CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

METAHEURÍSTICAS I

7° "A"

ACTIVIDAD 2.01: ALGORITMO GENÉTICO

Profesor: Francisco Javier Luna Roas

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags.**, 1°** de septiembre de 2021

Actividad 2.01: Algoritmo Genético

Objetivo:

Mediante el desarrollo de esta práctica, se implementará el algoritmo genético simple para la solución del problema OneMax aplicado a cadenas de 20 elementos binarios.

Introducción:

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

Estos algoritmos llevan ese nombre debido a que están inspirados en la evolución biológica. “El principio de su operación se basa en la creencia de que, si un individuo es apto para adaptarse a su medio, entonces tiene una mayor probabilidad de procrearse y transmitir sus características genéticas a su descendencia; haciendo así que las poblaciones de individuos sean mejores con el paso del tiempo” (Torres, 2020).

Estos algoritmos hacen evolucionar una población de individuos, o conjunto de soluciones posibles del problema, “sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica tales como mutaciones y recombinaciones genéticas” (Luna, 2021); así como también a “una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados” (Talbi, 2009).

Un algoritmo genético funciona de la siguiente manera: “dado un problema específico de optimización a resolver, el algoritmo genético requiere de un conjunto inicial de soluciones potenciales a ese problema, codificadas de alguna manera y de una función de aptitud que permite evaluar cuantitativamente a cada candidata a solución” (Goldberg, 1989). Koza (1992) complementa este proceso al aclarar que estas candidatas se suelen generar aleatoriamente. En su obra se menciona que “pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG depure las opciones válidas hasta escoger la mejor”.

Cada una de las soluciones potenciales es evaluada por la función de aptitud, una ecuación matemática, que le da una calificación para saber qué tan “buena” es con respecto a las demás soluciones. Por supuesto, “la mayoría de estas soluciones no funcionarán en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras, es decir, pueden mostrar parte de la solución, aunque ésta sea débil e imperfecta, hacia la solución final del problema. Por lo tanto un AG es un método de búsqueda dirigida basada en probabilidades” (Eiben, 2015).

Para hacer uso de un algoritmo genético en la solución de un problema se debe modelar el problema, estableciendo cómo se va a representar una solución, como se va a asignar el valor de la adaptabilidad (calidad de la solución) y cómo se pueden transformar soluciones (obtener soluciones nuevas a partir de ciertas soluciones).

Pregunta de Investigación:

¿De qué manera pueden adaptarse las ideas inspiradas en la biología para el desarrollo de un algoritmo genético para que una computadora desarrolle el método de forma más eficiente?

Predicción:

Creo que la complejidad de este problema requerirá herramientas de modelado y formas de representación de la solución que nos pueda servir para poder llevar a cabo el algoritmo genético y la solución del OneMax.

Materiales:

Una computadora con compilador de C.	Dos hojas de papel
Una calculadora.	Lápiz o plumas

Método (Variables):

Dependiente: El resultado que arroje el programa según el algoritmo genético encontrado.
Independiente: El algoritmo que uno como alumno se focalizará en elaborar.
Controlada: El procedimiento del algoritmo genético que se usará para evaluar.

Seguridad:

Realmente no se trabajó en campo, por lo que no se corren riesgos al elaborar el experimento.

Procedimiento:

1.- Haciendo uso del lenguaje de programación C, se realizó un programa que implemente la siguiente secuencia de eventos.

1) Sea $t = 0$ el contador de generaciones
2) Se inicializó $P(t)$
3) Se evaluó $P(t)$
4) Mientras no se cumpla un criterio de paro, se hizo:
    a. Para $i = 1, \dots, \frac{N}{2}$ se hizo
        i. Se seleccionaron dos padres de $P(t)$
        ii. Se aplicó el cruzamiento a los dos padres con probabilidad p_c.
        iii. Se mutó la descendencia con probabilidad p_m.
        iv. Se introdujeron los dos nuevos individuos en $P(t+1)$
    b. Fin

2.- El programa exhibe las siguientes características:

- La población inicial se genera aleatoriamente.
- El objetivo es maximizar la sumatoria de los elementos binarios que constituyen a cada individuo.
- Un individuo se constituye de una cadena de 20 dígitos binarios. La población contiene 50 individuos.
- El criterio de paro opcionalmente es el número de generaciones. Este valor es un parámetro que introduce el usuario, pero también se le da la opción de continuarlo indefinidamente.
- Las probabilidades de cruzamiento y mutación son parámetros que son introducidos por el usuario. Estos valores se leen como entero entre 0 y 100.
- Se implementó la selección por el método de la ruleta.

- El cruzamiento se realiza basado en un punto de cruza.
- La mutación se realiza en un solo bit de las cadenas.
- Se implementó el elitismo del mejor individuo de cada generación.

3.- El programa se probó con algunos grafos realizados previamente bajo este algoritmo. Los resultados se procesaron en el presente documento a continuación.

Obtención y Procesamiento de Datos:

Primeramente, se tomó un diseño de algoritmo para tener en cuenta todas las estructuras que se van a usar. Este diseño de algoritmo es el siguiente:

Como variables del programa denotamos las siguientes:

- `autom`: Modo de acción del programa. Si `autom = 0` entonces el procedimiento será manual y el paro lo decidirá el usuario. Si `autom = 1` entonces se realizará de golpe el algoritmo con una cantidad de generaciones especificada mostrando el resultado final únicamente.
- `qelitism`: Cantidad de cromosomas que se heredan automáticamente.
- `ig`: Iterador de generaciones.
- `pc`: Probabilidad de cruzamiento.
- `pm`: Probabilidad de mutación.
- `qe`: Cantidad de elementos (genes) que tendrá cada individuo.
- `qi`: Cantidad de individuos (cromosomas) que tendrá la muestra.
- `qg`: Cantidad de generaciones. Útil únicamente cuando `autom = 1`.

Con las variables anteriores se crean las siguientes estructuras:

- `sample`: Una matriz que tiene la población de muestra en la generación ig de tamaño qi filas y qe columnas:

$$\begin{bmatrix} i_1e_1 & i_1e_2 & i_1e_3 & \dots & i_1e_{qe} \\ i_2e_1 & i_2e_2 & i_2e_3 & \dots & i_2e_{qe} \\ i_3e_1 & i_3e_2 & i_3e_3 & \dots & i_3e_{qe} \\ \dots & \dots & \dots & \dots & \dots \\ i_{qi}e_1 & i_{qi}e_2 & i_{qi}e_3 & \dots & i_{qi}e_{qe} \end{bmatrix}$$

Donde i_1e_1 significa ser el dato del individuo 1 correspondiente a su elemento 1. Es decir, el gen 1 del cromosoma 1. i_ne_m significa ser el dato del individuo n correspondiente a su elemento m. Es decir, el gen m del cromosoma n.

- `results`: Una matriz con los resultados de las evaluaciones de cada cromosoma según qué tan convenientes son, con una columna para estos resultados y otra columna para los resultados de forma acumulada

$$\begin{bmatrix} ri_1 & ri_1 \\ ri_2 & ri_1 + ri_2 \\ ri_3 & ri_1 + ri_2 + ri_3 \\ \dots & \dots \\ ri_{qi} & ri_1 + ri_2 + ri_3 + \cdots + ri_{qi} \end{bmatrix} = \begin{bmatrix} ri_1 & rai_1 \\ ri_2 & rai_2 \\ ri_3 & rai_3 \\ \dots & \dots \\ ri_{qi} & rai_{qi} \end{bmatrix}$$

  Vemos que denotamos a la suma acumulada de los valores hasta n de modo que escribimos rai_n en vez de $ri_1 + ri_2 + ri_3 + \cdots + ri_n$.

