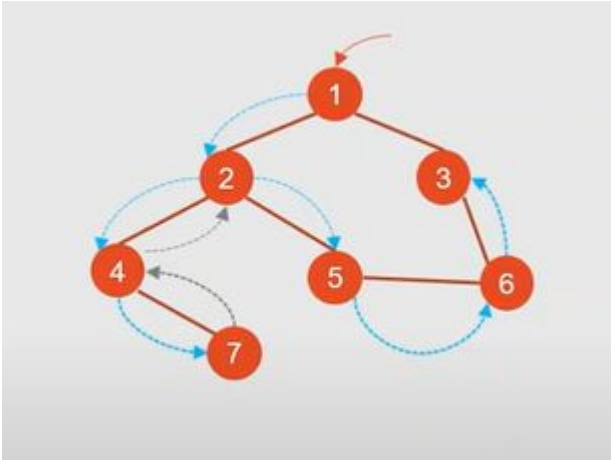
Universidad Nacional de San Martín

Licenciatura en Ciencia de Datos

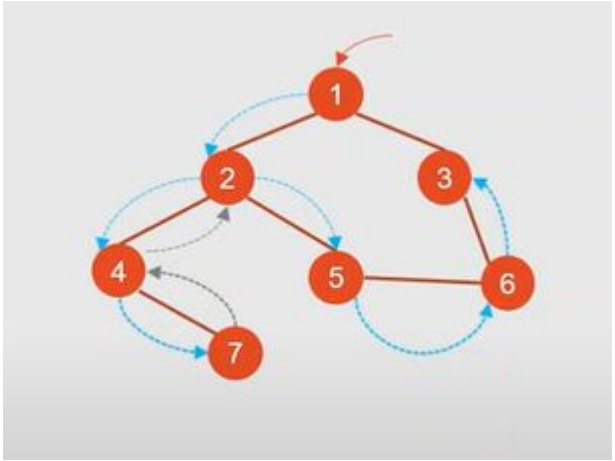
Algoritmos II

Recorrido DFS

Recorrido = 1-2-4-7-5-6-3



Recorrido DFS: Idea



DFS Principal:

1) **Inicializar recorrido vacío**
2) Buscar un **nodo sin visitar**
3) **Recorrer DFS** comenzando en el nodo obtenido

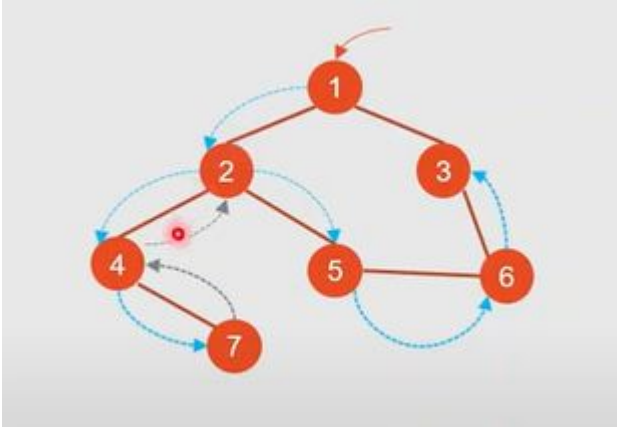
en 2) → en este caso se puede hacer por **grafo no dirigido** y **todos los nodos conectados**

4) Mientras **existan nodos sin visitar**, volver a 2) (cuando el grafo **no es conexo** y queremos **recorrer todo el grafo**)

DFS2:

