GRAFOS

Estructura de datos conformada por **vértices** (nodos) y **aristas**, usado para modelar relaciones entre objetos. Los objetos son los vértices, y las relaciones las aristas.

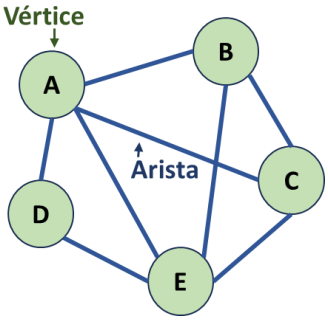


fig. 1. Grafos simple

Vertices = **{A, B, C, D, E}**
Aristas = **{{A,B}, {A,C}, {A,D}, {A,E}, {B,C}, {B,E}, {C,E}, {E,D}}**

Cuando un vértice se conecta a otro con una arista se dice que son **Adyacentes**

Representación de los grafos:

Lista de adyacencia:
Cada vértice tiene una lista de los vértices a los que es adyacente:

“A” = [“B”, ”C”, ”D”, ”E”]
“B” = [“A”, ”C”, ”E”]
“C” = [“A”, ”B”, ”E”]
“D” = [“A”, ”E”]
“E” = [“A”, “B”, ”C”, ”D”]

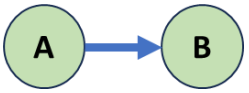
Esta es una representación como lista de adyacencia del grafo de la **fig. 1**

Matriz de adyacencia:
Una matriz cuadrada de tamaño **n**x**n** donde n es el número de vértices. La matriz se llena con 1 donde haya arista (o el número de peso que tenga la arista)

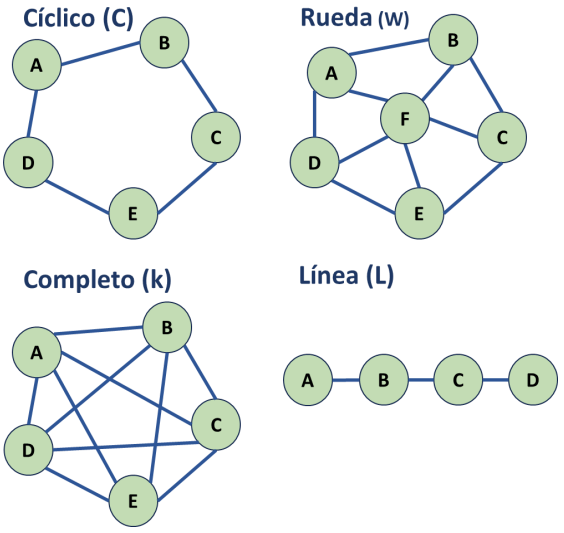
	A	B	C	D	E
A	0	1	1	1	1
B	1	0	1	0	1
C	1	1	0	0	1
D	1	0	1	0	1
E	1	1	1	1	0

Representación del grafo de la **fig.1** como matriz de adyacencia

“En grafos dirigidos la matriz no es simétrica puesto que la relación (A,B) no implica que exista la relación (B,A)”



Información extra:
Tipos de grafos comunes:

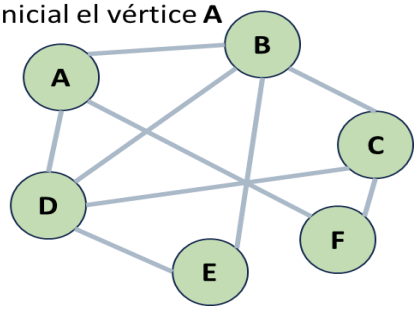


Recorridos de grafos

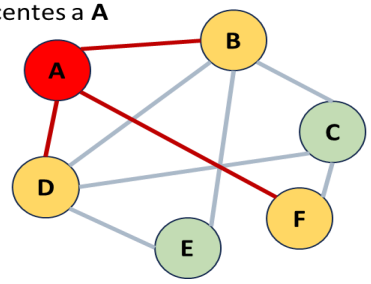
BFS (Breadth-First Search):

Visita todos los vértices adyacentes al vértice principal, luego los adyacentes a cada uno de estos.

