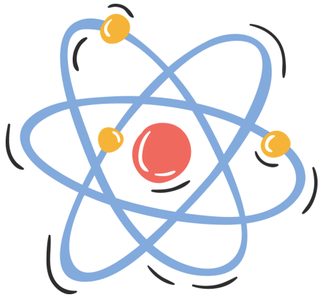
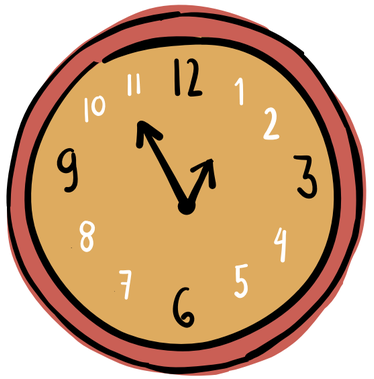
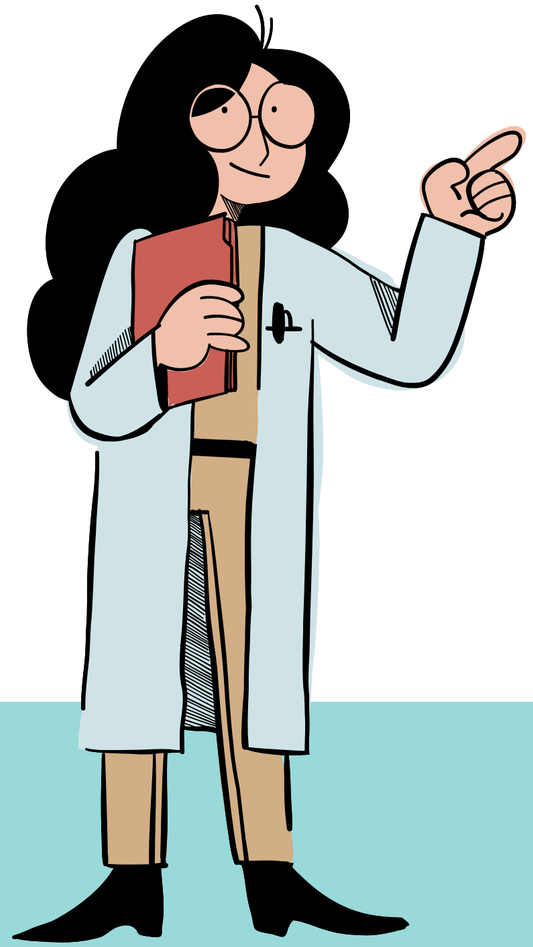


