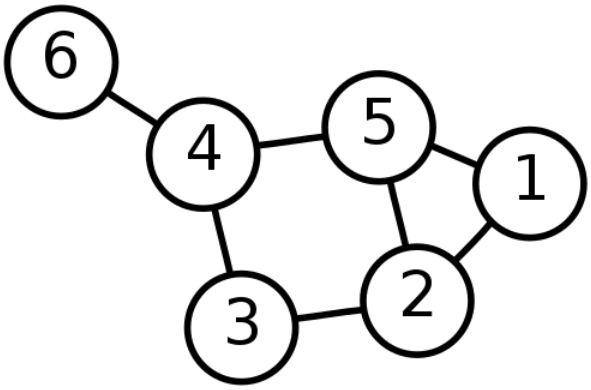
Grafos

Barreiro Valdez Alejandro

4 de noviembre de 2021

1. Conceptos básicos

Un grafo es una estructura de datos no lineal que está conformado por un conjunto de nodos o vértices y que son conectados por aristas o arcos. En los nodos se almacena algún dato y las aristas ayudan a relacionar esos nodos.



1.1. Vértice

En grafos, un vértice es cada uno de los nodos o de los elementos que conforman un grafo.

1.2. Arista

La arista entre dos vértices de un grafo es la conexión entre estos dos vértices.

1.3. Grado

El grado de un vértice es el número de aristas conectadas a un vértice.

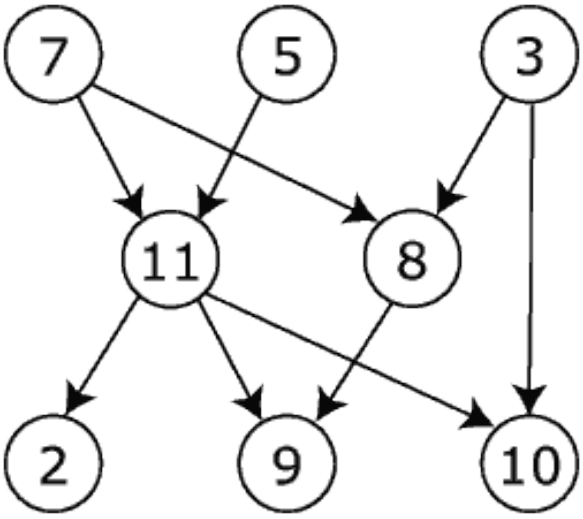
1.4. Adyacencia

Se dice que dos vértices son adyacentes si hay una arista conectándolos.

2. Clasificación de grafos

2.1. Grafos dirigidos

También es conocido como dígrafo, las aristas de este tipo de grafo tienen una dirección. Esto quiere decir que la arista parte de un nodo para llegar a otro.

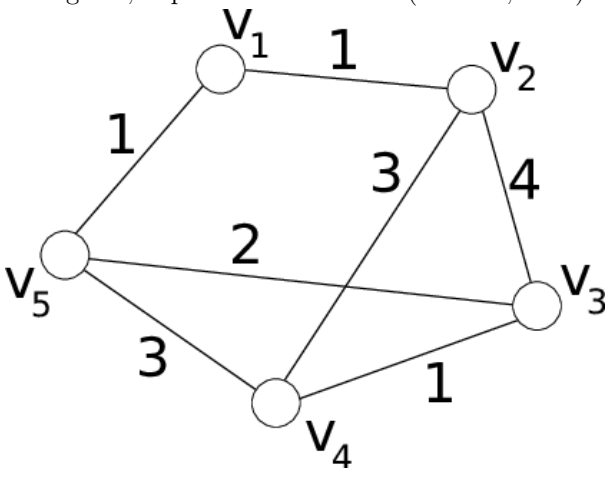


2.2. Grafos no dirigidos

Un grafo donde todos las aristas son bidireccionales. Cuando se tienen dos vértices adyacentes en un grafo no dirigido, se puede ir de uno a otro sin importar de cuál se está partiendo.

2.3. Grafo ponderado

Los grafos ponderados son aquellos donde las aristas tienen un valor. Los valores que se asocian a la arista se llaman peso. Estos valores pueden representar un costo, una longitud, el peso o la velocidad. (Sharma, 2021)



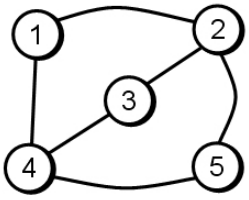
2.4. Grafo no ponderado

Este tipo de grafos no tiene un valor asociado con las aristas. Todos los grafos son no ponderados hasta que se asocia algún valor con las aristas.