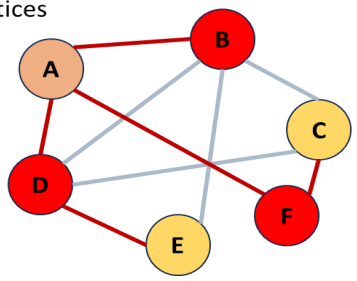
Tomamos como vértice inicial el vértice **A**



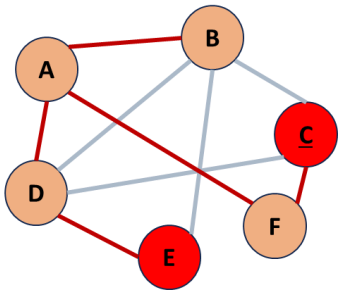
Vamos a cada uno de los vértices adyacentes a **A**



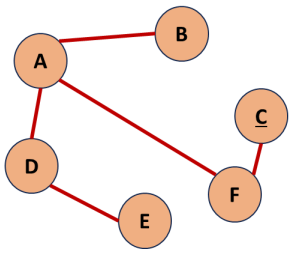
Definimos como principal los nuevos vértices



Vamos a sus vértices adyacentes (Si un vértice es adyacente a dos principales, nos quedamos con solo una arista)



Resultado del recorrido del grafo:

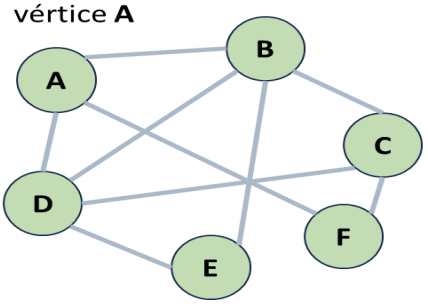


*Puede tener recorridos diferentes **(BFS)** tomando como principal el vértice **A***

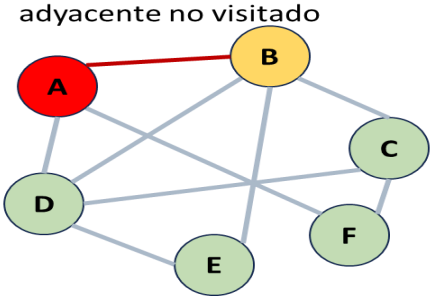
DFS (Depth-First Search):

1.Desde un vértice inicial visita un vértice cualquiera
2.Válida si desde ese vértice se puede ir a otro(s) vértices no recorridos
3.De ser posible va a otro vértice, de lo contrario se devuelve al vértice anterior y repite el paso 2 y 3

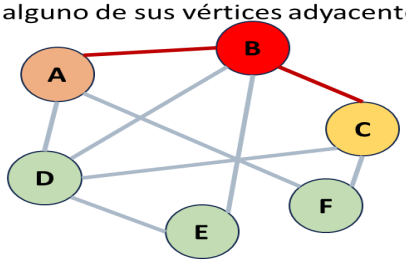
Tomamos como principal el vértice **A**



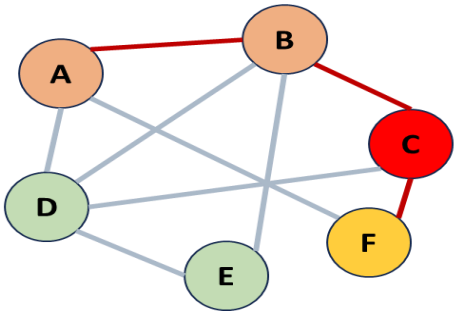
Vamos a cualquier vértice adyacente no visitado