Departamento de ciencias CCM

Optimización de la programación del departamento de ciencias

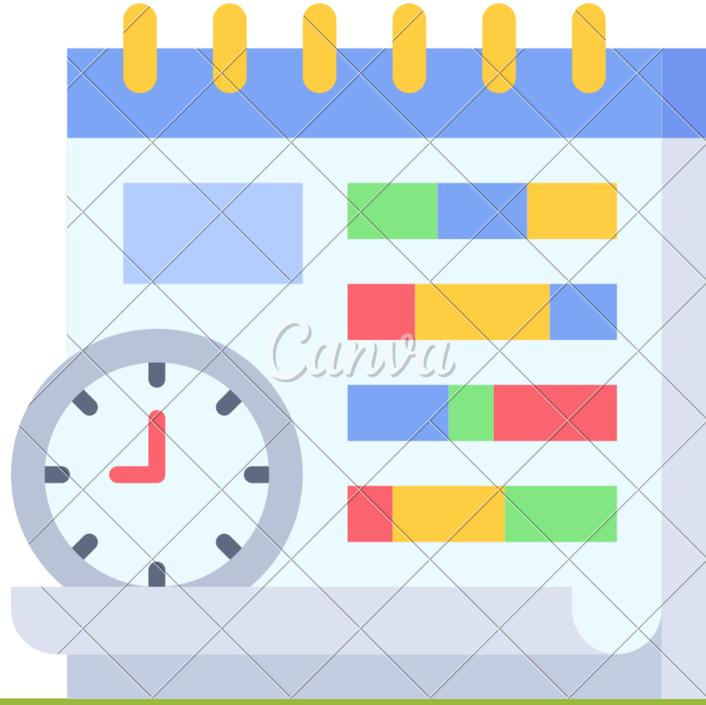




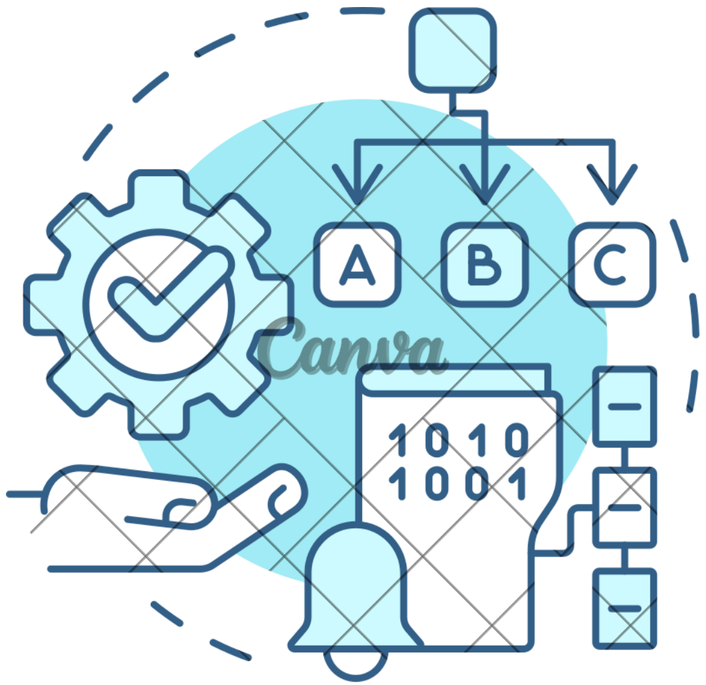
Omar Rodríguez Montiel A01750836

Introduccion

El desafío de la programación de horarios en las universidades consiste en la distribución eficiente de recursos, teniendo en cuenta diversas restricciones como la disponibilidad de profesores, aulas, capacidad de estudiantes, demanda de cursos y cambios de programas académicos. Aunque se han realizado intentos para abordar este desafío de manera empírica, la creación de software especializado, tal como lo realizó Bill Gates en 1971, puede contribuir significativamente a optimizar este proceso.

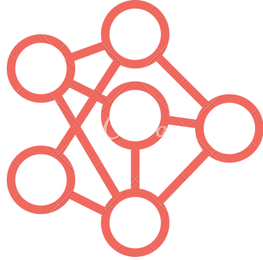


Planteamiento del problema



1. Analizar y definir los costos y la disponibilidad de horarios y materias para cada profesor
2. Definir y formular la función objetivo, sus variables y las restricciones
3. Programar un código en python usando la librería pulp (programación lineal), para obtener la solución.
4. Interpretar los resultados

Objetivo



El desafío radica en la optimización de la programación en el departamento de ciencias CCM. Se busca crear un algoritmo que, mediante una interfaz, pueda ordenar la programación de 9 asignaturas según la tabla suministrada, con el propósito de reducir los costos del departamento.

