Apuntes

Tema ***Grafos - Generalidades***

"La inteligencia consiste no sólo en el conocimiento,
sino también en la destreza de aplicar los conocimientos en la práctica."
Aristóteles (384 AC-322 AC) Filósofo griego.

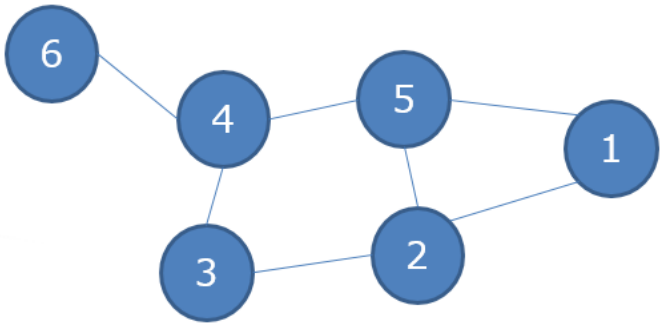
Grafos

¿Qué es un Grafo?

En matemáticas y ciencias de la computación, un grafo *(del griego grafos: dibujo, imagen)* es un conjunto de objetos llamados **vértices** o **nodos** unidos por enlaces llamados **aristas** o **arcos**, que permiten representar relaciones binarias entre elementos de un conjunto. *Son objeto de estudio de la teoría de grafos.*

Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras. Por ejemplo, una red de computadoras puede representarse y estudiarse mediante un grafo, en el cual los vértices representan terminales y las aristas representan conexiones (las cuales, a su vez, pueden ser cables o conexiones inalámbricas).

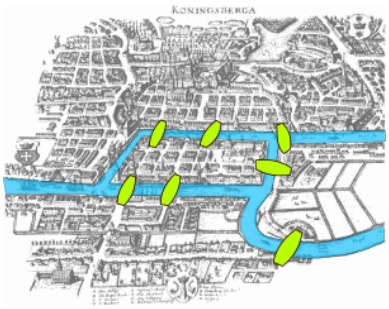
Prácticamente cualquier problema puede representarse mediante un grafo, y su estudio trasciende a las diversas áreas de las ciencias exactas y las ciencias sociales.



Historia

El primer artículo científico relativo a grafos fue escrito por el matemático suizo **Leonhard Euler** en 1736. Euler se basó en su artículo en el problema de los puentes de Königsberg. La ciudad de Kaliningrado, originalmente Königsberg, es famosa por sus **siete puentes** que unen ambas márgenes del río Pregel con dos de sus islas. Dos de los puentes unen la isla mayor con la margen oriental y otros dos con la margen occidental. La isla menor está conectada a cada margen por un puente y el séptimo puente une ambas islas. El problema planteaba lo siguiente: ***¿es posible dar un paseo comenzando desde cualquiera de estas regiones, pasando por todos los puentes,***

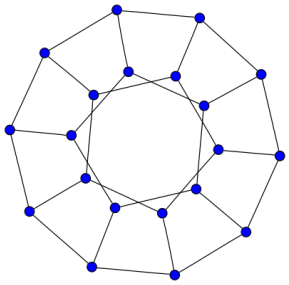
recorriendo solo una vez cada uno y regresando al mismo punto de partida?
Abstrayendo este problema y planteándolo con la (entonces aún básica) teoría de grafos, Euler consigue demostrar que el grafo asociado al esquema de puentes de Königsberg **no tiene solución**, es decir, no es posible regresar al vértice de partida sin pasar por alguna arista dos veces.



De hecho, Euler resuelve el problema más general: **¿qué condiciones debe satisfacer un grafo para garantizar que se puede regresar al vértice de partida sin pasar por la misma arista más de una vez?** Si definimos como **«grado»** al número de líneas que se encuentran en un punto de un grafo, entonces la respuesta al problema es que los puentes de un pueblo se pueden atravesar exactamente una vez si, salvo a lo sumo dos, todos los puntos tienen un grado par.

Tipos de Grafos

- ***Según conectividad de aristas***
  - **Grafo simple**: o simplemente grafo es aquel que acepta una sola arista uniendo dos vértices cualesquiera. Esto es equivalente a decir que una arista cualquiera es la única que une dos vértices específicos. Es la definición estándar de un grafo.
  - **Multigrafo**: o pseudografo son grafos que aceptan más de una arista entre dos vértices. Estas aristas se llaman múltiples o lazos (loops en inglés). Los grafos simples son una subclase de esta categoría de grafos. También se les llama grafos no-dirigido.



- ***Según la dirección de aristas***
  - **Grafo no dirigido**: Son grafos
  - **Grafo dirigido**: Son grafos en los cuales se ha añadido una orientación a las aristas, representada gráficamente por una flecha

$(v,w) = (w,v)$ 

$(v,w) \neq (w,v)$ 

- ***Otros tipo de Grafos***
  - **Grafo etiquetado**: Grafos en los cuales se ha añadido un peso a las aristas (número entero generalmente) o un etiquetado a los vértices.
  - **Grafo aleatorio**: Grafo cuyas aristas están asociadas a una probabilidad.
  - **Hipergrafo**: Grafos en los cuales las aristas tienen más de dos extremos, es decir, las aristas son incidentes a 3 o más vértices.
  - **Grafo infinito**: Grafos con conjunto de vértices y aristas de cardinal infinito.

Propiedades de los Grafos

- **Grado de un vértice:** (grafo no dirigido) Número de aristas que lo contienen.

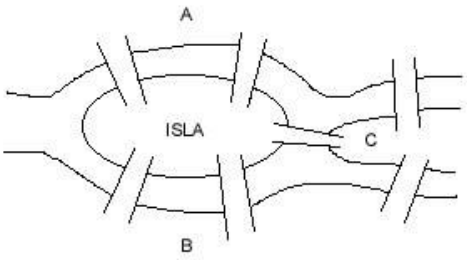
- **Grado de salida de un vértice v:** Número de arcos cuyo vértice inicial es v.
- **Grado de entrada de un vértice v:** Número de arcos cuyo vértice final es v
- **Adyacencia**: Decimos que un vértice w es adyacente a un vértice v si existe una arista (v,w) en E. En el caso de grafos no dirigidos, si w es adyacente a v, entonces v es adyacente a w. En el caso de dígrafos, esto no se cumple, a no ser que tanto (v,w) y (w.v) sean aristas del grafo.
    - **Nodos/vértices adyacentes**: Vértices conectados por una arista (o un arco).
    - **Aristas/arcos adyacentes**: Arcos/aristas con un vértice común.
- **Grafos conexos**: Un grafo es conexo si cada par de vértices está conectado por un camino, es decir, si para cualquier par de vértices (1, 2), existe al menos un camino posible desde 1 hacia 2.
- **Caminos**: Un camino es una secuencia de vértices w1,..,wN donde (wi,wi+1) ε E, para 1≤ i < N.
- **longitud de un camino**: La longitud de un camino *(sin pesos)* es el número de aristas en el camino. En el caso de grafos con pesos, la longitud de un camino está dada por la suma de los pesos.
- **Ciclo**: Un ciclo es un camino w1,..,wN donde w1 y wN son el mismo vértice, es decir, es un camino donde el vértice origen y destino coinciden. Se dice que un ciclo es hamiltoniano cuando tiene que recorrer todos los vértices exactamente una vez (excepto el vértice del que parte y al cual llega). Un grafo acíclico es, como su nombre lo sugiere, simplemente un grafo dirigido y que no contiene ciclos.
- **Bucle:** Arco/arista cuyos vértices inicial y final coinciden.
- **Orden topológico**: ordena los vértices de un grafo dirigido, donde si existe un camino de un vértice v a otro vértice w, entonces w debe aparecer después de u en la ordenación.