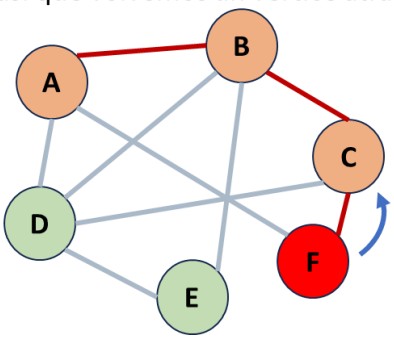
Desde el nuevo vértice vamos a alguno de sus vértices adyacentes



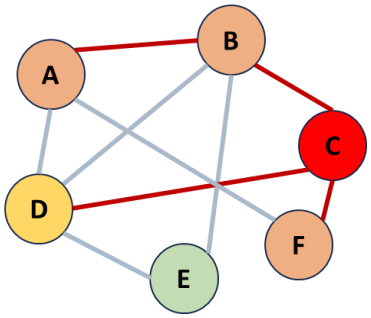
Repetimos el paso anterior y vamos a un nuevo vértice



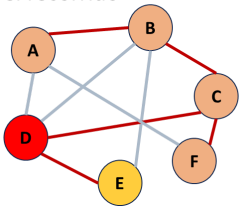
Aun hay vértices sin visitar y no podemos ir desde el vértice actual así que volvemos un vértice atrás

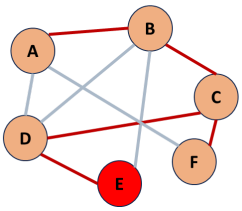


Desde el vértice actual vamos a alguno de sus vértices adyacentes no visitados

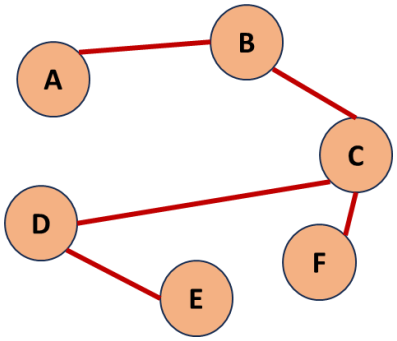


Realizar los pasos anteriores en el vértice actual, cuando no hay mas vértices por visitar, terminamos el recorrido





Resultado del recorrido **DFS**



*"Un grafo puede tener distintas rutas con un mismo vértice principal en un recorrido **DFS**"*

Algoritmo de Dijkstra:

El algoritmo de **Dijkstra** se fundamenta en el recorrido BFS y tiene como objetivo encontrar la ruta más corta en grafos con pesos (Un valor numérico en la arista), facilitando la optimización de tareas.

Inicialización:
Se asigna un valor de costo 0 al vértice origen y a los demás un valor infinito.

Exploración de vecinos:
Desde el vértice actual, se exploran sus nodos adyacentes (similar a BFS), pero **acumulando el costo**:

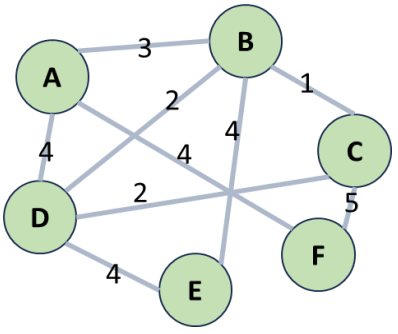
Costo al vecino = costo actual + peso de la arista.

Actualización de costos:
Si el nuevo costo calculado hacia un nodo vecino es **menor que el que teníamos registrado**, lo **actualizamos** y marcamos ese nodo como candidato para futuras visitas.

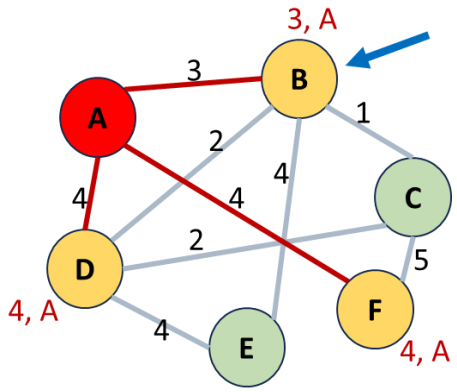
Elección del siguiente nodo:
Se elige el nodo **no visitado** con el **menor costo acumulado** y se repite el proceso.