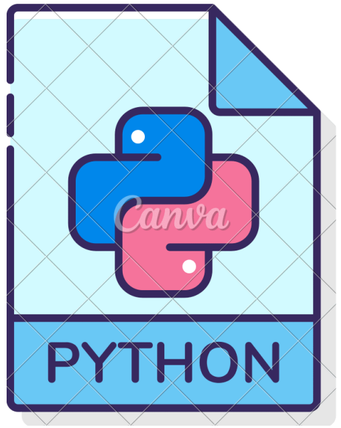
Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Disponibilidad de horario de cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A	k	1	a	jsk	123	10	10	20	50
B	j	2	b	om	246	20	5	20	50
C	o	3	c	rns	345	20	5	20	40
D	r	4	d	psq	689	10	5	18	40
E	p	5	e	jon	986	20	10	18	50
F	m	6	f	orq	135	10	10	18	50
G	n	7	g	rpk	468	10	5	16	40
H	s	8	h	pjm	136	20	5	16	40
I	q	9	i	jor	367	20	5	16	50

Tabla 1. Materias departamento de ciencias CCM

Fundamento teórico

Libreria PuLP en Python

Es una herramienta potente y versátil que posibilita la descripción y solución de problemas de optimización mediante modelos matemáticos.



Programación lineal

Se trata de una técnica matemática empleada para optimizar una función objetivo lineal, sujeta a un conjunto de restricciones lineales. En su formulación más básica, un problema de Programación Lineal consta de dos elementos principales: una función objetivo, que se presenta como una ecuación lineal a maximizar o minimizar, y un conjunto de restricciones, que se manifiestan como ecuaciones o desigualdades lineales que restringen los valores de las variables de decisión.

Max z:

Función objetivo: $c_1x_1 + c_2x_2 + \dots + c_nx_n$

sujeto a:

Restricciones:

$a_{11}x_1+a_{12}x_2+\dots+a_{1n}x_n \le b_1$

$a_{21}x_1+a_{22}x_2+\dots+a_{2n}x_n \le b_2$

.

.

.

$a_{m1}x_1+a_{m2}x_2+\dots+a_{mn}X_n \le b_m$

$x1, x2,\dots, xn \ge 0$ — Restricción de no negatividad

x_i Variables de decisión. i = 1, 2, ..., n

c_i Coeficientes de costos (aunque no necesariamente representen un costo, sino una distancia u otra cosa)

a_{ij} Coeficientes tecnológicos (conocidos)

b_j Recursos (conocidos)

Modelo matematico general

Formulación del problema

Variables de decisión

Podríamos definir una variable binaria para cada posible asignación de profesor, materia, horario y salón. Por ejemplo:

- podría ser 1 si el profesor i está enseñando, la materia j en el horario k, en el salón l , y 0 en caso contrario.

$$x_{ijkl}$$

Función objetivo

El objetivo es minimizar el costo total, que podría ser la suma de los costos de los profesores, las materias y los salones. Esto podría expresarse como:

$$\min \sum_{i,j,k,l} c_{ijkl} x_{ijkl}$$

Donde la función : $c_{ijkl} x_{ijkl}$ (es el costo de asignar al profesor i)

Restricciones

Necesitamos asegurarnos de que cada materia se imparta una vez, cada profesor tenga un horario que no se solape y cada salón se use una vez por horario. Además, cada profesor tiene restricciones específicas sobre las materias que pueden enseñar y sus horarios disponibles. Estas restricciones podrían expresarse como:

Para cada materia j, debe haber exactamente un profesor i, un horario k y un salón l tal que:

$$x_{ijkl} = 1$$

Pasos para la solución

Paso 1

El primer paso en el código fue importar la tabla de excel

```
[19] horario_Excel = pd.read_excel('/content/horarioprb.xlsx')
     horario_Excel
```

	Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Disponibilidad de horario de cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
0	A	k	1	a	jsk	123	10	10	20	50
1	B	j	2	b	om	246	20	5	20	50
2	C	o	3	c	ms	345	20	5	20	40
3	D	r	4	d	psq	689	10	5	18	40
4	E	p	5	e	jon	986	20	10	18	50
5	F	m	6	f	orq	135	10	10	18	50
6	G	n	7	g	rpk	468	10	5	16	40
7	H	s	8	h	pjm	136	20	5	16	40
8	I	q	9	i	jor	367	20	5	16	50

