**Teoría de Grafos**

En matemáticas y en ciencias de la computación, la teoría de grafos estudia las propiedades de los grafos. Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas (edges en inglés) que pueden ser orientados o no. Típicamente, un grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas)

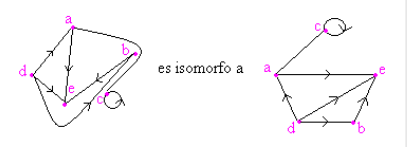
Grafos

Un grafo es una pareja de conjuntos G=(V,A), donde V es el conjunto de vértices, y A es el conjunto de aristas. Este último es un conjunto de pares de la forma (u,v) tal que . Para simplificar, notaremos la arista (a,b) como ab. En teoría de grafos, sólo queda lo esencial del dibujo: la forma de las aristas no son relevantes, sólo importa a qué vértices están unidas. La posición de los vértices tampoco importa, y se puede variar para obtener un dibujo más claro.

Muchas redes de uso cotidiano pueden ser modeladas con un grafo: una red de autopistas que conecta ciudades, una red eléctrica o la red de drenaje de una ciudad, etc.

Aristas Dirigidas y No Dirigidas

En algunos casos es necesario asignar un sentido a las aristas, por ejemplo, si se quiere representar la red de las calles de una ciudad con sus direcciones únicas. El conjunto de aristas será ahora un subconjunto de todos los posibles pares ordenados de vértices, con (a, b) ≠ (b, a). Los grafos que contienen aristas dirigidas se denominan grafos orientados (o dirigidos), como el siguiente:



Las aristas no orientadas se consideran bidireccionales para efectos prácticos (equivale a decir que existen dos aristas orientadas entre los nodos, cada una en un sentido). En el grafo anterior se ha utilizado una arista que tiene sus dos extremos idénticos: es un lazo (o bucle), y aparece también una arista bidireccional, y corresponde a dos aristas orientadas. Aquí V = { a, b, c, d, e }, y A = { (a, c), (d, a), (d, e), (a, e), (b, e), (c, a), (c, c), (d, b) }. Se considera la característica de "grado" (positivo o negativo) de un vértice p (y se indica como (v) ), como la cantidad de aristas que llegan o salen de él; para el caso de grafos no orientados, el grado de un vértice es simplemente la cantidad de aristas incidentes a este vértice. Por ejemplo, el grado positivo (salidas) de d es 3,mientras que el grado negativo (llegadas) de d es 0.

Caracterización de Grafos

Grafos simples

Un grafo es simple si a lo más existe una arista uniendo dos vértices cualesquiera. Esto es equivalente a decir que una arista cualquiera es la única que une dos vértices específicos. Un grafo que no es simple se denomina multigrafo.

Grafos conexos

Un grafo es conexo si cada par de vértices está conectado por un camino; es decir, si para cualquier par de vértices(a, b), existe al menos un camino posible desde a hacia b. Un grafo es doblemente conexo si cada par de vértices está conectado por al menos dos caminos disjuntos; es decir, es conexo y no existe un vértice tal que al sacarlo el grafo resultante sea disconexo.

Grafos completos

Un grafo es completo si existen aristas uniendo todos los pares posibles de vértices. Es decir, todo par de vértices (a,b) debe tener una arista e que los une.

El conjunto de los grafos completos es denominado usualmente K, siendo Kn el grafo completo de n vértice. Un grafo completo de n vértices tiene exactamente n(n-1)/2 aristas.

Grafos Bipartitos

Un grafo G es bipartito si puede expresarse como G= {V1 U V2, A} (es decir, sus vértices son la unión de dos grupos de vértices), bajo las siguientes condiciones:

•y son disjuntos y no vacíos.

• Cada arista de A une un vértice de V1 con uno de V2

.• No existen aristas uniendo dos elementos de V1; análogamente para V2.

Bajo estas condiciones, el grafo se considera bipartito, y puede describirse informalmente como el grafo que une o relaciona dos conjuntos de elementos diferentes, como aquellos resultantes de los ejercicios y puzzles en los que debe unirse un elemento de la columna A con un elemento de la columna B.

Distintas formas de representar un Grafo

1. Almacenando cada una de las aristas.

struct arista {

int fuente;

int destino;

}

1. Almacenando cada uno de los nodos, donde cada nodo tiene información sobre sus hijos en un arreglo

struct Nodo {

int valor;

struct Nodo sgtes[10];

}

1. Almacenando cada uno de los nodos, donde cada nodo tiene información sobre sus hijos en una lisa enlazada.

struct Nodo {

int valor;

struct Nodo \*sgtes;

}

1. A través de la matriz de adyacencia asociada al grafo

Por ejemplo, para un grafo con cinco nodos, donde cada nodo esta etiquetado con un número entero.

int matriz[5][5];

En cada celda (i,j) se almacenará un 1 si existe un paso (de largo 1) entre el nodo etiquetado con i y el nodo etiquetado con j. Se almacenará un 0 en caso contrario.

(veremos en clase un ejemplo sobre el uso de una matriz de adyacencia).