EDUCA2 – ¡Domina las Matemáticas y Programación!

Curso Intermedio de Programación en Python

Preparador: Alexanyer Naranjo

Guía Teórica N°5.2

Grafos

Nos tomaremos la oportunidad de conocer una de las estructuras de datos más interesantes en el mundo de la computación, los grafos. En esta guía teórica nos encontraremos con imágenes ilustrativas y ejemplos de códigos sobre la implementación de grafos con Python.

Al final, se listarán algunos algoritmos muy importantes a tomar en cuenta cuando de grafos estamos hablando. No nos detengamos y empecemos a conocer sobre esta estructura de datos.

Grafos:

Un grafo es una representación pictórica de un conjunto de objetos donde algunos pares de objetos están conectados por enlaces. Los objetos interconectados se representan mediante puntos denominados **vértices** y los vínculos que conectan los vértices se denominan **aristas**.

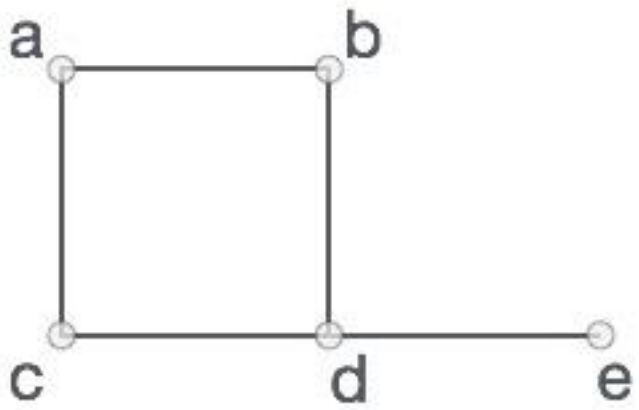
En esta guía veremos cómo crear un grafo y agregarle varios elementos de datos usando un programa de Python. A continuación, se muestran las operaciones básicas que realizamos en grafos:

- Mostrar vértices de un grafo
- Mostrar aristas de grafo
- Agregar un vértice
- Agrega una arista
- Crear un grafo

Un grafo se puede presentar fácilmente utilizando los tipos de datos del diccionario de Python. Representamos los vértices como las claves del diccionario y la conexión entre los vértices también llamados aristas como los valores en el diccionario.

```python
# Creamos el grafo con un diccionario
graph = { "a" : ["b","c"],
          "b" : ["a", "d"],
          "c" : ["a", "d"],
          "d" : ["e"],
          "e" : ["d"]
        }

# Imprimos el grafo
print(graph)
```



Cabe mencionar que existen dos tipos de grafos: Direccionales y No Direccionales. La diferencia entre ambos tipos de grafos es que las aristas presentan una dirección a donde ir (como si fuese una autopista donde hay un canal de circulación del lado derecho y otro del lado izquierdo que van en sentido contrario).

Estaremos trabajando con grafos sin dirección como se representa en las imágenes anteriormente mostradas.

Implementación de un Grafo en Python:

Ahora pasaremos al código donde implementaremos un grafo escrito en Python. Primero mostraremos el código final y luego explicaremos cómo trabaja cada uno de los métodos mostrados dentro del código.

```
  class graph:

    def __init__(self,gdict=None):
        if gdict is None:
            gdict = {}
        self.gdict = gdict

    def edges(self):
        return self.findedges()

    def getVertices(self):
        return list(self.gdict.keys())

    def addVertex(self, vrtx):
       if vrtx not in self.gdict:
            self.gdict[vrtx] = []

    def AddEdge(self, edge):
        edge = set(edge)
        (vrtx1, vrtx2) = tuple(edge)
        if vrtx1 in self.gdict:
            self.gdict[vrtx1].append(vrtx2)
        else:
            self.gdict[vrtx1] = [vrtx2]

    def findedges(self):
        edgename = []
        for vrtx in self.gdict:
            for nxtvrtx in self.gdict[vrtx]:
                if {nxtvrtx, vrtx} not in edgename:
                    edgename.append({vrtx, nxtvrtx})
        return edgename

graph_elements = { "a" : ["b","c"],
                   "b" : ["a", "d"],
                   "c" : ["a", "d"],
                   "d" : ["e"],
                   "e" : ["d"]
                   }

g = graph(graph_elements)
g.AddEdge({'a','e'})
g.AddEdge({'a','c'})
print(g.edges())
```

Obtener vértices:

Para mostrar los vértices del grafo, simplemente busquemos las claves del diccionario de grafo. Usamos el método **keys()**.

Añadir un nuevo vértice:

Agregar un vértice es sencillo, solo agregamos otra clave adicional al diccionario del grafo.

Obtener Aristas:

Encontrar las aristas del grafo es un poco más complicado que los vértices, ya que tenemos que encontrar cada uno de los pares de vértices que tienen una arista entre ellos. Entonces creamos una lista vacía de aristas y luego iteramos a través de los valores de las aristas asociados con cada uno de los vértices. Se forma una lista que contiene el grupo distinto de aristas que se encuentran a partir de los vértices.

Añadir una nueva arista:

Agregar una nueva arista a un grafo existente implica tratar el nuevo vértice como una Tupla y validar si la arista ya está presente. De lo contrario, se agrega la arista.

Algoritmos importantes sobre Grafos:

- Algoritmo de Prim
- Algoritmo de Kruskal
- Algoritmo de Dijkstra
- Algoritmo de Búsqueda en Anchura (BFS)
- Algoritmo de Búsqueda en Profundidad (DFS)

¡Nos vemos en la próxima clase!

De esta manera finalizamos el bloque de Estructuras de Datos Avanzada y pasamos a uno de los últimos bloques del curso, Manejo de Módulos y Paquetes.