Pasos para la solución

Paso 1

Y definir los costos y la disponibilidad de horarios y materias para cada profesor.

```
materias_p = horario_Excel['Materias que puede dar cada profesor'].tolist()

Materias_profes = {}
#que una las materias como listas ejemplo {'A': ['m1', 'm2', 'm3']}
for i in range(len(profesores)):
    Materias_profes[profesores[i]] = [j for j in materias_p[i]]

Materias_profes
```

```
{'A': ['j', 's', 'k'],
 'B': ['o', 'm'],
 'C': ['r', 'n', 's'],
 'D': ['p', 's', 'q'],
 'E': ['j', 'o', 'n'],
 'F': ['o', 'r', 'q'],
 'G': ['r', 'p', 'k'],
 'H': ['p', 'j', 'm'],
 'I': ['j', 'o', 'r']}
```

```
[22] dispon_horario = horario_Excel['Disponibilidad de horario de cada profesor'].tolist()

Disponibilidad_profes_horarios = {}
#que una los horarios queden de la siguiente manera: {'A': [1,2,3]}
for i in range(len(profesores)):
    Disponibilidad_profes_horarios[profesores[i]] = [j for j in str(dispon_horario[i])]
#convertir los horarios a enteros
for i in Disponibilidad_profes_horarios:
    Disponibilidad_profes_horarios[i] = [int(j) for j in Disponibilidad_profes_horarios[i]]

Disponibilidad_profes_horarios
```

```
{'A': [1, 2, 3],
 'B': [2, 4, 6],
 'C': [3, 4, 5],
 'D': [6, 8, 9],
 'E': [9, 8, 6],
 'F': [1, 3, 5],
 'G': [4, 6, 8],
 'H': [1, 3, 6],
 'I': [3, 6, 7]}
```

```
23] costos_profe = horario_Excel['Costos Profesores'].tolist()
    costos_profes = {
        profesores[i]: costos_profe[i] for i in range(len(profesores))
    }
    costos_profes
```

```
{'A': 10,
 'B': 20,
 'C': 20,
 'D': 10,
 'E': 20,
 'F': 10,
 'G': 10,
 'H': 20,
 'I': 20}
```

```
24] costos_materia = horario_Excel['Costos Materias'].tolist()
    costos_materias = {
        materias[i]: costos_materia[i] for i in range(len(materias))
    }
    costos_materias
```

```
{'k': 10, 'j': 5, 'o': 5, 'r': 5, 'p': 10, 'm': 10, 'n': 5, 's': 5, 'q': 5}
```

Pasos para la solución

Paso 2

El siguiente paso fue implementar la función que le permite al usuario definir profesores que no están disponibles

```python
profes_eliminar = input('Ingrese los profesores a eliminar separados por comas: ')

if profes_eliminar == '':
    profes_eliminar = []
else:
    profes_eliminar = profes_eliminar.split(',')
    profes_eliminar = [profesor.strip() for profesor in profes_eliminar]
    #actualizar los diccionarios
    for profesor in profes_eliminar:
        Materias_profes.pop(profesor)
        Disponibilidad_profes_horarios.pop(profesor)
        costos_profes.pop(profesor)

    print(Materias_profes)
    print(Disponibilidad_profes_horarios)
    print(costos_profes)
    #eliminar d ela lista de profesores
    profesores = [profesor for profesor in profesores if profesor not in profes_eliminar]
```

```
Ingrese los profesores a eliminar separados por comas: A
{'B': ['o', 'm'], 'C': ['r', 'n', 's'], 'D': ['p', 's', 'q'], 'E': ['j', 'o', 'n'], 'F': ['o', 'r', 'q'], 'G': ['r', 'p', 'k'], 'H': ['p', 'j', 'm'], 'I': ['j', 'o', '
{'B': [2, 4, 6], 'C': [3, 4, 5], 'D': [6, 8, 9], 'E': [9, 8, 6], 'F': [1, 3, 5], 'G': [4, 6, 8], 'H': [1, 3, 6], 'I': [3, 6, 7]}
{'B': 20, 'C': 20, 'D': 10, 'E': 20, 'F': 10, 'G': 10, 'H': 20, 'I': 20}
```

