TRABAJO PRÁCTICO:

Algoritmos Genéticos

- **CARRERA:** **Ingeniería en informática.**
- **ASIGNATURA:** **Simulación y Modelización.**
- **DOCENTES:**
  - **Dr. ASTEASUAIN, Fernando.**
- **INTEGRANTE:**
  - **CALONGE, Federico.**

-

1. Introducción

1.1 Definición de Algoritmos Genéticos (AGs):

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están **basados en el proceso genético de los organismos vivos**. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Los Algoritmos Genéticos surgen por imitación de este proceso (donde las especies que sobreviven son las más APTAS al medio que las rodea: NO las mas fuertes); y son **capaces de ir creando soluciones para problemas del mundo real.** La **evolución de dichas soluciones hacia valores óptimos del problema** depende en buena medida de una adecuada codificación de las mismas. A diferencia de las redes neuronales, los AGs NO son algoritmos que aprenden por si solos.

La idea de los AGs empezar con una **población inicial** AL AZAR. Luego, debemos **eliminar** una porción de esta población y realizar **cruzamientos** con la otra porción para obtener otra nueva población; la cual heredará los genes "buenos" de las anteriores; y que de esta manera vaya "evolucionando". Además de esto, se realizarán **mutaciones** a una pequeña parte de nuestra población para obtener mejores resultados. En un determinado tiempo deberíamos llegar a un determinado "valor de **fitness**" (mediante una "función fitness", la cual definiremos nosotros) detendremos el cruzamiento y las mutaciones **para obtener la solución más óptima a nuestro problema** (osea, obtener el **"mejor" individuo de nuestra población final**).

1.2-Componentes básicos en un AG:

- Población: Son todas las posibles SOLUCIONES a nuestro problema. Una población es un conjunto de individuos.
- Individuo: Es UNA solución para nuestro problema.
- Mutación: Es una probabilidad. Son las mutaciones de 1 individuo. Se sugiere que sea menos del 5%. La mutación es random, por eso si es MUY grande entonces nuestro algoritmos genéticos serían random (por esto, NO es recomendable hacer muy grande la mutación).
- Cruzamiento/reproducción: se hace entre 2 individuos (entre 2 soluciones). De esta manera obtenemos 2 hijos.
- Función Fitness: función que nos dice que tan apto es un individuo. La función fitness nos permite evaluar a cada individuo/solución y **puntuarlo**. De esta manera empezamos con una población 0 (al azar), luego vamos a una población 1 (mejor) y así iteramos hasta obtener una población N (hasta parar por esta "función fitness" con algún criterio, por ej. sacando el promedio del fitness para cada población).

1.3 Objetivo del Trabajo Práctico:

El **objetivo** de este Trabajo Práctico es, mediante técnicas de AGs, asignar horarios y profesores para una lista de materias de la Universidad, intentando respetar las preferencias de los docentes. Sin embargo, podría ser posible que algún docente tenga que dar clases fuera de su horario de preferencia en caso de no haber otra alternativa. **Lo importante de la solución a la que lleguemos, es que abran TODAS las materias, pudiendo quedar profesores sin materias o profesores donde no les respetemos el horario.**

2. Explicación del código utilizado para el Algoritmo Genético:

Para codificar el Algoritmo Genético se utilizó el lenguaje de programación ***Python3***.

A continuación detallaremos todos los módulos y clases necesarios/as:

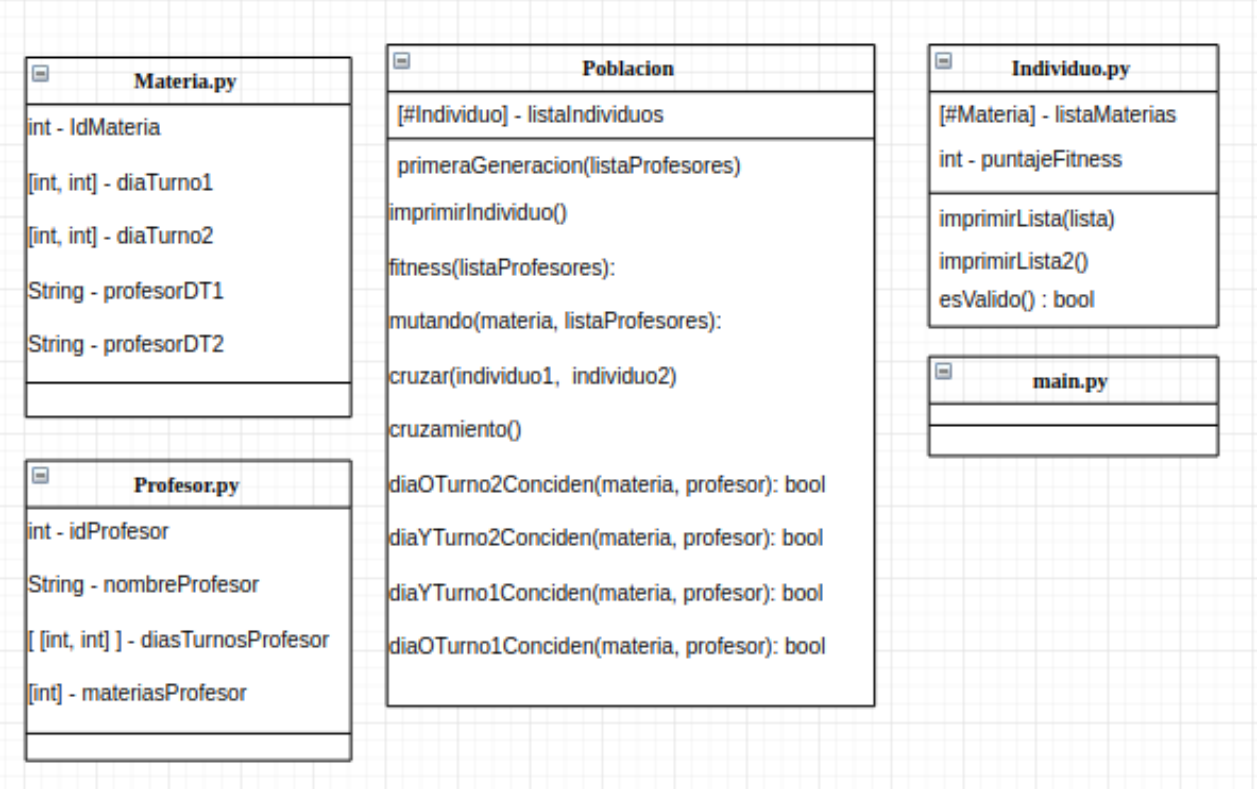

Diagrama de Clases

2.1-Modulo principal main.py:

main.py contiene las instrucciones principales de nuestro programa. De esta manera, se realizarán los siguientes pasos para encontrar la solución más óptima al problema:

Paso 1- Realizar la carga de datos (de los profesores).
Paso 2- Definir y generar nuestra **primera** población al azar.
Paso 3- Realizar un ciclo que consta de 4 sub-pasos:

Paso 3.1- Aplicar el método **fitness()** (la cual le da un cierto puntaje al Individuo dependiendo de los días, turnos y profesores asignados para cada materia).
Paso 3.2- Aplicar el método **cruzamiento()** (el cual eliminará una parte de la población y con el resto de la misma se obtendrán otras listas de individuos nuevos que reemplazarán a las eliminadas).
Paso 3.3- Aplicar el método **mutación(listaProfesores)** (el cual seleccionará una cantidad de individuos al azar de la población y les asignará parámetros randoms a todas sus materias)
Paso 3.4- Obtención del mejor individuo (solución más óptima).