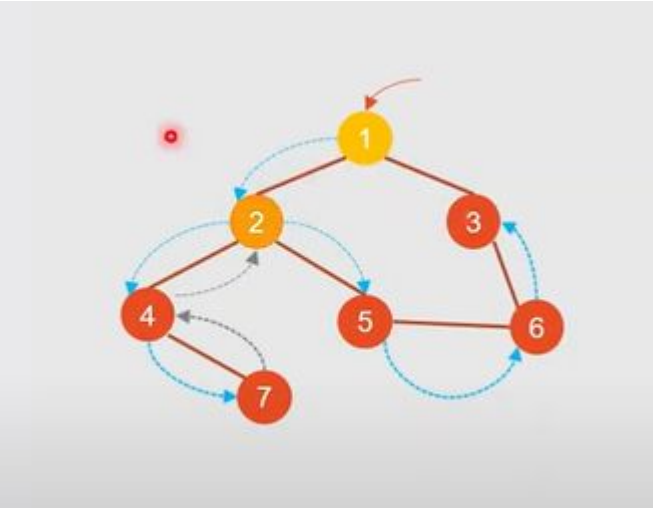
1) Si el **nodo actual** no fue visitado (sirve para **evitar** entrar en **bucles**):
   a) Agregarlo al **recorrido**
   b) Para **cada vecino** del nodo actual **recorrerlo en profundidad (recursión)**

Paso a paso



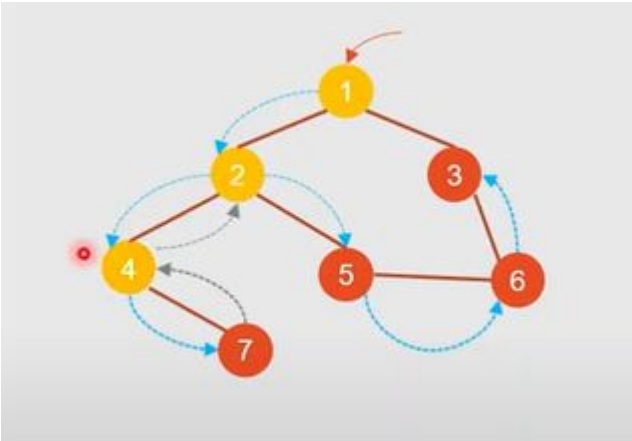
Empezamos por el **nodo 1**, sus **vecinos son 2 y 3**. Lo **agregamos** y visitamos **arbitrariamente** el nodo 2.

Paso a paso



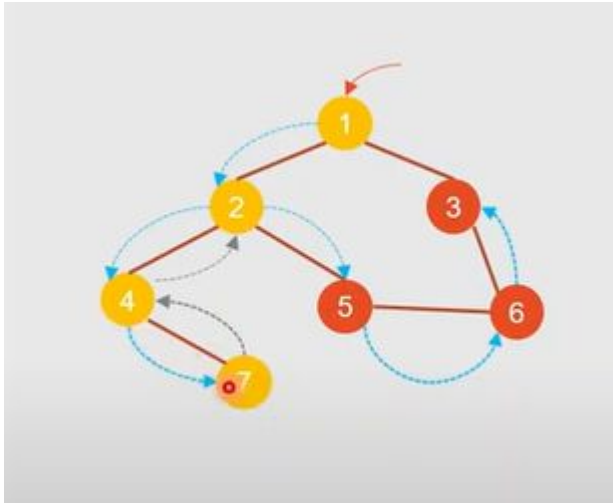
El **2 no está en el conjunto del recorrido**, entonces lo **agregamos**. Los **vecinos** del 2 son **1, 4, y 5**, pero el **1 ya fue visitado** (no tendríamos que entrar en la condición 1 de dfs2)

Paso a paso



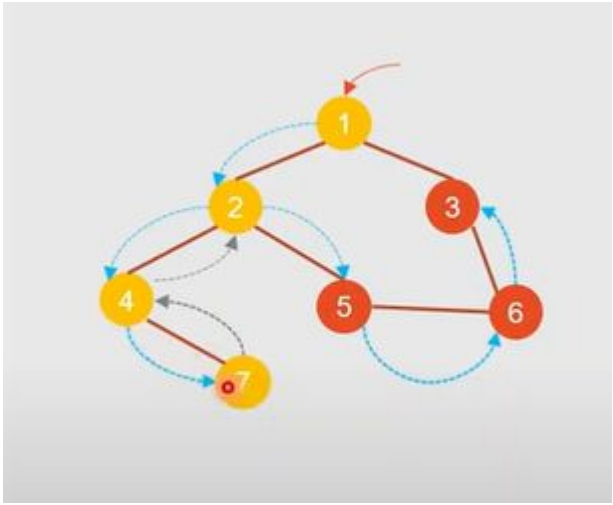
Entonces **vamos al 4**, **no** está en el **conjunto de visitados**, lo **agregamos**, y vemos que sus vecinos son el 2 y el 7. El **2 ya está**, por lo que continuamos con el **7**.