Pasos para la solución

Paso 2

Y horarios que no están disponibles.

```
[28] horarios_eliminar = input('Ingrese los horarios a eliminar separados por comas: ')

    if horarios_eliminar == '':
        horarios_eliminar = []
    else:
        horarios_eliminar = horarios_eliminar.split(',')
        horarios_eliminar = [horario.strip() for horario in horarios_eliminar]
        horarios_eliminar = [int(horario) for horario in horarios_eliminar]
        #actualizar los diccionarios
        for profesor in Disponibilidad_profes_horarios:
            Disponibilidad_profes_horarios[profesor] = [horario for horario in Disponibilidad_profes_horarios[profesor] if horario not in horarios_eliminar]
        print(Disponibilidad_profes_horarios)
        #eliminar de la lista de horarios
        horarios = [horario for horario in horarios if horario not in horarios_eliminar]
        print(horarios)

    Ingrese los horarios a eliminar separados por comas: 5,6
```

Pasos para la solución

Paso 3 $f(x) = ax^2 + bx + c$

Ya teniendo esto, procedemos a definir el problema de Programación Lineal, sus variables, función objetivo.

```
[29] from pulp import *

     prob = LpProblem("Horarios",LpMinimize)
```

```
[30] from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

Variables

```
[31] profes_materias_salon_horario = LpVariable.dicts("Profes_Materias_Salon_Horario",(profesores,materias,salones,horarios),0,1,LpBinary)
```

Funcion Objetivo

```
[32] prob += lpSum([profes_materias_salon_horario[p][m][s][h]*costos_profes[p] + profes_materias_salon_horario[p][m][s][h]*costos_materias[m] + profes_materias_salon_horar
```

Pasos para la solución

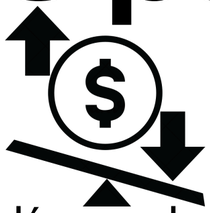
Paso 3 $f(x) = ax^2 + bx + c$

Y las restricciones.

```
[33] #una materia solo se imparte una vez
     for p in profesores:
         for m in materias:
             prob += lpSum([profes_materias_salon_horario[p][m][s][h] for h in horarios for s in salones]) <= 1

     #un profesor solo puede dar una materia a la vez
     for p in profesores:
         for h in horarios:
             for s in salones:
                 prob += lpSum([profes_materias_salon_horario[p][m][s][h] for m in materias]) <= 1

     #todas las materias se imparten una sola vez
     for m in materias:
         prob += lpSum([profes_materias_salon_horario[p][m][s][h] for p in profesores for h in horarios for s in salones]) == 1

     #restriccion de que materias pueden dar los profesores ejemplo profesor A solo puede dar las materias j,s,k
     for p in profesores:
         for m in materias:
             if m not in Materias_profes[p]:
                 prob += lpSum([profes_materias_salon_horario[p][m][s][h] for s in salones for h in horarios]) == 0

     #restriccion de horarios disponibles para los profesores ejemplo profesor A solo puede dar clases en los horarios 1,2,3
     for p in profesores:
         for h in horarios:
             if h not in Disponibilidad_profes_horarios[p]:
                 prob += lpSum([profes_materias_salon_horario[p][m][s][h] for m in materias for s in salones]) == 0

     #restriccion no se puede usar un salon en el mismo horario
     for s in salones:
         for h in horarios:
             prob += lpSum([profes_materias_salon_horario[p][m][s][h] for p in profesores for m in materias]) <= 1
```

Pasos para la solución

Paso 4



Las siguientes líneas de código son las que proporcionan su solución y te muestran el costo total.

