UNIVERSIDAD NACIONAL DEL ALTIPLANO

Facultad de Ingeniería Estadística e Informática

Docente: Fred Torres Cruz

Autor : Heydi Pamela Manasaya Quispe

Curso: Métodos de ptimización

Semestre: 5to semestre

Sección: A

Año: 2024

LINK GIT HUB: https://github.com/PamelaManasayaQ/FINESI.git

Actividad N° 2 - Restricciones

Desarrolle los siguientes enunciados de acuerdo con el objetivo planteado en cada problema, ya sea maximización o minimización. Además, implemente una solución computacional utilizando el lenguaje de programación o framework de su preferencia, con el propósito de generar una solución genérica que permita la visualización gráfica de los resultados en todos los casos.

PROBLEMA 1

Un algoritmo necesita procesar datos en lotes. Cada lote requiere x MB de memoria, pero la capacidad total de memoria disponible es de 1024 MB. El algoritmo puede procesar un máximo de 8 lotes. El objetivo es maximizar la cantidad de datos procesados, pero cada lote más allá del quinto reduce su eficiencia en un 20.

Variables:
x:memoria consumida por lote
n:numero de lotes

Restricciones:
* cantidad total de memoria que puede ser usada:

$$\text{n} \cdot x \leq 1024\,\text{MB}$$

* máximo de numero de lotes

$$\text{n} \leq 8$$

Eficiencia:
* lote: 1,2,3,4,5 eficiencia al 100 %

ya que a medida que va avanzando de lote la eficiencia baja en un 20 %:
* lote: 6 al 80 %
* lote: 7 al 60 %
* lote: 8 al 40 %

Función:

$$\begin{cases} n \cdot x & \text{si } n \leq 5 \\ 5x + 0{,}8x + 0{,}6x + 0{,}4x & \text{si } n > 5 \end{cases}$$

Solución:

1.- Si procesamos 8 lotes:

-Memoria requerida:

$8x \leq 1024\,\text{MB} \;\rightarrow\; x \leq \frac{1024}{8} = 128\,\text{MB}$ por lote

-Reemplazando X=128 MB:

P(128;8) = 5(128)+0.8(128)+0.6(128)+0.4(128)

P(128;8) = **870.4** datos procesados en 8 lotes

2.- Si procesamos 5 lotes:

-Memoria requerida:

$5x \leq 1024\,\text{MB} \;\rightarrow\; x \leq \frac{1024}{5} = 204{,}8\,\text{MB}$ por lote

-Reemplazando X=128 MB:

P(204,8;8) = 5*(204,8)

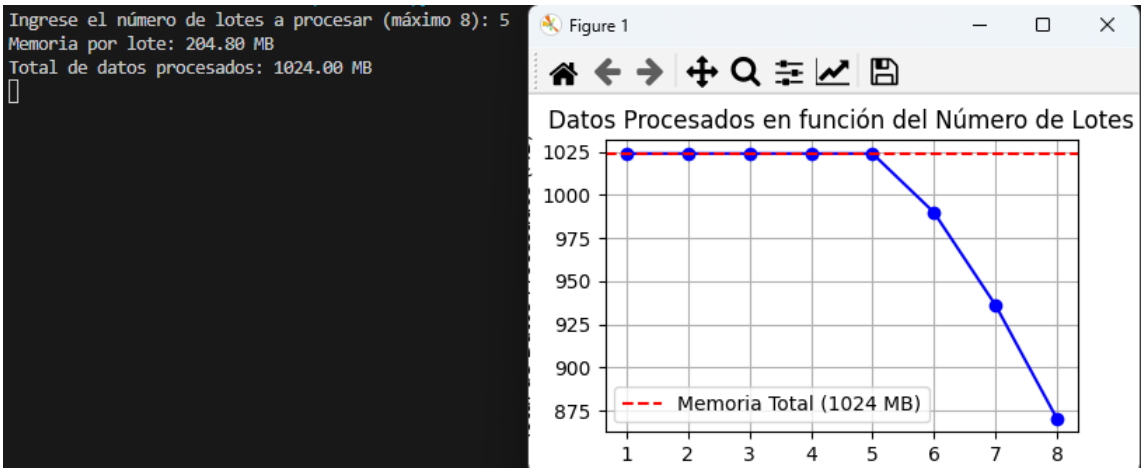
P(204,8;8) = **1024** datos procesados en 5 lotes

Respuesta:
*Procesando 8 lotes, se pueden procesar 870.4 MB de datos.
*Procesando solo 5 lotes (sin penalización de eficiencia), se pueden procesar 1024 MB de datos.

La solucion óptima seria procesar 5 lotes, cada uno de 204.8 MB, lo que permite maximizar la cantidad de datos procesados (1024 MB).

