Algoritmos y Estructuras de Datos (VJ1215) - Universitat Jaume I

Examen final - 2024/2025 - Primera parte
Recuperación del primer examen parcial

21 de enero de 2025

Nombre:

El examen final consta de dos partes, cada una con una duración de dos horas. La primera parte está destinada a aquellos estudiantes que han solicitado intentar mejorar la calificación obtenida en cualquiera de los exámenes parciales, renunciando a la misma. Este primer ejercicio de la primera parte está dirigido a quienes buscan mejorar la nota del primer examen parcial, y tiene una duración de 30 minutos.

La prueba es individual. No puedes consultar libros, apuntes ni dispositivos electrónicos. Escribe tu solución empleando el lenguaje C++, y no olvides los costes que se piden. 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.

Ejercicio 1 — 1 punto

Estamos utilizando una lista simplemente enlazada para realizar una implementación del Tipo Abstracto de Datos **Conjunto** a la que se ha añadido la posibilidad de consultar y eliminar el mínimo eficientemente. 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):

```
class Conjunto {
   struct Nodo {
      float dato;
      Nodo * siguiente;
      ...
   };
   Nodo * primero;
   ...
 public:
   ...
};
```

Los datos se guardan de modo tal que el coste temporal en el peor caso de la operación `consultarMinimo` es $O(1)$, el de `eliminarMinimo` es $O(1)$, el de `insertar` es $O(n)$, el de `eliminar` es $O(n)$ y el de `buscar` es $O(n)$, 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 alternar(float dato)
```

que haga lo siguiente: si el dato ya estaba en el conjunto, lo elimina del mismo; si no estaba, lo inserta.

Hazlo utilizando recursión, evitando por completo el uso de bucles. No es necesario que implementes los constructores de `Conjunto` y de `Conjunto::Nodo` necesarios para que tu solución funcione correctamente. Si utilizas otras funciones, debes implementarlas también, y debes hacerlo sin utilizar ningún bucle.

Tu solución debe ser eficiente. Además, se valorará que no siga recorriendo la lista enlazada innecesariamente. Indica cuáles son los siguientes costes de tu solución en función de n. No es necesario justificar las respuestas. En el cálculo del coste espacial, no consideres el espacio ocupado por la propia lista enlazada.

Coste temporal en el peor caso	$O($ $)$
Coste espacial en el peor caso sin contar la lista enlazada	$O($ $)$