

Teoría de Grafos

☰ Etiquetas

Un **grafo** es una estructura matemática que consiste en un conjunto de **vértices** (o nodos) y un conjunto de **aristas** (o enlaces) que conectan pares de vértices. Los grafos se usan para modelar relaciones entre objetos, como rutas de transporte, redes sociales, circuitos eléctricos, etc.

1. Grafos Dirigidos y No Dirigidos

- **Grafo No Dirigido:** Las aristas no tienen dirección. La conexión entre dos vértices es bidireccional.
 - Ejemplo: En una red social, la amistad entre dos personas suele ser mutua; si A es amigo de B, entonces B es amigo de A.
- **Grafo Dirigido:** Las aristas tienen una dirección específica, van de un vértice a otro.
 - Ejemplo: En Twitter, la relación de "seguir" es unidireccional; si A sigue a B, no necesariamente B sigue a A.

2. Grafos Ponderados

Un **grafo ponderado** es aquel en el que cada arista tiene un valor o peso asociado. Esto puede representar distancia, costo, tiempo, capacidad, etc.

- Ejemplo: En un mapa de carreteras, cada camino entre ciudades tiene un peso que puede ser la distancia en kilómetros o el tiempo estimado de viaje.

3. Matriz de Incidencia y Matriz de Adyacencia

Estas son dos formas comunes de representar un grafo en forma matricial.

- **Matriz de Incidencia:**

Es una matriz con filas que representan vértices y columnas que representan aristas.

- Para grafos no dirigidos: en la columna correspondiente a una arista, se coloca un 1 en las filas de los dos vértices que conecta.
- Para grafos dirigidos: se usa 1 para el vértice origen y -1 para el vértice destino.
- Caso de uso: Útil en problemas de flujo de red y análisis estructural.

- **Matriz de Adyacencia:**

Es una matriz cuadrada donde las filas y columnas representan los vértices.

- Si existe una arista entre el vértice i y j , en la posición (i,j) se coloca 1 o el peso de la arista.
- Para grafos no dirigidos, la matriz es simétrica.
- Caso de uso: Es común en algoritmos que requieren acceso rápido a la conexión entre nodos, como búsqueda y recorrido.

4. Circuitos y Rutas

- **Circuito:** Es un camino que comienza y termina en el mismo vértice sin repetir aristas.
- **Ruta:** Es una secuencia de vértices donde cada par consecutivo está conectado por una arista.

5. Caminos de Euler y Hamilton

- **Camino de Euler:** Recorre todas las aristas del grafo exactamente una vez.

- **Circuito de Euler:** Un camino de Euler que es un circuito, es decir, comienza y termina en el mismo vértice.
 - Condición: Un grafo tiene circuito de Euler si es conexo y todos sus vértices tienen grado par (para no dirigidos).
 - Caso de uso: Problemas como el de repartir correspondencia, limpieza de calles o rutas de inspección.
 - **Camino de Hamilton:** Recorre todos los vértices exactamente una vez.
 - **Circuito de Hamilton:** Un camino de Hamilton que termina en el mismo vértice donde empezó.
 - No hay una condición sencilla para determinar su existencia.
 - Caso de uso: Optimización en rutas de ventas, tours turísticos, circuitos de ensamblaje.
-

6. Algoritmo de Dijkstra

- Sirve para encontrar el camino más corto desde un vértice origen a todos los demás vértices en un grafo ponderado con pesos no negativos.
 - **Funcionamiento básico:**
 - Se comienza desde el nodo inicial y se actualizan las distancias mínimas a sus vecinos.
 - Se elige el vértice con la distancia mínima no visitado, y se repite el proceso hasta cubrir todos.
 - **Caso de uso:** Sistemas de GPS para encontrar la ruta más rápida, redes de comunicación para optimizar el tráfico de datos.
-

