Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I
Evaluación continua - 2024/2025

18 de octubre de 2024

Nombre:

La duración de esta prueba es de media hora. Consta de un solo ejercicio. La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Al finalizar entrega tu solución junto con el enunciado (no es necesario que entregues todas las hojas). Pon tu nombre en cada hoja que entregues.

Implementa en lenguaje C++ el método que se pide. No puedes utilizar ninguna clase de las bibliotecas de C++ (`queue`, `vector`, etc.). Si resuelves el ejercicio empleando algún bucle, optarás al 50 % de la nota.

Ejercicio — 1 punto

Estamos utilizando una lista enlazada para realizar una implementación del Tipo Abstracto de Datos **Conjunto** a la que se han añadido algunas de las operaciones propias de las colas de prioridad, concretamente las que se muestran a continuación. Para ello, tenemos los siguientes atributos, y no puedes añadir otros atributos en `Conjunto` ni en `Conjunto::Nodo` (los puntos suspensivos corresponden a posibles declaraciones de métodos privados):

```
class Conjunto {
   struct Nodo {
      float dato;
      Nodo * anterior;
      Nodo * siguiente;
      ...
   };
   Nodo * minimo;
   Nodo * maximo;
   ...
 public:
   Conjunto();                      // O(1)
   void  insertar(float);           // O(n)
   bool  buscar(float) const;       // O(n)
   void  eliminar(float);           // O(n)
   float consultarMinimo() const;   // O(1)
   float consultarMaximo() const;   // O(1)
   void  eliminarMinimo();          // O(1)
   void  eliminarMaximo();          // O(1)
};
```

Los datos se guardan de modo tal que el coste temporal en el peor caso de cada una de esas operaciones es el que se indica, siendo n la talla del conjunto. No se guardan datos repetidos. Piensa cómo conseguir que esas operaciones tengan esos costes y tenlo en cuenta para resolver lo que se pide a continuación.

Añade a la clase `Conjunto` un método `void eliminarComunes(const Conjunto &)`, de modo tal que `c1.eliminarComunes(c2)` elimine de `c1` todos los datos que aparecen también en `c2`. Hazlo utilizando recursión, sin utilizar ningún bucle. Si haces uso de otras funciones, debes implementarlas también sin utilizar ningún bucle. El conjunto `c2` no debe modificarse. Si no hay ningún dato de `c2` en `c1`, tampoco se modificará `c1`. Como casos particulares, tanto `c1` como `c2`, inicialmente y/o al finalizar, pueden ser conjuntos vacíos. Tu solución debe funcionar en todos los casos. No debes lanzar ninguna excepción.

Para que tu solución sea válida, debe ser eficiente. Indica cuáles son los siguientes costes de tu solución en función de a y b, siendo a la talla del primer conjunto y b la talla del segundo. No es necesario que lo justifiques.

Coste temporal en el peor caso	$O($ $)$
Coste espacial en el peor caso sin contar las dos listas enlazadas	$O($ $)$