INFORME TRABAJO PRÁCTICO

PROGRAMACION 3

|  |  |  |  |
| --- | --- | --- | --- |
| Alumnos | Camicha, Nicolas | Corrionero,Lisandro | Samore, Matias |
| LU | 1101634 | 99440 | 1034944 |

Profesor: Rodriguez, Guillermo Horacio



Índice

[Algoritmos Implementados: 2](#_Toc64491237)

[Desarrollo: 2](#_Toc64491238)

[Ejemplo de grafo de entrada de DFS: 3](#_Toc64491239)

[Ejemplo de grafo de entrada de BFS: 5](#_Toc64491240)

[Ejemplo de grafo de entrada de Floyd: 9](#_Toc64491241)

[Ejemplo de grafo de entrada de Kruskal: 11](#_Toc64491242)

# Algoritmos Implementados:

Los algoritmos realizados son:  
- DFS y BFS.  
- Floyd.  
- Kruskal

**DFS:**  Un Recorrido en profundidad (en inglés DFS o Depth First Search) es un algoritmo que permite recorrer todos los nodos de un grafo. Partimos desde un vértice fuente explorando recursivamente sus sucesores, y mostrara el conjunto de nodos desde los sucesores hasta el origen seleccionado para la búsqueda.

**BFS:** Un Recorrido en anchura (BFS o BREADTH FIRST SEARCH) es un algoritmo que recorre a lo ancho o por niveles. BFS devuelve el camino más corto desde el origen.

**Floyd:** Obtención del camino más corto entre cada par de vértices de un grafo rotulado (puede ser dirigido o no dirigido).

Floyd obtiene una tabla que brinda el costo menor requerido para ir de un vértice a otro.

**Kruskal:** Comienza con un bosque cuyos árboles son los vértices del grafo.

En cada paso, selecciona la arista de mínimo costo que conecta dos árboles distintos.

Termina cuando el bosque tiene un solo árbol.

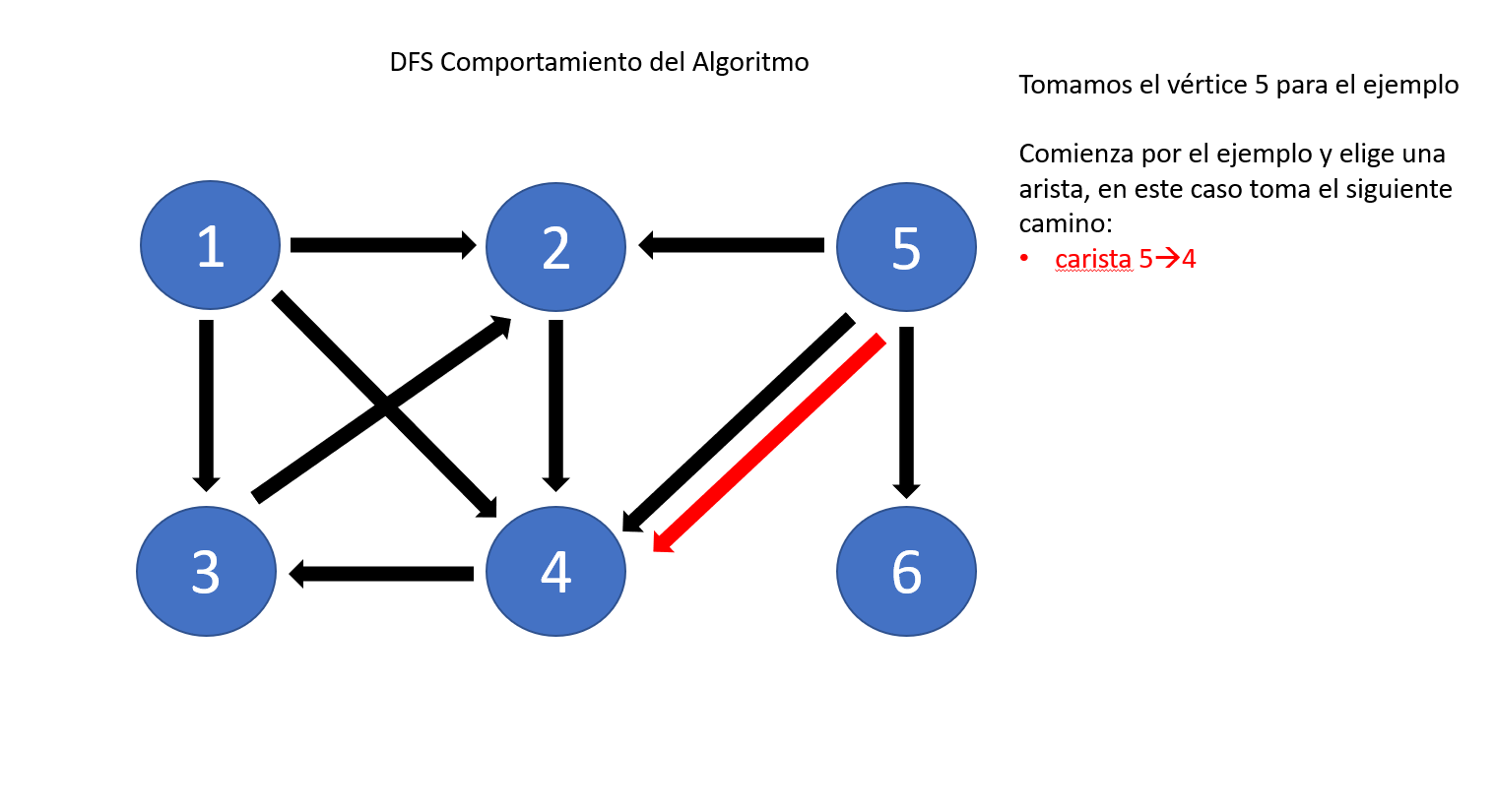
# Desarrollo:

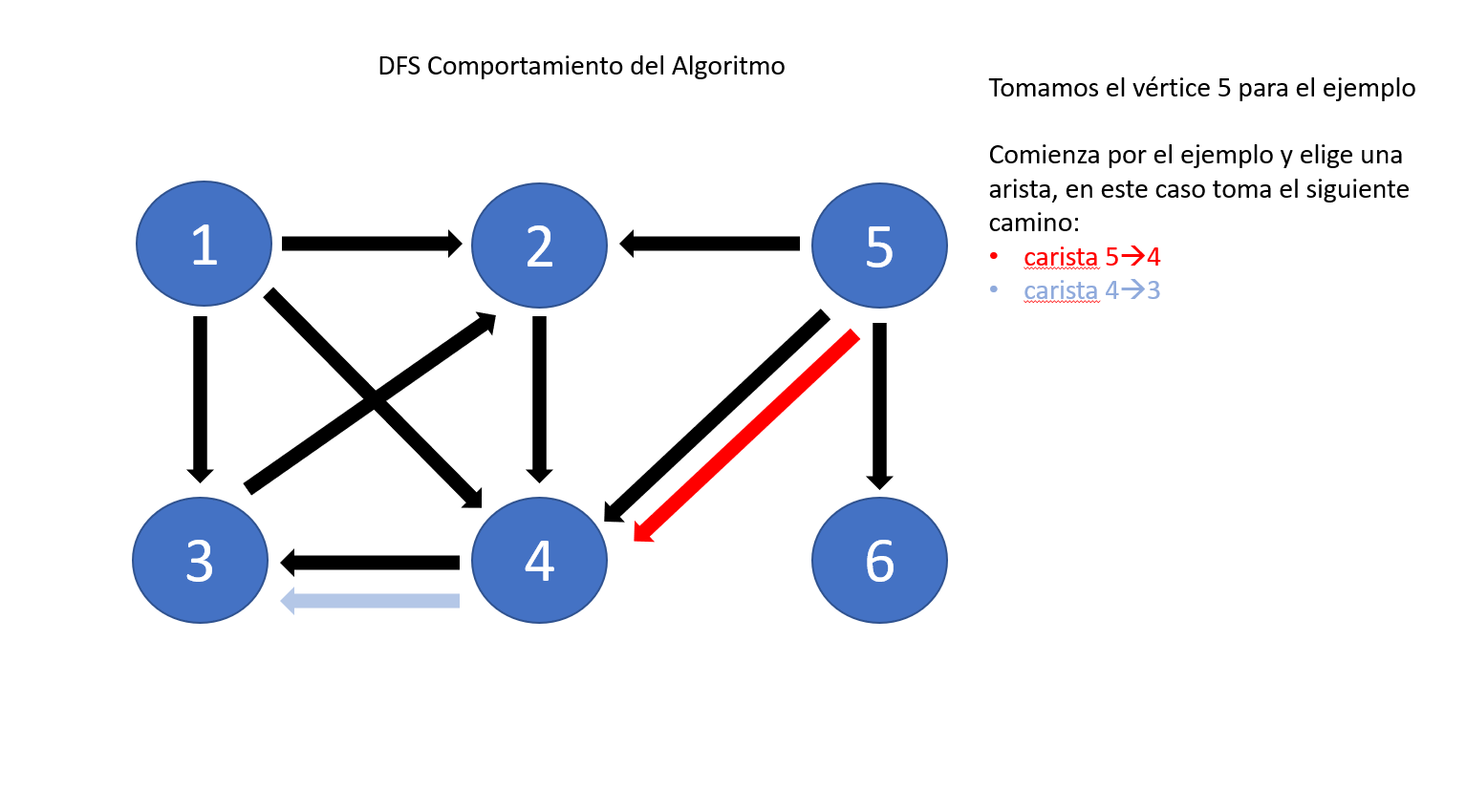
DFS: Generamos un grafo. Agregamos vértices y aristas entre los vértices, luego le pedimos al usuario que ingrese un vértice donde vamos a almacenar nuestro punto de origen para empezar a recorrer el grafo por anchura.

