CLASE 8

1. Conceptos Fundamentales de los TADs

Un Tipo Abstracto de Dato (TAD) es una estructura de datos que se define por su **interfaz** (qué operaciones se pueden hacer) y no por su implementación interna (cómo se hacen). Esto crea distintos roles con diferentes responsabilidades:

- **Diseñador:** Define la interfaz y la eficiencia esperada.
- **Usuario:** Utiliza las operaciones del TAD a través de su interfaz, sin conocer los detalles internos.
- **Implementador:** Elige las estructuras de datos internas para cumplir con la interfaz y la eficiencia prometida, manteniendo los **invariantes de representación**.

2. Análisis de Eficiencia (Complejidad)

La eficiencia mide el costo de las operaciones, generalmente en el **peor caso** y en función del número de elementos n. Las clasificaciones de eficiencia más comunes son:

- **Constante O(1):** El costo es siempre el mismo, sin importar el tamaño.
- **Logarítmico O(logn):** Muy eficiente, común en búsquedas en árboles.
- **Lineal O(n):** El costo crece proporcionalmente con el tamaño.
- **N log N O(nlogn):** Común en algoritmos de ordenamiento eficientes.
- **Cuadrática O(n2):** El costo crece con el cuadrado del tamaño; puede ser lento para grandes cantidades de datos.

Caso Práctico: TAD "Escuela de Magia"

Este es el ejemplo central de la presentación y es crucial para entender la aplicación de los conceptos.

A. La Interfaz (Lo que el Usuario ve)

El TAD EscuelaDeMagia permite gestionar magos y hechizos con operaciones como:

- fundarEscuela: O(1)
- registrar: O(logM)
- magos: O(M)
- hechizosDe: O(logM)
- egresarUno: O(logM)
- enseñar: O(MlogM+logH)

Punto clave: La eficiencia aquí depende de dos variables: M (cantidad de magos) y H (cantidad total de hechizos).

B. La Implementación (Lo que el Implementador hace)

Para lograr la eficiencia deseada, la EscuelaDeMagia se implementa internamente usando otros TADs:

```
data EscuelaDeMagia = EDM (Set Hechizo) (Map Nombre Mago) (PriorityQueue Mago)
```

- **Set Hechizo**: Almacena todos los hechizos enseñados en la escuela.
- **Map Nombre Mago**: Permite buscar un mago por su nombre de forma muy rápida (O(logM)).
- **PriorityQueue Mago**: Ordena a los magos por su "poder" (cantidad de hechizos), permitiendo obtener al más poderoso rápidamente.

C. ¡Punto Clave! Invariantes de Representación (INV. REP.)

Para que el TAD funcione correctamente, el implementador debe garantizar que estas condiciones **siempre** se cumplan:

1. **Consistencia entre Map y PriorityQueue**: Todos los magos que están en el Map deben estar también en la PriorityQueue, y viceversa. Esto es fundamental para mantener los datos sincronizados.
2. **Consistencia de Hechizos**: Todos los hechizos que sabe un mago deben estar registrados en el Set general de la escuela.
3. **Unicidad en la PriorityQueue**: No puede haber dos magos con el mismo nombre en la PriorityQueue.

Ejemplo de mantenimiento de invariantes: Al registrar un mago, se lo debe agregar tanto al Map como a la PriorityQueue para cumplir el invariante 1. Al egresarUno, se lo debe eliminar de ambas estructuras.

D. Uso del TAD para crear nuevas funciones

Como **usuario**, puedes crear funciones complejas combinando las operaciones de la interfaz. Por ejemplo:

- **hayUnExperto**: Para saber si hay un mago que sabe todos los hechizos, simplemente se obtiene al mago con más hechizos usando egresarUno (que aprovecha la PriorityQueue interna) y luego se verifica si le faltan hechizos por aprender con leFaltaAprender. La eficiencia es O(logM) porque las operaciones base lo son.
- **egresarExpertos**: Esta función se implementa de forma recursiva, llamando a hayUnExperto y egresarUno repetidamente hasta que no queden más expertos en la escuela. Su costo es O(MlogM).

Resumen Final y Puntos Clave para el Examen

1. **Distinguir Roles**: Ten muy claro qué información tiene el **usuario** (la interfaz y su eficiencia) y qué maneja el **implementador** (la estructura interna y los invariantes).
2. **Los Invariantes son la Clave**: Son las reglas que el implementador debe mantener para que el TAD sea consistente y eficiente. Son la parte más importante de la implementación.
3. **Construcción sobre TADs**: Es muy común que un TAD se implemente utilizando otros TADs más básicos (como Set, Map, Queue, etc.).
4. **Eficiencia con Múltiples Variables**: La eficiencia no siempre depende de un solo n. En este caso, depende de M (magos) y H (hechizos), y debes saber cómo se combinan en cada operación.