Camino Euleriano

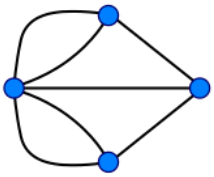
En la teoría de grafos, un **camino euleriano** es un camino que pasa por cada arista una y solo una vez. Un ciclo o circuito euleriano es un camino cerrado que recorre cada arista exactamente una vez. El problema de encontrar dichos caminos fue discutido por primera vez por **Leonhard Euler**, en el famoso problema de los **puentes de Königsberg**.

El origen de la teoría de los ciclos eulerianos fue planteado y resuelto por el propio Leonhard Euler en 1736 en un problema que tiene el nombre de Siete puentes de la ciudad de Königsberg (Prusia oriental en el siglo XVIII y actualmente, Kaliningrado, provincia rusa) dando origen a la Teoría de los grafos.

El problema se enuncia de la siguiente forma: Dos islas en el río Pregel, en Königsberg se unen entre ellas y con la tierra firme mediante siete puentes. ¿Es posible dar un paseo empezando por una cualquiera de las cuatro partes de tierra firme, cruzando cada puente una sola vez y volviendo al punto de partida?

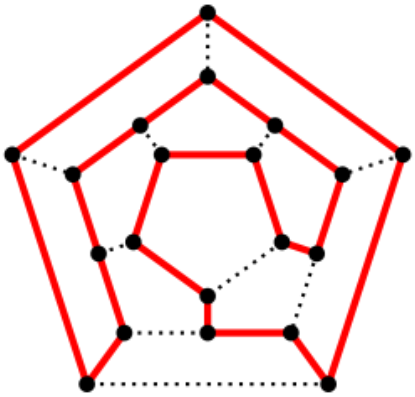


Euler enfocó el problema representando cada parte de tierra por un punto y cada puente, por una línea, uniendo los puntos que se corresponden. Entonces, el problema anterior se puede trasladar a la siguiente pregunta: ¿se puede recorrer el dibujo sin repetir las líneas?

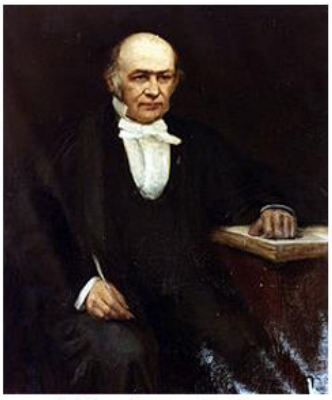


Euler demostró que no era posible puesto que el número de líneas que inciden en cada punto no es par (condición necesaria para entrar y salir de cada punto, y para regresar al punto de partida, por caminos distintos en todo momento).

Camino Hamiltoniano



Un camino hamiltoniano, en el campo matemático de la teoría de grafos, es un camino de un grafo, una sucesión de aristas adyacentes, que visita todos los vértices del grafo una sola vez. Si además el último vértice visitado es adyacente al primero, **el camino es un ciclo hamiltoniano**.
El problema de encontrar un ciclo (o camino) hamiltoniano en un grafo arbitrario se sabe que es **NP-completo**.
Los caminos y ciclos hamiltonianos se llaman así en honor de William Rowan Hamilton, inventor de un juego que consistía en encontrar un ciclo hamiltoniano en las aristas de un grafo de un dodecaedro. Hamilton resolvió este problema usando cuaterniones, aunque su solución no era generalizable a todos los grafos.



William Rowan Hamilton
(Dublín, 4 de agosto de 1805-ibídem, 2 de septiembre de 1865)

Definición: Un camino hamiltoniano es un camino que pasa por cada vértice exactamente una vez. Se denomina un ciclo hamiltoniano si es un ciclo que pasa por cada vértice exactamente una vez *(excepto el vértice del que parte y al cual llega)*. Un grafo que contiene un ciclo hamiltoniano se dice grafo hamiltoniano.
Estos conceptos se pueden extender para los grafos dirigidos los cuales son igual a un carro.

¿Para qué sirven los grafos?

Uno de los problemas más comunes en la Teoría de Grafos es encontrar el camino mínimo entre 2 vértices dados. Para ello vamos a analizar los siguientes algoritmos:

- Camino Mínimo sin Pesos con un único origen desde el vértice de origen a cualquier otro vértice del grafo
  - BFS (Búsqueda en anchura)
- Camino Mínimo con Pesos positivos y origen único, desde el vértice origen al resto de vértices del grafo

- Dijkstra

Representación de grafos

Existen diferentes formas de representar un grafo (simple), además de la geométrica y muchos métodos para almacenarlos en una computadora. La estructura de datos usada depende de las características del grafo y el algoritmo usado para manipularlo. Entre las estructuras más sencillas y usadas se encuentran las **listas** y las **matrices**, aunque frecuentemente se usa una combinación de ambas. Las listas son preferidas en grafos dispersos porque tienen un eficiente uso de la memoria. Por otro lado, las matrices proveen acceso rápido, pero pueden consumir grandes cantidades de memoria.

Representación Matricial

Matriz de adyacencia: El grafo está representado por una matriz cuadrada M de tamaño n^2, donde n es el número de vértices. Si hay una arista entre un vértice x y un vértice y, entonces el elemento m_{x, y} es 1, de lo contrario, es 0.

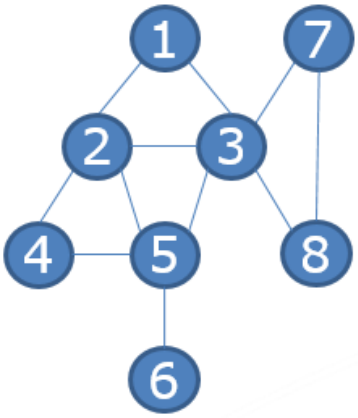
Ventaja:

- Acceso eficiente a una arista, Θ(1).