Paso a paso



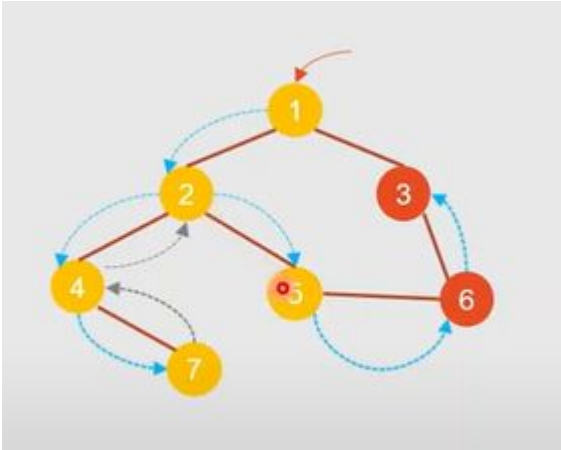
El **nodo 7** tiene el **nodo 4** ya **visitado** y **no más vecinos**. ¿Qué principio se aplica?

Paso a paso



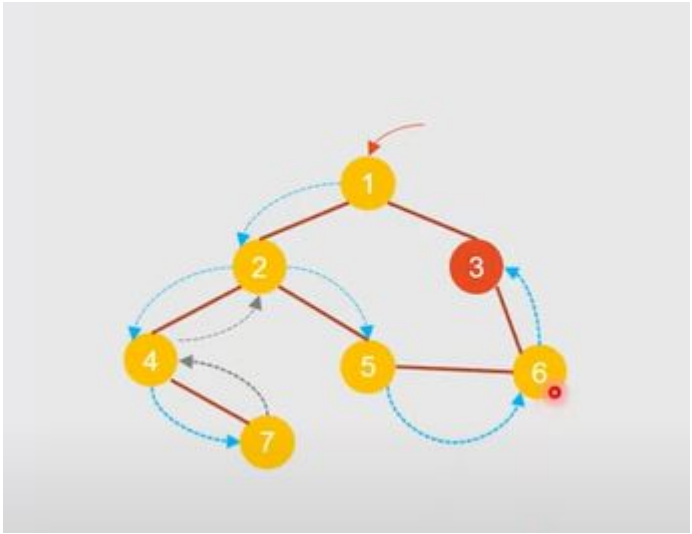
Se **vuelve al nodo 2** (vecinos 1, 4 y 5), el **5 está sin visitar**. Entonces **visitamos** el 5.

Paso a paso



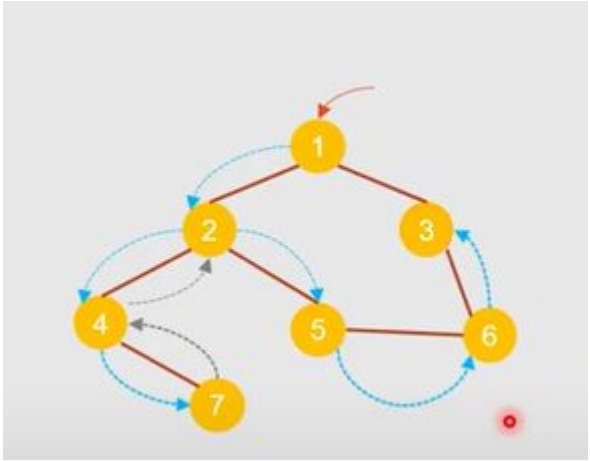
Seguimos con el **5**, que a su vez tiene como **vecinos el 2, ya visitado,** y el **6**. Entonces **seguimos por el 6.**

Paso a paso



El **6** tiene como **vecinos al 5 y al 3**, y el **3 no está visitado**, va al 3.