7. Algoritmo de Prim

- Busca construir un **Árbol de Expansión Mínima** (Minimum Spanning Tree, MST) para un grafo conectado y ponderado.
 - Un MST es un subgrafo que conecta todos los vértices con el menor costo posible, sin formar ciclos.
 - **Funcionamiento:**
 - Empieza desde un vértice y agrega la arista más barata que conecta un nuevo vértice al árbol en construcción, hasta incluir todos.
 - **Caso de uso:** Diseño de redes eléctricas o de telecomunicaciones, minimizando el costo del cableado.
-

8. Algoritmo de Kruskal

- También busca el árbol de expansión mínima, pero lo hace seleccionando aristas de menor peso en orden ascendente, asegurándose de no formar ciclos.
 - **Funcionamiento:**
 - Ordena todas las aristas por peso.
 - Va agregando las aristas más baratas que no formen ciclos, hasta conectar todos los vértices.
 - **Caso de uso:** Similar a Prim, útil para planificar infraestructuras de bajo costo.
-

9. Valencias

Valencias en un Grafo No Dirigido

- **Valencia (o grado) de un vértice** es simplemente el número de aristas que inciden en ese vértice.
- Como las aristas no tienen dirección, cada arista conecta dos vértices y cuenta como 1 para ambos.

Ejemplo:

Si un nodo A está conectado con 3 aristas a otros nodos, decimos que A tiene valencia 3.

Valencias en un Grafo Dirigido

Aquí la cosa cambia porque cada arista tiene dirección. Entonces, definimos dos tipos de valencias para cada vértice:

1. Grado de entrada:

Número de aristas que llegan al vértice v .

2. Grado de salida:

Número de aristas que salen del vértice v .

¿Por qué es importante conocer las valencias?

- En grafos no dirigidos, el grado ayuda a determinar propiedades del grafo, como conectividad o existencia de caminos de Euler.
- En grafos dirigidos, los grados de entrada y salida permiten estudiar el flujo de información, identificar nodos fuente (solo salida) y sumidero (solo entrada), y analizar redes de distribución o jerarquías.

Resumen con ejemplos de la vida real

Tema	Ejemplo práctico
Grafo No Dirigido	Amistades en Facebook
Grafo Dirigido	Seguidores en Twitter
Grafo Ponderado	Mapas con distancias entre ciudades
Matriz de Incidencia	Redes eléctricas, análisis estructural
Matriz de Adyacencia	Sistemas de recomendación (relación rápida entre usuarios)
Circuitos de Euler	Ruta de barrenderos que debe pasar por todas las calles
Caminos de Hamilton	Recorrido turístico por una ciudad visitando cada lugar una vez
Dijkstra	GPS para encontrar la ruta más rápida
Prim	Diseño de red eléctrica minimizando el costo
Kruskal	Planificación de rutas de fibra óptica

Recursos:

[personal.us.es](https://personal.us.es/drorganvidez/wp-content/uploads/2023/01/GRAFOS-teoria-y-ejercicios-resueltos.pdf)

<https://personal.us.es/drorganvidez/wp-content/uploads/2023/01/GRAFOS-teoria-y-ejercicios-resueltos.pdf>

[www.unipamplona.edu.co](https://www.unipamplona.edu.co/unipamplona/portallG/home_23/recursos/general/11072012/grafos3.pdf)

https://www.unipamplona.edu.co/unipamplona/portallG/home_23/recursos/general/11072012/grafos3.pdf

[galois.azc.uam.mx](https://galois.azc.uam.mx/mate/propaganda/Leccion14.pdf)

<https://galois.azc.uam.mx/mate/propaganda/Leccion14.pdf>

www.cartagena99.com

<https://www.cartagena99.com/recursos/alumnos/apuntes/grafos-tutorial%20%2882PAGS%29.pdf>


Aprende sobre teoría de grafos con cursos online | edX

Toma cursos online gratis sobre teoría de grafos para dominar este tema. Aprende con cursos de las mejores universidades en edX. Únete hoy.