Programa:

```
import matplotlib.pyplot as plt
def calcular_datos_procesados(lotes, memoria_total=1024):
    if lotes <= 0 or lotes > 8:
        return "El número de lotes debe estar entre 1 y 8."
    if lotes <= 5:
        x = memoria_total / lotes
        datos_procesados = lotes * x
        return x, datos_procesados, lotes
    else:
        x = memoria_total / lotes
        datos_procesados = 5 * x
        for i in range(6, lotes + 1):
            eficiencia = 1 - (i - 5) * 0.2
            datos_procesados += eficiencia * x
        return x, datos_procesados, lotes

def graficar_datos(lotes_max=8, memoria_total=1024):
    lotes = list(range(1, lotes_max + 1))
    datos_procesados_list = []
    for n in lotes:
        _, datos_procesados, _ = calcular_datos_procesados(n, memoria_total)
        datos_procesados_list.append(datos_procesados)

    plt.figure(figsize=(10, 6))
    plt.plot(lotes, datos_procesados_list, marker='o', linestyle='-', color='b')
    plt.title('Datos Procesados en función del Número de Lotes')
    plt.xlabel('Número de Lotes')
    plt.ylabel('Total de Datos Procesados (MB)')
    plt.xticks(lotes)
    plt.grid(True)
    plt.axhline(y=memoria_total, color='r', linestyle='--', label='Memoria Total (1024 MB)')
    plt.legend()
    plt.show()
lotes = int(input("Ingrese el número de lotes a procesar (máximo 8): "))
resultado = calcular_datos_procesados(lotes)
print(f"Memoria por lote: {resultado[0]:.2f} MB")
print(f"Total de datos procesados: {resultado[1]:.2f} MB")
graficar_datos()
```



PROBLEMA 2

Un sistema distribuido tiene 20 nodos. Cada nodo puede procesar x peticiones por segundo. El sistema en su conjunto no puede procesar más de 400 peticiones por segundo debido a limitaciones de red. Maximiza el número de peticiones procesadas sin exceder la capacidad de la red.

Variables:
n: número de nodos(n=20)
x: Número de peticiones que cada nodo puede procesar por segundo.
P: Total de peticiones procesadas por el sistema, que se calcula como (P=n*x)

Restricciones:
* cantidad total de memoria que puede ser usada: $P \leq 400$

Función:

$$20^*x \leq 400$$

$$x \leq \left(\frac{400}{20}\right) = 20$$

Esto significa que cada nodo puede procesar un máximo de 20 peticiones por segundo.

Solución:
Ahora que sabemos que x puede ser como maximo 20, podemos calcular el número total de peticiones procesadas:

$$P=20^*20$$
$$P=400$$

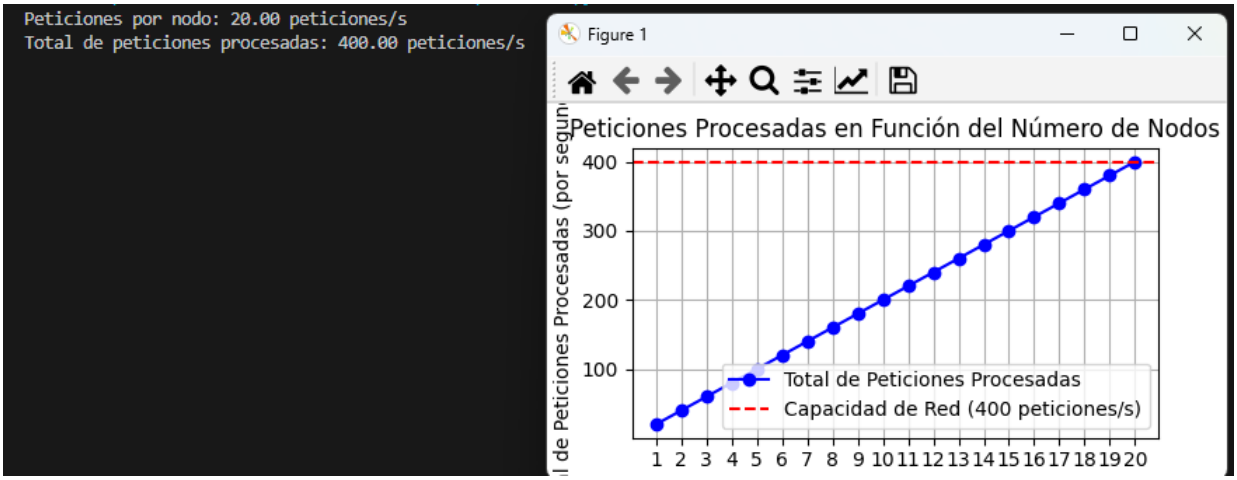
Respuesta:

Para maximizar el número de peticiones procesadas sin exceder la capacidad de la red, cada nodo debe procesar 20 peticiones por segundo, lo que resulta en un total de 400 peticiones por segundo para el sistema completo.

Programa:

```
import matplotlib.pyplot as plt
def calcular_maximo_peticiones(nodos=20, capacidad_total=400):
    max_peticiones_por_nodo = capacidad_total / nodos
    total_peticiones = nodos * max_peticiones_por_nodo
    return max_peticiones_por_nodo, total_peticiones

def graficar_peticiones(nodos=20, capacidad_total=400):
    max_peticiones_por_nodo, total_peticiones = calcular_maximo_peticiones(nodos,
capacidad_total)

    nodos_list = list(range(1, nodos + 1))
    peticiones_por_nodo_list = [capacidad_total / n for n in nodos_list]
    total_peticiones_list = [n * (capacidad_total / nodos) for n in nodos_list]

    plt.figure(figsize=(10, 6))
    plt.plot(nodos_list, total_peticiones_list, marker='o', linestyle='-', color='b', label='Total de
Peticiones Procesadas')
    plt.axhline(y=capacidad_total, color='r', linestyle='--', label='Capacidad de Red (400
peticiones/s)')
    plt.title('Peticiones Procesadas en Función del Número de Nodos')
    plt.xlabel('Número de Nodos')
    plt.ylabel('Total de Peticiones Procesadas (por segundo)')
    plt.xticks(nodos_list)
    plt.grid(True)
    plt.legend()
    plt.show()

max_peticiones_por_nodo, total_peticiones = calcular_maximo_peticiones()
print(f"Peticiones por nodo: {max_peticiones_por_nodo:.2f} peticiones/s")
print(f"Total de peticiones procesadas: {total_peticiones:.2f} peticiones/s")
graficar_peticiones()
```