- `resultsAux`: Un vector asociado a `results`.

$$\begin{bmatrix} ra_1 \\ ra_2 \\ ra_3 \\ \dots \\ ra_{qi} \end{bmatrix}$$

- `newSample`: Una matriz igual a `sample`.

El proceso del algoritmo es el siguiente

1) Se genera aleatoriamente cada valor del arreglo `sample`. Donde: $i_ne_m \in \{0,1\}$ $\forall n = 1,2,3, \dots q_i$ y $\forall m = 1,2,3, \dots q_e$.
2) Calculamos `results`. `results` se calcula a partir de `sample` de la siguiente manera:

   La primera columna de `results` se calcula de la siguiente manera:

$$ri_n = i_ne_1 + i_ne_2 + i_ne_3 + \cdots + i_ne_{qe}$$

   La segunda columna de `results` se calcula como anteriormente se ve con relación a la primera columna de `results`.
3) Se llenará `resultsAux` de los valores correspondientes a la primera columna de `results` de modo que:

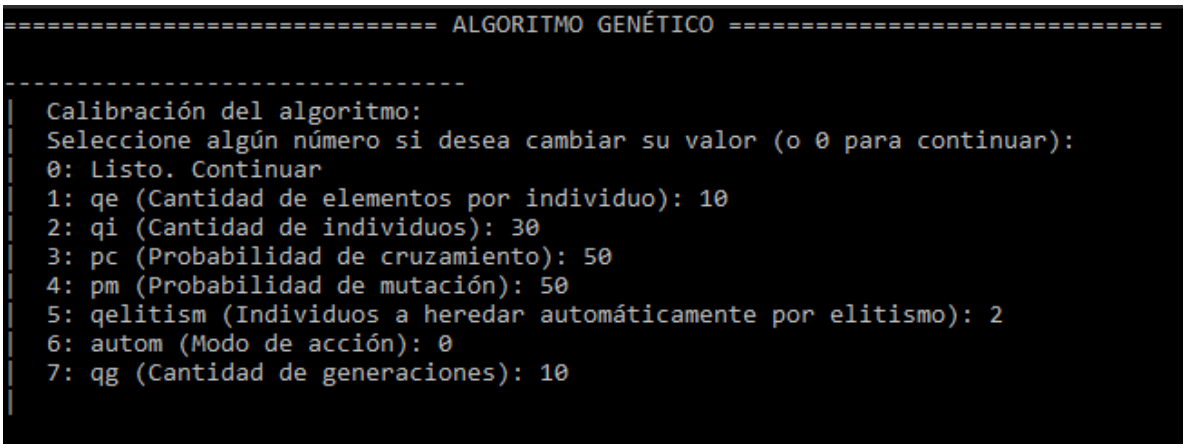
$$\begin{bmatrix} ra_1 \\ ra_2 \\ ra_3 \\ \dots \\ ra_{qi} \end{bmatrix} = \begin{bmatrix} ri_1 \\ ri_2 \\ ri_3 \\ \dots \\ ri_{qi} \end{bmatrix}$$

Ahora se aplicará a `resultsAux` el ordenamiento de la burbuja de manera descendente.

4) De `resultsAux` se tomarán los elementos desde ra_1 hasta $ra_{qelitism}$ y se buscarán estos mismos elementos en `results`. Los cromosomas pertenecientes a estos elementos de `results` se escribirán en `newSample`.
5) Se seleccionan dos cromosomas aleatoriamente usando la ruleta con base en la segunda columna de `results`.
6) Con los dos cromosomas seleccionados, se aplicará el cruzamiento si se respeta la probabilidad de cruzamiento y los dos hijos se añadirán a `newSample`.
7) Se repetirán los pasos 5 y 6 desde que un contador $k = qelitism + 1$ hasta $k = qi$ con paso $k = k + 2$ (porque en cada paso se agregan dos cromosomas nuevos).
8) Ahora se sobrescribirá `sample` con los valores de `newSample` y `newSample` se volverá una matriz nula.
9) Si `autom = 1`, se repetirá del paso 2 al paso 8 mientras `ig` sea menor a `qg`. Si `autom = 0`, se repetirá del paso 2 al paso 8 hasta que el usuario pare.

Este mismo algoritmo se introdujo en el código en C (véase anexo 1)

El mismo programa permite al usuario modificar muchas variables y personalizar incluso la forma en la que se desea ver el procedimiento. Por ejemplo, se puede establecer el modo de acción a cero y el desglose es muy preciso. El menú para calibrar el algoritmo es el siguiente:



Todas las variables anteriormente presentadas son presentadas al usuario posibles para su modificación, así, se pueden crear poblaciones de mayor tamaño, de mayor cantidad de genes, de explorar con las probabilidades de cruzamiento, mutación y de selección por elitismo. Igualmente, se añade la personalización de la cantidad de generaciones y para cuando este proceso se quiera experimentar con una descendencia alta, el modo automático ayuda a resumir los resultados finales de todo el procedimiento.

Puede establecerse el modo de acción en uno para que los resultados sean más resumidos con lo que se pide estrictamente en el código, claro, con la condición de paro de las generaciones, pues en el modo cero, el código continúa indefinidamente hasta que el usuario decida parar.

```
Los 2 mejores cromosomas de la generación 6 son:
1  1  1  1  1  1  1  1  1  1      10
1  1  1  1  0  1  1  1  1  1      9


Los 2 mejores cromosomas de la generación 7 son:
1  1  1  1  1  1  0  1  1  1      9
1  1  1  1  1  1  1  0  1  1      9


Los 2 mejores cromosomas de la generación 8 son:
1  1  1  1  1  1  1  0  1  1      9
1  1  1  1  1  1  1  0  1  1      9


Los 2 mejores cromosomas de la generación 9 son:
1  1  1  1  1  1  1  1  1  1      10
1  1  1  1  1  1  1  0  1  1      9


Los 2 mejores cromosomas de la generación 10 son:
1  1  1  1  1  1  1  1  1  1      10
1  1  1  1  0  1  1  1  1  1      9


La matriz final de muestras en la generación 11 es:
1  1  1  1  1  1  1  1  1  1
1  1  1  1  0  1  1  1  1  1
1  1  1  0  1  1  1  1  0  1
1  1  1  0  1  1  1  1  0  1
1  1  1  1  1  1  1  1  1  1
1  0  1  1  1  0  0  0  0  1
1  1  1  1  0  0  1  0  1  1
1  0  1  1  0  1  1  0  0  1
1  1  1  1  0  1  1  1  1  1
1  1  1  1  1  1  0  0  0  1
1  1  1  1  0  1  1  1  1  1
1  0  1  1  1  1  0  1  1  1
1  1  0  1  0  1  0  0  1  0
1  1  1  1  1  1  1  0  0  1
1  1  1  1  1  1  0  0  1  1
1  0  1  1  1  1  0  1  1  1
1  1  1  1  0  1  1  0  0  1
1  1  1  1  1  1  1  0  1  1
1  1  1  1  1  1  0  0  1  0
1  1  0  1  1  1  1  1  0  1
0  1  1  1  0  0  1  0  1  1
1  0  1  1  0  0  1  1  0  1
0  1  1  1  1  0  1  1  1  1
1  0  1  0  1  1  0  1  0  1
1  0  0  1  1  0  1  0  0  1
1  1  1  0  1  1  0  1  0  1
1  0  1  1  1  1  0  1  1  1
1  0  1  1  1  1  0  1  0  1
1  1  1  1  1  1  1  0  1  1
1  1  1  1  1  0  1  1  0  1
```

El programa, si se lleva en su modo manual, mostrará cada paso realizado. En principio muestra la generación cero generada aleatoriamente, posteriormente las evalúa con respecto a la cantidad de genes de valor 1 (la función F) así como también se muestra la función acumulada de valores 1 (la función Fa), pues el siguiente proceso será ordenarlos de mayor cantidad de valores de 1 a menor. Esto con el objetivo de obtener la selección por elitismo el cual es mostrado a continuación.

