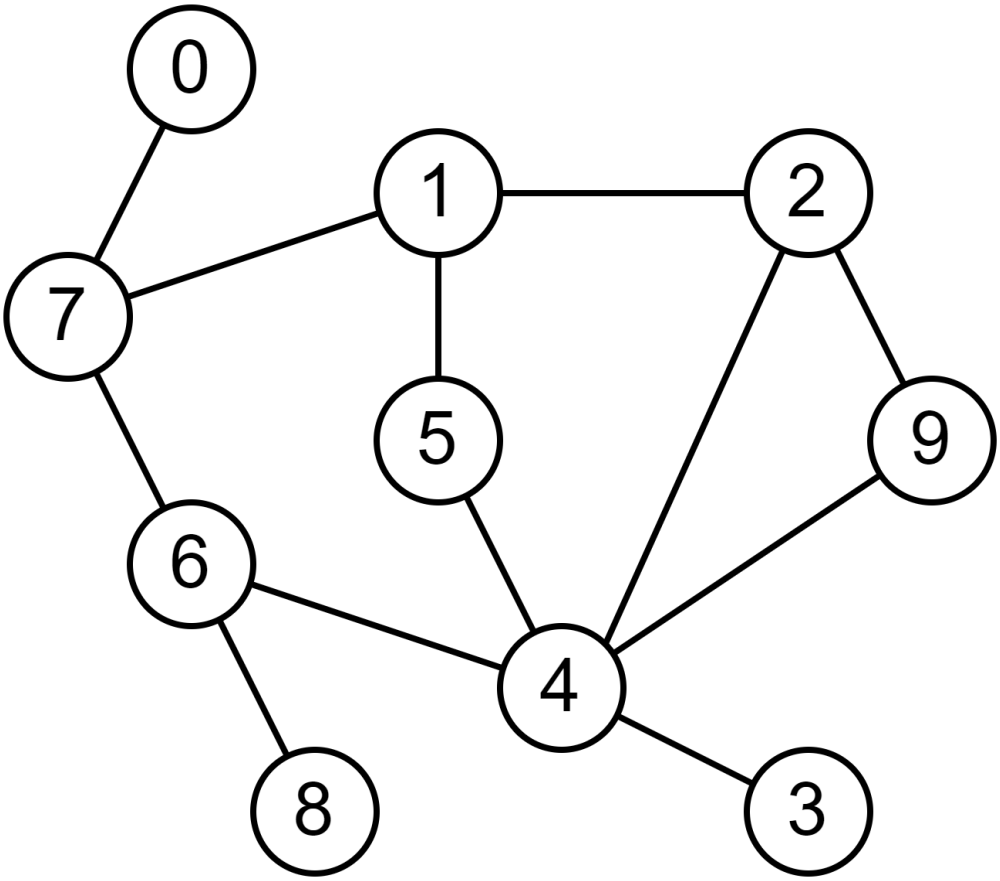
Proyecto TeoComp 2022 - Grafos

El trabajo obligatorio final consiste en la implementación de una clase Grafo. Esta clase deberá ser capaz de representar grafos no dirigidos, simples y sin lazos, conteniendo strings en sus nodos. Se deja a libre consideración del estudiante que método de representación del grafo utilizar, sin embargo, la clase grafo deberá dar solución a un conjunto de funcionalidades.

Funcionalidades

DFS (Búsqueda en profundidad)

Dado un Nodo como entrada, devolver el recorrido realizado por la búsqueda en profundidad partiendo de dicho nodo. Si el nodo no existe, devolver la lista vacía



Grafo G1

Ejemplo:

G1.dfs(’2’) → ['2', '9', '4', '6', '8', '7', '0', '1', '5', '3'] *

BFS (Búsqueda en amplitud)

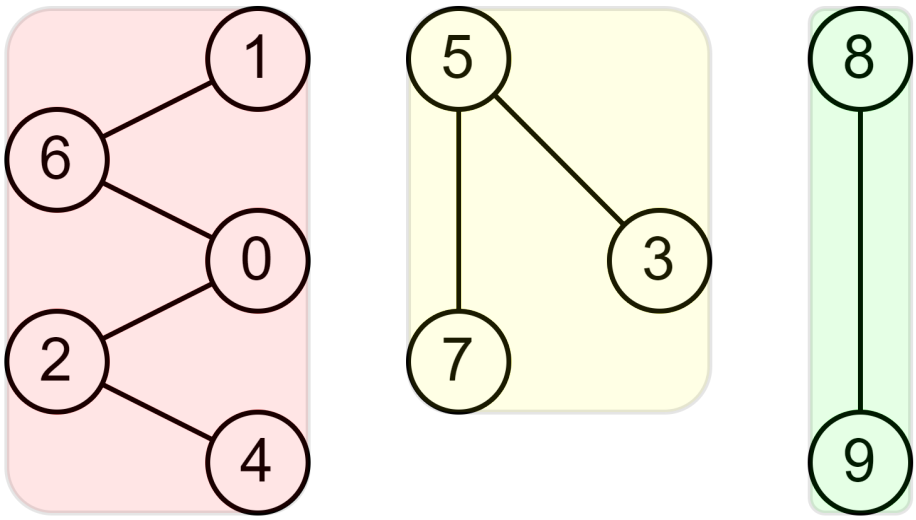
Dado un Nodo como entrada, devolver el recorrido realizado por la búsqueda en amplitud partiendo de dicho nodo. Si el nodo no existe, retornar la lista vacía.