Acti
Ve a



BLOBLEMA 3

Un script de Python tarda 5x+2 segundos en procesar x datos. Por cada dato adicional, el tiempo de ejecución crece linealmente. Sin embargo, el sistema tiene un límite de tiempo de ejecución de 50 segundos. ¿Cuál es el número máximo de datos que puede procesar el script?

Restricciones:

$$5X+2 \leq 50$$

Solución:
Despejar x:

$$5X+2 \leq 50$$

$$5X \leq 50 - 2$$

$$5X \leq 48$$

$$X \leq \left(\frac{48}{5}\right)$$

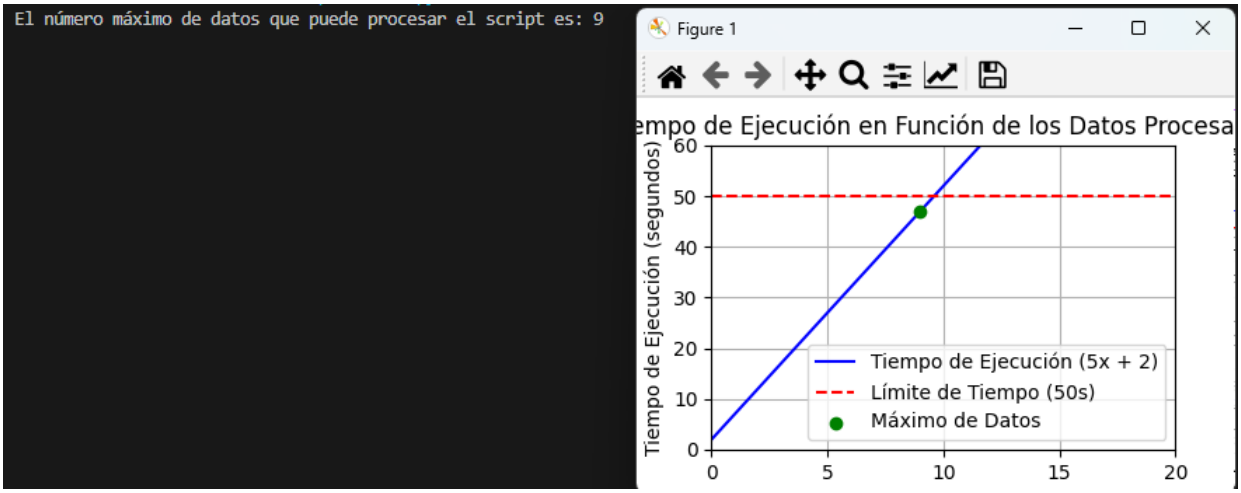
$$X \leq 9{,}6$$

Respuesta:
como x debe ser un número entero, tomamos el valor máximo entero menor o igual a 9.6, que es 9.

El número máximo de datos que puede procesar el script dentro del límite de tiempo de 50 segundos es 9 datos.

Programa:

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

def tiempo_ejecucion(x):
    return 5 * x + 2

limite_tiempo = 50

x_values = np.arange(0, 20, 1)
y_values = tiempo_ejecucion(x_values)

max_datos = 0
for x in x_values:
    if tiempo_ejecucion(x) <= limite_tiempo:
        max_datos = x
    else:
        break

print("El número máximo de datos que puede procesar el script es:", max_datos)

