Capítulo 1 –
Grafos

Autor: Ing. Enrique Reinosa

Introducción

En este capítulo se analizará lo esencial de los grafos, este análisis se concentrará en el punto de vista de la gestión de los datos.

La Teoría de Grafos involucra un análisis matemático formal el cual no trataremos en este capítulo, debido a que escapa a los objetivos perseguidos en esta publicación y que existen publicaciones específicas a tal fin, solo se analizarán las características relevantes de los grafos, como así también todo lo que involucra la modelización de problemas a través de grafos, su utilización en las Ciencias de la Computación y sus características diferenciales.

Concepto de Grafos

Un grafo puede definirse como **G = (V, A)**, donde **V** representa a un conjunto de puntos, llamados ***vértices o nodos***, y **A** es un conjunto de relaciones entre pares de vértices, llamadas ***aristas o arcos***. De esta forma un grafo es un conjunto de vértices y arcos que los relacionan.

Desde el concepto matemático un grafo es un conjunto, pero a diferencia de los conjuntos convencionales conformados solo por elementos de igual tipo, el grafo se conforma por elementos (vértices) y relaciones (aristas) que los vinculan en pares ordenados del tipo ***xy***, donde ***x*** es el vértice origen del arco e ***y*** el destino del mismo estableciendo el par ordenado ***xy***.

El siguiente gráfico ilustra un grafo sencillo compuesto por vértices y aristas que los vinculan.

Figura 1 - Grafo

Notamos que la representación gráfica de un grafo definido como ***G = (V, A)***, se compone de un conjunto ***V = {a, b, c, d}***, el cual representa los vértices o nodos del grafo y por el conjunto ***A = {a_1, a^2, a^3, a^4, a^5, a^6}***, el cual representa las aristas que establecen las relaciones existentes entre los vértices del grafo.

Los grafos tienen como objetivo fundamental modelizar un problema específico a través de un modelo abstracto que establezca elementos que participan en el problema (vértices) y las relaciones que pueden existir entre estos participantes (arcos).

Lo primero a resaltar en este aspecto es que los grafos son estructuras abstractas, o sea, que no existen realmente sino que solo sirven como una modelización virtual de un problema real, para su tratamiento computarizado los grafos requerirán de una representación computacional, la cual convertirá esta estructura abstracta en un almacenamiento concreto representado a través de alguna de las representaciones computacionales existentes.

En el enfoque matemático, se considera a los vértices como elementos de datos puros, por ejemplo una letra, un número o un valor atómico específico. En las ciencias de la computación, a estos vértices, no se los considera como valores absolutos atómicos y únicos, sino que por el contrario se los considera como valores que pueden ser compuestos por muchos valores absolutos, formando así una estructura de datos en si misma, es por ello que en estas ciencias a los vértices del grafo se los denomina ***nodos***, entendiendo que estos nodos pueden ser un único valor en cuyo caso hablamos de un nodo simple, o un conjunto de valores que caracteriza a cada uno de los elementos a ser representados en el problema a modelar, en estas situaciones hablamos de nodos compuestos.

Por ejemplo, asumiendo que utilizamos un grafo para representar las relaciones de amistad que existen en un grupo de alumnos de un curso, debemos considerar que cada uno de los alumnos que componen el curso será un nodo del grafo y que las relaciones entre dichos nodos estarán dadas por la relación de amistad que existe entre ellos. En este caso cada uno de los nodos del grafo representa un individuo que compone el curso, el cual posee una serie de características propias y no solo un valor que lo represente, se puede establecer por ejemplo características como nombre, edad, nacionalidad, promedio de notas, etc., cada una de estas características es relevante a la hora de evaluar el grafo, sin embargo todas juntas componen el nodo que representa al alumno en el problema, en este caso hablamos de un nodo compuesto.

En función del tipo de relación que intente modelar un grafo se pueden tener grafos con arcos dirigidos o sea con dirección específica o no, esto ocurre cuando se intenta modelar una relación que tiene jerarquía por ejemplo *"ser mayor que"*, en esta relación el primer elemento es mayor que el segundo en este caso el arco debe tener un sentido con un origen y un destino definido,

donde el origen es el vértice con valor mayor que el destino. Si por el contrario el grafo modela una relación no jerárquica como *"ser igual que"*, pierde sentido la dirección del arco y no se requiere, dado que el origen y el destino son iguales, por lo cual el sentido de la relación no tiene importancia. Los grafos que están conformados por aristas dirigidas se denominan **grafos dirigidos**, mientras que los que no requieren identificar sentido a la relación se denominan **grafos no dirigidos**.

Figura 2 – Grafo no dirigido

En la figura 2 se representa un grafo no dirigido, donde las aristas no tienen un sentido determinado, de forma tal que no se puede establecer ciertamente cual es el origen y cual es el destino de cada una de las aristas que establecen relaciones entre los vértices, por ello, en este tipo de grafos con arcos no dirigidos a dichos arcos se los considera bidireccionales, o sea, con origen y destino en cada uno de los dos vértices incidentes al arco. Considerando este ejemplo la arista a_3 se la considera con origen en *a* y destino en *b* y también con origen en *b* y con destino en *a*.

Figura 3 – Grafo dirigido

En la figura 3 se representa un grafo dirigido, donde las aristas si tienen un sentido determinado, de forma tal que no son bidireccionales y queda establecido claramente el origen y el destino de cada arco.

En la gestión y almacenamiento de datos, donde utilizaremos los grafos para modelar problemas del mundo real, es mucho más frecuente la utilización de grafos dirigidos que de grafos no dirigidos, esto se debe a que la mayoría de relaciones que se establecen mantienen una jerarquía de origen y destino que es necesario identificar en el modelado del problema.

Otra característica que se evalúa en los grafos es el ***grado*** de los nodos, este grado puede ser positivo o negativo, se conoce como grado positivo a la cantidad de arcos o flechas que salen de ese nodo, por otro lado el grado negativo es la cantidad de arcos o flechas que llegan a dicho nodo.