Inconvenientes:

- Θ(|V|2) en identificar identificar todas las aristas aristas.
- Espacio proporcional a |V|2 (se desperdicia memoria si el grafo es poco denso).

Ej.: Matriz de Adyacencia



	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

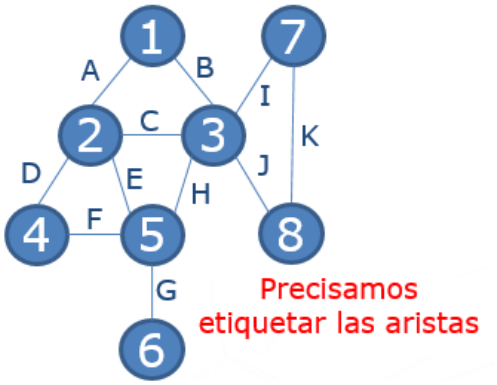
Simetría de la matriz al ser no dirigido

Matriz de incidencia: El grafo está representado por una matriz de A (aristas) por V (vértices), donde [vértice, arista] contiene la información de la arista (1 - conectado, 0 - no conectado)

Inconvenientes:

- Se desperdicia aún mas Espacio que en la Matriz de Adyacencia.
- Si la arista tuviese peso sería confuso representarlo.

Ej.: Matriz de Incidencia



	A	B	C	D	E	F	G	H	I	J	K
1	1	1	0	0	0	0	0	0	0	0	0
2	1	0	1	1	1	0	0	0	0	0	0
3	0	1	1	0	0	0	0	1	1	1	0
4	0	0	0	1	0	1	0	0	0	0	0
5	0	0	0	0	1	1	1	1	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0
7	0	0	0	0	0	0	0	0	1	0	1
8	0	0	0	0	0	0	0	0	0	1	1

Nótese que ya no es una matriz cuadrada. Ya que es poco probable que exista la misma cantidad de nodos que de aristas.

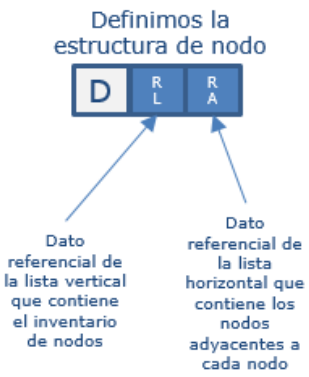
Representación por listas

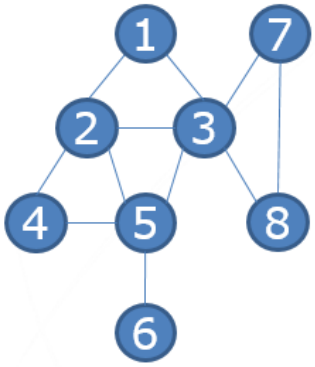
- **lista de incidencia**: Las aristas son representadas con un vector de pares (ordenados, si el grafo es dirigido), donde cada par representa una de las aristas.
- **lista de adyacencia**: Cada vértice tiene una lista de vértices los cuales son adyacentes a él. Esto causa redundancia en un grafo no dirigido (ya que A existe en la lista de adyacencia de B y viceversa), pero las búsquedas son más rápidas, al costo de almacenamiento extra.
- **lista de grados**: También llamada secuencia de grados o sucesión gráfica de un grafo no-dirigido es una secuencia de números, que corresponde a los grados de los vértices del grafo.

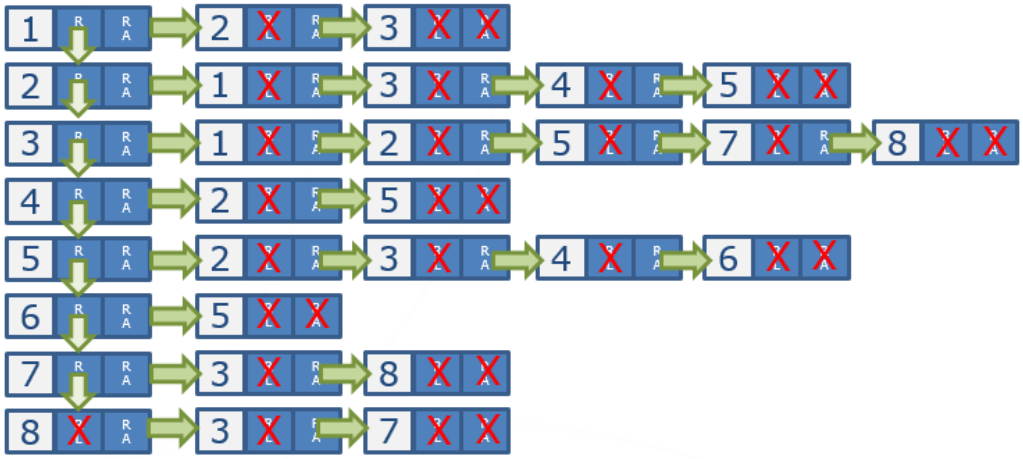
Nosotros veremos la Lista de Adyacencia:

Ventajas:
- Espacio proporcional a |V|+|E|.
- (representación adecuada para grafos poco densos).
- Grafos - Representación Representación
- Θ(|V|+|E|) (|V|+|E|) en identificar todas las aristas. en identificar todas las aristas.

Inconvenientes:
- Se tarda O(grado(u)) en comprobar si (u,v)∈E.
- Ineficiente para encontrar los arcos que llegan a un nodo
- (solución: usar estructuras de listas múltiples).