Finalización:
El algoritmo termina cuando se han visitado todos los nodos.

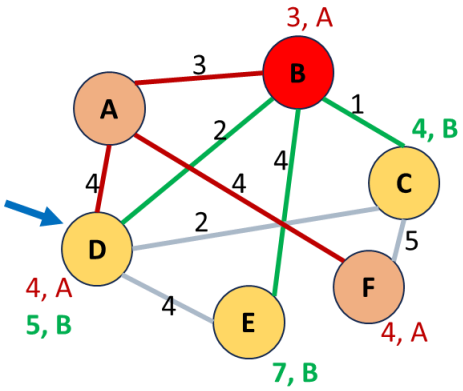
EJEMPLO VISUAL:
Hallar la ruta más corta a todos los vértices desde el vértice **A:**



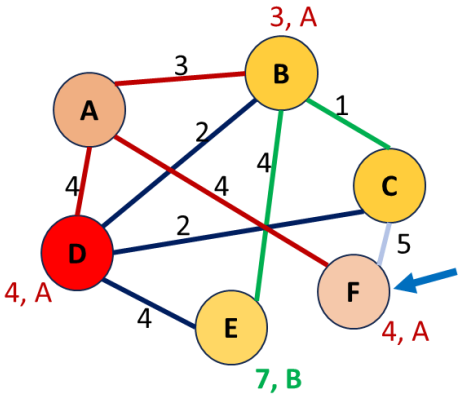
1. Se visitan los vecinos de A



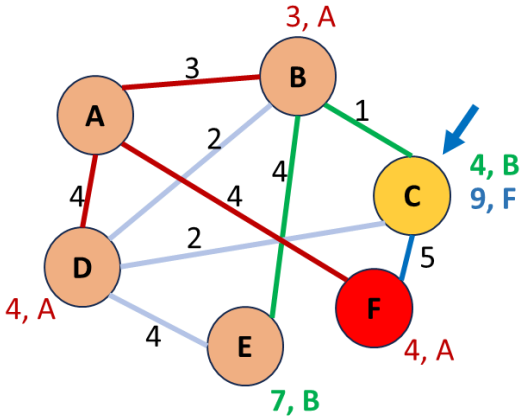
2. Se selecciona B (El de menor peso) y visitar sus vecinos



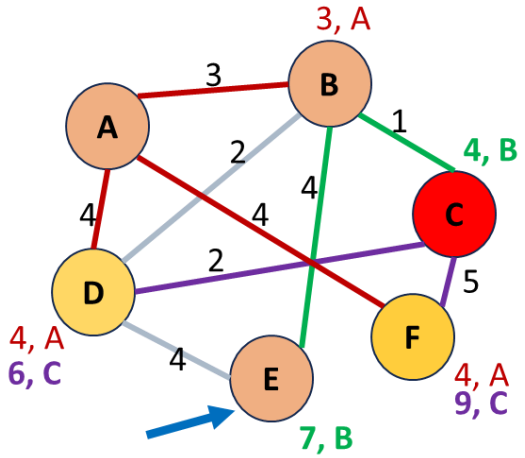
3. Se selecciona D (Peso 4) y visita sus vecinos



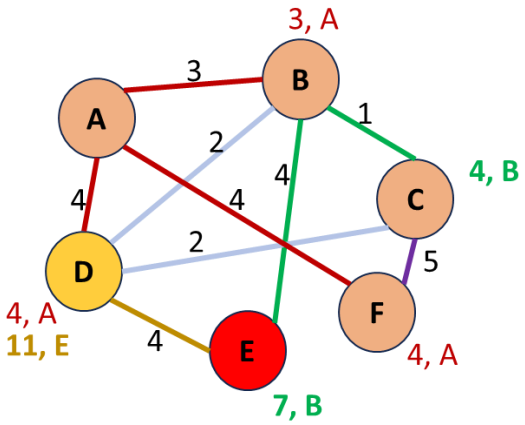
4. Se selecciona F (Peso 4) y visita sus vecinos



