1. Grafos

Lista de reproducción YouTube [14].

1.1. Propiedades

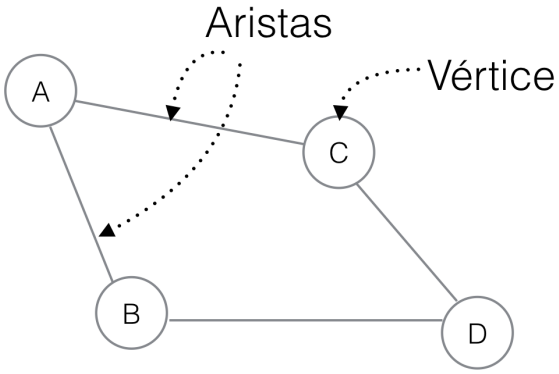


Figura 1: Aristas = Edge y Vértices o Nodos = Vertex

1. **Grafos orientados** (o dirigidos o digrafos) si las aristas (o arcos) que conectan sus vertices (también llamados nodos) están orientadas.
2. **Grafos no orientados** (o no dirigidos) si las aristas que conectan sus vertices no están orientadas.

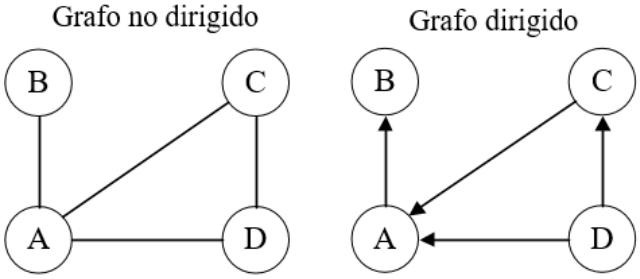


Figura 2: Grafos Dirigido y No Dirigido

3. **Ciclo:** camino que conteniendo vertices distintos, excepto el primero que coincide con el ultimo.

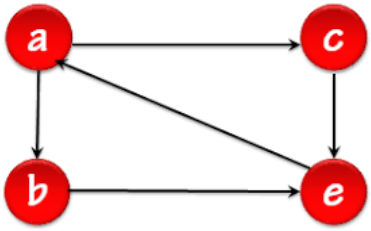


Figura 3: Grafos Ciclo: $a \to b \to e \to a$, longitud 3.

4. **Grafo no dirigido conexo:** Grafo no orientado es conexo si para todo vértice del grafo hay un camino que lo conecte con otro vertice cualquiera del grafo.
5. **Grafo dirigido fuertemente conexo:** Grafo dirigido es fuertemente conexo sii entre cualquier par de vértices hay un camino que los une. Ver Figura 3.
6. **Árbol libre:** Grafo no dirigido conexo sin ciclos.

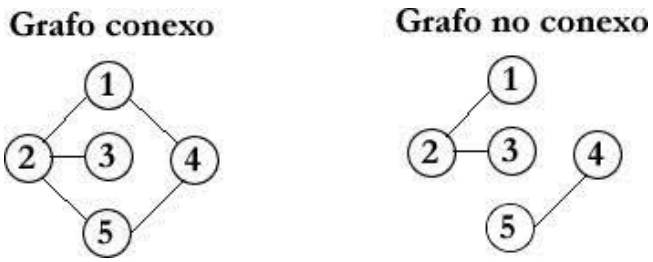


Figura 4: Grafos Conexo y No Conexo

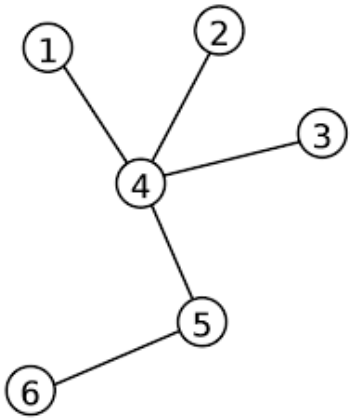


Figura 5: Grafos Árbol Libre

1.2. Estructuras para implementar grafos

1.2.1. Matriz de Adyacencia

Propiedades de la matriz de adyacencia:

- **Grafo no dirigido:** La matriz de adyacencia es simétrica.