Caracterización de Grafos

En función de las características diferenciales de los arcos que representan la relación que se modela dentro de un grafo podemos hablar de diferentes grafos característicos los cuales analizamos a continuación:

Grafo libre: es aquel grafo que representado como G = (V, A) donde V representa el conjunto de Vértices y A representa el conjunto de arcos, A es un

conjunto vacío. De esta forma un grafo libre es el grafo en el cual no existen arcos, o sea, que todos los vértices son aislados.

Figura 4 – Grafo libre

Como puede observase en la figura 4 donde se representa un *grafo libre*, los vértices *V = {a, b, c, d}*, que conforman el grafo no poseen ningún arco que los conecte, dado que el conjunto de arcos es *A = {}*.

Grafo completo: es aquel grafo que representado como G = (V, A) donde V representa el conjunto de Vértices y A representa el conjunto de arcos, A es un conjunto completo, o sea, que contiene todos los arcos posibles. De esta forma un grafo completo es el grafo en el cual cada vértice está conectado a todos los vértices que componen el grafo, incluido el mismo.

Figura 5 – Grafo completo

Como puede observase en la figura anterior, todos los vértices $V = \{a, b, c, d\}$, que conforman el grafo están conectados entre si, incluidos ellos mismos generando todas las combinaciones de arcos posibles o sea nueve relaciones tres por cada uno de los vértices para conectarse de todas las formas posibles con los restantes vértices, representadas por $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9\}$, a este tipo de grafos se los denomina completos.

Grafo regular: un grafo es regular de determinado grado ***g***, si cada vértice tiene grado ***g***, o sea, que todos los vértices tienen el mismo grado ***g***.

Figura 6 – Grafo regular

Como se explicó anteriormente, el grado es la cantidad de arcos que salen de un vértice (grado positivo), o la cantidad de arcos que llegan a un vértice (grado negativo), en la figura anterior se observa que el vértice *a* tiene dos arcos que salen de él a_1 y a_2, por otro lado el vértice *b* tiene dos arcos que salen de él a_5 y a_7, el vértice *c* tiene los arcos a_3 y a_6 de salida y por último el vértice *d* tiene dos arcos que salen de él a_4 y a_8, de esta forma se denota que todos los vértices que componen el grafo tienen el mismo grado positivo (g=2), a este tipo de grafos se los conoce como regulares, dado que todos los vértices tienen el mismo grado g.

Grafo simple: un grafo es simple si a lo sumo un arco une dos vértices cualesquiera, esto es, que existe solo una arista que une a dos vértices específicos.

Figura 7 – Grafo simple

Como vemos en la figura anterior, solo la arista a_1 vincula al vértice *a* con el vértice *b*, como así también solo la arista a_2 vincula al vértice *a* con el vértice *c* y solo la arista a_3 vincula al vértice *b* con el vértice *c*.

Grafo complejo: en forma inversa a un grafo simple un grafo complejo es aquel donde puede existir más de un arco que vincule dos vértices cualesquiera, por ello se considera que cualquier grafo que no cumpla con la condición de ser simple se considera complejo.

Figura 8 – Grafo complejo

En la figura 8 vemos que a diferencia de lo que ocurría en la figura 7, la arista a_1 y a_2 vinculan al vértice *a* con el vértice *b*, lo cual transforma al grafo en

complejo por no cumplir la regla que solo una arista vincula a dos vértices cualesquiera.

Grafo conexo: un grafo se considera conexo si todo par de vértices esta conectado por un camino, o sea, si para cualquiera par de vértices existe al menos un camino posible entre ellos, o dicho de otra forma que existe al menos una conexión entre todos los nodos que conforman el grafo, sea esta directa (a través de un arco entre ambos) o indirecta (a través de más de un arco entre ambos).

Figura 9 – Grafo conexo

Como se ve en la figura 9 para cualquier par de vértices existe al menos un camino posible entre ambos, o sea, que todos los vértices al menos están vinculados con otro vértice lo cual permite que exista al menos un camino posible entre cualquier par de vértices.

Grafo no conexo: se considera no conexo a un grafo donde un grupo de vértices no esta conectado con el resto de los vértices, o sea, cualquier grafo que no cumpla con la condición de ser conexo se considera no conexo.

Figura 10 – Grafo no conexo

A diferencia de la figura 9, en la figura 10, vemos que no es posible establecer al menos un camino, entre los vértices *a, b* o *c*, con los vértices *d* y *e*, encontrándose estos dos aislados del resto de los vértices que conforman el grafo, aunque ellos estén vinculados entre si a través de un arco.

Grafo complementario: dado un grafo G = (V, A), donde V representa el conjunto de vértices y A el conjunto de arcos, el grafo complementario denominado G^c, es aquel que esta compuesto por los mismos vértices que G y el conjunto de aristas son todas aquellas que le faltan a G para ser un grafo completo.

Figura 12 – Grafo G^c

La figura 11 muestra un grafo definido como G = (V, A), compuesto por el conjunto de vértices *V = {a, b,c}*, por otra parte la figura 12 muestra un grafo definido como G^c = (V, A^c), como el grafo complementario a G, o sea, aquel que contiene los mismos vértices (conjunto V) que G y los arcos que le faltan a G para ser un grafo completo conjunto A^c.

Figura 11 – Grafo G

Representación Computacional de Grafos

Como se desarrolló anteriormente, los grafos son estructuras abstractas, esto indica que son una abstracción de la realidad que para poder ser implementados computacionalmente requieren de alguna representación computacional, estos es, alguna forma que pueda ser representada en una computadora, dado que no es posible representar computacionalmente vértices y arcos.

En función de la forma y la dinámica que mantengan las representaciones computacionales pueden ser de dos tipos: ***dinámicas*** o ***estáticas***.

Las representaciones computacionales dinámicas se caracterizan por acompañar la dinámica del grafo, esto es que el espacio utilizado por la representación va cambiando en función de cómo va cambiando el grafo, mientras que en las representaciones computacionales estáticas, se establece un espacio fijo, el cual no cambia en función de la dinámica establecida por las altas y bajas del grafo, sino que contempla todas las posibilidades de relaciones posibles entre todos los vértices del grafo, existan o no dichas relaciones.

En función de su comportamiento, a primera instancia se podría pensar que las representaciones dinámicas ocupan menos espacio que las representaciones estáticas, estos es así en determinadas oportunidades pero no siempre.

