* Un grafo es: Un conjunto de aristas, y un conjunto de vértices no vacíos.
* Si las aristas tienen dirección🡪**Grafo Dirigido**.
* Si las aristas no tienen dirección🡪**Grafo No Dirigido**.
* **Grafo Pesado**: Es un grafo en el que las aristas tienen un peso asignado. Puede ser dirigido o no dirigido.
* **Grado De Salida**: Es la cantidad de aristas que salen de un vértice.
* **Grado De Entrada**: Cantidad de aristas que llegan a un vértice.
* Un **camino** es cuando se puede llegar de un Vértice a otro.
* **Longitud Del Camino**: La cantidad de Nodos por los que tengo que pasar para llegar a otro.
* **Ciclo**: Camino donde el primer y último vértice coinciden.
  + Para Dígrafo: Camino de longitud mayor que 1, que empieza y termina con el mismo vértice.
  + Para Grafo: Ciclo de longitud de al menos 3.
  + Acíclico: No tiene ciclos.
  + Para solucionar el problema de los ciclos, guardo la estructura donde ya se los que visite en un conjunto para que no se repita, y voy preguntando si ya se encuentra en el conjunto o no.
* **Grafo Conexo**:
  + Para Dígrafo:
    - * Fuertemente Conexo: Si existe un camino para cada par de vértices.
      * Débilmente Conexo: Si existe un camino para cualquier par de vértices sin tener en cuenta la orientación de los arcos.
  + **Para Grafo**: Si existe camino entre cada par de vértices.
* **Grafo Completo**: Si existe una arista entre cada par de vértices.
* **Componente fuertemente conexa**: Es un conjunto de vértices en el que hay un camino desde cualquier vértice en el conjunto a otro, del conjunto.
* **Recorridos**:
  + Búsqueda En Profundidad DFS:

Tendremos una estructura para saber que vértices ya fueron visitados.

Se selecciona un vértice START como vértice de partida.

El START se marca como visitado, y luego recorre cada vértice no visitado, adyacente a START.

Una vez recorrido todos los adyacentes a START, se selecciona u nuevo vértice y se repite el proceso para los que aún faltan visitar.

* + Búsqueda En Amplitud BFS:

A partir de un vértice START, empieza a recorrer sus adyacentes, una vez visitado todos los adyacentes, se empieza a recorrer los adyacentes de los adyacentes a START.

* Camino Mas Corto🡪Algoritmo de Dijkstra:
* Para crear un grafo necesitamos los vértices y sus adyacentes.
* No todos los vértices tienen que estar conectados.
* Representar **en MATRIZ** tiene **la desventaja** de desperdiciar lugar, y más **si el grafo es no dirigido.** Pero tiene **la ventaja** de ver si un nodo es adyacente a otro ya que tiene acceso directo a cada nodo, a diferencia de la lista adyacencia, que hay que recorrer todos los nodos para saber si llega o no al nodo requerido.
* Una **ventaja de la lista** es que si quiero ver todos los adyacentes de un nodo, solo recorro los adyacentes, mientras que en la matriz tengo que pasar nodo por nodo viendo si es o no adyacente.
* Para saber **si dos nodos son adyacentes** es mejor la matriz porque tiene O(1). Si el grafo tiene pocas conexiones es mejor usar la lista. Con la lista tenemos O(n)
* Otra opción sería usar un diccionario, donde la clave es el nodo, y el valor una lista de adyacentes.
* En la matriz cada posición tiene un valor Bool o Entero, mientras que la lista contiene el valor y el puntero. Pero al matriz tiene N^2 Entradas, mientras que en la lista el número de entradas es el número de aristas del grafo (En general menor a N^2) 🡪La lista requiere menos espacio.