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

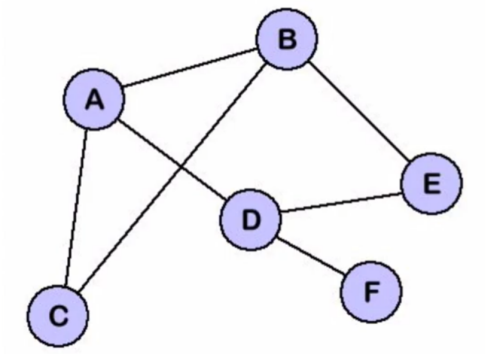


Figura 6: Grafo no dirigido y no pesado Matriz de Adyacencia

	A	B	C	D	E
A	0	3	0	0	0
B	0	0	6	1	5
C	0	0	0	6	0
D	0	0	0	0	7
E	0	0	0	0	0

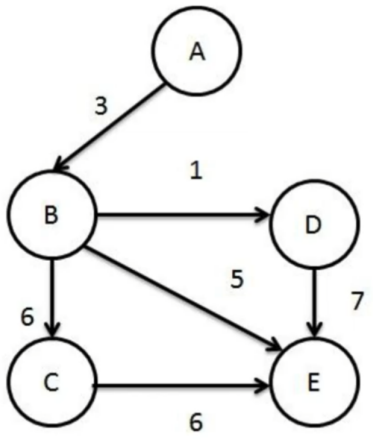


Figura 7: Grafo dirigido y pesado Matriz de Adyacencia

1.2.2. Matriz de Incidencia

	A	B	C	D	E	F
a1	1	1	0	0	0	0
a2	1	0	1	0	0	0
a3	0	1	1	0	0	0
a4	0	1	0	0	1	0
a5	0	0	0	1	1	0
a6	0	0	0	1	0	1
a7	1	0	0	1	0	0

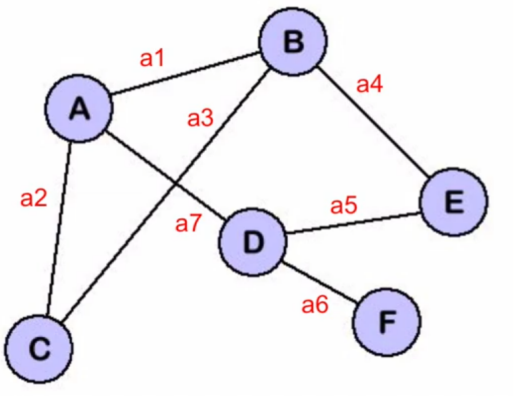


Figura 8: Grafo no dirigido y no pesado

	A	B	C	D	E
a1	-3	3	0	0	0
a2	0	-1	0	1	0
a3	0	-5	0	0	5
a4	0	0	-6	0	6
a5	0	-6	6	0	0
a6	0	0	0	-7	7

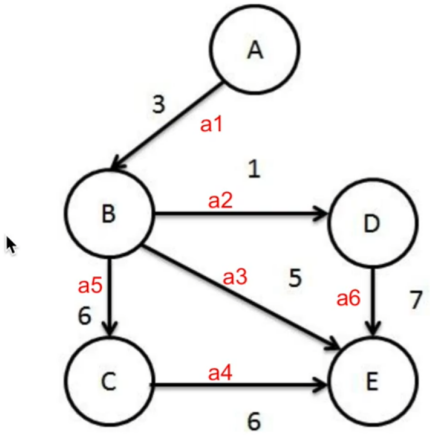


Figura 9: Grafo dirigido y pesado Matriz de Incidencia

1.2.3. Lista de adyacencia

Una **lista de adyacencia** es una representación de todas las aristas o arcos de un grafo en una lista.