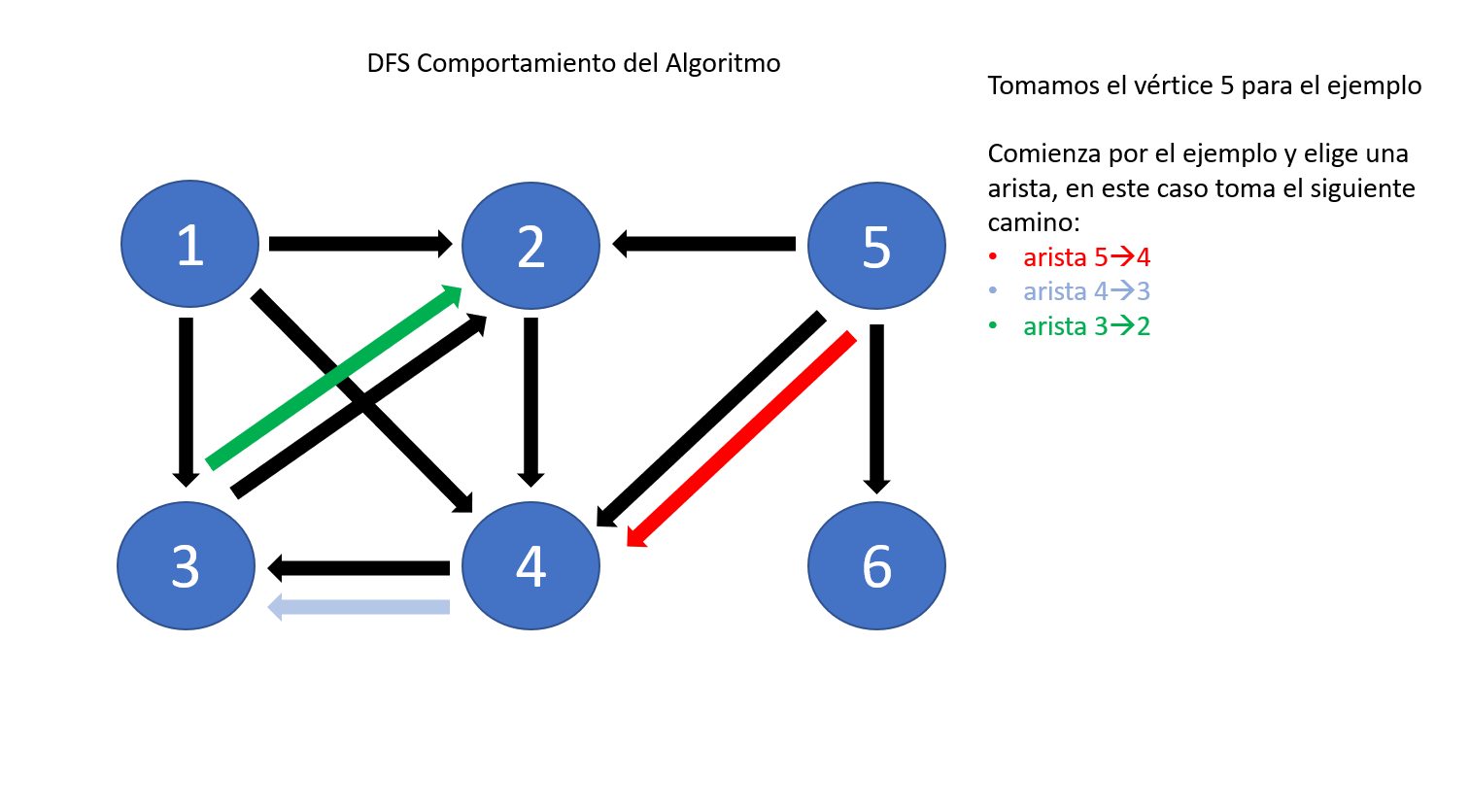
BFS: Lo que hicimos fue crear un grafo. Agregando vértices y aristas entre los vértices, luego le pedimos al usuario que ingrese un vértice donde vamos a almacenar nuestra partida (por donde empezamos a iterar) explorando a sus sucesores. El recorrido en este caso se hace por profundidad.

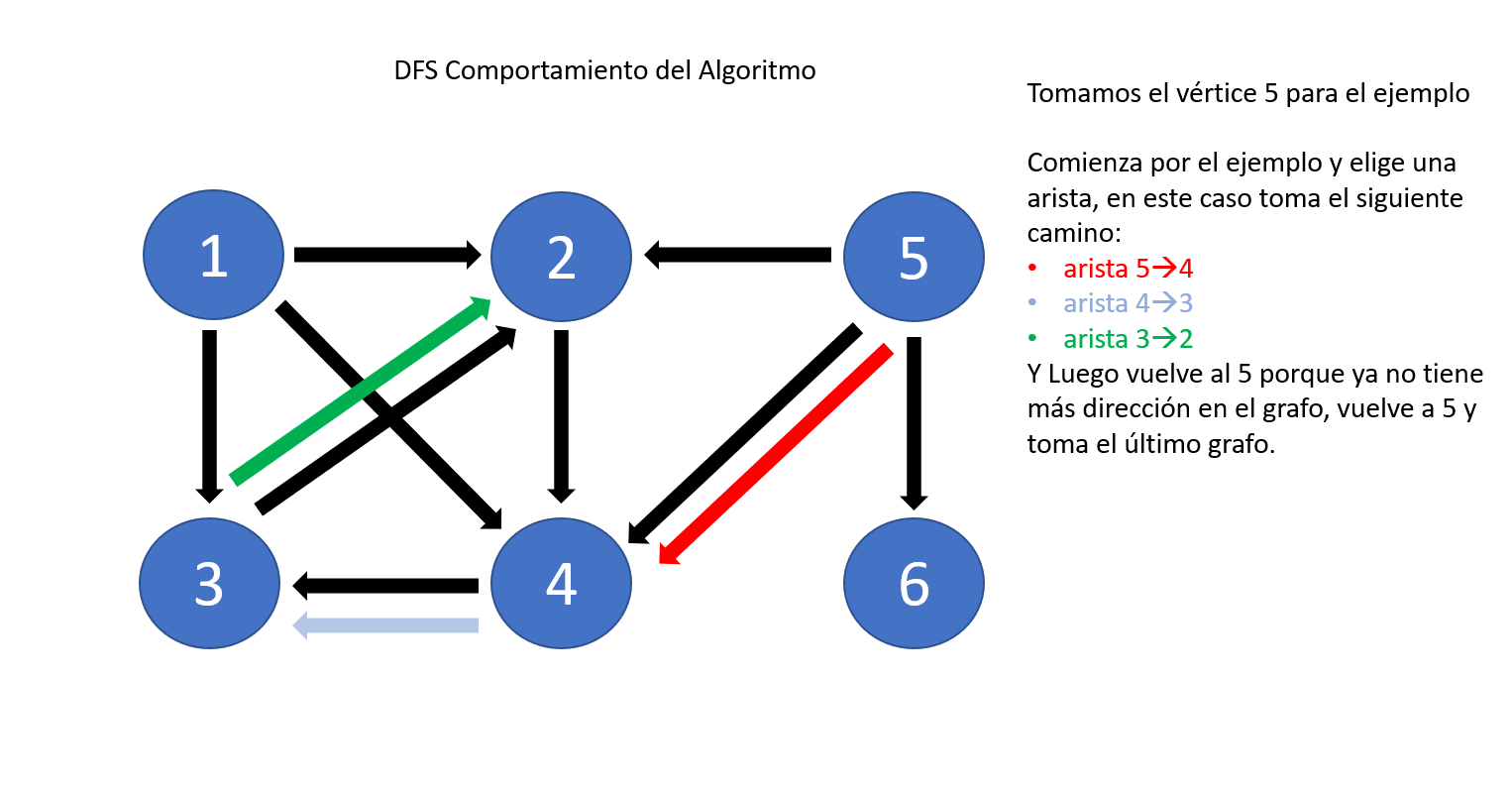
Floyd: Realizamos la carga de los vértices y de la matriz de costos (distancia entre vértices). Le pedimos al usuario que ingrese la opción 1, que es para mostrar un vértice determinado, o la opción 2, que es para mostrar todos los vértices.  
Luego llamamos al algoritmo que aplica la lógica de Floyd, buscando seleccionar entre todos los caminos posibles, el que corresponde al más corto (basándose siempre en la Matriz anteriormente nombrada).

