las etiquetas x_i y x_j. Tenemos dos resultados posibles, $x_i < x_j$ o $x_i >< x_j$, con lo que el árbol de decisión es binario. En el algoritmo de la burbuja, en cada comparación intervienen enteros adyacentes x_i y x_{i+1}, y las reglas que nos indican qué par hay que comparar en cada paso no dependen del resultado de comparaciones previas (desde luego, los valores actuales de x_i y x_{i+1} sídependen de las comparaciones previas). En la Figura 3.16 se ilustra el árbol de decisión del algoritmo de la burbuja para $n = 3$. Hay $3! = 6$ resultados posibles, que corresponden a las permutaciones del orden inicial $\alpha\beta\gamma$; las hojas del árbol de decisión representan estos resultados, junto con algunas situaciones imposibles.

Para un n cualquiera, el número de hojas del árbol de decisión es al menos $n!$, independientemente del algorirtmo que se use. Por su parte, la altura del árbol de decisión es igual al número de comparaciones necesarias (denotémoslo $s(n)$). Se sigue del teorema 3.7 que

$$s(n) \geq log_2(n!)$$

Ahora bien, $log_2(n!)$ es $O(nlogn)$, ya que es la suma de n términos $logi$ con

Figura 3.16: *El árbol de decisión del algoritmo de la burbuja (3 elementos).*

$1 \leq i \leq n$ y ningún término es mayor que $log_2 n$. Así pues, el número de comparaciones necesarias en cualquier algoritmo de ordenación es al menos $O(nlogn)$

Bibliografía

1. [*Biggs*] Norman L. Biggs, *Matemática Discreta*, Ed. Vicens Vives, 1994.

2. [*Bogart*] Kenneth P. Bogart, *Matemáticas Discretas*, Ed. Limusa, 1996.

3. [*Gimbert*] Gimbert J. y otros, *Apropament a la Teoria de Grafs i als seus Algorismes*, Edicions de la Universitat de Lleida, 1998.

4. [*Grassmann*] W. K. Grassmann y J. P. Tremblay, *Matemática Discreta y Lógica*, Ed. Prentice Hall, 1996.

5. [*Grimaldi*] Ralph P. Grimaldi, *Matemáticas Discreta y Combinatoria*, Ed. Addison-Wesley Iberoamericana, 1997.

6. [*Rosen*] Rosen K. H., *Discrete Mathematics and its Applications*, Ed. MacGraw-Hill International, 1999.