Rafael Enrique Almaraz Martínez 06/10/2017

Reporte 2° Bloque

Para este segundo bloque en el curso de matemáticas computacionales trabajamos con 5 algoritmos: Fila, Pila, Grafo, BFS y DFS

Fila

```
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)
```

El código consiste en poder ingresar ciertos valores de modo que los acomode en un espacio en memoria por “fila” de manera que cuando uno quiera acceder a los valores de la mencionada fila se realice de uno por uno empezando del primer elemento hasta el último elemento.

Pila

```
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)
```

Trabaja de manera muy similar a fila con la diferencia de que este trabajaría a la inversa, si bien fila te mostraba los elementos del primer elemento hasta el ultimo elemento, en pila te los daría comenzando desde el último elemento hasta llegar al primer elemento que fue ingresado.

Grafo

```
class
Grafo:
            def __init__(self):
              self.V = set()
              self.E = dict()
              self.vecinos = dict()
            def agrega(self, v):
              self.V.add(v)
              if not v in self.vecinos:
                self.vecinos(v) = set()
            def conecta(self, v, u, peso=1):
              self.agrega(v)
              self.agrega(u)
              self.E[(v, u)] = self.E[(u,v)] = peso
              self.vecinos[v].add(u)
              self.vecinos[u].add(v)
```

La manera en la que yo relacione grafo, es como cuando se ilustra un torneo, o cuando hacemos un árbol genealógico, colocamos un elemento principal del cual se van desprendiendo otros elementos secundarios hasta donde sea necesario; algo así vendría trabajando el código de grafo definiendo un inicio del cual se enlacarían otros caminos hasta donde sea requerido.

BFS

```
def
B_F_S(g,ni):
                        visit=[]
                        f=Fila()
                        f.met(ni)
                        while(f.lon>0):
                                na=f.obt()
                                visit.append(na)
                                ra=g.vecinos[na]
```

```
                for nodo in ra:
                                        if nodo not in visit:
                                                f.met(nodo)
                        return visit
```

DFS

```
def
D_F_S(g,ni):
                        visit=[]
                        f=Pila()
                        f.met(ni)
                        while(f.lon>0):
                                na=f.obt()
                                visit.append(na)
                                ra=g.vecinos[na]
                                for nodo in ra:
                                        if nodo not in visit:
                                                f.met(nodo)
                        return visit
```

Por un lado BFS trabaja a la par con los algoritmos de grafo y fila donde vendría a buscar un cierto elemento dentro de cierto grafo definido de principio a fin; por el otro lado DFS trabaj con también con grafo pero en ligar de fila, es pila, donde también busca un elemento del grafo comenzando desde el final.