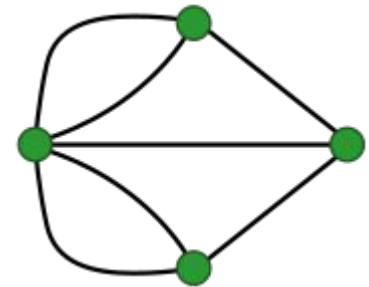
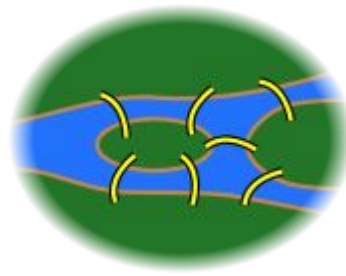
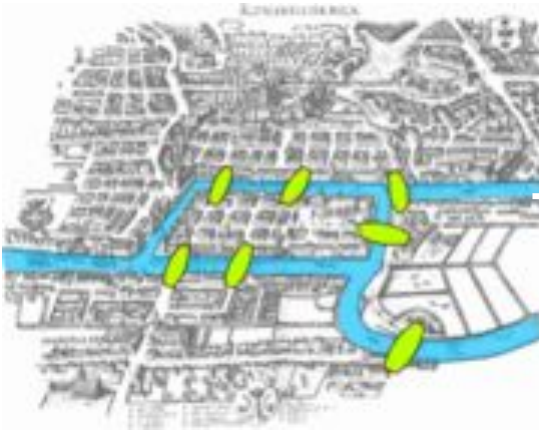


GRAFOS

INTRODUCCIÓN

El problema de *los siete puentes de Königsberg*, es un problema matemático resuelto por **Leonhard Euler en 1736** y cuya resolución dio origen a la teoría de grafos.



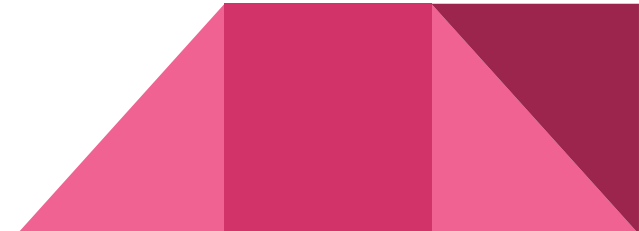
Grafos

Un grafo consiste en un **conjunto** de:

- **nodos** (vértices)
- **arcos** (aristas)

Los arcos establecen relaciones entre los nodos - en general de muchos a muchos.

Utilizaremos la notación $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ para designar al grafo cuyos conjuntos de vértices y aristas son, respectivamente, V y A .



Grafos

sea:

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$

$$A = \{v_1v_2, v_1v_3, v_1v_4, v_2v_4, v_2v_5\}$$

\Rightarrow

$G = (V, A)$ tiene

v_1, v_2, v_3, v_4 y v_5 como vértices

$v_1v_2, v_1v_3, v_1v_4, v_2v_4$ y v_2v_5 sus aristas.

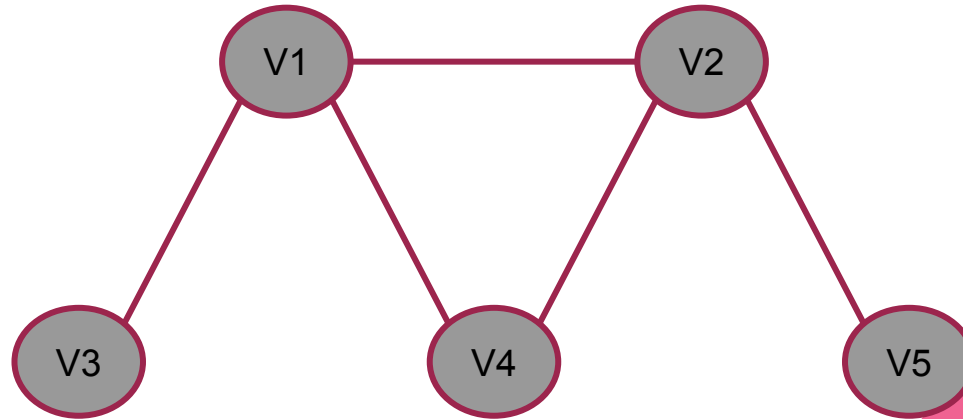
$V_i V_j$: existe comunicación entre v_i y v_j , y entre v_j y v_i



