Estructura de Datos

Grupos B, D e I

Convocatoria ordinaria 2020-2021

Ejercicio 1 [2,5 puntos]

Se desea añadir la operación `minimoACima` a la implementación del TAD `Pila` basada en nodos enlazados vista en clase, que lleve el elemento más pequeño de la pila a la cima y respete la disposición relativa de los demás elementos. Por ejemplo, dada una pila de enteros positivos como esta, donde `21` es la cima de la pila,

21
32
3
7
15

el resultado de aplicar sobre ella la operación `minimoACima` la convertirá en esta otra:

3
21
32
7
15

En caso de que haya varias apariciones del elemento mínimo en la pila se llevará a la cima la aparición que esté más cerca de la misma. Es decir, si partimos de una pila como esta

21
32
3
56
3

tras aplicar la operación `minimoACima` la pila se convertirá en

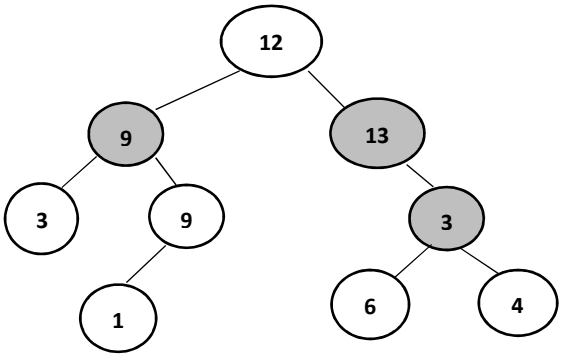
3
21
32
56
3

Si la pila está vacía o el elemento mínimo ya está en la cima, la operación no surtirá ningún efecto.

En la implementación de esta nueva operación no se permiten utilizar, ni directa, ni indirectamente, operaciones de manejo de memoria dinámica (`new`, `delete`), ni tampoco realizar asignación alguna entre contenidos de los nodos. Aparte de implementar la operación, debes determinar justificadamente su complejidad.

Ejercicio 2 [2,5 puntos]
Un nodo de un árbol binario de enteros es *correctivo* cuando tiene hijo derecho, y, además, su valor es igual a la diferencia entre la suma de los valores de su hijo derecho menos el número de nodos que hay en su hijo izquierdo.

A modo de ejemplo, en el siguiente árbol aparecen sombreados los nodos correctivos:



Desarrolla un algoritmo eficiente que cuente el número de nodos correctivos que contiene un árbol binario de enteros dado como entrada. Determina, asimismo, justificadamente su complejidad.

Ejercicio 3 [5 puntos]
En una empresa de transporte urgente de paquetería a cada paquete recepcionado se le asocia un identificador y se registran la dirección en la que debe ser entregado y su peso. Los paquetes esperan en fila a ser cargados en camiones y entregarse en la dirección correspondiente, según el orden en que se van recepcionando. Asimismo, cualquier destinatario de un paquete que sea un poco impaciente puede acudir a la empresa con el identificador del paquete y llevárselo sin esperar a que sea cargado en un camión. Tanto si un paquete es cargado en un camión como si es retirado por su destinatario, sus datos se eliminan del sistema.

La empresa cuenta además con una flota de camiones de los cuales se conoce el peso que pueden cargar. Los camiones esperan en fila a ser cargados, según el orden en el que van llegando al almacén de la empresa.

Nos han pedido implementar un sistema para la gestión de la recepción y el reparto de paquetes. Para ello desarrollaremos un TAD `Paqueteria` con un conjunto de operaciones que son las siguientes:

- `Paqueteria()`: operación constructora que crea un valor del TAD vacío.
- `recepciona_paquete (id, dir, peso)`: añade al sistema un nuevo paquete con identificador `id`, dirección de entrega `dir` y peso `peso`. En caso de que el identificador esté duplicado, la operación eleva la excepción `EPaqueteDuplicado`.
- `info_paquete(id, dir, peso)`: devuelve la dirección de entrega `dir` y el peso `peso` de un paquete con identificador `id`. En caso de que el identificador no exista en el sistema, la operación eleva la excepción `EPaqueteInexistente`.
- `hay_paquetes()`: devuelve `true` si hay paquetes esperando en la fila, y `false` en otro caso.
- `primero_en_fila(id)`: devuelve el identificador `id` del primer paquete que espera en la fila para ser cargado. Si no hay ningún paquete esperando en la fila, la operación eleva la excepción `ENingunPaqueteEnEspera`.

- `elimina(id)`: desaparece del sistema toda la información del paquete con identificador `id`. Si no hay ningún paquete con ese identificador en el sistema, la operación no realiza ningún efecto.
- `nuevo_camion(peso)`: incorpora por el final de la fila de camiones que hay en el almacén un nuevo camión capaz de cargar un peso `peso`.
- `primer_camion(peso)`: devuelve el peso `peso` que es capaz de cargar el primer camión de la fila de camiones. Si no hay ningún camión esperando a ser cargado, la operación eleva la excepción `ENingunCamionEnEspera`.
- `carga_camion(listaPaq)`: carga el primer camión de la fila de camiones con el mayor número de paquetes en espera posible, tal que el peso total de dichos paquetes no supere el peso que puede cargar el camión. La carga de los paquetes se realiza respetando el orden de espera. Se devuelve una lista con los identificadores de los paquetes cargados (el primero que esperaba será el primer elemento de la lista devuelta, el segundo que esperaba será el segundo en la lista devuelta, y así hasta el último paquete que se cargue), y tanto la información del camión como la de los paquetes cargados en él desparecen del sistema. Obsérvese que, en el caso de que el peso del primer paquete exceda el que admite el camión, se devuelve una lista vacía, y solo la información del camión desaparece del sistema. Si no hay camiones esperando a ser cargados, o no hay paquetes esperando a ser repartidos se eleva la excepción `EErrorCarga`.

Debes elegir una representación adecuada para el TAD que permita obtener una implementación de las operaciones lo más eficiente posible. Así mismo debes indicar y justificar la complejidad resultante de cada operación.