Paso a paso



Ya no tenemos más **vecinos por visita**r, entonces el **recorrido termina** acá.

Ejercicio

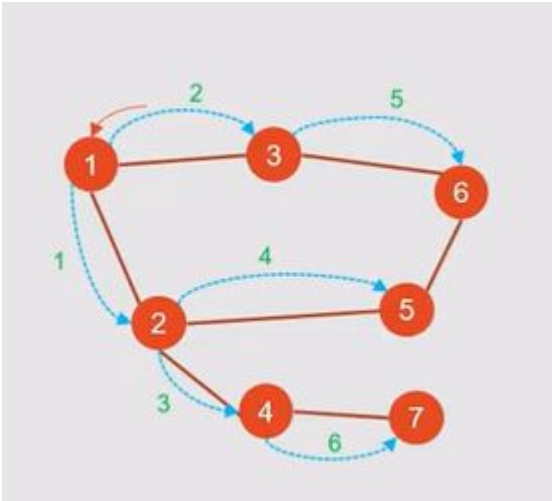
Implementar el **DFS** usando la **representación** de **grafos** de **conjunto de nodos y aristas.** Usar **recursividad**. Suponer grafos **no dirigidos** y **conexos**

Firma:

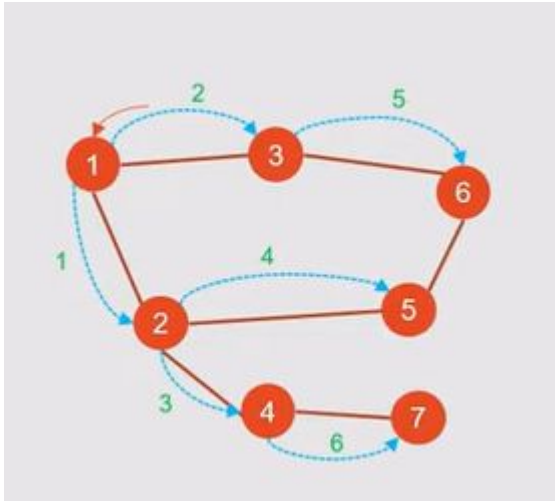
```
def dfs(self) -> list[T]:
```

Recorrido BFS

Recorrido = 1-2-3-4-5-6-7



Recorrido BFS: Idea



BFS Principal **(inicializa el recorrido)**

1) Inicializar **recorrido vacío**
2) Buscar un **nodo sin visitar** (hasta acá idem **dfs**)
3) Recorrer **BFS** comenzando con el **nodo obtenido**

en 2), **encolándolo** en una **cola. Diferencia**:no lo vamos a recorrer como parámetro

como un nodo más, que se pasa a **bfs2**.

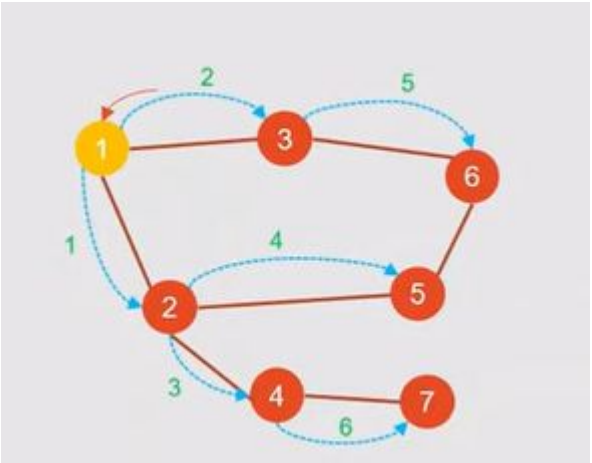
4) **Mientras haya nodos sin visitar,** recorrerlos con **BFS** como

en el **paso 3)** (hace los pasos 2) y 3)) para, partes **no conexas del grafo, o si fuera dirigido**)

BFS2 (realiza el recorrido a lo ancho)