plt.figure(figsize=(10, 6))
plt.plot(x_values, y_values, label='Tiempo de Ejecución (5x + 2)', color='blue')
plt.axhline(y=limite_tiempo, color='red', linestyle='--', label='Límite de Tiempo (50s)')
plt.scatter(max_datos, tiempo_ejecucion(max_datos), color='green', label='Máximo de Datos',
zorder=5)
plt.title('Tiempo de Ejecución en Función de los Datos Procesados')
plt.xlabel('Número de Datos')
plt.ylabel('Tiempo de Ejecución (segundos)')
plt.xlim(0, 20)
plt.ylim(0, 60)
plt.legend()
plt.grid()
plt.show()
```



PROBLEMA 4

Un servidor web procesa x peticiones por segundo, y el uso de CPU sigue la fórmula $2x^2 + 10x$.

La CPU no puede exceder el 80 % de uso. Minimiza el uso de CPU sin caer por debajo del umbral de procesamiento de 10 peticiones por segundo.

Restricciones:
* El uso de CPU no puede superar el 80 %

$$2X^2 + 10X \leq 80$$

*El servidor debe procesar al menos 10 repeticiones por segundo.

$$X \geq 10$$

Solución

Resolviendo la ecuación cuadrática:

$$2x^2 + 10x - 80 \leq 0$$

Simplificamos:

$$x^2 + 5x - 40 \leq 0$$

Resolviendo la ecuación cuadrática:

$$x = \frac{-5 \pm \sqrt{5^2 - 4(1)(-40)}}{2(1)}$$

$$x = \frac{-5 \pm \sqrt{185}}{2}$$

$$x_1 = 4.3 \quad \text{(no factible porque } x \geq 10\text{)}$$

$$x_2 = -9.3 \quad \text{(no factible porque } x \geq 10\text{)}$$

Como las soluciones no cumplen la restricción de $x \geq 10$, evaluamos directamente en $x = 10$:

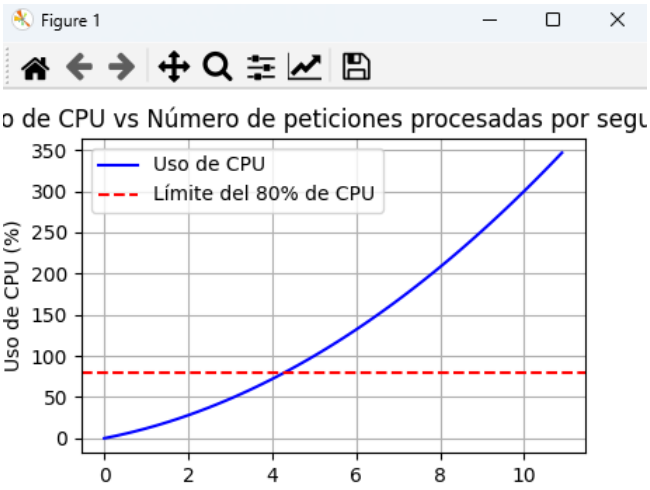
$$U(10) = 2(10)^2 + 10(10) = 300$$

El uso de CPU excede el límite del 80%, por lo que no existe una solución factible que cumpla con ambas restricciones.

Respuesta
Este problema no tiene una soluci´on factible que satisfaga ambas restricciones de procesamiento m´ınimo y l´ımite de uso de CPU. Se necesita ajustar alguna de las restricciones para que el sistema pueda funcionar correctamente.

Programa:

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

limite_uso_cpu = 80
max_peticiones = 20

def uso_cpu(x):
  return 2 * x**2 + 10 * x

x_val = np.arange(0, max_peticiones + 1, 0.1)
uso_cpu_values = uso_cpu(x_val)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_val, uso_cpu_values, label='Uso de CPU', color='b')
ax.axhline(limite_uso_cpu, color='r', linestyle='--', label=f'Límite del {limite_uso_cpu}% de
CPU')
ax.set_title('Uso de CPU vs Número de peticiones procesadas por segundo')
ax.set_xlabel('Número de peticiones procesadas por segundo')
ax.set_ylabel('Uso de CPU (%)')
ax.grid(True)
ax.legend()

plt.show()
```



BLOBLEMA 5

Durante el entrenamiento de un modelo de machine learning, el batch size x afecta el tiempo de entrenamiento

$$T(x) = \left(\frac{1000}{x}\right) + 0{,}1x$$

El tamaño del lote debe estar entre 16 y 128. Encuentra el batch size que minimiza el tiempo de entrenamiento.

Restricciones:

*El tamaño del lote debe estar entre 16 y 128

$$16 \leq x \leq 128$$

Función:

$$T(x) = \left(\frac{1000}{x}\right) + 0{,}1x$$

Solución:

Para encontrar el valor de X que minimiza T(X), tomamos la derivada de T(X)

$$T'(x) = -\frac{1000}{x^2} + 0.1$$

$$-\frac{1000}{x^2} + 0.1 = 0$$
$$\frac{1000}{x^2} = 0.1$$
$$x^2 = 10000$$
$$x = \sqrt{10000} = 100$$

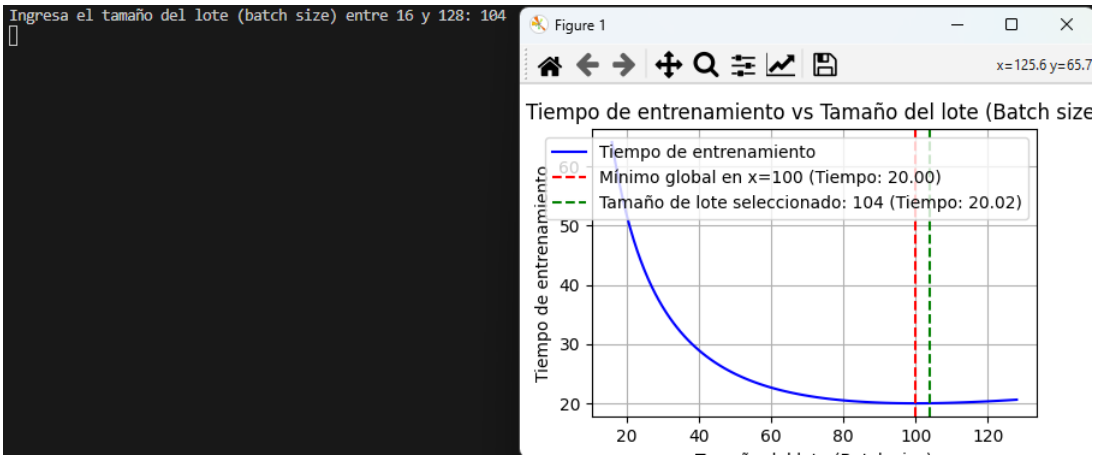
Evaluación en los extremos:

- Para $x = 16$:
$$T(16) = \frac{1000}{16} + 0.1(16) = 64.1$$
- Para $x = 128$:
$$T(128) = \frac{1000}{128} + 0.1(128) = 20.61$$
- Para $x = 100$:
$$T(100) = \frac{1000}{100} + 0.1(100) = 20$$

Respuesta:
El tama˜no del lote que minimiza el tiempo de entrenamiento es x = 100, con un tiempo m´ınimo de entrenamiento de T(100) = 20 segundos.

Programa:

```
import numpy as np
import matplotlib.pyplot as plt
def tiempo_entrenamiento(x):
   return (1000 / x) + 0.1 * x

batch_size = int(input("Ingresa el tamaño del lote (batch size) entre 16 y 128: "))

x_val = np.arange(16, 129, 1)
tiempos = tiempo_entrenamiento(x_val)

min_index = np.argmin(tiempos)
batch_size_min = x_val[min_index]
tiempo_min = tiempos[min_index]
tiempo_actual = tiempo_entrenamiento(batch_size)

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_val, tiempos, label='Tiempo de entrenamiento', color='b')
ax.axvline(x=batch_size_min, color='r', linestyle='--', label=f'Mínimo global en
x={batch_size_min} (Tiempo: {tiempo_min:.2f})')
ax.axvline(x=batch_size, color='g', linestyle='--', label=f'Tamaño de lote seleccionado:
{batch_size} (Tiempo: {tiempo_actual:.2f})')

ax.set_title('Tiempo de entrenamiento vs Tamaño del lote (Batch size)')
ax.set_xlabel('Tamaño del lote (Batch size)')
ax.set_ylabel('Tiempo de entrenamiento')
ax.grid(True)
ax.legend()
plt.show()

print(f'El tamaño de lote que minimiza el tiempo de entrenamiento es: {batch_size_min} con
un tiempo de {tiempo_min:.2f} segundos.')
print(f'El tiempo de entrenamiento para el tamaño de lote seleccionado ({batch_size}) es:
{tiempo_actual:.2f} segundos.')
```



PROBLEMA 6

Un sistema de transmisión de datos tiene un ancho de banda total de 1000 Mbps. Cada archivo que se transmite utiliza x Mbps. El sistema puede transmitir un máximo de 50 archivos a la vez, y cada archivo adicional más allá de 30 reduce el ancho de banda disponible en un 5 % Maximiza el número de archivos transmitidos.

Variables:
n: Número de archivos transmitidos.
x: Ancho de banda disponible, que dependerá de la cantidad de archivos transmitidos.

Restricciones:

*ancho de banda total: El sistema tiene un ancho de banda total de 1000 Mbps.

*Número maximo de archivos: El número máximo de archivos que se pueden transmitir es 50.

*Reducción del ancho de banda: ESi se transmiten más de 30 archivos, el ancho de banda disponible se reduce un 5 % por cada archivo adicional más allá de 30.

-Si se transmiten hasta 30 archivos, el ancho de banda disponible es el total de 1000 Mbps.
-Si se transmiten más de 30 archivos, el ancho de banda disponible se reduce en un 5 % por cada archivo adicional más allá de 30.

Función:

$$\begin{cases} f(x) = x \quad \text{donde } 1 \leq x \leq 30 & \text{si } n \leq 30 \\ b(x) = 1000 \times (1 - 0{,}05 \times (n - 30)) & \text{si } n > 30 \end{cases}$$

Resolución:
Queremos encontrar el valor de n tal que el ancho de banda disponible sea igual a 500 Mbps.

$$\text{b(x)} = 1000 \times (1 - 0{,}05 \times (n - 30))$$

encontrar n tal que el ancho de banda sea=500Mbps

$$500 = 1000 \times (1 - 0{,}05 \times (n - 30))$$

Función:

$$\frac{500}{1000} = 1 - 0{,}05 \times (n - 30)$$

$0.5 = 1 - 0.05 \times (n - 30)$

$0.05 \times (n - 30) = 0{,}5$

$n - 30 = \frac{0{,}5}{0{,}05} = 10$

$n = 30 + 10 = 40$

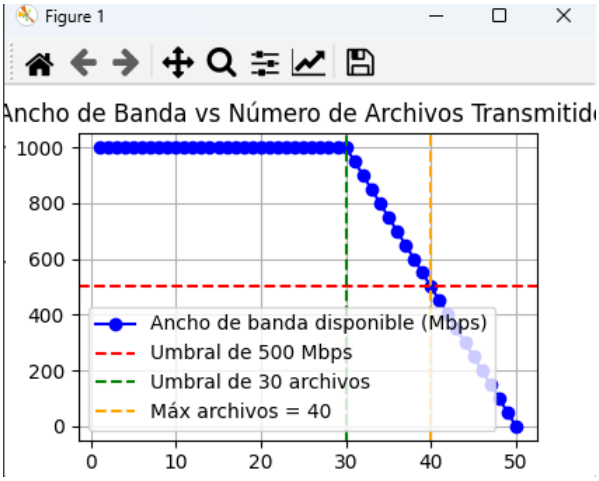
$n = 40$

Respuesta:
El número máximo de archivos que se pueden transmitir antes de que el ancho de banda disponible se reduzca a menos de 500 Mbps es 40 archivos.

