Aplicación de las matemáticas discretas para la resolución de problemas en la ciencia de la computación: Proyecto Computacional

Autores:

Juan Camilo Aguirre Gonzales
Mariana Guerrero Benavides
Alejandra Landinez Lamadrid

Dirigido a:
Lic. Ing. Alfonso M. Mancilla Herrera,
Esp. M.Sc. Ph.D. en Ciencias de la Computación

Estructuras Discretas
Universidad del Norte

Noviembre 2023
Barranquilla, Colombia

Video Equipo: Tres Trastes Tigres

Tabla de contenido

Índice

Lista de tablas

Índice de cuadros

Resumen

El siguiente proyecto computacional fue realizado, para atender los conceptos aprendidos en el transcurso de la materia de Estructuras Discretas. Los estudiantes dieron solución a distintos problemas, con propuestas de manera algebraica e informática.

En cuanto a las técnicas aplicadas, los estudiantes utilizaron las nociones de ecuaciones de recursividad, funciones generadoras ordinarias, aplicación del Método simbólico y Análisis Combinatorio. Además, se implementó el lenguaje de programación Python, para diseñar algoritmos, que dieran solución a casos de método simbólico, análisis combinatorio, cadenas de Markov, cifrado César y para la creación de un bot en Telegram. Lo anterior, fue realizado por medio de tres entregables para fechas entre las semanas 7 y 16 de la materia Estructuras Discretas. En estas entregas los estudiantes involucrados, debieron participar activamente en la resolución de los problemas planteados, trabajando de manera colectiva y colaborativa, donde debían analizar las estrategias y técnicas necesarias para desarrollar el proyecto de manera efectiva.

En resumen, el proyecto computacional abarca distintos problemas que requieren de su solución a través de la ciencia de la computación e informática. Además, es un trabajo que debe ser resuelto de forma colectiva, para llegar a cumplir con el cronograma y fines acordados. Es importante mencionar, que las soluciones a los problemas deben realizarse de forma asertiva, y estar relacionadas con la eficiencia y/o eficacia del producto.

1 Introducción

Hoy en día, el uso las ciencias de la computación y las matemáticas han tomado mayor relevancia para el estudio y análisis de problemas complejos, puesto que, permite la obtención nuevos métodos e implementaciones que ofrecen modelos de solución eficientes. Además, que con sus avances permiten mejorar dichos recursos.

Este proyecto aborda técnicas relacionadas con un proceso de construcción combinatoria. De lo anterior, se tiene el uso de las funciones generatrices ordinarias, que se han desarrollado en el proyecto, con el propósito de modelar cuestiones que requieran estructuras combinatorias y subconjuntos, que, han permitido la obtención de soluciones para el conteo de distintos casos, tales como contabilizar objetos o eventos, en el uso de la organización de datos y para el desarrollo de cálculos que demanden de una estructura combinatoria.

Dicho análisis se realizará, de lo aprendido por estudiantes de la asignatura de Estructuras Discretas del programa de Ingeniería de Sistemas de la Universidad del Norte durante el 2023. En su desarrollo se cuenta con el objetivo de resolver ejercicios de análisis combinatorio, a través del método simbólico y aplicación del principio de palomar, para obtener una función generadora ordinaria. También, se tiene el objetivo de desarrollar algoritmos en el lenguaje de programación Python, con el fin, de resolver problemas de matrices, funciones recursivas, cadenas de Markov y lenguaje natural.

2 Problema de investigación

Tomando en cuenta lo antes planteado, se formula el siguiente problema de investigación ¿en qué medida, implementar técnicas de estructuras discretas, influye en la construcción de soluciones eficientes en problemas de complejidad? De esta manera se analizarán distintas metodologías que apunten en hallar soluciones con la mayor eficiencia posible, y sus posibles consecuencias si se implementan o no, estas técnicas en las propuestas de solución.

En concordancia, se toma como referente el artículo realizado por [1] , que fue realizada para contribuir en el desarrollo de herramientas para la resolución de objetos combinatorios y conteo a su alrededor. Lo anterior, será de gran apoyo para la realización del informe, para sustentar una respuesta asertiva hacia el interrogante, por medio de una fuente de información, que permitirá enriquecer los aportes de este proyecto.

3 Justificación

Las estructuras discretas son fundamentales para las ciencias de la computación e informática, puesto que, muchas áreas de la tecnología requieren del uso de estos conceptos y aplicaciones. El conocimiento adquirido en la asignatura es de gran importancia para modelar soluciones de problemas en el mundo real. Además, en la programación son esenciales en el análisis de los algoritmos y su complejidad, optimización en una variedad de contextos, logística y diseño de redes.

Comprender los métodos para cuantificar y organizar los elementos de forma eficiente, es de gran importancia para tomar las decisiones, fundamentadas en el diseño de soluciones óptimas. De ahí que, el análisis combinatorio y las funciones generadoras ordinarias son cruciales en esta disciplina, pues, proporcionan unas nociones elementales para la formación de sistemas discretos, matemáticos e informáticos. En conjunto, estas permiten abordar problemáticas que requieren de un conteo, distribución, y organización de elementos, los cuales, son casos evidenciados en el contexto real y en sistemas de recursos informáticos.

En resumen, las estructuras discretas son una materia fundamental para el análisis de problemas complejos en el mundo real y en la ciencia de la computación, a través de una amplia variedad de soluciones, que ofrecen un sistema basado en la eficiencia y optimización. Asimismo, permiten el desarrollo de habilidades analíticas en la modelación y comprensión de sistemas discretos.

4 Objetivos

4.1 Objetivo general

Desarrollar en el estudiante habilidades direccionadas en la comprensión y aplicación de nociones esenciales para abordar problemas complejos, en la asignatura de Estructuras Discretas.

4.2 Objetivos específicos

- Analizar el concepto de las funciones generadoras ordinarias y su aplicación en estructuras combinatorias.
- Aplicar las técnicas de serie de potencias y fracciones parciales en la resolución de problemas relacionados con permutaciones y combinaciones.
- Relacionar el concepto de método simbólico con las funciones generadoras ordinarias, para hallar una formula en la solución de problemas de combinatoria.
- Comprender el concepto de la teoría del principio de palomar, para abordar problemas de combinatoria, y en su relación con las funciones generadoras ordinarias.

5 Marco teórico-conceptual

Desde el inicio de la asignatura se implementaron varios conceptos que fueron de gran aporte para la realización del siguiente proyecto computacional. A continuación, se detallan los conceptos tratados en el curso, para el desarrollo del proyecto.

1. **Funciones Generadoras Ordinarias (FGO)**
   Una función generadora ordinaria (FGO) es una serie de potencias donde los coeficientes corresponden a los términos de una secuencia.
   Estas secuencias numéricas pueden definirse a través de una función generadora, lo que significa que su solución es una secuencia de problemas con funciones.
   Dada una secuencia $\langle f(0),\ f(1),\ f(2),\ ..f(n),\ ...\rangle$, se define su función generadora como

$$F(z) = \sum_{n\geq 0} f(n)\cdot z^n = f(0)\cdot z^0 + f(1)\cdot z^1 + f(2)\cdot z^2 + \cdots + f(n)\cdot z^n + \cdots$$

   En este sentido, se toman los términos infinitos de la secuencia, y cada n-ésimo término, está multiplicado por la n-ésima potencia de z. Por otra parte, $f(n)$ representa todos los términos de la secuencia, los cuales, pasan a ser los coeficientes. Estos se hallan mediante las funciones generadoras y operaciones básicas. Luego, es posible hallar la fórmula para la secuencia.

$$F(z) \texttt{ Es la FGO de la secuencia } f(n).$$
$$F(z) = \sum_{n\geq 0} f(n)\cdot z^n \texttt{ Es la FGO de la secuencia } \langle f(n)\rangle_{n\geq 0}.$$

   Adicionalmente, las FGO son utilizadas para expresar coeficientes relacionados con estructuras combinatorias, tales como permutaciones y combinaciones. Puesto que, permiten resolver de manera sencilla dichos problemas, a través de una suma o multiplicación de polinomios.
   De ahí, que es posible que una función generadora se asocie a un problema de conteo, que al usar la FGO se pueda hallar una expresión cerrada para $f(n)$. Se hace uso de la siguiente notación $[z^n]F(z) = f_n$.

En la resolución de problemas de conteo se hace uso de estas funciones generadoras de manera general, por lo que es de gran aporte tener su registro:

$$[z^n]\frac{1}{(1-z)^m} = \binom{m+n-1}{n}$$
$$[z^n]\frac{z^m}{(1-z)^{m+1}} = \binom{n}{m}$$

Para resolver un problema de combinatoria, se puede hacer uso de los siguientes pasos [1]:

a) Modelar el problema como ecuación entera, incluyendo las restricciones. Para ello se toma en consideración la secuencia de los números a encontrar.

b) Plantear la función generadora para $f(n)$, de forma recursiva.

c) Hallar coeficiente de $F(Z)$, la cual, equivale a una expresión cerrada para cada elemento de la sucesión.

En cuanto a la realización del proyecto computacional, las FGO fueron una herramienta esencial para el cálculo de secuencias numéricas y en la resolución de problemas complejos, que requerían del conteo y análisis de datos.

2. **Método Simbólico**

El método simbólico es una técnica que permite definir estructuras combinatorias a partir de sus propiedades, mediante variables simbólicas y ecuaciones que representan dichas estructuras. Además, permite obtener la respectiva función generadora y una fórmula para las estructuras.
Este método es de gran utilidad en la resolución de problemas de conteo y organización de elementos o eventos. El diseño de este método se divide en cinco etapas:

a) Definir clase combinatoria (cadenas binarias, ternarias, cuaternarias, etc.)

b) Definir átomos de los elementos que conforman la clase combinatoria.

c) Construcción combinatoria.

d) Transformación a FGO.

e) Transformación a f(n).

Definición	Método Simbólico	Función Generadora
Clase Combinatoria	$\mathcal{A}$	$A(z)$
Átomo	$z_0,\ z_1,\ z_2,\ \ldots$	z
Vacío	ϵ	1
Unión	$\mathcal{A}+\mathcal{B}$	$A(z)+B(z)$
Pares Ordenados	$\mathcal{A}\times\mathcal{B}$	$A(z)\cdot B(z)$
Secuencia	$SEQ(\mathcal{A})$	$\dfrac{1}{1-A(z)}$

Cuadro 5.1: Método Simbólico

3. **Análisis Combinatorio**

La función combinatoria representa el número de subconjuntos de k elementos. Esta función responde a la pregunta ¿Cuántos elementos posee el conjunto de interés?

Su implementación fue denominada por una serie de movimientos, que son:

a) Seleccionar, escoger o colocar (Coeficiente binomial)

$$\binom{c}{c_k}, \quad 1 \leq c_k \leq c$$

b) Distribuir (Principio del Palomar)

$$\binom{n+k-1}{k}$$

c) Permutar (Coeficiente multinomial)

$$\binom{n}{\texttt{c veces}\, n_1, n_2, \ldots, n_c} = \frac{n!}{\texttt{c veces}\, n_1!, n_2!, \ldots, n_c!}$$

[2]

Asimismo, esta técnica fue desarrollada con la implementación del principio del palomar:
"Si k palomas vuelan e ingresan en n palomas y $k > n$, entonces al menos un palomar contendrá al menos dos palomas".

Cabe mencionar, que el principio asegura la existencia de un palomar con dos palomas o más, más no establece el localizador del palomar que contiene dos palomas o más [2]. Además, el principio es utilizado si en la solución es necesario distribuir elementos. Para aplicar el principio del palomar, es necesario definir los elementos que representan las palomas y los elementos que representan los palomares. Su representación está dada por:

$$\binom{n+k-1}{k} = \frac{(n+k-1)!}{k! \cdot (n-1)!}$$

Donde, su combinatoria representa el número total de formas en que se pueden distribuir k palomas en n palomares. En cuanto al diseño de la solución para el análisis combinatorio, esta se divide en cuatro etapas:

a) **Protocolo**
En esta primera etapa, se ingresan los datos de entrada donde n representa el número de elementos, c las categorías , y k son el número de elementos que voy a obtener del total de elementos.

b) **Prototipo**
El prototipo se representa con palomares abiertos, en donde se realiza el listado de caracteres del conjunto. Esta etapa puede realizarse de dos maneras.

- Se listan los caracteres que tienen la restricción.
- Se listan los caracteres que no tienen la restricción.

c) **Estrategia** [2]

Estrategias para la solución de problemas	Sin Palomar	Con Palomar
Directa	Estrategia 1	Estrategia 2
Por Complemento	Estrategia 3	Estrategia 4

Cuadro 5.2: Tabla de 2×2

a) **Desarrollo**

Se realiza la combinatoria dependiendo de la estrategia escogida.

Adicionalmente, se trabajaron los conceptos de lenguaje natural y los modelos de Markov, siendo implementados para el desarrollo de algoritmos, en el lenguaje de programación de Python.

Lenguaje Natural: El lenguaje natural es crucial en el desarrollo de algoritmos para el procesamiento y análisis de textos, y otras aplicaciones relacionadas. Su función radica en la comprensión de las palabras, frases y en la gramática del lenguaje de ahí que su implementación permite la obtención de un algoritmo eficiente.

Cadenas de Markov: Las cadenas de Markov son modelos que pertenecen a un sistema evolutivo en el tiempo, en el que su probabilidad de que suceda un evento depende del estado inmediatamente anterior con probabilidades que están fijas [3]. Son de gran utilidad en aplicaciones que requieren predicción de texto, series, modelado de sistema dinámico y análisis de redes.

En este caso, su uso fue esencial para el funcionamiento de algoritmos para predicción y análisis de determinados datos, en donde, a partir de un sitio web estático, se implementó un procesamiento de web scraping, en el que se utilizó un modelo de cadenas de Markov, del cual, se generó un texto aleatorio.

6 Metodología

En el transcurso de la asignatura de Estructuras Discretas, se ha aprendido de manera progresiva diversas estrategias, herramientas y técnicas que nos han servido para cumplir con las entregas propuestas en clase, y para su implementación en otras ramas y actividades de otras asignaturas. Un ejemplo de lo anterior, se observa en los entregables 1 y 2, en las que, con ayuda y guía del profesor, de manera practica y teórica explicaba el uso de las FGO (Funciones Generadoras Ordinarias) y la aplicación que tienen en el análisis combinatorio y método simbólico.

Para los problemas planteados en clase y asignados en los entregables,que requerían del diseño de algoritmos, se tomó en cuenta el uso del lenguaje de programación Python, implementándolo a través de la plataforma de *Google Colaboratory*, como herramienta complementaria. Adicionalmente, se hizo uso de diversas librerías para una mejor ejecución del programa a realizar, tales como *numpy* y *markovify*. Esto se evidencia en los entregables 1 y 3, donde, se utilizó *markovify* para generar las cadenas de texto aleatorias utilizando el proceso de *web scraping*, y para la creación de un bot de Telegram, el cual, se realizó utilizando un entorno virtual en Python.

Con esto en mente, durante el curso de Estructuras Discretas, se fomentó el uso del pensamiento creativo y eficaz, para la resolución de problemas matemáticos al aplicar distintas habilidades, técnicas, y herramientas computacionales, que son fundamentales para el profesional en Ingeniería de Sistemas y/o Ciencia de Datos.

7 Resultados

Entregable 1

En el primer entregable, principalmente nos enfocamos en aplicar las funciones generadoras ordinarias y secuencias numéricas. Además, contamos con ejercicios para codificar, en los cuales, logramos formular algoritmos que generaran distintos logos al aplicar tensores en Python. También, se trabajó en la visualización de gráficas en coordenadas de dos y tres dimensiones. Finalmente, adquirimos conocimiento en el procesamiento de las cadenas de Markov, para la generación de un texto ficticio.

Entregable 2

Por otra parte, en el segundo entregable profundizamos nuestro conocimiento en la implementación de técnicas de conteo y probabilidad. Se trabajó con el concepto de análisis combinatorio, siendo resuelto, por medio del método simbólico y las funciones generadoras, en donde, se realizaron cálculos de probabilidad y aplicación de distribuciones de distintos elementos. Asimismo, implementamos habilidades en el procesamiento de datos en Python, desde la obtención de datos de una página web, su análisis y conclusiones correspondientes, por medio de métodos gráficos.

Entregable 3

En el tercer entregable, los estudiantes pudieron desarrollar un bot en Telegram, logrando generar una comunicación con los usuarios por medio de distintos comandos. Además, cuenta con un menú iterativo con funcionalidades para aplicar distintas técnicas de matemáticas discretas, aplicado a la ciencia de la computación. Entre estas opciones, se cuenta con un cifrado césar, procesamiento de datos de probabilidad implementando cadenas de Markov, uso de relaciones de recurrencia lineal no homogéneas, y aplicación del método simbólico para estructuras combinatorias.

En general, el desarrollo de este proyecto computacional nos permitió abor-

dar problemas de estructuras complejas, en las que fue necesario el uso de habilidades matemáticas y científicas para dar un resultado acertado. Asimismo, logramos diseñar algoritmos de manera eficiente, calcular y analizar datos de diversos contextos. Durante el desarrollo de los entregables, pudimos fortalecer nuestro conocimiento en las técnicas aprendidas en el curso, y en el uso de la ciencia computacional, en donde, implementamos distintas herramientas y librerías de Python, que fueron de gran aporte en la formulación de algoritmos para los problemas propuestos en el proyecto.

8 Conclusión

En el transcurso del proyecto computacional, se han explorado diversas temáticas, entre estas, el uso de las FGO (Funciones Generadoras Ordinarias), tensores en Python, análisis combinatorio y de ciclos, entre muchos más. Lo anterior, junto a un número de habilidades y técnicas que al emplearlas correctamente, permiten colaborar en la solución de problemas computacionales, relacionadas con las matemáticas y en la programación. Los métodos de funciones recursivas en Python brindan una técnica eficiente de recursos, que son útiles en sistemas de gran escala con infinidad de datos y estructuras, y la implementación de funciones generadoras, es útil a la hora de enfrentar problemas matemáticos. Su eficacia a la hora del conteo y al encontrar secuencias numéricas, lo fortalece como una herramienta de gran nivel al estar en contra de situaciones de esta magnitud.

El proyecto computacional ha complementado nuestro conocimiento en la materia Estructuras Discretas, y ha proporcionado herramientas para encontrar situaciones, en las que, como ingenieros de sistemas y científicos de datos, son necesarias en la resolución de problemas complejos. Lo adquirido en este curso, tiene gran aplicación en temas relacionados a la computación numérica, como lo es la inteligencia artificial, el hallazgo de secuencias en optimización, criptografía y generación de texto, entre otras ramas relacionadas a la computación y uso/creación de software.

Referencias

[1] S. N. Benedetti, C., "Introducción a la combinatoria enumerativa. lecturas matemáticas," 2019.

[2] A. M. Herrera, "Análisis combinatorio."

[3] U. N. de Colombia, "Clase 23. aplicaciones: Cadenas de markov." 2014. [Online]. Available: https://ciencias.medellin.unal.edu.co/cursos/algebra-lineal/clases/8-clases/25-clase-23-aplicaciones-cadenas-de-markov.html

A Anexo

Funciones Generadoras Ordinarias

Descripción

Una función generadora es una serie de potencias cuyos coeficientes corresponden a los términos de una secuencia. De este modo, las secuencias pueden ser definidas mediante una función generadora, lo que a su vez significa que permiten transformar problemas con secuencias en problemas con funciones.

Dada una secuencia $\langle f(0),\ f(1),\ f(2),\ ..f(n),\ ...\rangle$, se define su función generadora como

$$F(z) = \sum_{n\geq 0} f(n)\cdot z^n = f(0)\cdot z^0 + f(1)\cdot z^1 + f(2)\cdot z^2 + \cdots + f(n)\cdot z^n + \cdots \quad (1)$$

En (1) se observa que se están tomando todos los términos de la secuencia, que son infinitos, y que cada n-ésimo término, de la secuencia, está multiplicado por la n-ésima potencia de z. El término $f(n)$ representa todos los términos de la secuencia, quienes en la función generadora, pasan a ser los coeficientes. **De modo que una vez se tiene la función generadora de una secuencia, es posible hallar la fórmula para la secuencia a partir del hallazgo de los coeficientes de la función generadora**.

Por ejemplo dada la secuencia de unos, donde $f_n = 1$, es decir, todos los términos de la secuencia son 1, su función generadora será

$$F(z) = \sum_{n\geq 0} 1\cdot z^n = 1\cdot z^0 + 1\cdot z^1 + 1\cdot z^2 + \cdots + 1\cdot z^n + \cdots \quad (2)$$

Esta función generadora, corresponde a una serie geométrica donde el radio es igual a 1, y converge a un valor que corresponde a

$$F(z) = \sum_{n\geq 0} 1\cdot z^n = \frac{1}{1-z}. \quad (3)$$

Si en lugar de una secuencia $\langle 1\rangle_{n\geq 0}$, se tuviera la secuencia de los enteros no

negativos, es decir, $\langle n \rangle_{n \geq 0}$, la función generadora sería:

$$F(z) = \sum_{n \geq 0} n \cdot z^n = 0 + 1 \cdot z^1 + 2 \cdot z^2 + \cdots + n \cdot z^n + \cdots \tag{4}$$

Al igual que la ecuación (2), esta función generadora converge a un valor

$$F(z) = \sum_{n \geq 0} n \cdot z^n = \frac{z}{(1-z)^2} \tag{5}$$

Es importante resaltar que en lo relacionado a funciones generadoras, $F(z)$, el objetivo es hallar los coeficientes, $f(n)$, a partir de una función generadora dada, y para ello se recurre a las funciones generadoras básicas (ver Tabla 1) y a las operaciones básicas (ver Tabla 2) relacionadas con estas.