GRAFOS

$V = \{v1, v2, v3, v4, v5\}$

$A = \{v1v2, v1v3, v1v4, v2v4, v2v5\}$

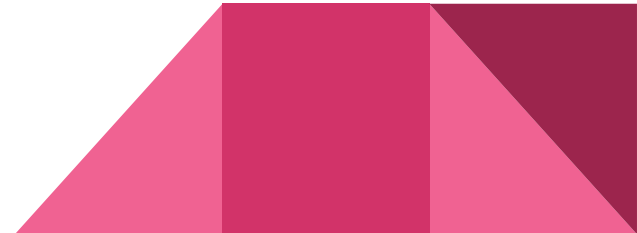


Grafos - Aplicación

Representación del conocimiento: representa el conocimiento en forma de relaciones entre entidades. Los **nodos pueden representar conceptos** o entidades, y las **aristas pueden representar relaciones** entre ellos.

Algunas aplicaciones:

- Redes neuronales
- Aprendizaje automático
- Minería de datos y análisis de redes
- Redes sociales
- Motores de búsqueda
- Sistemas de recomendación
- Rutas y optimización
- Análisis de redes y seguridad
- Bioinformática y genética
- Procesamiento del lenguaje natural



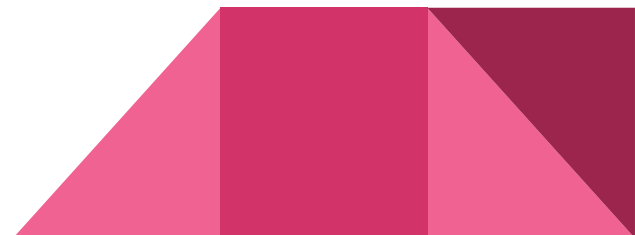
DÍGRAFOS o grafos dirigidos

Lo llamaremos grafo dirigido al par ordenado $G=(V, E)$ donde

- V es el conjunto no vacío de vértices
- E al conjunto de pares ordenados de V . Aristas definidas por tuplas (v,w)

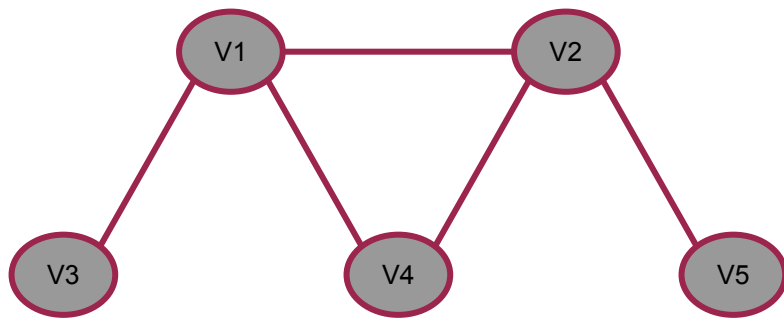
El orden se indica en el diagrama con una **flecha** y llamaremos **origen** o inicial al primer vértice de la arista y **fin** o terminal al segundo.

Permiten **modelar relaciones asimétricas, donde la dirección de la relación importa**. (representar flujos, dependencias, jerarquías, procesos secuenciales)