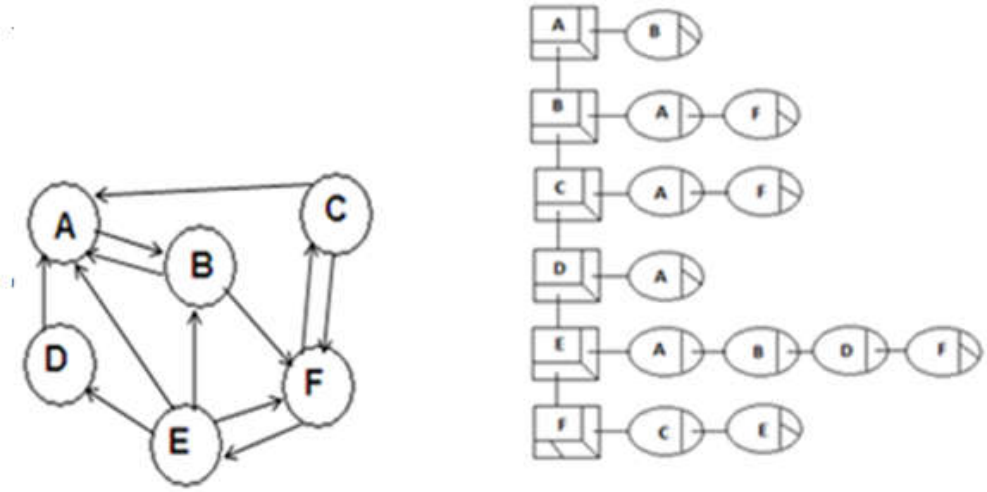


Figura 10: Grafo dirigido Lista de Adyacencia

1.2.4. Complejidad

1.3. Recorridos Grafos

Recorridos Grafos Dirigidos

- Anchura: BFS (Breadth First Search)
- Profundidad: DFS (Depth First Search)

BFS: Se lo puede implementar con una cola (es mas fácil verlo de esta manera).

DFS: Se lo puede implementar con recursividad (es mas fácil verlo de esta manera) o con una pila. Cuando se trabaja con grafos No conexos se puede implementar el algoritmo para que recorra todos los nodos no visitados.

Referencias:

- Video YouTube - Recorrido DFS Recursividad FIUBA [10].

	M. Incidencia	M. Adyacencia	Lista Adyacencia
Espacio:	$O(V \cdot E)$	$O(V^2)$	$O(V + E)$
Agregar un vértice:	$O(V \cdot E)$	$O(V^2)$	$O(1)$ o $O(V)$
Agregar una arista:	$O(V \cdot E)$	$O(1)$	$O(V)$
Si dos vértices son adyacentes:	$O(E)$	$O(1)$	$O(V)$
Obtener los abyacentes de un vértices:	$O(E)$	$O(V)$	$O(V)$

Cuadro 1: Coplejidad.

- Video YouTube - Recorrido BFS y DFS con Pilas y Colas [1].
- Video YouTube - Recorrido BFS y DFS con Pilas y Colas [2].
- Página simula recorrido BFS y DFS tiene errores [4].
- Página simula recorrido BFS y DFS SIN errores [3].

1.3.1. BFS: Recorrido en Anchura

Para grafos dirigidos y no dirigidos.

Procedimento:

- Seleccionar un vértice inicial.
- Marcarlo como visitado.
- Encolarlo.
- Mientras la **cola** (FIFO) no esté vacía :
  - Desencolar vértice.
  - Mostrarlo.
  - Marcar como visitados.
    - Los vertices adyacentes no visitado.
  - Encolar.

1.3.2. DFS: Recorrido en Profundidad

Para grafos dirigidos y no dirigidos. Para implementar el algoritmo suele usarse una **pila**, o bien un algoritmo recursivo.