La nueva generación comienza a construirse y el cruzamiento y la mutación se llevan a cabo, mostrando la nueva generación construida (véase anexo 2).

Conclusiones:

Es muy importante abstraer el concepto matemático y lógico del algoritmo, pues siento que esto es lo importante de este algoritmo, ya que fue el elemento principal para modelar el algoritmo.

Podemos concluir también que es muy útil inspirar los algoritmos de alguna técnica natural útil, transformando con sus pertinentes cambios para desarrollar lo que se busca. Este algoritmo busca ser mejor usando los mismos bits que sólo se generan al comienzo por lo que terminamos el desarrollo de la práctica con la utilidad y que puede realizar la convergencia correctamente y llegar a una buena solución.

Referencias:

Eiben, A. (2015). *Introduction to Evolutionary Computing*. Bristol: Springer.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston: Addison-Wesley.

Koza, J. (1992). *Genetic Programming On the Programming of Computers by Means of Natural Selection*. San Francisco: University of Stanford.

Luna, F. (2021). *Apuntes: Metaheurísticas I.* 7° ICI. México: Universidad Autónoma de Aguascalientes.

Purcell, E. (2007). *Cálculo.* Londres: Pearson Education.

Talbi, E. (2009). *Metaheuristics from design to implementation.* New Jersey: John Wiley & Sons Publication.

Torres, A. (2020). *Apuntes: Optimización Inteligente.* 5° ICI. México: Universidad Autónoma de Aguascalientes.

Anexos:

Anexo 1: Código del programa en lenguaje C:

```
/*
    Universidad Autónoma de Aguascalientes

           Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
              Metaheurísticas I

                    7° "A"

   Actividad 2.01: Algoritmo Genético Simple

     Profesor: Francisco Javier Luna Rosas

    Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 1° de septiembre del 2021
*/
//Cargamos las librerías
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#include <math.h>

void setValues(int autom, int pc, int pm, int qe, int qelitism, int qi, int qg, int
*automP, int *pcP, int *pmP, int *qeP, int *qelitismP, int *qiP, int *qgP);
void startTest(int qe, int qi, int sample[qi][qe]);
void nullNewSample(int qe, int qi, int newSample[qi][qe]);
void evaluate(int qe, int qi, int sample[qi][qe], int results[qi][2]);
void BubbleSort(int qe, int qi, int sample[qi][qe], int results[qi][2]);
void elitism(int qe, int qi, int qelitism, int ic, int sample[qi][qe], int
newSample[qi][qe]);
void crossing(int autom, int qe, int qi, int ic, int pc, int sample[qi][qe], int
results[qi][2], int newSample[qi][qe]);
void mutation(int autom, int qe, int qi, int pm, int newSample[qi][qe]);
void backToSample(int qe, int qi, int sample[qi][qe], int newSample[qi][qe]);

main()
{
       setlocale(LC_ALL,"");
       srand(time(NULL));

       //1. Declaramos las variables que usaremos
       /*
              autom: Modo de acción, donde 0 será Manual y 1 Automático, que:
                        Si autom = 0 ejecutará todo el proceso rápidamente lo hará qg
veces
                        Si autom = 1 ejecutará el proceso de uno en uno y el usuario en
cada generación decide si generar otra generación o parar
              ic: Iterador de cromosomas
              ig: Iterador de generaciones
              pc: Probabilidad de cruzamiento (0 < pc < 100)
              pm: Probabilidad de mutación (0 < pm < 100)
```

```
		qe: Cantidad de elementos
		qelitism: Cantidad de cromosomas que se herdan automáticamente por
elitismo
		qi: Cantidad de individuos
		qg: Cantidad de generaciones
	*/
	int autom, ic, ig, pc, pm, qe, qelitism, qi, qg, repeat, repeat1;

	printf("============================== ALGORITMO GENÉTICO
==============================\n");

	do
	{
		autom = 0;
		ic = 0;
		ig = 0;
		pc = 80;
		pm = 5;
		qe = 20;
		qelitism = 2;
		qi = 50;
		qg = 10;
		repeat = 0;

		//2. Calibramos el programa

	setValues(autom,pc,pm,qe,qelitism,qi,qg,&autom,&pc,&pm,&qe,&qelitism,&qi,&qg);

		//3. Declaramos más variables
		/*
			sample: La matriz de muestra que guardará los bits
			newSample: La matriz de muestra que guardará los nuevos bits
			results: Estructura para guardar la cantidad total de bits
		*/
		int sample[qi][qe], results[qi][2], newSample[qi][qe];

		//4. Generamos aleatoriamente la población inicial
		startTest(qe,qi,sample);

		printf("\n");
		printf("\n");
		printf("\n");
		int i,j;
		if(autom == 0)
		{
			printf("La matriz de muestras en la generación %d:\n",ig);
			for(i = 0; i < qi; i++)
			{
				for(j = 0; j < qe; j++)
				{
					printf("%d  ",sample[i][j]);
				}
				printf("\n");
			}
			getchar();
			getchar();
		}
```

```
		do
		{
			ic = 0;

			nullNewSample(qe,qi,newSample);

			//5. Evaluamos a todas las muestras junto con sus evaluaciones
acumuladas
			evaluate(qe,qi,sample,results);

			if(autom == 0)
			{
				printf("\n");
				printf("\n");
				printf("La matriz de muestras evaluada en F y Fa de la
generación %d:\n",ig);
				for(i = 0; i < qi; i++)
				{
					for(j = 0; j < qe; j++)
					{
						printf("%d  ",sample[i][j]);
					}
					printf("    %d   %d\n",results[i][0],results[i][1]);
				}
				getchar();
			}

			//6. Ordenamos
			BubbleSort(qe,qi,sample,results);
			evaluate(qe,qi,sample,results);

			if(autom == 0)
			{
				printf("\n");
				printf("\n");
				printf("La matriz de muestras ordenada y evaluada en F y Fa
de la generación %d:\n",ig);
				for(i = 0; i < qi; i++)
				{
					for(j = 0; j < qe; j++)
					{
						printf("%d  ",sample[i][j]);
					}
					printf("    %d   %d\n",results[i][0],results[i][1]);
				}
				getchar();
			}

			//7. Buscamos los qelitism mejores
			elitism(qe,qi,qelitism,ic,sample,newSample);

			ic = qelitism;

			if(autom == 0)
```

```
                        {
                                printf("\n");
                                printf("\n");
                                printf("Se aplica elitismo a qelitism = %d
cromosomas:\n",qelitism);
                                for(i = 0; i < qelitism; i++)
                                {
                                        for(j = 0; j < qe; j++)
                                        {
                                                printf("%d  ",sample[i][j]);
                                        }
                                        printf("    %d\n",results[i][0]);
                                }
                                getchar();

                                printf("\n");
                                printf("\n");
                                printf("Comenzamos a construir la nueva generación:\n");
                                for(i = 0; i < qi; i++)
                                {
                                        for(j = 0; j < qe; j++)
                                        {
                                                printf("%d  ",newSample[i][j]);
                                        }
                                        printf("\n");
                                }
                                getchar();
                        }

                        if (autom == 1)
                        {
                                printf("\n");
                                printf("\n");
                                printf("Los %d mejores cromosomas de la generación %d
son:\n",qelitism,ig);
                                for(i = 0; i < qelitism; i++)
                                {
                                        for(j = 0; j < qe; j++)
                                        {
                                                printf("%d  ",sample[i][j]);
                                        }
                                        printf("    %d\n",results[i][0]);
                                }
                        }

                        //8. Aplicamos cruzamiento

                        crossing(autom,qe,qi,ic,pc,sample,results,newSample);

                        if(autom == 0)
                        {
                                printf("\n");
                                printf("\n");
                                printf("La nueva generación ha pasado la etapa de
cruzamiento exitosamente:\n");
                                for(i = 0; i < qi; i++)
                                {
```

```
                                for(j = 0; j < qe; j++)
                                {
                                        printf("%d  ",newSample[i][j]);
                                }
                                printf("\n");
                        }
                        getchar();
                }

                //9. Aplicamos mutación
                mutation(autom,qe,qi,pm,newSample);

                ig++;

                if(autom == 0)
                {
                        printf("\n");
                        printf("\n");
                        printf("La nueva generación ha pasado la etapa de mutación
exitosamente y todos los operadores han sido aplicados\n");
                        printf("De modo que la matriz de muestras en la generación
%d es:\n",ig);
                        for(i = 0; i < qi; i++)
                        {
                                for(j = 0; j < qe; j++)
                                {
                                        printf("%d  ",newSample[i][j]);
                                }
                                printf("\n");
                        }
                        getchar();
                        printf("¿Desea construir una nueva generación?\n");
                        printf("0. No\n");
                        printf("1. Sí\n");
                        scanf("%d",&repeat1);
                        if(repeat1 == 1)
                        {
                                qg++;
                                getchar();
                        }
                        else
                        {
                                ig = qg + 1;
                        }
                }

                if(autom == 1 && ig == qg + 1)
                {
                        printf("\n");
                        printf("\n");
                        printf("La matriz final de muestras en la generación %d
es:\n",ig - 1);
                        for(i = 0; i < qi; i++)
                        {
                                for(j = 0; j < qe; j++)
                                {
                                        printf("%d  ",newSample[i][j]);
```

```