 <https://www.edx.org/es/aprende/teoria-de-grafos>

Teoría de GRAFOS en INFORMÁTICA: Que es un grafo, Tipos de Grafos, como representarlos y ejemplos

En el video de hoy voy a explicarte lo básico que tenés que saber sobre la teoría de grafos orientada a las ciencias de la computación. Vamos a ver que es un grafo, que tipos de grafos existen (No dirigido, dirigido, árboles, grafos acíclicos dirigidos, bipartitos), como representar un grafo en programación (matriz y lista de adyacencia) y algunos

 <https://youtu.be/F5Xjpg0-NhM?si=cXC08w3H26VAbT-2>

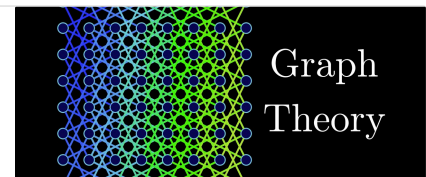
Introducción a la teoría de grafos



Introduction to Graph Theory: A Computer Science Perspective


In this video, I introduce the field of graph theory. We first answer the important question of why someone should even care about studying graph theory through an application perspective. Afterwards, we introduce definitions and essential terminology in graph theory, followed by a

 https://youtu.be/LFKZLXVO-Dg?si=_fn3NFvkJ4APg6vs



6.1 Graph Representation in Data Structure(Graph Theory)|Adjacency Matrix and Adjacency List

In this video, I have explained the two most popular methods(Adjacency Matrix and Adjacency List) for representing the Graph.

 https://youtu.be/5hPfm_uqXmw?si=ENotn64NdejbnAY1

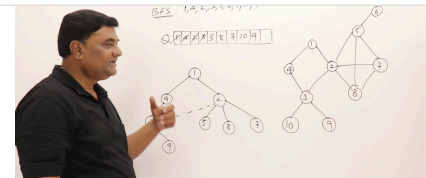


5.1 Graph Traversals - BFS & DFS -Breadth First Search and Depth First Search

Breadth First Search

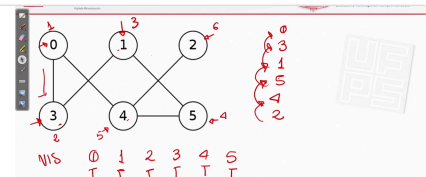
Depth First Search

 https://youtu.be/pcKY4hjDrxk?si=H_9ySOmTefmT78Nv




Recorridos sobre grafos:DFS -BFS

 <https://youtu.be/P45bRICTqHI?si=XxxOCQCdvXIPTARY>



Graph Data Structure — Theory and Python Implementation

A guide on how to implement the Graph data structure in Python.

 <https://python.plainenglish.io/graph-data-structure-theory-and-python-implementation-ee8c9795eae7>

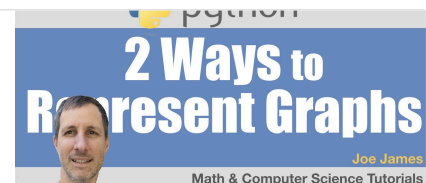


Python: 2 Ways to Represent GRAPHS

Two main ways of representing graph data structures are explained: using Adjacency Lists, and an Adjacency Matrix. This video also shows how to implement code for both in Python 3.

Download the latest code from my github site here, <https://github.com/joeyajames/Python>

 <https://youtu.be/HDUzBEG1GIA?si=xkyXKociA29TYSO4>



GTAC 2.6: Implementing a Graph Data Structure in Python

I walk through an implementation of a graph data structure from first principles in Python.


 <https://www.youtube.com/watch?v=uFaZY1dVnGs>

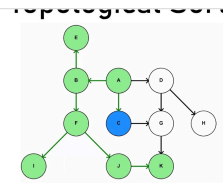


Topological Sort Visualized and Explained

My previous video on Depth-First Search <https://youtu.be/5GcSvYDgiSo>

This video should give you a quick overview of Topological Sort.


 <https://youtu.be/7J3GadLzyl?si=WIZIn2yBFv7okCF2>



Topological Sort Visualized and Explained

My previous video on Depth-First Search <https://youtu.be/5GcSvYDgiSo>

This video should give you a quick overview of Topological Sort.

 <https://youtu.be/7J3GadLzyl?si=WIZIn2yBFv7okCF2>