Procedimento:

- Seleccionar un vértice inicial.
- Marcarlo como visitado.
- Apilarlo.
- Mientras la **pila** (LIFO) no esté vacía :
  - Desapilar vértice.
  - Mostrarlo.
  - Recorrer todos los vértices adyacentes del vértice desapilado
    - Si el vértice adyacente no ha sido visitado, marcarlo como visitado y apilarlo.
    - Si el vértice adyacente ya ha sido visitado, continúa con el siguiente vértice adyacente.

Procedimento Recursivo:
Observando la figura 11. Se comienza el recorrido DFS por algun nodo. Se elige el nodo A:

$$A \rightarrow B \rightarrow E \rightarrow D$$

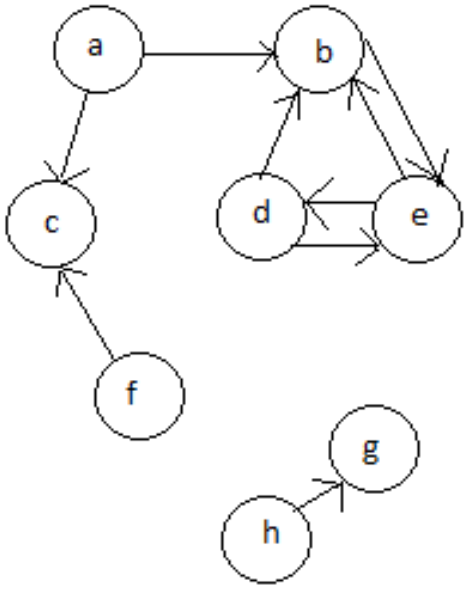


Figura 11: Grafo no dirigido DFS Recursividad

luego volvemos en la recursividad, por que no hay mas nuevos nodos adyacentes a D, hasta llegar a A.

$$A \to B \to E \to D \to C$$

no existe mas camino por donde ir, por lo tanto se elige un nuevo nodo no visitado y se repite el proceso.

$$A \to B \to E \to D \to C \to F$$

elige un nuevo nodo no visitado y se repite el proceso.

$$A \to B \to E \to D \to C \to F \to G$$

elige un nuevo nodo no visitado y se repite el proceso.

$$A \to B \to E \to D \to C \to F \to G \to H$$

1.3.3. Complejidad BFS y DFS

	Matriz Adyacencia	Matriz Incidencia
Anchura BFS	$O(V^2)$	$O(V+E)$
Profundidad DFS	$O(V^2)$	$O(V+E)$

Cuadro 2: Coplejidad.

1.3.4. Aplicaciones

DFS: Recorrido Profundidad

1. **Test de Aciclidad (Ciclos):** Se recorre el grafo teniendo en cuenta los recorridos paciales. Cl coste del algorimos es el mismo que la Tabla 2.
2. **Puntos de Articulación:** Un punto de articulación (o vértice de corte) de un grafo **no dirigido** y **conexo** es un vértice que verifica que al ser eliminado del grafo éste deja de ser conexo.
3. **Obtención de las componentes fuertemente conexas en un grafo dirigido:** Una componente fuertemente conexa es un subgrafo en el que para cada par de vértices existe un camino de uno a otro.

BFS: Recorrido Anchura

1. **Camino mínimo:** Si el grafo es no pesado, el camino mínimo entre dos vértices es el camino que tiene menos aristas.
2. **Árbol de expansión mínimo:** Si el grafo es pesado, el árbol de expansión mínimo es el subgrafo que tiene todos los vértices del grafo original y la suma de los pesos de sus aristas es la mínima posible.