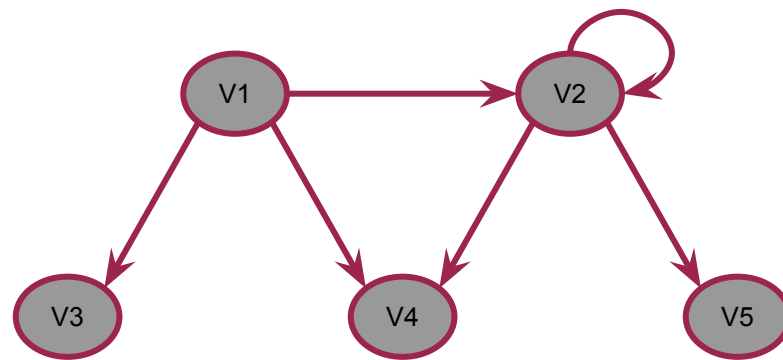


Tipos de grafos

no dirigido



dirigido



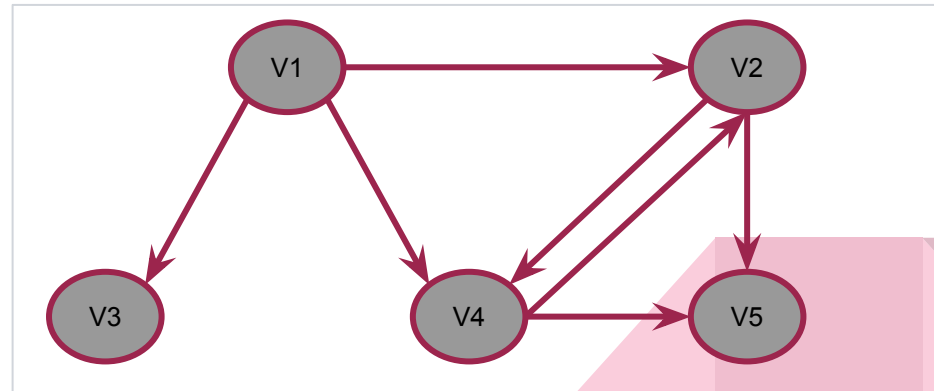
CAMINO DE UN GRAFO

El camino en un grafo G:

Es una secuencia ordenada de aristas que conecta una secuencia de nodos.

En el siguiente ejemplo, los caminos posibles entre los vértices V1 y V5, podrían ser:

- $(v1, v4), (v4, v2), (v2, v5)$
- $(v1, v4), (v4, v5)$
- $(v1, v2), (v2, v5)$
- $(v1, v2), (v2, v4), (v4, v5)$

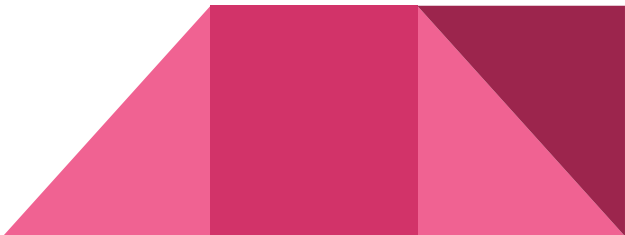


PROPIEDADES DE UN GRAFO

Las propiedades más comunes de un grafo son:

- **Orden:** es el número de nodos o vértices que contiene.
- **Tamaño:** es el número de aristas que contiene.
- **Grado de un nodo** en un grafo:
 - no dirigido - es la **cantidad de aristas que están conectadas a ese nodo**.
 - dirigido - se distingue **entre el grado de entrada** (número de aristas que llegan al nodo) y **el grado de salida** (número de aristas que salen del nodo).
- **Conectividad:** cómo son los caminos entre sus nodos.

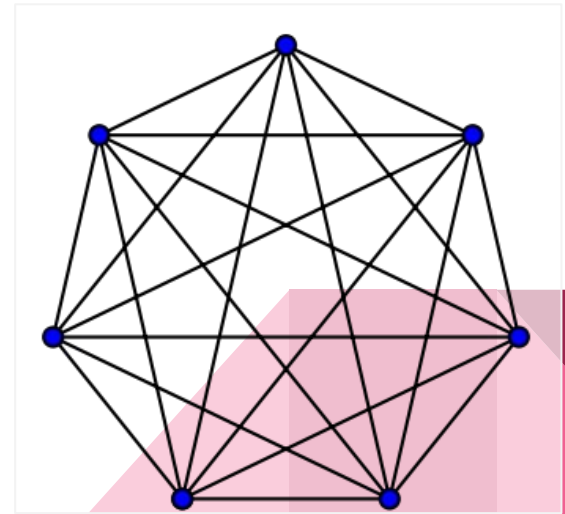
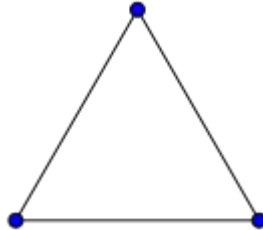
Dependiendo sus propiedades un grafo podrá ser:

- completo
 - conexo
 - denso
 - disperso
- 

Grafo Completo

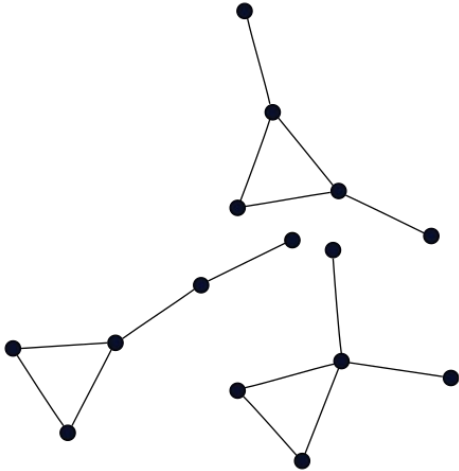
Es completo cuando **todos los nodos del grafo están conectados entre sí** por una arista.

Formalmente, un grafo $G = (V, E)$ se dice que es completo si **para cada par de nodos u, v en V (donde $u \neq v$), existe una arista (u, v) en E .**



Grafos Conexo

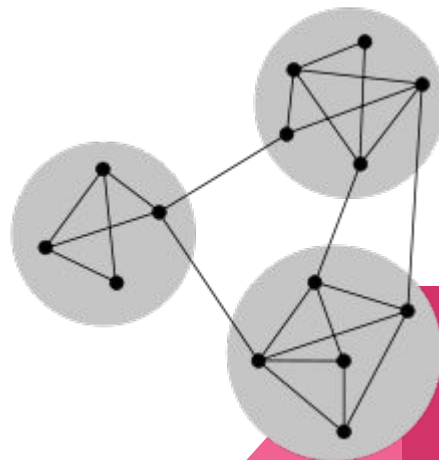
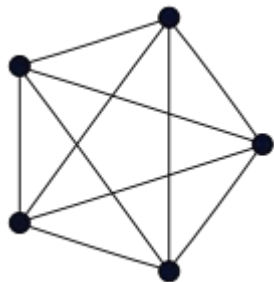
Es conexo **si cada par** de **sus vértices están conectados**. Es decir existe un camino entre ellos.



Grafos densos y dispersos

Un grafo **denso o fuertemente conexo**, es un grafo en el que **el número de aristas está cercano al número máximo de aristas posibles**.

Lo opuesto, un grafo con solo algunas aristas, es un grafo **disperso o débilmente conexo**.

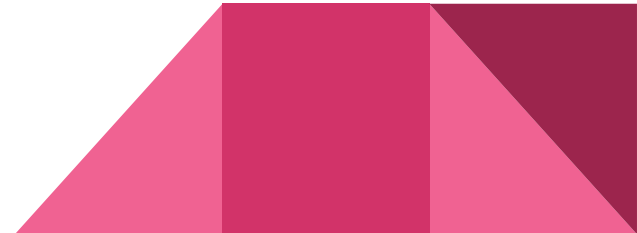


Tipos de implementaciones

Para representar un grafo dentro de la memoria de una computadora podemos usar **estructuras de datos, estáticas, dinámicas** o una combinación de ambas.

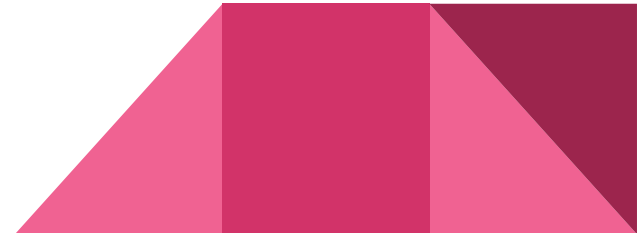
La elección dependerá fundamentalmente de las **características del grafo**.

