Facultad Regional Rosario
Universidad Tecnológica Nacional

**Carrera:**

Ingeniería en Sistemas de Información

**Asignatura:**

Algoritmos Genéticos

**Comisión:**

3ek03

**Ciclo lectivo:**

2024

**Título:**

Trabajo Práctico N°1

**Docentes:**

DIAZ, Daniela

LOMBARDO, Víctor

**Alumno:**

GALLEGOS, Nicolás Gabriel – Leg.: 51367 – nicogabrielgallegos@gmail.com

# Índice:

# Enunciado:

Hacer un programa que utilice un Algoritmo Genético Canónico para buscar un máximo de una función:

$$f(x) = \left(\frac{x}{coef}\right)^2, \qquad x \in [0, 2^{30} - 1]$$

donde

$$coef = 2^{30} - 1$$

teniendo en cuenta los siguientes datos:

- Probabilidad de Crossover = $0.75$
- Probabilidad de Mutación = $0.05$
- Población Inicial: $10$ $individuos$
- Ciclos del programa: $20$
- Método de Selección: $Definido\ en\ el\ inciso\ correspondiente$
- Método de Crossover: $1$ $punto$
- Método de Mutación: $Invertida$

Mostrar finalmente, el Cromosoma correspondiente al valor máximo, el valor máximo, el valor mínimo y promedio obtenido de cada población.

Imprimir las tablas correspondientes a ciclos de 20, 100 y 200 generaciones.

Comparar las salidas corriendo el mismo programa en distintos ciclos y con todos los cambios que se consideren oportunos en los parámetros de entrada de manera de enriquecer las conclusiones.

Se deberá realizar el proceso mencionado con las siguientes variaciones:

Inciso A:

- Método de Selección: $Ruleta$

Inciso B:

- Método de Selección: $Torneo$

Inciso C:

- Método de Selección: $Ruleta + Elitismo$

## Herramientas de programación:

Para abordar este trabajo práctico, se hará uso del lenguaje de programación Python, en su versión 3.12.2 64-bit.

El motivo de elegir este lenguaje de programación es su facilidad y flexibilidad sobre otros lenguajes como podrían ser C++, C# o Java; ya que Python tiene tanto una sintaxis como un funcionamiento menos restrictivo que los demás lenguajes mencionados, lo que resulta más sencillo a la hora de manipular variables y datos dentro del algoritmo genético a desarrollar.

Además, Python cuenta con diversas librerías que no están solo relacionadas a los algoritmos genéticos, sino también a la creación de otros tipos de archivos (como por ejemplo hojas de cálculo .xlsx).

Para este trabajo, se usará la librería `openpyxl`, que se describe a sí misma como: *"Una librería de Python para leer/escribir archivos Excel 2012 xlsx/xlsm"*, cuyas páginas de proyecto y documentación son las ya enlazadas. Con esta librería, el programa principal escribirá los datos de los distintos ciclos y generaciones directamente sobre una hoja de cálculos.

También se utilizarán librerías más generales como `os`, para limpiar la consola de salida; `random`, para la generación de números aleatorios; y `math`, para realizar algunas operaciones matemáticas como la potenciación.

Por último, se utilizarán dos librerías propias `agxl` y `pygen`.

La primera, `agxl`, hace un uso específico de la librería `openpyxl` para volcar los datos de cada generación y de cada ciclo directamente en la planilla de Excel.

La segunda, `pygen`, define la clase `AlgoritmoGenetico`, que cuenta con las funciones básicas para poder crear un programa que ejecute un algoritmo genético.

Más adelante en este mismo documento se hablará en detalle sobre estas librerías propias y su uso en el programa principal.

# Metodología de desarrollo abordada:

Para llevar a cabo el programa principal, debemos construir un algoritmo genético canónico o simple. Para construir uno, tenemos que implementar las siguientes funciones y operadores genéticos:

## Fitness

Será la encargada de computar la función evaluación de los individuos de la población. El resultado de esta función le deberá asignar a cada individuo un *"peso"* o *"valor"* dentro de la población en la que se encuentra. La fórmula que se utilizará para calcular el resultado de la evaluación es:

$$\frac{f(i)}{\sum_{k=1}^{10} f(i_k)}$$

donde $f(x)$ es la función definida en el enunciado, que a partir de ahora llamaremos *"función objetivo"*.

## Selection

Esta función es un operador genético que seleccionará dos individuos de la población para intentar realizar el cruce entre ellos. El criterio con el que se seleccionarán a los individuos variará dependiendo del inciso.

Inciso A:

En este primer inciso, se utilizará el método de la ruleta. Este consiste en elegir a los individuos con una probabilidad proporcional a su valor dentro de la población, es decir, al valor de su fitness.

Inciso B:

En este caso se usará el método de selección por torneo. En este método se eligen aleatoriamente una cierta cantidad de individuos, y el que tiene un mayor valor fitness es seleccionado finalmente para reproducirse. La cantidad de individuos que participarán en el torneo será el 40% de la población total.

Inciso C:

Muy similar al inciso A, con la diferencia de que se implementará un método elitista, donde un cierto porcentaje de los individuos mejores valuados pasa directamente a la próxima generación, además del uso de la ruleta para generar al resto de la población. Los individuos que pasarán a la siguiente generación mediante el elitismo será el 20% de la población total.