Para implementar una representación computacional dinámica es necesario la utilización de punteros que vinculan las diferentes posiciones de memoria que representan los vértices y los arcos o relaciones involucrados en el grafo, de esta forma el espacio ocupado se compone por el espacio de representación de vértices y arcos, más el espacio adicional necesario para mantener los punteros de memoria que los vinculan. Las representaciones computacionales estáticas se construyen sobre estructuras computacionales rígidas que utilizan el concepto de contigüidad como los vectores y matrices, o sea, que el siguiente vértice este a la derecha y el anterior a la izquierda de un vértice determinado, ocurriendo lo mismo con los arcos, por ello debe contemplar la existencia de todas las relaciones posibles entre todos los vértices existentes, de esta forma se utiliza espacio para representar dichas relaciones existan o no, pero por otra parte ocupan menos espacio al no requerir punteros adicionales para vincular

los arcos y los vértices entre si.

En función de lo expuesto, las representaciones computacionales dinámicas no siempre ocupan menos espacio que las representaciones computacionales estáticas, esto depende de la cantidad de relaciones o arcos con que cuente el grafo en función de todas las relaciones posibles, de esta forma, cuanto más arcos posea un grafo, menor será el espacio desperdiciado por las representaciones estáticas y mayor será el desperdicio representado por los punteros en las representaciones computacionales dinámicas.

En cuanto a la velocidad de operación, las representaciones computacionales estáticas tienden a ser más rápidas que las dinámicas, esto se debe a que para implementar nuevas relaciones no requieren realizar ningún alta en la representación, debido a que a priori el espacio ya esta dimensionado para la existencia de todas las relaciones posibles, con lo cual se ahorra el tiempo involucrado en el pedido y

asignación de memoria para almacenar dicha relación que requiere la representación dinámica. Sin embargo si la representación computacional estática es muy grande, está velocidad de acceso disminuye por el gran espacio asignado y la dificultad de acceder contiguamente a una posición determinada, siendo a veces más lenta que las representaciones computaciones dinámicas.

Por lo visto se observa que ambos tipos de representaciones son válidos para ser utilizados, normalmente se recomienda la utilización de representaciones computacionales dinámicas para grafos grandes, con gran cantidad de vértices, lo cual generaría en una representación estática un alto espacio ocupado, o también para grafos ***dispersos***, esto es, grafos con muchos nodos pero pocas relaciones entre ellos, mientras que las representaciones computacionales estáticas se recomiendan para grafos más pequeños donde pueda utilizarse la potencia de la velocidad de acceso de la representación estática, o también para grafos ***densos***, donde existe una gran cantidad de relaciones definidas dentro de todas las posibles.

A continuación se desarrollaran en detalle algunas de las estructuras existentes para la representación computacional de grafos tanto dinámica como estáticamente.

Representaciones computaciones
estáticas

Las representaciones computacionales estáticas se denominan de esta forma debido a que ocupan un espacio fijo independientemente de la dinámica del grafo a representar, o sea, que la representación computacional no va cambiando en función de las altas y bajas que se van produciendo en el grafo, dado que el espacio computacional ocupado contempla todas las relaciones posibles entre todos los nodos existentes como si el grafo que fuera a representar siempre fuera completo. Por ello, si se elimina un arco no afecta en el espacio ocupado por la estructura, sino que solo se anula la asignación de la posición determinada por la relación.

Debido a su comportamiento las representaciones computacionales estáticas se basan en matrices, donde la vinculación entre los nodos y los arcos se realiza en función a la contigüidad de los mismos, sin la necesidad de utilizar punteros de vinculación entre los diferentes arcos o vértices del grafo.

Dentro de las representaciones computacionales estáticas encontramos la ***Matriz de adyacencia*** y la ***Matriz de incidencia***, las cuales desarrollaremos a continuación.

Matriz de adyacencia

Si consideramos un grafo G = (V, A) con n vértices la matriz de adyacencia es aquella de dimensión MA_{nxn} con ***n*** como la cantidad de vértices, donde la

posición MA_{ij} es el número de aristas que unen los vértices V_i y V_j.

1
1

Si el grafo que se esta representando es un grafo no dirigido puede existir valores 0,1 y 2 en la matriz, si por el contrario el grafo que se representa es

dirigido solo puede haber valores 0 y 1, debido a que las aristas representadas tienen origen en la fila y destino en la columna, mientras que en el caso de los grafos no dirigidos deben ser consideradas bidireccionales, o sea, con origen y destino en cada uno de los vértices.

Si el grafo representado es no dirigido la matriz es simétrica, debido a que si se representa una relación entre el vértice *a* y el vértice *b* representado con un *1* en la intersección de la *fila a* y la *columna b*, debido a la bidireccionalidad de los arcos no dirigidos debe estar también la representación del arco con un *1* en la intersección de la *fila b* y la *columna a*, marcando la simetría de la matriz.

Figura 13 – Grafo y su Matriz de Adyacencia

En la figura 13 se representa un grafo y la matriz de adyacencia requerida para representar computacionalmente en forma estática al mismo, como vemos dicha matriz es cuadrada, o sea, de una dimensión *nxn*, siendo *n* la cantidad de vértices del grafo, en el caso del grafo de la figura *4x4*.

En la figura se representa con un 2 la intersección entre el vértice a y si mismo, esto se debe a que como el grafo es no dirigido todas las aristas son consideradas bidireccionales, de forma tal que son dos las aristas que tienen origen en el vértice *a* y destino en el vértice *a*, esta es la diferencia que se produce en la representación con los grados dirigidos donde no puede haber 2 en la matriz.

Si un vértice es aislado la fila y columna que le corresponden esta compuesta solo por ceros, dado que no le llegan ni le salen arcos.

Por otra parte si un grafo es simple entonces la matriz de adyacencia contiene solo ceros y unos convirtiéndose en una matriz booleana o binaria y con la característica particular que la diagonal esta compuesta sólo por ceros.

Si se representa un grafo dirigido la matriz de adyacencia no es simétrica, dado que un arco tiene un origen y un destino identificado y cada arco no se

considera doble como en los grafos no dirigidos.