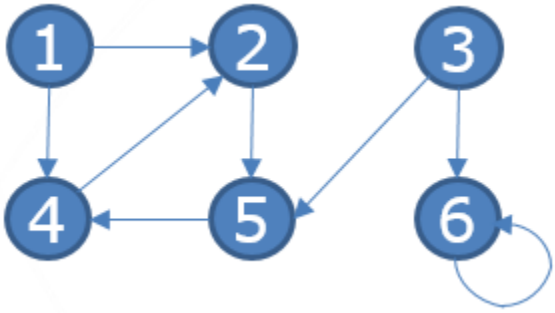




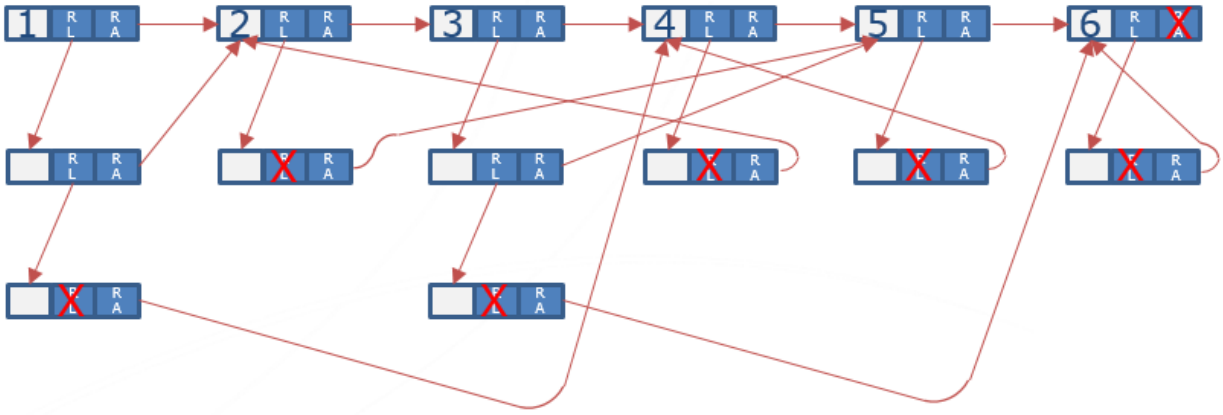


Como podemos ver es una lista de listas, pero para nada, podemos decir que es una representación óptima, ya que estamos repitiendo información

Un ejemplo de ***Estructura forma de lista, eficiente*** para la Representación de Grafos sería dado el siguiente grafo:

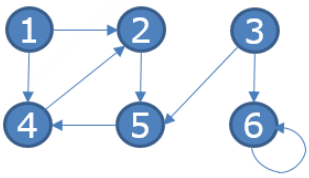


La estructura de lista de listas sería una lista de nodos, donde cada nodo apunta a una lista de aristas y cada arista apunta a cada nodo.

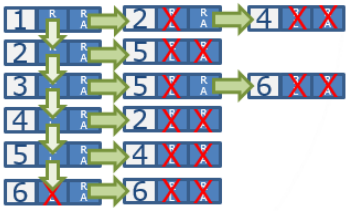


Ejemplos de representación

Grafo Dirigido



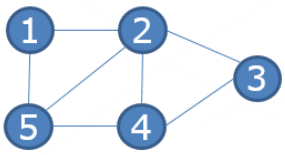
Lista de Adyacencia



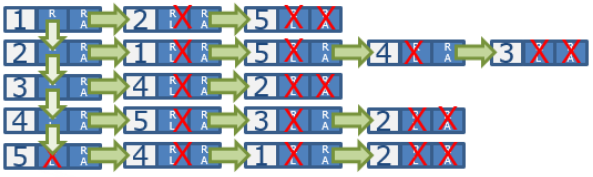
Matriz de Adyacencia

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

Grafo No Dirigido



Lista de Adyacencia



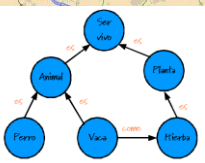
Matriz de Adyacencia

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

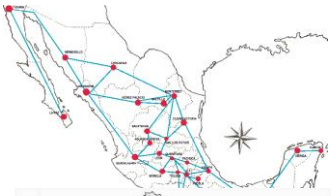
Aplicaciones de Grafos

Esta estructura de datos es muy utilizada dado que nos sirve para representar numerosas situaciones y problemas conocidos:

Ciudades y rutas: cada nodo representa una ciudad y cada arista una ruta que une dichas ciudades. ¿Cuál es el camino más corto? ¿Hay un camino que partiendo de una ciudad visite todas las ciudades una sola vez volviendo a la ciudad de partida?

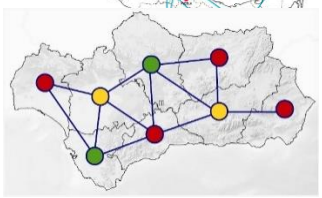


Actividades y dependencias: cada nodo representa una actividad. Las aristas sirven para representar dependencias entre actividades y de esta forma identificar tareas predecesoras. ¿Cuál es el camino crítico?

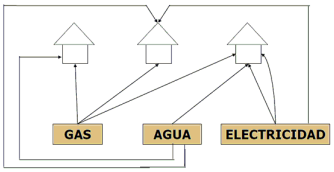


Mapas Conceptuales: cada nodo representa una idea. Las aristas la relación entre las ideas

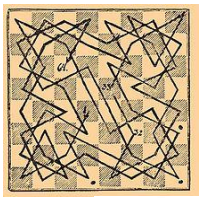
Problema del viajante de comercio: conocido como TSP por sus siglas en Ingles “Travelling salesman problem”. Dada una lista de las ciudades y la distancia entre ellas, la tarea consiste en encontrar la ruta más corta posible que visita cada ciudad exactamente una vez y regresa a la ciudad de origen.



Países limítrofes: cada vértice representa un país y cada arista muestra la relación de países limítrofes. ¿Cuántos colores como mínimo se necesita utilizar para colorear cada país de manera tal que ningún país limítrofe tenga el mismo color?



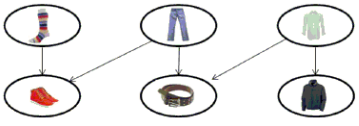
El problema de las casas y servicios: Supongamos 3 casas y 3 servicios (Luz, Gas, Teléfono). ¿Es posible brindarle servicio a cada casa sin que los cables se crucen?



Juego de ajedrez: Cada vértice representa un casillero y cada arista una posición válida para un caballo en el juego de ajedrez. ¿Es posible que el caballo parta de un casillero y visite todos los otros 63 casilleros una solo vez volviendo al punto inicial?

Juego de la casa: ¿Es posible dibujar la casa de la Figura sin levantar el lápiz y sin trazar 2 veces el mismo segmento?

Ordenamiento topológico: Un orden topológico ordena los vértices de un grafo dirigido, donde si existe un camino de un vértice v a otro vértice w, entonces w debe aparecer después de u en la ordenación.
Podemos utilizar este algoritmo para mostrar orden entre actividades. Supongamos el grafo de la Figura, donde se muestra la relación en el orden de la ropa para poder vestirse correctamente.



El orden topológico nos puede dar distintos resultados dado que no hay una única forma de ordenar los elementos. En la Figura 10 se muestras 2 ordenamientos correctos y uno que no lo es.