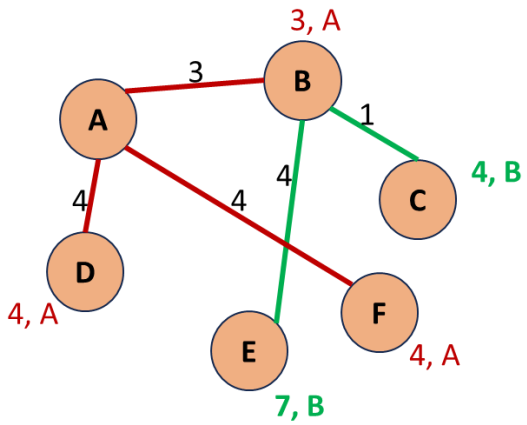
5. Se selecciona C(Peso 4) y visita sus vecinos



6. Seleccionamos el último sin visitar (E) y vamos a validar la ruta con sus vecinos



Resultado del algoritmo:

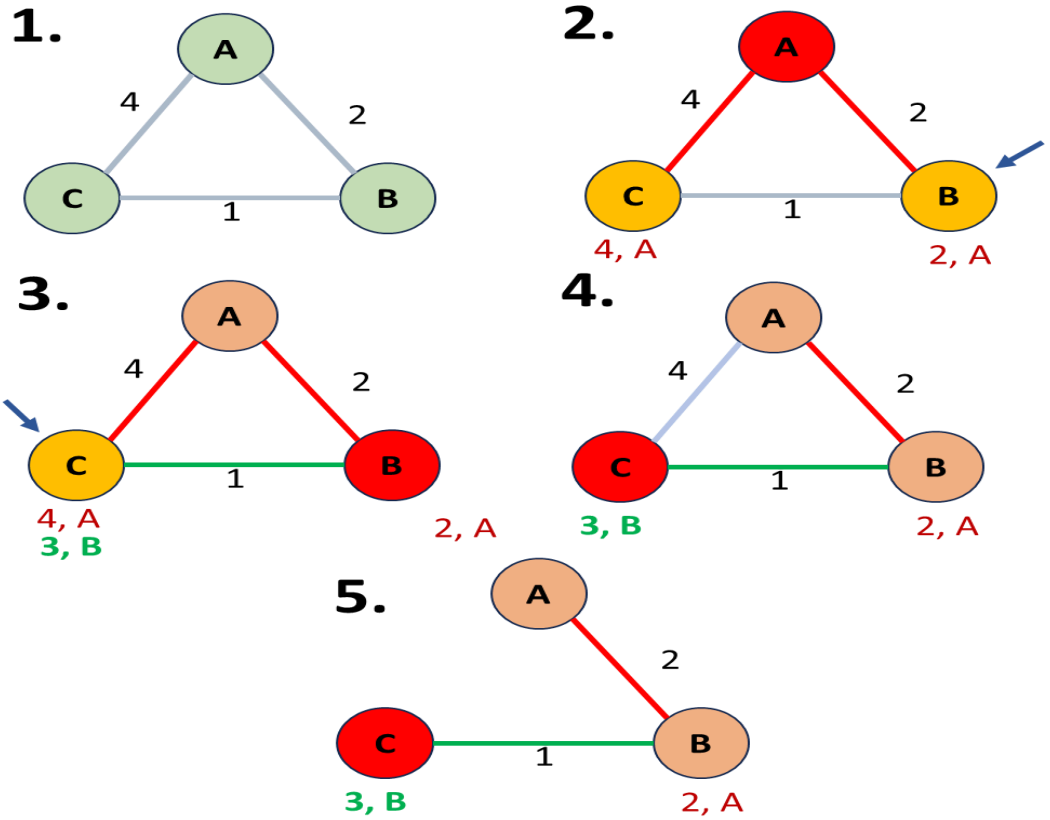


En este grafo se representan las rutas más cortas para ir desde el vértice A hasta cualquier vértice

