Ciencias computacionales

Propedéutico: Programación

Contents

1 Grafos

1.1 Vídeos sobre grafos

Introducción a los grafos.
Matriz de Adyacencia.
Algoritmo de Dijkstra.

1.2 Definición y conceptos

Los grafos son utilizados dentro de diversos campos como la ingeniería eléctrica, química, ingeniería industrial, compiladores, sistemas operativos, organización y recuperación de la información entre otras disciplinas. Los grafos son estructuras de datos, es decir, tipos de datos abstractos, comúnmente los grafos son utilizados para el modelado de problemas.

Definición

Un grafo es un conjunto no vacío de objetos o entes físicos que tienen relación entre ellos. Un grafo esta formado por vértices, muchas veces también llamados nodos (siendo estos los que representan a los objetos), y un conjunto de arcos que representan las relaciones entre los vértices, también llamadas aristas o *edges* en inglés.

Por lo tanto, los grafos se representan mediante los vértices conectados por líneas o aristas, estos pueden ser orientados o no orientados. Un grafo G puede ser descrito por el par de vértices y aristas, G= *(V, A)* [1]. En la Fig. 1, se puede observar la relación descrita anteriormente, dónde los vértices son $V = \{1, 4, 5, 7, 9\}$ y el conjunto de aristas $A = \{(1,4), (4,1), (5,1), (1,5), (7,9), (9,7), (7,5), (5,7), (4,9), (9,4)\}$, siendo este grafo no dirigido.

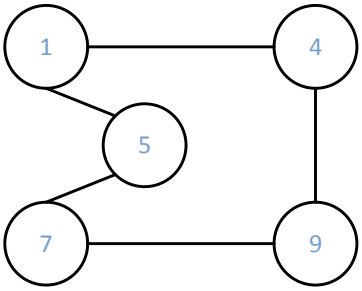

Figure 1: Grafo no dirigido.

Por otro lado también existen grafos dirigidos, es decir, cuentan con una flecha que indica la dirección de la relación que los vértices o nodos sostienen. Por ejemplo, la Fig. 2, muestra el grafo dirigido $G = (V, A)$, de la siguiente manera: sus vértices $V = (A, E, I, O, U)$ y sus aristas $A = \{(A, O), (O, U), (E, A), (E, I), (I, E)\}$.

Otra propiedad interesante de los grafos es que estos pueden tener prioridad o se le puede asignar un peso a las aristas. En otras palabras las aristas tienen asociada una magnitud llamada *peso*, en dicho caso al grafo se le conoce como *grafo valorado*.

1.3 Camino de un grafo

Los caminos se calculan de un vértice a otro, y se denomina *P (path en inglés)*. P tiene longitud n desde el vértice v_0 a v_n siguiendo la secuencia de *n+1* vértices $v_0, v_1, \ldots, v_n$ tal que $(v_i, v_{i+1}) \in A$ para $0 \leq i \leq n$. Por lo tanto, el camino está representado por: $P = (v_0, v_1, v_2, v_3, \ldots, v_n)$. De tal manera, que se puede concluir que la longitud del camino es el número de aristas que lo forman. La Fig. 3 ejemplifica lo comentado anteriormente, siendo un posible camino $P_1 = (17, 16, 6, 19, 21)$ de longitud 4. Otro camino es $P_2 = (18, 15)$ siendo de longitud 1.

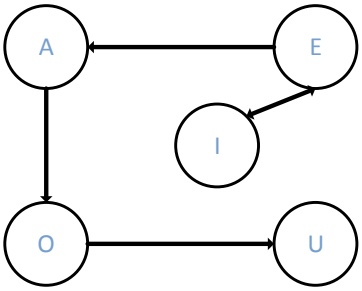

Figure 2: Grafo dirigido.

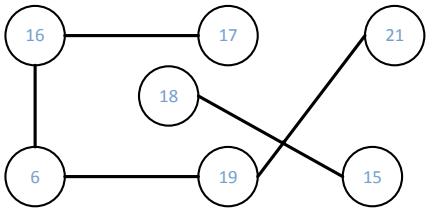

Figure 3: Camino de un grafo.

Cuando se trata de un grafo valorado o que contiene pesos, la longitud del camino es la suma de los pesos de las aristas en el camino.