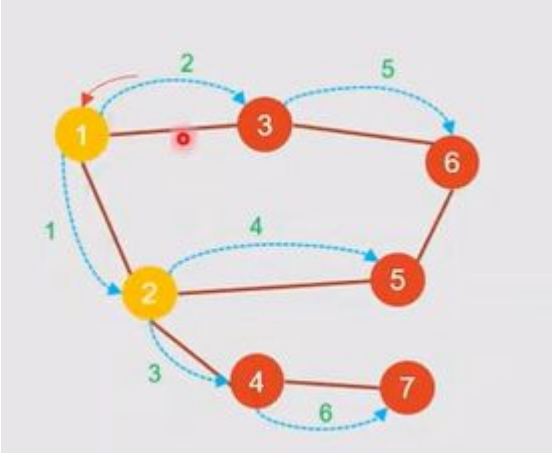
1) **Si hay nodos encolados**, continuar con el **paso 2)**
2) **Desencolar un nodo** de la cola
3) Si el nodo desencolado **no fue visitado**
   a) **Agregarlo al recorrido**
   b) **Para cada vecino** del nodo actual, **encolarlo**
4) **Invocar BFS** nuevamente con l**a cola modificada** (recursión)

Paso a paso



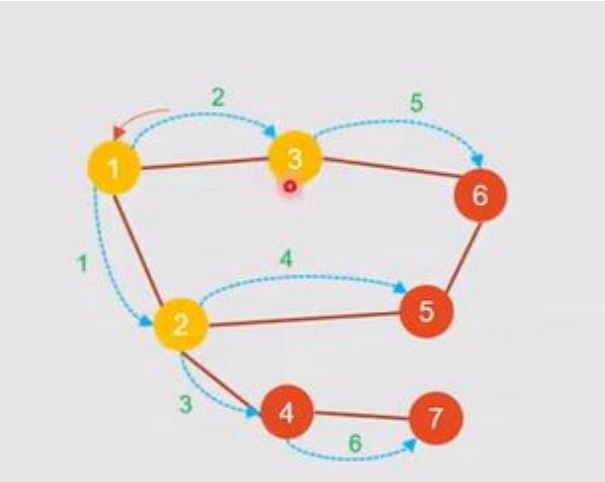
Visitamos el **1**, y visitamos al **2**, es e**l primer vecino** que vamos a **encolar**

Paso a paso



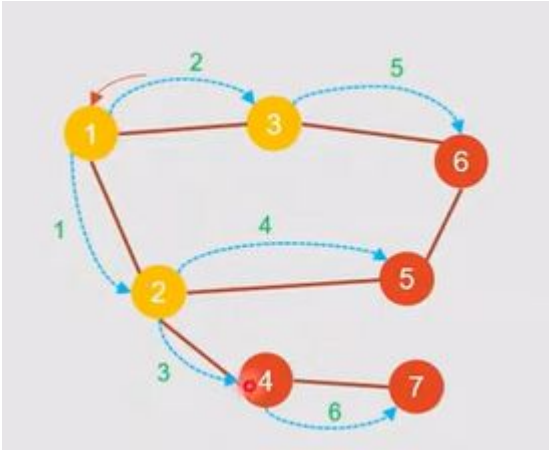
Y después vamos a **encolar el 3, porque son los dos vecinos del 1** (paso 2) del paso 3) del BFS2)

Paso a paso



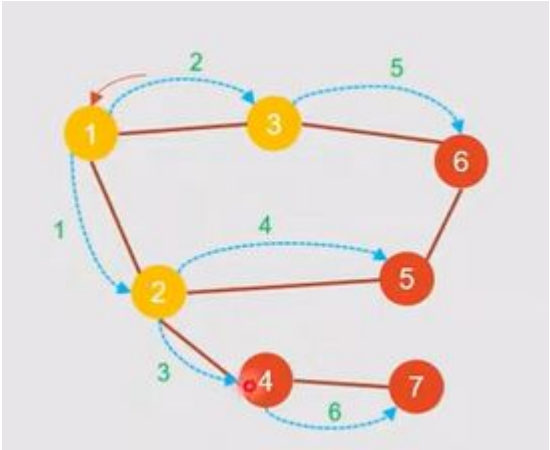
Cuando se **visita el nodo 2**, también se encolan el el **4 y el 5**, pero están encolados **después del 3**, entonces al **desencolar** se desencola **primero el 3**.

