Álgebras de subconjuntos de un conjunto finito (proyecto o tarea adicional)

Es un proyecto de programación para estudiantes de cursos de análisis matemático. Al resolver los ejercicios de este proyecto, el estudiante entenderá mejor la definición de álgebras de conjuntos.

Objetivos. En algún lenguaje de programación programar un algoritmo que determine si una lista de conjuntos es un álgebra de conjuntos. En algún lenguaje de programación programar un algoritmo que encontre todas las álgebras de subconjuntos del conjunto dado de n elementos.

Requisitos. Programación de funciones con ciclos, listas y listas de listas. Otra opción: programación funcional.

Para resolver esta tarea, se recomienda usar un lenguaje de programación de nivel alto, que tenga una sintaxis natural para trabajar con listas (o arreglos) y con listas de listas (de longitud variable). Por ejemplo, Python y Haskell son excelentes opciones. A continuación, se la usa la sintaxis de Python y se supone que los índices de arreglos empiezan desde 0. Se recomienda escribir las soluciones de los siguientes ejercicios como apuntes en LaTeX.

Estudiar el concepto de álgebra de conjuntos

Si $\mathcal{F}$ es un conjunto de subconjuntos de X (es decir, $\mathcal{F}$ es un subconjunto del conjunto potencia $\mathcal{P}(X)$), entonces decimos que $\mathcal{F}$ es una *colección* sobre X. Esta terminología no es obligatoria, pero es cómoda en este tema.

1 Ejercicio (anillo de conjuntos)**.** Encontrar en libros de análisis real (Halmos, Rudin, Folland, etc.) la definición del anillo de conjuntos. Si varios autores dan definiciones diferentes, hay que explicar su equivalencia.

2 Ejercicio (álgebra de conjuntos)**.** Sean X un conjunto y $\mathcal{F}$ una colección sobre X. Se dice que $\mathcal{F}$ es un *álgebra* de conjuntos sobre X, si $\mathcal{F}$ es un anillo de conjuntos y $X \in \mathcal{F}$. Encontrar en libros la definición del álgebra de conjuntos y comparar con la definición dada aquí.

3 Ejercicio (criterio de álgebra de conjuntos)**.** Sean X un conjunto y $\mathcal{F}$ una colección sobre X. Demostrar que $\mathcal{F}$ es álgebra sobre X si, y solo si, se cumplen las siguientes tres condiciones:

1) $\varnothing \in \mathcal{F}$,

2) $\forall A \in \mathcal{F} \quad X \setminus A \in \mathcal{F}$,

3) $\forall A, B \in \mathcal{F} \quad A \cup B \in \mathcal{F}$.

En otras palabras, demostrar que $\mathcal{F}$ es un álgebra de conjuntos sobre X si, y solo si, el conjunto vacío es un elemento de $\mathcal{F}$, la colección $\mathcal{F}$ es cerrada bajo complementos (respecto a X) y $\mathcal{F}$ es cerrada bajo la operación unión (de dos conjuntos).

4 Ejercicio (σ-álgebras de conjuntos)**.** Encontrar en libros la definición de σ-álgebra de conjuntos. Explicar por qué cada σ-álgebra sobre X es un álgebra sobre X.

5 Ejercicio (para X finito, cada álgebra sobre X es σ-álgebra)**.** Sea X un conjunto finito y sea $\mathcal{F}$ un álgebra sobre X. Demostrar que $\mathcal{F}$ es una σ-álgebra sobre X.

6 Ejercicio (todas las álgebras sobre un conjunto de dos elementos)**.** Encontrar a mano todas las álgebras de conjuntos sobre $X = \{0, 1\}$.

Programar operaciones elementales con conjuntos

7 Ejercicio (conjuntos como listas, pertenencia de un elemento a una lista)**.** Hay varias maneras de representar conjuntos en programación. Suponemos que X es de la forma $\{0, 1, \ldots, n-1\}$ y que los subconjuntos de X se representan como listas de sus elementos, sin repeticiones. Por ejemplo, representamos el conjunto $A = \{0, 3, 4\}$ como la lista `[0, 3, 4]`. Se recomienda aceptar el convenio que los elementos se escriben en el orden ascedendente. Escribir una función que verifique si el elemento dado pertenece a la lista dada:

```
def belongs(x, A):
   ...
```

Por ejemplo, `belongs(3, [0, 3, 4])` debe regresar True. Puede ser que el lenguaje de programación elegido ya tiene una operación estándar que verifica la pertenencia de un elemento a una lista; en este caso no es obligatorio escribir su propia función (aunque es un buen ejercicio de programación).

8 Ejercicio (la igualdad de dos conjuntos-listas)**.** Escribir una función (o encontrar una función estándar en el lenguaje de programación elegido) que determine si dos listas dadas son iguales.

```
def are_equal(A, B):
   ...
```

Por ejemplo, `are_equal([0, 3, 4], [0, 3])` debe regresar `False`.

9 Ejercicio (la diferencia de dos conjuntos-listas)**.** Escribir una función de dos argumentos, A y B, que regrese la lista de todos los elementos de A que no están en B.

```
def set_diff(A, B):
   ...
```

Por ejemplo, `set_diff([0, 3, 4, 5], [1, 2, 3, 5])` debe devolver la lista `[0, 4]`.