La forma utilizada para describir la relación descrita es:

$$F(z) \texttt{ Es la FGO de la secuencia } f(n).$$

Claramente, esto significa que

$$F(z) = \sum_{n \geq 0} f(n) \cdot z^n \texttt{ Es la FGO de la secuencia } \langle f(n) \rangle_{n \geq 0}.$$

Por ejemplo:

$$F(z) = \frac{1}{1-z} \texttt{ Es la FGO de la secuencia } f(n) = \langle 1 \rangle_{n \geq 0}.$$

$\langle 1,\, 1,\, 1,\, \ldots,\, 1,\, \ldots \rangle$	$\dfrac{1}{1-z} = \displaystyle\sum_{n\geq 0} z^n$
$\langle 0,\, 1,\, 2,\, 3,\, \ldots,\, n,\, \ldots \rangle$	$\dfrac{z}{(1-z)^2} = \displaystyle\sum_{n\geq 0} n z^n$
$\left\langle 0,\, 0,\, 1,\, 3,\, 6,\, \ldots,\, \binom{n}{2}, \ldots \right\rangle$	$\dfrac{z^2}{(1-z)^3} = \displaystyle\sum_{n\geq 2} \binom{n}{2} z^n$
$\left\langle 0,\, \ldots,\, 0,\, 1,\, m+1,\, \ldots,\, \binom{n}{m}, \ldots \right\rangle$	$\dfrac{z^m}{(1-z)^{m+1}} = \displaystyle\sum_{n\geq m} \binom{n}{m} z^n$
$\left\langle 1,\, m+1,\, \binom{m+2}{2},\, \binom{m+3}{3},\, \ldots,\, \binom{n}{m}, \ldots \right\rangle$	$\dfrac{1}{(1-z)^m} = \displaystyle\sum_{n\geq 0} \binom{n+m-1}{m} z^n$
$\langle 1,\, 0,\, 1,\, \ldots,\, 1,\, 0,\, \ldots \rangle$	$\dfrac{1}{1-z^2} = \displaystyle\sum_{n\geq 0} z^{2n} = \sum_{n\geq 0} \left(\dfrac{1+(-1)^n}{2} \right) z^n$
$\langle 2^0,\, 2^1,\, 2^2,\, 2^3,\, \ldots,\, 2^n,\, \ldots \rangle$	$\dfrac{1}{1-2z} = \displaystyle\sum_{n\geq 0} 2^n \cdot z^n = \sum_{n\geq 0} (2\cdot z)^n$
$\langle c^0,\, c^1,\, c^2,\, c^3,\, \ldots,\, c^n,\, \ldots \rangle$	$\dfrac{1}{1-cz} = \displaystyle\sum_{n\geq 0} c^n z^n$
$\left\langle \binom{m}{0}, \binom{m}{1}, \ldots, \binom{m}{n}, \ldots, , \binom{m}{1}, \binom{m}{0} \right\rangle.$	$(1+z)^m = \displaystyle\sum_{n\geq 0} \binom{m}{n} z^n$
$\left\langle 0,\, 1,\, \dfrac{1}{2},\, \dfrac{1}{3},\, \ldots,\, \dfrac{1}{n},\, \ldots \right\rangle$	$\ln\left(\dfrac{1}{1-z} \right) = \displaystyle\sum_{N\geq 1} \frac{z^n}{n}$

Cuadro A.1: Algunas Funciones Generadoras Básicas

En la Tabla 1 se muestran algunas funciones generadoras básicas y que mediante su manipulación, permiten hallar los coeficientes de otras funciones generadoras. Estas manipulaciones se realizan mediante el uso de las operaciones definidas para las funciones generadoras.

Suma (o resta)

$$F(z) + G(z) \qquad \sum_{n\geq 0} (f(n) + g(n)) \cdot z^n \qquad f(0)+g(0),\ f(1)+g(1),\ \ldots,\ f(n)+g(n),\ \ldots$$

Escala

$$F(k \cdot z) \qquad \sum_{n\geq 0} f(n) \cdot k^n \cdot z^n \qquad f_0, kf_1, k^2 f_2,\ \ldots,\ k^n f_n,\ \ldots$$

Derivada

$$F'(z) \qquad \sum_{n\geq 0} (n+1) \cdot f_{n+1} \cdot z^n \qquad f_1,\ 2{\cdot}f_2,\ 3{\cdot}f_3,\ \ldots, n{\cdot}f_n\ \ldots$$

Suma Parcial

$$\frac{F(z)}{1-z} \qquad \sum_{n\geq 0} \left(\sum_{0\leq k\leq n} f_k \right) z^n \qquad f_0,\ f_0+f_1, \ldots,\ \sum_{0\leq k\leq n} f_k, \ldots$$

Convolución

$$F(z) \cdot G(z) \qquad \sum_{n\geq 0} \left(\sum_{0\leq k\leq n} f_k \cdot g_{n-k} \right) z^n \qquad f_0 g_0,\ f_1 g_0 + f_0 g_1,\ \ldots,\ \sum_{0\leq k\leq n} f_k \cdot g_{n-k}$$

Desplazar a la derecha $\qquad 0,\ f_0,\ f_1,\ f_2, \ldots, f_n, \ldots$

$$z \cdot F(z) \qquad \sum_{n\geq 1} f(n-1) \cdot z^n$$

Diferencia (Telescópica)

$$(1-z) \cdot F(z) \qquad f(0) + \sum_{n\geq 1} (f_n - f_{n-1}) \cdot z^n \qquad f_0,\ f_1 - f_0, \ldots, f_n - f_{n-1}, \ldots$$

Cuadro A.2: Operaciones básicas con Funciones Generadoras

El procedimiento general para hallar la función generadora $F(z)$ de una secuencia $f(n)0$ es el de primero definir $F(z)$, esto es:

$$F(z) = \sum_{n \geq 0} f(n) \cdot z^n.$$

A partir de aquí, hay varias maneras de proceder:

- Una posible opción es la de modificar esta expresión de $F(z)$ utilizando las propiedades de las OGF para así llegar a las OGF básicas, de modo que $F(z)$ finalmente quede en función de estas.
- También es posible empezar con una OGF básica e ir aplicando distintas propiedades a estas hasta llegar a la expresión de la sumatoria de $F(z)$.

Ejemplos

Ejemplo 1 Dada la secuencia $2^{n+1}2$, halle su función generadora.

Este ejemplo lo resolveremos usando una de las funciones generadoras básicas y aplicando propiedades sucesivamente hasta llegar a la expresión buscada.

Lo primero es definir lo siguiente:

Dada la secuencia $2^{n+1}2$, definimos $F(z)$ como su función generadora.

$$F(z) = \sum_{n \geq 2} 2^{n+1} \cdot z^n.$$

Empezaremos usando la OGF más básica:

$$\frac{1}{1-z} = \sum_{n \geq 0} z^n.$$

El procedimiento a seguir es aplicar propiedades de las OGF hasta que el lado derecho de la ecuación sea igual al buscado, es decir, la expresión de la sumatoria en $F(z)$.

Primero aplicaremos la propiedad de escala, con $k = 2$:

$$\frac{1}{1-2z} = \sum_{n \geq 0} 2^n \cdot z^n.$$

Ahora, dado que la expresión buscada tiene 2^{n+1}, multiplicamos ambos lados de la expresión por 2.

$$2 \cdot \left(\frac{1}{1-2z} \right) = 2 \cdot \left(\sum_{n \geq 0} 2^n \cdot z^n \right)$$

$$\frac{2}{1-2z} = \sum_{n \geq 0} 2^{n+1} \cdot z^n$$

La expresión de la sumatoria es muy cercana a la buscada, sin embargo, aún falta el índice de la sumatoria en la expresión que buscamos. Por lo tanto, separamos la sumatoria.

Para aclarar lo que se hará, es importante recordar la definición de FGO:

$$\sum_{n \geq 0} f(n) \cdot z^n = f(0) \cdot z^0 + f(1) \cdot z^1 + f(2) \cdot z^2 + \cdots + f(n) \cdot z^n + \cdots$$

y que esto puede ser reescrito de diversas maneras, y en este caso, usando la propiedad asociativa de la suma, tenemos:

$$\sum_{n \geq 0} f(n) \cdot z^n = \big(f(0) + f(1) \cdot z^1 \big) + \big(f(2) \cdot z^2 + f(3) \cdot z^3 + \cdots \big)$$

$$\sum_{n \geq 0} f(n) \cdot z^n = \sum_{0 \leq n \leq 1} f(n) \cdot z^n + \sum_{n \geq 2} f(n) \cdot z^n$$

nos será bastante útil. Dado que ahora reescribimos

$$\frac{2}{1-2z} = \sum_{n \geq 0} 2^{n+1} \cdot z^n$$

como

$$\frac{2}{1-2z} = \sum_{0 \leq n \leq 1} 2^{n+1} \cdot z^n + F(z) \sum_{n \geq 2} 2^{n+1} \cdot z^n.$$

A partir de aquí, ya que tenemos $F(z)$, lo que hacemos es despejar.

$$\frac{2}{1-2z} = \sum_{0 \leq n \leq 1} 2^{n+1} \cdot z^n + F(z)$$

$$F(z) = \frac{2}{1-2z} - \sum_{0 \leq n \leq 1} 2^{n+1} \cdot z^n$$

$$F(z) = \frac{2}{1-2z} - 2^2 \cdot z^1 - 2^1 \cdot z^0$$

$$F(z) = \frac{2}{1-2z} - 4z - 2.$$

De aquí, tenemos que la respuesta es que la función generadora de la secuencia $2^{n+1}2$ es

$$F(z) = \frac{2}{1-2z} - 4z - 2.$$

Ejemplo 2 Dada la siguiente secuencia $f(n)0$, halle su OGF.

$$f_n = \begin{cases} n, & n \geq 4 \\ 1, & 1 \leq n < 4 \\ 0, & n = 0 \end{cases}$$

Primero, definimos $F(z)$ como la función generadora de la secuencia $f(n)$

$$F(z) = \sum_{1 \leq N \leq 3} 1 \cdot z^N + \sum_{n \geq 4} n \cdot z^n + 0 \cdot z^0$$

Este ejemplo lo resolveremos usando, nuevamente, la OGF más básica. Es importante resaltar que se puede empezar a partir de cualquier otra de las OGF, pero utilizamos esta para así mostrar más ejemplos de las operaciones.

$$\frac{1}{1-z} = \sum_{n \geq 0} z^n.$$

Primero procedemos a aplicar la operación de multiplicación por el índice a la función generadora. Por lo que al aplicar la operación siguiendo su definición en

la tabla, obtenemos:

$$\frac{1}{(1-z)^2} = \sum_{n\geq 0} (n+1) \cdot z^n.$$

Aquí podemos bien o aplicar el cambio de variable $n \to n-1$ en la sumatoria o bien, aplicar la operación de desplazar a la derecha [la secuencia]. En todo caso, el resultado es el mismo:

$$\frac{z}{(1-z)^2} = \sum_{n\geq 0} n \cdot z^n \left(= \sum_{n\geq 1} n \cdot z^n \right).$$

Esta expresión ya es más cercana a la que buscamos. Sin embargo, el índice de esta sumatoria inicia en 0, y el buscado inicia en 4, por lo que procedemos a restar estos elementos:

$$\begin{aligned}
\frac{z}{(1-z)^2} &= \sum_{n\geq 0} n \cdot z^n \\
&= \sum_{0\leq n\leq 3} n \cdot z^n + \sum_{n\geq 4} n \cdot z^n \\
\frac{z}{(1-z)^2} - \sum_{0\leq n\leq 3} N \cdot z^n &= \sum_{n\geq 4} n \cdot z^n \\
\frac{z}{(1-z)^2} - 3z^3 - 2z^2 - z &= \sum_{n\geq 4} n \cdot z^n.
\end{aligned}$$

Luego de este paso, vemos que aún falta la otra sumatoria de la definición de $F(z)$ para este ejercicio. Por lo que simplemente sumamos la expresión de ambos lados y obtenemos:

$$\frac{z}{(1-z)^2} - 3z^3 - 2z^2 - z + \sum_{1\leq n\leq 3} 1 \cdot z^n = F(z) \sum_{n\geq 4} n \cdot z^n + \sum_{1\leq n\leq 3} 1 \cdot z^n,$$

y aquí una vez más ya tenemos nuestro $F(z)$.

$$\begin{aligned} F(z) &= \frac{z}{(1-z)^2} - 3z^3 - 2z^2 - z + \sum_{1 \leq n \leq 3} 1 \cdot z^n \\ &= \frac{z}{(1-z)^2} - 3z^3 - 2z^2 - z + z + z^2 + z^3 \\ F(z) &= \frac{z}{(1-z)^2} - 2z^3 - z^2. \end{aligned}$$

Por lo que finalmente la solución a este ejemplo es que la OGF de la secuencia

$$f_n = \begin{cases} n, & n \geq 4 \\ 1, & 1 \leq n < 4 \\ 0, & n = 0 \end{cases}$$

es

$$F(z) = \frac{z}{(1-z)^2} - 2z^3 - z^2.$$

Entregable 01

Problema 1: Funciones Generadoras y Secuencias.

Encuentre una forma cerrada para la función generadora ordinaria, $F_i(z)$, para cada una de las siguientes secuencias $\langle f(n)\rangle_{n\geq n_0}$:

Importante

Para cada problema, excepto el número 2, el equipo tiene que hallar:

- La versión NO recurrente de $f(n)$
- Una de las versiones recurrentes de $f(n)$
- La FGO
- Graficar la NO recurrente y la FGO, 1000 primeros valores

1. [*] Sea $f(n)$ el número de subconjuntos, del conjunto $S = \{1, 2, ..., n\}$, que no contienen dos enteros consecutivos.

(Estudiante Alejandra Landinez Lamadrid - 200161946)

- Versión **NO** recurrente de $f(n)$:
  Utilizamos la fórmula de serie geométrica:

  $\frac{1}{1-x} = \sum_{n=0} x^n$

  $x = -z - z^2$

$$\begin{aligned}\frac{1}{1+z+z^2} &= \sum_{n=0}(-z-z^2)^n \\ &= \sum_{n=0}\binom{-n}{n}(-1)^{nn}\cdot(1+z)^n \\ &= \sum_{n=0}(-1)^n\cdot\binom{2n-1}{n}\cdot z^n(1+z)^n \\ f(n) &= (-1)^n\cdot\binom{2n-1}{n}\end{aligned}$$

- Versión recurrente de $f(n) : f(n-1) + f(n-2)$

- FGO:

$$\begin{aligned} F(z) &= \sum_{n\geq 0} f(n) \cdot z^n \\ &= \sum_{n\geq 0} (f(n-1) + f(n-2)) \cdot z^n \\ &= \sum_{n\geq 0} f(n-1) \cdot z^n + \sum_{n\geq 0} f(n-2) \cdot z^n \\ &= \sum_{n\geq 1} f(n) \cdot z^n + \sum_{n\geq 2} f(n) \cdot z^n \end{aligned}$$

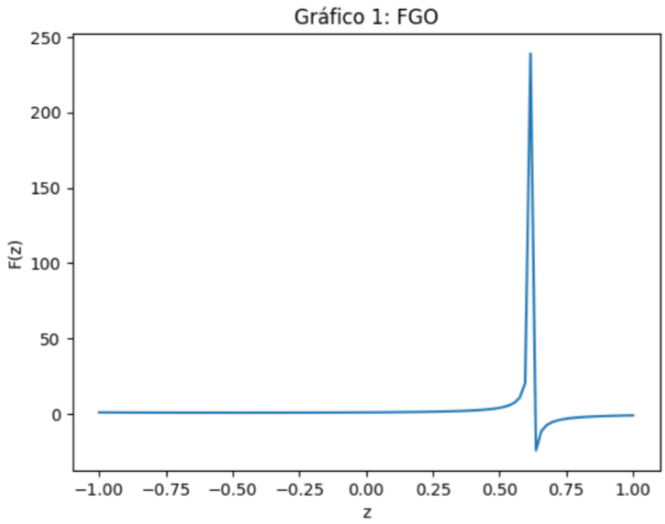
Factorizamos las sumatorias:

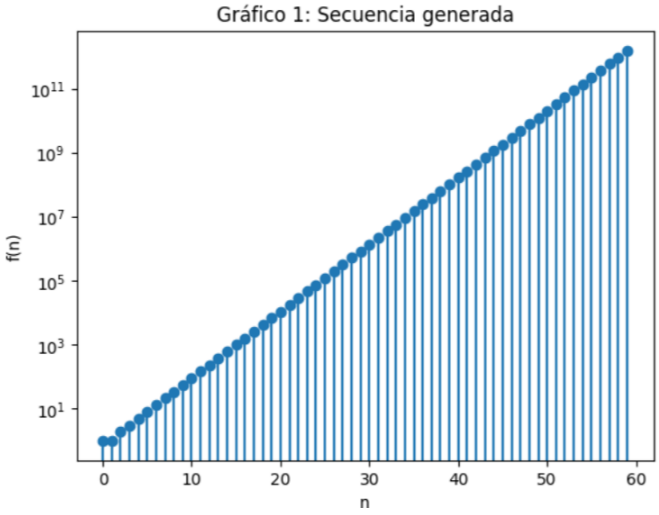
$$\sum_{n\geq 1} f(n) \cdot z^n = z \cdot \sum_{n\geq 1} f(n) \cdot z^{n-1} = z \cdot F(z)$$
$$\sum_{n\geq 2} f(n) \cdot z^n = z^2 \cdot \sum_{n\geq 2} f(n) \cdot z^{n-2} = z^2 \cdot F(z)$$

Reemplazamos los términos factorizados:

$$\begin{aligned} F(z) &= z \cdot F(z) + z^2 \cdot F(z) \\ &= z \cdot F(z) \cdot (1 + z) \\ F(z) &= \frac{1}{1 - z - z^2} \end{aligned}$$

[?]
Gráficas FGO y f(n)





2. [*] Sea $f(n)$ el número de particiones de un conjunto de n elementos.
La solución a este problema fue desarrollada por Eric Temple Bell. El n-ésimo número de Bell está definido por:

$$B(n+1) = \sum_{i=0}^{n} \binom{n}{i} \cdot B(i)$$

Ejemplo, para $B(3) = 5$ las particiones son:

- $\{\ \{1\} \cup \{2\} \cup \{3\}\ \}$
- $\{\ \{1,2\} \cup \{3\}\ \}$
- $\{\ \{1,3\} \cup \{2\}\ \}$
- $\{\ \{2,3\} \cup \{1\}\ \}$

- $\{1, 2, 3\}$

Imprima la secuencia de los números de Bell e ilustre, como en el ejemplo anterior $B(n)$, $n = 0, 1, 2, 3, 4$ y 5

 **(Estudiante Mariana Guerrero Benavides-200173479)**

```
Triángulo de bell
B(0) = 1
B(1) = 1
B(2) = 2
B(3) = 5
B(4) = 15
B(5) = 52

Secuencia generada:  [1, 1, 2, 5, 15, 52]
```

- Para $B(0) = 1$

  Particiones: $\{\{\}\}$

- Para $n = 0$

$$B(0+1) = \binom{0}{0} \cdot B(0) = 1$$

  $B(1) = 1$

  Particiones: $\{\{1\}\}$

- Para $n = 1$

$$B(1+1) = \binom{1}{0} \cdot B(0) + \binom{1}{1} \cdot B(1) =$$

$$B(2) = 1 + 1 = 2$$

  $B(2) = 2$

  Particiones:

  - $\{\ \{1\} \cup \{2\}\}$
  - $\{\ \{1, 2\}\}$

- Para $n = 2$

$$B(2+1) = \binom{2}{0} \cdot B(0) + \binom{2}{1} \cdot B(1) + \binom{2}{2} \cdot B(2) =$$

$$B(3) = B(0) + 2B(1) + B(2) = 1 + 2 \cdot 1 + 2 =$$

$B(3) = 5$ Particiones:

- $\{ \{1\} \cup \{2\} \cup \{3\}\}$
- $\{ \{1, 2, 3\}\}$
- $\{ \{1, 2\} \cup \{3\}\}$
- $\{ \{1, 3\} \cup \{2\}\}$
- $\{ \{2, 3\} \cup \{1\}\}$

- Para $n = 3$

$$B(3+1) = \binom{3}{0} \cdot B(0) + \binom{3}{1} \cdot B(1) + \binom{3}{2} \cdot B(2) + \binom{3}{3} \cdot B(3) =$$

$$B(4) = B(0) + 3B(1) + 3B(2) + B(3) = 1 + 3 \cdot 1 + 3 \cdot 2 + 5 =$$

$B(4) = 15$ Particiones:

- $\{ \{1\} \cup \{2\} \cup \{3\} \cup \{4\}\}$
- $\{ \{1, 2, 3, 4\}\}$
- $\{ \{1\} \cup \{2, 3, 4\}\}$
- $\{ \{2\} \cup \{1, 3, 4\}\}$
- $\{ \{3\} \cup \{1, 2, 4\}\}$
- $\{ \{4\} \cup \{1, 2, 3\}\}$
- $\{ \{1, 2\} \cup \{3, 4\}\}$
- $\{ \{1, 3\} \cup \{2, 4\}\}$
- $\{ \{1, 4\} \cup \{2, 3\}\}$
- $\{ \{1, 2\} \cup \{3\} \cup \{4\}\}$
- $\{ \{1, 3\} \cup \{2\} \cup \{4\}\}$

- $\{\ \{1,4\} \cup \{2\} \cup \{3\}\}$
- $\{\ \{2,3\} \cup \{1\} \cup \{4\}\}$
- $\{\ \{2,4\} \cup \{1\} \cup \{3\}\}$
- $\{\ \{3,4\} \cup \{1\} \cup \{2\}\}$

- Para $n = 4$

$$B(4+1) = \binom{4}{0} \cdot B(0) + \binom{4}{1} \cdot B(1) + \binom{4}{2} \cdot B(2) + \binom{4}{3} \cdot B(3) + \binom{4}{4} \cdot B(4) =$$

$$B(5) = B(0) + 4B(1) + 4B(2) + 4B(3) + B(4) = 4 \cdot 1 + 4 \cdot 1 + 4 \cdot 2 + 4 \cdot 5 + 15 =$$

$B(5) = 52$ Particiones:

- $\{\ \{1\} \cup \{2\} \cup \{3\} \cup \{4\} \cup \{5\}\}$
- $\{\ \{1,2,3,4,5\}\}$
- $\{\ \{1,2\} \cup \{2,3\} \cup \{3,4\} \cup \{4,5\}\}$
- $\{\ \{1,3\} \cup \{2,4\} \cup \{5\}\}$
- $\{\ \{1,3\} \cup \{2,5\} \cup \{4\}\}$
- $\{\ \{1,3\} \cup \{4,5\} \cup \{2\}\}$
- $\{\ \{1,4\} \cup \{2,5\} \cup \{3\}\}$
- $\{\ \{1,4\} \cup \{2,3\} \cup \{5\}\}$
- $\{\ \{1,4\} \cup \{3,5\} \cup \{2\}\}$
- $\{\ \{1,5\} \cup \{3,4\} \cup \{2\}\}$
- $\{\ \{1,5\} \cup \{2,3\} \cup \{4\}\}$
- $\{\ \{1,5\} \cup \{2,4\} \cup \{3\}\}$
- $\{\ \{1,2\} \cup \{3,4\} \cup \{5\}\}$
- $\{\ \{1,2\} \cup \{3,5\} \cup \{4\}\}$
- $\{\ \{1,2\} \cup \{4,5\} \cup \{3\}\}$
- $\{\ \{1\} \cup \{2,3,4,5\}\}$
- $\{\ \{2\} \cup \{1,3,4,5\}\}$
- $\{\ \{3\} \cup \{1,2,4,5\}\}$
- $\{\ \{4\} \cup \{1,2,3,5\}\}$

- $\{\ \{5\} \cup \{1,2,3,4\}\}$
- $\{\ \{1,2\} \cup \{3,4,5\}\}$
- $\{\ \{1,3\} \cup \{2,4,5\}\}$
- $\{\ \{1,4\} \cup \{2,3,5\}\}$
- $\{\ \{1,5\} \cup \{2,3,4\}\}$
- $\{\ \{1\} \cup \{2\} \cup \{3,4,5\}\}$
- $\{\ \{1\} \cup \{3\} \cup \{2,4,5\}\}$
- $\{\ \{1\} \cup \{4\} \cup \{2,3,5\}\}$
- $\{\ \{1\} \cup \{5\} \cup \{2,3,4\}\}$
- $\{\ \{2,3\} \cup \{1,4,5\}\}$
- $\{\ \{2,4\} \cup \{1,3,5\}\}$
- $\{\ \{2,5\} \cup \{1,3,4\}\}$
- $\{\ \{2\} \cup \{3\} \cup \{1,4,5\}\}$
- $\{\ \{2\} \cup \{4\} \cup \{1,3,5\}\}$
- $\{\ \{2\} \cup \{5\} \cup \{1,3,4\}\}$
- $\{\ \{3\} \cup \{4\} \cup \{1,2,5\}\}$
- $\{\ \{3\} \cup \{5\} \cup \{1,2,4\}\}$
- $\{\ \{4\} \cup \{5\} \cup \{1,2,3\}\}$
- $\{\ \{1,2,3\} \cup \{2,3,4\} \cup \{3,4,5\}\}$
- $\{\ \{1,2,3,4\} \cup \{2,3,4,5\}\}$
- $\{\ \{1,2,3\} \cup \{3,4,5\}\}$
- $\{\ \{1,2,3\} \cup \{4,5\}\}$
- $\{\ \{1,2,4\} \cup \{3,5\}\}$
- $\{\ \{1,3,4\} \cup \{2,5\}\}$
- $\{\ \{1\} \cup \{2\} \cup \{3\} \cup \{4,5\}\}$
- $\{\ \{1\} \cup \{2\} \cup \{4\} \cup \{3,5\}\}$
- $\{\ \{1\} \cup \{2\} \cup \{5\} \cup \{3,4\}\}$
- $\{\ \{1\} \cup \{3\} \cup \{4\} \cup \{2,5\}\}$
- $\{\ \{1\} \cup \{3\} \cup \{5\} \cup \{2,4\}\}$
- $\{\ \{1\} \cup \{4\} \cup \{5\} \cup \{2,3\}\}$
- $\{\ \{2\} \cup \{3\} \cup \{4\} \cup \{1,5\}\}$

- $\{\ \{2\} \cup \{3\} \cup \{5\} \cup \{1,4\}\}$
- $\{\ \{2\} \cup \{4\} \cup \{5\} \cup \{1,3\}\}$
- $\{\ \{3\} \cup \{4\} \cup \{5\} \cup \{1,2\}\}$

3. [*] $f(n) = \langle 1, 1, 2, 5, 14, 42, 132, \cdots \rangle_{n \geq 0}$

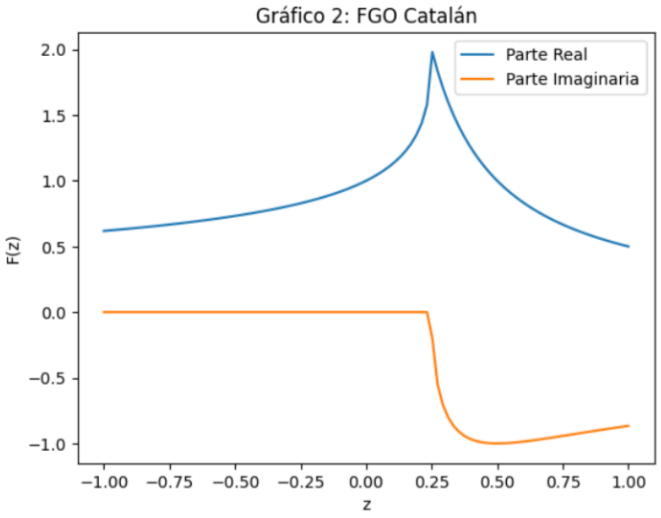
 **(Estudiante Alejandra Landinez Lamadrid - 200161946)**

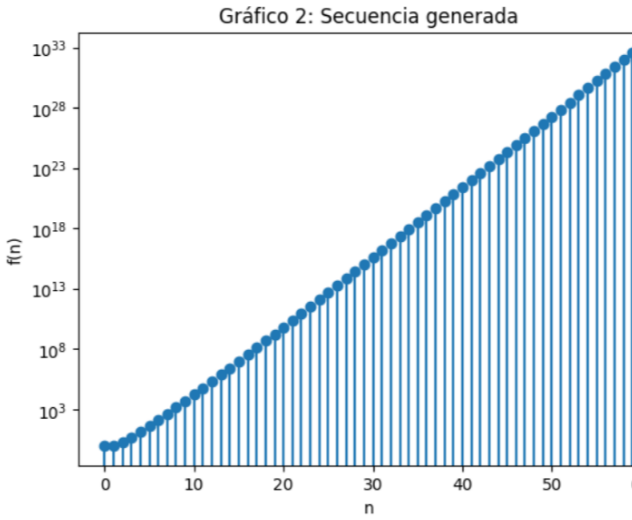
Sea F(z) la función generadora y C_n el n-ésimo número de Catalán, escribimos la ecuación que relaciona a F(z) con C_n y $C_n - 1$ como:

$F(z) = \sum_{n=0} C_n z^n = \sum_{n=0} (\sum_{i=0}^{n-1} C_i C_{n-1-i}) z^n = (\sum_{n=0} C_n zn)^2 - 1 = F(z)^2 - 1$

$$
\begin{aligned}
F(z) &= \sum_{n=0} C_n z^n \\
&= \sum_{n=0} \cdot (\sum_{i=0}^{n-1} C_i C_{n-1-i}) z^n \\
&= (\sum_{n=0} C_n zn)^2 - 1 \\
&= F(z)^2 - 1 \\
&= \frac{1 - \sqrt{1-4z}}{2z} \\
F(z) &= \frac{1}{2z} \cdot \sum_{n \geq 0} \binom{2n}{n} \cdot z^n
\end{aligned}
$$

Gráficas FGO y f(n)





4. [*] $f(n) = \left\langle \frac{1}{2}, -1, -\frac{3}{2}, -2, -2, 0, 8, 32, 96, 256, 640, 1536, \cdots \right\rangle_{n\geq 0}$
   No se realizó.

5. [*] $f(n) = \left\langle \sum_{0\leq k\leq n} k \cdot (-1)^n \cdot (-1)^k \right\rangle_{n\geq 3}$

    (Estudiantes: Mariana Guerrero- 200173479 y Alejandra Landinez Lamadrid - 200161946) [?]

   - Versión recurrente: K es una variable: Si es par, k es divisible por 2, luego, $(-1)^k = 1$. Si es impar, luego $(-1)^k = -1$
     Es posible dividir la sumatoria para k par e impar

     Para k par, $k = 2i$ para algún número entero i:

     $$\sum_{0\leq k\leq n} k \cdot (-1)^n \cdot (-1)^k = 2 \sum_{0\leq i\leq n/2} i$$

     Para k impar, $k = 2i + 1$ para algún número entero i:

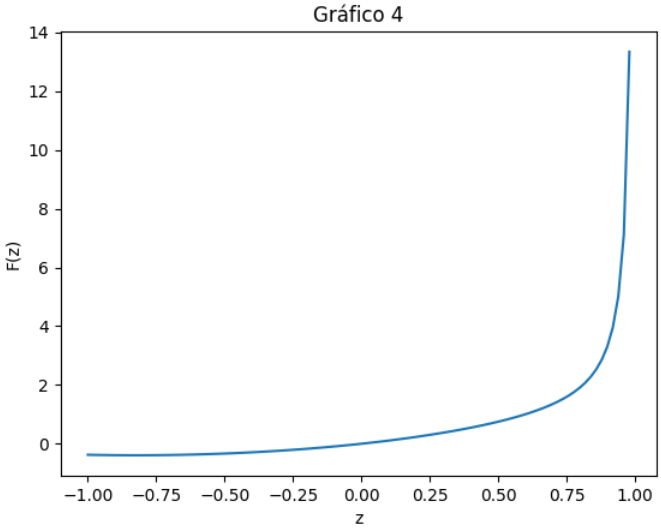
     $$\sum_{0\leq k\leq n} k \cdot (-1)^n \cdot (-1)^k = - \sum_{0\leq i\leq (n-1)/2} (2i + 1)$$

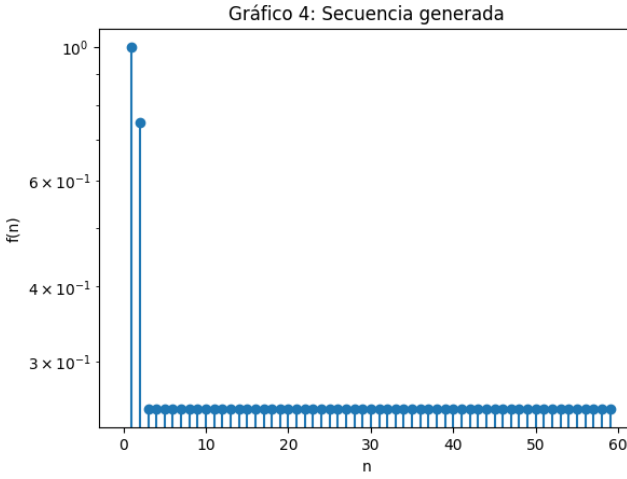
     $$f(n) = 2 \sum_{0\leq i\leq n/2} i - \sum_{0\leq i\leq (n-1)/2} (2i + 1)$$

$f(n) = \frac{n(n+3)}{4}$:

- FGO: $F(z) = \sum_{n\geq 0} f(n) \cdot z^n$
  $F(z) = \sum_{n\geq 0} \left(\frac{n(n+3)}{4}\right) \cdot z^n$
  $F(z) = \frac{1}{4} \sum_{n\geq 0} n(n+3) z^n$
  División en dos partes:
  $F(z) = \frac{1}{4} \left(\sum_{n\geq 0} n^2 \cdot z^n + 3 \sum_{n\geq 0} n\dot{z}^n\right)$
  Se encuentran la seria de potencias para n:
  $\sum_{n\geq 0} n^2 z^n = zF''(z)$ y $\sum_{n\geq 0} nz^n = zF'(z)$ respectivamente
  La FGO es $F(z) = \frac{1}{4}\left(z\left(zF''(z) + 2zF'(z) + \frac{1}{1-z}\right) + 3zF'(z)\right)$

Gráficas FGO y f(n)





6. $f_n = (n-1)(f_{n-1} + f_{n-2}), \quad f_1 = 0, \quad f_2 = 1, \quad n > 2.$


(Estudiante Mariana Guerrero- 200173479)

$f_n = (n-1)f_{n-1} + (n-1)f_{n-2}$
$f_n = (n-1)f_{n-1} + (n-2+1)f_{n-2}$
$f(n) = (n-1)f(n-1) + (n-2)f(n-2) + f(n-2)$
$\sum_{n\geq 3} f_n \cdot z^n = \sum_{n\geq 3}(n-1)f_{n-1} \cdot z^n + \sum_{n\geq 3}(n-2)f_{n-2} \cdot z^n + \sum_{n\geq 3} f_{n-2} \cdot z^n$
$-(f_0 \cdot z^0 + f_1 \cdot z^1 - f_2 \cdot z^2) + \sum_{n\geq 0} f_n \cdot z^n = z^2 \sum_{n\geq 0}(n+1)f_{n+1} \cdot z^n - z^2(1)f_1 +$
$z^3 \sum_{n\geq 0}(n+1)f_{n+1} \cdot z^n + z^2 \sum_{n\geq 0} f_n \cdot z^n$
$\sum_{n\geq 0} f_n \cdot z^n - f_0 z^0 - f_1 z^1 + f_2 z^2 = z^2 F'(z) + z^3 F'(z) + z^2 F(z)$
$F(z) - z^2 = z^2 F'(z) + z^3 F'(z) + z^2 F(z)$
Pasamos $F(z)$ para la izquierda

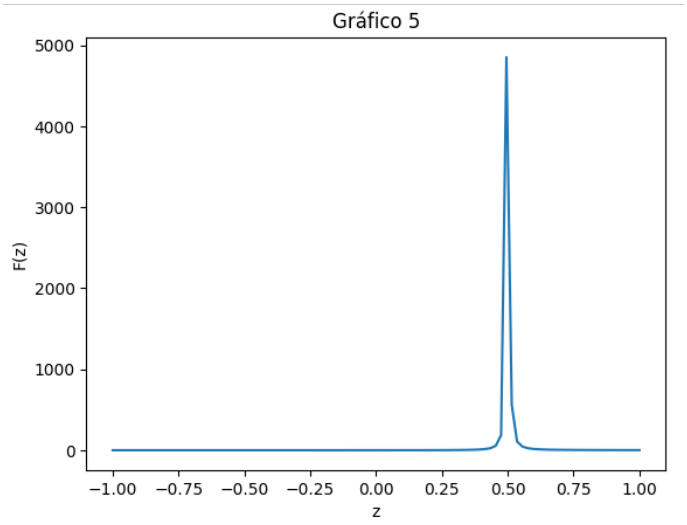
$F(z) - z^2F(z) = z^2F'(z) + z^3F'(z) + z^2$

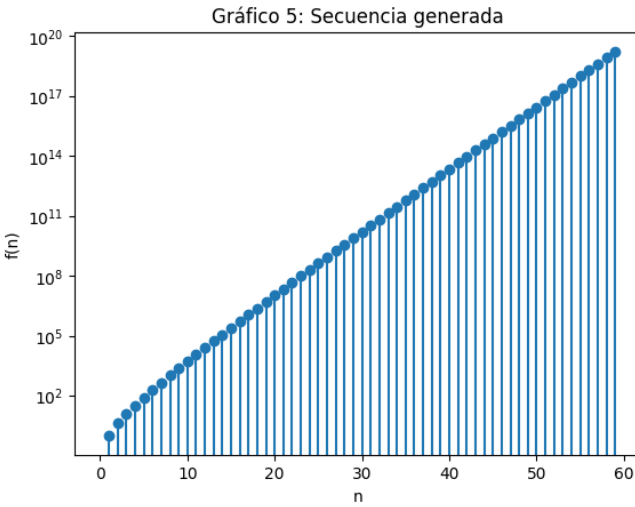
$(1 - z^2)F(z) = z^2F'(z) + z^3F'(z) + z^2$

Despejamos $F(z)$

$F(z) = \dfrac{z^2F'(z) + z^3F'(z) + z^2}{(1 - z^2)}$

Gráficas FGO y f(n)





Para cada una de las siguientes funciones generadoras ordinarias, encuentre una fórmula cerrada para la secuencia que esta determina.

