Problema 4 - Selección de equipo para olimpiada

Resumiendo el problema:

Hay un gryupo de estudiantes, cada uno con:

- UN promedio
- Una carrera **(unica)**

Y pues se desea formar un equipo para olimpiadas cumpliendo:

- Restriccion de que no se puede escoger a mas de un estudiante por carrera.
- Objetivo como maximizar la suma de promedios del equipo.

a. Demostrar que este problema se puede modelar como un matroide ponderado y proveer el algoritmo *greedy* que lo resuelve

Modelación como matroide

Definimos un matroide M = (E, I), donde:

- E: conjunto de todos los estudiantes.
- I: subconjuntos de estudiantes donde **no hay dos estudiantes de la misma carrera**.

Verificación de propiedades del matroide

1. **Conjunto vacío**: El conjunto vacío cumple la condición de no tener estudiantes de la misma carrera → $\emptyset \in I$.
2. **Herencia**: Si un conjunto de estudiantes no repite carreras, cualquier subconjunto tampoco lo hará.
3. **Propiedad de intercambio**: Si A, B $\in$ I y |A| < |B|, entonces hay un estudiante en (B − A) cuya carrera no está en A, y se puede agregar sin violar la independencia.

Se cumple que el sistema es un **matroide**, específicamente un **matroide de partición**, donde cada clase de la partición representa una carrera.

Matroide ponderado

Asignamos un **peso** a cada estudiante (su promedio). El objetivo es **maximizar la suma de promedios** del subconjunto seleccionado sin repetir carreras.

Algoritmo Greedy

1. Ordenar a los estudiantes en orden descendente por su promedio.
2. Inicializar el conjunto solución S ← ∅. Seleccionar a cada estudiante si su carrera aún no ha sido seleccionada.
3. Iterar sobre la lista ordenada:
    - Si el estudiante no pertenece a una carrera ya representada en S, agregarlo a S.

Este algoritmo **greedy** es óptimo porque trabaja sobre un matroide ponderado, lo que garantiza que seleccionar los elementos de mayor peso en orden válido produce la solución óptima.

Ejemplo de implementacion en python:

```
def seleccion_greedy(estudiantes):
    # estudiantes: lista de tuplas (nombre, carrera, promedio)
    estudiantes_ordenados = sorted(estudiantes, key=lambda x: x[2], reverse=True)
    equipo = []
    carreras_usadas = set()

    for nombre, carrera, promedio in estudiantes_ordenados:
        if carrera not in carreras_usadas:
            equipo.append((nombre, carrera, promedio))
            carreras_usadas.add(carrera)

    return equipo
```

Complejidad: O(nlogn) por la ordenación inicial.

---

b. Instancia del problema y aplicación del algoritmo

Lista de estudiantes

Nombre	Carrera	Promedio
Ana	Ing. Sistemas	90
Beto	Ing. Química	85
Carla	Ing. Sistemas	87
Diego	Ing. Civil	83
Elena	Ing. Química	92
Félix	Ing. Civil	78

Paso 1: Ordenar por promedio (descendente)

Nombre	Carrera	Promedio
Elena	Ing. Química	92
Ana	Ing. Sistemas	90
Carla	Ing. Sistemas	87
Beto	Ing. Química	85
Diego	Ing. Civil	83
Félix	Ing. Civil	78

Paso 2: Aplicación del algoritmo greedy

- Elena (Ing. Química): ✅
- Ana (Ing. Sistemas): ✅
- Carla: ❌ (ya hay alguien de Ing. Sistemas)
- Beto: ❌ (ya hay alguien de Ing. Química)
- Diego (Ing. Civil): ✅
- Félix: ❌ (ya hay alguien de Ing. Civil)

Resultado final

Nombre	Carrera	Promedio
Elena	Ing. Química	92
Ana	Ing. Sistemas	90
Diego	Ing. Civil	83

Suma total de promedios: 92 + 90 + 83 = **265**

Conclusión

- El problema se modela como un **matroide de partición ponderado**.
- El algoritmo *greedy* garantiza la solución óptima bajo estas condiciones.
- La estrategia es simple: **ordenar y seleccionar sin romper la independencia** (es decir, sin repetir carrera).