

# ***Programación Avanzada***

## **IIC2233 2024-2**

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Lucas Van Sint Jan - Francisca Cattán



# Experiencia 6: DCCine-graph

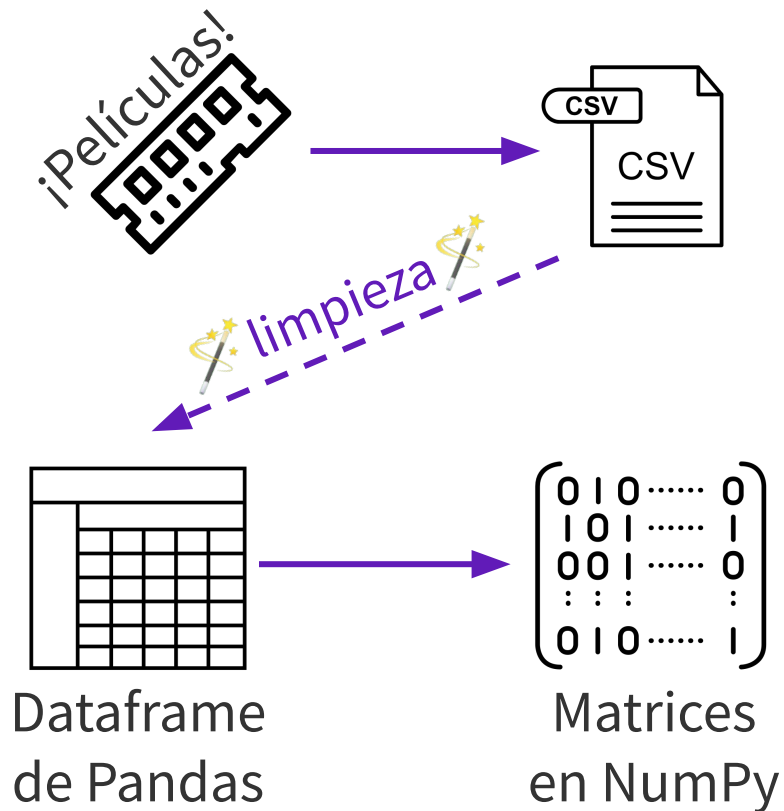
Algunos días del semestre, el Capítulo de Computación celebra un esperado evento que reúne a decenas de estudiantes, cansados de estudiar y rendir controles y exámenes (o hacer tareas). ¡Este es el 🍿 DCCine 🍿!

Se te ha encargado refinar la base de datos existente, y además utilizar tus nuevos conocimientos en grafos para realizar la recomendación sobre la decisión más difícil: **¿qué peli vemos?**

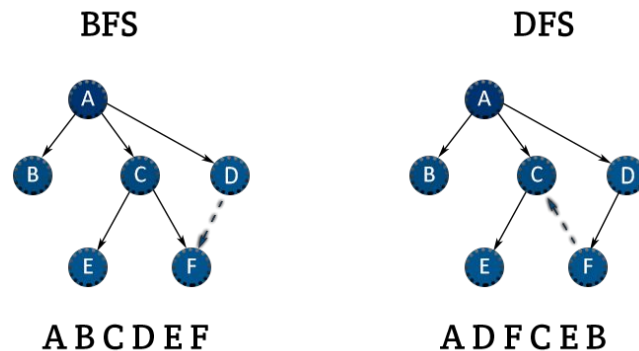
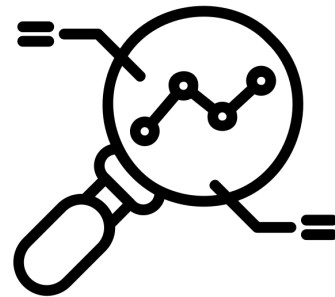
---

¿Qué queremos  
lograr?

# Base de datos



# Recomendación con grafos






# Resumen ejecutivo 🤔 :

1. Convertiremos un archivo CSV en un Dataframe de Pandas, y limpiaremos los datos usando SÓLO expresiones regulares.
2. Trabajaremos con matrices de NumPy para generar matrices de adyacencia según el género de las películas.
3. Crearemos nuestro propio grafo a partir de una matriz, y practicaremos con 2 sistemas de recomendación basados en algoritmos de búsqueda en grafos: BFS y DFS.

Al igual que con las experiencias anteriores, trabajaremos **en partes**. Al final del día se publicará la solución donde se detalla cada método del código correctamente programado, y en caso de tener errores, cuáles eran estos.

¿Qué tenemos?

# ¿Qué tenemos?

 Atributo  
 Método  
 Incompleto o no implementado

base\_datos.csv

main.py

```
! cargar_y_limpiar()  
! generar_matriz_genero()  
! computar_matriz_adyacencia()  
✓ crear_lista_adyacencia()  
! crear_grafo()
```

grafo.py

```
A lista_adyacencia: dict  
  
✓ agregar_nodo()  
✓ agregar_arista(nodo)  
✓ agregar_estilo(nodo1,nodo2)  
✓ recomendacion_bfs(peli_inicial:str,  
dataframe, profundidad_max=2)  
✓ recomendacion_dfs(peli_inicial:str,  
dataframe, profundidad_max=2)
```

**¡A trabajar!**



# Parte 1: Cargar y limpiar

## Actualmente...

- Sólo tenemos el CSV y la función **cargar\_y\_limpiar** está incompleta.