7. [*] $F(z) = \dfrac{z}{1 - 4z + 4z^2}$

(Estudiantes: Alejandra Landinez Lamadrid - 200161946)

$$\begin{aligned} F(z) &= \frac{z}{1 - 4z + 4z^2} \\ &= \frac{z}{4z^2 - 4z + 1} \\ &= \frac{z}{(2z - 1)^2} \end{aligned}$$

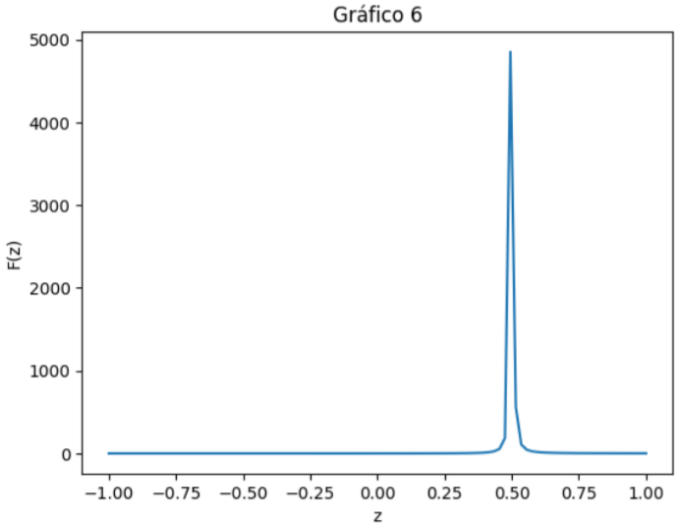
Usando fracciones parciales:

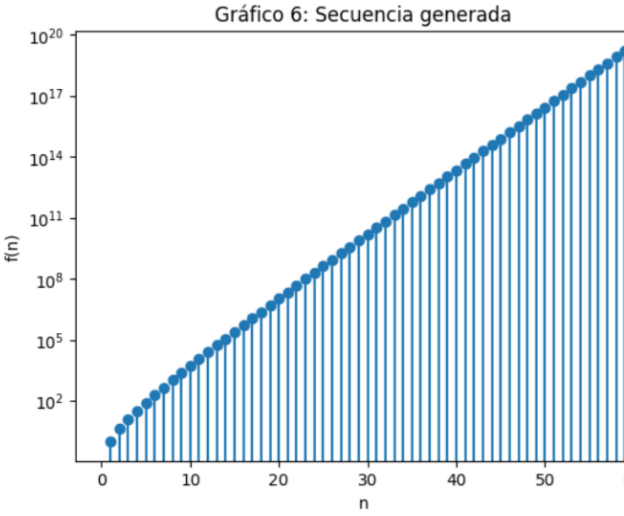
$$\begin{aligned} F(z) &= \frac{A}{2z-1} + \frac{B}{(2z-1)^2} \\ z &= \frac{A}{2z-1} \cdot (2z-1)^2 + \frac{B}{(2z-1)^2} \cdot (2z-1)^2 \\ &= A(2z-1) + B(1) \\ &= A(2z-1) + B \end{aligned}$$

Calculamos A:
$z = A(2z-1)$
$A = \frac{-z}{2z-1}$
Calculamos B en $z = 0$:
$z = B$
$B = 0$
Reemplazando los valores en F(Z):

$$\begin{aligned} F(z) &= \frac{-z}{2z-1} + 0 \\ &= -z \cdot \sum_{n \geq 0} (2z)^n \\ &= -\sum_{n \geq 0} 2^{n+1} \cdot z^{n+1} \\ &= -\sum_{n \geq 1} 2^n \cdot z^n \end{aligned}$$

$f(n) = -2^n, n \geq 1$
Gráficas FGO y f(n)





8. [*] $F(z) = \dfrac{z}{1 - 3z - z^2 + 3z^3}$

 (Estudiante: Alejandra Landinez Lamadrid - 200161946)

$$\begin{aligned} F(z) &= \frac{z}{1 - 3z - z^2 + 3z^3} \\ &= \frac{z}{3z^3 - z^2 - 3z + 1} \\ &= \frac{z}{(z-1)(3z^2 + 2z - 1)} \\ &= \frac{z}{(z-1)(z+1)(3z-1)} \end{aligned}$$

Usando fracciones parciales:

$$F(z) = \frac{A}{z-1} + \frac{B}{z+1} + \frac{C}{3z-1}$$
$$z = A(3z^2 + 2z - 1 + B(z-1)(z+1) + C(z-1)(3z-1)$$

z = 1

$$1 = A(2)(2) + B(0)(2) + C(0)(2)$$
$$1 = 4A$$
$$A = \frac{1}{4}$$

z = -1

$$-1 = A(-4)(0) + B(-2)(0) + C(-2)(-4)$$
$$-1 = -8C$$
$$C = \frac{1}{8}$$

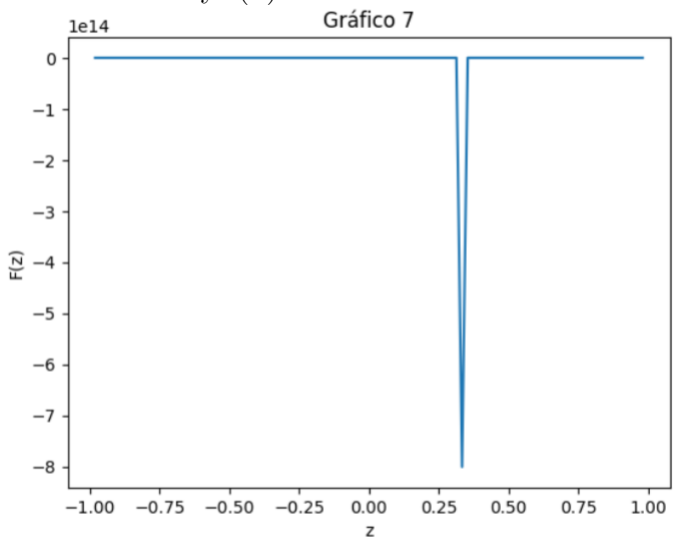
z = 0

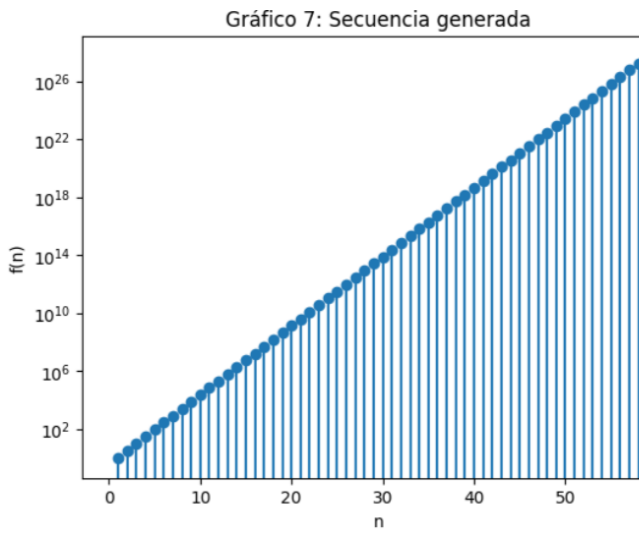
$$0 = A(3(0) - 1)(0 + 1) + B(0 - 1)(0 + 1) + C(0 - 1)(3(0) - 1)$$
$$0 = -A - B + C$$
$$0 = \frac{-1}{4} - B + \frac{1}{8}$$
$$\frac{1}{4} - \frac{1}{8} = -B$$
$$B = \frac{-1}{4} + \frac{1}{8}$$
$$B = -\frac{1}{8}$$

Reemplazamos en la ecuación original:

$$
\begin{aligned}
F(z) &= \frac{1}{4}\cdot\frac{1}{z-1} - \frac{1}{8}\cdot\frac{1}{z+1} + \frac{1}{8}\cdot\frac{1}{3z-1} \\
&= -\frac{1}{4}\cdot\sum_{n\geq 0} z^n - \frac{1}{8}\cdot\sum_{n\geq 0}(-1)^n z^n + \frac{1}{24}\cdot\sum_{n\geq 0}\left(\frac{1}{3}\right)^n \\
&= \sum_{n\geq 0} z^n\left(-\frac{1}{4} - \frac{1}{8}\cdot(-1)^n + \frac{1}{24}\left(\frac{1}{3}\right)^n\right) \\
f(n) &= -\frac{1}{4} - \frac{1}{8}\cdot(-1)^n + \frac{1}{24}\cdot\left(\frac{1}{3}\right)^n
\end{aligned}
$$

Gráficas FGO y f(n)





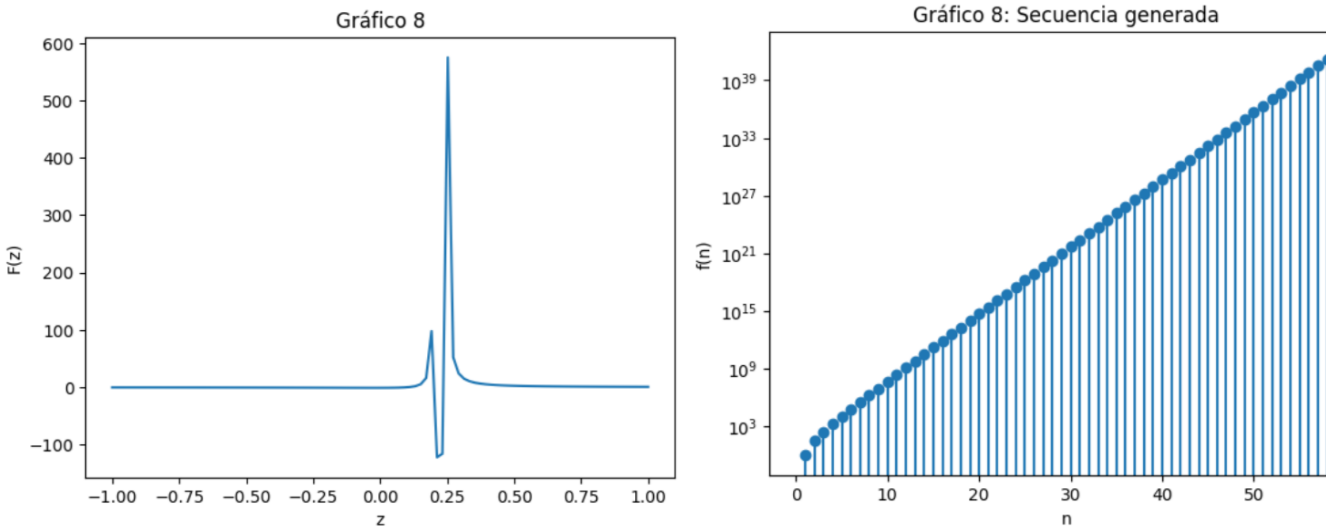
9. [*] $F(z) - z + 1 = 9\cdot z\cdot(F(z)+1) - 20\cdot z^2\cdot F(z)$

$$
\begin{aligned}
F(z) - z + 1 &= 9\cdot z\cdot(F(z)+1) - 20\cdot z^2\cdot F(z) \\
F(z) - 9\cdot z\cdot(F(z)-1) + 20\cdot z^2\cdot F(z) &= z - 1 \\
F(z) - 9zF(z) - 9z + 20\cdot z^2\cdot F(z) &= z - 1 \\
F(z)\cdot(1 - 9z + 20z^2) &= z - 1 + 9z \\
F(z)\cdot(1 - 9z + 20z^2) &= 10z - 1 \\
F(z) &= \frac{10z-1}{(1-5z)(1-4z)}
\end{aligned}
$$

Usando fracciones parciales:

$$
\begin{aligned}
F(z) &= \frac{A}{1-5z} + \frac{B}{1-4z} \\
10z - 1 &= A(1-4z) + B(1-5z) \\
10z - 1 &= A - 4Az + B - 5Bz \\
10z - 1 &= (A+B) + z(-4A-5B) \\
\frac{10z-1}{z} &= A + B - 4A - 5B
\end{aligned}
$$

Gráficas FGO y f(n)

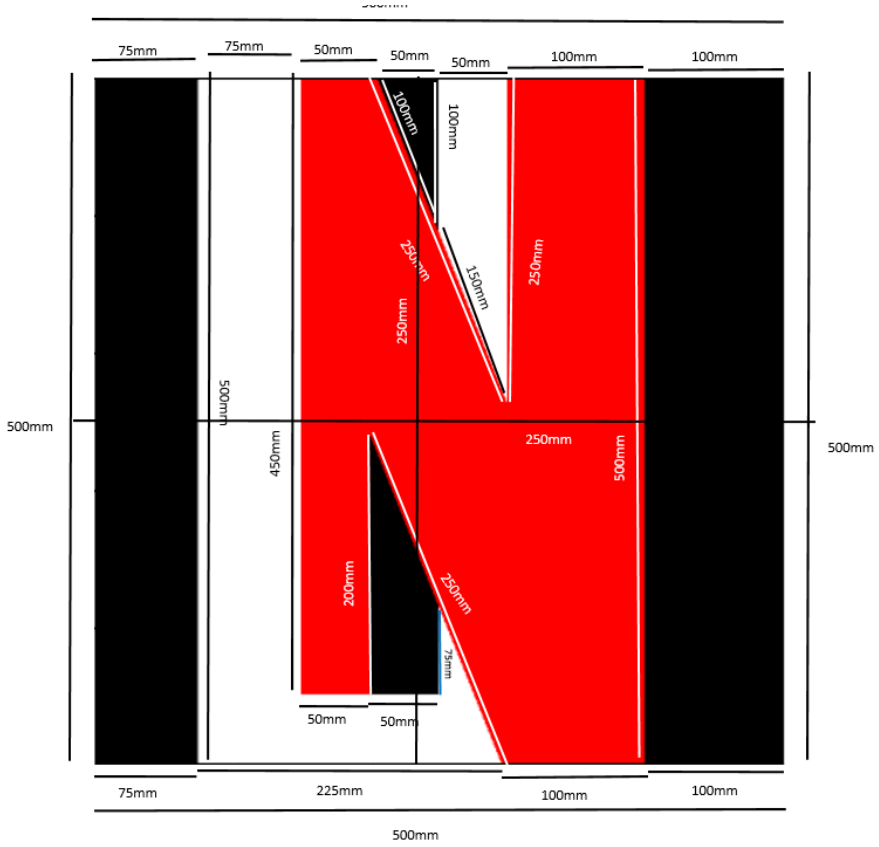


Problema 2: Ejercicios para codificar en python. PC-ED-El Man-202330.ipynb

Código Python en Colab

1. [*] Tensores
   Diseñe e implemente cinco emoji logo

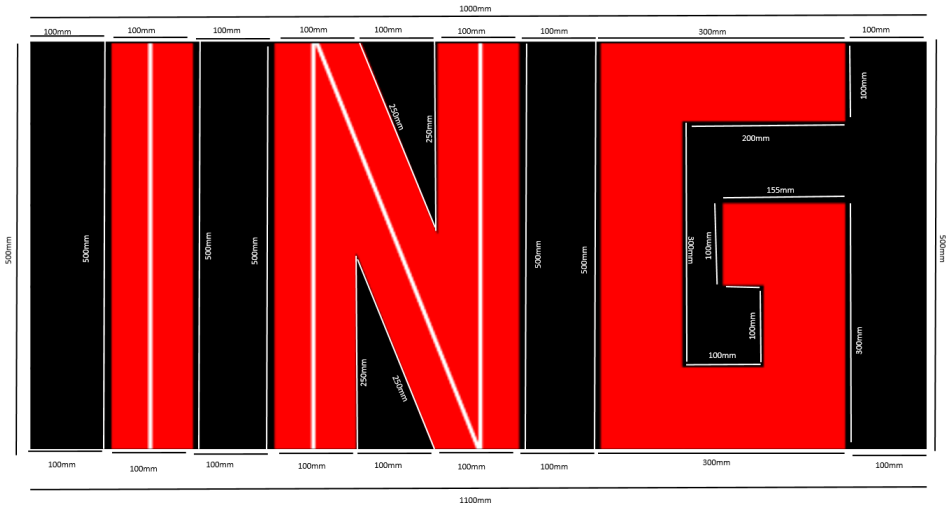
   - Uninorte (Estudiante Juan Camilo Aguirre - 200156480)




```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow

imagen = np.zeros((500, 500, 3), dtype=np.uint8)
imagen.fill(0)
plt.imshow(imagen)

imagen[:,:,:]=(0,0,0)

imagen[:,100:200,:]=(255,0,0)
imagen[:,300:400,:]=(255,0,0)

imagen[:,250:300,:]=(255,255,255)
imagen[:,100:150,:]=(255,255,255)
```

```
imagen[450:500,100:300,:]=(255,255,255)

color = (255, 0, 0)
grosor= 100
puntoUN1 = (150, 0)
puntoUN2 = (350, 500)
cv2.line(imagen, puntoUN1, puntoUN2, color, grosor)
imagen[:,75:150,:]=(255,255,255)

plt.imshow(imagen)
print("La U es blanca y la N es roja")
```

- División de Ingenierías (Estudiante Juan Camilo Aguirre - 200156480)

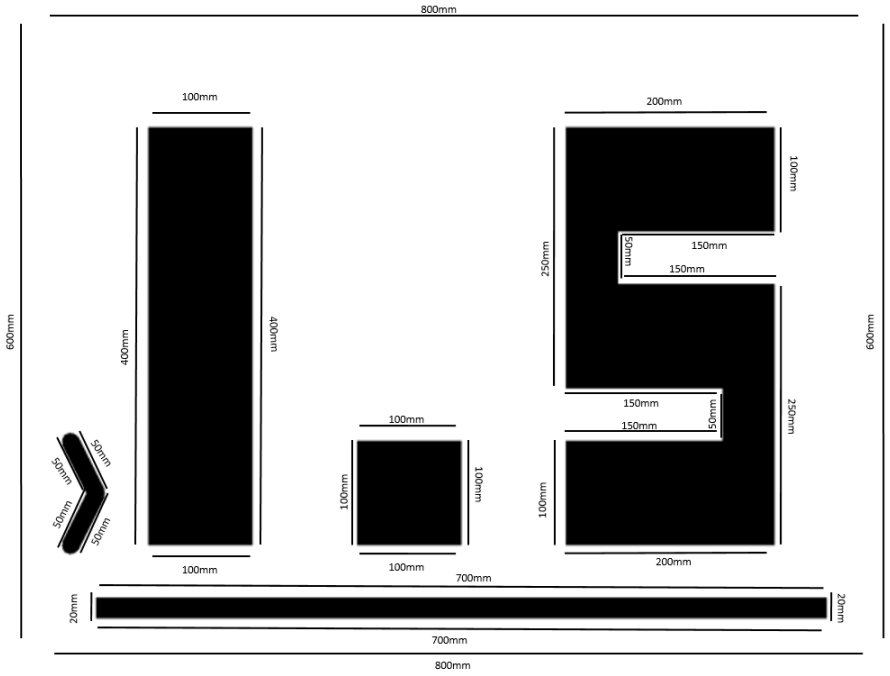



```
#Division de Ingenierias
import cv2
import numpy as np
import matplotlib.pyplot as plt

from google.colab.patches import cv2_imshow

imagen = np.zeros((500,1100, 3), dtype=np.uint8)
imagen.fill(255)
plt.imshow(imagen)
```

```
color_linea = (255, 0, 0)
grosor_linea = 100

imagen[:,:,:]=(0,0,0)

imagen[:,100:200,:]=(255,0,0)

imagen[:,300:400,:]=(255,0,0)
puntoING1=(350,0)
puntoING2=(550,500)
imagen[:,500:600,:]=(255,0,0)
cv2.line(imagen,puntoING1,puntoING2,color_linea,grosor_linea)


imagen[0:100,700:900,:]=(255,0,0)

imagen[400:500,750:1000,:]=(255,0,0)
imagen[0:100,750:1000,:]=(255,0,0)

imagen[:,700:800,:]=(255,0,0)

imagen[200:500,900:1000,:]=(255,0,0)

imagen[200:300,850:900,:]=(255,0,0)

imagen[:,145:150,:]=(255,255,255)
imagen[:,145:150,:]=(255,255,255)

imagen[:,345:350,:]=(255,255,255)

imagen[:,550:555,:]=(255,255,255)

puntoING3=(350,0)
puntoING4=(550,500)
```

```
color = (255, 255, 255)
grosor = 3
cv2.line(imagen,puntoING3,puntoING4,color,grosor)

plt.imshow(imagen)
```

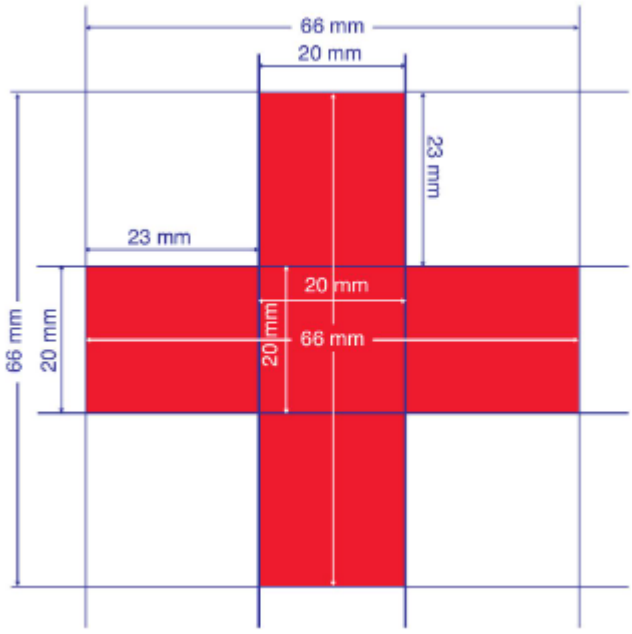
- Departamento de Ingeniería de Sistemas (Estudiante Juan Camilo Aguirre - 200156480)




```
#Logo del departamento de Ingenieria de Sistemas

import cv2
import numpy as np
import matplotlib.pyplot as plt

imagen = np.zeros((600,800, 3), dtype=np.uint8)
imagen.fill(255)
plt.imshow(imagen)

imagen[100:500,100:200,:]=(0,0,0)
imagen[400:500,300:400,:]=(0,0,0)
```

```
imagen[400:500,500:700,:]=(0,0,0)
imagen[100:200,500:700,:]=(0,0,0)
imagen[200:300,500:550,:]=(0,0,0)
imagen[200:300,500:550,:]=(0,0,0)
imagen[250:350,500:700,:]=(0,0,0)
imagen[300:400,650:700,:]=(0,0,0)
imagen[550:570,50:750,:]=(0,0,0)

puntoIS1=(25,400)
puntoIS2=(25,500)
puntoIS3=(50,450)
color=(0,0,0)
grosor=15

cv2.line(imagen,puntoIS1,puntoIS3,color,grosor)
cv2.line(imagen,puntoIS2,puntoIS3,color,grosor)

plt.imshow(imagen)
```