En este tipo de grafos la matriz siempre es binaria o booleana. También mantiene la característica que el número de unos que aparecen en una fila es igual al grado de salida del correspondiente vértice y el número de unos que aparecen en una determinada columna es igual al grado de entrada del correspondiente vértice.

1
2

Matriz de incidencia

Si se considera un grafo G = (V, A) donde V representa los ***n*** vértices y A los ***m*** arcos que componen al grafo, su matriz incidencia es la matriz de orden ***nxm***, MI_{nxm}, donde MI_{ij} es 1 si V_i es incidente con A_j y MI_{ij} es 0 en caso contrario.

Al igual que en la matriz de adyacencia si el grafo es dirigido la matriz es binaria o booleanas, si en cambio el grafo es no dirigido la matriz puede contener un 2 para el caso de los bucles, siendo solo binaria para el caso de los grafos simples o grafos que no contengan bucles.

Figura 14 – Grafo y su Matriz de Incidencia

En la figura 14 se representa un grafo y la matriz de incidencia, definiendo una matriz con tantas filas como vértices conformen el grafo y tantas columnas con

arcos lo conformen, en el caso del ejemplo nos encontramos con una matriz de 4x6, dado que el grafo cuenta con 4 vértices *(a, b, c, d)* y 6 arcos *(a_1, a^2, a^3, a^4, a^5, a^6)*.

La sumatoria de los números que aparece en cada fila es igual al grado del vértice correspondiente, por otro lado si una fila esta compuesta sólo por ceros corresponde a un vértice aislado.

Representaciones computacionales
Dinámicas

Las representaciones computacionales dinámicas se denominan de esta forma debido a que acompañan la dinámica del grafo a representar, o sea, que la representación va cambiando en función de las altas y bajas que se van produciendo en el grafo y el espacio computacional ocupado depende exactamente de la cantidad de vértices y arcos que compongan en cada instante el grafo. De esta forma, si se elimina un vértice o un arco, también es eliminado de la representación computacional disminuyendo el espacio ocupado.

Debido a su comportamiento las representaciones computacionales dinámicas se basan en nodos de memoria vinculados a través de punteros lo cual permite que los mismos se creen y eliminen en función de la dinámica del grafo.

Dentro de las representaciones computacionales dinámicas encontramos las ***Listas de adyacencia***, las cuales desarrollaremos a continuación.

Listas de adyacencia

Las listas de adyacencia es un tipo de representación que se conforma por una lista que representa los nodos que componen el grafo, donde cada una de los

elementos que componen dicha lista de nodos mantiene otra lista asociada que representa los arcos o relaciones que salen de dicho nodo.

Figura 15 – Grafo G

La figura 15 esquematiza el grafo a ser representado a través de una lista de adyacencia.

1
4

Figura 16– Lista de Adyacencia del Grafo G

En la figura 16 se expone la representación computacional a través de una lista de adyacencia del Grafo G de la figura 15. Como se observa se compone de

una lista expuesta en forma vertical compuesta por cada uno de los vértices que conforman el grafo en este caso 3 *(a, b, c)*, cada uno de los nodos que conforman la lista de vértices tiene tres componentes de datos, uno para la representación del nodo, otro para apuntar al siguiente de la lista y un tercer puntero a la lista de arcos que salen de ese vértice.

Los nodos de cada una de las listas de arcos asociadas a cada vértice están compuestos por dos punteros, uno que apunta al destino del arco con origen en el vértice en cuestión y el otro que apunta al siguiente arco con origen en el mismo vértice.

En particular en este caso vemos que del nodo que representa al vértice *a* sale un puntero al siguiente vértice en este caso *b* y otro puntero que apunta a lista de arcos que salen del vértice *a*, en este caso compuesta solo por un arco *a1* que tiene como destino el mismo nodo *a*.

El espacio de almacenamiento ocupado por esta representación es el utilizado por los nodos, más los punteros que los vinculan en la lista, como así también los arcos existentes conjuntamente con los punteros que los vinculan en las diferentes listas de arcos de cada nodo.

Es importante destacar que para los grafos no dirigidos el arco está representado dos veces una vez en la lista de uno de los vértices que lo conforman y otra vez en el otro vértice, esto se debe a que todo arco en un grafo no dirigido se lo considera de ida vuelta o sea doble o bidireccional.

Clasificación de Grafos

Como se fue describiendo en el presente capítulo, los grafos tienen una gran cantidad de caracterizaciones, que hace que se pueda clasificar a los mismos según diferentes características.

En cuanto a la clasificación formal de los grafos, nos concentraremos en este libro en las clasificaciones más relevantes desde el punto de vista del almacenamiento y administración de los grafos, en este contexto los grafos pueden clasificarse según dos criterios significativos.

En primer lugar, los grafos se clasifican en función de si los arcos que lo componen poseen o no dirección clasificándose en:

***Grafos dirigidos*:** son aquellos en los cuales los arcos que vinculan a los vértices tienen una dirección definida, o sea, son arcos con sentido, dicho sentido o dirección marca una jerarquía en la relación que se esta modelando, donde cada arco tiene un vértice de origen y otro vértice destino de la misma,

este tipo de relaciones son jerárquicas y resultan ser la mayoría de utilizaciones de los grafos, dentro de estas relaciones encontramos por ejemplo, *"ser mayor que"*, *"ser menor que"*, *"ser padre de"*, *"ser componente de"*, etc.

Grafos no dirigidos**:** son aquellos donde los arcos no tienen una dirección o sentido definido, o sea, donde la relación no establece jerarquía de forma tal que es irrelevante quien es el origen y quien el destino de la relación. Esta situación solo se da en relaciones simétricas donde el arco realmente representa una relación doble con origen y destino en cada uno de los vértices. Este tipo de relaciones son las menos utilizadas y dentro de ellas encontramos por ejemplo, *"ser igual a"*, *"ser hermano de"*, *"ser cónyuge de"*, etc. Como puede observase en todas estas relaciones es irrelevante el orden de evaluación de los vértices que conforman la relación.