Paso 1:

```
#################################### CARGA DE DATOS (profesores) ##############################

#Definimos nuestra lista de profesores (inicialmente vacia):
listaProfesores=[]

#Definimos 2 listas para el primer profesor (1-Kipper):
listaDiasTurnos1=[[1,2],[2,2],[3,2],[4,2],[5,2],[1,1],[2,1],[3,1],[4,1],[5,1]]
#Preferencia de horarios del profe. Dupla [dia,turno] donde "dia" es 1-lunes,...,5-viernes
#y "turno" es 1-manana o 2-tarde.
listaMaterias1=[15,16,17,18]  #Materias que puede dar el profe.

#Instanciamos un objeto Profesor (instanciamos al objeto Kipper):
Kipper = Profesor(1,'Kipper',listaDiasTurnos1,listaMaterias1)
listaProfesores.append(Kipper) #agregamos a la lista de profesores al objeto Kipper con sus atributos.

#Ahora hacemos lo mismo para el profesor 2 (Miguel):
#Definimos 2 listas para el segundo profesor (2-Miguel)
listaDiasTurnos2=[[3,2],[3,3],[5,3],[1,1],[2,1]]
listaMaterias2=[25,3,4,9,5]

#Instanciamos un objeto Profesor (instanciamos al objeto miguel):
Miguel = Profesor(2,'Miguel',listaDiasTurnos2,listaMaterias2)
listaProfesores.append(Miguel)
```

E hicimos de igual manera la carga para los demás 18 profesores.

Paso 2:

```
############################################ Definimos nuestra poblacion ###

#Instanciamos una poblacion:
poblacion = Poblacion()
#Generamos nuestra primera poblacion:
poblacion.primeraGeneracion(listaProfesores)
```

Paso 3:

```
###################### Aplicamos fitness, cruzamiento y mutacion para obtener la mejor lista ##############

listaEncontrada=False
ciclosEvolutivos=0

while(listaEncontrada == False):
    poblacion.fitness(listaProfesores)
    #El metodo de arriba nos devuelve la poblacion (la lista de individuos) ordenada en base a
    #su puntajeFitness de MAYOR a MENOR. De esta manera tenemos una lista con el MEJOR puntaje de fitness
    #en la 1ra posicion (indice 0), entonces agarramos este elemento y lo evaluamos hasta que su
    #puntajeFitness mayor o igual a 90 (condicion de corte).

    if((poblacion.listaIndividuos[0].puntajeFitness)>=90):
        listaEncontrada=True
    else:
        poblacion.cruzamiento()
        poblacion.mutacion(listaProfesores)

    ciclosEvolutivos=ciclosEvolutivos+1
    print "Ciclo numero: ", ciclosEvolutivos
    print "Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):"
    print poblacion.listaIndividuos[0].puntajeFitness

print "Mejor solucion (individuo) encontrada/o: "
print "Se necesitaron: ", ciclosEvolutivos, " ciclos para encontrarla."
print "Y la mejor lista es:"
poblacion.listaIndividuos[0].imprimirLista2() #Imprimimos la mejor lista (la 1ra, que tendra el mejor fitness)
```

2.2-Explicación de cada uno de los pasos:

Paso 1-
Al final de la carga de los 20 profesores obtendremos una *listaProfesores* con profesores de la clase *#Profesor* (*ver Sección 2.3- Profesor.py*) con su nombre, días y turnos en que puede dar la materia y las materias que puede dictar.

Paso 2-
Al instanciar la población generamos una *poblacion* de la clase *#Poblacion* (la cual tendrá al inicio una *listaIndividuos* vacía (*ver metodo __init__(self) de la Clase #Poblacion: Sección 2.6- Poblacion.py).*
Luego llamamos al metodo *primeraGeneracion* de la clase *#Poblacion* pasándole como parámetro la *listaProfesores* obtenida en el Paso 1-
Lo que hará este método es agregar a su atributo *listaIndividuos* una lista de 100 individuos (ver Sección *2.5- Individuo.py:*). Cada uno de estos individuos tendrán un *puntajeFitness* inicial de 0 y una *listaMaterias* con las 27 materias dadas como dato (las cuales tendrán los atributos *idMateria*, *diaTurno1*, *diaTurno2*, *profesorDT1* y *profesorDT2*: ver Sección 2.4-Materia.py*)*. **Estas materias se llenaran de manera aleatoria (random).**
Además, este método asegurará que el profesor da esa materia.