Paso a paso



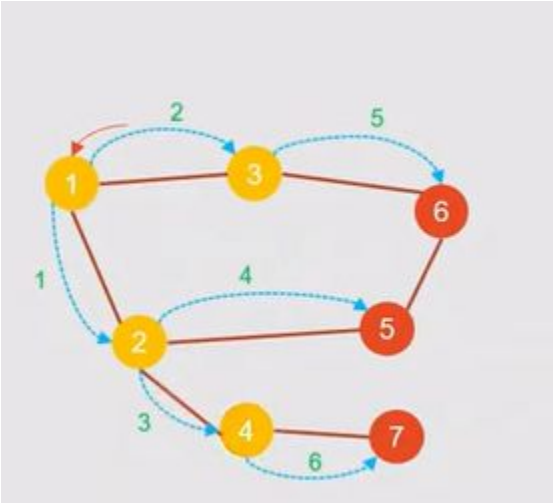
El **3 encola el 6**, que va a estar **después del 1, del 5 y del 4**

Paso a paso



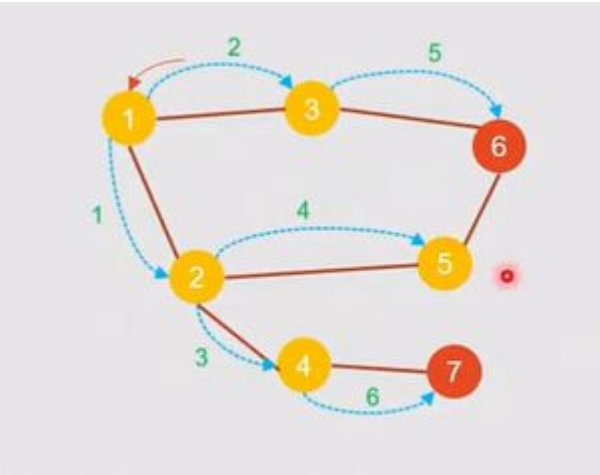
Va a **continuar por el 1**, que fue el primero que se encoló, pero como **ya fue visitado, no va a hacer nada**. Va a seguir por el **próximo nodo** que es el **4.**

Paso a paso



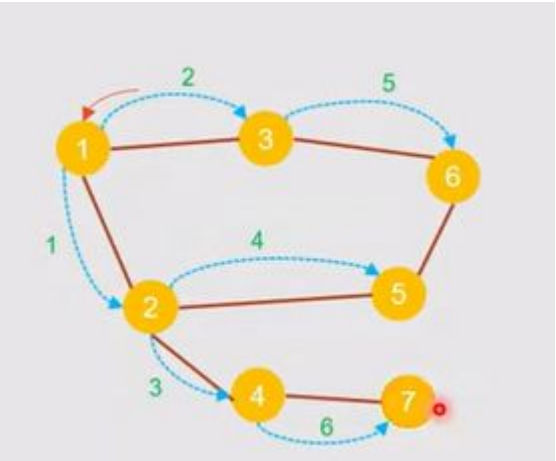
El **4 va a encolar el 7**, pero **después** de los vecinos que ya teníamos en la **cola**, entonces **seguimos por el 5**

Paso a paso



El 5 encola el 6.

Paso a paso



Y **finalmente nos queda el 7**

Nota: en el medio hubo nodos encolados, pero como no entra en la condición 3) del BFS2 no los vuelve a agregar al recorrido.

Recorrido BFS

Lo interesante: **nos va a garantizar encontrar el nodo más cercano al original cuando el grafo no es ponderado**, porque siempre va por **niveles**.

Ejercicio

Implementar **BFS recursiva** usando la representación de **grafos de conjunto de grafos y aristas**, **no ponderados**. Suponer grafos **no dirigidos** y **conexos.**

Firma:

```
def bfs(self) -> list[T]:
```