                                }
                                printf("\n");
                        }
                        getchar();
                }

                //10. Transformamos newSample en sample
                backToSample(qe,qi,sample,newSample);
        }
        while(ig <= qg);


        printf("\n");
        printf("¿Desea repetir el código?\n");
        printf("0. No\n");
        printf("1. Sí\n");
        scanf("%d",&repeat);
    }
    while(repeat == 1);
    getchar();
}

void setValues(int autom, int pc, int pm, int qe, int qelitism, int qi, int qg, int
*automP, int *pcP, int *pmP, int *qeP, int *qelitismP, int *qiP, int *qgP)
{
    int aux, done = 0;
    *automP = autom;
    *pcP = pc;
    *pmP = pm;
    *qeP = qe;
    *qelitismP = qelitism;
    *qiP = qi;
    *qgP = qg;

    do
    {
        printf("\n");
        printf("--------------------------------\n");
        printf("|  Calibración del algoritmo:\n");
        printf("|  Seleccione algún número si desea cambiar su valor (o 0 para
continuar):\n");
        printf("|  0: Listo. Continuar\n");
        printf("|  1: qe (Cantidad de elementos por individuo): %d\n",qe);
        printf("|  2: qi (Cantidad de individuos): %d\n",qi);
        printf("|  3: pc (Probabilidad de cruzamiento): %d\n",pc);
        printf("|  4: pm (Probabilidad de mutación): %d\n",pm);
        printf("|  5: qelitism (Individuos a heredar automáticamente por
elitismo): %d\n",qelitism);
        printf("|  6: autom (Modo de acción): %d\n",autom);
        printf("|  7: qg (Cantidad de generaciones): %d\n",qg);
        printf("|  ");
        scanf("%d",&aux);

        switch (aux)
        {
            case 0:
                {
```

```
                                        //Se continúa con el programa

                                        done = 1;

                                        break;
                                }
                        case 1:
                                {
                                        //Se modifica qe

                                        printf("|     Inserte un nuevo valor para qe (Antiguo
valor para qe: qe = %d)\n",qe);
                                        printf("|  ");
                                        scanf("%d",&qe);
                                        *qeP = qe;

                                        break;
                                }
                        case 2:
                                {
                                        //Se modifica qi

                                        printf("|     Inserte un nuevo valor para qi (Antiguo
valor para qi: qi = %d)\n",qi);
                                        printf("|  ");
                                        scanf("%d",&qi);
                                        *qiP = qi;

                                        break;
                                }
                        case 3:
                                {
                                        //Se modifica pc

                                        printf("|     Inserte un nuevo valor para pc (Antiguo
valor para pc: pc = %d)\n",pc);
                                        printf("|  ");
                                        scanf("%d",&pc);
                                        *pcP = pc;

                                        break;
                                }
                        case 4:
                                {
                                        //Se modifica pm

                                        printf("|     Inserte un nuevo valor para pm (Antiguo
valor para pm: pm = %d)\n",pm);
                                        printf("|  ");
                                        scanf("%d",&pm);
                                        *pmP = pm;

                                        break;
                                }
                        case 5:
                                {
                                        //Se modifica qelitism
```

```
                                        printf("|     Inserte un nuevo valor para qelitism
(Antiguo valor para qelitism: qelitism = %d)\n",qelitism);
                                        printf("|  ");
                                        scanf("%d",&qelitism);
                                        *qelitismP = qelitism;

                                        break;
                                }
                        case 6:
                                {
                                        //Se modifica autom

                                        printf("|     Inserte un nuevo valor para autom
(Antiguo valor para autom: autom = %d)\n",autom);
                                        printf("|  ");
                                        scanf("%d",&autom);
                                        *automP = autom;

                                        break;
                                }
                        case 7:
                                {
                                        //Se modifica qg

                                        printf("|     Inserte un nuevo valor para qg (Antiguo
valor para qg: qg = %d)\n",qg);
                                        printf("|  ");
                                        scanf("%d",&qg);
                                        *qgP = qg;

                                        break;
                                }
                        default:
                                {
                                        //Valor no válido

                                        printf("|     Ha insertado un número inválido\n");

                                        break;
                                }
                }
        }
        while(done == 0);

        return;
}

void startTest(int qe, int qi, int sample[qi][qe])
{
        int aux = 0,i,j;

        for(i = 0; i < qi; i++)
        {
                for(j = 0; j < qe; j++)
                {
                        aux = rand() % 2;
```

```
                        sample[i][j] = aux;
                }
        }

        return;
}

void nullNewSample(int qe, int qi, int newSample[qi][qe])
{
        int i,j;
        for(i = 0; i < qi; i++)
        {
                for(j = 0; j < qe; j++)
                {
                        newSample[i][j] = 0;
                }
        }

        return;
}

void evaluate(int qe, int qi, int sample[qi][qe], int results[qi][2])
{
        int i,j;

        for(i = 0; i < qi; i++)
        {
                results[i][0] = 0;
                results[i][1] = 0;
                for(j = 0; j < qe; j++)
                {
                        results[i][0] = results[i][0] + sample[i][j];
                }
                results[i][1] = results[i][1] + results[i][0];
                if(i > 0)
                {
                        results[i][1] = results[i][1] + results[i - 1][1];
                }
        }
        return;
}

void BubbleSort(int qe, int qi, int sample[qi][qe], int results[qi][2])
{
        int a,aux[qe],i,j;

        for(i = 0; i < qi - 1; i++)
        {
                if(results[i][0] < results[i + 1][0])
                {
                        a = results[i][0];
                        results[i][0] = results[i + 1][0];
                        results[i + 1][0] = a;

                        for(j = 0; j < qe; j++)
                        {
                                aux[j] = sample[i][j];
```

```
                        sample[i][j] = sample[i + 1][j];
                        sample[i + 1][j] = aux[j];
                }

                i = -1;
        }
    }

    return;
}

void elitism(int qe, int qi, int qelitism, int ic, int sample[qi][qe], int
newSample[qi][qe])
{
    int i;
    while(ic < qelitism)
    {
        for(i = 0; i < qe; i++)
        {
            newSample[ic][i] = sample[ic][i];
        }
        ic++;
    }

    return;
}

void crossing(int autom, int qe, int qi, int ic, int pc, int sample[qi][qe], int
results[qi][2], int newSample[qi][qe])
{
    int c1[qe],c2[qe],i,j,r,random;

    if(autom == 0)
    {
        printf("\n");
        printf("\n");
        printf("Se aplicará cruzamiento con una probabilidad de pc = %d%%:\n",pc);
        getchar();
        printf("\n");
        printf("Debido al elitismo, ya existen %d cromosomas en la nueva
generación:\n",ic);
        printf("Para denotar mejor el cruzamiento, marcaremos a los individuos
\n");
        printf("como: () del primer cromosoma y como [] del segundo cromosoma\n");
    }

    do
    {
        random = rand() % 100;

        if(random < pc)
        {
            if(autom == 0)
            {
                printf("\n");
                printf("\n");
```

```
                        printf("  El cruzamiento se ha activado (Probabilidad: %d <
%d)\n",random,pc);
                }

                //Seleccionamos por ruleta
                r = rand() % (results[qi - 1][1]);

                if(autom == 0)