Programa:

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

def ancho_banda_disponible(N):
    if N <= 30:
        return 1000
    else:
        return 1000 * (1 - 0.05 * (N - 30))
def max_archivos_transmitidos():
    for N in range(31, 51):
        if ancho_banda_disponible(N) < 500:
            return N - 1
    return 50

N_values = np.arange(1, 51)
ancho_banda_values = [ancho_banda_disponible(N) for N in N_values]
max_archivos = max_archivos_transmitidos()

plt.plot(N_values, ancho_banda_values, label='Ancho de banda disponible (Mbps)', color='b',
marker='o')
plt.axhline(y=500, color='red', linestyle='--', label='Umbral de 500 Mbps')
plt.axvline(x=30, color='green', linestyle='--', label='Umbral de 30 archivos')
plt.axvline(x=max_archivos, color='orange', linestyle='--', label=f'Máx archivos =
{max_archivos}')
plt.title('Ancho de Banda vs Número de Archivos Transmitidos')
plt.xlabel('Número de archivos transmitidos (N)')
plt.ylabel('Ancho de banda disponible (Mbps)')
plt.legend()
plt.grid(True)
plt.show()
```

Activa
Ve a Cor



PROBLEMA 7
Un sistema de colas procesa x trabajos por segundo. La función del tiempo de respuesta T(x)= $\frac{100}{x} + 2x$

Minimiza el tiempo de respuesta del sistema, considerando que el sistema debe procesar al menos 5 trabajos por segundo.

Variables:
x: Número de trabajos que el sistema procesa por segundo.
T(x): Tiempo de respuesta del sistema en segundos.

Restricciones:
El sistema debe procesar al menos 5 trabajos por segundo, lo que significa que

$$x \geq 5$$

Ecuación:

$$T(x) = \frac{100}{x} + 2$$

Solución:

derivada de T(x):

$$T'(x) = -\frac{100}{x^2} + 2$$

Igualamos la derivada a cero para encontrar el punto crítico:

$$-\frac{100}{x^2} + 2 = 0$$

$$\frac{100}{x^2} = 2$$

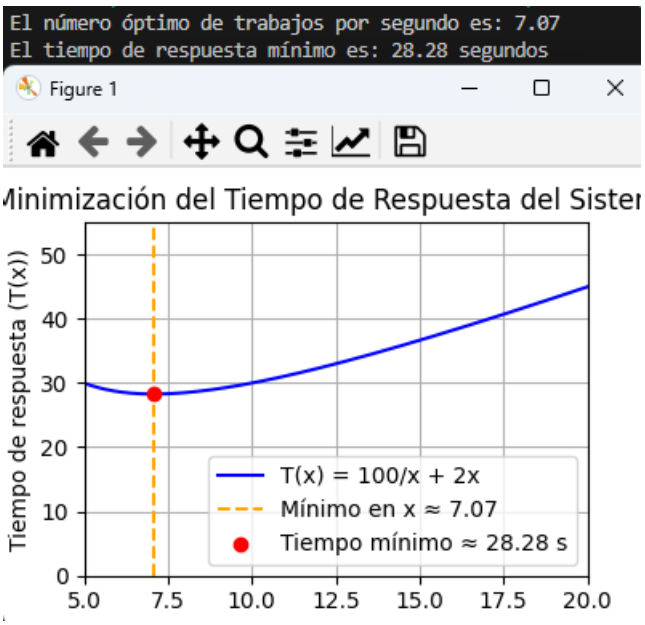
$$x^2 = \frac{100}{2} = 50$$

$$x = \sqrt{50} \approx 7.07$$

Respuesta:
El tiempo de respuesta del sistema es mínimo cuando se procesan aproximadamente 7.07 trabajos por segundo. Este valor cumple la restricción $x \geq 5$, por lo que es la solución óptima.

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

def tiempo_respuesta(x):
    return (100 / x) + 2 * x

def derivada_tiempo_respuesta(x):
    return -100 / (x**2) + 2

x_values = np.linspace(5, 20, 500)
T_values = tiempo_respuesta(x_values)
x_min = np.sqrt(50)
T_min = tiempo_respuesta(x_min)

plt.plot(x_values, T_values, label='T(x) = 100/x + 2x', color='b')
plt.axvline(x=x_min, color='orange', linestyle='--', label=f'Mínimo en x ≈ {x_min:.2f}')
plt.scatter(x_min, T_min, color='red', zorder=5, label=f'Tiempo mínimo ≈ {T_min:.2f} s')
plt.xlim(5, 20)
plt.ylim(0, max(T_values) + 10)

plt.title('Minimización del Tiempo de Respuesta del Sistema')
plt.xlabel('Número de trabajos por segundo (x)')
plt.ylabel('Tiempo de respuesta (T(x))')
plt.legend()
plt.grid(True)
plt.show()
print(f"El número óptimo de trabajos por segundo es: {x_min:.2f}")
print(f"El tiempo de respuesta mínimo es: {T_min:.2f} segundos")
```

```
El número óptimo de trabajos por segundo es: 7.07
El tiempo de respuesta mínimo es: 28.28 segundos
```



PROBLEMA 8

El entrenamiento de un modelo de deep learning en una GPU consume x unidades de energía por lote. El objetivo es maximizar el tamaño del lote x, pero el consumo de energía total no puede exceder las 200 unidades por segundo, y cada lote adicional más allá del 10 reduce el rendimientoen un 10 %.