Un camino cerrado es un ciclo en un grafo dirigido, es decir, un ciclo comienza y finaliza en un mismo nodo $v_0 = v_n$, con la restricción de que debe tener más de un arco. Por el contrario, el grafo dirigido acíclico (GDA) no tiene ciclos.

A los grafos donde existe un camino entre cualquier par de nodos o vértices se les conoce como grafos *conexos*. Por otro lado un se le conoce como grafo *completo* aquel que tiene una arista para cualquier par de nodos o vértices.

1.4 Representaciones matriciales

Matriz de adyacencia

Sirve para representar las relaciones entre pares de nodos o vértices. Esto se logra mediante el uso de una matriz de tantos renglones/columnas como vértices [2].

Sea $G = (V, A)$ un grafo de n vértices o nodos. Donde $V = \{v_0, v_1, v_2, \ldots, v_{n-1}\}$ representa el conjunto de vértices y $A = \{(v_i, v_j)\}$ representa las aristas o arcos. Por lo tanto, se utiliza una matriz X de nxn elementos, denominada matriz de adyacencia. Y cada elemento x_{ij} puede tener los siguientes valores:

$$x_{ij} = \begin{cases} \text{si } existearista(v_i, v_j) & 1 \\ \text{si } sinoexistearista(v_i, v_j) & 0 \end{cases} \tag{1}$$

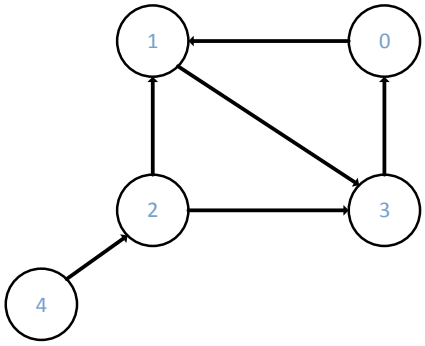

Figure 4: Grafo dirigido con vértices {0, 1, 2, 3, 4}.

La Fig. 4, representa un grafo dirigido. Considere el orden de los vértices como $\{0, 1, 2, 3, 4\}$, entonces su matriz de adyacencia:

$$X = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2}$$

Por otro lado, los grafos no dirigidos siempre tienen como matriz de adyacencia, una matriz simétrica. Es decir, si v_i está relacionado con v_j entonces esto implica que v_j esté relacionado con v_i.

De igual manera, cuando se trata de un grafo valorado o con aristas con peso, se sustituye dicho peso en la matriz de adyacencia. La Fig. 5, muestra un grafo dirigido con peso en sus aristas.

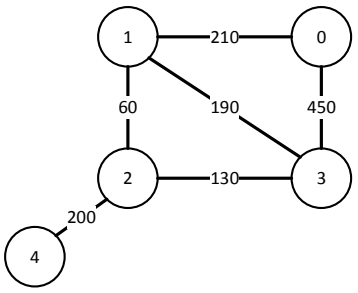

Figure 5: Grafo no dirigido con peso.

La matriz de adyacencia, simétrica es la siguiente:

$$X = \begin{pmatrix} 0 & 210 & 0 & 450 & 0 \\ 210 & 0 & 60 & 190 & 0 \\ 0 & 60 & 0 & 130 & 200 \\ 450 & 190 & 130 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 \end{pmatrix} \tag{3}$$

Listas de adyacencia

Las matrices de adyacencia son poco eficientes cuando contienen muchos ceros, se desperdicia memoria. Para grafos dispersos la matriz de adyacencia tiene el mismo espacio en memoria que un grafo que tiene muchas aristas. Cuando esto ocurre se utiliza otra representación mediante el uso de listas, a esto se le conoce como listas de adyacencia.

En las listas de adyacencia existe la noción de vértice origen, de esté se desprenden o enlazan los demás nodos o vértices adyacentes [2].