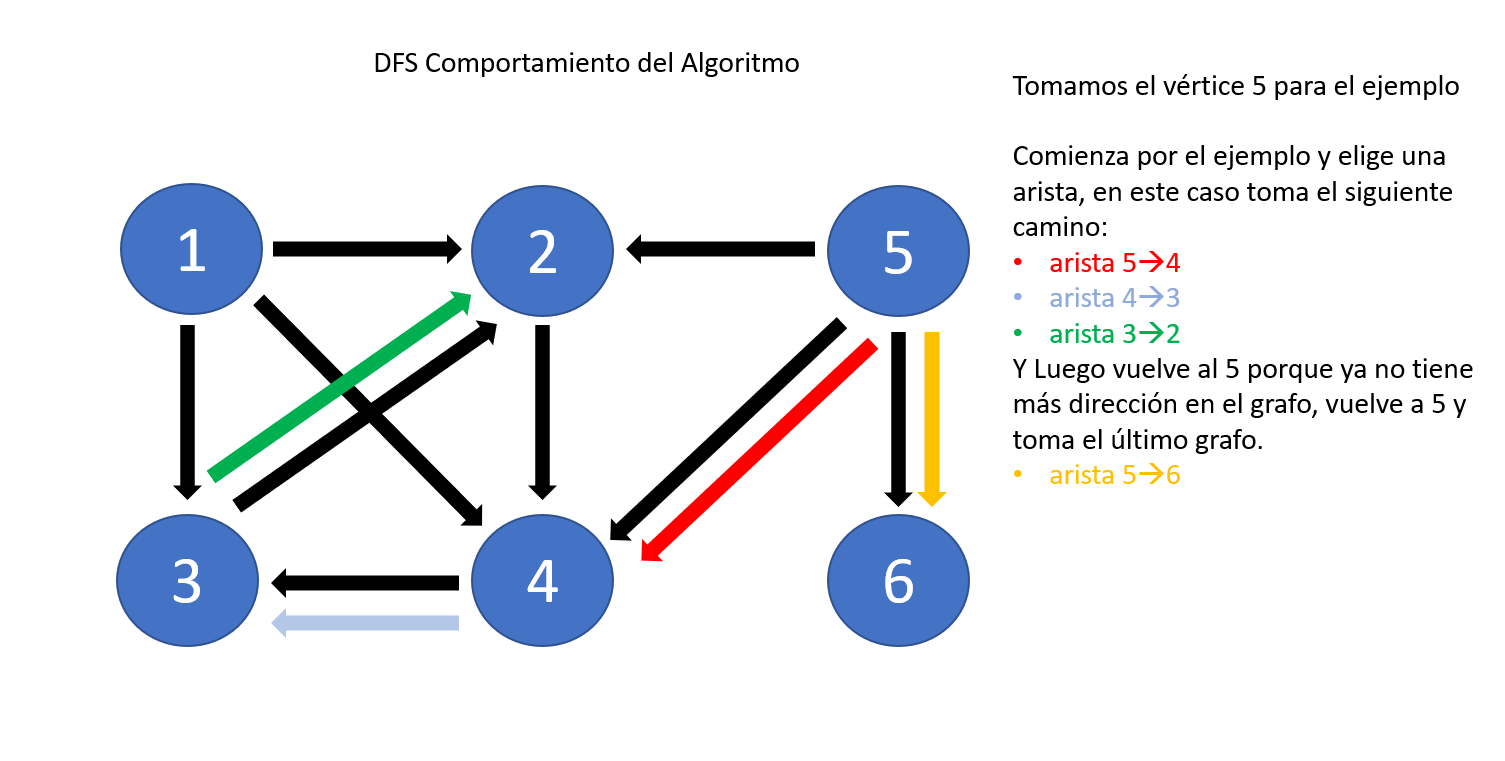
Kruskal: Primero cargamos los vértices y aristas que van a ser evaluados por el algoritmo. Una vez hecho esto, ejecutamos la función desarrollada, la cual busca la arista de menor costo/peso y une a los vértices que la determinan. Este comportamiento vuelve a repetirse hasta lograr obtener un grafo conexo.

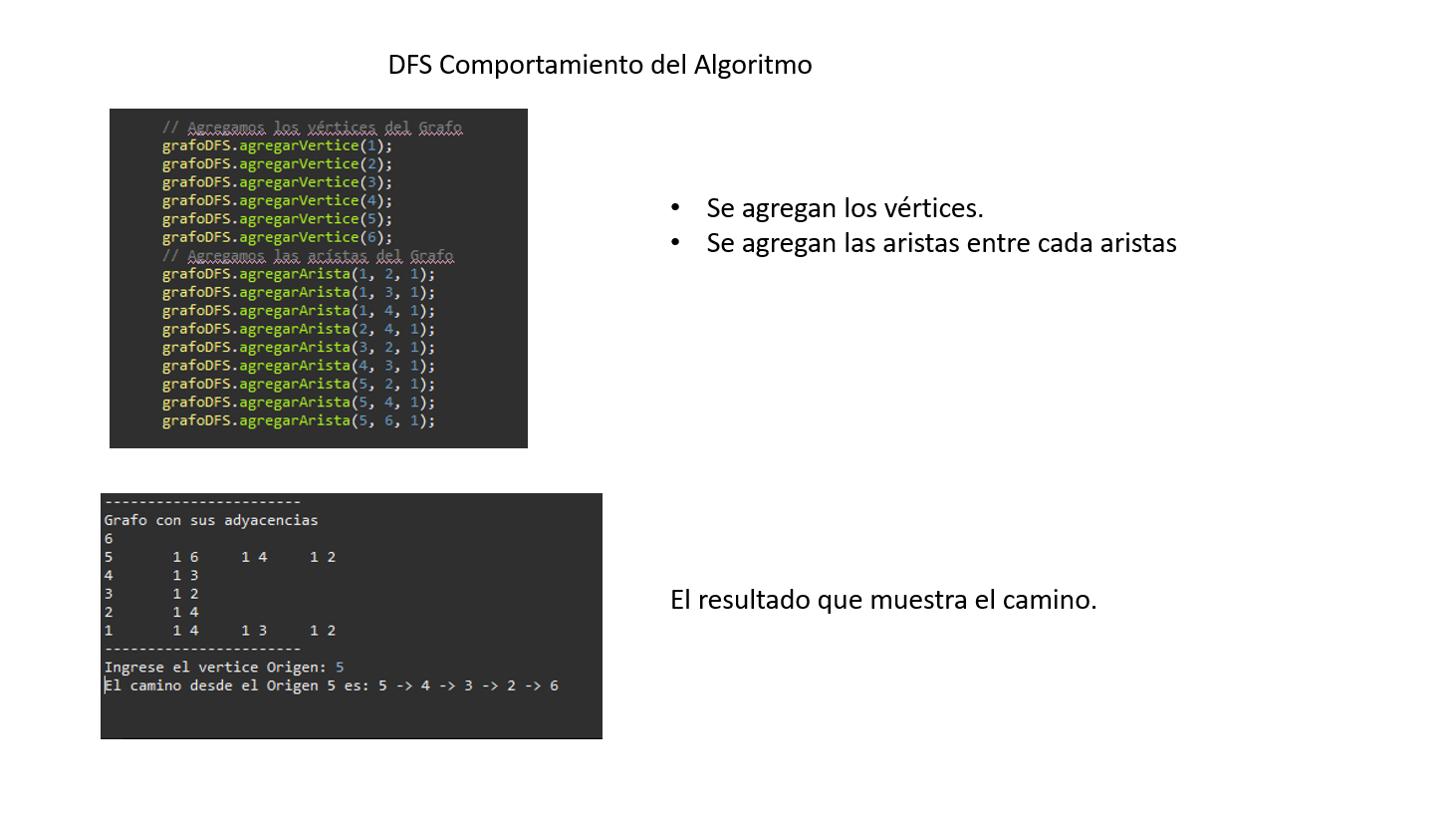
Ejemplo de grafo de entrada de DFS:  
  