- Matriz de adyacencia
- Lista de adyacencia
- Lista de aristas
- Matriz de incidencia



Tipos de implementaciones

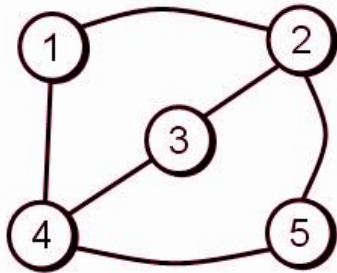
- **Matriz de adyacencia:** Se utiliza una matriz bidimensional para representar el grafo.
- **Lista de adyacencia:** En esta implementación, se utiliza una lista de listas o un arreglo de listas enlazadas para representar el grafo.
- **Lista de aristas:** Se utiliza una lista o un arreglo para almacenar todas las aristas del grafo.
- **Matriz de incidencia:** Utiliza una matriz bidimensional donde las filas representan los vértices y las columnas representan las aristas.



Matriz de adyacencia

Si existe entre sus nodos una conexión directa, se hace explícita a través de un valor distinto a cero.

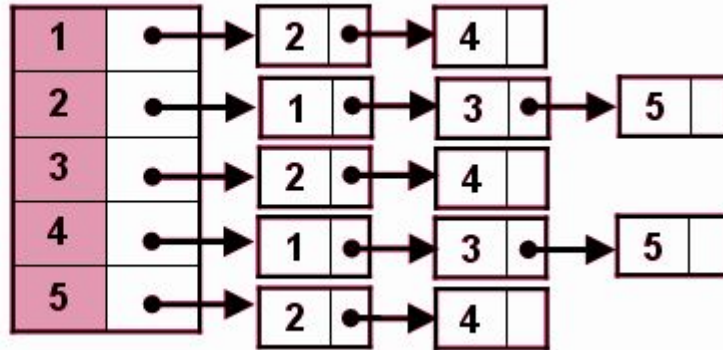
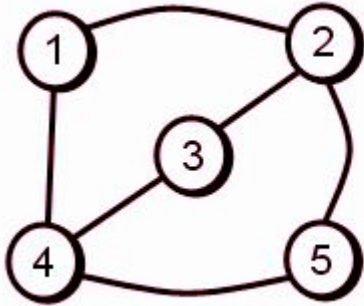
- Acceso eficiente a una arista, (orden 1)
- Espacio de memoria (N^2 , con N = cantidad de nodos) . Ojo! se desperdicia memoria si el grafo es poco denso.



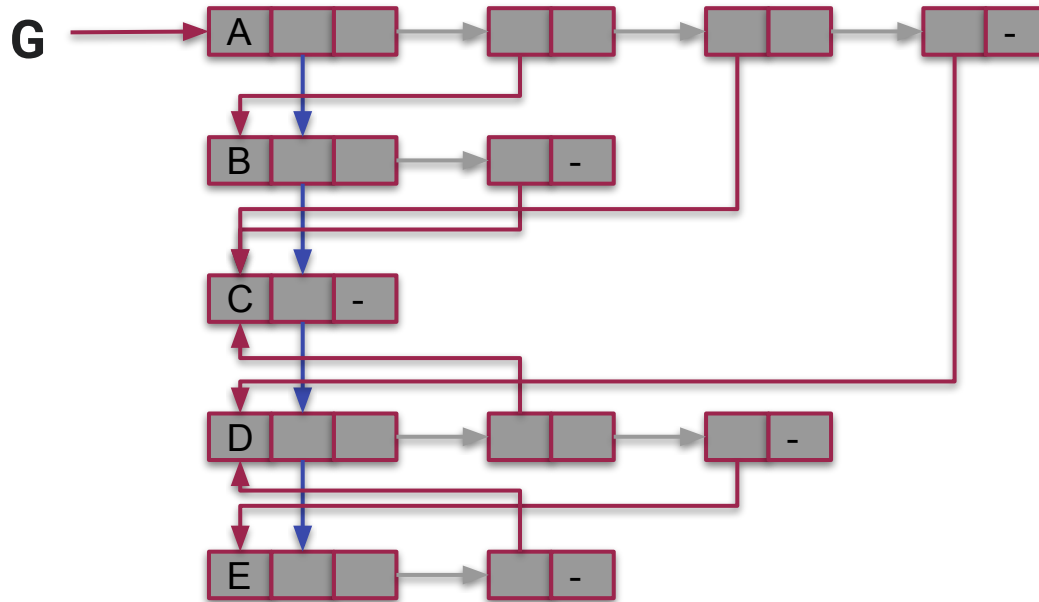
M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Lista de adyacencia

Usamos esta representación si el grafo es disperso o débilmente conexo
(combinamos estática y dinámica)



Lista de adyacencia



Conviene cuando el grafo es disperso y **la cantidad de nodos es variable**.