## Crossover

Este será el principal operador genético de nuestro algoritmo genético ya que, sin él, sería imposible generar nuevas poblaciones basadas en la población ya existente.

Luego del proceso de selección de dos individuos, estos tendrán una probabilidad de reproducirse. Esta probabilidad se encuentra en el enunciado y es del 75%.

En el caso de que los individuos seleccionados no logren reproducirse, estos pasarán a formar parte de la población de la siguiente generación.

El tipo de cruce a utilizar será el crossover de 1 punto, en donde los dos individuos seleccionados se *"cortan"* por un punto, y su material genético es intercambiado por el del otro individuo.

Este punto de corte también es elegido de forma aleatoria.

## Mutation

Este será nuestro operador genético secundario, y se encargará de alterar una porción del material genético de un individuo que pertenecerá a la población de la siguiente generación.

El individuo puede mutar antes de entrar a la nueva población con una probabilidad de 5% definida también en el enunciado.

El tipo de mutación a utilizar será la mutación invertida, en donde es elegido un bit aleatorio del individuo y en caso de ser este un 1, será cambiado por un 0 y viceversa.

## Funciones adicionales

Además de las funciones principales ya mencionadas, necesitaremos otras funciones para procesos más triviales como la creación de la población inicial del algoritmo genético, el bucle principal del ciclo reproductivo, el pasaje de los datos al documento Excel, etc. En la siguiente sección se detallará el proceso específico que realiza cada función

## Código del programa principal:

Se explicará de forma resumida el funcionamiento del programa principal. Sin embargo, el código completo del programa se encuentra subido en el repositorio AG-TP / Enunciado 1er Trabajo Práctico / de GitHub.

### Archivo `dependencias/agxl.py`:

Este módulo propio hace un uso especializado de la librería `openpyxl` y algunas de sus clases como `Worksheet` y `Workbook` para escribir los datos obtenidos del programa principal en un archivo de Excel.

`general_formateo`: da formato a la primera fila y columna de una página dentro del archivo Excel.

`generaciones_formateo`: da un formato más específico a la hoja de datos de las generaciones.

`generaciones_insertar_tabla`: inserta una tabla correspondiente a los datos de la población de una generación.

`generaciones_insertar_grafica`: inserta una gráfica correspondiente a los datos de la población de una generación.

`ciclos_formateo`: da un formato más específico a la hoja de datos de los ciclos.

`ciclos_insertar_tabla`: inserta una tabla correspondiente a los datos de las generaciones de un ciclo.

`ciclos_insertar_grafica`: inserta una gráfica correspondiente a los datos de las generaciones de un ciclo.

## Archivo `dependencias/pygen.py`:

Como fue mencionado anteriormente, en este módulo se define la clase `AlgoritmoGenetico` que puede ser instanciada con una serie de parámetros que definirán el modo de trabajo del algoritmo genético.

Estos parámetros son:

`tipo_seleccion:str`, especifica el tipo de selección que utilizará el algoritmo genético. Default = `'ruleta'`.

`tipo_crossover:str`, especifica el tipo de cruce que utilizará el algoritmo genético. Default = `'1pto'`.

`tipo_mutacion:str`, especifica el tipo de mutación que utilizará el algoritmo genético. Default = `'invertida'`.

`prob_crossover:float`, especifica la probabilidad de cruce entre los individuos. Default = `1`.

`prob_mutacion:float`, especifica la probabilidad de mutación entre los individuos. Default = `0`.

`porcentaje_elitismo:float`, especifica el porcentaje de la población que pasa directamente a la siguiente generación mediante elitismo. Esto es independiente del tipo de selección elegido. Default = `0`.

`porcentaje_seleccion:float`, especifica el porcentaje de la población que será seleccionada dentro de tipos de selecciones especiales como por ejemplo la selección por torneo. Esta variable no tiene efecto alguno con tipos de selección como el de ruleta. Default = `0`.

`cant_individuos:int`, especifica el tamaño de la población. Default = `10`.

`cant_generaciones:int`, especifica la cantidad de iteraciones que debe realizar el algoritmo genético antes de llegar a su población final. Default = `20`.

`dominio:tuple[int, int]`, especifica el dominio de la función objetivo. Default = `[0, 255]`.

`funcion_objetivo:function`, especifica la función objetivo que se desea maximizar. Default = `lambda x : x`.

Una vez inicializada una instancia de esta clase, esta puede acceder a distintos métodos de instancia que se encargarán de simular la ejecución del algoritmo genético.

Los métodos principales son:

## Archivo `main.py`:

Cuenta con la inicialización de parámetros de la simulación, como la cantidad de individuos, ciclos y generaciones que se ejecutarán. Así como también define los datos de la función

## Glosario:

**Programa principal**: implementación del algoritmo genético en el lenguaje de programación utilizado.

**Generación**: población existente durante una *"corrida"* o *"iteración"* del algoritmo genético.

**Ciclo**: conjunto de generaciones dentro de una misma ejecución del programa principal.

**Función objetivo**: función que se busca maximizar con el algoritmo genético.