Paso 3-
Lo que hacemos para esto es aplicar un ciclo *while* que corte cuando hayamos encontrado la lista.
Lo primero que hacemos es evaluar el **fitness** de nuestra población con "*poblacion.fitness(listaProfesores)*". Dicho método nos devolverá la población (la lista de individuos) ordenada en base a su puntajeFitness de MAYOR a MENOR. De esta manera tenemos una lista con el MEJOR puntaje de fitness en la 1ra posicion (indice 0).
Luego agarramos este elemento y lo evaluamos hasta que su puntajeFitness mayor o igual a 90 (condicion de corte). Si ocurre esto (*if*) entonces ya hayamos nuestra lista.
Y si esto NO ocurre (*else*) entonces modificamos a la población **cruzándola** entre si y obteniendo nuevos hijos y nuevas poblaciones y además **mutando** una pequeña parte de la misma.

Luego de obtener nuestra lista "óptima", la imprimimos junto con los ciclos while que necesitamos para llegar a dicha solución.

Paso 3.1-Metodo fitness():
Este método dará un cierto puntaje al Individuo dependiendo de los días, turnos y profesores asignados para cada materia.

1. Recorremos la población de individuos previamente creada.
2. Si el individuo NO es valido le seteamos un puntajeFitness de 0.
3. Si el individuo ES valido entonces...:
    I. Si el individuo es valido, automáticamente le subimos UN punto al fitness.
    II. Si ademas de **I.** algún profesor asignado cumple con el día **O** el turno de la materia para la **1ra** tupla de dia-turno, le subimos OTRO punto al fitness.
    III. Si ademas de de **II.** algún profesor asignado cumple con el día **Y** turno de la materia para la **1ra** tupla de dia-turno, le subimos OTRO punto al fitness.
    IV. Si ademas de **I.** algún profesor asignado cumple con el día **O** el turno de la materia para la **2da** tupla de dia-turno, le subimos OTRO punto al fitness.
    V. Si ademas de **IV.** algún profesor asignado cumple con el día **Y** turno de la materia para la **2da** tupla de dia-turno, le subimos OTRO punto al fitness.

4. Luego de asignar los puntajes de Fitness a cada Individuo de la Poblacion, lo que hacemos es ORDENAR la Poblacion (la lista de Individuos) en base a su *puntajeFitness* (de MAYOR a MENOR). De esta manera con "*self.listaIndividuos.sort(key=lambda objeto: objeto.puntajeFitness, reverse=True)*" ordenamos la lista de Individuos en base a su atributo "*puntajeFitness*"). Así, las 1ras listas son las que tienen MAYOR puntajeFitness; y las últimas listas (la mayoría) tienen puntajeFitness=0.

Entonces como máximo el fitness llegará a 109, ya que… (para los **V** casos de arriba):

-Para el caso **I.** como máximo el fitness es **1** (ya que es un solo individuo).
-Y para cada uno de los demás casos (**II.**, **III.**, **IV.** y **V.)** el fitness puede llegar como máximo a 27 (ya que son 27 materias). Por esto, si se cumplen los 4 casos pueden dar 27x4=**108** de fitness como máximo.

Paso 3.2- Método cruzamiento():

Este método eliminará una parte de la población y con el resto de la misma se obtendrán otras listas de individuos nuevos que reemplazarán a las eliminadas. De esta manera:

1. Ordenamos nuevamente la lista de individuos según el fitness de cada una (mediante sort).
2. Removemos la 2da mitad (la mitad con los peores puntajes Fitness) con el "*for x in range(0, 50):*"
3. Usamos la 1ra mitad (las listas con los mejores puntajes Fitness) para obtener una lista nueva que reemplazara a la 2da mitad removida anteriormente con el "f*or x in range(0, 50,2)*:" y utilizando el método *cruzar(individuo1, individuo2).*

Paso 3.3- Método mutacion(listaProfesores):

Este método seleccionará una cantidad de individuos al azar de la población y les asignará parámetros randoms a todas sus materias. La mutación serà de un 10% de la población.