Lista de aristas - Matriz de incidencias

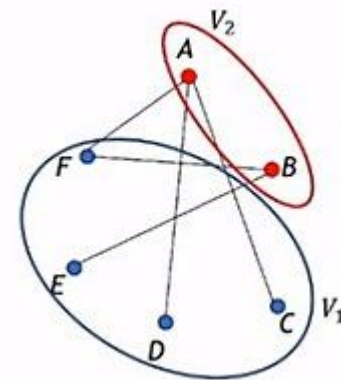


A	1	1	0	0
B	1	0	1	1
C	0	1	1	0
D	0	0	0	1

Otros temas para profundizar

Un concepto importante en la teoría de grafos son los grafos bipartitos.

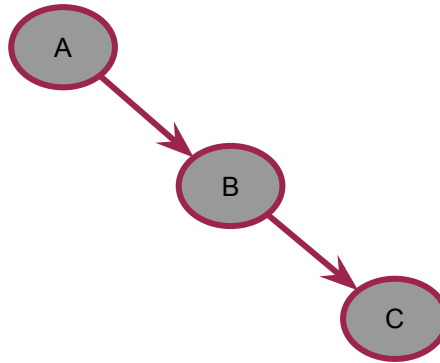
- **Algoritmo de emparejamiento máximo:** busca el conjunto más grande posible de aristas no adyacentes entre sí. Puede ser utilizado para encontrar una asignación óptima de tareas a trabajadores, maximizando el número de asignaciones sin violar restricciones.
- **Algoritmo de emparejamiento perfecto:** busca encontrar un emparejamiento en el que cada vértice esté emparejado con otro vértice sin que haya vértices sin emparejar.
- **Algoritmo de camino aumentante:** Para encontrar emparejamientos adicionales en un grafo bipartito después de encontrar un emparejamiento inicial. Iterativamente, busca caminos alternativos.
- **Algoritmo húngaro** (también conocido como algoritmo de asignación de tareas o **algoritmo Munkres**): Resuelve el problema de asignación óptima en grafos bipartitos completos, donde se busca encontrar una asignación que minimice la suma de los costos asociados con las aristas seleccionadas.



Recorridos de un grafo

Recorrer un grafo significa **tratar de alcanzar** los nodos que estén relacionados con uno que llamaremos **nodo de salida**.

Se parte de un nodo dado y se visitan los vértices del grafo de manera ordenada y sistemática, pasando de un vértice a otro a través de las aristas del grafo.

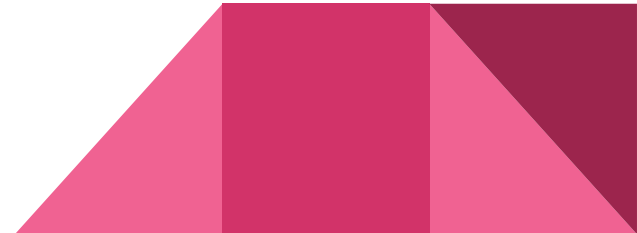


Recorridos de un grafo

Existen básicamente tres técnicas para recorrer un grafo:

- Recorrido en profundidad
- Recorrido en anchura
- Recorrido camino más corto

Recorrer un grafo consiste en visitar todos los vértices alcanzables a partir de uno dado.

