Reporte

Andrea Estefanía Galván Irigoyen **M.**1735583

Gpo. 001 Matemáticas Computacionales

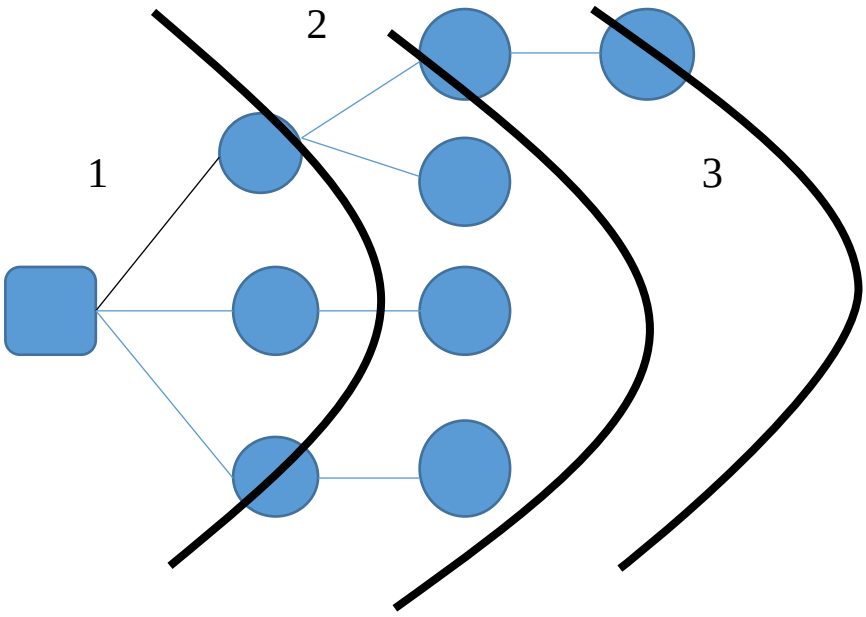
En este reporte se mostrará lo que es una estructura de datos y como se programará en Python, utilizando los programas de pila, fila y grafo así mismo especificando de lo que se tratan estos tres; con esto creando un BFS (búsqueda por amplitud) y DFS (búsqueda de profundidad)

La estructura de datos, o lo que nosotros utilizaremos, un grafo es un esquema en donde se presentan unos datos conectados entre ellos, como un árbol, conformado por vértices y aristas.

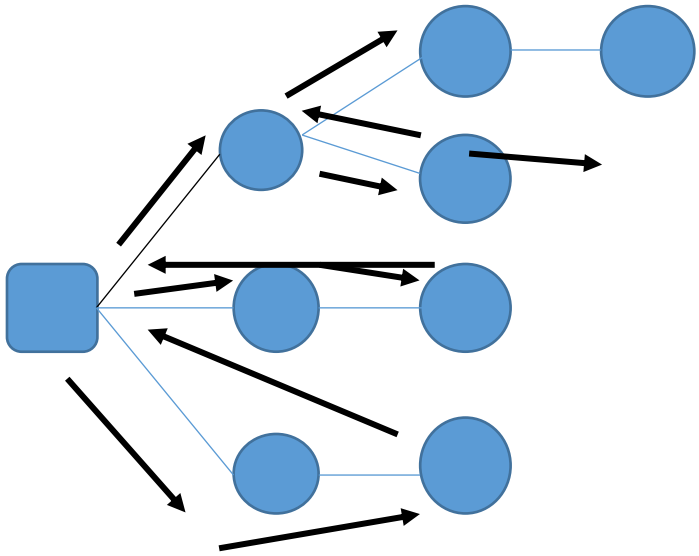
Una Fila es una estructura de datos de forma lineal en el que al ingresar el primer dato ese mismo será el primero en salir, y así sucesivamente con todos los datos, se ingresaran por la parte inferior de la fila.

Una Pila, al igual que la fila, es una estructura de datos pero en éste al querer ingresar algún dato se ingresan por los extremos y no por la parte inferior como se ingresan en la Fila y al igual si se quiere sacar algún dato sería por algún extremo.

Un BFS (búsqueda por amplitud) es una búsqueda en un grafo de tal manera que se visitan los vértices colocados en un tipo rango sobre el ancho del grafo como se muestra en la imagen



Un DFS (búsqueda por profundidad) es una búsqueda en un esquema de tal manera que para visitar todos los vértices primero se va con el más cercano, si ese vértice tiene un hijo el buscador se va hacia él, y así hasta que ya no pueda seguir y después de esto se regresa y sigue con el siguiente vértice así como se muestra en la imagen.



Programadas en Python

Para una búsqueda de BFS y DFS se ocupara una fila, pila y grafo y el código de dicha búsqueda.

```
class Fila:
        def __init__ (self):
                self.fila = []
        def obtener(self):
                return self.fila.pop()
        def meter(self,e):
                self.fila.insert(0,e)
                return len(self.fila)
        @property
        def longitud(self):
                return len(self.fila)
>>> class Pila:
        def __init__(self):
                self.pila = []
        def obtener(self):
                return self.pila.pop()
        def meter(self,e):
                self.pila.append(e)
                return len(self.pila)
        @property
        def longitud(self):
                return len(self.pila)
>>> class Grafo:
      def __init__(self):
       self.V = set() # un conjunto
       self.E = dict() # un mapeo de pesos de aristas
       self.vecinos = dict() # un mapeo


   def agrega(self, v):
       self.V.add(v)
       if not v in self.vecinos: # vecindad de v
          self.vecinos[v] = set() # inicialmente no tiene nada
```

```
    def conecta(self, v, u, peso=1):
     self.agrega(v)
     self.agrega(u)
     self.E[(v, u)] = self.E[(u, v)] = peso # en ambos sentidos
     self.vecinos[v].add(u)
     self.vecinos[u].add(v)
    def complemento(self):
     comp= Grafo()
     for v in self.V:
        for w in self.V:
           if v != w and (v, w) not in self.E:
              comp.conecta(v, w, 1)
     return comp
>>> def DFS(g,ni):
           visitados =[]
           f=Pila()
           f.meter(ni)
           while(f.longitud>0):
                    na =f.obtener()
                    visitados.append(na)
                    ln = g.vecinos[na]
                    for nodo in ln:
                             if nodo not in visitados:
                                      f.meter(nodo)
           return visitados
>>> def BFS(g,ni):
           visitados =[]
           f=Fila()
           f.meter(ni)
           while(f.longitud>0):
                    na =f.obtener()
                    visitados.append(na)
                    ln = g.vecinos[na]
                    for nodo in ln:
                             if nodo not in visitados:
```

```
                f.meter(nodo)
    return visitados
```