Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignaturas: Estructuras de Datos y Algoritmos - Algoritmos I
Segundo Cuatrimestre de 2023

Trabajo Práctico No 1

La resolución de este trabajo práctico debe ser enviada a través de GitHub Classroom.
El código provisto como parte de la solución a los ejercicios deberá estar documentado apropiadamente (por ejemplo, con comentarios en el código). Aquellas soluciones que no requieran programación, como así también la documentación adicional de código que se desee proveer, debe entregarse en archivos de **texto convencionales**, en el mismo repositorio de entrega de las soluciones, con nombres que permitan identificar fácilmente su contenido.
Los grupos para la resolución del trabajo deben ser conformados por tres integrantes.
La fecha límite para la entrega de la solución del TP es **10** días después de la fecha de entrega del enunciado.

En matemáticas un *Bag* (multiconjunto), es una modificación del concepto de conjunto, ya que permite múltiples instancias para cada uno de sus elementos, es decir, cada miembro del mismo tiene asociada una multiplicidad (un número natural), indicando cuántas veces el elemento es miembro del conjunto. Por ejemplo, en el multiconjunto $\{a, a, b, b, b, c\}$, las multiplicidades de los miembros a, b, y c son $2, 3$, y 1, respectivamente.
El álgebra de Bag genérico se especifica de la siguiente manera:

$$\langle \{T\}_{bag}, \emptyset, ins, \uplus \rangle.$$

Este TAD, se puede puede implementar utilizando listas de pares.

En este trabajo se pide completar una implementación parcial del álgebra de Bag, que utiliza ArrayList de pares, en el lenguaje Java. Deberá completar la implementación de `MultiSetArray` con las siguientes operaciones y, en algunos casos, con las restricciones de complejidad en tiempo especificadas:

- `isEmpty`: chequea si el bag no tiene elementos.
- `insert`: agrega el elemento especificado al bag.
- `contains`: chequea si el elemento especificado pertenece al bag. La complejidad de este método, en el peor caso, debe ser $O(\log n)$.
- `remove`: quita del bag el elemento especificado, una vez.
- `union`: retorna la unión de los elementos del bag, con los elementos de otro bag que toma como parámetro. Un ejemplo de unión de bags sería:
  $\{a, a, b, b, b, c\} \uplus \{a, a, b, f, f\} = \{a, a, a, a, b, b, b, b, c, f, f\}$
- `max`: retorna el máximo elemento del bag. La complejidad de este método, en el peor caso, debe ser $O(1)$.

Además la implementación de MultiSetArray, tiene que preservar el invariante de representación, especificado a través de las siguientes propiedades:

- $items \neq null$

- $(\forall i : 0 \leq i < items.length - 1 : items[i].first < items[i+1].first)$
- $(\forall i : 0 \leq i < items.length : items[i].second > 0)$

Donde *items* es la lista de pares que representa un Bag.

Para probar la implementación del TAD **MultiSet**, se debe proveer una clase Main con operaciones que implementen los siguientes ítems:

- Dado el nombre de un archivo de texto, deberá crear un bag con las palabras contenidas en el archivo especificado.
- Una vez cargado el bag con palabras, deberá proveer un mecanismo que permita saber si, dada una palabra, pertenece o no al texto contenido en el archivo.