1. *Agarramos 10 individuos al azar mediante "individuoAzar = random.randint(1, 100)"*
2. Luego les cambiamos aleatoriamente el valor a todas sus materias mediante el método *mutando(materia, listaProfesores).*

Paso 3.4- Obtención del mejor individuo (solución más óptima):

Cuando nuestra "mejor" lista de de la lista de individuos (con el mayor puntaje fitness) haya llegado o superado los 90 (nuestra condición de corte), obtendremos nuestro mejor individuo; el cual imprimiremos mediante el método *imprimirLista2*() junto con los ciclos que necesitamos hasta encontrar dicha solución.

2.3- Profesor.py:

```
class Profesor(object):

    ''' Constructor '''
    def __init__(self,idProfesor,nombreProfesor,diasTurnosProfesor,materiasProfesor):
        self.idProfesor=idProfesor #int
        self.nombreProfesor=nombreProfesor #string
        self.diasTurnosProfesor=diasTurnosProfesor  #lista de duplas ints dia-turno
        self.materiasProfesor=materiasProfesor #lista de ints.

    '''Para poder imprimir el objeto Profesor en caso de necesitarlo:'''
    def __repr__(self):
        return "<ID:%s Nombre:%s Dias-Turnos:%s Materias:%s> " % (self.idProfesor, self.nombreProfesor, self.diasTurnosProfesor, self.materiasProfesor)
```

2.4-Materia.py:

```
class Materia(object):

    ''' Constructor '''
    def __init__(self,IdMateria,diaTurno1,diaTurno2,profesorDT1,profesorDT2):
        self.IdMateria=IdMateria #int
        self.diaTurno1=diaTurno1 #DUPLA de int dia e int turno.
        self.diaTurno2=diaTurno2
        self.profesorDT1=profesorDT1 #int.
        self.profesorDT2=profesorDT2

    '''Para poder imprimir el objeto Materia en caso de necesitarlo:'''
    def __repr__(self):
        return "<ID materia:%s Dia-Turno 1:%s Dia-Turno 2:%s Profesor para el D.T. 1:%s Profesor para el D.T. 2:%s> " %
        (self.IdMateria, self.diaTurno1, self.diaTurno2, self.profesorDT1, self.profesorDT2)
```

2.5-Individuo.py:

```
class Individuo(object):

    ''' Constructor '''
    def __init__(self):
        self.listaMaterias=[] #lista ints
        self.puntajeFitness=0 #int

    '''Para poder imprimir el objeto Individuo en caso de necesitarlo:'''
    def __repr__(self):
        return "<Lista Materias:%s Puntaje Fitness:%s> " % (self.imprimirLista(self.listaMaterias), self.puntajeFitness)
```

2.6-Poblacion.py:

```
class Poblacion(object):

    ''' Constructor '''
    def __init__(self):
        self.listaIndividuos=[] #lista de individuos (de clase Individuo). MI POBLACION. Al principio esta vacia.

    '''Para poder imprimir el objeto Poblacion en caso de necesitarlo:'''
    def __repr__(self):
        return "<Lista Individuos:%s> " % (self.imprimirIndividuo(self.listaIndividuos))
```

***Nota:** Los métodos para 2.5-Individuo.py y 2.6-Poblacion.py están explicados detalladamente en el código con #comentarios.*

3. Ejecución:

En esta ocasión se ejecutó con el Sublime Text (teniendo Python3 instalado) en Ubuntu. Para esto, abrimos el “main.py” desde Sublime Text y vamos a la pestaña “Tools” y luego a “Build”.

Sino, teniendo Python3 instalado, lo que hacemos es ubicarnos en la carpeta donde esté nuestro main.py (en mi caso TP-Algoritmos-Geneticos) utilizando ***cd***. Y luego poner lo siguiente:

```
federicio@federiciohp:~/Escritorio/TP7-Geneticos/TP-Algoritmos-Geneticos$ python3 main.py
```

Para nuestra **1ra ejecución** tardamos 55 ciclos en llegar a una de solución “óptima” para el problema. Aproximadamente Python tardó (en mi máquina) 2.30 segundos para recorrer 1 solo ciclo. Entonces, para recorrer los 55 ciclos deberían pasar: 126,5 segundos = 2.10 minutos. = aproximadamente los 127.4 segundos que nos devolvió Python (ver al final de la ejecución).

Desde el ciclo 1 hasta el 9no nuestro “mejor” Individuo de nuestra Población de 100 tuvo un fitness máximo de 85 (hasta el 9no ciclo NO hubieron cambios):

```
Ciclo numero:  1
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
85
Ciclo numero:  2
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
85
```

Pero luego, debido a la reproducción y mutación, nuestro fitness aumentó un punto en el 10mo ciclo:

```
Ciclo numero:  10
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
86
Ciclo numero:  11
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
86
```

Y para el 17vo ciclo llegamos a 87:

```
Ciclo numero:  17
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
87
Ciclo numero:  18
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
87
```

Por ultimo, llegamos a un fitness de 90 en el ciclo 55:

```
Ciclo numero:  55
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
90
Mejor solucion (individuo) encontrada/o:
Se necesitaron:  55  ciclos para encontrarla.
Y la mejor lista es:
```

Devolviéndonos la solución al problema (asignación de profesores para las 27 materias):

```
La materia:  1
Se da el dia (Dia 1):  1  en el turno (Turno 1):  2 , teniendo como profesor a:  20
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  20
La materia:  2
Se da el dia (Dia 1):  2  en el turno (Turno 1):  1 , teniendo como profesor a:  17
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  2  teniendo como profesor a:  20
La materia:  3
Se da el dia (Dia 1):  2  en el turno (Turno 1):  1 , teniendo como profesor a:  2
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  6
La materia:  4
Se da el dia (Dia 1):  1  en el turno (Turno 1):  2 , teniendo como profesor a:  18
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  1  teniendo como profesor a:  2
La materia:  5
Se da el dia (Dia 1):  4  en el turno (Turno 1):  1 , teniendo como profesor a:  12
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  2  teniendo como profesor a:  12
La materia:  6
Se da el dia (Dia 1):  4  en el turno (Turno 1):  2 , teniendo como profesor a:  3
Y se da el dia (Dia 2):  4  en el turno (Turno 2):  1  teniendo como profesor a:  3
La materia:  7
Se da el dia (Dia 1):  4  en el turno (Turno 1):  1 , teniendo como profesor a:  3
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  1  teniendo como profesor a:  3
La materia:  8
Se da el dia (Dia 1):  3  en el turno (Turno 1):  2 , teniendo como profesor a:  3
Y se da el dia (Dia 2):  3  en el turno (Turno 2):  1  teniendo como profesor a:  3
La materia:  9
Se da el dia (Dia 1):  5  en el turno (Turno 1):  1 , teniendo como profesor a:  18
Y se da el dia (Dia 2):  3  en el turno (Turno 2):  1  teniendo como profesor a:  13
La materia:  10
Se da el dia (Dia 1):  4  en el turno (Turno 1):  2 , teniendo como profesor a:  5
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  5
La materia:  11
Se da el dia (Dia 1):  2  en el turno (Turno 1):  1 , teniendo como profesor a:  16
Y se da el dia (Dia 2):  3  en el turno (Turno 2):  2  teniendo como profesor a:  10
La materia:  12
Se da el dia (Dia 1):  1  en el turno (Turno 1):  1 , teniendo como profesor a:  6
Y se da el dia (Dia 2):  3  en el turno (Turno 2):  2  teniendo como profesor a:  6
La materia:  13
Se da el dia (Dia 1):  5  en el turno (Turno 1):  1 , teniendo como profesor a:  8
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  2  teniendo como profesor a:  8
La materia:  14
Se da el dia (Dia 1):  5  en el turno (Turno 1):  1 , teniendo como profesor a:  8
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  8
La materia:  15
Se da el dia (Dia 1):  4  en el turno (Turno 1):  2 , teniendo como profesor a:  1
Y se da el dia (Dia 2):  4  en el turno (Turno 2):  2  teniendo como profesor a:  1
La materia:  16
Se da el dia (Dia 1):  4  en el turno (Turno 1):  2 , teniendo como profesor a:  1
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  1
La materia:  17
Se da el dia (Dia 1):  1  en el turno (Turno 1):  1 , teniendo como profesor a:  1
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  2  teniendo como profesor a:  1
La materia:  18
Se da el dia (Dia 1):  2  en el turno (Turno 1):  2 , teniendo como profesor a:  19
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  2  teniendo como profesor a:  1
```

```
La materia:  19
Se da el dia (Dia 1):  2  en el turno (Turno 1):  1 , teniendo como profesor a:  7
Y se da el dia (Dia 2):  3  en el turno (Turno 2):  2  teniendo como profesor a:  7
La materia:  20
Se da el dia (Dia 1):  3  en el turno (Turno 1):  1 , teniendo como profesor a:  13
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  2  teniendo como profesor a:  13
La materia:  21
Se da el dia (Dia 1):  4  en el turno (Turno 1):  1 , teniendo como profesor a:  15
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  2  teniendo como profesor a:  19
La materia:  22
Se da el dia (Dia 1):  1  en el turno (Turno 1):  1 , teniendo como profesor a:  16
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  2  teniendo como profesor a:  16
La materia:  23
Se da el dia (Dia 1):  5  en el turno (Turno 1):  1 , teniendo como profesor a:  8
Y se da el dia (Dia 2):  2  en el turno (Turno 2):  1  teniendo como profesor a:  8
La materia:  24
Se da el dia (Dia 1):  3  en el turno (Turno 1):  1 , teniendo como profesor a:  9
Y se da el dia (Dia 2):  5  en el turno (Turno 2):  1  teniendo como profesor a:  9
La materia:  25
Se da el dia (Dia 1):  1  en el turno (Turno 1):  1 , teniendo como profesor a:  19
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  2  teniendo como profesor a:  2
La materia:  26
Se da el dia (Dia 1):  5  en el turno (Turno 1):  2 , teniendo como profesor a:  4
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  1  teniendo como profesor a:  4
La materia:  27
Se da el dia (Dia 1):  5  en el turno (Turno 1):  2 , teniendo como profesor a:  11
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  1  teniendo como profesor a:  11
[Finished in 127.4s]
```