Variables:
x: tamaño de lote
E(x): Energía consumida por segundo, limitada a 200 unidades por segundo
R(x): Rendimiento del modelo en función de x.

Restricciones:
*El consumo de energía por segundo no puede exceder 200 unidades, por lo que:

$$E(X) \leq 200$$

*Para x$\leq$ 10

el rendimiento es constante, pero para cada lote adicional más allá de 10, el rendimiento disminuye en un 10 %

Función:

$$E(x) = \begin{cases} x & \text{si } x \leq 10 \\ x \cdot (1 + 0.1 \cdot (x - 10)) & \text{si } x > 10 \end{cases}$$

x.E(x)$\leq$ 200

Solución:

Caso 1: $x \leq 10$

$$x^2 \leq 200 \quad \Rightarrow \quad x \leq \sqrt{200} \approx 14.14$$

Caso 2: $x > 10$

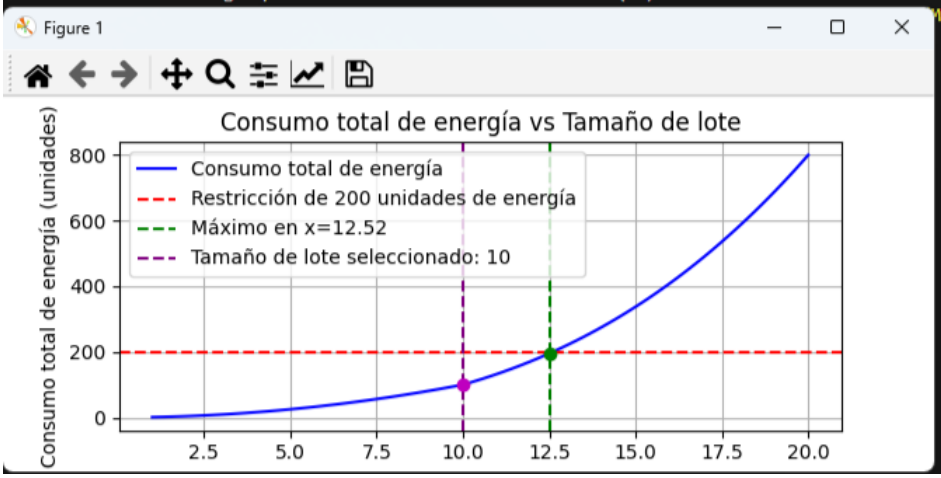
$$x \cdot x \cdot (1 + 0.1(x - 10)) \leq 200$$

Respuesta:
El tamaño ideal del lote x depende de si n es menor o igual a 10, o si es mayor que 10. En el primer caso, se hace el lote más grande posible sin perder calidad; en el segundo caso, la calidad baja y se necesita ajustar la energía para no pasarse del límite.

Programa:

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

print('Consumo Total de Energía vs Tamaño de Lote')
print('')

tamaño_lote = int(input('Selecciona el tamaño del lote (x) (entre 1 y 20): '))
def energia_consumida(x):
  if x <= 10:
    return x
  else:
    return x * (1 + 0.1 * (x - 10))
def consumo_total(x):
  return x * energia_consumida(x)
x_values = np.linspace(1, 20, 100)
consumo = np.array([consumo_total(x) for x in x_values])
x_max = np.max(x_values[consumo <= 200])
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x_values, consumo, label='Consumo total de energía', color='b')
ax.axhline(y=200, color='r', linestyle='--', label='Restricción de 200 unidades de energía')
ax.axvline(x=x_max, color='g', linestyle='--', label=f'Máximo en x={x_max:.2f}')
ax.plot(x_max, consumo_total(x_max), 'go')
ax.axvline(x=tamaño_lote, color='purple', linestyle='--', label=f'Tamaño de lote seleccionado:
{tamaño_lote}')
ax.plot(tamaño_lote, consumo_total(tamaño_lote), 'mo')
ax.set_title('Consumo total de energía vs Tamaño de lote')
ax.set_xlabel('Tamaño de lote (x)')
ax.set_ylabel('Consumo total de energía (unidades)')
ax.grid(True)
ax.legend()
plt.show()

print(f"El tamaño de lote máximo que satisface la restricción de 200 unidades de energía es
aproximadamente: {x_max:.2f}.")
print(f"Consumo total de energía para el tamaño de lote seleccionado ({tamaño_lote}):
{consumo_total(tamaño_lote):.2f} unidades.")
```