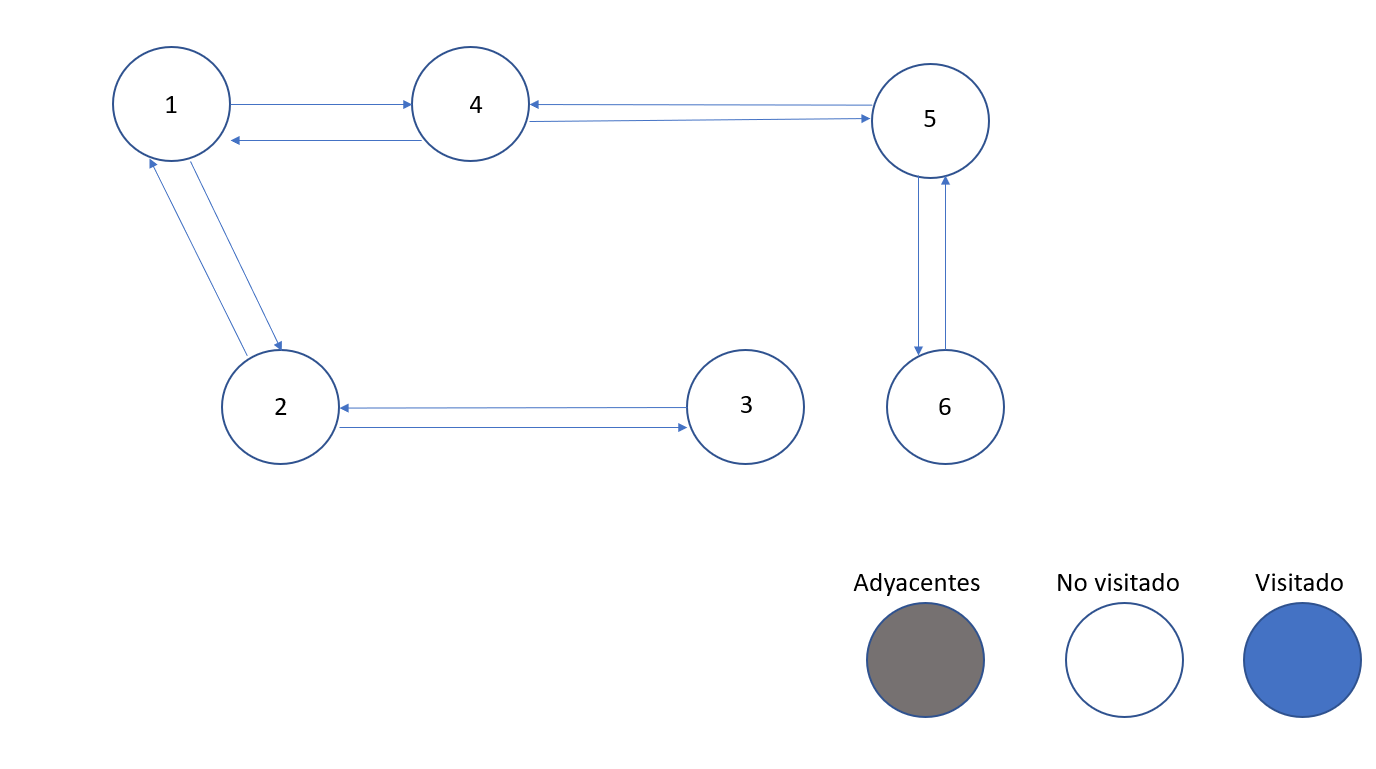


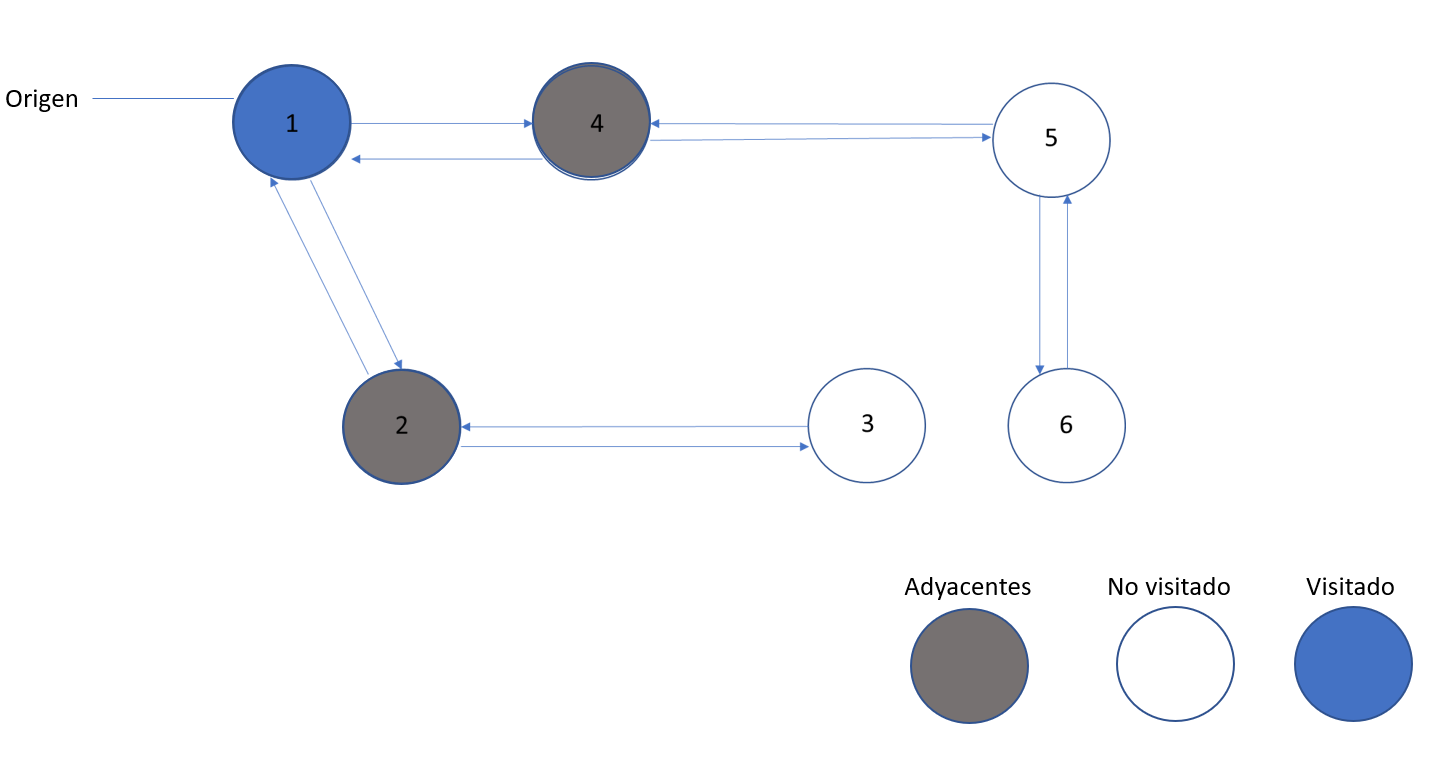


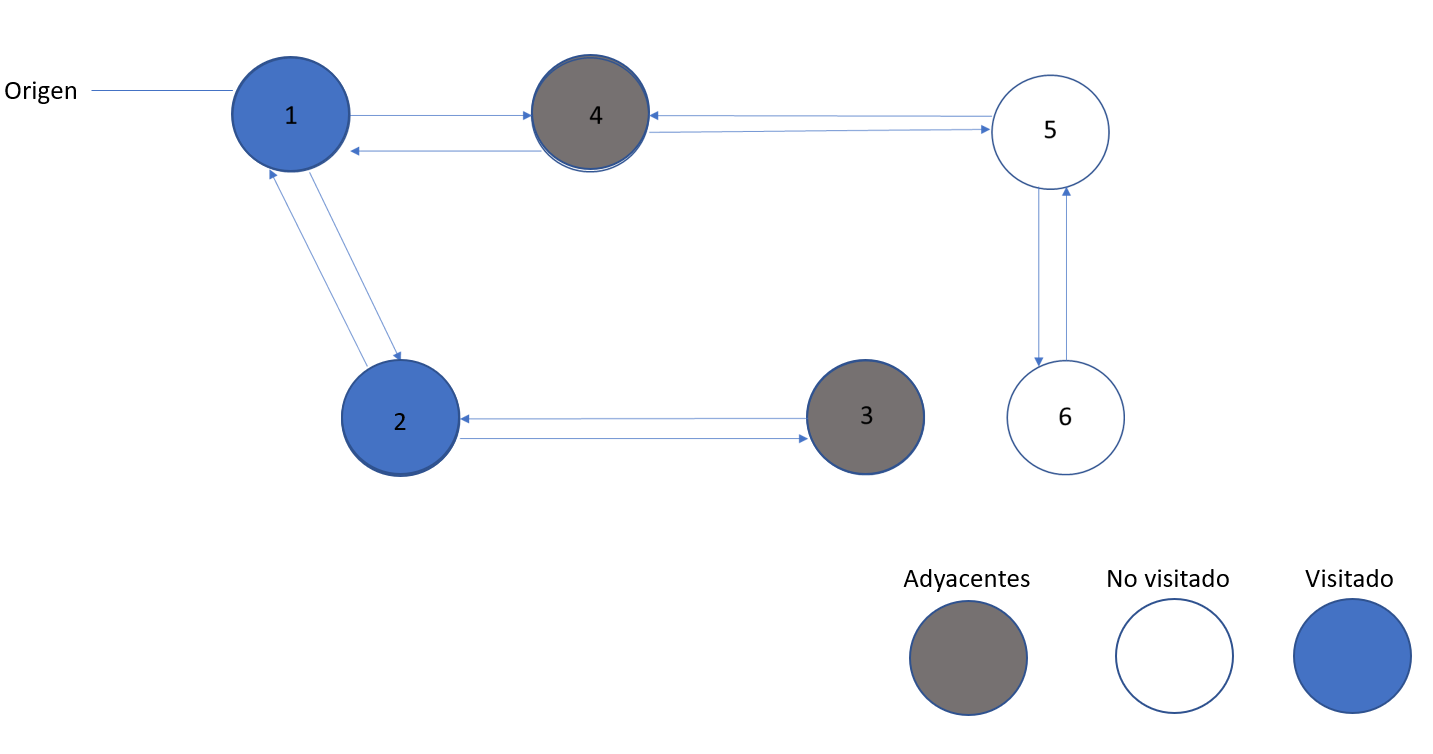