10 Ejercicio (el complemento de un conjunto-lista)**.** Escribir una función de dos argumentos, n y A, que devuelva la lista de todos los elementos de $\{0, \ldots, n-1\}$ que no están en A.

```
def set_complement(n, A):
    ...
```

Por ejemplo, `set_complement(5, [0, 3])` debe devolver `[1, 2, 4]`.

11 Ejercicio (la unión de dos conjuntos-listas)**.** Escribir una función de dos argumentos, A y B, que devuelva la lista de todos los elementos que están en A o están en B.

```
def set_union(A, B):
    ...
```

Por ejemplo, `set_union([0, 3, 5, 6], [2, 5])` debe devolver `[0, 2, 3, 5, 6]`.

12 Ejercicio. Como ejercicios adicionales, se pueden programar también la intersección y la diferencia simétrica.

Verificar si una colección dada es un álgebra

Representamos colecciones sobre X como listas de listas. Por ejemplo, representamos la colección

$$\{\varnothing, \{0, 2\}, \{1, 2, 5\}\}$$

como la lista `[[], [0, 2], [1, 2, 5]]`.

13 Ejercicio (pertenencia de un conjunto a una colección)**.** Programar una función de dos argumentos (una lista `A` y una lista de listas `F`) que verifique si `A` es un elemento de `F`.

```
def set_belongs_to_collection(A, F):
    ...
```

Puede ser que el lenguaje de programación elegido ya tiene una operación estándar que sustituye esta función. En este caso, este ejercicio es opcional.

14 Ejercicio (la colección dada es cerrada bajo complementos)**.** Escribir una función de dos argumentos, n y $\mathcal{F}$, que verifique si la colección $\mathcal{F}$ es cerrada bajo los complementos respecto a $\{0, 1, \ldots, n-1\}$.

```
def collection_is_closed_under_complement(n, F):
    ...
```

Por ejemplo, para $n = 4$ y $\mathcal{F} = \{\varnothing, \{1, 3\}, \{0, 1, 2, 3\}\}$ la función debe devolver `False`.

15 Ejercicio (la colección dada es cerrada bajo uniones de dos conjuntos)**.** Dada una colección $\mathcal{F}$, verificar si se cumple la condición 2) del Ejercicio 3.

```
def collection_is_closed_under_union(F):
    ...
```

Por ejemplo, para

$$\mathcal{F} = \{\varnothing, \{1,3\}, \{2,3\}, \{0\}, \{0,1,2,3\}\}$$

la función debe devolver `False`.

16 Ejercicio (la colección dada es álgebra)**.** Programar una función que verifique si la colección dada es un álgebra sobre $\{0, \ldots, n-1\}$.

```
def collection_is_algebra(n, F):
    ...
```

Por ejemplo, para $n = 3$ y

$$\mathcal{F} = \{\varnothing, \{0,2\}, \{1\}, \{0,1,2\}\}$$

la función debe devolver `True`.

17 Ejercicio. Hacer varias pruebas de la función del ejercicio anterior. Considerar al menos 5 ejemplos con $n = 4$ y al menos 5 ejemplos con $n = 5$.

Generar todas las álgebras sobre un conjunto finito

Esta parte de la tarea es más complicada y no es obligatoria. Representamos conjuntos de colecciones como listas de listas de listas de números enteros.

18 Ejercicio (¿cuántas colecciones hay sobre un conjunto finito?)**.** Para un conjunto finito $X = \{0, 1, \ldots, n-1\}$, consideramos el conjunto de todas las colecciones sobre X, es decir, el conjunto $\mathcal{P}(\mathcal{P}(X))$. Por ejemplo, para $n = 1$ el conjunto $\mathcal{P}(\mathcal{P}(X))$ tiene 4 elementos:

$$\mathcal{P}(\mathcal{P}(\{0\})) = \{\{\}, \{\varnothing\}, \{\{0\}\}, \{\varnothing, \{0\}\}\}.$$

Para un n natural y $X = \{0, 1, \ldots, n-1\}$, ¿cuántos elementos tiene $\mathcal{P}(\mathcal{P}(X))$?

19 Ejercicio (generar todas las colecciones sobre un conjunto finito)**.** Dado un n natural, construir y devolver la lista de todas las colecciones sobre $\{0, \ldots, n-1\}$.

```
def all_collections(n):
    ...
```

Por ejemplo, para $n = 1$ (es decir, cuando $X = \{0\}$), la función debe devolver una lista de 4 listas de listas:

```
[
  [],
  [[]],
  [[0]],
  [[], [0]]
]
```

Se recomienda ejecutar esta función con $n = 1$, $n = 2$ y $n = 3$. Verificar si las respuestas tienen longitud correcta.

20 Ejercicio (encontrar todas las álgebras sobre un conjunto finito)**.** Escribir una función que construya todas las álgebras de conjuntos sobre $\{0, \ldots, n-1\}$, para n dado.

```
def all_algebras(n):
   ...
```

Para $n = 2$, comparar el resultado con la respuesta del Ejercicio 6.

21 Ejercicio (buscar métodos más eficientes para generar todas las álgebras de conjuntos)**.** Como una tarea opcional aún más complicada, puede buscar métodos más eficientes para generar todas las álgebras de conjuntos sobre $\{0, \ldots, n-1\}$, evitando la búsqueda sobre todas las colecciones. Ideas claves: *particiones* de X, *átomos* de un álgebra. Puede revisar otra tarea adicional, sobre la estructura de σ-álgebras finitas o numerables.