Ejemplo:

G1.bfs(Nodo(2)) → ['2', ‘9', '4', '6', '8', '7', '0', '1', '5', '3']*

Componentes conexas

Dado un grafo, devolver una lista de listas de nodos. Cada sub lista representara una componente conexa del grafo.



Grafo G2

Ejemplo:

G2.componentes_conexas() → [[’6’, ‘1’, ‘2’, ‘0’, ‘4’], [’7’, ‘5’, ‘3’], [’9’, ‘8’]]*

Cantidad de Componentes Conexas

Dado un grafo, retorna la cantidad de componentes conexas que lo componen.

Ejemplo:

G2.cantidad_componentes_conexas() → 3

G1.cantidad_componentes_conexas() → 1

Es conexo?

Dado un grafo, se devuelve True si el grafo es conexo. False en cualquier otro caso

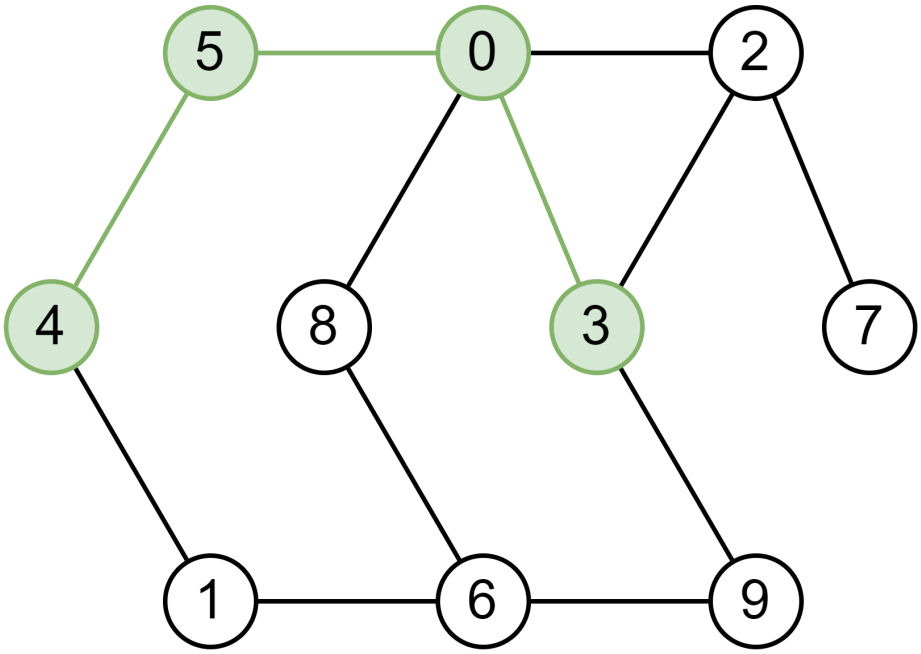
Ejemplo:

G2.es_conexo() → False

G1.es_conexo() - > True

Camino más corto entre 2 nodos

Dados dos Nodos, devolver el camino mas corto partiendo del primero y llegando al segundo. Este camino debe ser representado como una lista de nodos. En caso de que el camino no exista, se devuelve la lista vacía.



Grafo G3

Ejemplo:

G3.camino_mas_corto('4', '3') → ['4', '5', '0', '3']*

G2.camino_mas_corto('9', '1') → []

Largo del camino más corto

Dados dos nodos se retorna la distancia entre ellos.

Ejemplo:

G3.largo_camino_mas_corto(’4’, ’3’) → 3

Verificar si un camino es el más corto

Dado una lista de nodos, verificar si el camino representado por ella, es el **más** corto.

(OJO!: Pueden existir varios caminos **más** cortos)

Ejemplo:

G3.verificar_camino_mas_corto([’4’, ’5’, ’0’, ’3’]) → True

G3.verificar_camino_mas_corto([’4’, ’1’, ’6’, ’9’, ’3’]) → False

Observaciones:

1. El trabajo será grupal, con los equipos previamente organizados en el curso.
2. La tarea será realizada con Python 3.6 o superior
3. El código deberá ser original
4. No se permite el uso de bibliotecas que implementen grafos. Si se podrán utilizar bibliotecas que implementen estructuras auxiliares necesarias, sea tanto para la realización de las funciones como para visualizar los resultados

Entrega

La entrega, con fecha límite el lunes 5 de diciembre de 2022 a las 23:59, deberá realizarse vía Web Asignatura, una por grupo, constando con los siguientes elementos:

- Código fuente original realizado por los estudiantes, completo y con todo lo necesario para ser ejecutado
- Breve informe detallando los siguientes puntos:

- Motivo de elección de la implementación de grafos realizada, fundamentada sobre las características solicitadas de los grafos a representar, así como las funcionalidades requeridas por la tarea.
- Observaciones de implementación de funciones (ej. elección de algoritmos recursivos sobre iterativos, o viceversa)
- Comentarios que los estudiantes consideren pertinentes.
- El informe debe ser breve. Se calificará calidad, no cantidad.