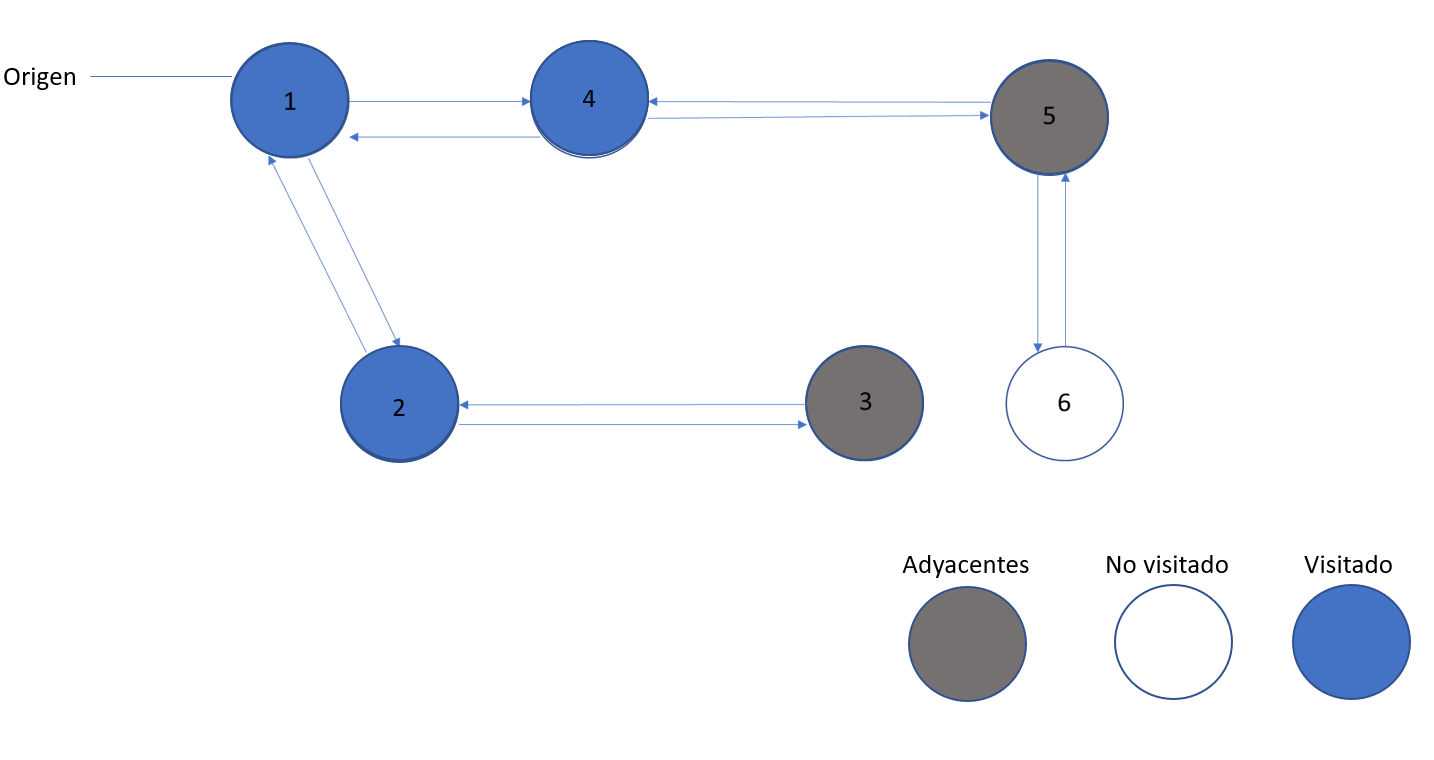


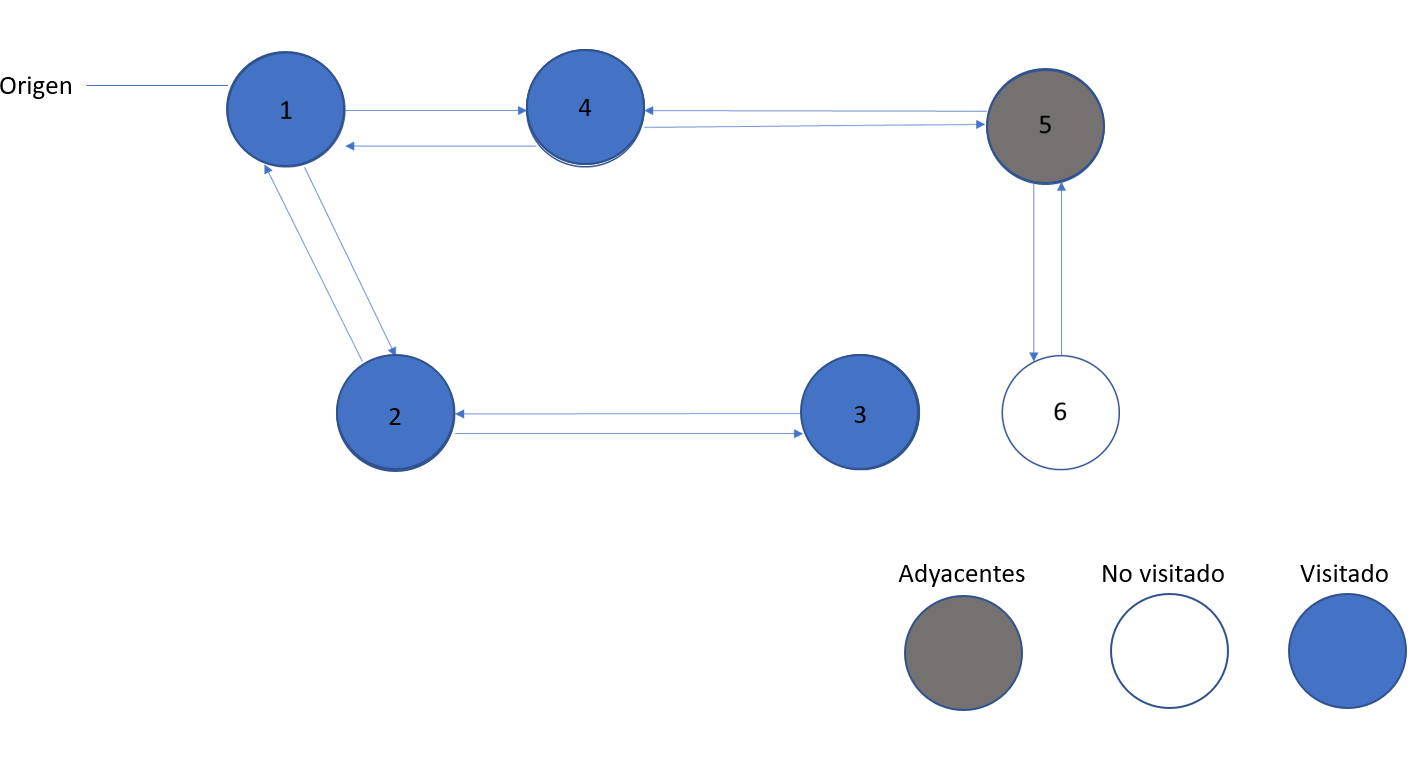
# Ejemplo de grafo de entrada de BFS:

