Unidad II

Teoría de grafos

2.1. Conceptos básicos de grafos.

La **teoría de grafos** (también llamada **teoría de las gráficas**) es un campo de estudio de las matemáticas y las ciencias de la computación, que estudia las propiedades de los grafos (también llamadas *gráficas*) estructuras que constan de dos partes, el conjunto de vértices, nodos o puntos; y el conjunto dearistas, líneas o lados (*edges* en inglés) que pueden ser orientados o no.

La teoría de grafos es una rama de la matemáticas discretas y aplicadas, y es una disciplina que unifica diversas áreas
como combinatoria, álgebra,probabilidad, geometría de
polígonos, aritmética y topología.

La Teoría de Grafos es parte de Matemáticas Discreta que es una asignatura del plan de estudios de Ingeniería en Informática, que tiene un marcado enfoque práctico, aplicado y computacional, además de un acentuado carácter formativo. El contenido referido a esta temática se plantea como respuesta a una variada serie de problemas de la «vida real» (diseño de bloques, flujo de redes, diseño de circuitos, transporte de viajeros, asignaciones horarias o de tareas, programación, etc.), lo que le confiere el enfoque aplicado que señalamos arriba, aprendiendo el alumno, además, a buscar modelos matemáticos adecuados para gran número de situaciones diferentes, lo que suele ser muy habitual en el desarrollo profesional.

2.2. Clasificación de grafos.

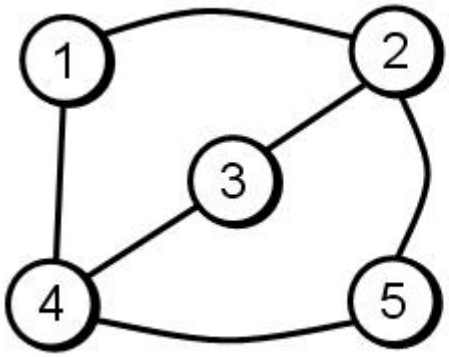
Un **grafo** en el ámbito de las ciencias de la computación es una estructura de datos, en concreto un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos. El concepto de grafo TAD desciende directamente del concepto matemático de grafo.

Informalmente se define como G = (V, E), siendo los elementos de V los vértices, y los elementos de E, las aristas (*edges* en inglés). Formalmente, ungrafo, G, se define como un par ordenado, G = (V, E), donde V es un conjunto finito y E es un conjunto que consta de dos elementos de V.

2.3. Representación de estructuras mediante grafos.

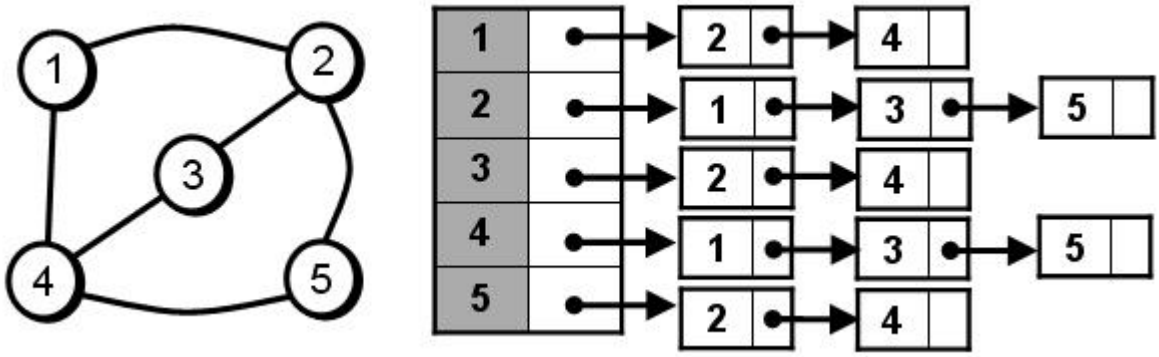
Existen diferentes implementaciones del tipo grafo: con una matriz de adyacencias (forma acotada) y con listas y multilistas de adyacencia (no acotadas).