En segundo lugar, los grafos se clasifican en función de las restricciones que pueden ser aplicadas a las relaciones que modelan, los grafos se clasifican en:

Grafos restrictos**:** son aquellos grafos en los cuales la relación que se modela **no debe cumplir** las propiedades de reflexividad, simetría y transitividad. De esta forma al modelar una relación nos garantizamos que el grafo no contiene bucles, porque debido a que debe ser anti-reflexivos ningún vértice está relacionado consigo mismo, tampoco pueden existir ciclos simples, dado que al ser anti-simétricos un vértice no puede relacionarse con otro y este a su vez con el primero y por último al ser anti-transitivo se disminuyen los diferentes caminos posibles entre cualquier par de nodos. Por todo ello los grafos restrictos son más fáciles de administrar y operar, es por ello que los diseñadores que utilizan grafos para modelar problemas tratan de restringir lo más posible las relaciones que se representan a través de ellos.

Grafos irrestrictos**:** son aquellos grafos en los cuales no se aplica ninguna restricción a la relación que se modela, pudiendo o no modelar relaciones que sean reflexivas, simétricas o transitivas. Estos grafos son los más complicados de administrar, esto se debe a que como no existen restricciones sobre las

relaciones entre los nodos que lo conforman, pueden darse situaciones donde se creen ciclos, lo cual suele generar más complicaciones algorítmicas a la hora de evaluar propiedades, pasos, caminos, etc., por la situación que se debe llevar registro de que nodo fue visitado debido a que un arco puede llevarnos de nuevo al mismo nodo del que se partió o a un nodo que ya fue visitado anteriormente, de modo que si no se controlará dicha situación generaría un ciclo infinito en el algoritmo.

Cabe señalar que dentro de las ciencias de la computación los grafos más utilizados son los grafos dirigidos, debido a que la mayoría de las relaciones que se representan poseen algún tipo de jerarquía, donde es relevante el origen y el destino de cada arco o relación.

En los que respecta a los grafos restrictos, son el inicio de lo que se conoce como estructuras básicas de datos, concepto que profundizaremos en el capítulo siguiente.

Caminos, pasos y ciclos en Grafos

Un **camino** entre dos nodos *a* y *b* se establece cuando existe una vinculación directa o indirecta entre ambos, esto es cuando se pueden vincular entre sí mediante uno o más arcos. Dado que este concepto no está vinculado con la dirección de los arcos, puede existir camino tanto en grafos dirigidos como en grafos no dirigidos, considerando que en los grafos dirigidos existe camino entre dos vértices cuando existe vinculación entre ellos, independientemente del sentido de los arcos.

Un **paso** entre dos nodos *a* y *b* se produce cuando existe un camino entre ambos pero con un sentido preestablecido, esto es que partiendo del nodo *a* y siguiendo el sentido de los arcos se llega al nodo *b*. Como en este caso es relevante el sentido, solo se evalúan pasos en los grafos dirigidos, dado que en los grafos no dirigidos todos los arcos se consideran bidireccionales con lo cual el concepto de paso se iguala al de camino. Un **ciclo** entre dos nodos *a* y *b* es un paso o un camino donde el origen y el destino son iguales, esto es, el vértice de inicio y el vértice de destino son iguales, pudiendo estar compuesto el ciclo por uno o más arcos.

Figura 17– Caminos, pasos, y ciclos

En la figura 17 se ven representados los conceptos detallados, en dicho grafo se pueden observar dos pasos entre el vértice *a* y el vértice *b*, un paso conformado solamente por el arco a_1 y otro paso conformado por el arco a_4 y a_3. También se ve representado un ciclo establecido por el arco a_4 y a_5. Se observan varios caminos entre vértices, como por ejemplo un camino entre el vértice a y el b conformado por las arcos a_5 y a_3. En función de lo descripto, se desprende que el concepto de camino está asociado con los grafos no dirigidos y el concepto de paso con los grafos dirigidos, aunque también es posible evaluar caminos en grafos dirigidos y de hecho se suele hacer cuando se desee evaluar si dos nodos están vinculados independientemente si de uno de ellos puedo llegar al otro siguiendo el sentido de los arcos.

Como se desprende de los conceptos vertidos, en un grafo pueden existir muchos caminos, pasos o ciclos entre dos nodos, esto genera la necesidad de medir las diferentes longitudes de cada uno. De esta forma, vamos a definir

longitud de paso, camino o

ciclo como la cantidad de arcos involucrados en los mismos o la cantidad de vértices involucrados menos 1.

Como caso particular definimos el **bucle** como el ciclo compuesto por un solo arco que tiene el mismo vértice como origen y destino (representado en la figura 17 por los arcos a_6 y a^2). En este caso en particular se define la longitud del bucle como un ciclo de longitud 0, dado que se da una diferencia entre las dos opciones para medir la longitud, si contamos la cantidad de arcos la longitud es 1 y si tomamos la cantidad de vértices menos 1 la longitud es 0, por eso por convención se establece que la longitud es 0.

Búsqueda en Grafos

Para efectuar una búsqueda de un camino o un paso en un grafo existen dos métodos o técnicas distintas, que se diferencian en la forma en que realizan el recorrido del grafo para identificar el paso o el camino según corresponda.

El objetivo es identificar la existencia o no de un paso o camino entre dos nodos cualesquiera de un grafo, de forma tal de poder establecer que partiendo de un determinado vértice ***a***, se puede acceder a través de uno o más arcos a un destino determinado como puede ser el vértice ***b***, de esta forma se establece que existe paso o camino dependiendo que se tenga en cuenta o no el sentido de los arcos, con origen en el vértice ***a*** y destino en el vértice ***b*** y la longitud del mismo queda establecida por los arcos que lo componen.

Búsqueda en profundidad (Deepth First)

Esta técnica se caracteriza por avanzar en profundidad, esto es, sin mantener un orden jerárquico de evaluación, de forma tal la técnica avanza hasta el momento que no puede avanzar más y ahí retrocede para tomar otra relación y seguir avanzando.

El método comienza en el nodo origen del paso o camino y avanza al primer vértice que está conectado con él, verificando si ese vértice es quien se está buscando como destino, de no ser así se traslada a este nuevo nodo a través del arco utilizado y vuelve a realizar la misma operación comparando con el nuevo destino, esta operación la realiza hasta que no puede avanzar más por no existir arcos que salgan del nodo en que se encuentra, en ese momento regresa al nodo anterior que lo llevó hasta allí y repite la operación realizada tomando el primer arco y verificando si el destino de dicho arco es el mismo que se esta buscando.