                {
                        printf("\n");
                        printf("    Aplicamos selección por ruleta para el primer
padre obteniendo %d\n",r);
                        printf("    Por lo tanto, el cromosoma a escoger será ");
                }

                for(i = 0; i < qi; i++)
                {
                        if(r < results[i][1])
                        {
                                for(j = 0; j < qe; j++)
                                {
                                        c1[j] = sample[i][j];
                                        if(autom == 0)
                                        {
                                                printf("%d ",c1[j]);
                                        }
                                }
                                if(autom == 0)
                                {
                                        printf("\n");
                                }
                                break;
                        }
                }

                r = rand() % (results[qi - 1][1]);

                if(autom == 0)
                {
                        printf("\n");
                        printf("    Aplicamos selección por ruleta para el segundo
padre obteniendo %d\n",r);
                        printf("    Por lo tanto, el cromosoma a escoger será ");
                }

                for(i = 0; i < qi; i++)
                {
                        if(r < results[i][1])
                        {
                                for(j = 0; j < qe; j++)
                                {
                                        c2[j] = sample[i][j];
                                        if(autom == 0)
                                        {
                                                printf("%d ",c2[j]);
                                        }
                                }
```

```
                                        if(autom == 0)
                                        {
                                                printf("\n");
                                        }
                                        break;
                                }
                        }

                        //Seleccionamos una posición aleatoria para hacer el corte
                        random = 1 + (rand() % (qe - 1));

                        if(autom == 0)
                        {
                                printf("\n");
                                printf("    Se ha seleccionado aleatoriamente un punto de
corte entre el gen %d y %d\n",random,random + 1);
                                printf("    Serán los cromosomas %d y %d de la nueva
generación\n",ic + 1, ic + 2);
                        }

                        for(i = 0; i < qe; i++)
                        {
                                if(i < random)
                                {
                                        newSample[ic][i] = c1[i];
                                        newSample[ic + 1][i] = c2[i];

                                        if(autom == 0)
                                        {
                                                printf("    (%d)
[%d]\n",newSample[ic][i],newSample[ic+1][i]);
                                        }
                                }
                                else
                                {
                                        newSample[ic][i] = c2[i];
                                        newSample[ic + 1][i] = c1[i];

                                        if(autom == 0)
                                        {
                                                printf("    [%d]
(%d)\n",newSample[ic][i],newSample[ic+1][i]);
                                        }
                                }
                        }
                }
                else
                {
                        if(autom == 0)
                        {
                                printf("\n");
                                printf("\n");
                                printf("  El cruzamiento no se ha activado (Probabilidad: !
%d < %d  !)\n",random,pc);
                        }

                        //Seleccionamos por ruleta
```

```
                        r = rand() % (results[qi - 1][1]);

                        if(autom == 0)
                        {
                                printf("\n");
                                printf("    Aplicamos selección por ruleta para el primer
padre obteniendo %d\n",r);
                                printf("    Por lo tanto, el cromosoma a escoger será ");
                        }

                        for(i = 0; i < qi; i++)
                        {
                                if(r < results[i][1])
                                {
                                        for(j = 0; j < qe; j++)
                                        {
                                                c1[j] = sample[i][j];
                                                if(autom == 0)
                                                {
                                                        printf("%d ",c1[j]);
                                                }
                                        }
                                        if(autom == 0)
                                        {
                                                printf("\n");
                                        }
                                        break;
                                }
                        }

                        r = rand() % (results[qi - 1][1]);

                        if(autom == 0)
                        {
                                printf("\n");
                                printf("    Aplicamos selección por ruleta para el segundo
padre obteniendo %d\n",r);
                                printf("    Por lo tanto, el cromosoma a escoger será ");
                        }

                        for(i = 0; i < qi; i++)
                        {
                                if(r < results[i][1])
                                {
                                        for(j = 0; j < qe; j++)
                                        {
                                                c2[j] = sample[i][j];
                                                if(autom == 0)
                                                {
                                                        printf("%d ",c2[j]);
                                                }
                                        }
                                        if(autom == 0)
                                        {
                                                printf("\n");
                                        }
                                        break;
```

```
                        }
                }

                //No hay cruzamiento, se pasan igual
                if(autom == 0)
                {
                        printf("\n");
                        printf("    No hay punto de corte\n");
                        printf("    Serán los cromosomas %d y %d de la nueva
generación\n",ic + 1, ic + 2);
                }

                for(i = 0; i < qe; i++)
                {
                        newSample[ic][i] = c1[i];
                        newSample[ic + 1][i] = c2[i];
                        if(autom == 0)
                        {
                                printf("    (%d)
[%d]\n",newSample[ic][i],newSample[ic+1][i]);
                        }
                }
                if(autom == 0)
                {
                        printf("\n");
                }
        }

        ic = ic + 2;
    }
    while(ic<qi);

    if(autom == 0)
    {
        printf("\n");
        printf("Cruzamiento finalizado\n");
        getchar();
    }
}

void mutation(int autom, int qe, int qi, int pm, int newSample[qi][qe])
{
    int i,r,random;

    if(autom == 0)
    {
        printf("\n");
        printf("\n");
        printf("Se aplicará mutación con una probabilidad de pm = %d%%:\n",pm);
        getchar();
    }

    for(i = 0; i < qi; i++)
    {
        random = rand() % 100;

        if(random < pm)
```

```
		{
			if(autom == 0)
			{
				printf("\n");
				printf("  La mutación se ha activado para el cromosoma %d
(Probabilidad: %d < %d)\n",i + 1,random,pm);
			}
			r = rand() % qe;
			newSample[i][r] = (newSample[i][r] + 1) % 2;
		}
		else
		{
			if(autom == 0)
			{
				printf("\n");
				printf("  La mutación no se ha activado para el cromosoma %d
(Probabilidad: !  %d < %d  !)\n",i + 1,random,pm);
			}
		}
	}

	if(autom == 0)
	{
		printf("\n");
		printf("Mutación finalizada\n");
		getchar();
	}

	return;
}

void backToSample(int qe, int qi, int sample[qi][qe], int newSample[qi][qe])
{
	int i,j;
	for(i = 0; i < qi; i++)
	{
		for(j = 0; j < qe; j++)
		{
			sample[i][j] = newSample[i][j];
		}
	}
	return;
}
```