- *Matriz de adyacencias:* se asocia cada fila y cada columna a cada nodo del grafo, siendo los elementos de la matriz la relación entre los mismos, tomando los valores de 1 si existe la arista y 0 en caso contrario.



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

- *Lista de adyacencias:* se asocia a cada nodo del grafo una lista que contenga todos aquellos nodos que sean adyacentes a él.



Especificación de los tipos abstractos de datos de un grafo no dirigido

---

Generadores

Crear un grafo vacío: Devuelve un grafo vacío.

- op crearGrafo : -> Grafo [ctor] .

Añadir una arista: Dado un grafo, añade una relación entre dos nodos de dicho grafo.

- op añadirArista : Grafo Nodo Nodo -> [Grafo] [ctor] .

Añadir un nodo: Dado un grafo, incluye un nodo en él, en caso en el que no exista previamente.

- op añadirNodo : Grafo Nodo -> Grafo [ctor] .

Constructores

Borrar nodo: Devuelve un grafo sin un nodo y las aristas relacionadas con él. Si dicho nodo no existe se devuelve el grafo inicial.

- op borrarNodo : Grafo Nodo -> Grafo .

Borrar arista: Devuelve un grafo sin la arista indicada. En caso de que la arista no exista devuelve el grafo inicial.

- op borrarArista : Grafo Nodo Nodo -> Grafo .

Selectores

Grafo Vacio: Comprueba si un grafo no tiene ningún nodo.

- op esVacio : Grafo -> Bool .

Contener Nodo: Comprueba si un nodo pertenece a un grafo.

- op contiene : Grafo Nodo -> Bool .

Adyacentes: Comprueba si dos nodos tienen una arista que los relacione.

- op adyacentes : Grafo Nodo Nodo -> Bool .

Para la especificación de un grafo dirigido tenemos que modificar algunas de las ecuaciones de las operaciones borrarArista y añadirArista para que no se considere el caso de aristas bidireccionales.

Y sustituir la operación adyacentes por:

- op predecesor : Grafo Nodo Nodo -> Bool .

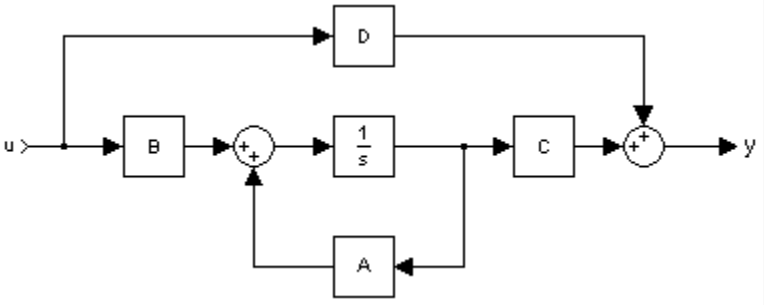
op sucesor : Grafo Nodo Nodo -> Bool .

2.4. Espacio de estados.

En ingeniería de control, una **representación de espacios de estados** es un modelo matemático de un sistema físico descrito mediante un conjunto de entradas, salidas y variables de estado relacionadas por ecuaciones

diferenciales de primer orden que se combinan en una ecuación diferencial matricial de primer orden. Para prescindir del número de entradas, salidas y estados, las variables son expresadas como vectores y las ecuaciones algebraicas se escriben en forma matricial (esto último sólo puede hacerse cuando el sistema dinámico es lineal e invariante en el tiempo). La representación de espacios de estado (también conocida como **aproximación en el dominio del tiempo**) provee un modo compacto y conveniente de modelar y analizar sistemas con múltiples entradas y salidas. Con p entradas y q salidas, tendríamos que escribir $q \times p$ veces la transformada de Laplace para procesar toda la información del sistema. A diferencia de la aproximación en el dominio de la frecuencia, el uso de la representación de espacios de estado no está limitada a sistemas con componentes lineales ni con condiciones iniciales iguales a cero. El **espacio de estado** se refiere al espacio de n dimensiones cuyos ejes coordenados están formados por variables de estados. El estado del sistema puede ser representado como un vector dentro de ese espacio.