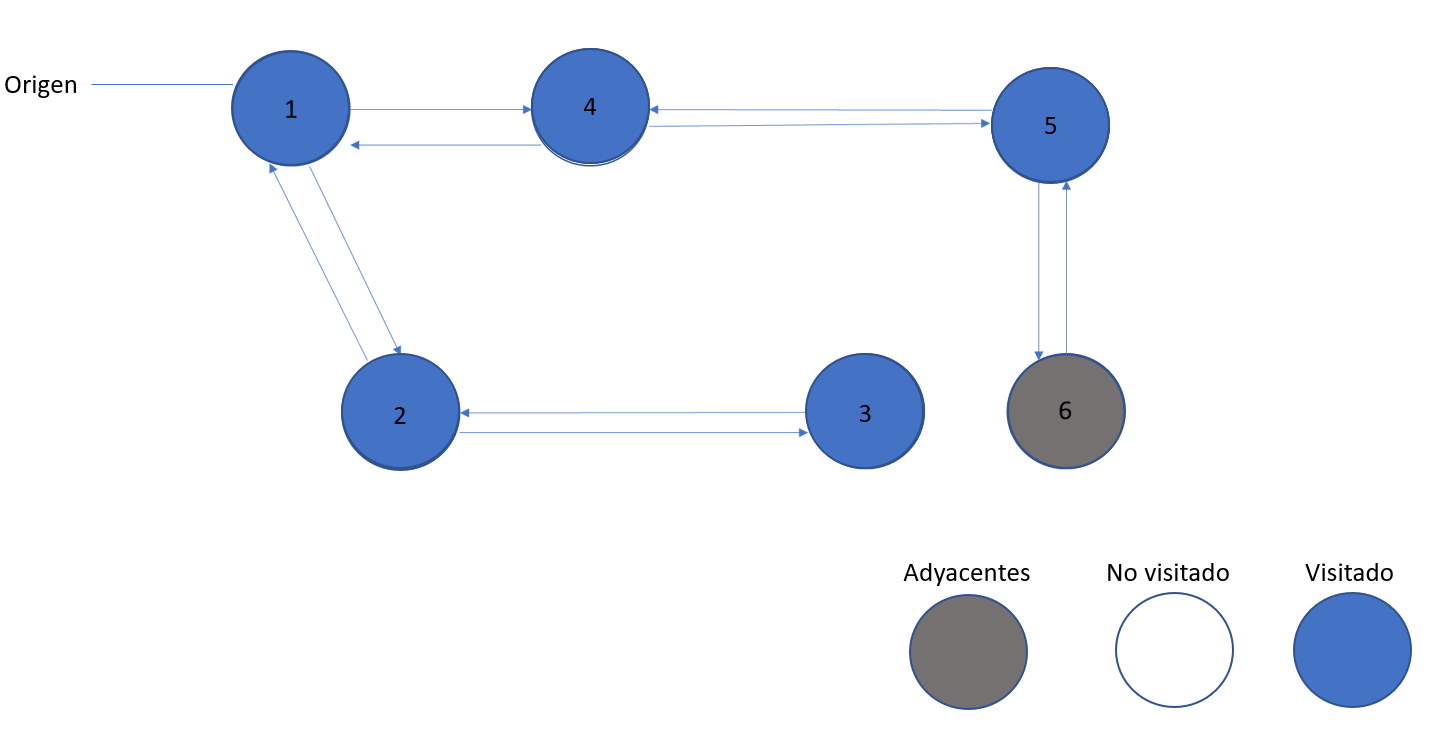


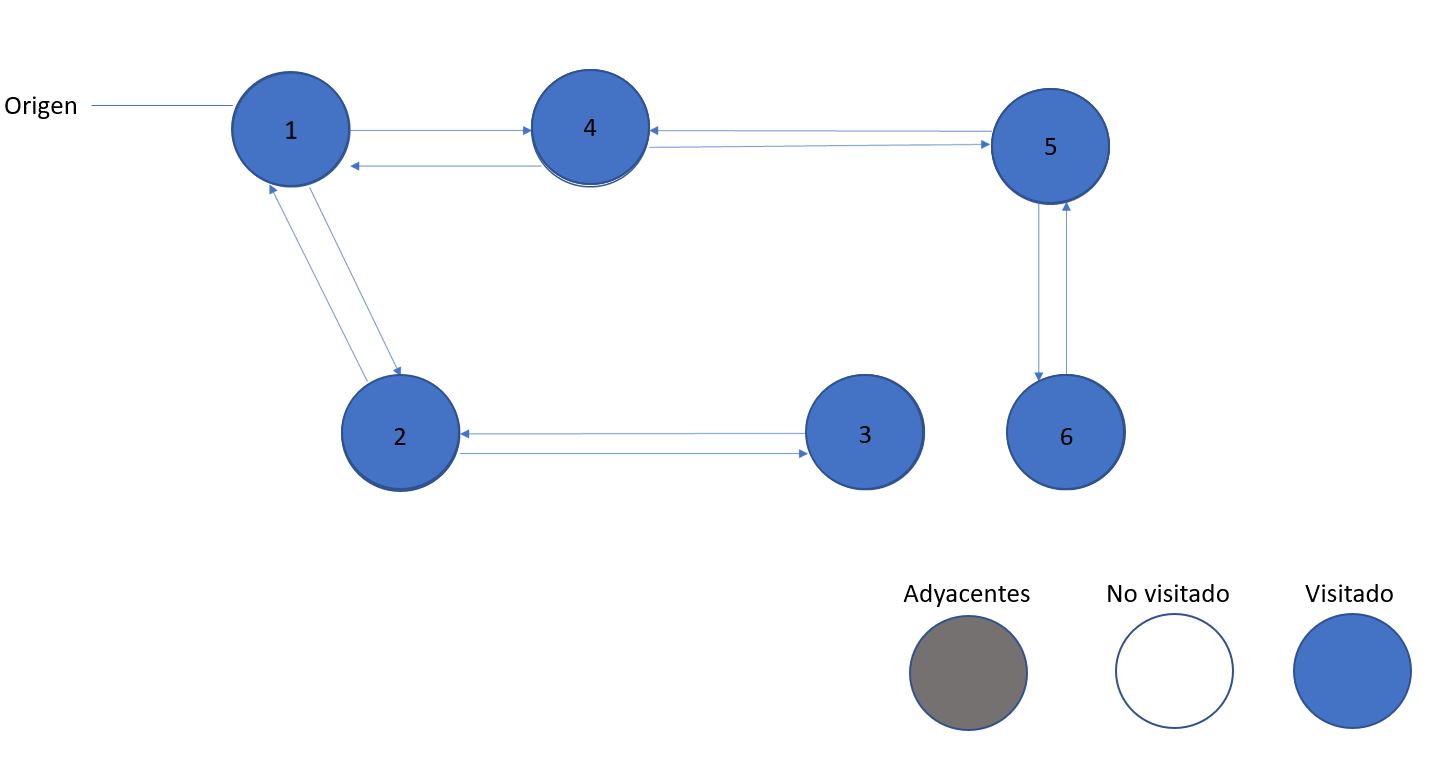






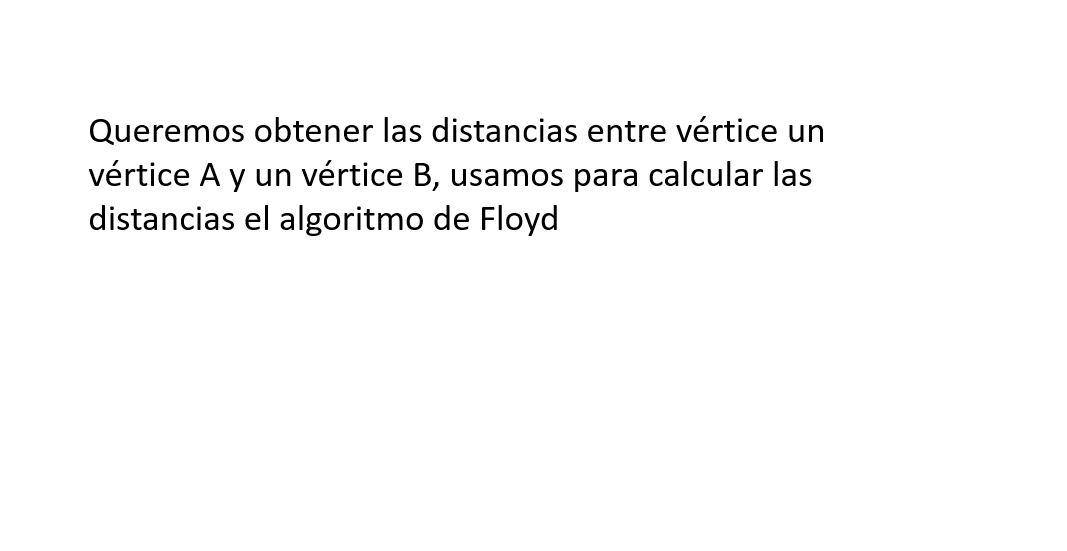