Anexo 2: Impresión de pantalla completa al ejecutar el programa

```
============================== ALGORITMO GENÉTICO ==============================

--------------------------------
|  Calibración del algoritmo:
|  Seleccione algún número si desea cambiar su valor (o 0 para continuar):
|  0: Listo. Continuar
|  1: qe (Cantidad de elementos por individuo): 10
|  2: qi (Cantidad de individuos): 30
|  3: pc (Probabilidad de cruzamiento): 50
|  4: pm (Probabilidad de mutación): 50
|  5: qelitism (Individuos a heredar automáticamente por elitismo): 2
|  6: autom (Modo de acción): 0
|  7: qg (Cantidad de generaciones): 10
|
```

```
La matriz de muestras en la generación 0:
1  1  0  0  1  0  0  1  0  0
0  0  1  1  0  1  1  0  1  1
0  0  1  1  1  0  0  0  1  0
0  1  0  0  1  1  0  1  0  1
1  0  0  1  1  1  1  1  1  0
0  1  0  1  1  0  0  1  1  0
0  0  0  1  1  0  1  0  0  1
0  1  0  0  1  0  0  0  0  0
0  0  0  1  1  1  1  0  1  1
1  1  1  1  0  1  1  1  0  1
0  1  0  1  1  0  1  1  1  0
1  0  0  1  1  0  0  1  1  1
1  1  1  0  1  0  0  1  1  0
0  0  1  1  0  1  1  1  1  0
0  0  1  0  0  1  1  0  0  0
0  0  1  0  1  0  0  1  0  0
1  0  1  0  0  0  0  0  0  0
1  1  0  1  0  0  1  1  1  1
1  1  0  1  0  0  1  1  1  1
0  0  0  1  0  0  0  1  0  0
0  1  1  1  1  0  1  0  1  0
0  0  0  0  1  0  1  1  1  1
0  1  0  0  0  0  1  0  0  0
0  0  0  1  1  1  1  1  0  1
0  0  0  1  0  1  0  0  0  0
1  1  0  1  0  0  1  1  0  0
1  0  0  0  1  1  1  1  0  0
0  1  1  1  0  0  0  1  1  1
0  0  1  1  1  0  1  0  0  1
1  0  1  0  1  0  1  1  0  1
```

```
La matriz de muestras evaluada en F y Fa de la generación 0:
1  1  0  0  1  0  0  1  0  0      4   4
0  0  1  1  0  1  1  0  1  1      6   10
0  0  1  1  1  0  0  0  1  0      4   14
0  1  0  0  1  1  0  1  0  1      5   19
1  0  0  1  1  1  1  1  1  0      7   26
0  1  0  1  1  0  0  1  1  0      5   31
0  0  0  1  1  0  1  0  0  1      4   35
0  1  0  0  1  0  0  0  0  0      2   37
0  0  0  1  1  1  1  0  1  1      6   43
1  1  1  1  0  1  1  1  0  1      8   51
0  1  0  1  1  0  1  1  1  0      6   57
1  0  0  1  1  0  0  1  1  1      6   63
1  1  1  0  1  0  0  1  1  0      6   69
0  0  1  1  0  1  1  1  1  0      6   75
0  0  1  0  0  1  1  0  0  0      3   78
0  0  1  0  1  0  0  1  0  0      3   81
1  0  1  0  0  0  0  0  0  0      2   83
1  1  0  1  0  0  1  1  1  1      7   90
1  1  0  1  0  0  1  1  1  1      7   97
0  0  0  1  0  0  0  1  0  0      2   99
0  1  1  1  1  0  1  0  1  0      6   105
0  0  0  0  1  0  1  1  1  1      5   110
0  1  0  0  0  0  1  0  0  0      2   112
0  0  0  1  1  1  1  1  0  1      6   118
0  0  0  1  0  1  0  0  0  0      2   120
1  1  0  1  0  0  1  1  0  0      5   125
1  0  0  0  1  1  1  1  0  0      5   130
0  1  1  1  0  0  0  1  1  1      6   136
0  0  1  1  1  0  1  0  0  1      5   141
1  0  1  0  1  0  1  1  0  1      6   147
```

```
La matriz de muestras ordenada y evaluada en F y Fa de la generación 0:
1  1  1  1  0  1  1  1  0  1      8   8
1  0  0  1  1  1  1  1  1  0      7   15
1  1  0  1  0  0  1  1  1  1      7   22
1  1  0  1  0  0  1  1  1  1      7   29
0  0  1  1  0  1  1  0  1  1      6   35
0  0  0  1  1  1  1  0  1  1      6   41
0  1  0  1  1  0  1  1  1  0      6   47
1  0  0  1  1  0  0  1  1  1      6   53
1  1  1  0  1  0  0  1  1  0      6   59
0  0  1  1  0  1  1  1  1  0      6   65
0  1  1  1  1  0  1  0  1  0      6   71
0  0  0  1  1  1  1  1  0  1      6   77
0  1  1  1  0  0  0  1  1  1      6   83
1  0  1  0  1  0  1  1  0  1      6   89
0  1  0  0  1  1  0  1  0  1      5   94
0  1  0  1  1  0  0  1  1  0      5   99
0  0  0  0  1  0  1  1  1  1      5   104
1  1  0  1  0  0  1  1  0  0      5   109
1  0  0  0  1  1  1  1  0  0      5   114
0  0  1  1  1  0  1  0  0  1      5   119
1  1  0  0  1  0  0  1  0  0      4   123
0  0  1  1  1  0  0  0  1  0      4   127
0  0  0  1  1  0  1  0  0  1      4   131
0  0  1  0  0  1  1  0  0  0      3   134
0  0  1  0  1  0  0  1  0  0      3   137
0  1  0  0  1  0  0  0  0  0      2   139
1  0  1  0  0  0  0  0  0  0      2   141
0  0  0  1  0  0  0  1  0  0      2   143
0  1  0  0  0  0  1  0  0  0      2   145
0  0  0  1  0  1  0  0  0  0      2   147
```

```
Se aplica elitismo a qelitism = 2 cromosomas:
1  1  1  1  0  1  1  1  0  1      8
1  0  0  1  1  1  1  1  1  0      7
```

```
Comenzamos a construir la nueva generación:
1  1  1  1  0  1  1  1  0  1
1  0  0  1  1  1  1  1  1  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
```

```
El cruzamiento se ha activado (Probabilidad: 7 < 50)

  Aplicamos selección por ruleta para el primer padre obteniendo 51
  Por lo tanto, el cromosoma a escoger será 1 0 0 1 1 0 0 1 1 1

  Aplicamos selección por ruleta para el segundo padre obteniendo 28
  Por lo tanto, el cromosoma a escoger será 1 1 0 1 0 0 1 1 1 1

  Se ha seleccionado aleatoriamente un punto de corte entre el gen 9 y 10
  Serán los cromosomas 27 y 28 de la nueva generación
  (1) [1]
  (0) [1]
  (0) [0]
  (1) [1]
  (1) [0]
  (0) [0]
  (0) [1]
  (1) [1]
  (1) [1]
  [1] (1)


El cruzamiento no se ha activado (Probabilidad: !  66 < 50  !)

  Aplicamos selección por ruleta para el primer padre obteniendo 96
  Por lo tanto, el cromosoma a escoger será 0 1 0 1 1 0 0 1 1 0

  Aplicamos selección por ruleta para el segundo padre obteniendo 73
  Por lo tanto, el cromosoma a escoger será 0 0 0 1 1 1 1 1 0 1

  No hay punto de corte
  Serán los cromosomas 29 y 30 de la nueva generación
  (0) [0]
  (1) [0]
  (0) [0]
  (1) [1]
  (1) [1]
  (0) [1]
  (0) [1]
  (1) [1]
  (1) [0]
  (0) [1]
```

```
La nueva generación ha pasado la etapa de cruzamiento exitosamente:
1  1  1  1  0  1  1  1  0  1
1  0  0  1  1  1  1  1  1  0
0  0  1  1  1  0  0  0  1  0
0  1  0  0  0  0  1  0  0  0
0  0  1  1  0  1  1  0  1  0
0  1  1  1  1  0  1  0  1  1
0  0  1  1  0  1  1  1  1  1
0  1  1  1  0  0  0  1  1  0
0  0  1  1  0  1  1  1  1  0
0  0  1  1  1  0  1  0  0  1
1  1  1  1  0  0  1  1  1  0
0  1  0  1  1  1  1  1  0  1
1  0  0  0  1  0  1  1  0  1
1  0  1  0  1  1  1  1  0  0
1  1  1  0  1  1  1  1  0  0
1  0  0  0  1  0  0  1  1  0
0  0  0  1  1  0  1  0  0  1
0  0  0  1  1  0  1  0  0  1
1  0  1  0  1  0  1  1  0  1
1  1  0  1  0  0  1  1  1  1
1  1  1  1  1  0  0  0  1  0
0  0  1  1  0  1  1  1  0  1
1  0  0  1  1  0  0  0  1  0
0  0  1  1  1  0  0  1  1  1
0  0  0  1  1  1  1  1  0  1
1  1  0  1  0  0  1  1  1  1
1  0  0  1  1  0  0  1  1  1
1  1  0  1  0  0  1  1  1  1
0  1  0  1  1  0  0  1  1  0
0  0  0  1  1  1  1  1  0  1
```

```
La mutación no se ha activado para el cromosoma 2 (Probabilidad: !  61 < 50  !)
La mutación no se ha activado para el cromosoma 3 (Probabilidad: !  69 < 50  !)
La mutación se ha activado para el cromosoma 4 (Probabilidad: 9 < 50)
La mutación se ha activado para el cromosoma 5 (Probabilidad: 14 < 50)
La mutación se ha activado para el cromosoma 6 (Probabilidad: 20 < 50)
La mutación no se ha activado para el cromosoma 7 (Probabilidad: !  83 < 50  !)
La mutación no se ha activado para el cromosoma 8 (Probabilidad: !  67 < 50  !)
La mutación se ha activado para el cromosoma 9 (Probabilidad: 30 < 50)
La mutación no se ha activado para el cromosoma 10 (Probabilidad: !  72 < 50  !)
La mutación no se ha activado para el cromosoma 11 (Probabilidad: !  58 < 50  !)
La mutación no se ha activado para el cromosoma 12 (Probabilidad: !  73 < 50  !)
La mutación no se ha activado para el cromosoma 13 (Probabilidad: !  97 < 50  !)
La mutación no se ha activado para el cromosoma 14 (Probabilidad: !  75 < 50  !)
La mutación se ha activado para el cromosoma 15 (Probabilidad: 28 < 50)
La mutación se ha activado para el cromosoma 16 (Probabilidad: 27 < 50)
La mutación no se ha activado para el cromosoma 17 (Probabilidad: !  98 < 50  !)
La mutación no se ha activado para el cromosoma 18 (Probabilidad: !  99 < 50  !)
La mutación se ha activado para el cromosoma 19 (Probabilidad: 48 < 50)
La mutación se ha activado para el cromosoma 20 (Probabilidad: 46 < 50)
La mutación no se ha activado para el cromosoma 21 (Probabilidad: !  70 < 50  !)
La mutación se ha activado para el cromosoma 22 (Probabilidad: 44 < 50)
La mutación no se ha activado para el cromosoma 23 (Probabilidad: !  57 < 50  !)
La mutación no se ha activado para el cromosoma 24 (Probabilidad: !  81 < 50  !)
La mutación no se ha activado para el cromosoma 25 (Probabilidad: !  79 < 50  !)
La mutación se ha activado para el cromosoma 26 (Probabilidad: 24 < 50)
La mutación se ha activado para el cromosoma 27 (Probabilidad: 48 < 50)
La mutación no se ha activado para el cromosoma 28 (Probabilidad: !  85 < 50  !)
La mutación se ha activado para el cromosoma 29 (Probabilidad: 36 < 50)
La mutación se ha activado para el cromosoma 30 (Probabilidad: 8 < 50)
```

```
La nueva generación ha pasado la etapa de mutación exitosamente y todos los operadores han sido aplicados
De modo que la matriz de muestras en la generación 1 es:
1  1  1  1  0  1  1  1  0  1
1  0  0  1  1  1  1  1  1  0
0  0  1  1  1  0  0  0  1  0
0  1  0  1  0  0  1  0  0  0
0  0  1  1  1  1  1  0  1  0
0  1  1  0  1  0  1  0  1  1
0  0  1  1  0  1  1  1  1  1
0  1  1  1  0  0  0  1  1  0
0  0  1  1  0  1  1  1  1  1
0  0  1  1  1  0  1  0  0  1
1  1  1  1  0  0  1  1  1  0
0  1  0  1  1  1  1  1  0  1
1  0  0  0  1  0  1  1  0  1
1  0  1  0  1  1  1  1  0  0
1  1  1  0  0  1  1  1  0  0
1  0  0  0  1  0  1  1  1  0
0  0  0  1  1  0  1  0  0  1
0  0  0  1  1  0  1  0  0  1
1  0  1  0  0  0  1  1  0  1
0  1  0  1  0  0  1  1  1  1
1  1  1  1  1  0  0  0  1  0
0  0  0  1  0  1  1  1  0  1
1  0  0  1  1  0  0  0  1  0
0  0  1  1  1  0  0  1  1  1
0  0  0  1  1  1  1  1  0  1
1  1  0  1  1  0  1  1  1  1
1  0  0  1  1  0  0  0  1  1
1  1  0  1  0  0  1  1  1  1
0  1  0  1  1  0  0  1  1  1
0  0  0  1  0  1  1  1  0  1
```