Tomemos por ejemplo el grafo dirigido de la Fig. 4, su lista de adyacencia quedaría representada de la siguiente manera:

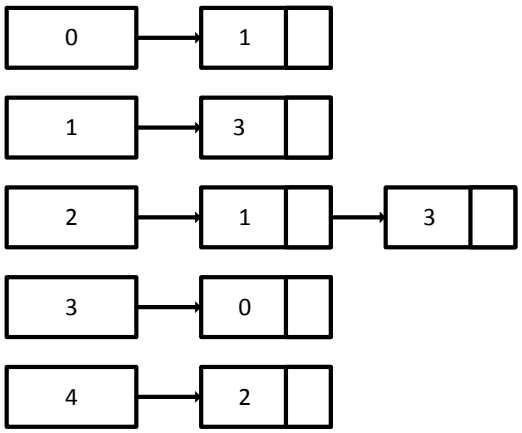

Figure 6: Lista de adyacencia de la Fig. 4.

1.5 Recorridos

Al igual que con los árboles binarios, el recorrido de un grafo se da a partir de un nodo o vértice particular, y el recorrer el grafo consiste en visitar o procesar la información almacenada en cada nodo. Esta tarea conlleva varios problemas ya que es necesario examinar lar aristas o arcos y procesar, en muchos casos todos, los nodos del grafo.

Existen dos recorridos conocidos para recorrer los grafos: *recorrido en profundidad* y *recorrido en anchura*. Es común, que para recorrer una nodos tratados como una cola se utilice el recorrido en anchura. Por el contrario, si se tratan los nodos como pila es común que el recorrido se haga en profundidad.

Recorrido en anchura

Este recorrido usa una cola como estructura para almacenar los nodos o vértices marcados como visitados, que se van a procesar posteriormente [2]. La cola ayuda a tratar a los elementos desde un vértice inicial, v, posteriormente se procesan o visitan sus nodos adyacentes para que después se visiten los nodos adyacentes a éstos, con la restricción de que no hayan sido visitados anteriormente.

El recorrido en anchura de un grafo se puede simplificar siguiendo una serie de pasos:

1. Visitar el nodo o vértice de partida v.
2. Encolar el vértice de partida.
3. Repetir paso 4 y 5 hasta que la cola se encuentre vacía.
4. Desencolar nodo frente de la cola, w, posteriormente visitar w.
5. Encolar todos los vértices adyacentes a w que no hayan sido visitados, posteriormente marcar estos vértices.

El recorrido en anchura y sus pasos están ejemplificados utilizando la Fig. 7 y la Tabla 1. El recorrido se lleva a cabo partiendo del nodo o vértice *D*.

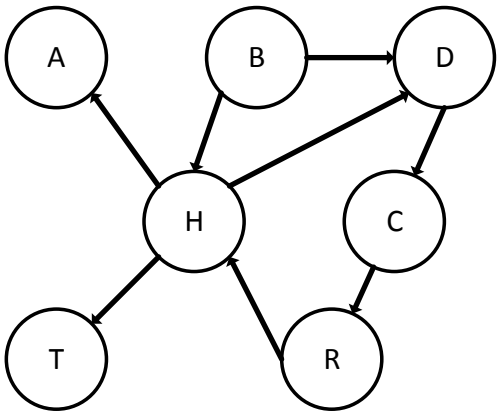

Figure 7: Recorrido en anchura.

Table 1: Recorrido en anchura del grafo de la Fig. 7.

Cola			Vértices procesados
D			
B	C		**D**
C	H		**B**
H	R		**C**
R	A	T	**H**
A	T		**R**
T			**A**
Cola Vacía			**T**

Recorrido en profundidad

El recorrido en profundidad tiene el mismo objetivo que el recorrido en anchura, el cual es visitar y procesar cada uno de los vértices del grafo [2]. Este recorrido difiere ya que utiliza la estructura pila, es decir, siempre se quita el último elemento añadido.

El recorrido comienza con algún nodo elegido, v, éste se marca como visitado y se hace la función push o meter en pila. Después se recorre en profundidad todos los vértices adyacentes a v no visitados, de esta manera hasta que no existan nodos no visitados. Se denomina profundidad por la dirección en visitar los vértices es hacia adelante, mientras sea posible.

A continuación se muestra el recorrido en profundidad de la Fig. 7, comenzando desde el vértice D, la Tabla 2 muestra el uso de la pila y los vértices visitados.

Table 2: Recorrido en profundidad del grafo de la Fig. 7.

