Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Final, convocatoria extraordinaria
6 de septiembre de 2018

1. (3 puntos) Se dice que un vector forma una "quasi-secuencia de Fibonacci" si todas las posiciones cumplen que `v[i] = v[i-1] + v[i-2]` o `v[i] = v[i-1]`.

Especifica, diseña e implementa un algoritmo iterativo que procese un vector de tamaño n ($2 \leq n \leq 1000$) y compruebe si forma una quasi-secuencia de Fibonacci en la que la suma de todos los elementos no supera un cierto umbral dado $k \geq 0$.

Entrada							**Salida**
n	k	v					
5	5	1	1	1	1	1	SI
5	4	1	1	1	1	1	NO
5	5	1	1	1	2	3	NO
5	5	0	0	0	0	0	SI
5	5	0	0	0	0	1	NO
0							

2. (2 puntos) Implementa una función que reciba tres enteros, *ini*, n ($2 \leq n \leq 100$) y k ($k \geq 0$) que genere todos los vectores de tamaño n que formen una quasi-secuencia de Fibonacci en los que `v[0]=ini` y cuya suma total no supere k. Es decir, todos los vectores con `v[0]=ini` para los que la función del ejercicio anterior devuelva `SI` con el n y k dados.

Importante: No debes utilizar la función desarrollada en el ejercicio anterior

Entrada			**Salida**
ini	n	k	
1	5	4	
1	5	5	1 1 1 1 1
1	5	6	1 1 1 1 1
			1 1 1 1 2
1	5	7	1 1 1 1 1
			1 1 1 1 2
			1 1 1 2 2
0	5	10	0 0 0 0 0
0	0	0	

3. (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.

4. (3 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_cliente(codigo_cliente)`: Añade un nuevo cliente al sistema, con código de identificación `codigo_cliente`. Si ya está dado de alta en el sistema un cliente con dicho código, la operación lanzará una excepción `EClienteExistente`.
- `an_concierto(codigo_concierto)`: Añade un nuevo concierto al sistema, con código de identificación `codigo_concierto`. Si ya está dado de alta un concierto con dicho código, la operación lanzará una excepción `EConciertoExistente`.
- `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.
- `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 cocierto `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.

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.