Grafos CPM

El método de la ruta crítica o del camino crítico es un algoritmo utilizado para el cálculo de tiempos y plazos en la planificación de proyectos. Este sistema de cálculo conocido por sus siglas en inglés ***CPM (Critical Path Method)***, fue desarrollado en 1957 en los Estados Unidos de América, por un centro de investigación de operaciones para las firmas Dupont y Remington Rand, buscando el control y la optimización de los costos mediante la **planificación** y programación adecuadas de las **actividades** componentes del **proyecto.**

En administración y gestión de proyectos, una **ruta crítica** es la secuencia de los elementos terminales de la red de proyectos con la mayor duración entre ellos, determinando el tiempo más corto en el que es posible completar el proyecto. La duración de la ruta crítica determina la duración del proyecto entero. Cualquier retraso en un elemento de la ruta crítica afecta a la fecha de término planeada del proyecto, y se dice que no hay holgura en la ruta crítica.

Computadora Univac

Un proyecto puede tener varias rutas críticas paralelas. Una ruta paralela adicional a través de la red con la duración total cercana a la de la ruta crítica, aunque necesariamente menor, se llama ruta sub-crítica.

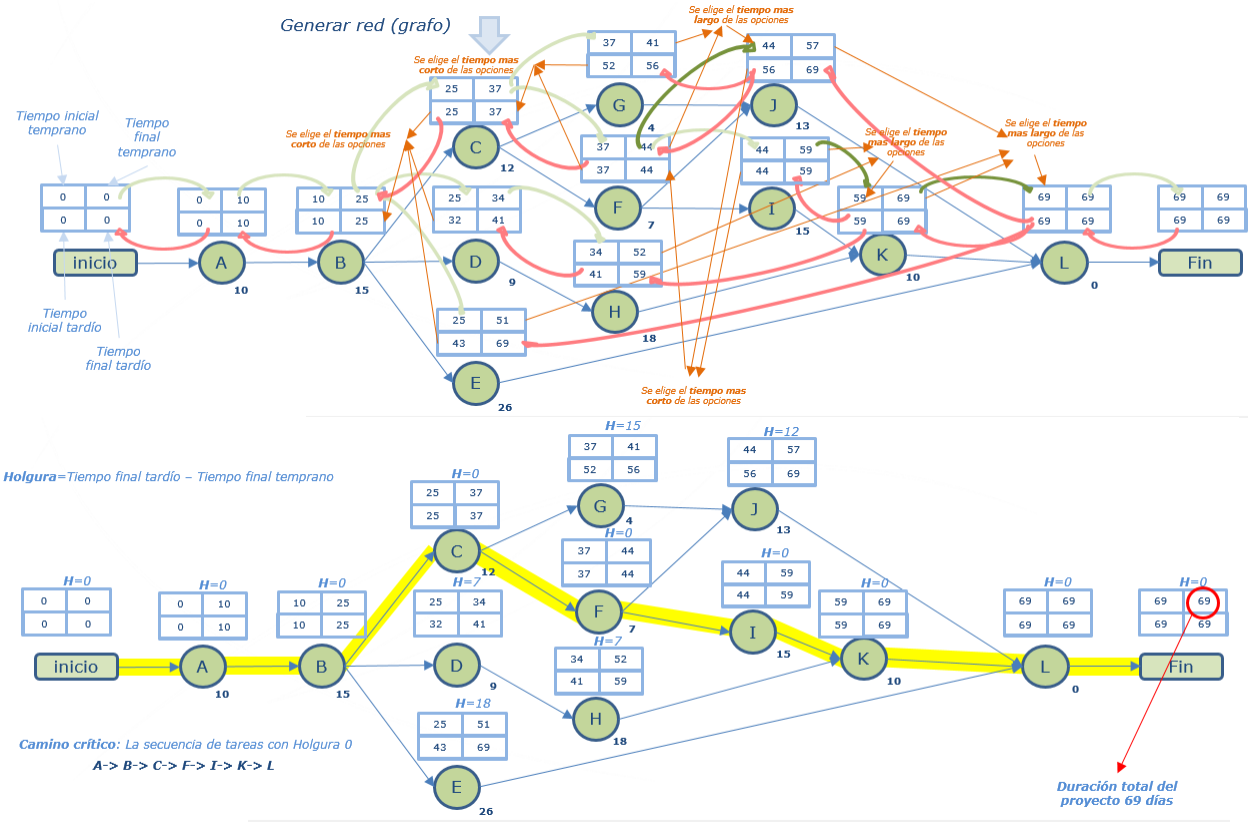
Originalmente, el método de la ruta crítica consideró solamente dependencias entre los elementos terminales. Un concepto relacionado es la cadena crítica, la cual agrega dependencias de recursos. Cada recurso depende del manejador en el momento donde la ruta crítica se presente.

A diferencia de la técnica de revisión y evaluación de programas (PERT), el método de la ruta crítica usa tiempos ciertos (reales o **deterministas**). Sin embargo, la elaboración de un proyecto basándose en redes CPM y PERT son similares y consisten en:

- Identificar todas las actividades que involucra el proyecto, lo que significa, determinar relaciones de precedencia, tiempos técnicos para cada una de las actividades.
- Construir una red con base en nodos y actividades (o arcos, según el método más usado), que implican el proyecto.
- Analizar los cálculos específicos, identificando la ruta crítica y las holguras de las actividades que componen el proyecto.

Ejemplo:

Actividad	Descripción Actividad	Precedencias	Tiempos
A	Obras preliminares		10
B	Movimiento de Tierra	A	15
C	Cimientos	B	12
D	Muros	B	9
E	Columnas	B	26
F	Techos	C	7
G	Inst. Sanitarias	C	4
H	Inst. Eléctricas	D	18
I	Revestimientos	F	15
J	Torre tanque de agua	F, G	13
K	Acabados	H, I	10
L	Entrega de obra	E, J, K	0



Grafos PERT

El origen de los trabajos de la técnica PERT empezaron formalmente en enero de 1957, siendo paralelo al del CPM, pero su origen fue en el ámbito militar. Se desarrolló en la Oficina de Proyectos Especiales de la Armada de los EEUU, al reconocer el almirante William. F. Raborn que se necesitaba una planificación integrada y un sistema de control fiable para el programa de misiles balísticos Polaris. Con su apoyo se estableció un equipo de investigación para desarrollar el PERT o "**Program Evaluation Research Task**".

William Francis Raborn (1905-1990) Militar estadounidense.

Así, la Oficina de Proyectos Especiales de la Marina de los Estados Unidos de América, en colaboración con la división de Sistemas de Misiles Lockheed (fabricantes de proyectiles balísticos) y la consultora Booz, Allen & Hamilton (ingenieros consultores), se plantean un nuevo método para solucionar el problema de planificación, programación y control del proyecto de construcción de submarinos atómicos armados con proyectiles «Polaris», donde tendrían que coordinar y controlar, durante un plazo de cinco años a 250 empresas, 9000 subcontratistas y numerosas agencias gubernamentales.