```
prob.solve()
#print("Status:", LpStatus[prob.status])

""" for v in prob.variables():
    print(v.name, "=", v.varValue) """
#solo imprimir los valores que son 1
for v in prob.variables():
    if v.varValue == 1:
        print(v.name, "=", v.varValue)
```

```
Profes_Materias_Salon_Horario_B_m_g_4 = 1.0
Profes_Materias_Salon_Horario_D_p_c_8 = 1.0
Profes_Materias_Salon_Horario_D_q_h_9 = 1.0
Profes_Materias_Salon_Horario_D_s_g_9 = 1.0
Profes_Materias_Salon_Horario_E_j_d_9 = 1.0
Profes_Materias_Salon_Horario_E_n_c_9 = 1.0
Profes_Materias_Salon_Horario_F_o_c_3 = 1.0
Profes_Materias_Salon_Horario_G_k_g_8 = 1.0
Profes_Materias_Salon_Horario_G_r_d_8 = 1.0
```

Costo total

```
[35] print("Costo total = ", value(prob.objective))
```

```
Costo total =  690.0
```

Pasos para la solución

Paso 5

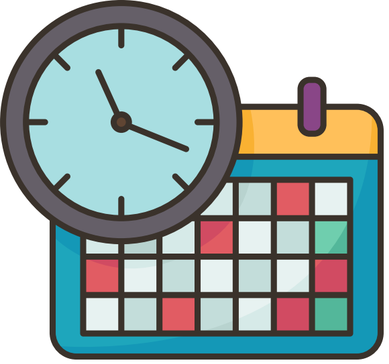
Y por último, te imprime la solución y te crea un archivo de excel con la tabla final que muestra que profesor imparte cada materia, en que salón y horario se imparte la clase, el costo de cada materia impartida (suma de materia, horario, profesor y salon) y el costo total del problema.

```
[37] # crear un archivo excel con df
     from openpyxl import load_workbook

     df.to_excel('solucion_horario.xlsx', index=False)
     wb = load_workbook('solucion_horario.xlsx')
     ws = wb.active
     ws['K3'] = 'Costo Total:'
     ws['L3'] = value(prob.objective)

     wb.save('solucion_horario.xlsx')
     wb.close()
```

Abrir el archivo

```
import os

#Abrir "solucion_horario.xlsx"
os.system('start excel solucion_horario.xlsx')
```