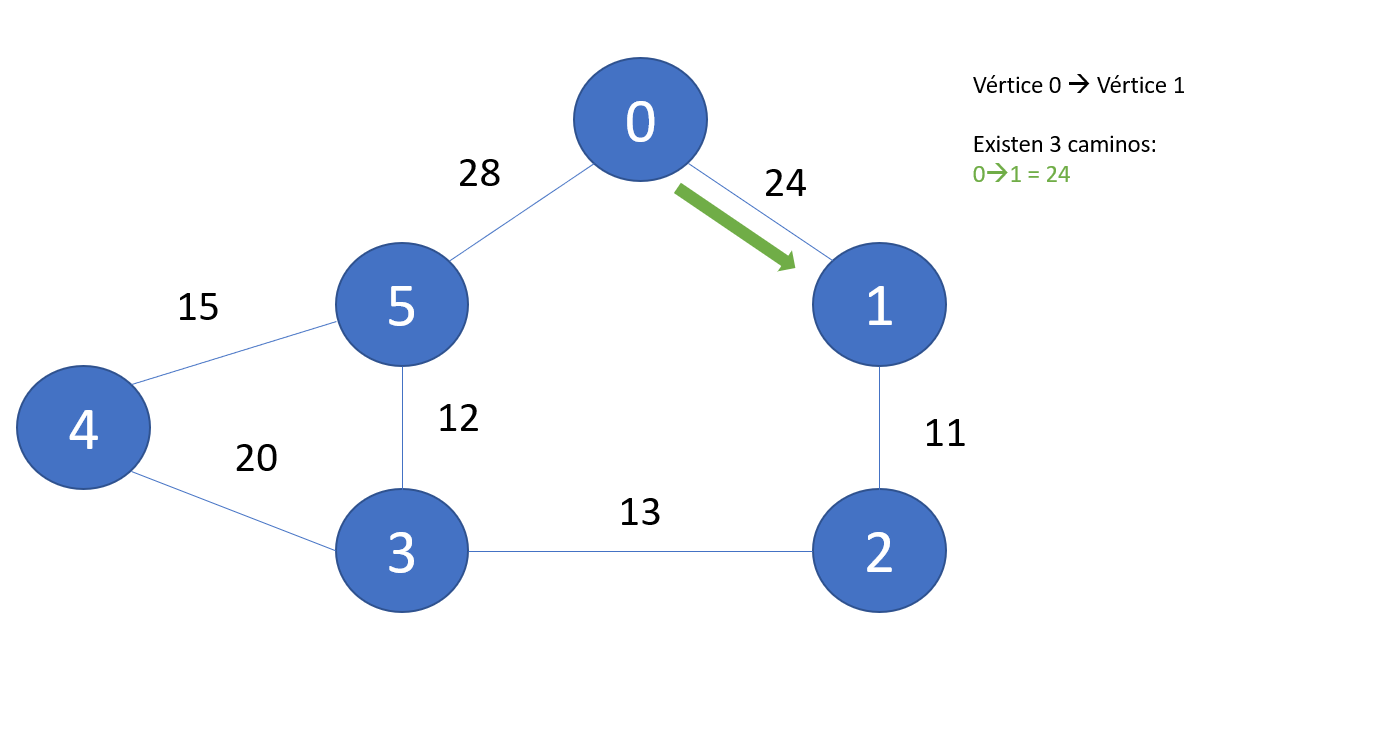


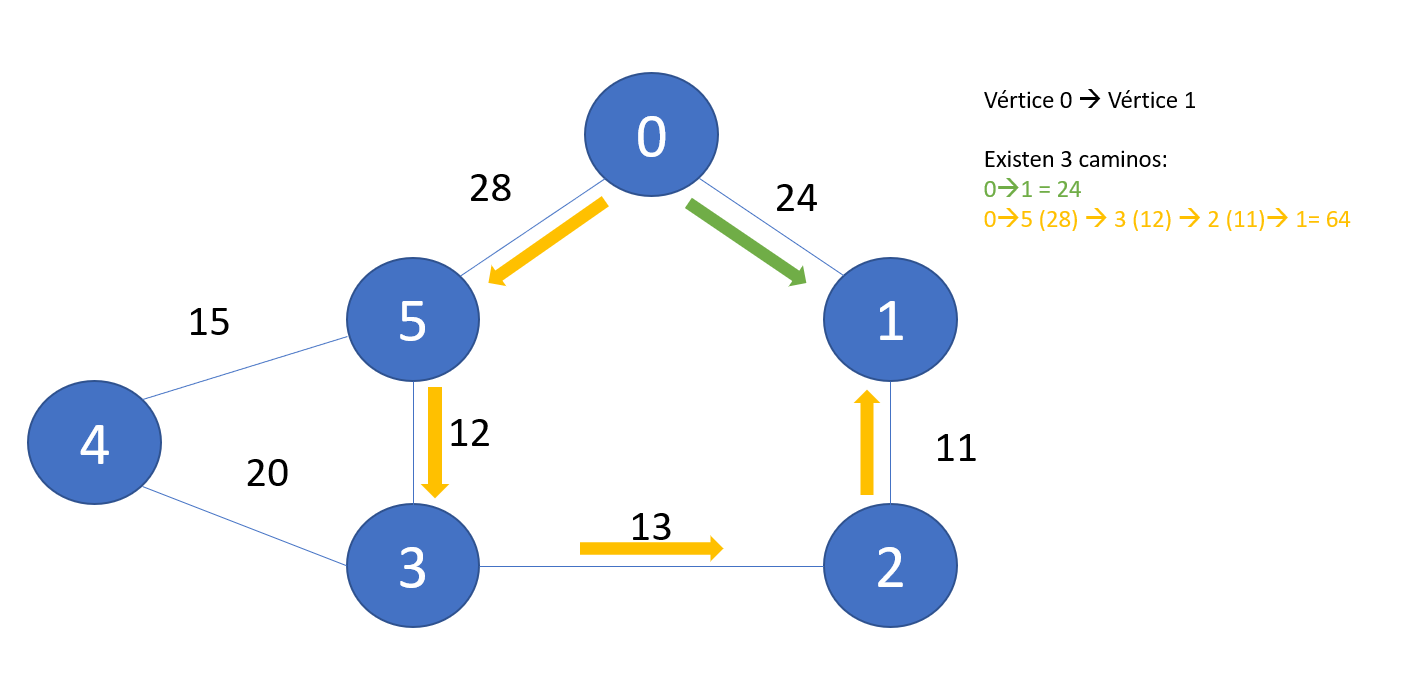


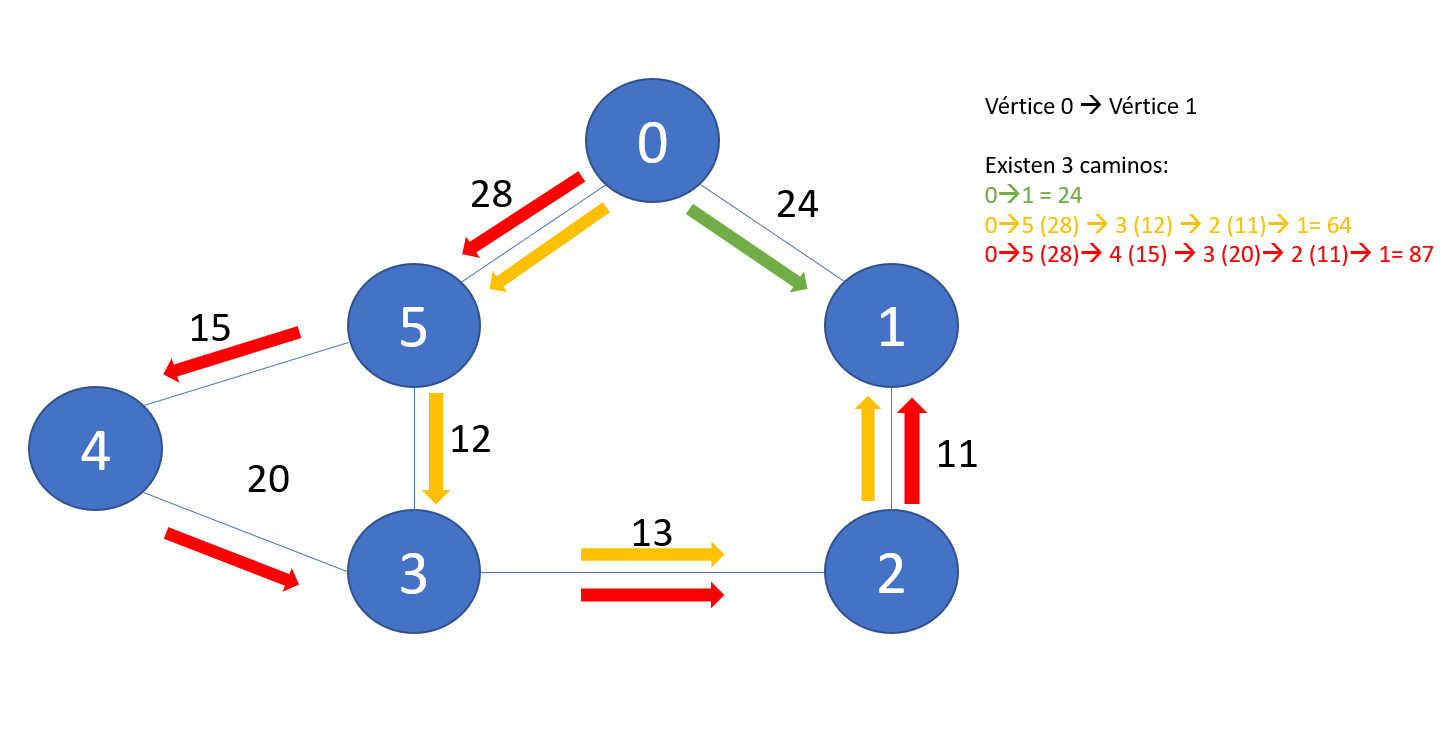


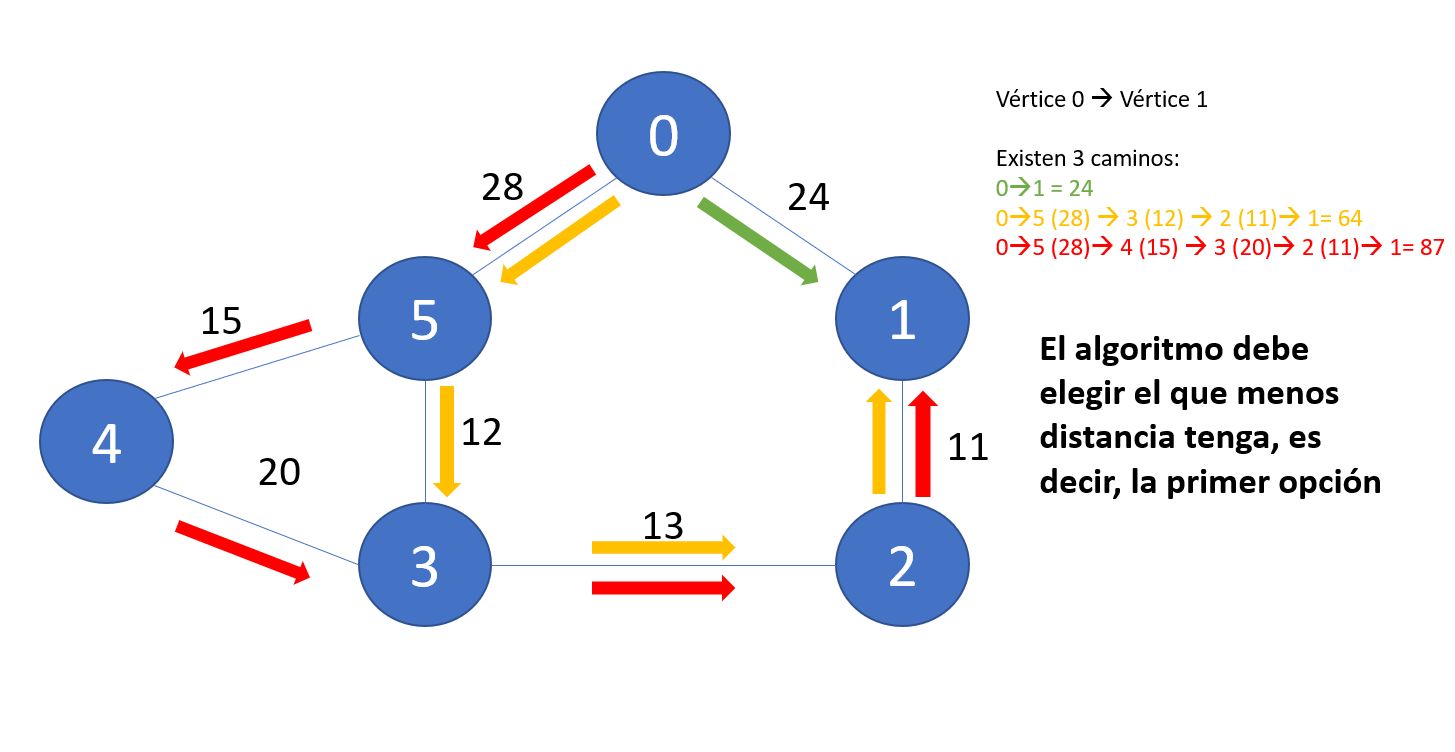