Cola			Vértices procesados
D			
B	C		**D**
B	R		**C**
B	H		**R**
B	A	T	**H**
B	A		**T**
B			**A**
Pila Vacía			**B**

1.6 Algoritmos de búsqueda de rutas

Varios problemas pueden ser modelados usando grafos en diversos campos o disciplinas. Sin embargo, estos problemas requieren examinar todos los vértices y aristas del grafo. En particular, los algoritmos de recorrido imponen implícitamente un orden al momento de visitar los nodos, por ejemplo, se visitaría el nodo más próximo o las aristas más cercanas. Pero, no todos los algoritmos requieren un orden concreto al recorrer el grafo.

Algoritmo de Dijkstra

Uno de los problemas que más se plantea para los grafos es el encontrar el camino más corto entre dos vértices o nodos cualesquiera. Por ejemplo, determinar el camino más corto desde un punto de la ciudad a otro, teniendo varias rutas o caminos que se pueden elegir. El problema anterior, se puede resolver mediante el uso de un grafo dirigido que tenga factor de peso, donde este peso da el costo c_{ij} entre el par de vértices (v_i, v_j).

La longitud de camino esta dada por la siguiente formula:

$$LC = \sum_{i=1}^{k-1} c_{i,i+1} \tag{4}$$

El problema consiste en encontrar el camino de longitud mínima. Para resolver este problema, desde un vértice origen cualesquiera en un grafo valorado, se propone el algoritmo de Dijkstra.

Sea $G = (V, A)$ un grafo valorado cuyos valores son positivos, el algoritmo de Dijkstra es sencillo y eficiente para encontrar el camino más corto desde un nodo hacia el resto de los vértices [1].

El algoritmo selecciona en cada paso un vértice v, cuya distancia es desconocida entre los vértices que tienen la distancia más corta al vértice seleccionado o origen (s). De esta forma, se selecciona el camino más corto de s a v, el vértice v queda marcado como visitado para no visitarlo de nuevo. Así sucesivamente se van visitando o marcando los vértices desconocidos hasta que estén todos visitados o marcados, momento en el cual se conoce la distancia mínima del origen s al resto de los vértices.

Para construir el camino de longitud mínima que lleva de s a cada vértice v del grafo se almacena el último vértice o nodo con costo mínimo, esto se hace para cada vértice. El siguiente ejemplo muestra

los pasos que sigue el algoritmo de Dijkstra para encontrar el camino con longitud mínima de todos los vértices.

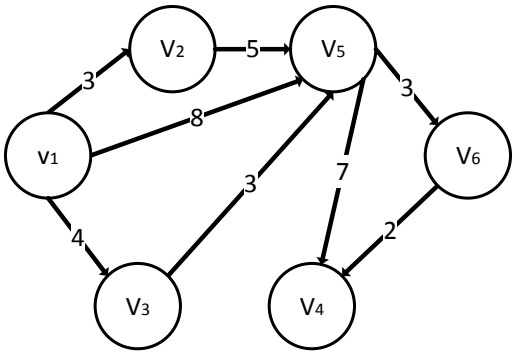

Figure 8: Grafo dirigido (Algoritmo de Dijsktra).

Los pasos que sigue el algoritmo de Djisktra se muestran en la Tabla 3. Donde se pretende encontrar el camino más corto desde el vértice S al vértice F. Se utiliza un vector distancia D.

Table 3: Recorrido en profundidad del grafo de la Fig. 8.

Paso	F	v	D[2]	D[3]	D[4]	D[5]	D[6]
Inicial	1		3	4		8	
1	1 ,2	2	3	4		8	
2	1, 2, 3	3	3	4		7	
3	1, 2, 3, 5	5	3	4	14	7	10
4	1, 2, 3, 5, 6	6	3	4	12	7	10
5	1, 2, 3, 5, 6, 4	4	3	4	12	7	10

El camino con longitud mínima es de S a F es 10. La secuencia de vértices que forman el camino es: *S, C, E, F* .

References

[1] Kurt Mehlhorn and Peter Sanders. *Algorithms and data structures: The basic toolbox.* Springer Science & Business Media, 2008.

[2] Clifford A Shaffer. *A practical introduction to data structures and algorithm analysis.* Prentice Hall Upper Saddle River, NJ, 1997.