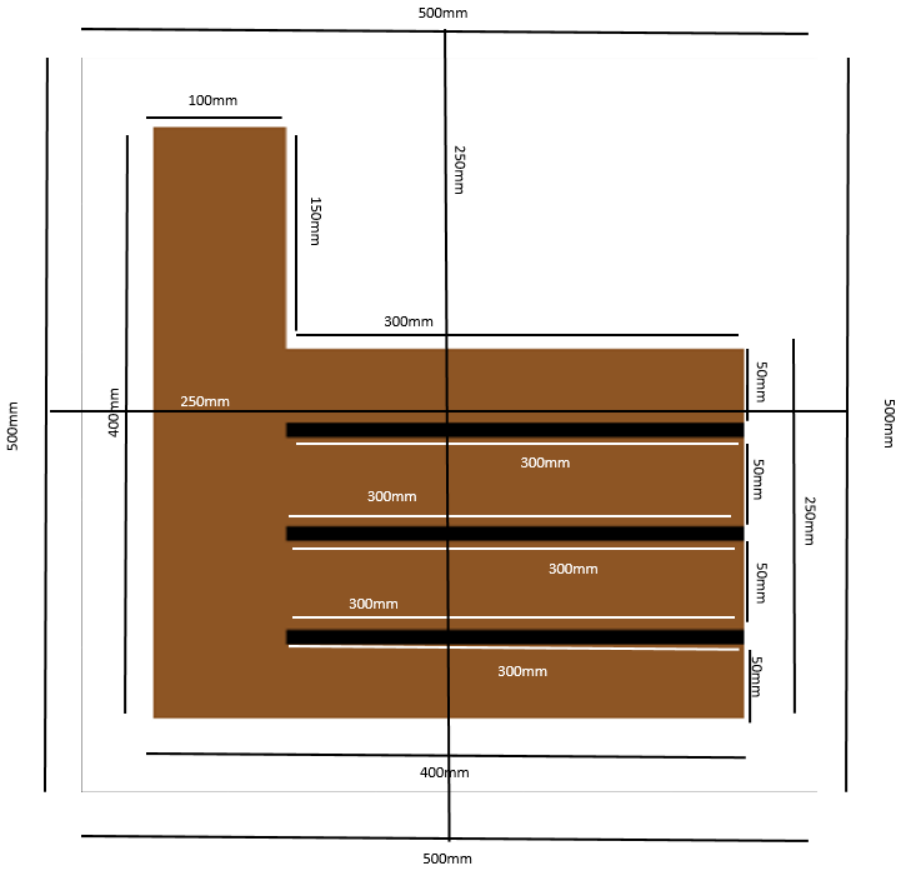
- Cruz roja internacional (Estudiante Mariana Guerrero Benavides-200173479)




```
import numpy as np
```

```
import matplotlib.pyplot as plt

# medidas estandar del logo Cruz Roja (en mm)
ancho = 66  ; alto = 66
# espesor del brazo de la cruz (20mm)
b = int(ancho / 3)
# espacio alrededor de la cruz(6 mm)
margen = 3
# area total del gr fico
ancho_total = ancho + 2 * margen ; alto_total = alto + 2 * marg
# tensor con pixeles blancos
cruz_roja = np.ones((alto_total, ancho_total, 3), dtype=np.uint
cruz_roja=cruz_roja[:,:,::-1]
rojo = (255, 0, 0)
# fila central
fila = alto_total // 2
cruz_roja[fila - b // 2 : fila + b // 2, margen : ancho_total -
# columna central
columna = ancho_total // 2
cruz_roja[margen : alto_total - margen, columna - b // 2 : colur
plt.imshow(cruz_roja)
plt.show()
```

- Excelente (Estudiante Juan Camilo Aguirre - 200156480)




```
#Excelente
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

imagen = np.zeros((500, 500, 3), dtype=np.uint8)
imagen.fill(255)
plt.imshow(imagen)

imagen[200:450, :, :]=[141,85,36] #piel
imagen[:,50:140,:]=[141,85,36] #piel
imagen[450:500, :, :]=[255,255,255] #blanco
imagen[0:50, :, :]=[255,255,255] #blanco
imagen[:,0:50,:]=[255,255,255] #blanco
imagen[:,450:500,:]=[255,255,255] #blanco
```

```
imagen[250:260, :, :]=[0,0,0] #linea negra
imagen[320:330, :, :]=[0,0,0] #linea negra
imagen[390:400, :, :]=[0,0,0] #linea negra
imagen[:,0:50,:]=[255,255,255] #blanco
imagen[:,450:500,:]=[255,255,255] #blanco
imagen[:,50:140,:]=[141,85,36] #piel
imagen[450:500, :, :]=[255,255,255] #blanco
imagen[0:50, :, :]=[255,255,255] #blanco

plt.imshow(imagen)
```

Anexe la plantilla correspondiente, con las dimensiones establecidas

 (Mariana Guerrero Benavides y Juan Camilo Aguirre (200173479 y 200156480 respectivamente))

2. [*] Imprima el triángulo de *Bell* y las particiones correspondientes para $0 \leq n < 10$. Ejemplo, para $B(3) = 5$ las particiones son:

   - $\{ \{1\} \cup \{2\} \cup \{3\} \}$
   - $\{ \{1,2\} \cup \{3\} \}$
   - $\{ \{1,3\} \cup \{2\} \}$
   - $\{ \{2,3\} \cup \{1\} \}$
   - $\{1,2,3\}$

   [**?**]

    (Estudiante Mariana Guerrero Benavides-200173479)

```
def triangulo_bell(n):
    # inicializamos el triangulo con B(0) = 1
    triangulo = [[0] * (n + 1) for _ in range(n + 1)]
    triangulo[0][0] = 1

    # calculamos el triangulo de Bell utilizando recurrencia
    for i in range(1, n + 1):
```

```
            triangulo[i][0] = triangulo[i - 1][i - 1]
            for j in range(1, i + 1):
                triangulo[i][j] = triangulo[i - 1][j - 1] + triangulo[i][j
        return triangulo

    def particiones_conj(n):
        # se calcula el triangulo de bell
        triangulo = triangulo_bell(n)

        # generamos particiones correspondientes para 0 <= n < 10
        particiones = []
        for k in range(1, n + 1):
            particion_k = []
            for i in range(n + 1):
                if triangulo[i][k] != 0:
                    particion_k.append(set(range(i - k + 1, i + 1)))
            particiones.append(particion_k)
        return particiones

    # imprimimos el triangulo de bell
    print(f"Triangulo de Bell para 0 <= n < 10:")
    for row in triangulo_bell(9):
        row_values = [str(value) for value in row if value != 0]
        print(" ".join(row_values))
    print()
    # imprimimos particiones correspondientes para 0 <= n < 10
    for n in range(10):
        valor = triangulo_bell(n)[n][0]
        particiones = particiones_conj(n)
        print(f"B({n}) = {valor}, las particiones correspondientes son:")
        for i, particion in enumerate(particiones):
            print(f"- {particion}")
    print()
```

3. [*] Gráficas de las secuencias y FGO de todos los ejercicios de Funciones

Generadoras y Secuencias..
Entrada: $F(z), -1 < z < 1$ y $f(n), n >= 0$
Salida: Las gráficas correspondientes

[?]

(Estudiante Mariana Guerrero Benavides-200173479)

```
    from numbers import Real
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from IPython.display import display, Math

z = sp.symbols('z') # definimos z como una variable simb lica

# valores de z en intervalo (-1, 1)
Z = np.linspace(-1, 1, 100)



# definimos las FGO
def fgo_1(z):
    return 1 / (1 - z- z**2)

def fgo_2(z):
    return (1 - (sp.sqrt(1-4*z)))/(2*z)
'''
def fgo_3(z):
    return
'''
def fgo_4(z):
  Fz = z # ajustar segun necesidad
  F_doble_prima = Fz.diff(z, 2) # derivada de Fz
  F_prima = Fz.diff(z) # segunda derivada de Fz

  return (1/4) * (z * (z * F_doble_prima + 2 * z * F_prima + 1 / (1 - z
```

```
def fgo_5(z):
    return z / (1 - 4 * z + 4 * z**2)

def fgo_6(z):
  return z/(1-4*z+ 4*z**2)

def fgo_7(z):
  return z/(1-3*z-z**2+3*z**3)

def fgo_8(z):
  return ((10*z-1)/((1-5*z)*(1-4*z)))


n=60
# Crea y muestra el gr fico primer FGO
plt.plot(Z, [fgo_1(z) for z in Z])
plt.title('Gr fico 1: FGO')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math(sp.latex(fgo_1(z)))) # imprimimos la FGO en L tex
# calculamos la secuencia generada a partir de la FGO
Fz_1 = sp.Poly(fgo_1(z).series(x=z, x0=0,n=n).removeO())
fn_1=Fz_1.all_coeffs()
print('Secuencia generada\n',fn_1[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_1 = np.arange(len(fn_1))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_1, fn_1[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 1: Secuencia generada')
plt.show()

# Crea y muestra el gr fico segunda FGO
```

```
# Calcula la funci n fgo_2 para los valores de Z
valores = [fgo_2(z_val) for z_val in Z]
# Separa las partes real e imaginaria de los resultados
real = [sp.re(val) for val in valores]
imaginario = [sp.im(val) for val in valores]
# Crea y muestra el gr fico de la parte real e imaginaria
plt.plot(Z, real, label='Parte Real'), plt.plot(Z, imaginario, label='P
plt.title('Gr fico 2: FGO Catal n')
plt.xlabel('z'), plt.ylabel('F(z)')
plt.legend(), plt.show()
display(Math("FGO: "+ sp.latex(fgo_2(z)))) # imprimimos la FGO en L te
# calculamos la secuencia generada a partir de la FGO
Fz_2 = sp.Poly(fgo_2(z).series(x=z, x0=0,n=n).removeO())
fn_2=Fz_2.all_coeffs()
print('Secuencia generada\n',fn_2[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_2 = np.arange(len(fn_2))
plt.yscale('log')  # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_2, fn_2[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 2: Secuencia generada')
plt.show()
'''
# Crea y muestra el gr fico tercera FGO
plt.plot(Z, [fgo_3(z) for z in Z])
plt.title('Gr fico 3')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO: "+ sp.latex(fgo_3(z)))) # imprimimos la FGO en L te
# calculamos la secuencia generada a partir de la FGO
Fz_3 = sp.Poly(fgo_3(z).series(x=z, x0=0,n=n).removeO())
fn_3=Fz_3.all_coeffs()
print('Secuencia generada\n',fn_3[::-1]) # imprimimos la secuencia gene
```

```
# rango de valores de n
n_3 = np.arange(len(fn_3))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_3, fn_3[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 3: Secuencia generada')
plt.show()
'''
# Crea y muestra el gr fico cuarta FGO
fun_num1 = sp.lambdify(z, fgo_4(z), 'numpy') # convertimos la funci n
val_func1 = fun_num1(Z)
plt.plot(Z, val_func1)
plt.title('Gr fico 4')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO:"+ sp.latex(fgo_4(z)))) # imprimimos la FGO en L te
# calculamos la secuencia generada a partir de la FGO
Fz_4 = sp.Poly(fgo_4(z).series(x=z, x0=0,n=n).removeO())
fn_4=Fz_4.all_coeffs()
print('Secuencia generada\n',fn_4[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_4 = np.arange(len(fn_4))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_4, fn_4[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 4: Secuencia generada')
plt.show()


# Crea y muestra el gr fico quinta FGO
plt.plot(Z, [fgo_5(z) for z in Z])
plt.title('Gr fico 5')
```

```
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO:-"+sp.latex(fgo_5(z)))) # imprimimos la FGO en L tex
# calculamos la secuencia generada a partir de la FGO
Fz_5 = sp.Poly(fgo_5(z).series(x=z, x0=0,n=n).removeO())
fn_5=Fz_5.all_coeffs()
print('Secuencia generada\n',fn_5[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_5 = np.arange(len(fn_5))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_5, fn_5[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 5: Secuencia generada')
plt.show()

# Crea y muestra el gr fico sexta FGO
plt.plot(Z, [fgo_6(z) for z in Z])
plt.title('Gr fico 6')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO:-"+sp.latex(fgo_6(z)))) # imprimimos la FGO en L tex
# calculamos la secuencia generada a partir de la FGO
Fz_6 = sp.Poly(fgo_6(z).series(x=z, x0=0,n=n).removeO())
fn_6=Fz_6.all_coeffs()
print('Secuencia generada\n',fn_6[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_6 = np.arange(len(fn_6))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_6, fn_6[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 6: Secuencia generada')
plt.show()
```

```
# Crea y muestra el gr fico septima FGO
plt.plot(Z, [fgo_7(z) for z in Z])
plt.title('Gr fico 7')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO:"+ sp.latex(fgo_7(z)))) # imprimimos la FGO en L te
# calculamos la secuencia generada a partir de la FGO
Fz_7 = sp.Poly(fgo_7(z).series(x=z, x0=0,n=n).removeO())
fn_7=Fz_7.all_coeffs()
print('Secuencia generada\n',fn_7[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_7 = np.arange(len(fn_7))
plt.yscale('log') # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_7, fn_7[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 7: Secuencia generada')
plt.show()

# Crea y muestra el gr fico octava FGO
fun_num = sp.lambdify(z, fgo_8(z), 'numpy') # convertimos la funci n e
val_func = fun_num(Z)
plt.plot(Z, val_func)
plt.title('Gr fico 8')
plt.xlabel('z') , plt.ylabel('F(z)')
plt.show()
display(Math("FGO:"+ sp.latex(fgo_8(z)))) # imprimimos la FGO en L te
# calculamos la secuencia generada a partir de la FGO
Fz_8 = sp.Poly(fgo_8(z).series(x=z, x0=0,n=n).removeO())
fn_8=Fz_8.all_coeffs()
print('Secuencia generada\n',fn_8[::-1]) # imprimimos la secuencia gene
# rango de valores de n
n_8 = np.arange(len(fn_8))
```

```
plt.yscale('log')  # Cambia la escala a logar tmica en el eje y
# Crea y muestra gr fico de la secuencia generada
plt.stem(n_8, fn_8[::-1])
plt.xlabel('n') , plt.ylabel('f(n)')
plt.title('Gr fico 8: Secuencia generada')
plt.show()
```

4. [*] Gráficas 2D en coordenadas

   - Cartesianas
   - Polares

   Gráficas 3D en coordenadas

   - Rectangulares
   - Esféricas
   - Cilíndricas

   Diseñe los menus respectivos.
   Ayuda: utilice la siguiente referencia Coordenadas en 3D

   [?]

    (Estudiante Mariana Guerrero Benavides-200173479)

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Funci n que calcula f(x, y, z)
def funcion(x, y, z):
    return 3* np.sin(x) + 5* np.cos(y) + 2  # Cambiar la funci n seg

def coord_cartesianas(funcion):
    # grillas de coordenadas
    x = np.linspace(-2*np.pi, 2*np.pi, 100)
    # evaluamos la funci n en cada valor de x
```

```
    y = []
    for val in x:
        y.append(funcion(val, 0, 0))

    plt.plot(x,y)
    plt.title('Coordenadas Cartesianas 2D')
    plt.xlabel('X'); plt.ylabel('Y')
    plt.show()

def coord_polares(funcion):
    # grillas de coordenadas
    theta = np.linspace(0, 2*np.pi, 100)
    # evaluamos la funci n en cada valor de theta
    r = []
    for val in theta:
      r.append(funcion(1, val, 0))

    plt.polar(theta, r)
    plt.title('Coordenadas Polares 2D')
    plt.show()

def coord_rectangulares(funcion):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    # grillas de coordenadas
    x, y = np.linspace(-2, 2, 100), np.linspace(-2, 2, 100)
    X, Y = np.meshgrid(x, y)
    # evaluamos la funci n en x y y
    Z = []
    for x_row, y_row in zip(X, Y):
        row = []
        for x_val, y_val in zip(x_row, y_row):
            value = funcion(x_val, y_val, 0)
            row.append(value)
```

```
            Z.append(row)
        Z = np.array(Z)

        ax.plot_surface(X, Y, Z)
        plt.title('Coordenadas Rectangulares 3D')
        plt.show()

    def coord_esfericas(funcion):
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        # grillas de coordenadas
        phi = np.linspace(0, 2*np.pi, 100)
        theta = np.linspace(0, np.pi, 100)
        PHI, THETA = np.meshgrid(phi, theta)
        # evaluamos la funci n en phi y theta
        R = []
        for phi_row, theta_row in zip(PHI, THETA):
            row = []
            for phi_val, theta_val in zip(phi_row, theta_row):
                value = funcion(1, theta_val, phi_val)
                row.append(value)
            R.append(row)
        R = np.array(R)
        # transformaci n coordenadas esf ricas a rectangulares
        X = R * np.sin(THETA) * np.cos(PHI)
        Y = R * np.sin(THETA) * np.sin(PHI)
        Z = R * np.cos(THETA)

        ax.plot_surface(X, Y, Z)
        plt.title('Coordenadas Esf ricas 3D')
        plt.show()

    def coord_cilindricas(funcion):
        fig = plt.figure()
```

```
    ax = fig.add_subplot(111, projection='3d')
    # grillas de coordenadas
    theta = np.linspace(0, 2*np.pi, 100)
    z = np.linspace(-2, 2, 100)
    Theta, Z = np.meshgrid(theta, z)
    # evaluamos la funci n en theta y z
    R = []
    for theta_row, z_row in zip(Theta, Z):
        row = []
        for theta_val, z_val in zip(theta_row, z_row):
            value = funcion(1, theta_val, z_val)
            row.append(value)
        R.append(row)
    R = np.array(R)
    # transformaci n coordenadas cil ndricas a rectangulares
    X = R * np.cos(Theta)
    Y = R * np.sin(Theta)

    ax.plot_surface(X, Y, Z)
    plt.title('Coordenadas Cil ndricas 3D')
    plt.show()


# Men  principal
while True:
    print("** Men  principal **")
    print("Seleccione dimensi n de la gr fica:")
    print("1. Coordenadas en 2D")
    print("2. Coordenadas en 3D")
    print("3. Salir")
    opcion = int(input("Escoja opci n: "))
    print("- - - - - - - - - -")

    if opcion == 1:
```

```
    print("Coordenadas en 2D")
    while True:
      print("Seleccione el tipo de gr fica:")
      print("1. Coordenadas Cartesianas 2D")
      print("2. Coordenadas Polares 2D")
      print("3. Atras")
      print("----------")
      op = int(input("Escoja opci n:"))
      if op == 1:
          coord_cartesianas(funcion)
      elif op == 2:
          coord_polares(funcion)
      elif op == 3:
        break
      else:
          print("Opci n no v lida. Por favor, ingrese una opci n v

  elif opcion == 2:
    print("Coordenadas en 3D")
    while True:
      print("Seleccione el tipo de gr fica:")
      print("1. Coordenadas Rectangulares 3D")
      print("2. Coordenadas Esf ricas 3D")
      print("3. Coordenadas Cil ndricas 3D")
      print("4. Atras")
      print("----------")
      op = int(input("Escoja opci n:"))
      if op == 1:
          coord_rectangulares(funcion)
      elif op == 2:
          coord_esfericas(funcion)
      elif op== 3:
          coord_cilindricas(funcion)
      elif op == 4:
```

```
            break
        else:
            print("Opci n no v lida. Por favor, ingrese una opci n v

    elif opcion == 3:
        print("Salida con  xito ")
        break
    else:
        print("Opci n no v lida. Por favor, ingrese una opci n v li
```

5. [*] Solución de RRLNHCCC.
   Entrada: La expresión cerrada de la secuencia recurrente RRLNHCCC, $f(n) = f^H(n) + f^p(n)$, más condiciones iniciales.
   Salida: La expresión cerrada para secuencia NO recurrente $f(n)$ y los 25 primeros términos generados por ambas.
    (Estudiante Mariana Guerrero Benavides-200173479)

```
import sympy as sp

# Definimos variables simb licas
n = sp.symbols('n')
x = sp.symbols('x')

# Definimos valor inicial
i0 = 1.0  # Ajustar seg n necesidades

# Encuentra ra ces del polinomio
def calcular_raiz(coeficientes, i0):
  # Creamos el polinomio a partir de los coeficientes
  polinomio = sum(coef * x**i for i, coef in enumerate(coeficientes[::-
  raices = []
  for _ in range(len(coeficientes)):
    raiz = sp.solve(polinomio, x)
    if raiz:
      raices.append(raiz)
```

```
    coeficientes.pop(0)
    polinomio = sum(coef * x**i for i, coef in enumerate(coeficientes[:
  return (raices)

# Encuentra la secuencia no recurrente
def calcular_no_recurrente(raices):
  expresiones = []

  # Se calcula una expresi n para las ra ces
  for valor in raices:
    expresion = valor[0]**n
    expresiones.append(expresion)
  solucion = sum(expresiones)

  return(solucion)

# Encontramos t rminos secuencia no recurrente para 25 t rminos
def calcular_secuencia(secuencia):
  terminos=[secuencia.subs(n, i0+i) for i in range(25)]
  return (terminos)

# Pedimos al usuario que ingrese los coeficientes de la funci n
coef_str = input("Ingrese coeficientes de f(n): [cn1 cn2 ... cnk]=")
coef_ = [int(x) for x in coef_str.split()]

# Pedimos al usuario que ingrese las condiciones iniciales
condiciones_str = input("Ingrese las condiciones iniciales: [a0 a1 ...
ci = [int(x) for x in condiciones_str.split()]

raices = calcular_raiz(coef_, i0)
print('Ra ces:\n', raices)
if not raices:
  print("No hallamos ra ces reales")
else:
```

```
sol = calcular_no_recurrente(raices)
print('RRLNHCCC:\n', sol)

terminos = calcular_secuencia(sol)
print('Secuencia no recurrente:\n')
secuencial = [termino.evalf() for termino in terminos]
print(secuencial)
```

6. Dado un sitio web estático, implemente *web scraping* para obtener el texto plano del mismo. A partir de este, implemente un modelo de cadenas de Markov para generar un texto ficticio y retórnelo al usuario.

 (Estudiante Juan Camilo Aguirre- (200156480))

[?]