Hay veces que en un solo ciclo nuestro programa ya encontró la lista con un fitness mínimo de 90:

```
Ciclo numero:  1
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
92
Mejor solucion (individuo) encontrada/o:
Se necesitaron:  1  ciclos para encontrarla.
Y la mejor lista es:
El fitness del individuo es:  92
```

Y hay otras veces en que al inicio tenemos un fitness máximo de 0 y se tarda muchísimo en llegar a los 90, por eso lo recomendable cuando ocurre esto es correr al programa nuevamente (en el caso de abajo igualmente tuvimos “suerte” debido a la reproducción y mutación y ya en el ciclo 6 llego a un fitness de 87):

```
Ciclo numero:  1
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
0
Ciclo numero:  2
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
0
Ciclo numero:  3
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
0
Ciclo numero:  4
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
0
Ciclo numero:  5
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
0
Ciclo numero:  6
Puntaje Fitness del 1er elemento de la lista de individuos (el de MEJOR fitness):
87
```

Comparamos los resultados obtenidos de nuestra **1ra ejecución** a ver si corresponden a la lista que nos dieron inicialmente:

Hay casos donde la asignación de materias **coinciden exactamente** con los horarios en que los profesores prefieren dar las mismas. Por ejemplo, para la materia 4:

```
La materia:  4
Se da el dia (Dia 1):  1  en el turno (Turno 1):  2 , teniendo como profesor a:  18
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  1  teniendo como profesor a:  2
```