3. Representación de grafos

3.1. Matriz de adyacencia

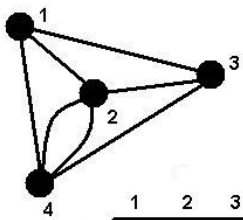
Una matriz de adyacencia es una matriz cuadrada (nxn) donde se indican las relaciones de adyacencia entre nodos. Si existe una arista entre el vértice i y el vértice j se debe poner un número en la posición i,j de la matriz. Para los grafos no ponderados se utiliza un uno si existe la arista y un cero si no. Para los grafos ponderados se pone el valor del peso o cero en caso de que no exista la arista. En los grafos no dirigidos se obtiene como resultado una matriz simétrica porque se tienen relaciones bidireccionales. La representación es fácil de implementar y de seguir. Para recorrer todos los vecinos de un nodo el proceso se vuelve tardado.



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

3.2. Matriz de incidencia

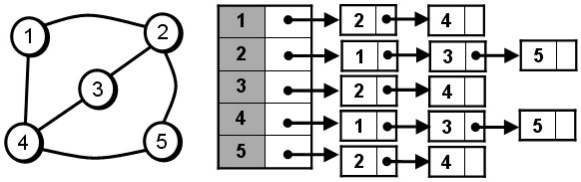
Esta matriz se construye utilizando el número de vértices como el número de filas y el número de aristas como el número de columnas. La matriz se llena con los valores de 1, -1 y 0. Se utiliza 1 para aquellas aristas que salen del vértice. Se utiliza -1 para aquellas aristas que ingresan al vértice. Se utiliza 0 cuando la arista no tiene conexión con el vértice. (Deepali, 2021)



	1	2	3	4	5	6	7
1	1	0	0	1	0	0	1
2	1	1	0	0	1	1	0
3	0	1	1	0	0	0	1
4	0	0	1	1	1	1	0

3.3. Lista de adyacencia

Una lista de adyacencia es un conjunto de listas ligadas donde se realiza una lista de los vecinos de cada uno de los vértices del grafo. Cada una de las listas de los vértices contiene a los vértices adyacentes. Las listas de adyacencia ayudan a ahorrar espacio y se puede editar de manera fácil. Saber si dos vértices son adyacentes o no se vuelve más lento. (JavaTPoint, s.f.)



4. Recorrido de grafos

Los dos algoritmos utilizados para recorrer todos los nodos de un grafo son el DFS y el BFS.

4.1. DFS (Depth First Search)

1. Poner un vértice inicial del grafo en el tope de la pila.
2. Poner el tope de la pila en la lista de visitados.
3. Crear una lista de los nodos adyacentes al vértice de la lista de visitados. Agregar al tope de la pila aquellos vértices que no estén en la pila.
4. Repetir los pasos 2 y 3 hasta que la pila esté vacía.

(Programiz, s.f.-b) Este algoritmo es utilizado para encontrar ciclos y encontrar el camino entre dos vértices.

4.2. BFS (Breadth-First Search)

1. Iniciar encolando el vértice inicial del grafo.
2. Desencolar un vértice y agregarlo a la lista de visitados.
3. Crear una lista de los nodos adyacentes al vértice de la lista de visitados. Agregar a la cola aquellos vértices que no estén en la pila.
4. Repetir los pasos 2 y 3 hasta que la cola esté vacía.

(Programiz, s.f.-a) Este algoritmo puede ser utilizado para encontrar los nodos vecinos de un vértice y para encontrar caminos entre dos vértices. Este algoritmo es usado en redes y en GPS. (Chakraborty, 2019)

Referencias

Chakraborty, A. (2019, 27 de agosto). Applications of dfs and bfs in data structures. *Tutorials Point*. Descargado de https://www.tutorialspoint.com/applications-of-dfs-and-bfs-in-data-structures

Deepali. (2021, 23 de septiembre). Graphs in data structure: Types, representation, operations. *Naukri Learning*. Descargado de https://www.naukri.com/learning/articles/author/deepali/

JavaTPoint. (s.f.). Graph theory. *JavaTPoint*. Descargado de https://www.javatpoint.com/graph-theory-graph-representations

Programiz. (s.f.-a). Breadth first search. *Programiz*. Descargado de https://www.programiz.com/dsa/graph-bfs

Programiz. (s.f.-b). Depth first search. *Programiz*. Descargado de https://www.programiz.com/dsa/graph-dfs

Sharma, R. (2021, 7 de octubre). Graphs in data structure: Types, storing traversal. *upGrad*. Descargado de https://www.upgrad.com/blog/graphs-in-data-structure/