Misil balístico intercontinental Polaris A3 siendo disparado desde un submarino

En julio de 1958 se publica el primer informe del programa al que denominan "**Program Evaluation and Review Technique**", decidiendo su aplicación en octubre del mismo año y consiguiendo un adelanto de dos años sobre los cinco previstos. D. G. Malcolm, J. H. Roseboom, C. E. Clark y W. Fazar, todos del equipo de investigación patrocinado por la Armada, fueron los autores del primer documento publicado sobre el PERT (Malcolm et al., 1959). Este método se basa en la **probabilidad de la duración de las actividades**. Hoy día se sigue utilizando este método si bien, tal y como apuntan algunos autores (ver Ahuja et al., 1995), la estimación calculada por PERT suele subestimar la duración real de los proyectos.

Ejemplo:

Actividad	Descripción Actividad	Precedencias	Tiempo Optimista a	Tiempo Probable m	Tiempo Pesimista b	Tiempo Esperado	Varianza de ruta crítica
A	Obras preliminares		7	10	14	10,17	1,36
B	Movimiento de Tierra	A	12	15	19	15,17	1,36
C	Cimientos	B	12	13	15	13,17	0,25
D	Muros	B	7	9	12	9,17	
E	Columnas	B	24	27	28	26,67	
F	Techos	C	6	8	10	8	0,44
G	Inst. Sanitarias	C	4	5	7	5,17	
H	Inst. Eléctricas	D	16	19	20	18,67	
I	Revestimientos	F	14	16	17	15,83	0,25
J	Torre tanque de agua	F, G	11	14	15	13,67	
K	Acabados	H, I	8	10	13	10,17	0,69
L	Entrega de obra	E, J, K	0	1	3	1,17	0,25

Se pretende calcular *Calcular Holgura y camino crítico y para ello definimos*

a =Tiempo Optimista

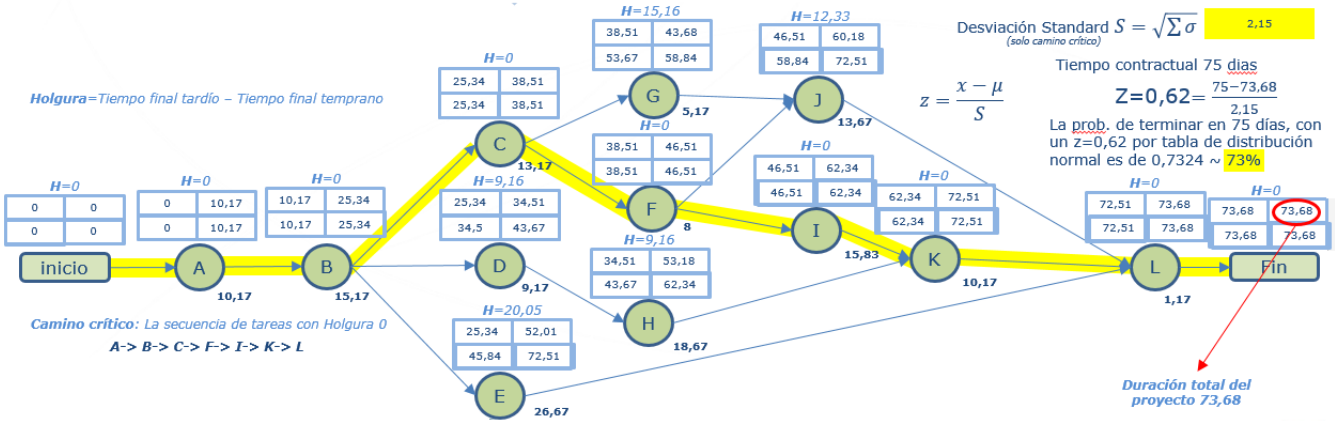
m =Tiempo Probable

b =Tiempo Pesimista

$$Te = \frac{a + 4m + b}{6}$$

$$\sigma = \frac{(b-a)^2}{36}$$

Desviación Standard $S = \sqrt{\sum \sigma}$



Recorridos de Búsqueda

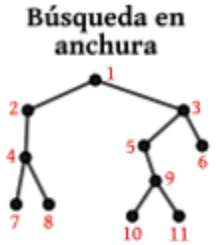
Una de las operaciones más importantes para realizar dentro de las estructuras de grafos es la búsqueda de un determinado vértice. Por ejemplo, imagine en un mapa de ciudades y las carreteras que las unen si queremos identificar qué ciudades pueden alcanzarse a partir de una ciudad dada. Algunas ciudades podrán ser visitadas mientras que otras no.
Supongamos que creamos un grafo para modelar la situación anterior. Ahora necesitamos un algoritmo que pueda comenzar la búsqueda desde un nodo de origen y recorriendo las aristas moverse a otros vértices de modo tal que, una vez que finalice el recorrido, se puedan identificar los nodos que fueron alcanzados desde el origen y aquellos que no.
Para resolver esta problemática existen 2 soluciones distintas:

Búsqueda en anchura (BFS, Breadth first search)
Búsqueda en profundidad (DFS, Depth first search)

Si bien estos 2 algoritmos van a encontrar todos los vértices que se encuentran conectados al vértice origen, la forma en que recorren el grafo para resolverlo es distinta. La búsqueda en anchura recorre el grafo a lo ancho, es decir, recorre todos los adyacentes al vértice origen antes de seguir procesando los siguientes adyacentes. Este algoritmo utiliza una cola como estructura auxiliar para su implementación.
La búsqueda en profundidad recorre en grafo de manera tal que va entrando en los distintos niveles de profundidad de un grafo antes de recorrer toda la lista de adyacentes. Este algoritmo utiliza una pila como estructura auxiliar para su implementación.

Búsqueda en Anchura (BFS)

En Ciencias de la Computación, Búsqueda en anchura (en inglés BFS - **Breadth First Search**) es un algoritmo para recorrer o buscar elementos en un grafo (usado frecuentemente sobre árboles). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.