Está operación se realiza hasta que se encuentra el destino buscado o cuando no hay más arcos que utilizar, en cuyo caso se puede afirmar que no existe paso o camino entre los nodos propuestos.

Esta técnica se utiliza cuando necesitamos encontrar respuesta a un problema sobre un grafo sin condiciones de optimización, esto se debe a que de existir varios pasos o caminos entre los dos nodos esta técnica no garantiza que encuentra el más corto u óptimo, sino que encuentra uno que puede ser o no el mejor.

Este tipo de técnica se basa en la recursividad, debido a que va trasladando el problema al vecino más cercano, dejando pendiente la instancia anterior, la cual es almacenada en la pila de procesos pendientes, hasta el momento que no puede avanzar por no contar con nuevos vecinos y regresa al último vecino pendiente en la pila de procesos y pasa a evaluar la siguiente relación si existiese.

Para evitar que el algoritmo pueda entrar en ciclo infinito generado porque las relaciones lo retornan a un nodo ya visitado, se debe verificar que el nodo al cual se le

pasará la búsqueda no hay sido ya evaluado, en cuyo caso se pasa a la siguiente relación para evitar entrar en un ciclo infinito.

Figura 18– Búsqueda de paso en profundidad

Como se indico este método partiendo del vértice origen del paso buscado evalúa el destino del primer arco que tiene origen en este vértice coincide con el buscado, si coincide, se retorna la existencia del paso, de lo contrario, se traslada la búsqueda a este nuevo nodo y se repite la operación hasta encontrar el paso buscado o hasta que no haya más vértices para procesar.

Supongamos que contando con el grafo de la figura 18 se desea conocer si existe un paso entre los vértices *d* y *c*. Si aplicamos el método primero en profundidad iniciando en el vértice *d*, habría que evaluar los arcos que tienen

origen en *d* (a_4 y a_8), de estos arcos se observa si el destino del primer arco en este caso *a* es igual al destino buscado en nuestro caso *c*, como no lo es, se traslada la búsqueda a este nuevo vértice destino del arco a_4 que es *a*.

Desde este nuevo vértice se realiza la misma operación obteniendo los arcos que tienen origen en el vértice *a* (a_1 y a_2) analizando si el destino del primero de ellos es igual al destino del paso buscado, como no lo es se repite la operación, en este caso al trasladar la búsqueda al destino del primer arco (a_1) se vuelve a un vértice procesado, por lo cual se descarta y se pasa al destino del arco siguiente (a_2), como el destino de este arco (*b*) tampoco es el buscado (*c*), se repite la operación evaluando los arcos con origen en el nuevo vértice (*b*). Aquí encontramos el arco a_5 y a_7, cuando se evalúa el destino del arco a_5 se obtiene el mismo destino buscado con lo cual se encuentra un paso entre *d* y *c*.

Búsqueda en anchura (Breath
First)

A diferencia de la búsqueda en profundidad, la búsqueda en anchura, evalúa primero todos los destinos de todos los arcos que parten del vértice origen del paso o

2
1

camino a evaluar, de forma tal de evaluar primero todos los destinos directos antes de pasar al siguiente.

De esta forma esta técnica no requiere retroceder, debido a que a priori verifica cada uno de los destinos de los arcos que establecen relaciones directas con él guardándolos como futuros nodos a procesar.

Una vez verificados todos los vértices vecinos directos del origen y siempre y cuando ninguno de ellos coincida con el destino buscado se almacenan en una estructura auxiliar los diferentes nodos conectados directamente con el origen de forma tal de llevar el orden para su futuro procesamiento.

Considerando que para este método es importante el orden de evaluación de los vecinos, debido a que primero evalúa primero todos los vecinos directos al origen, la estructura que debe almacenar los nodos a procesar debe contemplar el concepto de orden, por ellos esta estructura es una cola, la cual mantiene un orden de llegada y de salida, de forma tal de procesar primero a quien primero se insertó en dicha estructura y último al último ingresado al igual que ocurre en una caja de un Banco o de un Supermercado.

Es así que una vez que se cargaron en cola todos los nodos vecinos directos a procesar, se continúa trasladando la búsqueda al primer elemento a procesar de la cola, realizando el mismo proceso e ingresando ahora en la cola todos los nodos relacionados directamente con este nodo al final de la estructura.

El método finaliza cuando se encuentra el paso buscado o cuando no quedan más nodos para procesar en la cola, dado que la cola tiene un extremo donde se ingresan los nodos y otro extremo desde donde se extraen, se garantiza que primero se analizan todos los destinos de los arcos que se relacionan directamente con el origen y no se pasa a los vértices vinculados indirectamente con el nodo original, sin primero haber evaluado todos los vecinos conectados directamente.

Esta técnica se utiliza para resolver problemas en los que se pide hallar una solución óptima entre varias posibles, dado que el paso encontrado como solución es el más corto porque como se ve, la técnica va buscando primero pasos de longitud uno, después de longitud dos, después tres y así sucesivamente.

Figura 19– Búsqueda de paso en ancho

Supongamos que contando con el grafo de la figura 19 se desea conocer si existe un paso entre los vértices *d* y *c*. Si aplicamos el método primero en ancho iniciando en el vértice *d*, habría que evaluar los arcos que tienen origen en *d* (a_4 y a_8), de estos arcos se observa si el destino de cada uno de ellos (*a* y *b*) es igual al destino buscado en nuestro caso *c*, como no lo es, se traslada la búsqueda al vértice destino arco a_4 que es *a*, manteniendo en la lista de vértices pendientes a *b* que es el destino del arco a_8.

Desde este nuevo vértice se realiza la misma operación obteniendo los arcos que tienen origen en el vértice *a* (a_1 y a_2) analizando si el destino de cada uno de ellos (a y b) es igual al destino del paso buscado, como no lo es se agregan a la lista de nodos a procesar los nuevos vértices en este caso ambos se descartan porque el vértice *a* ya se proceso y el vértice *b* ya existe en la lista de nodos a procesar.