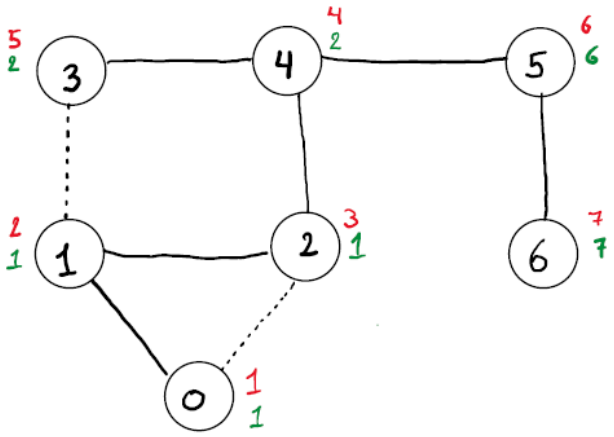
1.3.5. Puntos de Articulación:

Propiedades:

- Si r es raíz es el árbol DFS y tiene más de un hijo en el árbol, entonces r es un punto de articulación.
- bajo(u) = mínimo número asignado en el recorrido en profundidad.
- Para todo vértice u que no sea raíz, es punto de articulación si y sólo si tiene al menos un hijo x tal que bajo(x) $\geq$ numero asignado al vértice u en el recorrido en profundidad.

Procedimento:

1. Recorrer el grafo en profundidad y numerar los vértices en el orden en que se los visita.
2. Asignar el valor bajo(u) a cada vértice u comenzando desde el último hasta el primero de los visitados.



Vértice	Posición	Hijo	Bajo(u)
0	1	1	1
1	2	2	1
2	3	4	2
4	4	5	2
3	5	-	-
5	6	7	7
6	7	-	-

Figura 12: Puntos de articulación.

1.4. Problemas sobre caminos en un grafo

- **Camino más corto entre un vértice y todos los demás:** Consiste en determinar el costo del camino más corto desde el vértice considerado origen a todos los otros vértices. Para grafos dirigidos y no dirigidos con aristas ponderadas no negativas.
  - **Estrategia greedy:**
- **Camino más corto entre todos los pares de vértices:** Si el grafo es no pesado, el camino más corto entre todos los pares de vértices es el camino que tiene menos aristas.
  - **Programación Dinámica:**

1.4.1. Árbol de expansión mínimo

- **Árbol de expansión mínimo:** Si el grafo es pesado, el árbol de expansión mínimo es el subgrafo que tiene todos los vértices del grafo original y la suma de los pesos de sus aristas es la mínima posible.

1.4.2. Algoritmo de Dijkstra

- **Algoritmo de Dijkstra:** El algoritmo de Dijkstra es un algoritmo para la determinación del camino más corto desde un vértice origen, hacia el resto de los vértices en un grafo dirigido no dirigidos que tiene pesos (NO NEGATIVOS) en cada arista.

1.4.3. Algoritmo de Floyd

- **Algoritmo de Floyd:** El algoritmo de Floyd es un algoritmo para la determinación del camino más corto entre todos los pares de vértices en un grafo DIRIGIDOS que tiene pesos (NO NEGATIVOS) en cada arista.

Ver video YouTube [11].
Se usa una matriz de adyacencia que contiene ceros y unos donde el cero significa que no hay camino y el uno significa que hay camino.

1.4.4. Algoritmo de Warshal - Cerradura transitiva

- **Algoritmo de Warshall:** El algoritmo de Warshall permite determinar qué pares de vertices están enlazados entre si por algún camino, sin importar la longitud (peso en la arista) de éste. Es para grafos dirigidos y no ponderados.

Ver video YouTube [12].

1.5. Algunos problemas sobre Grafos no dirigidos

Un **árbol libre** es un grafo no dirigido conexo sin ciclos.

Se verifica que

- En todo árbol libre con N vértices ($N > 1$), el árbol contiene $N - 1$ aristas.
- Si se agrega una arista a un árbol libre, aparece un ciclo.
- Si u y v son dos vertices distintos de un árbol, entonces hay un solo camino que los une.

Algoritmos que permiten obtener el árbol de expansión de coste mínimo

- **Algoritmo de Prim:** El algoritmo de Prim es un algoritmo para la determinación del árbol de expansión mínimo de un grafo no dirigido, conex y con aristas no negativas.
- **Algoritmo de Kruskal:** El algoritmo de Kruskal es un algoritmo para la determinación del árbol de expansión mínimo de un grafo conexo y no dirigido.