```
!pip install markovify

import requests
from bs4 import BeautifulSoup
import markovify


print("Ingrese el URL de la página que desea generar texto:")
url_sitio_web = str(input(""))
print("————————————————————————————————————————————————————")



#1 (Funcion para extraer el texto del URL establecido {WebScraping})
def web_scraping(url):
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        texto = ' '.join([p.get_text() for p in soup.find_all('p')])
        return texto
```

```
#2 (Convertir el texto anterior a una lista donde se escojan terminos c
def generar_texto_markov(texto_plano, longitud_deseada=200):
        modelo_markov = markovify.Text(texto_plano)
        texto_ficticio = ""
        while len(texto_ficticio.split()) < longitud_deseada:
            oracion = modelo_markov.make_sentence()
            if oracion:
                texto_ficticio += " " + oracion
        return texto_ficticio.strip()


texto = web_scraping(url_sitio_web)

if texto:
   generado = generar_texto_markov(texto, longitud_deseada=200)
   if generado:
    print("Texto generado:")
    print(generado)
```

Entrega 02: Método Simbólico, Análisis Combinatorio, Titulares de Fake News

Problema 1: Método Simbólico

En el análisis combinatorio existe un método que permite la definición formal de estructuras combinatorias a partir de sus propiedades. Este método se conoce como el método simbólico. Este método provee definiciones básicas que permiten definir estructuras combinatorias, y que a través de esta definición y el *teorema de transferencia*, se permite obtener la respectiva **función generadora** y así, una fórmula para las estructuras.

Este método está basado primeramente en la definición de *clases combinatorias*. Por ejemplo, $\mathcal{B}$ se podría definir como la clase combinatoria de todas las cadenas binarias. Adicional a esta definición, se requiere de la definición de átomo, que corresponde a los elementos que conforman la clase combinatoria. En el caso de las cadenas binarias, está conformado por los caracteres 0 y 1, y en el método simbólico, serían definidos como z_0 y z_1. Por último está la definición del elemento nulo o vacío, ϵ.

Definición	Método Simbólico	Función Generadora
Clase Combinatoria	$\mathcal{A}$	$A(z)$
Átomo	$z_0,\ z_1,\ z_2,\ \ldots$	z
Vacío	ϵ	1
Unión	$\mathcal{A} + \mathcal{B}$	$A(z) + B(z)$
Pares Ordenados	$\mathcal{A} \times \mathcal{B}$	$A(z) \cdot B(z)$
Secuencia	$SEQ(\mathcal{A})$	$\dfrac{1}{1 - A(z)}$

Cuadro A.3: Método Simbólico

- Los átomos son las unidades básicas de la clase combinatoria.

- $\mathcal{A} + \mathcal{B}$ es la clase que contiene copias disyuntas de los miembros de $\mathcal{A}$ y $\mathcal{B}$.
- $\mathcal{A} \times \mathcal{B}$ es la clase de pares ordenados (producto cartesiano) de $\mathcal{A}$ y $\mathcal{B}$.

Ejemplos

Ejemplo 1 Aplique el método simbólico para determinar el número total de cadenas ternarias de longitud n.

1. **Clase combinatoria**. $\mathcal{T}$ es la clase combinatoria de las cadenas ternarias de longitud n.

2. **Átomos**. Dado que las cadenas ternarias están conformadas por los caracteres 0, 1 y 2, definimos los átomos z_0, z_1 y z_2.

3. **Construcción combinatoria**. Las cadenas ternarias se pueden definir como los átomos (0, 1 y 2), unidos a una cadena ternaria. Y en efecto, mediante la sucesiva unión de 0, 1 y 2, se pueden obtener todas las cadenas ternarias.

$$\mathcal{T} = \epsilon + (z_0 + z_1 + z_2) \times \mathcal{T}.$$

Nótese que en caso de que no se incluyera el elemento vacío, la clase combinatoria no estaría generando nada, dado que los átomos se le están adjuntando a elementos de la clase ya generados.

4. **Transformación a funciones generadoras.**

Esto se hace por medio de la aplicación de lo establecido en la tabla A.3.

$$\begin{aligned} T(z) &= 1 + (z + z + z) \cdot T(z) \\ T(z) &= 1 + 3z \cdot T(z) \\ T(z)[1 - 3z] &= 1 \\ T(z) &= \frac{1}{1 - 3z}. \end{aligned}$$

5. **Transformación a secuencia f(n).**

De la tabla de funciones generadoras básicas se sabe que

$$T(z) = \sum_{n \geq 0} f(n) \cdot z^n$$

$$T(z) = \frac{1}{1-3z} = \sum_{n \geq 0} 3^n \cdot z^n$$

$$\rightarrow \boxed{f(n) = 3^n}$$

Por tanto, se obtiene que el número de cadenas ternarias de longitud n es 3^n.

Ejemplo 2. Ahora resolveremos el mismo ejercicio, pero de un modo un poco diferente.

1. **Clase combinatoria**. $\mathcal{T}$ es la clase combinatoria de las cadenas ternarias.

2. **Átomos**. Dado que las cadenas ternarias están conformadas por los caracteres 0, 1 y 2, definimos los átomos z_0, z_1 y z_2.

3. **Construcción combinatoria**. Las cadenas ternarias también pueden ser vistos como una *secuencia* de ceros, unos y dos. Esto es:

$$\mathcal{T} = SEQ(z_0 + z_1 + z_2).$$

Aquí no se incluye el elemento vacío, porque la misma secuencia lo incluye.

4. **Transformación a funciones generadoras**

$$T(z) = \frac{1}{1-(z+z+z)}$$

$$T(z) = \frac{1}{1-3z}$$

5. **Transformación a secuencia f(n).**

Se procede de igual forma que el ejemplo anterior.

Ejemplo 3. Cadenas binarias que No contienen la subcadena `11`.

1. **Clase combinatoria**. $\mathcal{B}_{\overline{11}}$: Cadenas binarias que no contienen la subcadena `11`.

2. **Átomos**.

$$z_0,\ z_1$$

3. **Construcción combinatoria**.

Basados en el diseño (árbol binario),

$$\mathcal{B}_{\overline{11}} = \epsilon + z_0 + (z_1 + z_0 \times z_1) \times \mathcal{B}_{\overline{11}}$$

Al añadir z_1 delante de z_0, se evita que hayan 2 unos consecutivos, dado que estará siempre seguido de por lo menos un 0, a excepción del caso donde solo hay un 1, que como se ve, está incluido aparte.

4. **Transformación a funciones generadoras**

$$\begin{aligned} B(z) &= 1 + z + (z + z^2) \cdot B(z) \\ B(z)[1 - z - z^2] &= 1 + z \\ B(z) &= \frac{1 + z}{1 - z - z^2} \\ B(z) &= \frac{1}{1 - z - z^2} + \frac{z}{1 - z - z^2} \end{aligned}$$

5. **Transformación a secuencia f(n).**

En clase mostramos que la FGO $\dfrac{1}{1 - z - z^2}$ genera la secuencia $f(n) = Fib(n)$

Entonces: $B(z)$ genera $f(n) = Fib(n) + Fib(n-1) = Fib(n+1)$, donde $Fib(n)$ es el n-ésimo número de Fibonacci.

Para más información sobre este método, se recomienda consultar [**?**], [**?**] y [**?**].

Para los siguientes problemas debe definir la respectiva expresión mediante el método simbólico, obtener la función generadora, y sacar sus coeficientes. Adicionalmente, debe crear un programa en Python que a través de un menú, permita seleccionar cualquiera de estas opciones y dado un valor de n, mostrar cuántas y cuáles cadenas de longitud n cumplen con la condición pedida. El programa **no puede** generar todas las cadenas de longitud n y de estas, eliminar las que no cumplan.

1. [*] Cadenas Binarias, de longitud n, que:

   - Contengan la subcadena '10'
     1.**Clase combinatoria:**Cadenas binarias que contienen la subcadena 10
     2.**Átomos:** z_0, z_1
     3.**Construcción combinatoria:**

$$B(z) = \epsilon + (z_0 + z_1) \cdot B(z)$$

     4.**Transformación a funciones generadoras:**

$$\begin{aligned} B(z) &= \epsilon + (z_0 + z_1) \cdot B(z) \\ B(z) &= 2z \cdot B(z) + z^2 \cdot (S(z) - B(z)) \\ B(z) \cdot (1 - 2z + z^2) &= \frac{z^2}{1 - 2z} \\ B(z) &= \frac{z^2}{(1 - 2z)(1 - 2z - z^2)} \end{aligned}$$

     (Estudiante: Alejandra Landinez - 200161946)

   - Que No contengan '010'
     Créditos: Profe Alfonso Mancilla
     1.**Clase combinatoria:**Cadenas binarias que no contienen la subcadena 010

2.**Átomos:** z_0, z_1

3.**Construcción combinatoria:**

$$B(z) = z_1 \cdot B(z) + SEQ(z_0) + (\epsilon + z_0 \cdot z_1 \cdot \epsilon + z_0 \cdot z_1(z)$$

4.**Transformación a funciones generadoras:**

$$\begin{aligned} B(z) &= z_1 \cdot B(z) + SEQ(z_0) + (\epsilon + z_0 \cdot z_1 \cdot \epsilon + z_0 \cdot z_1(z) \\ B(z) \cdot \left(\frac{1 - z - z^2}{1 - z}\right) &= \frac{1 + z^2}{1 - z} \\ B(z) \cdot (1 - 2z + z^2 - z^3) &= (1 + z^2) \\ B(z) &= \frac{1 + z^2}{1 - 2z + z^2 - z^3} \end{aligned}$$

2. [*] Cadenas Ternarias, de longitud n, que:

   - No contengan la subcadena '12'

     Créditos: Profe Alfonso Mancilla

     1.**Clase combinatoria:**Cadenas ternarias que no contienen la subcadena 12

     2.**Átomos:** z_0, z_1, z_2

     3.**Construcción combinatoria:**

$$T(z) = (z_0 + z_2) \cdot T(z) + SEQ(z_1) \cdot (\epsilon + z_1 \cdot z_0 \cdot T(z)$$

     4.**Transformación a funciones generadoras:**

$$T(z) = (z_0 + z_2) \cdot T(z) + SEQ(z_1) \cdot (\epsilon + z_1 \cdot z_0 \cdot T(z)$$
$$T(z) = 2z \cdot T(z) + \frac{(1 + z^2 \cdot T(z))}{1 - z}$$
$$T(z) \cdot \left[\frac{1 + z^2 \cdot T(z)}{1 - z}\right] = \frac{1}{1 - z}$$
$$T(z) \cdot \left[1 + 2z^2 - 3z - z^2\right] = 1$$
$$T(z) = \frac{1}{1 - 3z + z^2}$$

(Estudiante: Alejandra Landinez - 200161946)

- Que contengan la subcadena '012'
  Créditos: Profe Alfonso Mancilla 1.**Clase combinatoria:**Cadenas ternarias que contienen la subcadena 012
  2.**Átomos:** z_0, z_1, z_2
  3.**Construcción combinatoria:**

$$T(z) = \epsilon + (z_0 + z_1 + z_2) \cdot T(z)$$

4.**Transformación a funciones generadoras:**

$$T(z) = \epsilon + (z_0 + z_1 + z_2) \cdot T(z)$$
$$T(z) = 3z \cdot T(z) + z^3 \cdot (S(z) - T(z))$$
$$T(z) \cdot (1 - 3z + z^3) = \frac{z^3}{1 - 3z}$$
$$T(z) = \frac{z^3}{(1 - 3z)(1 - 3z - z^3)}$$

- Que contengan un número par de unos
  1.**Clase combinatoria:**Cadenas ternarias que contienen un número par de unos

2.**Átomos:** z_0, z_1, z_2

3.**Construcción combinatoria:**

$$T(z) = \epsilon + (z_0 + z_1 + z_2) \cdot T(z)$$

4.**Transformación a funciones generadoras:**

$$\begin{aligned} T(z) &= \epsilon + (z_0 + z_1 + z_2) \cdot T(z) \\ T(z) &= 1 + 3z \cdot T(z) \\ T(z) &= T(z) \cdot 3z + 1 \\ T(z) \cdot (1 - 3z) &= 1 \\ T(z) &= \frac{1}{1 - 3z} \end{aligned}$$

(Estudiante: Alejandra Landinez - 200161946)

3. [*] Cadenas numéricas, de longitud n, de base cinco que:

   - Tengan sus caracteres en orden creciente. Ejemplo '01122234'

     Créditos: Profe Alfonso Mancilla

     1.**Clase combinatoria:**Cadenas numéricas de base cinco que contienen sus caracteres en orden creciente

     2.**Átomos:** z_0, z_1, z_2, z_3, z_4

     3.**Construcción combinatoria:**

$$Q(z) = SEQ(z_0) \cdot SEQ(z_1) \cdot SEQ(z_2) \cdot SEQ(z_3) \cdot SEQ(z_4)$$

     4.**Transformación a funciones generadoras:**

$$Q(z) = SEQ(z_0) \cdot SEQ(z_1) \cdot SEQ(z_2) \cdot SEQ(z_3) \cdot SEQ(z_4)$$
$$Q(z) = \left[\frac{1}{1-z}\right]^5$$
$$Q(z) = \binom{n+5-1}{5}$$

- Que No contengan las subcadenas '01' ni '43'

Script:

```
import matplotlib.pyplot as plt
import numpy as np
import itertools

def menu_principal():
    print("-" * 50)
    print("Men - m todo - simb lico:")
    print("1. - Cadenas - binarias")
    print("2. - Cadenas - ternarias")
    print("3. - Cadenas - n mericas")
    print("-" * 50)
    opcion = input("Selecciona - una - opci n: - ")
    if opcion == "1":
        menu_binarias()
    elif opcion == "2":
        menu_ternarias()
    elif opcion == "3":
        menu_numericas()
    else:
        print("Error - opci n - no - valida, - digite - una - v lida")

def menu_binarias():
    while True:
```

```
        print("_" * 50)
        print("Cadenas binarias de longitud n que:")
        print("1. Contengan la subcadena '10'")
        print("2. No contengan la subcadena '010'")
        print("3. Regresar")
        opcion_11 = input("Seleccione una opci n:")
        if opcion_11 == "1":
          n = int(input("Digite n:"))
          if n <= 0:
            print("Valor no v lido, ingrese un entero positivo v lido")
            return
          for i in range(2**n):
            binary_string = format(i, f'0{n}b')
            if '10' in binary_string:
              print(binary_string)
        elif opcion_11 == "2":
          n = int(input("Digite la longitud de la cadena:"))
          if n <= 0:
            print("Valor no v lido, ingrese un entero positivo v lido")
            return
          for i in range(2**n):
            binary_string = format(i, f'0{n}b')
            if '010' not in binary_string:
              print(binary_string)
        elif opcion_11 == "3":
          return
        else:
          print("Error opci n no v lida, escoja otra")

def menu_ternarias():
  print("_"*50)
  print("Cadenas ternarias de longitud n que:")
  print("1. No contengan la subcadena '12'")
  print("2. Contengan la subcadena  012  '")
```

```
print("3. Contengan un n mero par de unos")
print("4. Regresar")
opcion_12 = input("Seeleccione una opci n:")
if opcion_12 == "1":
  n = int(input("Digite n:"))
  if n <= 0:
    print("Valor no v lido , ingrese un entero positivo v lido")
    return
  cadenas = []
  for seq in itertools.product('012', repeat=n):
      if '12' not in "".join(seq):
          cadenas.append("".join(seq))
  for string in cadenas:
      print(string)
elif opcion_12 == "2":
  n = int(input("Digite n:"))
  if n <= 0:
    print("Valor no v lido , ingrese un entero positivo v lido")
    return
  cadenas_2 = []
  for seq in itertools.product('012', repeat=n):
      ternary_string = "".join(seq)
      if '012' in ternary_string:
          cadenas_2.append(ternary_string)
  for string in cadenas_2:
      print(string)
elif opcion_12 == "3":
  if n <= 0:
    print("Valor no v lido , ingrese un entero positivo v lido")
    return
  cadenas_3 = []
  for seq in itertools.product('01', repeat=n):
      ternary_string = "".join(seq)
      ones_count = ternary_string.count('1')
```

```
            if ones_count % 2 == 0:
                cadenas_3.append(ternary_string)
        for string in cadenas_3:
            print(string)
    elif opcion_12 == "4":
        return
    else:
        print("Error opci n no v lida , escoja otra")

def menu_numericas():
    print("_"*50)
    print("Cadenas n mericas de longitud n, de base cinco que:")
    print("1. Tengan sus caracteres en urden ascendente(01122234)")
    print("2. No contengan las subcadenas 01 ' ni '43'")
    print("3. Regresar")
    opcion_13 = input("Seleccione una opci n:")
    if opcion_13 == "1":
        n = int(input("Digite n:"))
        if n <= 0:
            print("Valor no v lido , ingrese un entero positivo v lido")
            return
        ascendente = []
        for seq in itertools.product('01234', repeat=n):
            cadena_num = "".join(seq)
            if cadena_num == "".join(sorted(cadena_num)):
                ascendente.append(cadena_num)
        for string in ascendente:
            print(string)
    elif opcion_13 == "3":
        n = int(input("Digite n:"))
        if n <= 0:
            print("Valor no v lido , ingrese un entero positivo v lido")
            return
        cadena_3 = []
```

```
        for seq in itertools.product('2345', repeat=n):
            cadena_num = "".join(seq)
            if '01' not in cadena_num and '43' not in cadena_num:
                cadena_3.append(cadena_3)
        for string in cadena_3:
            print(string)
    else:
        print("Error opci n no v lida , escoja otra")