```
Selecciona el tamaño del lote (x) (entre 1 y 20): 10
El tamaño de lote máximo que satisface la restricción de 200 unidades de energía es aproximadamente: 12.52.
Consumo total de energía para el tamaño de lote seleccionado (10): 100.00 unidades.
```



PROBLEMA 9

Una empresa almacena datos en la nube. El costo de almacenamiento por TB es de 50+5x dólares, donde x es la cantidad de TB de almacenamiento utilizado. La empresa tiene un presupuesto de 500 dólares. Maximiza la cantidad de datos almacenados sin exceder el presupuesto.

Variables:

x: Cantidad de terabytes (TB) de almacenamiento utilizado.

Restricciones:

La empresa tiene un presupuesto máximo de 500 dólares, por lo que el costo total no puede exceder este valor:

$$50x+5x^2 \leq 500$$

Función:

$$C(x)=50 + 5x \text{ (en dolares por TB almacenado)}$$

Solución:

$$50x+5x^2 \leq 500$$

dividir entre 5 ambos lados

$$10x+x^2 \leq 100$$

$$x^2 + 10x - 100 \leq 0$$

Ahora resolvemos esta ecuación cuadrática utilizando la fórmula general:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Donde:

- $a = 1$
- $b = 10$
- $c = -100$

Calculamos el discriminante:

$$\Delta = b^2 - 4ac = 10^2 - 4(1)(-100) = 100 + 400 = 500$$

Sustituyendo en la fórmula general:

$$x = \frac{-10 \pm \sqrt{500}}{2}$$

$$x = \frac{-10 \pm 22.36}{2}$$

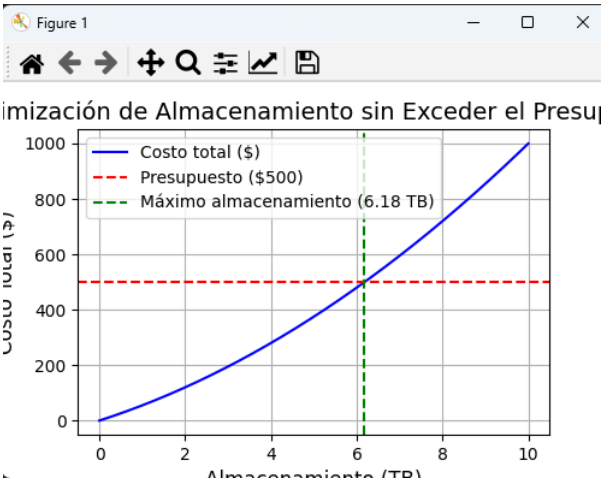
Obtenemos dos soluciones:

$$x_1 = \frac{-10 + 22.36}{2} = 6.18$$

Respuesta:
La empresa puede almacenar un máximo de 6 TB sin exceder su presupuesto de 500 dólares.

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

def costo_total(x):
  return 50 * x + 5 * x**2

presupuesto = 500
x_values = np.linspace(0, 10, 400)

costo_values = costo_total(x_values)

plt.figure(figsize=(8, 6))
plt.plot(x_values, costo_values, label='Costo total ($)', color='blue')
plt.axhline(y=presupuesto, color='red', linestyle='--', label=f'Presupuesto ($500)')
plt.axvline(x=6.18, color='green', linestyle='--', label=f'Máximo almacenamiento (6.18 TB)')

plt.title('Maximización de Almacenamiento sin Exceder el Presupuesto', fontsize=14)
plt.xlabel('Almacenamiento (TB)', fontsize=12)
plt.ylabel('Costo Total ($)', fontsize=12)
plt.legend()
plt.grid(True)
plt.show()
```

Activar
Ve a Con



PROBLEMA 10

Un sistema de mensajería tiene una latencia L(x) = 1002x, donde x es el número de mensajes por segundo. La latencia no puede ser inferior a 20 ms debido a restricciones del protocolo. Maximiza el número de mensajes enviados sin que la latencia caiga por debajo de este límite.

Variables:

x: Número de mensajes enviados por segundo.

Restricciones:

La latencia no puede ser inferior a 20 ms debido a las restricciones del protocolo, por lo tanto:

L(x)$\geq$ 20

Sustituimos la función de latencia en la restricción:

100 2x$\geq$ 20

Solución:

Resolvemos para x:

$$100 - 2x \geq 20$$

Restamos 100 a ambos lados:

$$-2x \geq 20 - 100$$

$$-2x \geq -80$$

Dividimos ambos lados entre -2 (cambiando el signo de la desigualdad):

$$x \leq 40$$

Respuesta:

Para mantener la latencia por encima de 20 ms, el sistema puede enviar un máximo de 40 mensajes por segundo.

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

def latencia(x):
   return 100 - 2 * x

latencia_minima = 20
x_values = np.linspace(0, 50, 400)
latencia_values = latencia(x_values)

plt.figure(figsize=(8, 6))
plt.plot(x_values, latencia_values, label='Latencia (ms)', color='blue')
plt.axhline(y=latencia_minima, color='red', linestyle='--', label=f'Latencia mínima (20 ms)')
plt.axvline(x=40, color='green', linestyle='--', label=f'Máximo mensajes (40
mensajes/segundo)')

plt.title('Maximización de Mensajes sin Exceder la Latencia Mínima', fontsize=14)
plt.xlabel('Mensajes por segundo', fontsize=12)
plt.ylabel('Latencia (ms)', fontsize=12)
plt.legend()
plt.grid(True)

plt.show()
```

Activar \
Ve a Confic