Variables de estado



Modelo de espacio de estado típico

Artículo principal: Variable de estado (sistema dinámico)

Las *variables de estado* son el subconjunto más pequeño de variables de un sistema que pueden representar su estado dinámico completo en un determinado instante. Estas variables de estado deben ser linealmente independientes; una variable de estado no puede ser una combinación lineal de otras variables de estado. El número mínimo de variables de estado necesarias para representar un sistema dado, *n*, es normalmente igual al orden de la ecuación diferencial que define al sistema. Si el sistema es representado en forma de función de transferencia, el número mínimo de variables de estado es igual al orden del denominador de la función transferencia después de haber sido reducido a una fracción propia. Cabe destacar que al convertir una representación de espacio de estado a la forma de función de transferencia puede perderse información interna sobre el sistema, pudiendo por ejemplo describir un sistema como estable aun cuando la representación de espacio de estado indica que es inestable en ciertos puntos. En circuitos eléctricos, el número de variables de estado es a menudo,

pero no siempre, igual al número de elementos almacenadores de energía, como bobinasy condensadores.

2.5. Representación mediante espacio de estados.

El modelado y control de sistemas basado en la transformada de Laplace, es un enfoque muy sencillo y de fácil aplicación. Permite analizar sistemas utilizando una serie de reglas algebraicas en lugar de trabajar con ecuaciones diferenciales. En este enfoque tiene más valor la simplicidad que la exactitud.

Sin embargo, la descripción de sistemas mediante la función de transferencia tiene las siguientes limitaciones:

• No proporciona información sobre la estructura física del sistema.

• Solo es válida para sistemas lineales con una entrada y una salida e invariantes en el tiempo.

• No proporciona información de lo que pasa dentro del sistema.

• Se necesita que las condiciones iniciales del sistema sean nulas.

Ningún sistema dinámico de interés cumple con estos requisitos, es decir: Los sistemas reales presentan no linealidades, pueden tener más de una entrada o salida, sus parámetros cambian en el tiempo y sus condiciones iniciales no siempre tienen un valor de cero.

2.6. Estrategia y algoritmos de búsqueda.

Los algoritmos de búsqueda en grafos nacen por la necesidad de crear un mecanismo de navegación autónoma, bien sea de robots, coches, o personajes en un videojuego. Algunos de los más conocidos son A*, LPA*, o D*.

Un grafo, representa un conjunto de nodos unidos en una red. Si dos nodos están unidos, al viajar de uno a otro se considerara sucesor el nodo al que nos movemos, y predecesor el nodo del que venimos. Además, normalmente existirá un coste vinculado al desplazamiento entre nodos. Un algoritmo de búsqueda tratará, de encontrar un camino optimo entre dos nodos como por ejemplo un camino que minimice el coste de desplazamiento, o el numero de pasos a realizar. La principal diferencia entre los algoritmos es la información que guardan a cerca del grafo. Algunos de ellos no guardan información alguna, simplemente expanden la búsqueda desde el nodo inicial, hasta que se llega al nodo final, otros guardan

el coste de viajar desde el origen hasta ese nodo, o incluso una estimación de lo prometedor que es un nodo para conducir el camino a su objetivo. La expansión de la búsqueda se realiza en forma de árbol. Partiendo del nodo inicial, se extenderá la búsqueda a sus nodos vecinos, de cada uno de estos nodos vecinos, a sus respectivos nodos vecinos, y así hasta que uno de los nodos a los que se expande la búsqueda es el nodo objetivo. En esta página se desarrollará un algoritmo de búsqueda lo suficientemente general para trabajar en la mayoría de los grafos, y que da paso a otros métodos de búsqueda más complejos.

Notación

El algoritmo consta de dos listas, Abierta, y Cerrada. En la lista Abierta se guardan los nodos que aun no se han expandido para la búsqueda, que en otras palabras, serian las hojas de un árbol. En la lista Cerrada, se guardan los nodos que ya se han procesado y expandido, estos nodos se guardan porque la expansión de la búsqueda podría intentar volver a pasar por uno de esos nodos y de estar almacenados, se tiene constancia de los nodos que ya se han procesado. Además, cada nodo, almacenará información a cerca de quien es su nodo predecesor.