# Ejemplo de grafo de entrada de Floyd:

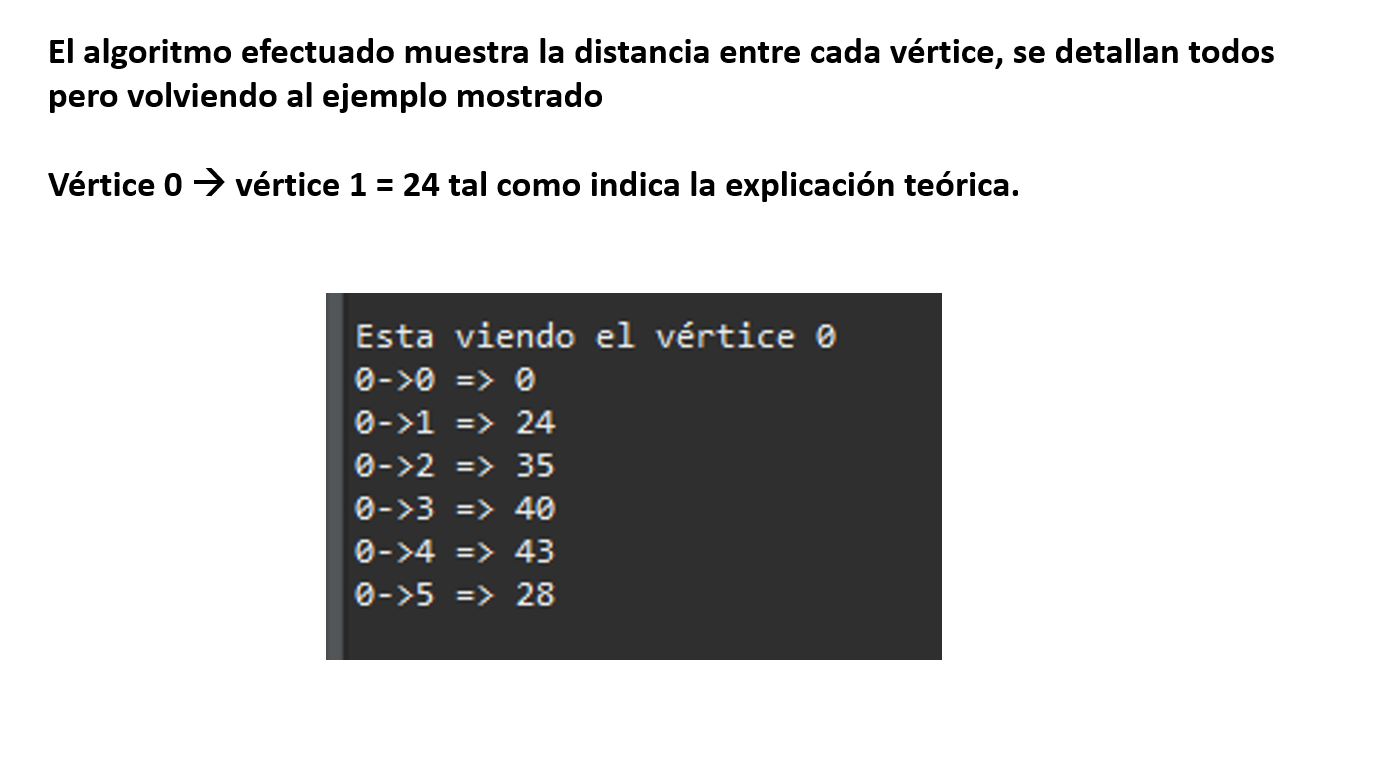












# Ejemplo de grafo de entrada de Kruskal:

