



ALGORITMO GENÉTICO

Inteligencia artificial

DESCRIPCIÓN BREVE

Documentación de programa que implementa el algoritmo genético para instrumentos de inversión.

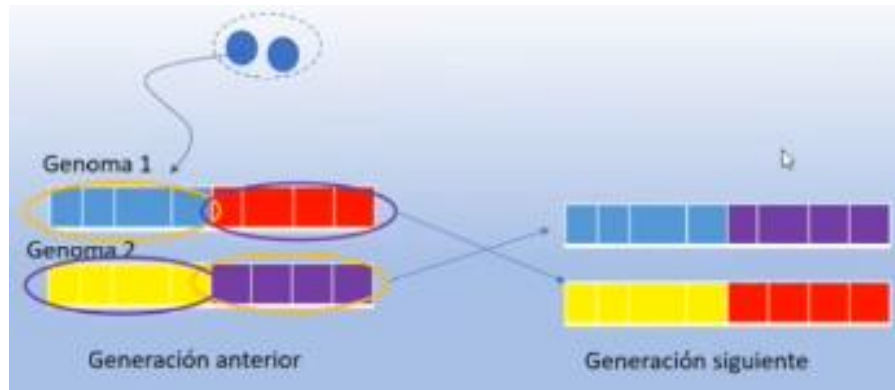
Daniel Salazar

Universidad Autónoma de Coahuila – Facultad de sistemas – Profesor: Ernesto Guiñan

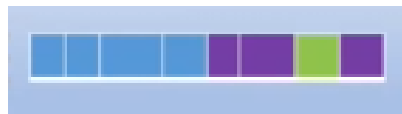
¿Qué es un algoritmo genético?

Un algoritmo genético es un algoritmo en el cual dentro de una generación se tienen varios individuos y a través de dos individuos se crean otros dos individuos, y de vez en cuando puede suceder una mutación, estos dos individuos creados formarían parte de la segunda generación, esto se continuaría hasta que todos los individuos en la generación anterior hayan sido mezclados para crear sus dos hijos para la siguiente generación.

Cada uno de estos individuos tiene genes, entonces al combinarlos se combinan los genes de cada individuo, algo así:



Teniendo un cruce de genes de cada individuo, y el individuo creado tendría genes de cada uno de sus padres. También como se comentó puede haber mutaciones, donde pasaría algo así:



Teniendo un individuo con uno de sus genes mutado.

Este proceso se realiza hasta n generaciones.

Algoritmo genético para instrumentos de inversión:

Teniendo distintos instrumentos de inversión, con su inversión requerida y su rendimiento, nosotros tenemos que crear una primera generación de forma aleatoria, y seguidamente empezar a mezclar los individuos de tu generación actual para generar la siguiente. Contando con que puede haber mutaciones en alguna ocasión. Al final tenemos que evaluar cual fue el mejor individuo creado que nos de el mejor rendimiento sin pasar de un límite de inversión que te da el enunciado del problema.

Ejemplo de problema de uso de algoritmo genético para inversiones:

1.- Un inversionista desea maximizar el rendimiento de su capital, cuenta con 6 tipos de inversiones (ver la tabla). El monto máximo que desea invertir en los distintos instrumentos es de 30,000. Aplique algoritmos genéticos para obtener el mejor esquema de inversión para obtener el máximo rendimiento posible. Genere 4 generaciones de 6 individuos cada generación. Aplique mutación a un individuo por cada generación, el punto de cruce debe ser aleatorio (no fijo).

Instrumento	Rendimiento del monto invertido	Monto mínimo	Rendimiento en \$
1	10%	1,500	\$150
2	11%	5,000	\$550
3	13%	10000	\$1300
4	10%	8000	\$800
5	9.5%	6000	\$570
6	10%	5000	\$500

GENERACIÓN 2

GENERACIÓN 1

		Inversión	Rendimiento
1	1 1 0 1 0 1	14,500	\$1,500
	1 0 0 0 0 1	19,000	\$1,870
	0 0 0 1 1 0	14,500	\$1,450
2	0 1 0 1 0 1	10,000	\$1,050
	1 0 0 0 0 0	22,500	\$2,520
3	1 1 1 1 1 1	6,500	\$700

La primera generación es creada aleatoriamente, y la segunda sale de combinaciones de individuos, con distintos puntos de cruce. Después se saca la inversión y rendimiento de la generación que fue creada.

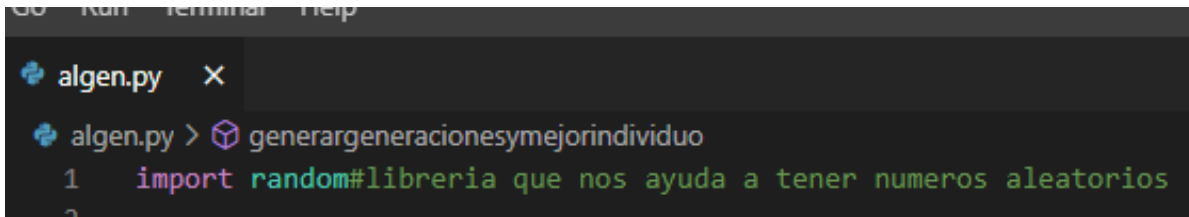
MEJOR OPCIÓN:

GENERACIÓN 4

	Inversión	Rendimiento
1 1 1 1 0 1	\$29,500	\$3,300

Al final sacamos la mejor opción, la cual concluimos que estuvo en la generación 4, sin rebasar el límite de inversión y generando la mayor cantidad de rendimiento.

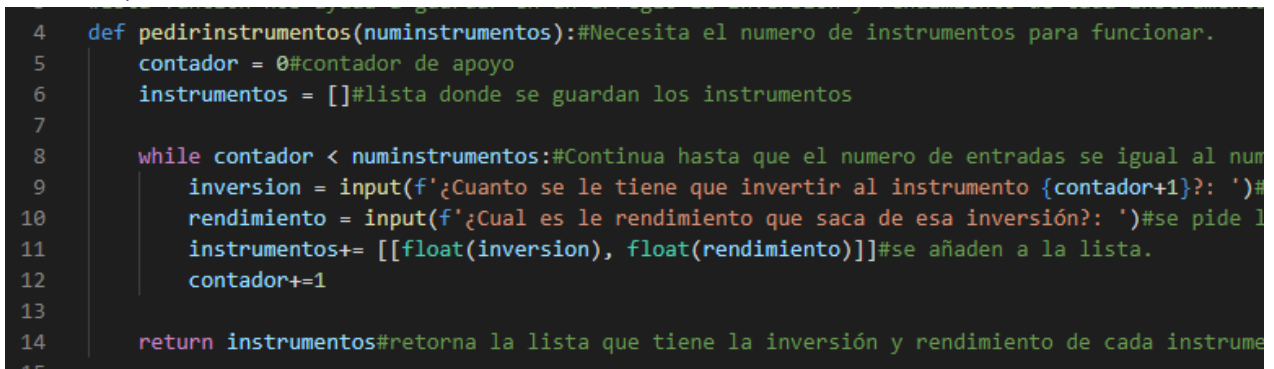
Documentación del programa que implementa el algoritmo genético:



```
algen.py x
algen.py > generargeneracionesymejorindividuo
1 import random#libreria que nos ayuda a tener numeros aleatorios
2
```

El programa comienza con la importación de la librería random, la cual contiene métodos que nos ayudan a generar números aleatorios.

Función pedirinstrumentos:



```
4 def pedirinstrumentos(numinstrumentos):#Necesita el numero de instrumentos para funcionar.
5     contador = 0#contador de apoyo
6     instrumentos = []#lista donde se guardan los instrumentos
7
8     while contador < numinstrumentos:#Continúa hasta que el numero de entradas se igual al num
9         inversion = input(f'¿Cuanto se le tiene que invertir al instrumento {contador+1}?: ')#
10        rendimiento = input(f'¿Cual es le rendimiento que saca de esa inversión?: ')#se pide l
11        instrumentos+= [[float(inversion), float(rendimiento)]]#se añaden a la lista.
12        contador+=1
13
14    return instrumentos#retorna la lista que tiene la inversión y rendimiento de cada instrume
15
```

Variables que ocupa para funcionar:

- numinstrumentos: variable que nos dice el numero de instrumentos que se usarán.

Objetivo de la función: Guardar la inversión y el rendimiento de cada uno de los instrumentos que vayan a utilizarse.

Variables:

- contador: contador de apoyo para un ciclo.
- Instrumentos: lista donde se guarda la inversión y rendimiento de cada instrumento.
- Inversión: variable donde se guarda la inversión del instrumento.
- rendimiento: variable donde se guarda el rendimiento en cantidad (no porcentajes) del instrumento.

Resultado de ejemplo de la función:

```
[[150, 75], [500, 250], [100, 50]]
```

Esto es un ejemplo del resultado de la función donde “numinstrumentos” es igual a tres, dándonos una lista que guarda 3 listas donde esta la inversión y rendimiento de cada instrumento.

Función primerageneración:

```
17 def primerageneracion(numinstrumentos, instrumentos):#necesita
18     contadorprincipal = 0#contador de apoyo
19     generacion = []#lista donde se guarda la primera generacion
20     generaciones = []#lista donde se guardaran todas las generaciones
21     individuo = []#lista donde se guardaran los datos de cada individuo
22     inversion = 0#sirve para guardar la inversión de un individuo
23     rendimiento = 0#sirve para guardar el rendimiento de un individuo
24
25     while contadorprincipal < numinstrumentos:#el ciclo de los instrumentos
26
27         contador1 = 0#contador para los individuos dentro de cada instrumento
28
29         while contador1 < numinstrumentos:#el ciclo de los individuos
30             individuo += [random.randint(0, 1)]#cada individuo se genera aleatoriamente
31             contador1 +=1
32             #de aqui puede salir algo asi: [0, 1, 0] que es un individuo
33
34             contador2 = 0#contador para la inversión y rendimiento de cada individuo
35
36             for gen in individuo:#el ciclo pasa por cada gen del individuo
37                 if gen == 1:
38                     inversion += instrumentos[contador2][0]#se suma la inversión
39                     rendimiento += instrumentos[contador2][1]#se suma el rendimiento
40                     contador2+=1
41             #se calcula la inversión y el rendimiento de cada individuo
42
43             contadorprincipal+=1
44
45             individuo += [inversion, rendimiento]#aquí se junta la inversión y el rendimiento
46
47             generacion += [individuo]#se ingresa el individuo a la generacion
48
49             individuo = []#se limpia el individuo para generar el siguiente
50             inversion = 0#se regresa a cero para calcular la inversión
51             rendimiento = 0#se regresa a cero para calcular el rendimiento
52
53             generaciones += [generacion]#se ingresa la generacion a la lista de generaciones
54
55     return generaciones#se retorna la lista que contiene todas las generaciones
```

Variables que ocupa para funcionar:

- numinstrumentos: variable que nos dice el número de instrumentos en el ejercicio.
- Instrumentos: lista que contiene la inversión y rendimiento de cada instrumento.

Objetivo de la función: Crear la primera generación de individuos aleatoriamente.

Variables:

- contadorprincipal: contador de apoyo para el ciclo while principal.
- generación: lista donde se guarda la generación creada.
- generaciones: lista donde se guardan todas las generaciones.
- individuo: lista donde se guarda el individuo con todo y la inversión que ocupa y el rendimiento que genera.
- Inversión: variable donde se guarda la inversión del individuo.
- rendimiento: variable donde se guarda el rendimiento del individuo.

Explicación de la función:

Se tiene un ciclo principal el cual hace un numero de recorridos igual al numero de instrumentos en el ejercicio, para así si hay tres instrumentos generar tres individuos de tres genes por generación.

El segundo ciclo while crea aleatoriamente el gen de cada individuo, el cual puede ser 1 o 0, los va concatenando en la lista individuo y así se crea un individuo.

El tercer ciclo que es un for suma la inversión y el rendimiento que genera el individuo anteriormente creado. Si el gen es igual a 0 quiere decir que no se suma nada, si es igual a 1 si se suma su inversión y rendimiento correspondiente al instrumento que representa.

Resultado de ejemplo de la función:

```
[[[0, 0, 1, 100, 50], [1, 0, 1, 250, 125] [0, 1, 1, 600, 300]]]
```

Este resultado podría ser creado con esta función si en el ejercicio hubiera 3 instrumentos, el instrumento 1 tuviera una inversión de 150 y un rendimiento de 75, el instrumento 2 una inversión de 500 y un rendimiento de 250, y el instrumento 3 una inversión de 100 y un rendimiento de 50.

Función generargeneracionesymejorindividuo:

```
58 def generargeneracionesymejorindividuo(numinstrumentos, instrumentos, generaciones, numgeneraciones, inversionmaxima):
59     #Esta función necesita el numero de instrumentos, la lista de los instrumentos, la lista de las generaciones, el numero de generaciones
60     contadorprincipal = 0#contador principal de los ciclos
61     contadorsecundario = 0#contador secundario de los ciclos
62     generacionanterior = generaciones[0]#guarda la generación que será usada para crear la siguiente generación
63     contadoruno = 0#contador para referirse al primer individuo que se va a combinar
64     contadordos = 1#contador para referirse al segundo individuo que se va a combinar
65     contadorinversiones1 = 0#contador para las inversiones del primer individuo nuevo generado
66     contadorinversiones2 = 0#contador para las inversiones del segundo individuo nuevo generado
67     generacion = []#guarda la generación actual
68     individuo = []#guarda el primer individuo de una combinación generado
69     individuo2 = []#guarda el segundo individuo de una combinación generado
70     inversion = 0#guarda la inversión de un individuo
71     rendimiento = 0#guarda el rendimiento de un individuo
72     inversion2 = 0#guarda la inversión del segundo individuo de una combinación
73     rendimiento2 = 0#guarda el rendimiento del segundo individuo de una combinación
74     numindividuo = 0#nos dice cuantos individuos se han creado
75     contadorinversiones3 = 0#contador para la inversión y rendimiento de un individuo que sale de una mutación
76     mejoropcion = []#guarda el individuo que es mejor opción, ganando más rendimiento sin revasar la inversión maxima
77     contadoropcion = 0#contador para apoyar la mejor opcion inicial
78
79     while contadoropcion < numinstrumentos+2:#es como un constructor de la lista mejor opción, la llena de 0 dependiendo el numero de instr
80         mejoropcion = mejoropcion + [0]
81         contadoropcion += 1
82
83     print(f'Primera generacion generada aleatoriamente: {generacionanterior}')#se imprime la primera generación
84
85     while contadorprincipal < numgeneraciones-1:#El ciclo funciona hasta que sean creadas todas las generaciones solicitadas
86
87         for gen in generaciones:#se chequea cual es la mejor opción en cada individuo de las generaciones creadas hasta ahora.
88             for ind in gen:
89                 if ind[-2] <= inversionmaxima and ind[-1] >= mejoropcion[-1]:
90                     mejoropcion = ind
91
92         print(f'La mejor opción por ahora es: {mejoropcion[:-2]}, con inversión de: {mejoropcion[-2]} y rendimiento de: {mejoropcion[-1]}')
93
94         contadorsecundario = numinstrumentos#se inicializa el contador secundario cada corrida del ciclo.
95
96         print(f'Generación de donde vienen los padres y o muta algún individuo: {generacionanterior}')#se imprime la generación de donde se
97
98         while contadorsecundario > 0:#El ciclo funciona hasta que se chequen todos los instrumentos en la generación pasada.
99
100             if contadorsecundario > 1:#si quedan dos o más individuos para combinar se entra aqui.
101
102                 limite = len(generacionanterior[numindividuo]) - 3#se genera el limite del punto de cruce, este por que si los individuos se
103                 #se podría seleccionar hasta el 2, ya que en 3 esta la inversion y en 4 el rendimiento.
104
105                 puntodecruce = random.randint(1, limite)#se genera el punto de cruce, de 1 a el limite generado an
```

```
105                 puntodecruce = random.randint(1, limite)#se genera el punto de cruce, de 1 a el limite generado an
106
107                 print(f'Padres: {generacionanterior[contadoruno][:-2]} y {generacionanterior[contadordos][:-2]}')#
108
109                 print(f'El punto de cruce es: {puntodecruce}')#imprime el punto de cruce
110
111                 print(f'Parte 1: {generacionanterior[contadoruno][:puntodecruce]}')#imprime la parte uno del nuevo
112                 print(f'Parte 2: {generacionanterior[contadordos][puntodecruce:-2]}')#imprime la segunda parte del
113
114                 individuo = individuo + generacionanterior[contadoruno][:puntodecruce]#se concatena la primera par
115                 individuo = individuo + generacionanterior[contadordos][puntodecruce:-2]#se concatena la segunda p
116
117                 for gen in individuo:#se saca la inversión y el rendimiento del nuevo individuo
118                     if gen == 1:
119                         inversion += instrumentos[contadorinversiones1][0]
120                         rendimiento += instrumentos[contadorinversiones1][1]
121                         contadorinversiones1+=1
122
123                 individuo += [inversion, rendimiento]#se concatena la inversión y el rendimiento al individuo.
```



```

124
125     print(f'Hijo uno : {individuo}')#se imprime el individuo creado.
126
127     print(f'Parte 1: {generacionanterior[contadordos][:puntodecruce]}')#imprime
128     print(f'Parte 2: {generacionanterior[contadoruno][puntodecruce:-2]}')#imprime
129
130     individuo2 = individuo2 + generacionanterior[contadordos][:puntodecruce]#se
131     individuo2 = individuo2 + generacionanterior[contadoruno][puntodecruce:-2]#
132
133     for gen in individuo2:#se saca la inversión y el rendimiento del segundo nu
134         if gen == 1:
135             inversion2 += instrumentos[contadorinversiones2][0]
136             rendimiento2 += instrumentos[contadorinversiones2][1]
137             contadorinversiones2+=1
138
139     individuo2 += [inversion2, rendimiento2]#se concatena la inversión el rendi
140
141     print(f'Hijo dos : {individuo2}')#se imprime el segundo nuevo individuo cre
142
143     contadoruno += 2
144     contadordos += 2
145     #estos contadores apoyan a saber que individuos son padres de los que se es
146     #la segunda vez serán el individuo 2 y el individuo 3, y así continuamente.
147
148     #se añaden los individuos creados de combinaciones a la generación actual.
149     generacion += [individuo]
150     generacion += [individuo2]
151
152     #se limpian las listas de cada individuo para que se puedan crear sin resid
153     individuo = []
154     individuo2 = []
155
156     #se suman los dos individuos creados a numero de individuos y se quitan dos
157     numindividuo+=2
158     contadorsecundario-=2
159
160     #se regresan a su estado inicial todas las variables auxiliares usadas.
161     inversion2 = 0
162     inversion = 0
163     rendimiento2 = 0
164     rendimiento = 0
165     contadorinversiones1 = 0
166     contadorinversiones2 = 0

```



```

167
168         else:#aquí solo se entra si no hay suficientes individuos para combinar, enton
169
170             genquemuta = random.randint(0, len(generacionanterior[numindividuo])-3)#se
171
172             individuo = []#se establece el individuo como una lista vacia
173
174             individuo = individuo + generacionanterior[numindividuo][:-2]#se llena el
175
176             print(f'Individuo que va a mutar: {generacionanterior[numindividuo][:-2]}')
177
178             if individuo[genquemuta] == 0:#muta el gen seleccionado, si era 0 pasa a 1
179                 individuo[genquemuta] = 1
180             else:
181                 individuo[genquemuta] = 0
182
183             for gen in individuo:#se calcula la inversión y rendimiento del individuo
184                 if gen == 1:
185                     inversion += instrumentos[contadorinversiones3][0]
186                     rendimiento += instrumentos[contadorinversiones3][1]
187                     contadorinversiones3+=1
188
189             individuo += [inversion, rendimiento]#se concatenan la inversión y el rend
190
191             print(f'Individuo con gen mutado : {individuo}')#imprime el individuo con
192
193             generacion += [individuo]#se concatena el individuo a la generación actual
194
195             individuo = []#se vuelve a establecer el individuo como una lista vacia pa
196
197             #se suma un individuo al numero de individuos generados y se quita uno de
198             numindividuo+=1
199             contadorsecundario-=1
200
201             #se regresa a su estado inicial las variables auxiliares usadas.
202             contadorinversiones3 = 0

```

```

203
204         #la generación que se acaba de crear se establece como la siguiente g
205         generacionanterior = generacion
206         generaciones += [generacion]
207
208         #se regresa a su estado inicial todas las variables auxiliares usadas
209         generacion = []
210         contadoruno = 0
211         contadordos = 1
212         numindividuo = 0
213         inversion = 0
214         rendimiento = 0
215         inversion2 = 0
216         rendimiento2 = 0
217         contadorinversiones1 = 0
218         contadorinversiones2 = 0
219         contadorinversiones3 = 0
220
221         #se aumenta el contador principal, indicando que se creo otra generac
222         contadorprincipal+=1

```

```

223
224 #Con estas lineas de codigo se imprime de forma bonita todas las generaciones creadas.
225 contgen = 1
226 print(f'Todas las generaciones:')
227 for gen in generaciones:
228     print(f'Generacion {contgen}:')
229     for ind in gen:
230         print(f'{ind[:2]} inversion: {ind[-2]} rendimiento: {ind[-1]}')
231     contgen+=1
232
233 #se busca la mejor opción en todas las generaciones que fueron creadas
234 for gen in generaciones:
235     for ind in gen:
236         if ind[-2] <= inversionmaxima and ind[-1] >= mejoropcion[-1]:
237             mejoropcion = ind
238
239 print(f'La mejor opción al final es: {mejoropcion[:2]}, con inversión de: {mejoropcion[-2]} y rendimiento de: {mejoropcion[-1]}') #
240
241 return generaciones#retorna todas las generaciones creadas.

```

Variables que ocupa para funcionar:

- numinstrumentos: variable que nos dice el número de instrumentos en el ejercicio.
- Instrumentos: lista que contiene la inversión y rendimiento de cada instrumento.
- generaciones: lista donde se van guardando todas las generaciones y que ya debe contener la primera generación que se crea aleatoriamente.
- numgeneraciones: variable donde viene el número de generaciones que se tienen que crear.
- inversionmaxima: variable donde viene la cantidad máxima que puede tener un individuo de inversión para ser seleccionable como mejor opción.

Objetivo de la función: Crear todas las generaciones después de la primera que solicita el ejercicio y darnos la mejor opción de inversión que se puede encontrar en todas las generaciones que se van creando.

Variables:

- contadorprincipal: contador que apoya al ciclo while principal.
- contadorsecundario: contador que apoya al ciclo while dentro del ciclo principal.
- generacionanterior: lista donde se guarda la ultima generación creada para usarla y mezclar sus individuos para crear los individuos de la siguiente generación.
- contadoruno: contador que hace referencia al primer individuo que se usa para mezclar.
- contadordos: contador que hace referencia al segundo individuo que se usa para mezclar.
- contadorinversiones1: contador que apoya para sumar la inversión y rendimiento del primer individuo creado de una mezcla.
- contadorinversiones2: contador que apoya para sumar la inversión y rendimiento del segundo individuo creado de una mezcla.
- generación: lista donde se guarda la generación que se crea cada recorrido del ciclo principal.
- individuo: lista donde se guarda el primer individuo creado de una mezcla.
- individuo2: lista donde se guarda el segundo individuo creado de una mezcla.

- inversión: variable donde se guarda la inversión total del primer individuo creado de una mezcla.
- rendimiento: variable donde se guarda el rendimiento total del primer individuo creado de una mezcla.
- inversion2: variable donde se guarda la inversión total del segundo individuo creado de una mezcla.
- rendimiento2: variable donde se guarda el rendimiento total del segundo individuo creado de una mezcla.
- numindividuo: contador que nos dice en que numero de individuo vamos en la generación que se está creando.
- contadorinversiones3: contador que apoya para sumar la inversión y rendimiento de un individuo que como no tuvo otro individuo para mezclarse se va a mutar uno de sus genes.
- mejorinversion: lista donde se va guardando el mejor individuo encontrado que es igual o menor a la inversión máxima que se permite y tiene mejor rendimiento.
- contadoropcion: contador que apoya en el ciclo que construye la lista "mejorinversion"

Explicación de la función:

Se tiene un ciclo while principal, en el cual cada recorrido del ciclo se crea una nueva generación de individuos, dentro se tiene un ciclo for que analiza todos los individuos en las generaciones que se han creado en cada corrido del while principal para encontrar la mejor opción de individuo, el cual tenga una inversión igual o menor a la inversión máxima y que tenga un mayor rendimiento. También se inicializa el contador secundario para saber cuantos individuos se deben de crear en cada generación.

En el ciclo while secundario dentro del ciclo principal tenemos un if, el if funciona de la siguiente manera:

Si el contador secundario es mayor a 1 quiere decir que quedan suficientes individuos para mezclar y crear más individuos. Se establece un limite para el punto de cruce y se crea el punto de cruce, a continuación se crea el individuo uno, después se suma su inversión y rendimiento y luego se crea el individuo 2 y se suma su inversión y rendimiento, se actualizan las variables contadoruno y contadordos y después estos individuos se añaden a la generación que se esta creando actualmente, a continuación solo se regresan las diferentes variables a sus estados iniciales para la siguiente corrida del ciclo secundario.

Si el contador secundario es igual o menor a 1 quiere decir que no quedan suficientes individuos para mezclar, entonces se va a mutar un gen en el individuo que queda. Se decide de forma aleatoria el gen que va a mutar del individuo, después si ese gen es igual a 0 se convierte en 1 y si el gen es igual a 1 se convierte a 0, se añade este individuo con gen mutado a la generación que se esta creando actualmente y se actualizan algunas variables y se regresa otras a su estado inicial para el siguiente recorrido del ciclo secundario.

El ciclo secundario se seguirá recorriendo hasta que la variable "contadorsecundario" nos indique que ya no quedan individuos por crear.

Al salir del ciclo secundario se establece la generación que se acaba de crear como la generación que se usará para el siguiente recorrido del ciclo principal y se añade la generación que se creó a la lista que contiene todas las generaciones, después solo se regresa todas las variables utilizadas a su estado inicial para el siguiente recorrido del ciclo principal.

Al salir del ciclo principal se imprimen todas las generaciones que fueron creadas y después se busca cual fue la mejor opción de individuo entre todas las generaciones de individuos creadas y se imprime, después se retorna la lista que contiene todas las generaciones.

Resultado de ejemplo de la función:

```
[[[0, 0, 1, 100, 50], [1, 0, 1, 250, 125] [0, 1, 1, 600, 300]], [[0, 1, 0, 500, 250], [1, 1, 1, 750, 325] [0, 0, 0, 0, 0]], [[0, 0, 1, 100, 50], [1, 0, 0, 150, 75] [0, 1, 1, 600, 300]]]
```

Este resultado podría ser retornado con esta función si en el ejercicio hubiera 3 instrumentos y se pidieran crear 3 generaciones, el instrumento 1 tuviera una inversión de 150 y un rendimiento de 75, el instrumento 2 una inversión de 500 y un rendimiento de 250, y el instrumento 3 una inversión de 100 y un rendimiento de 50.

La mejor opción se va buscando e imprimiendo dentro de la función, pero no es retornada.

Pruebas

Prueba 1:

```
Go Run Terminal Help algen.py - AlgoritmoGe

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

r/AppData/Local/Microsoft/WindowsApps/python3.9.exe "c:/Users/User/Desktop/Repositorio perso/Intelig
encia Artificial/AlgoritmoGenetico/algen.py"
¿Cuántos instrumentos de inversión va a ingresar?: 6
¿Cuál es el monto máximo que se puede invertir?: 30000
¿Cuántas generaciones va a generar?: 4
¿Cuánto se le tiene que invertir al instrumento 1?: 1500
¿Cuál es el rendimiento que saca de esa inversión?: 150
¿Cuánto se le tiene que invertir al instrumento 2?: 5000
¿Cuál es el rendimiento que saca de esa inversión?: 550
¿Cuánto se le tiene que invertir al instrumento 3?: 10000
¿Cuál es el rendimiento que saca de esa inversión?: 1300
¿Cuánto se le tiene que invertir al instrumento 4?: 8000
¿Cuál es el rendimiento que saca de esa inversión?: 800
¿Cuánto se le tiene que invertir al instrumento 5?: 6000
¿Cuál es el rendimiento que saca de esa inversión?: 570
¿Cuánto se le tiene que invertir al instrumento 6?: 5000
¿Cuál es el rendimiento que saca de esa inversión?: 500
Primera generación generada aleatoriamente: [[0, 1, 1, 1, 1, 1, 34000.0, 3720.0], [1, 1, 1, 1, 1, 0,
30500.0, 3370.0], [1, 0, 0, 1, 1, 0, 15500.0, 1520.0], [0, 1, 0, 1, 0, 1, 18000.0, 1850.0], [0, 0,
0, 1, 1, 0, 14000.0, 1370.0], [0, 0, 1, 0, 0, 0, 10000.0, 1300.0]]
La mejor opción por ahora es: [0, 1, 0, 1, 0, 1], con inversión de: 18000.0 y rendimiento de: 1850.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 1, 1, 1, 1, 1, 34000.0, 3720.0]
, [1, 1, 1, 1, 1, 0, 30500.0, 3370.0], [1, 0, 0, 1, 1, 0, 15500.0, 1520.0], [0, 1, 0, 1, 0, 1, 18000
.0, 1850.0], [0, 0, 0, 1, 1, 0, 14000.0, 1370.0], [0, 0, 1, 0, 0, 0, 10000.0, 1300.0]]
Padres: [0, 1, 1, 1, 1, 1] y [1, 1, 1, 1, 1, 0]
El punto de cruce es: 3
Parte 1: [0, 1, 1]
Parte 2: [1, 1, 0]
Hijo uno : [0, 1, 1, 1, 1, 0, 29000.0, 3220.0]
Parte 1: [1, 1, 1]
Parte 2: [1, 1, 1]
Hijo dos : [1, 1, 1, 1, 1, 1, 35500.0, 3870.0]
Padres: [1, 0, 0, 1, 1, 0] y [0, 1, 0, 1, 0, 1]
El punto de cruce es: 5
Parte 1: [1, 0, 0, 1, 1]
Parte 2: [1]
Hijo uno : [1, 0, 0, 1, 1, 1, 20500.0, 2020.0]
Parte 1: [0, 1, 0, 1, 0]
Parte 2: [0]
Hijo dos : [0, 1, 0, 1, 0, 0, 13000.0, 1350.0]
Padres: [0, 0, 0, 1, 1, 0] y [0, 0, 1, 0, 0, 0]
El punto de cruce es: 2
Parte 1: [0, 0]
Parte 2: [1, 0, 0, 0]
Hijo uno : [0, 0, 1, 0, 0, 0, 10000.0, 1300.0]
Parte 1: [0, 0]
Parte 2: [0, 1, 1, 0]
Hijo dos : [0, 0, 0, 1, 1, 0, 14000.0, 1370.0]
La mejor opción por ahora es: [0, 1, 1, 1, 1, 0], con inversión de: 29000.0 y rendimiento de: 3220.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 1, 1, 1, 1, 0, 29000.0, 3220.0]
, [1, 1, 1, 1, 1, 1, 35500.0, 3870.0], [1, 0, 0, 1, 1, 1, 20500.0, 2020.0], [0, 1, 0, 1, 0, 0, 13000
.0, 1350.0], [0, 0, 1, 0, 0, 0, 10000.0, 1300.0], [0, 0, 0, 1, 1, 0, 14000.0, 1370.0]]
Padres: [0, 1, 1, 1, 1, 0] y [1, 1, 1, 1, 1, 1]
El punto de cruce es: 5
Parte 1: [0, 1, 1, 1, 1]
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Parte 2: [1]
Hijo uno : [0, 1, 1, 1, 1, 1, 34000.0, 3720.0]
Parte 1: [1, 1, 1, 1, 1]
Parte 2: [0]
Hijo dos : [1, 1, 1, 1, 1, 0, 30500.0, 3370.0]
Padres: [1, 0, 0, 1, 1, 1] y [0, 1, 0, 1, 0, 0]
El punto de cruce es: 4
Parte 1: [1, 0, 0, 1]
Parte 2: [0, 0]
Hijo uno : [1, 0, 0, 1, 0, 0, 9500.0, 950.0]
Parte 1: [0, 1, 0, 1]
Parte 2: [1, 1]
Hijo dos : [0, 1, 0, 1, 1, 1, 24000.0, 2420.0]
Padres: [0, 0, 1, 0, 0, 0] y [0, 0, 0, 1, 1, 0]
El punto de cruce es: 2
Parte 1: [0, 0]
Parte 2: [0, 1, 1, 0]
Hijo uno : [0, 0, 0, 1, 1, 0, 14000.0, 1370.0]
Parte 1: [0, 0]
Parte 2: [1, 0, 0, 0]
Hijo dos : [0, 0, 1, 0, 0, 0, 10000.0, 1300.0]
La mejor opción por ahora es: [0, 1, 1, 1, 1, 0], con inversión de: 29000.0 y rendimiento de: 3220.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 1, 1, 1, 1, 1, 34000.0, 3720.0],
, [1, 1, 1, 1, 1, 0, 30500.0, 3370.0], [1, 0, 0, 1, 0, 0, 9500.0, 950.0], [0, 1, 0, 1, 1, 1, 24000.0,
, 2420.0], [0, 0, 0, 1, 1, 0, 14000.0, 1370.0], [0, 0, 1, 0, 0, 0, 10000.0, 1300.0]]
Padres: [0, 1, 1, 1, 1, 1] y [1, 1, 1, 1, 1, 0]
El punto de cruce es: 3
Parte 1: [0, 1, 1]
Parte 2: [1, 1, 0]
Hijo uno : [0, 1, 1, 1, 1, 0, 29000.0, 3220.0]
Parte 1: [1, 1, 1]
Parte 2: [1, 1, 1]
Hijo dos : [1, 1, 1, 1, 1, 1, 35500.0, 3870.0]
Padres: [1, 0, 0, 1, 0, 0] y [0, 1, 0, 1, 1, 1]
El punto de cruce es: 4
Parte 1: [1, 0, 0, 1]
Parte 2: [1, 1]
Hijo uno : [1, 0, 0, 1, 1, 1, 20500.0, 2020.0]
Parte 1: [0, 1, 0, 1]
Parte 2: [0, 0]
Hijo dos : [0, 1, 0, 1, 0, 0, 13000.0, 1350.0]
Padres: [0, 0, 0, 1, 1, 0] y [0, 0, 1, 0, 0, 0]
El punto de cruce es: 4
Parte 1: [0, 0, 0, 1]
Parte 2: [0, 0]
Hijo uno : [0, 0, 0, 1, 0, 0, 8000.0, 800.0]
Parte 1: [0, 0, 1, 0]
Parte 2: [1, 0]
Hijo dos : [0, 0, 1, 0, 1, 0, 16000.0, 1870.0]
Todas las generaciones:
Generacion 1:
[0, 1, 1, 1, 1, 1] inversion: 34000.0 rendimiento: 3720.0
[1, 1, 1, 1, 1, 0] inversion: 30500.0 rendimiento: 3370.0
[1, 0, 0, 1, 1, 0] inversion: 15500.0 rendimiento: 1520.0
[0, 1, 0, 1, 0, 1] inversion: 18000.0 rendimiento: 1850.0
```

```

[0, 0, 0, 1, 1, 0] inversion: 14000.0 rendimiento: 1370.0
[0, 0, 1, 0, 0, 0] inversion: 10000.0 rendimiento: 1300.0
Generacion 2:
[0, 1, 1, 1, 1, 0] inversion: 29000.0 rendimiento: 3220.0
[1, 1, 1, 1, 1, 1] inversion: 35500.0 rendimiento: 3870.0
[1, 0, 0, 1, 1, 1] inversion: 20500.0 rendimiento: 2020.0
[0, 1, 0, 1, 0, 0] inversion: 13000.0 rendimiento: 1350.0
[0, 0, 1, 0, 0, 0] inversion: 10000.0 rendimiento: 1300.0
[0, 0, 0, 1, 1, 0] inversion: 14000.0 rendimiento: 1370.0
Generacion 3:
[0, 1, 1, 1, 1, 1] inversion: 34000.0 rendimiento: 3720.0
[1, 1, 1, 1, 1, 0] inversion: 30500.0 rendimiento: 3370.0
[1, 0, 0, 1, 0, 0] inversion: 9500.0 rendimiento: 950.0
[0, 1, 0, 1, 1, 1] inversion: 24000.0 rendimiento: 2420.0
[0, 0, 0, 1, 1, 0] inversion: 14000.0 rendimiento: 1370.0
[0, 0, 1, 0, 0, 0] inversion: 10000.0 rendimiento: 1300.0
Generacion 4:
[0, 1, 1, 1, 1, 0] inversion: 29000.0 rendimiento: 3220.0
[1, 1, 1, 1, 1, 1] inversion: 35500.0 rendimiento: 3870.0
[1, 0, 0, 1, 1, 1] inversion: 20500.0 rendimiento: 2020.0
[0, 1, 0, 1, 0, 0] inversion: 13000.0 rendimiento: 1350.0
[0, 0, 0, 1, 0, 0] inversion: 8000.0 rendimiento: 800.0
[0, 0, 1, 0, 1, 0] inversion: 16000.0 rendimiento: 1870.0
La mejor opción al final es: [0, 1, 1, 1, 1, 0], con inversión de: 29000.0 y rendimiento de: 3220.0
PS C:\Users\User\Desktop\Repositorio perso\Inteligencia Artificial\AlgoritmoGenetico>

```

Prueba 2:

```

Go Run Terminal Help algen.py - AlgoritmoGene
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\User\Desktop\Repositorio perso\Inteligencia Artificial\AlgoritmoGenetico> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.9.exe "c:/Users/User/Desktop/Repositorio perso/Inteligencia Artificial/AlgoritmoGenetico/algen.py"
¿Cuántos instrumentos de inversión va a ingresar?: 4
¿Cuál es el monto máximo que se puede invertir?: 10000
¿Cuántas generaciones va a generar?: 4
¿Cuánto se le tiene que invertir al instrumento 1?: 5000
¿Cuál es el rendimiento que saca de esa inversión?: 1500
¿Cuánto se le tiene que invertir al instrumento 2?: 3000
¿Cuál es el rendimiento que saca de esa inversión?: 350
¿Cuánto se le tiene que invertir al instrumento 3?: 2000
¿Cuál es el rendimiento que saca de esa inversión?: 1560
¿Cuánto se le tiene que invertir al instrumento 4?: 1500
¿Cuál es el rendimiento que saca de esa inversión?: 750
Primera generación generada aleatoriamente: [[0, 0, 0, 1, 1500.0, 750.0], [1, 1, 1, 1, 11500.0, 4160.0], [0, 1, 1, 0, 5000.0, 1910.0], [1, 0, 0, 1, 6500.0, 2250.0]]
La mejor opción por ahora es: [1, 0, 0, 1], con inversión de: 6500.0 y rendimiento de: 2250.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 0, 0, 1, 1500.0, 750.0], [1, 1, 1, 1, 11500.0, 4160.0], [0, 1, 1, 0, 5000.0, 1910.0], [1, 0, 0, 1, 6500.0, 2250.0]]

```



```

Padres: [0, 0, 0, 1] y [1, 1, 1, 1]
El punto de cruce es: 1
Parte 1: [0]
Parte 2: [1, 1, 1]
Hijo uno : [0, 1, 1, 1, 6500.0, 2660.0]
Parte 1: [1]
Parte 2: [0, 0, 1]
Hijo dos : [1, 0, 0, 1, 6500.0, 2250.0]
Padres: [0, 1, 1, 0] y [1, 0, 0, 1]
El punto de cruce es: 1
Parte 1: [0]
Parte 2: [0, 0, 1]
Hijo uno : [0, 0, 0, 1, 1500.0, 750.0]
Parte 1: [1]
Parte 2: [1, 1, 0]
Hijo dos : [1, 1, 1, 0, 10000.0, 3410.0]
La mejor opción por ahora es: [1, 1, 1, 0], con inversión de: 10000.0 y rendimiento de: 3410.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 1, 1, 1, 6500.0, 2660.0], [1, 0, 0, 1, 6500.0, 2250.0], [0, 0, 0, 1, 1500.0, 750.0], [1, 1, 1, 0, 10000.0, 3410.0]]
Padres: [0, 1, 1, 1] y [1, 0, 0, 1]
El punto de cruce es: 1
Parte 1: [0]
Parte 2: [0, 0, 1]
Hijo uno : [0, 0, 0, 1, 1500.0, 750.0]
Parte 1: [1]
Parte 2: [1, 1, 1]
Hijo dos : [1, 1, 1, 1, 11500.0, 4160.0]
Padres: [0, 0, 0, 1] y [1, 1, 1, 0]
El punto de cruce es: 3
Parte 1: [0, 0, 0]
Parte 2: [0]
Hijo uno : [0, 0, 0, 0, 0, 0]
Parte 1: [1, 1, 1]
Parte 2: [1]
Hijo dos : [1, 1, 1, 1, 11500.0, 4160.0]
La mejor opción por ahora es: [1, 1, 1, 0], con inversión de: 10000.0 y rendimiento de: 3410.0

```

0 0 0

```

Generación de donde vienen los padres y o muta algún individuo: [[0, 0, 0, 1, 1500.0, 750.0], [1, 1, 1, 1, 11500.0, 4160.0], [0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 11500.0, 4160.0]]
Padres: [0, 0, 0, 1] y [1, 1, 1, 1]
El punto de cruce es: 3
Parte 1: [0, 0, 0]
Parte 2: [1]
Hijo uno : [0, 0, 0, 1, 1500.0, 750.0]
Parte 1: [1, 1, 1]
Parte 2: [1]
Hijo dos : [1, 1, 1, 1, 11500.0, 4160.0]
Padres: [0, 0, 0, 0] y [1, 1, 1, 1]
El punto de cruce es: 3
Parte 1: [0, 0, 0]
Parte 2: [1]
Hijo uno : [0, 0, 0, 1, 1500.0, 750.0]
Parte 1: [1, 1, 1]
Parte 2: [0]
Hijo dos : [1, 1, 1, 0, 10000.0, 3410.0]

```

```
Todas las generaciones:
Generacion 1:
[0, 0, 0, 1] inversion: 1500.0 rendimiento: 750.0
[1, 1, 1, 1] inversion: 11500.0 rendimiento: 4160.0
[0, 1, 1, 0] inversion: 5000.0 rendimiento: 1910.0
[1, 0, 0, 1] inversion: 6500.0 rendimiento: 2250.0
Generacion 2:
[0, 1, 1, 1] inversion: 6500.0 rendimiento: 2660.0
[1, 0, 0, 1] inversion: 6500.0 rendimiento: 2250.0
[0, 0, 0, 1] inversion: 1500.0 rendimiento: 750.0
[1, 1, 1, 0] inversion: 10000.0 rendimiento: 3410.0
Generacion 3:
[0, 0, 0, 1] inversion: 1500.0 rendimiento: 750.0
[1, 1, 1, 1] inversion: 11500.0 rendimiento: 4160.0
[0, 0, 0, 0] inversion: 0 rendimiento: 0
[1, 1, 1, 1] inversion: 11500.0 rendimiento: 4160.0
Generacion 4:
[0, 0, 0, 1] inversion: 1500.0 rendimiento: 750.0
[1, 1, 1, 1] inversion: 11500.0 rendimiento: 4160.0
[0, 0, 0, 1] inversion: 1500.0 rendimiento: 750.0
[1, 1, 1, 0] inversion: 10000.0 rendimiento: 3410.0
La mejor opción al final es: [1, 1, 1, 0], con inversión de: 10000.0 y rendimiento de: 3410.0
PS C:\Users\User\Desktop\Repositorio perso\Inteligencia Artificial\AlgoritmoGenetico>
```

0 0 0

Prueba 3:

```
PS C:\Users\User\Desktop\Repositorio perso\Inteligencia Artificial\AlgoritmoGenetico> & C:/Users/User/gen.py"
¿Cuántos instrumentos de inversión va a ingresar?: 3
¿Cual es el monto maximo que se puede invertir?: 6500
¿Cuántas generaciones va a generar?: 2
¿Cuanto se le tiene que invertir al instrumento 1?: 1500
¿Cual es le rendimiento que saca de esa inversión?: 850
¿Cuanto se le tiene que invertir al instrumento 2?: 3000
¿Cual es le rendimiento que saca de esa inversión?: 2600
¿Cuanto se le tiene que invertir al instrumento 3?: 1000
¿Cual es le rendimiento que saca de esa inversión?: 960
Primera generacion generada aleatoriamente: [[0, 1, 1, 4000.0, 3560.0], [0, 1, 0, 3000.0, 2600.0], [
La mejor opción por ahora es: [0, 1, 1], con inversión de: 4000.0 y rendimiento de: 3560.0
Generación de donde vienen los padres y o muta algún individuo: [[0, 1, 1, 4000.0, 3560.0], [0, 1, 0
Padres: [0, 1, 1] y [0, 1, 0]
El punto de cruce es: 2
Parte 1: [0, 1]
Parte 2: [0]
Hijo uno : [0, 1, 0, 3000.0, 2600.0]
Parte 1: [0, 1]
Parte 2: [1]
Hijo dos : [0, 1, 1, 4000.0, 3560.0]
Individuo que va a mutar: [0, 0, 1]
Individuo con gen mutado : [0, 0, 0, 0, 0]
Todas las generaciones:
Generacion 1:
[0, 1, 1] inversion: 4000.0 rendimiento: 3560.0
[0, 1, 0] inversion: 3000.0 rendimiento: 2600.0
[0, 0, 1] inversion: 1000.0 rendimiento: 960.0
Generacion 2:
[0, 1, 0] inversion: 3000.0 rendimiento: 2600.0
[0, 1, 1] inversion: 4000.0 rendimiento: 3560.0
[0, 0, 0] inversion: 0 rendimiento: 0
La mejor opción al final es: [0, 1, 1], con inversión de: 4000.0 y rendimiento de: 3560.0
PS C:\Users\User\Desktop\Repositorio perso\Inteligencia Artificial\AlgoritmoGenetico>
```

0 0 0

Ln 221, Col 82 Spaces: 4 UTF-8 CRLF Python