Anexo 3: Algunas capturas del código en el IDE

```c
/*
    Universidad Autónoma de Aguascalientes

        Centro de Ciencias Básicas
  Departamento de Ciencias de la Computación
            Metaheurísticas I

                7° "A"

   Actividad 2.01: Algoritmo Genético Simple

    Profesor: Francisco Javier Luna Rosas

    Alumno: Joel Alejandro Espinoza Sánchez

  Fecha de Entrega: 1° de septiembre del 2021
*/
//Cargamos las librerías
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <time.h>
#include <math.h>

void setValues(int autom, int pc, int pm, int qe, int qelitism, int qi, int qg, int *automP, int *pcP, int *pmP, int *qeP, int *qelitismP, int *qiP, int *qgP);
void startTest(int qe, int qi, int sample[qi][qe]);
void nullNewSample(int qe, int qi, int newSample[qi][qe]);
void evaluate(int qe, int qi, int sample[qi][qe], int results[qi][2]);
void BubbleSort(int qe, int qi, int sample[qi][qe], int results[qi][2]);
void elitism(int qe, int qi, int qelitism, int ic, int sample[qi][qe], int newSample[qi][qe]);
void crossing(int autom, int qe, int qi, int ic, int pc, int sample[qi][qe], int results[qi][2], int newSample[qi][qe]);
void mutation(int autom, int qe, int qi, int pm, int newSample[qi][qe]);
void backToSample(int qe, int qi, int sample[qi][qe], int newSample[qi][qe]);
```

```c
main()
{
    setlocale(LC_ALL,"");
    srand(time(NULL));

    //1. Declaramos las variables que usaremos
    /*
        autom: Modo de acción, donde 0 será Manual y 1 Automático, que:
                Si autom = 0 ejecutará todo el proceso rápidamente lo hará qg veces
                Si autom = 1 ejecutará el proceso de uno en uno y el usuario en cada generación decide si generar otra generación o parar
        ic: Iterador de cromosomas
        ig: Iterador de generaciones
        pc: Probabilidad de cruzamiento (0 < pc < 100)
        pm: Probabilidad de mutación (0 < pm < 100)
        qe: Cantidad de elementos
        qelitism: Cantidad de cromosomas que se herdan automáticamente por elitismo
        qi: Cantidad de individuos
        qg: Cantidad de generaciones
    */
    int autom, ic, ig, pc, pm, qe, qelitism, qi, qg, repeat, repeat1;

    printf("============================ ALGORITMO GENÉTICO ============================\n");

    do
    {
        autom = 0;
        ic = 0;
        ig = 0;
        pc = 50;
        pm = 50;
        qe = 10;
        qelitism = 2;
        qi = 30;
        qg = 10;
        repeat = 0;
```

```c
//2. Calibramos el programa
setValues(autom,pc,pm,qe,qelitism,qi,qg,&autom,&pc,&pm,&qe,&qelitism,&qi,&qg);

//3. Declaramos más variables
/*
    sample: La matriz de muestra que guardará los bits
    newSample: La matriz de muestra que guardará los nuevos bits
    results: Estructura para guardar la cantidad total de bits
*/
int sample[qi][qe], results[qi][2], newSample[qi][qe];

//4. Generamos aleatoriamente la población inicial
startTest(qe,qi,sample);

printf("\n");
printf("\n");
printf("\n");
int i,j;
if(autom == 0)
{
    printf("La matriz de muestras en la generación %d:\n",ig);
    for(i = 0; i < qi; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",sample[i][j]);
        }
        printf("\n");
    }
    getchar();
    getchar();
}
```

```
do
{
    ic = 0;

    nullNewSample(qe,qi,newSample);

    //5. Evaluamos a todas las muestras junto con sus evaluaciones acumuladas
    evaluate(qe,qi,sample,results);

    if(autom == 0)
    {
        printf("\n");
        printf("\n");
        printf("La matriz de muestras evaluada en F y Fa de la generación %d:\n",ig);
        for(i = 0; i < qi; i++)
        {
            for(j = 0; j < qe; j++)
            {
                printf("%d  ",sample[i][j]);
            }
            printf("    %d   %d\n",results[i][0],results[i][1]);
        }
        getchar();
    }

    //6. Ordenamos
    BubbleSort(qe,qi,sample,results);
    evaluate(qe,qi,sample,results);
```

```
if(autom == 0)
{
    printf("\n");
    printf("\n");
    printf("La matriz de muestras ordenada y evaluada en F y Fa de la generación %d:\n",ig);
    for(i = 0; i < qi; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",sample[i][j]);
        }
        printf("    %d   %d\n",results[i][0],results[i][1]);
    }
    getchar();
}

//7. Buscamos los qelitism mejores
elitism(qe,qi,qelitism,ic,sample,newSample);

ic = qelitism;

if(autom == 0)
{
    printf("\n");
    printf("\n");
    printf("Se aplica elitismo a qelitism = %d cromosomas:\n",qelitism);
    for(i = 0; i < qelitism; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",sample[i][j]);
        }
        printf("    %d\n",results[i][0]);
    }
    getchar();
```

```
    printf("\n");
    printf("\n");
    printf("Comenzamos a construir la nueva generación:\n");
    for(i = 0; i < qi; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",newSample[i][j]);
        }
        printf("\n");
    }
    getchar();
}

if (autom == 1)
{
    printf("\n");
    printf("\n");
    printf("Los %d mejores cromosomas de la generación %d son:\n",qelitism,ig);
    for(i = 0; i < qelitism; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",sample[i][j]);
        }
        printf("    %d\n",results[i][0]);
    }
}

//8. Aplicamos cruzamiento

crossing(autom,qe,qi,ic,pc,sample,results,newSample);
```

```c
//8. Aplicamos cruzamiento

crossing(autom,qe,qi,ic,pc,sample,results,newSample);

if(autom == 0)
{
    printf("\n");
    printf("\n");
    printf("La nueva generación ha pasado la etapa de cruzamiento exitosamente:\n");
    for(i = 0; i < qi; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",newSample[i][j]);
        }
        printf("\n");
    }
    getchar();
}

//9. Aplicamos mutación
mutation(autom,qe,qi,pm,newSample);

ig++;

if(autom == 0)
{
    printf("\n");
    printf("\n");
    printf("La nueva generación ha pasado la etapa de mutación exitosamente y todos los operadores han sido aplicados\n");
    printf("De modo que la matriz de muestras en la generación %d es:\n",ig);
```

```
    }
    getchar();
    printf("¿Desea construir una nueva generación?\n");
    printf("0. No\n");
    printf("1. Sí\n");
    scanf("%d",&repeat1);
    if(repeat1 == 1)
    {
        qg++;
        getchar();
    }
    else
    {
        ig = qg + 1;
    }
}

if(autom == 1 && ig == qg + 1)
{
    printf("\n");
    printf("\n");
    printf("La matriz final de muestras en la generación %d es:\n",ig);
    for(i = 0; i < qi; i++)
    {
        for(j = 0; j < qe; j++)
        {
            printf("%d  ",newSample[i][j]);
        }
        printf("\n");
    }
    getchar();
}

//10. Transformamos newSample en sample
backToSample(qe,qi,sample,newSample);
```