**Grafo**

Una forma de representar mediante esquemas gráficos los elementos que constituyen un sistema o conjunto llamados nodos o vértices, estos se encuentras unidos por enlaces llamados aristas. Los grafos nos sirven para medir o analizar la mejor ruta hacia algún punto partiendo de otro, gracias a que las aristas tienen un valor numérico, los nodos o vértices pueden ser tomados como un objeto de destino y las aristas el tiempo, distancia o la medida que se este tomando en que se tarde de nodo en nodo.

EL código en Python es de la siguiente forma:

**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 agregar(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.agregar(v)

self.agregar(u)

self.E[(v, u)] = self.E[(v, u)] = peso # en ambos sentidos

self.vecinos[v].add(u)

self.vecinos[u].add(v)

def complemento(self):

comp= Grafo()

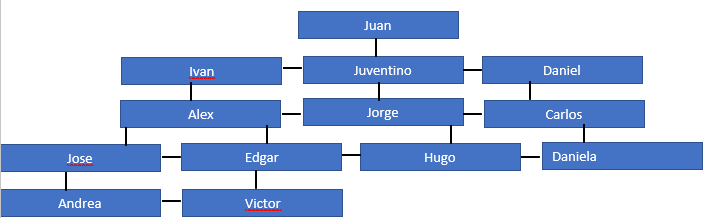
for v in self.V:

for w in self.V:

if v != w and (v, w) not in self.E:

comp.conecta(v, w, l)

return comp

Mi grafo es el siguiente:

**Pila**

Una pila es lista de datos organizados de una manera especifica sin variaciones, la cual es simple, pero tiene un problema el cual es este, al ingresar un objeto se coloca hasta arriba de la columna y para obtener uno se va sacando los de arriba de uno en uno, el problema es que al querer obtener un objeto en especifico si este se encuentra hasta lo más debajo de la pila será tardado.

EL código en Python es de la siguiente forma:

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)

Objeto 5

Objeto 3

Objeto 6

Objeto4

Objeto 2

Objeto 1

**Fila**

Una fila es lista de datos organizados de una manera específica sin variaciones, la cual es simple, pero tiene un problema el cual es este, al ingresar un objeto se coloca hasta arriba de la columna y para obtener uno se va sacando los de abajo de uno en uno, entonces si se desea obtener el ultimo objeto puesto en la fila se tendría que pasar por todos.

EL código en Python es de la siguiente forma:

class Fila:

def \_\_init\_\_(self):

self.fila = []

def obtener(self):

return self.fila.pop(0)

def meter(self,e):

self.fila.append(e)

return len(self.fila)

@property

def longitud(self):

return len(self.fila)













**Búsqueda por profundidad**

Es un método utilizado para verificar cada una de las rutas posibles desde un nodo en especifico hasta cada una de las ramas hacia abajo, sin pasar por el mismo nodo 2 veces.

EL código en Python es de la siguiente forma:

def DFS(g, ni):

visitados =[]

f= Pila()

f.meter(ni)

while(f.longitud > 0):

na = f.obtener()

if na not in visitados:

visitados.append(na)

ln = g.vecinos[na]

for nodo in ln:

if nodo not in visitados:

f.meter(nodo)

return visitados

**Búsqueda por amplitud**

Es un método utilizado para verificar cada una de las rutas posibles desde un nodo en específico hasta cada una de las ramas hacia la derecha pasando de nivel en nivel hacia abajo, sin pasar por el mismo nodo 2 veces.

EL código en Python es de la siguiente forma:

def BFS(g, ni):

visitados =[]

f= Fila()

f.meter(ni)

while(f.longitud > 0):

na = f.obtener()

if na not in visitados:

visitados.append(na)

ln = g.vecinos[na]

for nodo in ln:

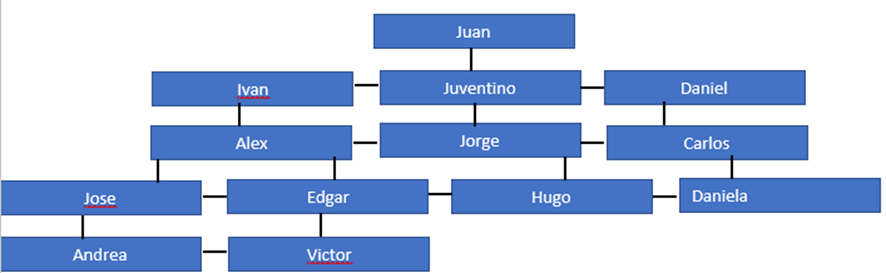
if nodo not in visitados:

f.meter(nodo)

return visitados

**Poniendo en marcha el programa**

Se utilizará el grafo hecho anteriormente.



Este grafo se programa de la siguiente manera:

g= Grafo()

g.conecta('juan','juventino')

g.conecta('juventino','ivan')

g.conecta('juventino','jorge')

g.conecta('juventino','daniel')

g.conecta('ivan','alex')

g.conecta('alex','jorge')

g.conecta('jorge','carlos')

g.conecta('alex','jose')

g.conecta('alex','edgar')

g.conecta('jorge','hugo')

g.conecta('jose','andrea')

g.conecta('edgar','victor')

g.conecta('hugo','daniela')

Aplicando el programa partiendo desde un nodo en aleatorio

X=random.choice(list(g.V))