menu_principal()
```

Problema 2: Conteo mediante funciones generadoras ordinarias

Las funciones generadoras ordinarias (FGO) nos permiten resolver problemas relacionados con conteo, de una manera sencilla; de modo que mediante manipulaciones sencillas sobre estas, problemas que sería difícil resolver directamente, son resueltos por medio de operaciones mucho más simples como la suma de polinomios o la multiplicación de estos.

Recordemos que la función generadora $F(z)$ de una secuencia f_n es de la forma

$$F(z) = \sum_{n \geq 0} f_n \cdot z^n$$

Es importante tener en cuenta que de aquí en adelante f_n siempre hace referencia al número de conjuntos de n objetos idénticos.

En esta sección se mostrará que este f_n puede ser asociado a un problema de conteo y que por medio de el uso de FGO, se puede hallar con facilidad una expresión cerrada para f_n. En estos problemas, obtendremos la FGO de f_n y luego hallaremos el coeficiente de esta función generadora. Es decir, a a partir de $F(z)$ hallaremos f_n. Recordemos la notación $[z^n]F(z) = f_n$.

En este tipo de ejercicios, las siguientes funciones generadoras suelen ser bas-

tante comunes, por lo que es muy importante tenerlas presentes:

$$[z^n]\frac{1}{(1-z)^m} = \binom{m+n-1}{n}$$
$$[z^n]\frac{z^m}{(1-z)^{m+1}} = \binom{n}{m}$$

Veamos algunos ejemplos.

Ejemplo 1 Sea f_n el número de maneras de **seleccionar** n pelotas a partir de infinitas pelotas verdes, azules y doradas.

Es posible plantear el problema como una ecuación entera, de tal modo que f_n sea el número de soluciones a esta ecuación. En este caso, es:

$$S_\mathrm{v} + S_\mathrm{a} + S_\mathrm{d} = n, \quad 0 \leq S_\mathrm{v},\, S_\mathrm{a},\, S_\mathrm{d}.$$

Donde S_v representa el número de pelotas verdes, S_a representa el número de pelotas azules y S_d representa el número de pelotas doradas. Claramente, no puede haber un número de pelotas negativo. La ecuación dicta que la suma de la cantidad de pelotas debe ser igual a n, en este caso, sin restricciones, tal como indica la definición de f_n.

Este problema puede plantearse de la siguiente manera, sin recurrir a funciones generadoras:

$$f_n = \sum_{S_\mathrm{v}+S_\mathrm{a}+S_\mathrm{d}=n} 1,$$

sin embargo, al aproximarnos a la solución utilizando funciones generadoras, el proceso será más sencillo que si se tratara de empezar utilizando esta expresión.

Solución mediante FGO

La solución a este tipo de problemas mediante FGO consiste en los siguientes pasos:

1. El primer paso a realizar es **modelar el problema como una ecuación entera, incluyendo las restricciones**.

2. El segundo paso a realizar es el de plantear la función generadora $F(z)$ para f_n a partir de la ecuación planteada.

3. Finalmente, se procede a hallar el coeficiente de $F(z)$, que al ser la función generadora definida a partir del problema, equivale a una **expresión cerrada** para f_n.

Sea $f_n \equiv$ el número de maneras de **seleccionar** n pelotas a partir de infinitas pelotas verdes, azules y doradas, entonces la ecuación que modela este problema es

$$S_{\mathrm{v}} + S_{\mathrm{a}} + S_{\mathrm{d}} = n, \quad 0 \leq S_{\mathrm{v}},\, S_{\mathrm{a}},\, S_{\mathrm{d}}.$$

Una vez se ha modelado el problema como una ecuación entera, definimos $F(z)$ como la función generadora de f_n y la expresamos a partir de la ecuación planteada. Esto es:

$$F(z) = \left(\sum_{S_{\mathrm{v}} \geq 0} z^{S_{\mathrm{v}}}\right) \cdot \left(\sum_{S_{\mathrm{a}} \geq 0} z^{S_{\mathrm{a}}}\right) \cdot \left(\sum_{S_{\mathrm{d}} \geq 0} z^{S_{\mathrm{d}}}\right).$$

En este caso usamos las mismas variables usadas en la ecuación, pero también es posible usar la letra n:

$$F(z) = S_{\mathrm{v}}\left(\sum_{n \geq 0} z^n\right) \cdot S_{\mathrm{a}}\left(\sum_{n \geq 0} z^n\right) \cdot S_{\mathrm{d}}\left(\sum_{n \geq 0} z^n\right).$$

La elección es cuestión de preferencia. Pero es importante, en cualquiera de los dos casos, señalar a qué variable de la ecuación corresponde cada sumatoria.

Una vez planteada la función generadora de f_n, $F(z)$, pasamos a hacer uso de las FGO básicas. En este caso, sabemos que $\sum_{n \geq 0} z^n = \dfrac{1}{1-z}$, por lo tanto:

$$\begin{aligned} F(z) &= \left(\sum_{n \geq 0} z^n\right) \cdot \left(\sum_{n \geq 0} z^n\right) \cdot \left(\sum_{n \geq 0} z^n\right) \\ &= \left(\sum_{n \geq 0} z^n\right)^3 = \left(\frac{1}{1-z}\right)^3. \end{aligned}$$

Aquí, hemos llegado a que

$$F(z) = \frac{1}{(1-z)^3}.$$

Una vez llegamos a una expresión de este tipo para $F(z)$, el siguiente paso es hallar el coeficiente f_n de esta expresión. Aquí, nuevamente recurrimos a la tabla de FGO básicas. Y recordemos que

$$[z^n]\frac{1}{(1-z)^m} = \binom{m+n-1}{n},$$

es decir, el coeficiente de FGO de esta forma es $\binom{m+n-1}{n}$. Aplicando esto a este ejemplo, con $m = 3$, tenemos que: $[z^n]F(z) = \binom{3+n-1}{n}$, es decir:

$$f_n = \binom{3+n-1}{n}.$$

Con lo que finalmente hemos resuelto este ejemplo.

Ejemplo 2 Sea f_n el número de maneras de distribuir n pelotas en 3 cajas, de tal modo que en las dos primeras cajas haya al menos una pelota, halle una expresión para f_n.

Solución: La ecuación que modela este problema es la siguiente:

$$caja\,1 + caja\,2 + caja\,3 = n, \quad caja\,1 \geq 1,\, caja\,2 \geq 1,\, caja\,3 \geq 0.$$

Ya que tenemos la ecuación, procedemos a plantear la función generadora a partir de esta ecuación.

$$F(z) = caja\,1\left(\sum_{n\geq 1} z^n\right) \cdot caja\,2\left(\sum_{n\geq 1} z^n\right) \cdot caja\,3\left(\sum_{n\geq 0} z^n\right)$$

aquí reescribimos las expresiones hasta llegar a una función conocida.

$$\begin{aligned}
F(z) &= \left(\sum_{n\geq 0} z^n - z^0\right) \cdot \left(\sum_{n\geq 0} z^n - z^0\right) \cdot \left(\sum_{n\geq 0} z^n\right) \\
&= \left(\frac{1}{1-z} - 1\right) \cdot \left(\frac{1}{1-z} - 1\right) \cdot \left(\frac{1}{1-z}\right) \\
&= \left(\frac{1}{1-z} - 1\right)^2 \cdot \left(\frac{1}{1-z}\right) = \left(\frac{1}{(1-z)^2} - \frac{2}{1-z} + 1\right) \cdot \left(\frac{1}{1-z}\right) \\
F(z) &= \frac{1}{(1-z)^3} - \frac{2}{(1-z)^2} + \frac{1}{1-z}
\end{aligned}$$

En este punto, nuevamente usamos la FGO

$$[z^n]\frac{1}{(1-z)^m} = \binom{m+n-1}{n},$$

y la aplicamos para obtener:

$$\begin{aligned}
F(z) = m = 3\frac{1}{(1-z)^3} - m = 2\frac{2}{(1-z)^2} + m = 1\frac{1}{1-z} \\
[z^n]F(z) = \binom{3+n-1}{n} - 2 \cdot \binom{2+n-1}{n} + \binom{1+n-1}{n}.
\end{aligned}$$

Por lo que, finalmente, la solución a este ejercicio es:

$$f_n = \binom{3+n-1}{n} - 2 \cdot \binom{2+n-1}{n} + \binom{1+n-1}{n}.$$

Ejemplo 3

¿De cuántas maneras se pueden distribuir n objetos idénticos en 2 cajas, de tal modo que en la primera caja haya un número par de objetos?

Solución La ecuación que modela este problema es:

$$S_1 + S_2 = n, \quad S_1 \text{ mód } 2 = 0,\ S_2 \geq 0.$$

A partir de esta ecuación, definimos $F(z)$ como la función generadora de f_n y viene dada por

$$F(z) = S_1\left(\sum_{\substack{n\geq 0\\ n \text{ mód } 2=0}} z^n\right) \cdot S_2\left(\sum_{n\geq 0} z^n\right),$$

utilizando las FGO conocidas, tenemos:

$$\begin{aligned} F(z) &= \left(\sum_{n\geq 0} z^{2n}\right) \cdot \left(\sum_{n\geq 0} z^n\right) \\ &= \left(\frac{1}{1-z^2}\right) \cdot \left(\frac{1}{1-z}\right) = \frac{1}{(1-z^2)\cdot(1-z)} = \frac{1}{(1-z)\cdot(1+z)\cdot(1-z)} \\ &= \frac{1}{(1-z)^2\cdot(1+z)}. \end{aligned}$$

En este punto, aplicamos fracciones parciales y obtenemos algo de la forma

$$\begin{aligned} F(z) &= \frac{Az+B}{(1-z)^2} + \frac{C}{1+z} \\ &= \frac{Az}{(1-z)^2} + \frac{B}{(1-z)^2} + \frac{C}{1+z} \end{aligned}$$

y aquí, empleando las FGO básicas, se obtiene:

$$f_n = A\cdot\binom{n}{1} + B\cdot\binom{2+n-1}{n} + C\cdot(-1)^n.$$

Aquí reemplazamos los coeficientes de las fracciones parciales, para mayor facilidad:

$$A = -\frac{1}{4},\ B = \frac{3}{4},\ C = \frac{1}{4}.$$

Entonces

$$f_n = -\frac{\binom{n}{1}}{4} + \frac{3 \cdot \binom{2+n-1}{n}}{4} + \frac{(-1)^n}{4}$$
$$= \frac{3 \cdot \binom{2+n-1}{n} - \binom{n}{1} - (-1)^n}{4}.$$

Método 2: También es posible seguir de la siguiente manera, mediante la operación de suma parcial y aplicando algunas sumatorias conocidas.

$$F(z) = \left(\sum_{n\geq 0} z^{2n}\right) \cdot \left(\sum_{n\geq 0} z^n\right) = \frac{1}{1-z} \cdot \left(\frac{1}{1-z^2}\right)$$

aquí es necesario definir el coeficiente de $\dfrac{1}{1-z^2}$, que sabemos se puede escribir como

$$[z^n]\frac{1}{1-z^2} = \begin{cases} 1, & n \text{ mód } 2 = 0 \\ 0, & n \text{ mód } 2 = 1 \end{cases},$$

es decir, es 1 para los valores pares de n y 0 para los valores impares de n. Sin embargo, una expresión cerrada y que es más útil en este ejercicio, es la siguiente:

$$[z^n]\frac{1}{1-z^2} = \frac{1+(-1)^n}{2}.$$

Entonces, aplicando esto sobre $F(z)$ obtenemos:

$$F(z) = \frac{1}{1-z} \cdot \left(\sum_{n\geq 0} \frac{1+(-1)^n}{2} \cdot z^n\right).$$

Recordemos que la suma parcial está definida así:

$$\frac{1}{1-z} \cdot \sum_{n\geq 0} g_n \cdot z^n = \sum_{n\geq 0} \sum_{0\leq k\leq n} g_k \cdot z^n.$$

Entonces, en este ejemplo, $g_n = \dfrac{1+(-1)^n}{2}$, entonces $g_k = \dfrac{1+(-1)^k}{2}$ y por ende

$$F(z) = \sum_{n\geq 0} f_n \sum_{0\leq k\leq n} \frac{1+(-1)^k}{2} \cdot z^n,$$

por lo tanto,

$$f_n = \sum_{0\leq k\leq n} \frac{1+(-1)^k}{2} = \frac{1}{2}\cdot\left(\sum_{0\leq k\leq n} 1 + \sum_{0\leq k\leq n} (-1)^k\right) = \frac{1}{2}\cdot\left((n+1) + \frac{1-(-1)^{n+1}}{1-(-1)}\right)$$
$$= \frac{1}{2}\cdot\left((n+1) + \frac{1-(-1)^{n+1}}{2}\right) = \frac{2n+2+1-(-1)^{n+1}}{4}$$
$$f_n = \frac{2n+3-(-1)^{n+1}}{4}.$$

Con lo que finalmente, obtenemos la respuesta a este ejercicio:

$$f_n = \frac{2n+3-(-1)^{n+1}}{4}.$$

Es fácilmente demostrable que esta expresión es igual de válida que la obtenida con el primer método.

Ejercicios

1. [*] Si se seleccionan n posts de la red social X, entre con hashtag y sin hashtag, ¿cuál es la probabilidad de escoger una cantidad impar de posts con hashtag.

   Los hashtags (escritos con el signo # antepuesto) se usan para indexar palabras clave o temas en la red social X(antigua twitter)
   (Estudiante Mariana Guerrero Benavides-200173479)
   La ecuación que modela este problema es:

   $$S_h + S_n = n, \quad S_h \text{ mód } 3 = 0,\ S_n \geq 0.$$

   S_h son posts con hashtags y S_n son posts sin hashtags.

$$F(z) = S_h\left(\sum_{\substack{n\geq 0 \\ n \text{ mód } 3=0}} z^n\right) \cdot S_n\left(\sum_{n\geq 0} z^n\right),$$

Utilizando las FGO conocidas, tenemos:

$$\begin{aligned} F(z) &= \left(\sum_{n\geq 0} z^{3n}\right) \cdot \left(\sum_{n\geq 0} z^n\right) \\ &= \left(\frac{1}{1-z^3}\right) \cdot \left(\frac{1}{1-z}\right) = \frac{1}{(1-z^3)\cdot(1-z)} = \frac{1}{(1-z)\cdot(z^2+z+1)\cdot(1-z)} \\ &= \frac{1}{(1-z)^3\cdot(z^2+z+1)}. \end{aligned}$$

Se aplican fracciones parciales:

$$F(z) = \frac{A(z+1)}{(z^2+z+1)} + \frac{B}{(1-z)} + \frac{C}{(1-z)^2}$$

$$f(n) = A\cdot\langle 1,0,-1,1,0,-1,1\rangle - B + C\cdot\binom{2+n-1}{n}$$

$$A = -\frac{1}{3}, \quad B = \frac{1}{3}, \quad C = \frac{1}{3}$$

Reemplazamos los valores de los coeficientes A,B, y C:

$$\begin{aligned} f(n) &= -\frac{1}{3}\cdot\langle 1,0,-1,1,0,-1,1\rangle - \frac{1}{3} + \frac{1}{3}\cdot\binom{2+n-1}{n} \\ &= \frac{\langle 1,0,-1,1,0,-1,1\rangle}{3}\cdot -\frac{1}{3} + \frac{\binom{2+n-1}{n}}{3} \\ &= \frac{\binom{2+n-1}{n} + \langle 1,0,-1,1,0,-1,1\rangle - 1}{3} \end{aligned}$$

Ahora, para hallar la probabilidad de escoger una cantidad impar de posts con hashtag, entonces se debe dividir entre n, quedando de la siguiente ma-

nera:

$$P(S_h) = \frac{\dfrac{\binom{2+n-1}{n} + \langle 1,0,-1,1,0,-1,1\rangle - 1}{3}}{n}$$

2. [*] Encuentre el número de soluciones de la ecuación $x_1+x_2+x_3+x_4 = 17$, donde x_i, $i = 1,2,3,4$ son enteros no negativos tal que $x_1 \leq 3, x_2 \leq 4, x_3 \leq 5$, y $x_4 \leq 8$. (Profesor Alfonso Mancilla Herrera)

$$F(z) = \left(\sum_{n=0}^{3} z^n\right) \cdot \left(\sum_{n=0}^{4} z^n\right) \cdot \left(\sum_{n=0}^{5} z^n\right) \cdot \left(\sum_{n=0}^{8} z^n\right)$$
$$F(z) = \left(\frac{1-z^4}{1-z}\right) \cdot \left(\frac{1-z^5}{1-z}\right) \cdot \left(\frac{1-z^6}{1-z}\right) \cdot \left(\frac{1-z^9}{1-z}\right)$$
$$F(z) = \frac{1}{(1-z)^4} \cdot (1-z^4) \cdot (1-z^5) \cdot (1-z^6) \cdot (1-z^9)$$
$$[z^n]F(z) = \binom{4+n-1}{n} \cdot (1-z^4) \cdot (1-z^5) \cdot (1-z^6) \cdot (1-z^9)$$
$$[z^n]F(z) = \binom{n+3}{n} \cdot (z^{24} - z^{20}z^{19} - z^{18} + z^{14} + z^{13} + z^{11} + z^{10} - z^6 - z^5 - z^4 + 1)$$
$$[z^n]F(z) = \binom{n+3}{3} \cdot (z^{24} - z^{20}z^{19} - z^{18} + z^{14} + z^{13} + z^{11} + z^{10} - z^6 - z^5 - z^4 + 1)$$
$$[z^n]F(z) = \binom{n+3}{3} \cdot (z^{14} + z^{13} + z^{11} + z^{10} - z^6 - z^5 - z^4 + 1)$$
$$f(n) = \binom{n+3}{3} \cdot (z^{14} + z^{13} + z^{11} + z^{10} - z^6 - z^5 - z^4 + 1)$$

ahora evaluando para n=17

$$f(17) = \binom{17+3}{3} \cdot (1^{14} + 1^{13} + 1^{11} + 1^{10} - 1^6 - 1^5 - 1^4 + 1)$$
$$f(17) = \binom{20}{3} \cdot (1+1+1+1-1-1-1+1)$$
$$f(17) = \binom{20}{3} = \frac{20!}{3!(20-3)!} = 1140$$

Entonces, 1140 es el número de soluciones de la ecuación $x_1+x_2+x_3+x_4 =$

17

3. [*] ¿Cuántas posibles distribuciones de 18 objetos en 3 diferentes cajas son posibles si una caja tiene el doble de objetos que las otras 2 juntas?

   (Estudiante Juan Camilo Aguirre Gonzalez - 200156480)

   En esta situación, tendremos en cuenta que tienen que estar los 18 objetos presentes, donde tomaremos la primera caja como x, la segunda caja como y, y la tercera caja z.

   Estas seguirian las siguientes propiedades:

   $x \geq 0$

   $y \geq 0$

   $z \geq 0$

   $x = 2 \cdot (y + z)$

   $x + y + z = 18$

   Usemos principios de combinación para poder hallar todas las posibles distribuciones:

   Para 18 objetos, podemos distribuirlos de la siguiente manera:

   $n(s_1) = [\binom{18}{12} \cdot \binom{18-12}{6} \cdot \binom{18-12-6}{0}]\binom{18}{12,6,0}$

   (En este caso, en la primera caja se escojen 12 de los 18 elementos, en la segunda caja se van a escojer 6 de los elementos que quedaron de la eleccion de la caja 1, (es decir, 18-12), y en la tercera caja se escogieron 0, porque ya no hay elementos a escoger. En este caso la suma de la segunda y tercera es 6, y $2 \cdot 6 = 12$, por lo tanto, este caso cumple. Al final se coloca la distribucion de los elementos: 12, 6 y 0.)

   Se hara de forma analoga con el resto de los casos:

   $n(s_2) = [\binom{18}{12} \cdot \binom{18-12}{5} \cdot \binom{18-12-5}{1}]\binom{18}{12,5,1}$

   $n(s_3) = [\binom{18}{12} \cdot \binom{18-12}{4} \cdot \binom{18-12-4}{2}]\binom{18}{12,4,2}$

   $n(s_4) = [\binom{18}{12} \cdot \binom{18-12}{3} \cdot \binom{18-12-3}{3}]\binom{18}{12,3,3}$

   Esto se podria resumir como la siguiente suma:

$$n(S) = \sum_{i=1}^{4} n(Si)$$

4. [*] Cuántos subconjuntos de seis elementos escogidos del conjunto $S = \{1, 2, 3, \ldots, 19, 20\}$ hay sin enteros consecutivos ?. Por ejemplo, si cinco está en el subconjunto, entonces 4 y 6 no pueden estar en éste.
   (Profesor Alfonso Mancilla Herrera)
   Esta función se halló en el primer inciso de 'Problema 1: Funciones Generadoras y Secuencias.' del entregable 1, por medio de funciones generadoras ordinarias. El resultado es: $f(n) = f(n-1) + f(n-2)$

   Al tomar $n = 20$ el resultado que se obtiene es el siguiente:

$$\begin{aligned} f(n) &= f(n-1) + f(n-2) \\ f(20) &= f(20-1) + f(20-2) = f(19) + f(18) \\ f(20) &= 6765 + 4181 \\ f(20 &= = 10946 \end{aligned}$$

   Entonces, hay 10946 subconjuntos de seis elementos escogidos del conjunto $S = 1, 2, \ldots, 20$, sin enteros consecutivos.

5. [*] Cuántos números enteros de $n \geq 1$ cifras hay, que tengan al menos un cero? (Estudiante Juan Camilo Aguirre Gonzalez - 200156480) y (Profesor Alfonso Mancilla Herrera)

   Para poder hallar el numero de numeros enteros que tienen una o mas cifras, y contienen el numero cero, podemos conocerlo de la siguiente manera:

   Para una cifra, como el primero digito no puede ser 0, y como el siguiente numero SI puede ser cualquiera de los 10 numeros (0-10), podemos usar la siguiente formula cerrada:

   $9 \cdot 10^{(n-1)}$ (cantidad de numeros enteros)

   Los numeros que no contienen el 0 se pueden escribir de la siguiente forma:

   Como de los 10 numeros (0-9), excluimos el 0, quedaria como

   $9^{(n)}$

Con esto en mente, solo seria juntar las dos ecuaciones y tendriamos el numero de numeros que contienen por lo menos 0, para n¿=1 cifras.

$9 \cdot 10^{(n-1)} - 9^n$

Y tambien se puede escribir de la siguiente manera:

$$n(S) = \sum_{i=1}^{n} [\binom{0}{i}\binom{1}{1}] \cdot \binom{n-i}{n-i}\binom{9}{1}$$

6. [*] ¿Cuántos enteros entre 1 y 1,000,000 tienen la suma de dígitos igual a 20?
(Estudiante Mariana Guerrero Benavides-200173479)
Se descarta el número 1,000,000 ya que la suma de sus valores no llegan a 20. Por lo que, se simplifica a considerar valores que contienen entre 1 y 6 enteros, que al sumarlos den 20.

$$= x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 20$$
$$= x_i \leq 9; i = 1, 2, 3, 4, 5, 6$$
$$n(S) = \left[\binom{20}{20}\binom{6+20-1}{20}\right] - \left[\binom{10}{10}\binom{6}{1}\right] \cdot \left[\binom{20-10}{20-10} \cdot \binom{6+(20-10)-1}{20-10}\right]$$

La cantidad de enteros entre 1 y 1,000,000 que tienen la suma de dígitos igual a 20, está dado por $n(S)$

Problema 3: Titulares de Fake News

Código Python en Colab

Los *Fake News* son cada vez más populares, si bien muchos de ellos son escritos manualmente para difundir contenido falso, otros son elaborados de manera automatizada. Entender cómo se generan, permite identificarlos de forma más sencilla. Por lo cual, realizará cada una de las distintas etapas que se ven involucradas en dicha labor. De esta manera, se requiere que descargue una cantidad n de titulares de un portal de noticias como BBC Mundo, mediante técnicas de *web-scrapping*. Esta información debe ser almacenada en un archivo .csv autogenerado por el programa, que contenga la fecha de la noticia y el titular, y cualquier

otro dato, de ser necesario. Así mismo, con los datos recopilados debe generar su propio titular falso utilizando Cadenas de Markov. Finalmente, retorne gráficos que permitan realizar un análisis de la información recopilada:

- Distribución de frecuencia de las palabras utilizadas en todos los enunciados. Tener en cuenta una etapa previa de limpieza de la información para evitar las stop-words.
- Palabra más usada por fecha de publicación de la noticia. Puede escoger el tipo de gráfico que considere más conveniente.
- Cantidad de artículos publicados por fecha.
- Nube de palabras más utilizadas.
- Distribución de los 10 bigramas y trigramas más frecuentes.

Genere 30 titulares aleatorios, basado en la información recopilada, y observe si los resultados se mantienen al realizar nuevamente el análisis anterior. Los resultados y las conclusiones obtenidas deben ser incluídas en el informe.

Nota: La solución debe implementarse utilizando las siguientes librerías de Python: Pandas, seaborn, matplotlib, bs4, urllib, nltk. Se recomienda utilizar técnicas de procesamiento de lenguaje natural (NLP, por sus siglas en inglés) para llevar a cabo los análisis de palabras.

Ejemplo

Para este ejemplo, se ha escogido la sección internacional de BBC News, y se ha ejecutado un programa que extrae 200 titulares de noticias, junto con su fecha de publicación. Esta información es almacenada en un archivo .csv que será utilizado posteriormente. A continuación se muestra un extracto de estos:

```
...
19:58 3-03-2022;"Estamos aterrorizados": el asedio de las tropas rusas a la
ciudad portuaria de Mariúpol
18:36 3-03-2022;Los voluntarios polacos que abren sus casas y escuelas a
refugiados ucranianos
```

```
18:11 3-03-2022;Los gobiernos de Francia y Alemania incautan yates de
oligarcas rusos
17:23 3-03-2022;Quién pertenece al círculo íntimo de Putin que dirige la
invasión rusa en Ucrania
14:26 3-03-2022;6 momentos clave de la primera semana de invasión de Rusia
a Ucrania
...
```

Nótese que como delimitador se ha utilizado el punto y coma (';'). Una vez extraído cada texto, se procedió a utilizar cadenas de Markov para generar 30 titulares aleatorios.