```
32512
```

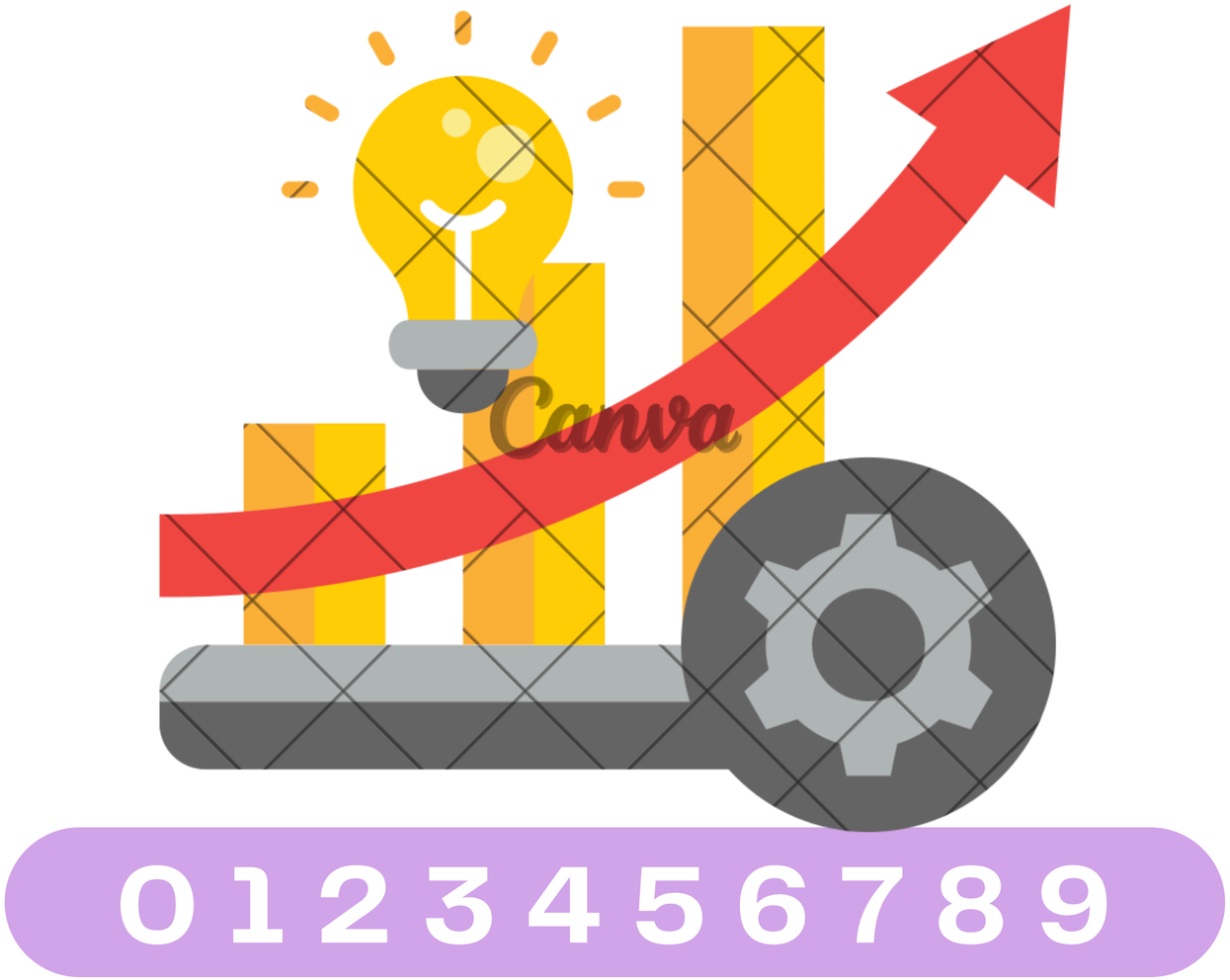


Solución del problema

Profesor	Materia	Salon	Horario	osto Profes	psto Mater	Costo Salon	psto Horari	Costo			
A	j	h	3	10	5	40	20	75			
D	p	h	8	10	10	40	16	76		Costo Tota	682
D	q	c	9	10	5	40	16	71			
D	s	g	9	10	5	40	16	71			
E	n	g	8	20	5	40	16	81			
F	o	c	5	10	5	40	18	73			
G	k	c	8	10	10	40	16	76			
G	r	d	8	10	5	40	16	71			
H	m	g	6	20	10	40	18	88			

Áreas de mejora

Consideramos que el código se ejecutó de manera óptima tras implementar las modificaciones en los profesores y horarios. La principal área de oportunidad radica en la mejora de la visualización y estética de los datos, a pesar de que actualmente se presentan de forma clara y comprensible.



Conclusiones

Al final, hemos alcanzado nuestro objetivo de determinar el precio óptimo para asignar maestros a sus respectivos horarios, asignaturas y aulas. Asimismo, logramos hallar la solución óptima al considerar posibles limitaciones, como la disponibilidad de maestros o de horarios. Para lograrlo, implementamos la librería "Pulp" de Python y aplicamos los conocimientos adquiridos durante el curso en programación lineal, formulación de restricciones, y la habilidad de comprender y adaptar un problema complejo como este, permitiéndonos resolverlo mediante programación lineal.

Referencias

- (1). Lifeder. (s.f.). Programación lineal: para qué sirve, modelos, restricciones, aplicaciones. Recuperado de https://www.lifeder.com/programacion-lineal/

- (2). QuestionPro. (s.f.). Programación lineal: Qué es, usos y pasos para realizarla. Recuperado de https://www.questionpro.com/blog/es/programacion-lineal/

¡Gracias por su atención!