Vemos las duplas de los profesores 2 y 18:

```
listaDiasTurnos2=[[3,2],[3,3],[5,3],[1,1],[2,1]]
listaMaterias2=[25,3,4,9,5]

#Instanciamos un objeto Profesor (instanciamos al objeto miguel):
Miguel = Profesor(2,'Miguel',listaDiasTurnos2,listaMaterias2)
listaProfesores.append(Miguel)
```

```
listaDiasTurnos18=[[1,2],[2,2],[3,2],[4,2],[5,2]]
listaMaterias18=[3,4,5,9]
Smith = Profesor(18,'Smith',listaDiasTurnos18,listaMaterias18)
listaProfesores.append(Smith)
```

De esta manera, vemos que la materia 4 se dictará el Lunes (día 1) a la tarde (turno 2) y el profesor será el 18. Y también se dictará el Lunes (día 1) a la mañana (turno 1) y el profesor sera el 2.

Comparando los resultados obtenidos con las duplas iniciales de los profesores 18 y 2, vemos que ambos profesores dan la materia 4. Y además, vemos que **las preferencias horarias también coinciden exactamente** (dupla [1,2] en el profesor 18; y dupla [1,1] en el profesor 2).

Pero también, hay casos donde la asignación de materias **NO coincide exactamente** con los horarios en que los profesores prefieren dar las mismas. Por ejemplo, para la materia 25:

```
La materia:  25
Se da el dia (Dia 1):  1  en el turno (Turno 1):  1 , teniendo como profesor a:  19
Y se da el dia (Dia 2):  1  en el turno (Turno 2):  2  teniendo como profesor a:  2
```

Vemos las duplas de los profesores 2 y 19:

```
listaDiasTurnos2=[[3,2],[3,3],[5,3],[1,1],[2,1]]
listaMaterias2=[25,3,4,9,5]

#Instanciamos un objeto Profesor (instanciamos al objeto miguel):
Miguel = Profesor(2,'Miguel',listaDiasTurnos2,listaMaterias2)
listaProfesores.append(Miguel)
```

```
listaDiasTurnos19=[[1,1],[2,2]]
listaMaterias19=[9,18,21,25]
Perez = Profesor(19,'Perez',listaDiasTurnos19,listaMaterias19)
listaProfesores.append(Perez)
```

De esta manera, vemos que la materia 25 se dictará el Lunes (día 1) a la mañana (turno 1) y el profesor sera el 19. Y también se dictará el lunes (día 1) a la tarde (turno 2) con el profesor 2.

Comparando los resultados obtenidos con las duplas iniciales de los profesores 19 y 2, vemos que ambos profesores dan la materia 25. Pero además, vemos que las preferencias horarias no coinciden exactamente (la dupla [1,1] está en el profesor 19; pero la dupla [1,2] NO está en el profesor 2).

4. Conclusiones:

Hubieron casos en que nuestro programa NO acató las preferencias horarias de los profesores (Ver *5. Mejoras:* **fitness**). Igualmente, se comprobó manualmente que **SI llegamos a una solución esperada** ya que NUNCA hubieron materias asignadas a profesores que no las den; y además NO hubieron materias que se dictaron en el mismo horario (día y turno): esto es debido al método ***esValido*** de la clase ***#Individuo***.

5. Mejoras:

- Se debería mejorar el método ***fitness*** de la clase ***#Poblacion*** para que la asignación de fitness sea mas precisa, pues no siempre acata a las preferencias horarias de los docentes. Otra opción es **aumentar el numero del fitness del ciclo de corte en el main hasta un máximo de 109** (en nuestro caso era de 90). Pero la desventaja es que se tardará mayor tiempo / mayor cantidad de ciclos en llegar a ese numero.
- Que la **mutación** tenga en cuenta el puntaje de fitness (ya que en nuestro caso no sucede esto y podría mutar a un individuo con alto valor de Fitness). En una ejecución ocurrió que el puntajeFitness maximo era 88 y luego disminuyó a 84 (esto fue debido a que se mutó al Individuo con puntajeFitness de 88). Pero igualmente hay una baja probabilidad de que ocurra esto.
- Usar ***UnitTest*** envés de testear en el main.
- Usar la librería ***openpyxl*** para armar automáticamente una tabla en Excel con las preferencias horarias que nos devuelve Python en nuestro programa.