## Resultado esperado tras terminar

- Dataframe de películas con las columnas: Nombre, Género, Rating.



*DataFrame de películas:*

	Nombre	Género	Rating
0	Her	Drama Romance	9.5

*(...)*

*Presiona enter para seguir*

# Parte 1: Cargar y limpiar

1. Observaremos los datos . Algunos nombres de películas están extraños, y hay columnas que parecen no pertenecer a este dataset .
2. Crearemos un Dataframe usando Pandas, y usaremos esta herramienta para dejar sólo los datos importantes (*¡ojo con los tipos de datos!*).
3. Limpiaremos la data de los nombres de películas SÓLO con patrones de regex.

*DataFrame de películas:*

	Nombre	Género	Rating
0	Her	Drama Romance	9.5
(...)			

*Presiona enter para seguir*

**¡A programar!**  

# Parte 2.1: Matriz de géneros

## Actualmente...

- La función **generar\_matriz\_genero** está incompleta.

## Resultado esperado tras terminar

- Crearemos un nuevo Dataframe de pandas que contenga la información de los géneros de cada película. Cada fila es una película, y tiene un indicador binario.

(...)

*Matriz de géneros:*

	<i>Acción</i>	<i>Aventura</i>	<i>Comedia</i>	<i>Crimen</i>	<i>Drama</i>	<i>Biografía</i>	<i>Romance</i>	<i>Scifi</i>	<i>Thriller</i>
0	0	1	0	0	1	0	0	0	0

(...)

*Presiona enter para seguir*

# Parte 2.2: ¿Matrices? ¿Para qué?

## Actualmente...

- Tenemos la matriz de géneros, pero esta está en formato Dataframe

## Resultado esperado tras terminar

- Crearemos un array de NumPy a partir de la matriz, y lo usaremos para buscar los géneros en común entre películas.

*Matriz de Adyacencia basada en Género:*

```
[[0 0 0 0 1 1 1 0 0 0 1 1 1 1 0 1 0 0 0 0 2 0 0]
```

```
[0 0 0 2 0 0 0 1 1 1 1 0 0 0 2 0 0 0 0 1 0 1 2]
```

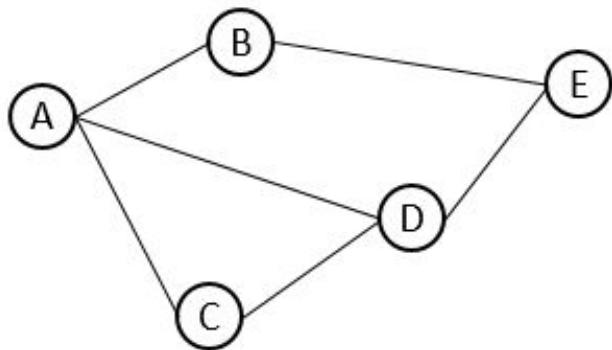
(...)

*Presiona enter para seguir*

## Parte 2.2: ¿Matrices? ¿Para qué?

**Espera pero, ¿qué es una matriz de adyacencia y para qué la quiero?**

- La matriz de adyacencia de géneros nos ayuda a representar los datos, pensando en que cada película es un nodo, y las aristas representan los géneros compartidos entre ellas.



	A	B	C	E	D
A	0	1	2	2	0
B	1	0	0	0	2
C	2	0	0	1	0
D	2	0	1	0	1
E	0	2	0	1	0

# Parte 2.2: ¿Matrices? ¿Para qué?

**Espera pero, ¿qué es una matriz de adyacencia y para qué la quiero?**

1. Convertiremos el Dataframe a un array de NumPy.

[LISTO] Calcularemos el producto punto entre el vector de género y dos películas dado el género que comparten. Con esto tendremos una matriz simétrica donde cada elemento  $[i][j]$  representa la fuerza de conexión entre la película  $i$  y  $j$ .

2. Dejaremos la diagonal en ceros, para indicar que no pueden existir conexiones consigo mismas.

**¡A programar!**  

# Parte 3: Crear grafo y realizar la búsqueda

## Actualmente...

- Tenemos una clase Grafo que parece modelar correctamente nodos, aristas.
- Las dos funciones de búsqueda están implementadas.
- ¡Pero no sabemos cómo crear un grafo! Tenemos que completar **crear\_grafo**, para que pueda ser utilizado en la búsqueda y en la recomendación.

## Resultado esperado tras terminar

- El grafo podrá ser usado para la recomendación.

¡A programar!  

# Parte 4: TOP recomendados!

## Actualmente...

- Nos entrega las recomendaciones, pero no están ordenadas de ninguna forma.
- Es importante para el usuario que los TOP 5 tengan algún sentido.

## Resultado esperado tras terminar

- Las recomendaciones se visualizarán de forma ordenada, primero por número de géneros compartidos y luego por rating.
- Finalmente probaremos con otros TOP y con los dos métodos de búsqueda.

**¡A programar!**





# ***Programación Avanzada***

## **IIC2233 2024-2**

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Lucas Van Sint Jan - Francisca Cattán