```
1;Las fuerzas aéreas más alta del presidente de Ucrania.
2;El sobreviviente del mítico motín del avión de defensa de bombardeos
indiscriminados sobre el emotivo discurso del Hollywood de una zona
3;El gigantesco costo económico que se ha adoctrinado a Ucrania.
4;El nuevo auge de seguridad privadas que le está atrapada en Moscú.
5;El beso de farsa.
6;Las joyas arquitectónicas que Rusia de los bombardeos con protestas el
sitio de Chernóbil bajo vigilancia rusa.
...
```

Finalmente, se realizó las funciones solicitadas para la sección de NLP y se obtuvo lo siguiente:

Distribución de frecuencia de las palabras más utilizadas en todos los enunciados:

```
[('ucrania', 77), ('rusia', 60), ('invasión', 29), ('guerra', 27),
('putin', 18), ('rusa', 18), ('rusos', 15), ('ruso', 14), ('mariúpol', 13),
('rusas', 13)]
```

Palabra más usada por fecha de publicación de la noticia: Se escogió la fecha '2022-03-13'.

```
[('grandes', 2), ('rusia', 2), ('ucrania', 2), ('cerca', 2), ('muere', 1),
('william', 1), ('hurt', 1), ('ganador', 1), ('oscar', 1), ('beso', 1)]
```

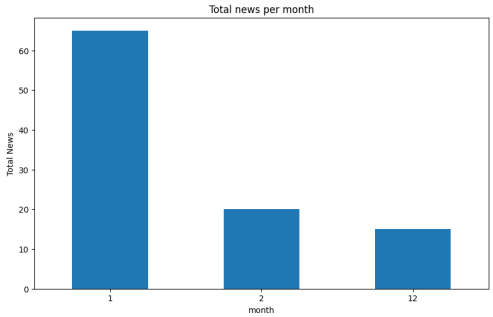


Figura A.1: Total de noticias publicadas por mes.

Figura A.2: Word cloud.

Cantidad de artículos publicados por fecha:

Nube de palabras más utilizadas:

Distribución de los 10 bigramas y trigramas más frecuente:

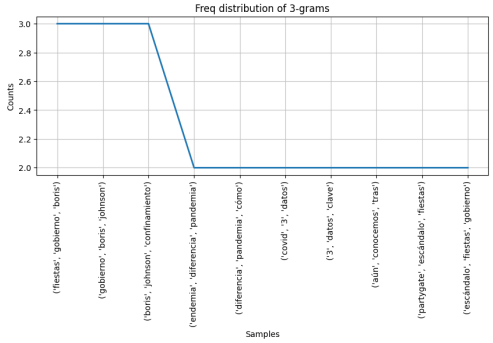


Figura A.3: Distribución de frecuencia de los 10 primeros trigramas.

(Estudiante Mariana Guerrero Benavides-200173479)

Con ayuda de las siguientes referencias: [?] [?] Utilizando la fuente base de [?] BBC Mundo

Anexo Código

```
import json
import pandas as pd
from datetime import datetime
import seaborn as sns
import matplotlib.pyplot as plt
from bs4 import BeautifulSoup
import nltk
from nltk.corpus import stopwords
from nltk.util import ngrams
from wordcloud import WordCloud
from collections import Counter
import requests
import random

# Funci n para extraer datos de una pag web espec fica
def scrape_pag(soup, titulares): # BeautifulSoup para analizar y extraer da
  # Toma el <script> del tipo de archivo
  script_tag = soup.find('script', string=lambda x: x and 'window.SIMORGH_D
  # Se extraen los datos del <script>
  json_data = script_tag.string
  # Se limpia el contenido para validarlo
  json_data = json_data.split('=', 1)[1].strip()
  # Parsea el archivo
  data = json.loads(json_data)
  # Accede a la clase que contiene datos de noticias
  curations = data['pageData']['curations']
  # Itera sobre los datos de la clase. Extrae los titulares y fechas
  for curation in curations:
    for summ in curation['summaries']:
      titulo = summ['title']
```

```
        fecha = summ['firstPublished']
        titulares.append({
            'Fecha': fecha,
            'T tulo': titulo
        })

# Url de la p gina a scrapear
pag_web = 'https://www.bbc.com/mundo/topics/c7zp57yyz25t'
pag = requests.get(pag_web) # Se realiza la solicitud a la p gina
soup = BeautifulSoup(pag.text, 'html.parser') # Parsea la p gina
titulares = [] # Inicializa la lista de titulares
scrape_pag(soup, titulares) # Scrapea la p gina inicial
# Toma el siguiente elemento del html
next_pag = soup.find('li', class_='next')
# Scrapear siguiente p gina
while next_pag is not None:
  next_pag_relativa = next_pag.find('a', href=True)['href']
  pag = requests.get(pag_web + next_pag_relativa)
  soup = BeautifulSoup(pag.text, 'html.parser')
  scrape_pag(soup, titulares)
  next_pag = soup.find('li', class_='next')

# Se crea un dataframe y se almacenan los datos extra dos
df = pd.DataFrame(titulares)
# Se guarda df en csv
df.to_csv('titulares.csv', index=False, encoding='utf-8') # exportaci n a

# C denas de Markov: 30 titulares aleatorios de los titulares existentes
def tokeniza_titulares(df): #Tokenizaci n titulares
  tok = []
  for titular in df['T tulo']:
      titulares_tok = titular.split()
      tok.extend(titulares_tok)
  return tok
```

```
titulares_token = tokeniza_titulares(df) # Tokenizaci n palabras de titula

# Convertir el texto anterior a una lista donde se escojan terminos aleator
def texto_random(palabras, longitud_deseada=24):
  random.shuffle(palabras) # Mezcla los tokens
  generado = " ".join(palabras[:longitud_deseada])  #Combina los tokens
  return generado

# Se generan los titulares aleatorios
num_titulares_random = 30
for i in range(num_titulares_random):
  nuevos_titulares = texto_random(titulares_token, longitud_deseada=24)
  print(f"{i + 1}: {nuevos_titulares}")
print("-"*90)

nltk.download('stopwords') # descarga de las soptowords
stop_palabras = set(stopwords.words("spanish")) # se obtiene el conjunto de

# Limpia y tokenizaci n de las palabras: Funci n que divide los enunciado
def tokeniza_limpia(text):
  palabras = text.lower().split()
  limpiado = [word for word in palabras if word.isalpha() and word not in s
  return limpiado

todas_palabras = [word for titulo in titulares for word in tokeniza_limpia(
cont_palabras = Counter(todas_palabras) # usado para contar la frecuencia a

# Genera las 10 palabras m s usadas en enunciados
top_palabras = cont_palabras.most_common(10)
print("\n Distribuci n de frecuencia de las palabras m s utilizadas en t
print(top_palabras)
print("-"*90)
```

```
# Palabras m s usadas por fecha de publicaci n
cont_dia = df.groupby("Fecha")["T tulo"].apply(lambda x: "-".join(x)).rese
cont_dia["Titular_limpio"] = cont_dia["T tulo"].apply(tokeniza_limpia)
cont_dia["Palabra_comun"] = cont_dia["Titular_limpio"].apply(lambda x: Cou
buscar_dia = "2023-10-04" # modificar fecha a elecci n
df["Fecha"] = pd.to_datetime(df["Fecha"]) # Pasamos la columna de fecha a d
buscar_dia = datetime.strptime(buscar_dia, "%Y-%m-%d")
buscar_a o = buscar_dia.year # Se toma a o y mes de lo que se busca
buscar_mes = buscar_dia.month
buscar_titular = df[df["Fecha"].apply(lambda x: x.year == buscar_a o and x
if not buscar_titular.empty:
  palabra_comun = Counter("-".join(buscar_titular["T tulo"]).split()).most
  print(f"Palabra m s usada por fecha de publicaci n de la noticia: Se es
  print(palabra_comun)
else:
  print(f"No se encontraron palabras para la fecha {buscar_dia}")
print("-" * 90)

# Cantidad de art culos publicados por fecha
df["Fecha"] = pd.to_datetime(df["Fecha"])
df["Dia"] = df["Fecha"].dt.day # Extraer fecha por dia
articulo_dia = df["Dia"].value_counts().sort_index() # Cuenta el n mero de
# Gr fica de la cantidad de art culos publicados por fecha
plt.figure(figsize=(8, 5))
articulo_dia.plot(kind="bar")
plt.xlabel("Dia"), plt.ylabel("Cantidad de art culos")
plt.title("Cantidad de art culos publicados por fecha")
plt.show()
print("-"*90)

# Nube de las palabras m s usadas
nube = WordCloud(width=800, height=400, background_color="white").generate(
plt.figure(figsize=(8, 5))
plt.imshow(nube, interpolation="bilinear")
```

```
plt.title("Nube de palabras m s utilizadas")
plt.show()
print("-"*90)

# Distribuci n de los 10 bigramas y trigramas m s frecuentes
def buscar_ngrams(texto, n):
  ngram_cont = Counter(ngrams(texto, n))
  return ngram_cont.most_common(10)

# Se encuentra los 10 m s comunes bigramas y trigramas
bigramas = buscar_ngrams(todas_palabras, 2)
trigramas = buscar_ngrams(todas_palabras, 3)
# Gr fica de bigramas
niveles_bigrama, bigrama_cont = zip(*bigramas)
niveles_bigrama = [' '.join(label) for label in niveles_bigrama]
plt.figure(figsize=(8, 5))
plt.barh(range(10), bigrama_cont, tick_label = niveles_bigrama)
plt.xlabel('Frecuencia'), plt.ylabel('Bigramas')
plt.title('Distribuci n de los 10 bigramas m s frecuentes')
plt.show()
# Gr fica de trigramas
niveles_trigrama, trigrama_cont = zip(*trigramas)
niveles_trigrama = [' '.join(label) for label in niveles_trigrama]
plt.figure(figsize=(8, 5))
plt.barh(range(10), trigrama_cont, tick_label = niveles_trigrama)
plt.xlabel('Frecuencia'), plt.ylabel('Trigramas')
plt.title('Distribuci n de los 10 trigramas m s frecuentes')
plt.show()
```

Explicación de los métodos implementados

Primeramente se realiza el web scraping para extraer los datos de la página web 'BBC NEWS MUNDO' por medio de las librerías request y BeautifulSoup. Posteriormente, la librería Pandas para la manipulación y el manejo de los datos recolectados.

- El método pd.DataFrame() se utilizó para generar un dataframe a partir de los enunciados extraídos.
- Para exportar esto datos a un archivo csv, se utiliza el método df.to_csv(), guardándolo como 'titulares.csv'. En este, se hace énfasis en que los índices de la fila no se incluyan en el archivo.
- Para normalizar el texto (eliminar signos de puntuación y/o caracteres especiales), se utilizó la función download() de NLTK para descargar los *stopwords*, que, son palabras que ayudan a dar estructura al contenido, más no generan un aporte significativo en el texto. Al eliminarlas, se reduce la complejidad del texto, puesto que, estas suelen repetirse constantemente. Esta limpieza, se realizó mediante el método tokeniza_limpia(), que devuelve una lista de palabras limpias.
- La función counter() de la librería collections se utilizó para poder contar la frecuencia de las palabras luego de haber tokenizado el contenido. Estas se guardan en un diccionario.
- Para generar los 30 titulares aleatorios se hizo uso de las cádenas de markov, por medio del método texto_random().
- De la librería Pandas se utilizó el método df.groupby(), en el cual agrupamos los datos en función de la columna 'Fecha'. Al utilizar apply() se concatenan los titulares en un grupo de fecha.
- Se utilizó pd.to_datetime() para poder convertir la columna 'Fecha' a objetos de fecha y hora de Pandas, para permitir su análisis de manera más accesible.

- Para contar los valores de la columna 'Dia' se utilizó el método value_counts(), y se organizaron tales días de manera ascendente por medio del método sort_index().
- Para la realización del diagrama de nubes, se utilizó el método generate() de la librería WordClouds de Python. Este permite generar tal diagrama, a partir de un *text string*.
- Se utilizaron los n-gramas como técnica para el procesamiento del lenguaje natural, en donde podríamos obtener la serie de los n términos (en este caso las palabras). En este caso, $n = 2, 3$ para la representación del bígama y trígrama respectivamente.

Entregable

- Archivo de Jupyter Notebook `.ipynb` con el código utilizado para generar las gráficas a partir del archivo `csv`. El archivo debe quedar de tal manera que si se cambia el `csv`, al ejecutar todas las celdas las gráficas se deben generar **todas** correctamente, así como también el `csv` con solo las columnas indicadas. Este punto es de vital importancia pues su código será probado con distintos archivos, con varias distribuciones de las probabilidades de las láminas.
- Versión en PDF del informe, el cual debe tener:
  - Explicación de su código.

    Deben explicar los métodos utilizados de las librerías **NumPy** y **Pandas**, para qué sirven de manera general y para qué les sirvieron en específico para la realización del trabajo.

    Además, es **obligatorio** el uso de la librería **Pandas** para la lectura y generación de los `csv` pedidos.
- Código fuente en LaTeXdel informe.
- Los items listados deben ser enviados en forma separada. **NO** es necesario que adjunten los `csv` generados, pues su código debe ser capaz de generarlos.

Entrega 03: Bot de Telegram,Cifrado César, lenguajes naturales y cadenas de Markov, Diseño de funciones.

Estudiantes: Juan Camilo Aguirre- (200156480), Mariana Guerrero Benavides-200173479), Alejandra Landinez Lamadrid - 200161946)

Problema 1: Bot de Telegram

Estudiante Juan Camilo Aguirre- (200156480)
Para los siguientes casos deben crear **UN** bot de Telegram (ver [**?**])
Código Python

- Criptografía. El Cifrado César
- Procesamiento de Lenguajes Naturales. Modelos de Markov

El bot debe implementar las funcionalidades descritas a continuación. Es **obligatorio** un comando `\ayuda` que describa qué hace el bot y qué comandos tiene. Además, en cada punto debe haber validación de los datos ingresados por el usuario. Cada comando debe mostrar o facilitar la comprensión del formato o la forma en que el usuario ingresa los datos.

Cifrado César

Estudiante Mariana Guerrero Benavides-200173479) Realice una función que codifique un mensaje utilizando el Cifrado César. Teniendo en cuenta el desplazamiento ingresado por el usuario, y que el mensaje puede contener mayúsculas, números y puntuación.
Realice el descifrado de un mensaje codificado con Cifrado César. Para esta opción el usuario **no** ingresará la cantidad de letras desplazadas, y el mensaje cumple las especificaciones del ítem anterior.

El enunciado se refiere a que el usuario debe poder codificar el mensaje tal cual y como se envía. Es decir, su mensaje puede incluir números, tildes, y caracteres especiales. En este sentido, la idea consiste en modificar el cifrado tradicional de

César, que solo contempla letras en su corrimiento o desplazamiento. El orden de estas no influye, siempre y cuando se mantenga en el cifrado y descifrado. Por otra parte, se le debe indicar al usuario cuál será el desplazamiento que pueda realizar para el cifrado.

Un posible ejemplo de esto, sería descifrar el siguiente mensaje que tiene corrimiento de 15 carácteres:

```
THIt tH 92 BtCHpyt rúuG¿só R32 CkBtGDH N rpGprItGtH tHEtr_x¿AtH_f
```

Su respectivo descifrado es:

```
Este es UN mensaje c1fr4d0 CON números y caracteres espec_i4les_.
```

alfabeto =

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_-.áéíóú?!¿¡
```

Lenguajes naturales y cadenas de Markov

Estudiantes Juan Camilo Aguirre- (200156480), Alejandra Landinez Lamadrid - 200161946)

Dado un sitio web estático, implemente *web scraping* para obtener el texto plano del mismo. A partir de este, implemente un modelo de cadenas de Markov para generar un texto y retórnelo al usuario.

Implementar un modelo de Markov de orden K a partir del texto retornado. A partir de este modelo, deberán generar texto pseudo-aleatorio.

Modelos de Markov Un modelo de Markov de orden $K = 0$ asume que cada caracter tiene una probabilidad fija, independiente de los caracteres anteriores. Por ejemplo, si el texto base es .ªbabbabcdd", entonces las probabilidades asignadas son:

$$Pr(X = a) = \frac{3}{10} = 0{,}30, \qquad Pr(X = b) = \frac{4}{10} = 0{,}40,$$
$$Pr(X = c) = \frac{1}{10} = 0{,}10, \qquad Pr(X = d) = \frac{2}{10} = 0{,}20.$$

Es decir, a la hora de generar texto, el siguiente caracter será generado a partir de las probabilidades ya calculadas. Este ejemplo es válido para un modelo de

Markov de orden $K = 0$. Sin embargo, para diferentes valores de K, lo que sucede es que las probabilidades dependen de los K caracteres anteriores. Por ejemplo, si la cadena es .ªbbbcabcbbbcbz $K = 2$, entonces habrá cinco K-cadenas distintas: ab, ca, cb, bc y bb. Y a partir de estas cadenas, se determinan las probabilidades. Esto es: dada una K-cadena, cuál es la probabilidad de que el siguiente caracter sea X. En este ejemplo, las probabilidades serían:

Llave	Probabilidades
ab	c: $1/2 = 0{,}5$; b: $1/2 = 0{,}5$
ca	b: $1/1 = 1{,}0$
cb	b: $1/1 = 1{,}0$
bc	a: $1/3 \approx 0{,}33$; b: $2/3 \approx 0{,}67$
bb	c: $2/4 = 1/2 = 0{,}5$; b: $2/4 = 1/2 = 0{,}5$

Cuadro A.4: Tabla de probabilidades

Esto se interpreta de este modo: Si en el texto actual, los últimos 2 caracteres son "bc", el siguiente caracter será .ªçon un 33.3 % de probabilidad, o "bçon un 66.7 % de probabiilidad.

A partir de esto, deben realizar un programa en Python que haga lo siguiente:

- Dado un sitio web estático, implemente *web scraping* para obtener el texto plano del mismo, guardar en archivo .txt .
- con el archivo *.txt* de entrada y los parámetro K y N. A partir del texto y del parámetro K, construir un modelo de Markov de orden K. Y usando este modelo, genere un texto de N caracteres, donde los primeros K caracteres del texto generado son los mismos K primeros caracteres del archivo de texto.
- Debe permitir visualizar un histograma de las frecuencias de las K-tuplas de caracteres.

El programa debe permitir seleccionar si se desea visualizar el histograma o si se desea generar un texto, pero permitiendo cambiar de opción sin tener que reabrir el programa.

Problema 2:
Programa ECO-Uninorte. Reseña de artículo científico

Estudiantes Juan Camilo Aguirre- (200156480)

Link de la reseña realizada

Cada equipo tiene que realizar la reseña de un artículo escogido de la lista que entregará el profesor, sobre los temas estudiados en la asignatura, acorde con la guía anexa en el enlace anterior.

Problema 3: Diseño de funciones recursivas(Enunciado pendiente)

Estudiantes Alejandra Landinez Lamadrid - 200161946), Mariana Guerrero Benavides-200173479)

Acerca del video

En esta entrega deben grabar un video, similar a este, explicando la solución a tres problemas, uno por cada entregable
Cada estudiante del equipo, presentará la solución del problema que acuerde con el profesor lapso de $5 \leq t \leq 10$ minutos. Además, el equipo completo tiene que realizar una demostración del funcionamiento de su bot al profesor.

Entregable

- Versión en PDF del informe. El cual debe tener:
  - Presentación(Portada)
  - Tabla de contenido
  - Lista de tablas o figuras

    - Glosario (si se requiere)
    - Resumen o “Abstract”
    - Introducción
    - Problema de investigación, Objetivos y Justificación de la investigación
    - Marco Teórico-conceptual
    - Metodología
    - Resultados
    - Conclusiones
    - Bibliografía
    - Anexos (Código con la correspondiente explicación.)
- Código fuente en LaTeXdel informe, en formato `.zip`.
- Los archivos de Python correspondientes al bot. Adicionalmente, **deben** incluir el archivo *requirements.txt* que lista los paquetes necesarios para la correcta ejecución de sus archivos de Python. Se recomienda ver **este enlace** para consultar sobre el formato de este archivo; pues los paquetes requeridos se instalarán usando el comando `pip install -r requirements.txt`.
- Todo debe ser montado a Brightspace por el capitán del equipo.