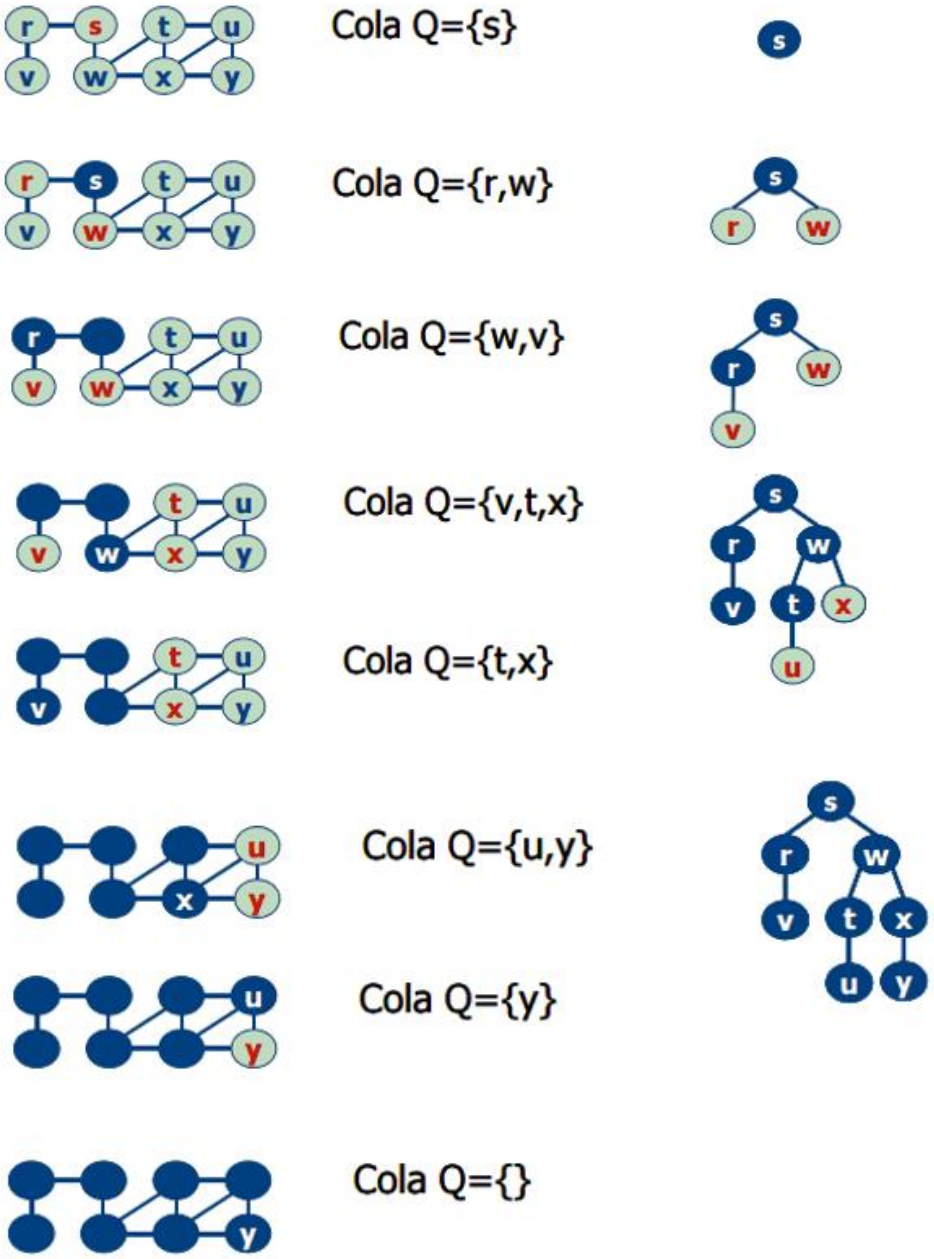
permite recorrer todos los vértices de un árbol de manera ordenada, recorriendo primero los vértices vecinos al inicial, luego los vértices vecinos a los recorridos en el paso anterior y así sucesivamente hasta agotar la gráfica.

Formalmente, **BFS** es un **algoritmo de búsqueda sin información**, que expande y examina todos los nodos de un árbol sistemáticamente para buscar una solución. El algoritmo no usa ninguna estrategia heurística.

Algoritmo:
Dado un vértice fuente S, Breadth-first search sistemáticamente explora los vértices del G para “descubrir” todos los vértices alcanzables desde S.
Calcula la distancia (menor número de vértices) desde s a todos los vértices alcanzables.
Después produce un árbol BF con raíz en S y que contiene a todos los vértices alcanzables.

El camino desde S a cada vértice en este recorrido contiene el mínimo número de vértices. Es el camino más corto medido en número de vértices.
Su nombre se debe a que expande uniformemente la frontera entre lo descubierto y lo no descubierto. Llega a los nodos de distancia k, sólo tras haber llegado a todos los nodos a distancia k-1.

Ejemplo



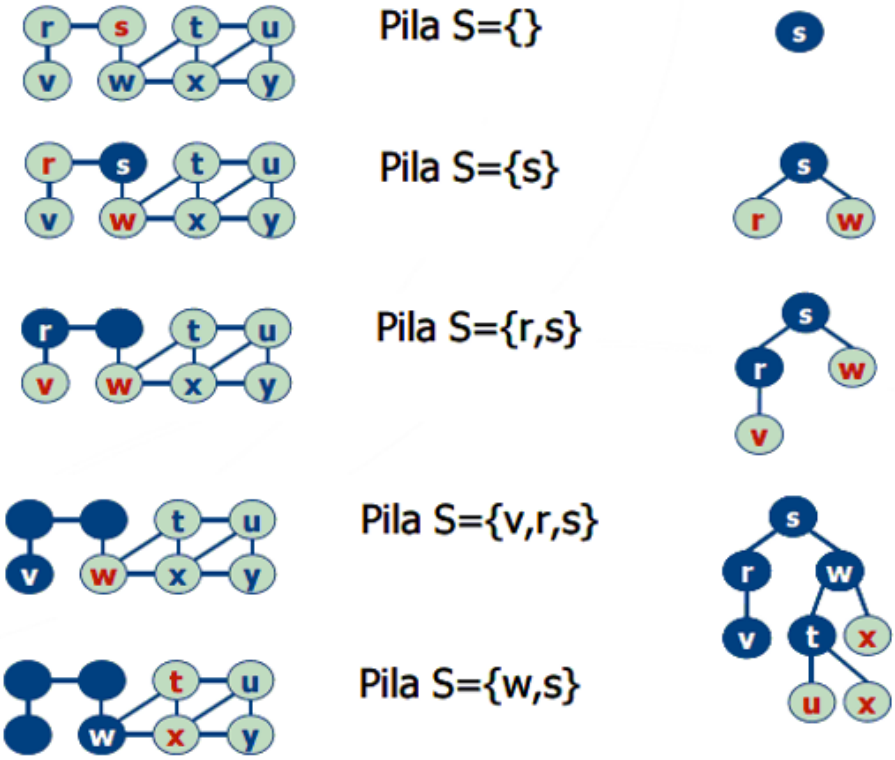
Búsqueda en Profundidad (DFS)