La ruta más corta para ir hasta los vértices es:
E: *A-B-E*
B: *A-B*
F: *A-F*
D: *A-D*
C: *A-B-C*

En el ejemplo no se han actualizado pesos, puesto que en ninguna actualización hubo una ruta más corta posible desde un nuevo nodo, sin embargo de haber ocurrido, solo hay que actualizar el peso del nodo y su apuntador.

Caso básico demostrativo:



Como se puede observar, el valor de **C** cambia una vez se compara la ruta **desde A** y la ruta **desde B,** puesto que la nueva ruta **(desde B)** tiene un peso menor que el que ya poseía.}

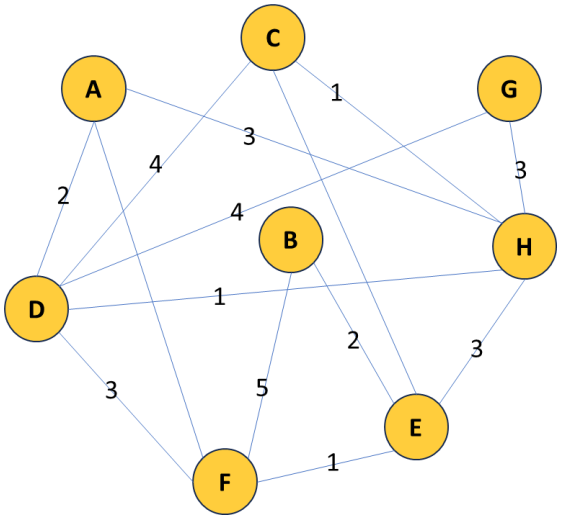
EJERCICIOS

---

1. Se requiere entender el conjunto de estaciones de un tren, lo único que se tiene es la lista de estaciones y las conexiones directas que existen entre ellas, interpretado como grafo cada estación es un vértice y cada arista es una conexión directa de dos estaciones:
Estaciones (V) = (A, B, C, D, E, F, G, H)
Conexiones (E) = ({A,B}, {A,D}, {A,F},{A,H}, {B,E}, {B,F}, {B,G}, {C,D}, {C,E}, {C,F}, {C,H}, {D,E}, {D,F}, {D,G}, {E,F}, {E,H}, {D,H}, {F,G})

   A. **Dibuje el grafo que representa las estaciones del tren y sus conexiones.**
   B. **Haga la matriz de adyacencia del grafo.**
   C. **Haga la lista de adyacencia del grafo.**

2. Una persona quiere ir desde una ciudad A, hasta una ciudad B, tiene distintas opciones de rutas para ir, pero quiere tomar la ruta óptima, así que investigando conoció las carreteras, su estado actual, límite de velocidad permitido y distancia en kilómetros. De lo investigado y tomando en cuenta cada uno de los aspectos, decidió dibujar un grafo con vértices (ciudades) y aristas (rutas) con pesos.



A. Encuentre la ruta que conlleva menos tiempo (de menor peso) para ir desde la ciudad A hasta la ciudad B y escríbala.
Ej. (A-C-D…B)

B. Encuentre la ruta más cara que conlleva menos tiempo desde D hasta cualquier ciudad.
(Así como las rutas cortas encontradas en la explicación de algoritmo de Dijkstra)

//De ser posible dibuje la secuencia de pasos que siguió para encontrar los resultados.

Nota: Facebook es un grafo social en el cual la relación que une cada vértice es la relación de **amistad**, es un **grafo no dirigido** en el cual si eres amigo de una persona la persona también es tu amiga en la red social.
Un ejemplo de un **grafo dirigido** es Instagram, en el cual el que sigas a alguien no implica que esa persona tambien te siga, por ende la relación de **seguir** es una relación dirigida.