Al evaluar los vértices pendientes de procesamiento se toma el vértice *b* y se evalúan los arcos que tienen origen en él (a_5 y a_7), en este caso al evaluar los destinos de estos arcos se detecta que el destino del arco a_5 es el vértice *c* que es el destino buscado por lo cual se da por finalizada la búsqueda informando la existencia del paso.

2
3

Caminos mínimos en Grafos Dirigidos

Una de las situaciones más comunes a resolver es desde un origen determinado evaluar de todos los caminos o pasos posibles a un destino, extractar aquel que cumple la condición de ser el mínimo.

Esto ocurre cuando los arcos tienen asociado un atributo que indica el largo o la distancia del mismo, donde no es lo mismo un arco que otro, porque tienen asociado un peso diferencial en distancia, de esta forma la longitud de un camino entre dos nodos será la suma de las distancias de cada uno de los

arcos que los unen, de manera que el objetivo buscado es encontrar aquel camino cuya suma de distancias sea la menor.

El análisis de este problema tiene dos ópticas, una es encontrar el camino con distancia mínima desde un nodo origen determinado a otro nodo destino, otra óptica es encontrar todos los caminos con distancias mínimas para todos los pares de nodos cualesquiera del grafo establecidos como origen y destino.

En función de que óptica se considere existen diferentes técnicas o algoritmos para encontrar la solución buscada, a continuación describiremos cada una de ellas.

Algoritmo de Dijkstra

El objetivo de este algoritmo es partiendo de un origen conocido, encontrar todos los caminos mínimos con el resto de los nodos del grafo. De este modo está técnica pertenece al grupo de algoritmos que parten de un origen conocido y que desde él encuentran todos los caminos mínimos con los restantes nodos o con uno de ellos en particular.

El algoritmo se sustenta sobre la base de ir armando un conjunto de nodos a los cuales se puede acceder desde el nodo origen de forma óptima, así partiendo del origen el primer conjunto de nodos estará formado por todos los nodos a los cuales se puede acceder desde este origen con mínima distancia.

Luego en forma iterativa el algoritmo agrega caminos a nuevos nodos partiendo de los nodos que componen el conjunto existente, o sea caminos donde todos los nodos existentes deben ser nodos que se encuentren en el conjunto de nodos existentes

Por último se evalúa el nodo que no está en el conjunto cuya camino tentativo es óptimo, en cuyo caso se agrega al conjunto de nodos con camino óptimo.

Esta operación se repite para todos los nodos, de forma tal de ubicar para cada uno de ellos el camino óptimo desde un origen determinado.

A continuación la Figura 20 ejemplifica un grafo dirigido con el peso o distancia de cada arco representado con un valor en el propio arco, sobre el cual analizaremos todos los caminos con origen en el vértice *a* utilizando el algoritmo.

Figura 20 – Algoritmo de Dijkstra

Al aplicar el algoritmo de Dijsktra sobre el grafo de la figura 20, en la primer iteración se conforma el conjunto D = (*b=1*) conformado por todos los nodos a los cuales se puede acceder y las distancias mínimas iniciales desde el nodo *a*.

En la segunda iteración se agrega al conjunto uno de los nodos a los cuales se puede acceder en nuestro caso b y el conjunto de nodos como D = (*b=1*, *c=5*, *d=7*), incluyendo a los nodos *c* y *d*, que no eran alcanzables desde el origen *a* sin el nodo *b*.

En la próxima iteración se agregan los camino desde los nodos c y d, quedando conformado el conjunto como D = (*b=1*, *c=5*, *d=7*), a esta altura ya se analizaron todos los caminos posibles desde el origen definido en nuestro caso a, dado que no pudo disminuirse las distancias obtenidas, dejando en el conjunto los nodos intermedios desde los cuales se acceden a los caminos mínimos a todos los vértices posibles con origen en *a*.

Algoritmo de Floyd-Warshall de Clausura transitiva

A diferencia del algoritmo anterior, el Algoritmo de Floyd-Warshall no parte de

un origen conocido, sino que busca todos los caminos posibles de longitud mínima con origen y destino en cada uno de los nodos que componen el grafo.

El algoritmo comienza con la creación de una matriz de pesos de caminos directos para cada uno de los nodos que denominaremos P_{nxn}, donde se encuentran las distancias de acceder en forma directa desde cada nodo a cada uno de los restantes, si no se puede acceder de un determinado nodo a algún otro el valor que se coloca en la matriz es infinito, como se observa la matriz es cuadrada y tiene como dimensión la cantidad de nodos *n* que conforman el grafo a analizar.

Luego el algoritmo es iterativo, en cada iteración se va agregando un nodo en la evaluación, no solo considerando los accesos directos, sino también pasando por este nodo intermedia, evaluando nuevos posibles caminos para ver si estos son mejores que los encontrados, o sea, si su longitud es menor que la existente, de esta forma en la n- ésima iteración se evaluaron todos los caminos desde cada uno de los nodos a todos los restantes, quedando representados en la matriz los caminos mínimos a este es momento.

2
5

Para analizar este algoritmo utilizaremos el mismo grafo de la figura 20 utilizado para evaluar el algoritmo de Dijkstra.

La matriz inicial que se crea en función de los caminos mínimos directos queda conformada como se representa a continuación:

Figura 21 – Matriz de distancias de caminos directos de Floyd-Warshall

Aquí vemos que el peso de acceder directamente de cada uno de los nodos

con los restantes es el que se expresa en la matriz y para el caso de no existir caminos se representa con infinito *"inf"*.

En la primer iteración del algoritmo se agrega como nodo intermedio al primer nodo, o sea, aparte de considerar los pasos directos también se consideran los indirectos pasando por el nodo *a* que en nuestro caso es el primero modificando el estado de la Matriz de caminos mínimos como queda representado en la figura 22.

Figura 22 – Matriz de distancias de Caminos iteración 1 agregando el nodo *a* como intermedio.

En esta figura marcamos en rojo los nuevos caminos que se pueden acceder con las distancias mínimas considerando como nodo intermedio al nodo *a*.