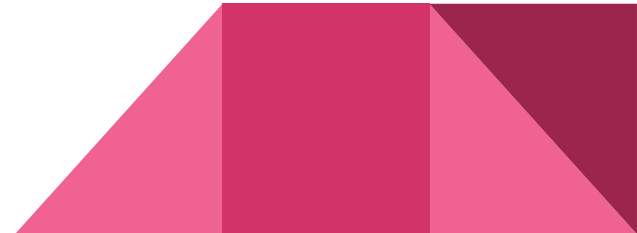


Recorrido en profundidad

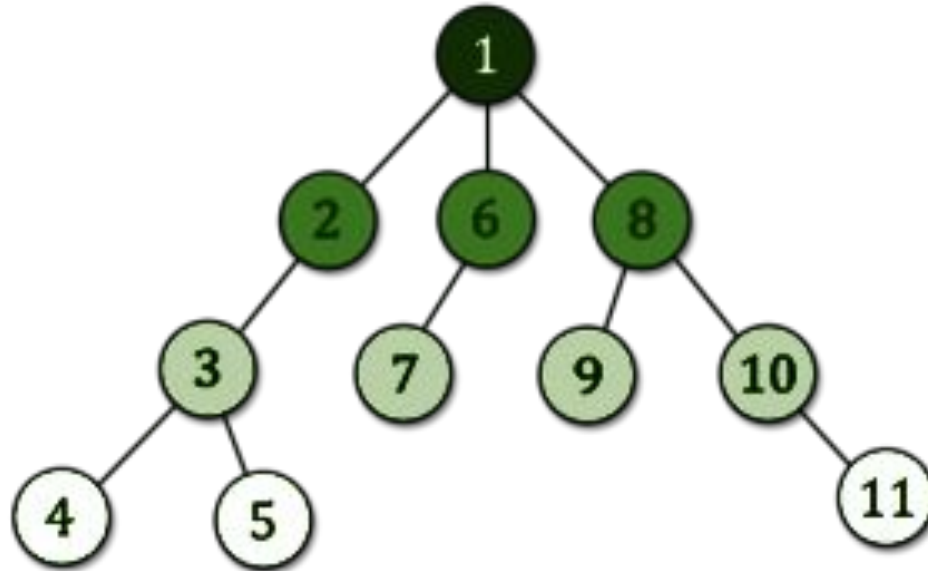
Explora las ramas o aristas del grafo de manera que primero **se visitan los nodos adyacentes a los visitados más recientemente**. De esta manera se va profundizando en el grafo, es decir alejándose progresivamente del nodo inicial.

Equivalente al recorrido en preorden de un árbol.

Para la implementación del algoritmo **utilizamos una pila para ir guardando los nodos adyacentes**.



Recorrido en profundidad



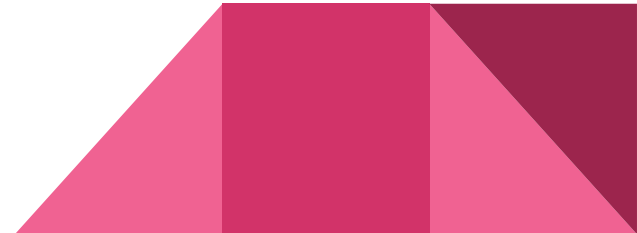
pre-orden : 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11

Recorrido en anchura

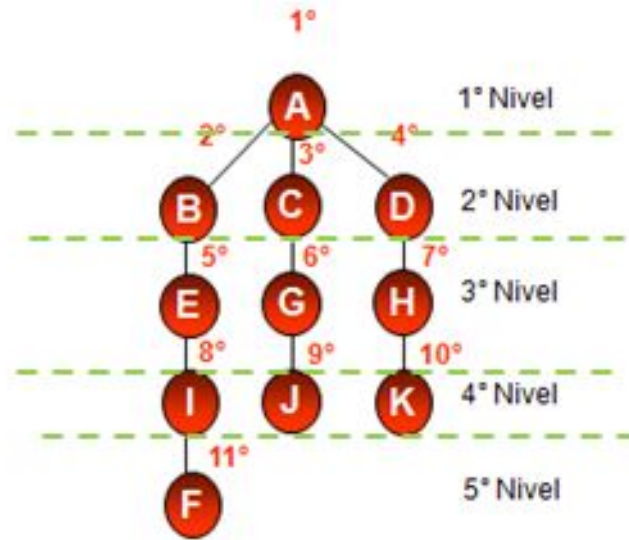
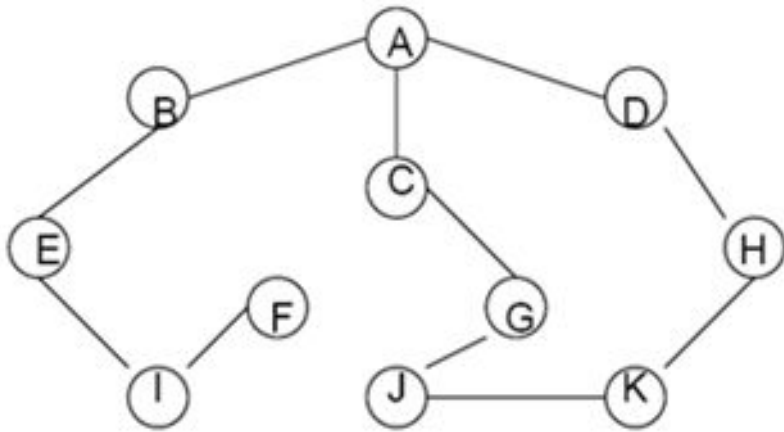
Explora todas las ramas o aristas del grafo de manera que primero se visitan los nodos o vértices más cercanos a un nodo inicial.