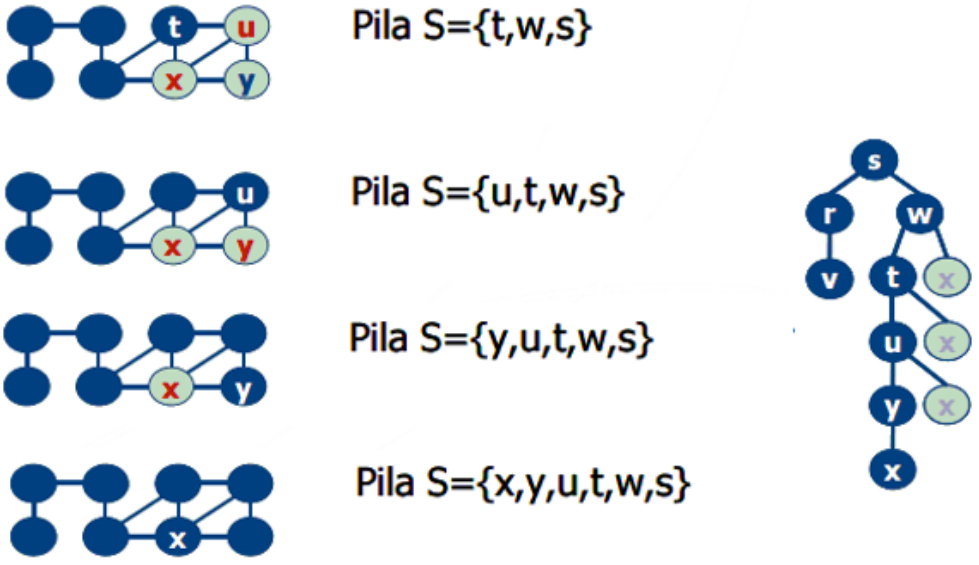
Una Búsqueda en profundidad (en inglés DFS o Depth First Search) es un algoritmo que permite recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (Backtracking), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.

Es necesario llevar la cuenta de los nodos visitados (y no visitados).
El recorrido no es único: depende del vértice inicial y del orden de visita de los vértices adyacentes.
El orden de visita de unos nodos puede interpretarse como un árbol: árbol de expansión en profundidad asociado al grafo.

permite recorrer todos los vértices de un árbol de manera ordenada, avanzando sobre cada rama hasta que no haya posibilidad de continuar y luego se retrocede hasta la última bifurcación para seguir por otra rama. Puede usarse para recorrer un grafo cualquiera si se usa un árbol generador del grafo.

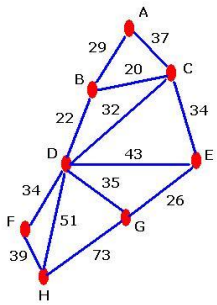
Ejemplo





Problema del Camino más corto

En la teoría de grafos, el problema del camino más corto es el problema que consiste en encontrar un camino entre dos vértices (o nodos) de tal manera que la suma de los pesos de las aristas que lo constituyen es mínima. Un ejemplo de esto es encontrar el camino más rápido para ir de una ciudad a otra en un mapa. En este caso, los vértices representarían las ciudades y las aristas las carreteras que las unen, cuya ponderación viene dada por el tiempo que se emplea en atravesarlas.

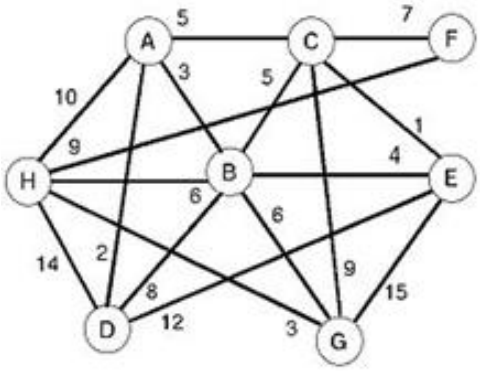


El problema del camino más corto puede ser definido para grafos no dirigidos, dirigidos o mixtos. La siguiente es una definición para grafos no dirigidos, en el caso de grafos dirigidos la definición de camino requiere que los vértices adyacentes estén conectados por una apropiada arista dirigida.

Dos vértices son adyacentes cuando poseen una arista común. Un camino en un grafo no dirigido es una secuencia de vértices $P = (v_1, v_2 \dots v_n) \in VxVxVx \dots xV$ tal que todo vértice v_i es adyacente con el vértice v_{i+1}. Un camino P se dice que es de longitud n si va desde $\boldsymbol{v_1}$ hasta $\boldsymbol{v_n}$.

Sea $e_{i,j}$ la arista incidente con los vértices v_i y v_j. Dada una función de variable real ponderada f: $E \to \mathbb{R}$ y un grafo no dirigido G, el camino más corto desde v hasta v' es el camino $P = (v_1, v_2, \dots, v_n)$ (donde $v_1 = v$ y $v_n = v'$) sobre todos los posibles n que minimiza la suma $\sum_{i=1}^{n-1} f(e_{i,i+1})$. Cuando cada arista en el grafo tiene un peso unitario o $f: E \to \{1\}$, hallar el camino más corto es equivalente a encontrar el camino con menor número de aristas

El problema es también conocido como el problema de los caminos más cortos entre dos nodos, para diferenciarlo de las siguientes generalizaciones:



El problema de los caminos más cortos desde un origen, en el cual tenemos que encontrar los caminos más cortos de un vértice origen v a todos los demás vértices del grafo.
El problema de los caminos más cortos con un destino, en el cual tenemos que encontrar los caminos más cortos desde todos los vértices del grafo a un único vértice destino, esto puede ser reducido al problema anterior invirtiendo el orden.
El problema de los caminos más cortos entre todos los pares de vértices, el cual tenemos que encontrar los caminos más cortos entre cada par de vértices (v, v') en el grafo.

Los algoritmos más importantes para resolver este problema son:

- **Algoritmo de Dijkstra**. *resuelve el problema de los caminos más cortos desde un único vértice origen hasta todos los otros vértices del grafo.*
- **Algoritmo de Bellman – Ford**, *resuelve el problema de los caminos más cortos desde un origen si la ponderación de las aristas es negativa.*
- **Algoritmo de Búsqueda A*,** *resuelve el problema de los caminos más cortos entre un par de vértices usando la heurística para intentar agilizar la búsqueda.*
- **Algoritmo de Floyd – Warshall***, resuelve el problema de los caminos más cortos entre todos los vértices.*
- **Algoritmo de Johnson**, *resuelve el problema de los caminos más cortos entre todos los vértices y puede ser más rápido que el de Floyd-Warshall en grafos de baja densidad.*
- **Algoritmo de Viterbi***, resuelve el problema del camino estocástico más corto con un peso probabilístico adicional en cada vértice.*