Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen 2º Parcial, convocatoria extraordinaria
6 de septiembre de 2018

1. (2 puntos) Extiende la implementación basada en nodos doblemente enlazados del TAD `Lista` con la siguiente operación:

```
void invertir()
```

Dicha operación debe invertir los elementos de la lista. De esta forma, si la lista contiene los siguientes elementos (de principio a fin):

1 2 3 4 5 6 7

tras ejecutar dicha operación los elementos de la lista, de principio a fin, serán:

7 6 5 4 3 2 1

En la implementación no deben crearse ni destruirse nodos, ni tampoco realizarse asignaciones entre los contenidos de los nodos. Aparte de implementar la operación, debes indicar la complejidad de la misma.

2. (2 puntos) Un nodo en un árbol binario de enteros se dice que es *curioso* cuando su valor coincide con el resultado de sumar su nivel al número de nodos de su hijo izquierdo. Se debe programar una función

```
int num_curiosos(const Arbin<int>& a)
```

que devuelva el número de nodos *curiosos* que contiene el árbol `a` dado como parámetro. Debe, además, determinarse justificadamente la complejidad de dicha función.

3. (6 puntos) Nos han encargado implementar un sistema para la gestión de las listas de espera en un sistema de venta online de entradas de conciertos.

Cuando un cliente se registra en el sistema, se le asigna un código de identificación único (cadena de caracteres). El sistema permite además dar de alta conciertos identificados también mediante un código único (cadena de caracteres), así como gestionar las listas de espera de clientes para comprar las entradas de los conciertos dados de alta.

Para llevar a cabo la implementación de este sistema hemos decidido desarrollar un TAD `SistemaVentas` con las siguientes operaciones:

- `crea()`: Operación constructora que crea un sistema de gestión de venta de entradas vacío.
- `an_concierto(codigo_concierto)`: Añade un nuevo concierto al sistema, con código de identificación `código_concierto`. Si ya está dado de alta un concierto con dicho código, la operación lanzará una excepción `EConciertoExistente`.
- `an_cliente(codigo_cliente, codigo_concierto)`: Añade un nuevo cliente al sistema con código de identificación `codigo_cliente`,y lo pone a la espera de compra de entradas para el concierto con código de indentificación `codigo_concierto`. En caso de que no exista el cliente, o no exista el concierto, la operación lanzará una excepción `EAltaNoAdmitida`.

- `borra_concierto(codigo_concierto)`: Elimina el concierto con código de identificación `codigo_concierto` del sistema. Para ello dicho concierto debe existir; si no es así, la operación lanzará una excepción `EConciertoInexistente`. Así mismo, su lista de espera debe estar vacía; en otro caso, la operación lanzará una excepción `EConciertoConEsperas`.
- `borra_cliente(codigo_cliente)`: Elimina todo rastro del cliente con código de identificación `codigo_cliente` del sistema. Si el cliente no existe, la operación lanzará una excepción `EClienteInexistente`.
- `hay_clientes_en_espera(codigo_concierto)->boolean`: Devuelve cierto si hay clientes a la espera de comprar entradas para el concierto `codigo_concierto`, y falso en otro caso. El código del concierto debe existir; si no, la operación lanzará un excepción `EConciertoInexistente`.
- `proximo_cliente(codigo_concierto)->codigo_cliente`: Devuelve el `codigo_cliente` del primer cliente en la lista de espera para el concierto `codigo_concierto`. Si no existe un concierto con el código dado, la operación lanzará una excepción `EConciertoInexistente`. Si el concierto existe pero no tiene lista de espera, la operación lanzará una excepción `EConciertoSinEsperas`.
- `venta(codigo_concierto)`: Realiza la venta de entrada del concierto `codigo_concierto`. Para ello, elimina el primer cliente de la lista de espera, y dicho cliente queda en disposición de realizar nuevas compras. Lanzará la excepción `EConciertoInexistente` en caso de que el concierto con codigo `codigo_concierto` no exista. Lanzará la excepción `EConciertoSinEsperas` en caso de que el concierto exista pero no tenga lista de espera.
- `abandona(codigo_cliente)`: Registra el abandono, por parte del cliente `codigo_cliente`, de la lista donde se encuentra esperando. Como consecuencia, el cliente se elimina de dicha lista de espera y el cliente pasa a estar en diposición de realizar nuevas compras. Se lanzará la excepcion `EClienteInexistente` en caso de que el cliente no exista. Si el cliente existe pero no se encuentra esperando en ninguna lista de espera, la operación no tendrá efecto.
- `pon_en_espera(codigo_cliente,codigo_concierto)`: Pone al cliente `codigo_cliente` en espera para comprar una entrada en el concierto `codigo_concierto`. El sistema admite únicamente que un cliente esté esperando para comprar entradas en, a lo sumo, un concierto. La operación elevará una excepción `EEsperaNoAdmitida` si no existe un cliente con código `codigo_cliente`, si no existe un concierto con código `codigo_concierto` o si el cliente está ya en una lista de espera.
- `clientes()->lista`: Devuelve una lista, ordenada alfabéticamente, con los códigos de los clientes registrados en el sistema.
- `num_clientes()->entero`: Devuelve el número de clientes registrados en el sistema.
- `num_conciertos()->entero`: Devuelve el número de conciertos registrados en el sistema.

Dado que éste es un sistema crítico, la implementación de las operaciones debe ser lo más eficiente posible. Por tanto, debes elegir una representación adecuada para el TAD, implementar las operaciones y justificar la complejidad de cada una de ellas.