Como vemos se incorporan dos caminos uno entre el nodo *c* y el nodo *b* con una distancia de peso 4 conformada por 3 que es el peso del paso entre el nodo *c* y el nodo *a* y 1 que es el peso del paso entre el nodo *a* y el nodo *b*. Del mismo modo se incorpora el paso entre el nodo *d* y el *c* con peso 3 formado por 2 que es el peso del paso entre el nodo *d* y el nodo *a* y 1 que es el peso del paso entre el nodo *a* y el nodo *b*.

En las siguientes iteraciones se van incorporando los siguientes nodos como nuevos nodos como intermedios hasta llegar a incorporar todas las combinaciones posibles lo cual garantiza que en la matriz quedan los caminos mínimos que existen entre todos los nodos de un grafo determinado.

Las siguientes figuras van mostrando los diferentes cambios que se van produciendo en la matriz en función de cada uno de los nodos que se va a agregando en el análisis de los caminos mínimos.

En la segunda iteración agregando a la evaluación el nodo b, nos queda:

Figura 23 – Matriz de distancias de Caminos iteración 2 agregando el nodo *a* y *b* como intermedios.

En este caso se incorporan dos caminos, uno de peso 5 entre el nodo *a* y el nodo *c* conformado por 1 que es el peso del paso entre el nodo *a* y el nodo *b* y 4 que es el peso del paso entre el nodo *b* y el nodo *c* y otro camino de peso 7 entre el nodo *a* y el nodo *d*, conformado por 1 que es el peso del paso entre el nodo *a* y el nodo *b* y 6 que es el peso del paso entre el nodo *b* y el nodo *d*.

En la próxima iteración donde se agrega el nodo *c*, se modifica la matriz de distancias de caminos de la siguiente forma:

Figura 24 – Matriz de distancias de Caminos iteración 3 agregando el nodo *a*, *b* y *c* como intermedios.

Aquí se incorpora un camino de peso 7 entre el nodo *b* y el nodo *a*, conformado por 4 que es el peso entre el nodo *b* y el nodo *c* y 3 que es el peso entre el nodo *c* y el nodo *a*.

Por último vemos que la agregar en la evaluación el nodo *d*, no varía el estado obtenido de la matriz de distancias, dado que involucrando en los caminos posibles el nodo *d* no se puede mejorar ninguno de los caminos ya obtenidos, como lo representa la figura 25.

Figura 24 – Matriz de distancias de Caminos iteración 3 agregando el nodo *a*, *b*, *c* y *d* como intermedios.

Aplicación de Grafos

Como ya mencionamos al inicio del capítulo, los grafos son un conjunto de nodos y relaciones entre ellos, de esta forma la mayor utilización de estas estructuras abstractas es para modelar problemas de la vida real, con el objeto de poder simular en estas estructuras una situación real y poder resolver problemas a partir de las técnicas y propiedades que hemos analizado de dichos grafos.

Existen innumerable cantidad de aplicaciones de modelos reales que se resuelven desde la óptima de un grafo, en este apartado enumeraremos algunas aplicaciones prácticas y muy conocidas de la modelización aplicando grafos.

Un ejemplo de aplicación de grafos para la solución de problemas reales es la modelización de una ciudad a través de un grafo dirigido, donde cada una de las esquinas de una ciudad representa un nodo, desde los cuales salen y llegan arcos, los cuales representan las calles que llegan y parten de cada una de las esquinas. Esta modelización permite crear un mapa de la ciudad y de esta forma establecer puntos de interés turístico, posibles formas de llegar de un punto a otro estableciendo *pasos* o *caminos* entre dos puntos determinados, definir recorridos de un punto a otro, como por ejemplo un recorrido en auto, donde el grafo debe ser considerado como un grafo dirigido debido a que se requiere identificar el sentido del arco, el cual representa el sentido de circulación de la calle, otro recorrido podría ser a pie, considerando el grafo como no dirigido dado que el sentido de la calle no es relevante al realizar un recorrido de esta forma. Este ejemplo se va materializado en productos de software que reflejan mapas digitales como muchos de los existentes actualmente en el mercado.

Otra aplicación de los grafos se da en el diseño de redes, sean estas de agua, cloacales, de energía, telefónicas, de datos o hasta de subterráneos o ferrocarriles, sobre una superficie determinada, como puede ser un área geográfica o también una ciudad. En este caso el grafo modela el área con sus diferentes accidentes geográficos, identificando cada uno de ellos y las necesidades de la red a tender de pasar por determinados puntos importantes del área, de esta forma el grafo a partir de la búsqueda de caminos mínimos nos permitirá establecer el diseño de la red de la forma más eficiente disminuyendo los costos de materiales e instalación de la misma.

Por ultimo podemos mencionar como otro ejemplo de aplicación de grafos, lo que se denomina despiece de un elemento, donde muchas industrias establecen a través de un grafo el modelado de la composición de cada uno de sus productos, permitiendo de esta forma establecer que es necesario para fabricar cada uno de ellos. En este caso el grafo estaría formado por nodos que representan productos, como por ejemplo podría ser en una industria automotriz, inyectoras, neumáticos, embragues, butacas, etc., los arcos establecerían las relaciones entre cada uno de estos nodos y los nodos que lo conforman y son requeridos para su fabricación, por ejemplo un auto está compuesto por 5 ruedas (contando el auxilio del mismo), cada rueda a su vez esta compuesta por una llanta, un neumático, cuatro tuercas, etc. De esta

forma ante la falta de determinado producto (representado en el grafo por un nodo), se podría establecer rápidamente todos los productos requeridos para su fabricación o en forma inversa todos los productos que no pueden fabricarse sin este producto.

Como se ve, son muy variadas las aplicaciones de grafos en las ciencias de la computación, utilizadas para la solución de problemas reales, el lector debe tener en cuenta que el objetivo primordial de los grafos es *modelar* una situación específica permitiendo al desarrollador *simplificar la solución del problema* basándose en las propiedades, herramientas y características conocidas del grafo utilizado. Esto permite simplificar la solución algorítmica que es el objetivo final de los datos y las estructuras de datos permitirle al programador que a través de un modelado correcto del problema se simplifique la programación del algoritmo que de la solución buscada.