Equivalente al recorrido de un árbol por niveles.

Utilizamos una cola para su recorrido.



Recorrido en anchura

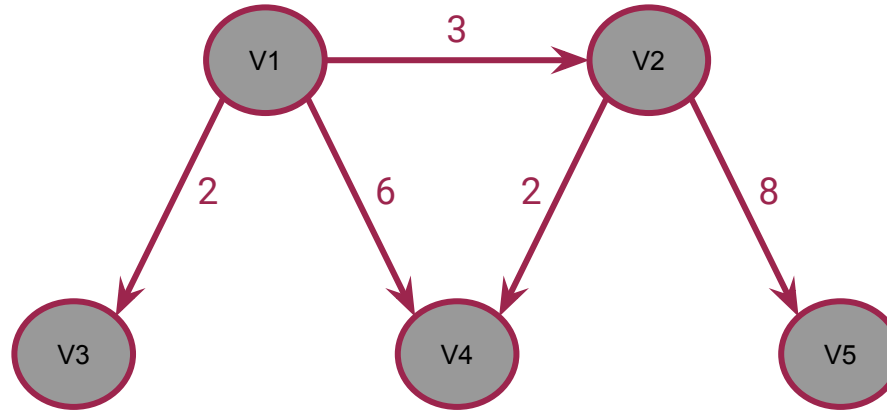


por nivel : A - B - C - D - E - G - H - I - J - K - F

Recorrido camino más corto

Para encontrar el camino óptimo dado un determinado criterio, por ejemplo el más corto, **se utilizan grafos ponderados**.

En este tipo de grafos **existe un valor numérico (peso) asociado a cada arista o arco**.

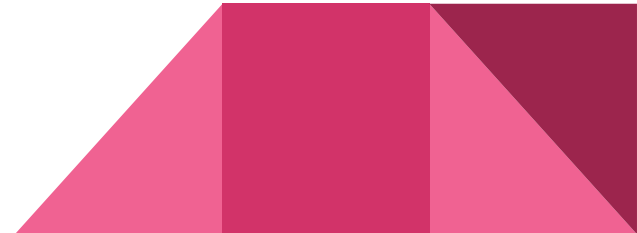


Camino más corto

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.

El camino no necesariamente tiene la menor cantidad de nodos. Algunos algoritmos:

- Algoritmo de Dijkstra (no funciona para valores negativos)
- Algoritmo de Bellman - Ford (si funciona para valores negativos)
- Algoritmo de Búsqueda A*
- Algoritmo de Floyd - Warshall
- Algoritmo de Johnson
- Algoritmo de Viterbi.



Bibliografía

- › Object-Oriented Data Structures (using java) - Nell Dale - Daniel T. Joyce - Chip Weems
- › Data Structures and Algorithms Made Easy. Narasimha Karumanchi.
- › Data Structures and Algorithms. 6ta Ed. Michael T. Goodrich -Roberto Tamassia - Michael H. Goldwasser
- › **Estructuras de datos y algoritmos. M, Weiss**

